# Handbook
# Statistical foundations of machine learning

Gianluca Bontempi

Machine Learning Group
Computer Science Department
Universite Libre de Bruxelles, ULB
Belgique

# Contents

**Index** **2**

**1 Introduction** **9**
1.1 Notations . . . 15

**2 Foundations of probability** **19**
2.1 The random model of uncertainty . . . 19
2.1.1 Axiomatic definition of probability . . . 21
2.1.2 Symmetrical definition of probability . . . 21
2.1.3 Frequentist definition of probability . . . 22
2.1.4 The Law of Large Numbers . . . 22
2.1.5 Independence and conditional probability . . . 24
2.1.6 Combined experiments . . . 25
2.1.7 The law of total probability and the Bayes' theorem . . . 27
2.1.8 Array of joint/marginal probabilities . . . 27
2.2 Random variables . . . 29
2.3 Discrete random variables . . . 30
2.3.1 Parametric probability function . . . 31
2.3.2 Expected value, variance and standard deviation of a discrete r.v. . . . 31
2.3.3 Moments of a discrete r.v. . . . 33
2.3.4 Entropy and relative entropy . . . 33
2.4 Continuous random variable . . . 34
2.4.1 Mean, variance, moments of a continuous r.v. . . . 35
2.5 Joint probability . . . 35
2.5.1 Marginal and conditional probability . . . 37
2.5.2 Entropy in the continuous case . . . 38
2.6 Common discrete probability functions . . . 39
2.6.1 The Bernoulli trial . . . 39
2.6.2 The Binomial probability function . . . 39
2.6.3 The Geometric probability function . . . 39
2.6.4 The Poisson probability function . . . 40
2.7 Common continuous distributions . . . 41
2.7.1 Uniform distribution . . . 41
2.7.2 Exponential distribution . . . 41
2.7.3 The Gamma distribution . . . 41
2.7.4 Normal distribution: the scalar case . . . 42
2.7.5 The chi-squared distribution . . . 43
2.7.6 Student's $t$-distribution . . . 44
2.7.7 F-distribution . . . 44
2.7.8 Bivariate continuous distribution . . . 45
2.7.9 Correlation . . . 46

2.7.10 Mutual information . . . . . 47
2.8 Normal distribution: the multivariate case . . . . . 48
2.8.1 Bivariate normal distribution . . . . . 49
2.9 Linear combinations of r.v. . . . . 51
2.9.1 The sum of i.i.d. random variables . . . . . 51
2.10 Transformation of random variables . . . . . 52
2.11 The central limit theorem . . . . . 52
2.12 The Chebyshev's inequality . . . . . 53

**3 Classical parametric estimation** **55**
3.1 Classical approach . . . . . 55
3.1.1 Point estimation . . . . . 57
3.2 Empirical distributions . . . . . 57
3.3 Plug-in principle to define an estimator . . . . . 58
3.3.1 Sample average . . . . . 59
3.3.2 Sample variance . . . . . 59
3.4 Sampling distribution . . . . . 59
3.5 The assessment of an estimator . . . . . 61
3.5.1 Bias and variance . . . . . 61
3.5.2 Bias and variance of $\hat{\boldsymbol{\mu}}$ . . . . . 62
3.5.3 Bias of the estimator $\hat{\boldsymbol{\sigma}}^2$ . . . . . 63
3.5.4 Bias/variance decomposition of MSE . . . . . 65
3.5.5 Consistency . . . . . 65
3.5.6 Efficiency . . . . . 66
3.5.7 Sufficiency . . . . . 66
3.6 The Hoeffding's inequality . . . . . 67
3.7 Sampling distributions for Gaussian r.v.s . . . . . 67
3.8 The principle of maximum likelihood . . . . . 68
3.8.1 Maximum likelihood computation . . . . . 69
3.8.2 Properties of m.l. estimators . . . . . 72
3.8.3 Cramer-Rao lower bound . . . . . 72
3.9 Interval estimation . . . . . 73
3.9.1 Confidence interval of $\mu$ . . . . . 73
3.10 Combination of two estimators . . . . . 76
3.10.1 Combination of $m$ estimators . . . . . 77
3.11 Testing hypothesis . . . . . 78
3.11.1 Types of hypothesis . . . . . 78
3.11.2 Types of statistical test . . . . . 78
3.11.3 Pure significance test . . . . . 79
3.11.4 Tests of significance . . . . . 79
3.11.5 Hypothesis testing . . . . . 81
3.11.6 Choice of test . . . . . 82
3.11.7 UMP level-$\alpha$ test . . . . . 83
3.11.8 Likelihood ratio test . . . . . 84
3.12 Parametric tests . . . . . 84
3.12.1 z-test (single and one-sided) . . . . . 85
3.12.2 t-test: single sample and two-sided . . . . . 86
3.12.3 $\chi^2$-test: single sample and two-sided . . . . . 87
3.12.4 t-test: two samples, two sided . . . . . 87
3.12.5 F-test: two samples, two sided . . . . . 87
3.13 A posteriori assessment of a test . . . . . 88
3.13.1 Receiver Operating Characteristic curve . . . . . 89

**4 Nonparametric estimation and testing** **91**
4.1 Nonparametric methods . . . 91
4.2 Estimation of arbitrary statistics . . . 92
4.3 Jacknife . . . 93
4.3.1 Jacknife estimation . . . 93
4.4 Bootstrap . . . 95
4.4.1 Bootstrap sampling . . . 95
4.4.2 Bootstrap estimate of the variance . . . 95
4.4.3 Bootstrap estimate of bias . . . 96
4.5 Bootstrap confidence interval . . . 97
4.5.1 The bootstrap principle . . . 98
4.6 Randomization tests . . . 99
4.6.1 Randomization and bootstrap . . . 101
4.7 Permutation test . . . 101
4.8 Considerations on nonparametric tests . . . 102

**5 Statistical supervised learning** **105**
5.1 Introduction . . . 105
5.2 Estimating dependencies . . . 108
5.3 The problem of classification . . . 110
5.3.1 Inverse conditional distribution . . . 112
5.4 The problem of regression estimation . . . 114
5.4.1 An illustrative example . . . 114
5.5 Generalization error . . . 117
5.5.1 The decomposition of the generalization error in regression . 117
5.5.2 The decomposition of the generalization error in classification 120
5.6 The supervised learning procedure . . . 121
5.7 Validation techniques . . . 122
5.7.1 The resampling methods . . . 122
5.8 Concluding remarks . . . 124

**6 The machine learning procedure** **125**
6.1 Introduction . . . 125
6.2 Problem formulation . . . 126
6.3 Experimental design . . . 126
6.4 Data pre-processing . . . 126
6.5 The dataset . . . 127
6.6 Parametric identification . . . 128
6.6.1 Error functions . . . 128
6.6.2 Parameter estimation . . . 128
6.7 Structural identification . . . 132
6.7.1 Model generation . . . 133
6.7.2 Validation . . . 134
6.7.3 Model selection criteria . . . 138
6.8 Concluding remarks . . . 139

**7 Linear approaches** **141**
7.1 Linear regression . . . 141
7.1.1 The univariate linear model . . . 141
7.1.2 Least-squares estimation . . . 142
7.1.3 Maximum likelihood estimation . . . 144
7.1.4 Partitioning the variability . . . 145
7.1.5 Test of hypotheses on the regression model . . . 145
7.1.6 Interval of confidence . . . 146

7.1.7 Variance of the response . . . 146
7.1.8 Coefficient of determination . . . 150
7.1.9 Multiple linear dependence . . . 150
7.1.10 The multiple linear regression model . . . 150
7.1.11 The least-squares solution . . . 151
7.1.12 Variance of the prediction . . . 153
7.1.13 The HAT matrix . . . 153
7.1.14 Generalization error of the linear model . . . 153
7.1.15 The expected empirical error . . . 154
7.1.16 The PSE and the FPE . . . 156
7.2 The PRESS statistic . . . 158
7.3 The weighted least-squares . . . 160
7.3.1 Recursive least-squares . . . 161
7.4 Discriminant functions for classification . . . 164
7.4.1 Perceptrons . . . 168
7.4.2 Support vector machines . . . 170

**8 Nonlinear approaches** **177**
8.1 Nonlinear regression . . . 179
8.1.1 Artificial neural networks . . . 180
8.1.2 From global modeling to divide-and-conquer . . . 187
8.1.3 Classification and Regression Trees . . . 188
8.1.4 Basis Function Networks . . . 193
8.1.5 Radial Basis Functions . . . 193
8.1.6 Local Model Networks . . . 194
8.1.7 Neuro-Fuzzy Inference Systems . . . 196
8.1.8 Learning in Basis Function Networks . . . 196
8.1.9 From modular techniques to local modeling . . . 201
8.1.10 Local modeling . . . 201
8.2 Nonlinear classification . . . 212
8.2.1 Naive Bayes classifier . . . 212
8.2.2 SVM for nonlinear classification . . . 214

**9 Model averaging approaches** **217**
9.1 Stacked regression . . . 217
9.2 Bagging . . . 218
9.3 Boosting . . . 221
9.3.1 The Ada Boost algorithm . . . 221
9.3.2 The arcing algorithm . . . 224
9.3.3 Bagging and boosting . . . 225

**10 Feature selection** **227**
10.1 Curse of dimensionality . . . 227
10.2 Approaches to feature selection . . . 228
10.3 Filter methods . . . 229
10.3.1 Principal component analysis . . . 229
10.3.2 Clustering . . . 230
10.3.3 Ranking methods . . . 230
10.4 Wrapping methods . . . 232
10.4.1 Wrapping search strategies . . . 232
10.5 Embedded methods . . . 233
10.5.1 Shrinkage methods . . . 233
10.6 Averaging and feature selection . . . 234
10.7 Feature selection from an information-theoretic perspective . . . 234

10.7.1 Relevance, redundancy and interaction . . . . . 235
10.7.2 Information theoretic filters . . . . . 237
10.8 Conclusion . . . . . 238

**11 Conclusions** **239**
11.1 Causality and dependencies . . . . . 240

**A Unsupervised learning** **243**
A.1 Probability density estimation . . . . . 243
A.1.1 Nonparametric density estimation . . . . . 243
A.1.2 Semi-parametric density estimation . . . . . 245
A.2 K-means clustering . . . . . 248
A.3 Fuzzy clustering . . . . . 249
A.4 Fuzzy c-ellyptotypes . . . . . 250

**B Some statistical notions** **253**
B.1 Useful relations . . . . . 253
B.2 Convergence of random variables . . . . . 253
B.3 Limits and probability . . . . . 254
B.4 Expected value of a quadratic form . . . . . 254
B.5 The matrix inversion formula . . . . . 255
B.6 Proof of Eq. (5.4.22) . . . . . 255
B.7 Biasedness of the quadratic empirical risk . . . . . 255

**C Kernel functions** **257**

**D Datasets** **259**
D.1 USPS dataset . . . . . 259
D.2 Golub dataset . . . . . 259

# Chapter 1

# Introduction

In recent years, a growing number of organizations have been allocating vast amount of resources to construct and maintain databases and data warehouses. In scientific endeavours, data refers to carefully collected observations about some phenomenon under study. In business, data capture information about economic trends, critical markets, competitors and customers. In manufacturing, data record machinery performances and production rates in different conditions. There are essentially two reasons why people gather increasing volumes of data: first, they think some valuable assets are implicitly coded within them, and computer technology enables effective data storage at reduced costs.

The idea of extracting useful knowledge from volumes of data is common to many disciplines, from statistics to physics, from econometrics to system identification and adaptive control. The procedure for finding useful patterns in data is known by different names in different communities, viz., knowledge extraction, pattern analysis, data processing. More recently, the set of computational techniques and tools to support the modelling of large amount of data is being grouped under the more general label of *machine learning* [46].

The need for programs that can learn was stressed by Alan Turing who argued that it may be too ambitious to write from scratch programs for tasks that even human must learn to perform. This handbook aims to present the statistical foundations of machine learning intended as the discipline which deals with the automatic design of models from data. In particular, we focus on *supervised learning* problems (Figure 1.1), where the goal is to model the relation between a set of *input* variables, and one or more *output* variables, which are considered to be dependent on the inputs in some manner.

Since the handbook deals with artificial learning methods, we do not take into consideration any argument of biological or cognitive plausibility of the learning methods we present. Learning is postulated here as a problem of statistical estimation of the dependencies between variables on the basis of empirical data.

The relevance of statistical analysis arises as soon as there is a need to extract useful information from data records obtained by repeatedly measuring an observed phenomenon. Suppose we are interested in learning about the relationship between two variables $x$ (e.g. the height of a child) and $y$ (e.g. the weight of a child) which are quantitative observations of some phenomenon of interest (e.g. obesity during childhood). Sometimes, the *a priori* knowledge that describes the relation between $x$ and $y$ is available. In other cases, no satisfactory theory exists and all that we can use are repeated measurements of $x$ and $y$. In this book our focus is the second situation where we assume that only a set of observed data is available. The reasons for addressing this problem are essentially two. First, the more complex is the input/output relation, the less effective will be the contribution of a human

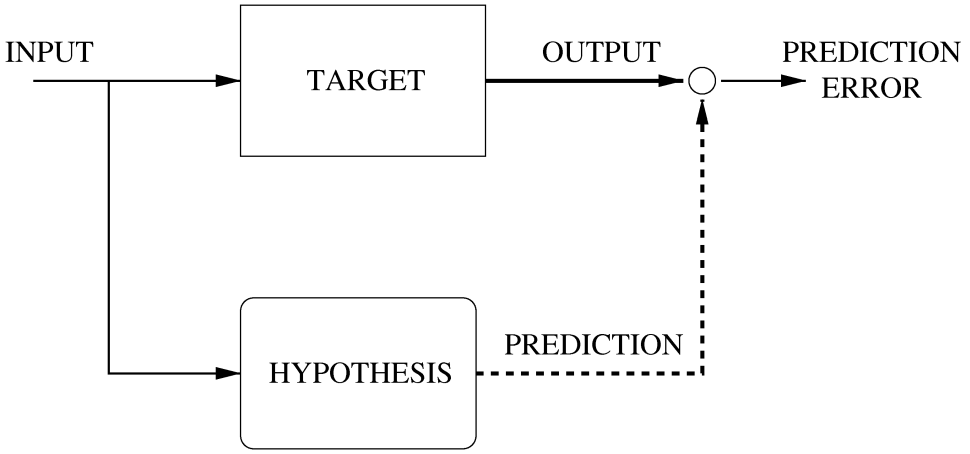



Figure 1.1: The supervised learning setting. Machine learning aims to infer from observed data the best model of the stochastic input/output dependency.

expert in extracting a model of the relation. Second, data driven modelling may be a valuable support for the designer also in modelling tasks where he can take advantage of existing knowledge.

## Modelling from data

Modelling from data is often viewed as an art, mixing an expert's insight with the information contained in the observations. A typical modelling process cannot be considered as a sequential process but is better represented as a loop with many feedback paths and interactions with the model designer. Various steps are repeated several times aiming to reach, through continuous refinements, a good description of the phenomenon underlying the data.

The process of modelling consists of a *preliminary* phase which brings the data from their original form to a structured configuration and a *learning* phase which aims to select the *model*, or *hypothesis*, that best approximates the data (Figure 1.2).

The preliminary phase can be decomposed in the following steps:

**Problem formulation.** Here the model designer chooses a particular application domain, a phenomenon to be studied, and hypothesizes the existence of a (stochastic) relation (or dependency) between the measurable variables.

**Experimental design.** This step aims to return a dataset which, ideally, should be made of samples that are well-representative of the phenomenon in order to maximize the performance of the modelling process [34].

**Pre-processing.** In this step, raw data are cleaned to make learning easier. Pre-processing includes a large set of actions on the observed data, such as noise filtering, outlier removal, missing data treatment [78], feature selection, and so on.

Once the preliminary phase has returned the dataset in a structured input/output form (e.g. a two-column table), called *training set*, the learning phase begins. A graphical representation of a training set for a simple learning problem with one input variable $x$ and one output variable $y$ is given in Figure 1.3. This manuscript

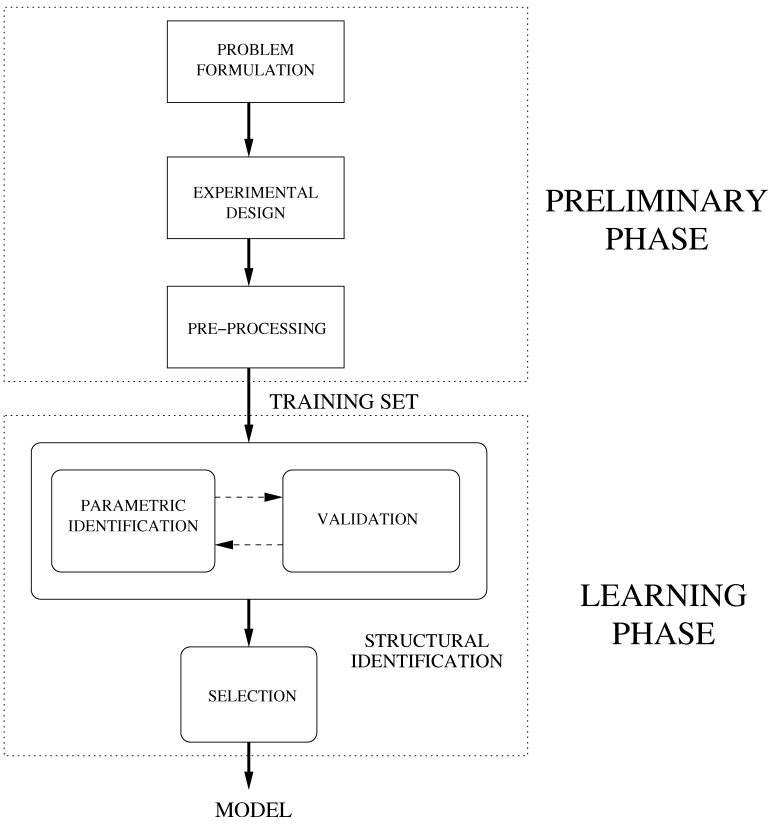



Figure 1.2: The modelling process and its decomposition in preliminary phase and learning phase.

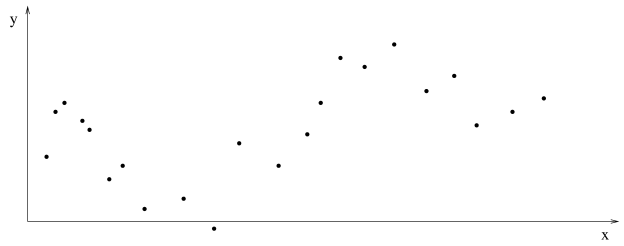



Figure 1.3: A training set for a simple learning problem with one input variable $x$ and one output variable $y$. The dots represent the observed samples.

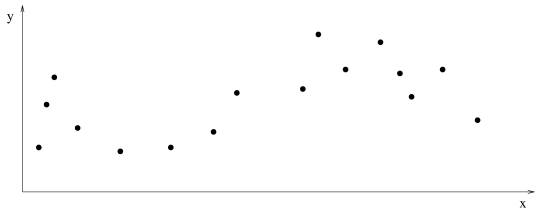



Figure 1.4: A second realization of the training set for the same phenomenon observed in Figure 1.3. The dots represent the observed samples.

will focus exclusively on this second phase assuming that the preliminary steps have already been performed by the model designer.

Suppose that, on the basis of the collected data, we wish to learn the unknown dependency existing between the $x$ variable and the $y$ variable. In practical terms, the knowledge of this dependency could shed light on the observed phenomenon and allow us to predict the value of the output $y$ for a given input (e.g. what is the expected weight of child which is 120cm tall?). What is difficult and tricky in this task is the finiteness and the random nature of data. For instance a second set of observations of the same pair of variables could produce a dataset (Figure 1.4) which is not identical to the one in Figure 1.3 though both originate from the same measurable phenomenon. This simple fact suggest that a simple interpolation of the observed data would not produce an accurate model of the data.

The goal of machine learning is to formalize and optimize the procedure which brings from data to model and consequently from data to predictions. A learning procedure can be concisely defined as a search, in a space of possible model configurations, of the model which best represents the phenomenon underlying the data. As a consequence, a learning procedure requires both a *search space*, where possible solutions may be found, and an *assessment criterion* which measures the quality of the solutions in order to select the best one.

The search space is defined by the designer using a set of nested classes with increasing complexity. For our introductory purposes, it is sufficient to consider here a *class* as a set of input/output models (e.g. the set of polynomial models)

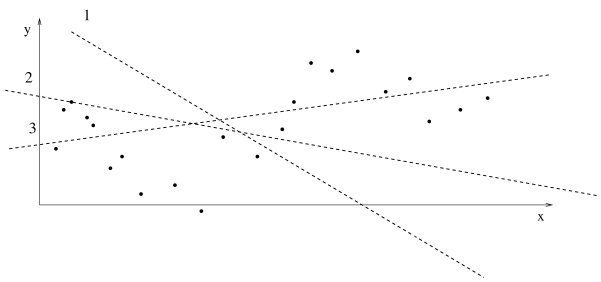



Figure 1.5: Training set and three parametric models which belong to the class of first order polynomials.

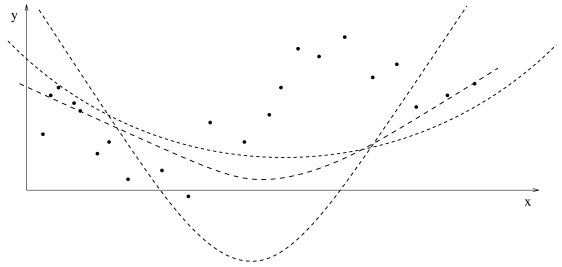



Figure 1.6: Training set and three parametric models which belong to the class of second order polynomials.

with the same *model structure* (e.g. second order degree) and the *complexity* of the class as a measure of the set of input/output mappings which can approximated by the models belonging to the class.

Figure 1.5 shows the training set of Figure 1.3 together with three parametric models which belong to the class of first-order polynomials. Figure 1.6 shows the same training set with three parametric models which belongs to the class of second-order polynomials.

The reader could visually decide whether the class of second order models is more adequate or not than the first-order class to model the dataset. At the same time she could guess which among the three plotted models is the one which produces the best fitting.

In real high-dimensional settings, however, a visual assessment of the quality of a model is not sufficient. Data-driven quantitative criteria are therefore required. We will assume that the goal of learning is to attain a good *statistical generalization*. This means that the selected model is expected to return an accurate prediction of the dependent (output) variable when values of the independent (input) variables, which are not part of the training set, are presented.

Once the classes of models and the assessment criteria are fixed, the goal of a learning algorithm is to search i) for the best class of models and ii) for the best parametric model within such a class. Any supervised learning algorithm is then made of two nested loops denoted as the *structural* identification loop and the *parametric* identification loop.

Structural identification is the outer loop which seeks the model structure which is expected to have the best performance. It is composed of a *validation* phase,

which assesses each model structure on the basis of the chosen assessment criterion, and a *selection* phase which returns the best model structure on the basis of the validation output. Parametric identification is the inner loop which returns the best model for a fixed model structure. We will show that the two procedures are intertwined since the structural identification requires the outcome of the parametric step in order to assess the goodness of a class.

## Statistical machine learning

On the basis of the previous section we could argue that learning is nothing more than a standard problem of optimization. Unfortunately, reality is far more complex. In fact, because of the finite amount of data and their random nature, there exists a strong correlation between parametric and structural identification steps, which makes non-trivial the problem of assessing and, finally, choosing the prediction model. In fact, the random nature of the data demands a definition of the problem in stochastic terms and the adoption of statistical procedures to choose and assess the quality of a prediction model. In this context a challenging issue is how to determine the class of models more appropriate to our problem. Since the results of a learning procedure is found to be sensitive to the class of models chosen to fit the data, statisticians and machine learning researchers have proposed over the years a number of machine learning algorithms. Well-known examples are linear models, neural networks, local modelling techniques, support vector machines and regression trees. The aim of such learning algorithms, many of which are presented in this book, is to combine high generalization with an effective learning procedure.

However, the ambition of this handbook is to present machine learning as a scientific domain which goes beyond the mere collection of computational procedures. Since machine learning is deeply rooted in conventional statistics, any introduction to this topic must include some introductory chapters to the foundations of probability, statistics and estimation theory. At the same time we intend to show that machine learning widens the scope of conventional statistics by focusing on a number of topics often overlooked by statistical literature, like nonlinearity, large dimensionality, adaptivity, optimization and analysis of massive datasets.

This manuscript aims to find a good balance between theory and practice by situating most of the theoretical notions in a real context with the help of practical examples and real datasets. All the examples are implemented in the statistical programming language R [101]. For an introduction to R we refer the reader to [33, 117]. This practical connotation is particularly important since machine learning techniques are nowadays more and more embedded in plenty of technological domains, like bioinformatics, robotics, intelligent control, speech and image recognition, multimedia, web and data mining, computational finance, business intelligence.

## Outline

The outline of the book is as follows. Chapter 2 summarize thes relevant background material in probability. Chapter 3 introduces the parametric approach to parametric estimation and hypothesis testing.

Chapter 4 presents some nonparametric alternatives to the parametric techniques discussed in Chapter 3.

Chapter 5 introduces supervised learning as the statistical problem of assessing and selecting a hypothesis function on the basis of input/output observations.

Chapter 6 reviews the steps which lead from raw observations to a final model. This is a methodological chapter which introduces some algorithmic procedures

underlying most of the machine learning techniques.

Chapter 7 presents conventional linear approaches to regression and classification.

Chapter 8 introduces some machine learning techniques which deal with nonlinear regression and classification tasks.

Chapter 9 presents the model averaging approach, a recent and powerful way for obtaining improved generalization accuracy by combining several learning machines.

Although the book focuses on supervised learning, some related notions of unsupervised learning and density estimation are given in Appendix A.

## 1.1 Notations

Throughout this manuscript, boldface denotes random variables and normal font is used for istances (realizations) of random variables. Strictly speaking, one should always distinguish in notation between a random variable and its realization. However, we will adopt this extra notational burden only when the meaning is not clear from the context.

As far as variables are concerned, lowercase letters denote scalars or vectors of observables, greek letters denote parameter vectors and uppercase denotes matrices. Uppercase in italics denotes generic sets while uppercase in greek letters denotes sets of parameters.

## Generic notation

| | |
|---|---|
| $\theta$ | Parameter vector. |
| $\boldsymbol{\theta}$ | Random parameter vector. |
| $M$ | Matrix. |
| $[N \times n]$ | Dimensionality of a matrix with $N$ rows and $n$ columns. |
| $M^T$ | Transpose of the matrix $M$. |
| $\text{diag}[m_1, \ldots, m_N]$ | Diagonal matrix with diagonal $[m_1, \ldots, m_N]$ |
| $\mathbf{M}$ | Random matrix. |
| $\hat{\theta}$ | Estimate of $\theta$. |
| $\hat{\boldsymbol{\theta}}$ | Estimator of $\theta$. |
| $\tau$ | Index in an iterative algorithm. |

## Probability Theory notation

| | |
|---|---|
| $\Omega$ | Set of possible outcomes. |
| $\omega$ | Outcome (or elementary event). |
| $\{\mathcal{E}\}$ | Set of possible events. |
| $\mathcal{E}$ | Event. |
| $\text{Prob}\{\mathcal{E}\}$ | Probability of the event $\mathcal{E}$. |
| $(\Omega, \{\mathcal{E}\}, \text{Prob}\{\cdot\})$ | Probabilistic model of an experiment. |
| $\mathcal{Z}$ | Domain of the random variable $\mathbf{z}$. |
| $P(z)$ | Probability distribution of a discrete random variable $\mathbf{z}$. Also $P_{\mathbf{z}}(z)$. |
| $F(z) = \text{Prob}\{\mathbf{z} \leq z\}$ | Distribution function of a continuous random variable $\mathbf{z}$. Also $F_{\mathbf{z}}(z)$. |
| $p(z)$ | Probability density of a continuous r.v.. Also $p_{\mathbf{z}}(z)$. |
| $E[\mathbf{z}]$ | Expected value of the random variable $\mathbf{z}$. |
| $E_{\mathbf{x}}[\mathbf{z}] = \int_{\mathcal{X}} z(x,y)p(x)dx$ | Expected value of the random variable $\mathbf{z}$ averaged over $\mathbf{x}$. |
| $\text{Var}[\mathbf{z}]$ | Variance of the random variable $\mathbf{z}$. |
| $L(\theta)$ | Likelihood of a parameter $\theta$. |
| $l(\theta)$ | Log-Likelihood of a parameter $\theta$. |
| $l_{emp}(\theta)$ | Empirical Log-likelihood of a parameter $\theta$. |

## Learning Theory notation

| | |
|---|---|
| $\mathbf{x}$ | Multidimensional input variable. |
| $\mathcal{X} \subset \Re^n$ | Input space. |
| $\mathbf{y}$ | Multidimensional output variable. |
| $\mathcal{Y} \subset \Re$ | Output space. |
| $x_i$ | $i^{\text{th}}$ realization of the random vector $\mathbf{x}$. |
| $f(x)$ | Target regression function. |
| $\mathbf{w}$ | Random noise variable. |
| $z_i = \langle x_i, y_i \rangle$ | Input-output sample: $i^{\text{th}}$ case in training set. |
| $N$ | Number of available samples. |
| $D_N = \{z_1, z_2, \ldots, z_N\}$ | Training set. |
| $\Lambda$ | Class of hypothesis. |
| $\alpha$ | Hypothesis parameter vector. |
| $h(x, \alpha)$ | Hypothesis function. |
| $\Lambda_s$ | Hypothesis class of complexity $s$. |
| $L(y, f(x, \alpha))$ | Loss function. |
| $R(\alpha)$ | Functional risk. |
| $\alpha_0$ | $\arg\min_{\alpha \in \Lambda} R(\alpha)$. |
| $R_{\text{emp}}(\alpha)$ | Empirical functional risk. |
| $\alpha_N$ | $\arg\min R_{emp}(\alpha)$. |
| $G_N$ | Mean integrated squared error (MISE). |
| $l$ | Number of folds in cross-validation. |
| $\hat{G}_{\text{cv}}$ | Cross-validation estimate of $G_N$. |
| $\hat{G}_{\text{loo}}$ | Leave-one-out estimate of $G_N$. |
| $N_{tr}$ | Number of samples used for training in cross-validation. |
| $N_{ts}$ | Number of samples used for test in cross-validation. |
| $\alpha^i_{N_{tr}} \quad i = 1, \ldots, l$ | Parameter which minimizes the empirical risk of $D_{N_{tr}}$. |
| $D_{(i)}$ | Training set with the $i^{\text{th}}$ sample set aside. |
| $\alpha_{N(i)}$ | Parameter which minimizes the empirical risk of $D_{(i)}$. |
| $\hat{G}_{\text{boot}}$ | Bootstrap estimate of $G_N$. |
| $D^*_N$ | Bootstrap training set of size $N$ generated by $D_N$ with replacement. |
| $N_b$ | Number of bootstrap samples. |

## Data analysis notation

| | |
|---|---|
| $x_{ij}$ | $j^{\text{th}}$ element of vector $x_i$. |
| $X_{ij}$ | $ij^{\text{th}}$ element of matrix X. |
| $q$ | Query point (point in the input space where a prediction is required). |
| $\hat{y}_q$ | Prediction in the query point. |
| $\hat{y}_i^{-j}$ | Leave-one-out prediction in $x_i$ with the $j^{\text{th}}$ sample set aside. |
| $e_j^{\text{loo}} = y_j - \hat{y}_j^{-j}$ | Leave-one-out error with the $j^{\text{th}}$ sample set aside. |
| $K(\cdot)$ | Kernel function. |
| $B$ | Bandwidth. |
| $\beta$ | Linear coefficients vector. |
| $\hat{\beta}$ | Least-squares parameters vector. |
| $\hat{\beta}^{-j}$ | Least-squares parameters vector with the $j^{\text{th}}$ sample set aside. |
| $h_j(x, \alpha)$ | $j^{th}$, $j = 1, \ldots, m$, local model in a modular architecture. |
| $\rho_j$ | Activation or basis function. |
| $\eta_j$ | Set of parameters of the activation function. |

# Chapter 2

# Foundations of probability

Probability theory is the discipline concerned with the study of uncertain (or random) phenomena and probability is the mathematical language adopted for quantifying uncertainty. Such phenomena, although not predictable in a deterministic fashion, may present some regularities and consequently be described mathematically by idealized probabilistic models. These models consist of a list of all possible outcomes together with the respective probabilities. The theory of probability makes possible to infer from these models the patterns of future behaviour.

This chapter presents the basic notions of probability which serves as a necessary background to understand the statistical aspects of machine learning. We ask the reader to become acquainted with two aspects: the notion of random variable as a compact representation of uncertain knowledge and the use of probability as an effective formal tool to manipulate and process uncertain information. In particular, we suggest the reader give special attention to the notions of conditional and joint probability. As we will see in the following, these two related notions are extensively used by statistical modelling and machine learning to define the dependence and the relationships between random variables.

## 2.1 The random model of uncertainty

We define a *random experiment* as any action or process which generates results or observations which cannot be predicted with certainty. *Uncertainty stems from the existence of alternatives.* In other words, each uncertain phenomenon is characterized by a multiplicity of possible configurations or outcomes. Weather is uncertain since it can take multiple forms (e.g. sunny, rainy, cloudy,...). Other examples of random experiments are tossing a coin, rolling dice, or measuring the time to reach home.

A random experiment is then characterized by a *sample space* $\Omega$ that is a (finite or infinite) set of all the possible outcomes (or configurations) $\omega$ of the experiment. The elements of the set $\Omega$ are called *experimental outcomes* or *realizations.* For example, in the die experiment, $\Omega = \{\omega_1, \omega_2, \ldots, \omega_6\}$ and $\omega_i$ stands for the outcome corresponding to getting the face with the number $i$. If $\omega$ is the outcome of a measurement of some physical quantity, e.g. pressure, then we could have $\Omega = \mathbb{R}^+$.

*The representation of an uncertain phenomenon is the result of a modelling activity and as such it is not necessarily unique.* In other terms different representations of a random experiment are possible. In the die experiment, we could define an alternative sample space made of two sole outcomes: numbers equal to and different from 1. Also we could be interested in representing the uncertainty of two consecutive tosses. In that case the outcome would be the pair $(\omega_i, \omega_j)$.

*Uncertainty stems from variability.* Each time we observe a random phenomenon, we may observe different outcomes. In the probability jargon, observing a random phenomenon is interpreted as the realization of a random experiment. A single performance of a random experiment is called a *trial*. This means that after each trial we observe one outcome $\omega_i \in \Omega$.

A subset of experimental outcomes is called an *event*. Consider a trial which generated the outcome $\omega_i$: we say that an event $\mathcal{E}$ occurred during the trial if the set $\mathcal{E}$ contains the element $\omega_i$. For example, in the die experiment, an event is the set of even values $\mathcal{E} = \{\omega_2, \omega_4, \omega_6\}$. This means that when we observe the outcome $\omega_4$ the event *even number* takes place.

An event composed of a single outcome, e.g. $\mathcal{E} = \{\omega_1\}$ is called an *elementary event*.

Note that since events $\mathcal{E}$ are subsets, we can apply to them the terminology of the set theory:

- $\Omega$ refers to the certain event i.e. the event that occurs in every trial.
- the notation
$$\mathcal{E}^c = \{\omega : \omega \notin \mathcal{E}\}$$
denotes the complement of $\mathcal{E}$.
- the notation
$$\mathcal{E}_1 \cup \mathcal{E}_2 = \{\omega \in \Omega : \omega \in \mathcal{E}_1 \text{ OR } \omega \in \mathcal{E}_2\}$$
refers to the event that occurs when $\mathcal{E}_1$ or $\mathcal{E}_2$ or both occur.
- the notation
$$\mathcal{E}_1 \cap \mathcal{E}_2 = \{\omega \in \Omega : \omega \in \mathcal{E}_1 \text{ AND } \omega \in \mathcal{E}_2\}$$
refers to the event that occurs when both $\mathcal{E}_1$ and $\mathcal{E}_2$ occur.
- two events $\mathcal{E}_1$ and $\mathcal{E}_2$ are *mutually exclusive* or *disjoints* if
$$\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$$
that is each time that $\mathcal{E}_1$ occurs, $\mathcal{E}_2$ does not occur.
- a partition of $\Omega$ is a set of disjoint sets $\mathcal{E}_j$, $j = 1, \ldots, n$ such that
$$\cup_{j=1}^{n} \mathcal{E}_j = \Omega$$
- given an event $\mathcal{E}$ we define the *indicator function* of $\mathcal{E}$ by
$$I_{\mathcal{E}}(\omega) = \begin{cases} 1 & \text{if } \omega \in \mathcal{E} \\ 0 & \text{if } \omega \notin \mathcal{E} \end{cases} \tag{2.1.1}$$

Let us consider now the notion of *class of events*. An arbitrary collection of subsets of $\Omega$ is not a class of events. We require that if $\mathcal{E}_1$ and $\mathcal{E}_2$ are events, the same holds also for the intersection $\mathcal{E}_1 \cap \mathcal{E}_2$ and the union $\mathcal{E}_1 \cup \mathcal{E}_2$. A set of events that satisfies these conditions is called, in mathematical terms, a *Borel field* [91]. We will consider only Borel fields since we want to deal not only with the probabilities of single events, but also with the probabilities of their unions and intersections.

### 2.1.1 Axiomatic definition of probability

*Probability is a measure of uncertainty.* Once a random experiment is defined, we call *probability of the event* $\mathcal{E}$ the real number $\text{Prob}\{\mathcal{E}\} \in [0,1]$ assigned to each event $\mathcal{E}$. The function $\text{Prob}\{\cdot\} : \Omega \to [0,1]$ is called *probability measure* or *probability distribution* and must satisfy the following three axioms:

1. $\text{Prob}\{\mathcal{E}\} \geq 0$ for any $\mathcal{E}$.
2. $\text{Prob}\{\Omega\} = 1$
3. $\text{Prob}\{\mathcal{E}_1 + \mathcal{E}_2\} = \text{Prob}\{\mathcal{E}_1\} + \text{Prob}\{\mathcal{E}_2\}$ if $\mathcal{E}_1$ and $\mathcal{E}_2$ are mutually exclusive.

These conditions are known as the axioms of the theory of probability [76]. The first axiom states that all the probabilities are nonnegative real numbers. The second axiom attributes a probability of unity to the universal event $\Omega$, thus providing a normalization of the probability measure. The third axiom states that the probability function must be additive, consistently with the intuitive idea of how probabilities behave.

All probabilistic results are based directly or indirectly on the axioms and only the axioms, for instance

$$\mathcal{E}_1 \subset \mathcal{E}_2 \Rightarrow \text{Prob}\{\mathcal{E}_1\} \leq \text{Prob}\{\mathcal{E}_2\} \tag{2.1.2}$$

$$\text{Prob}\{\mathcal{E}^c\} = 1 - \text{Prob}\{\mathcal{E}\} \tag{2.1.3}$$

There are many interpretations and justifications of these axioms and we discuss briefly the frequentist and the Bayesian interpretation in Section 2.1.3. What is relevant here is that the probability function is a formalization of uncertainty and that most of its properties and results appear to be coherent with the human perception of uncertainty [69].

So from a mathematician point of view, probability is easy to define: it is a countably additive set function defined on a *Borel field* , with a total mass of one.

In practice, however, a major question remains still open: how to compute the probability value $\text{Prob}\{\mathcal{E}\}$ for a generic event $\mathcal{E}$? The assignment of probabilities is perhaps the most difficult aspect of constructing probabilistic models. Although the theory of probability is neutral, that is it can make inferences regardless of the probability assignments, its results will be strongly affected by the choice of a particular assignment. This means that, if the assignments are inaccurate, the predictions of the model will be misleading and will not reflect the real behaviour of the modelled phenomenon. In the following sections we are going to present some procedures which are typically adopted in practice.

### 2.1.2 Symmetrical definition of probability

Consider a random experiment where the sample space is made of $M$ symmetric outcomes (i.e., they are equally likely to occur). Let the number of outcomes which are favourable to the event $\mathcal{E}$ (i.e. if they occur then the event $\mathcal{E}$ takes place) be $M_{\mathcal{E}}$.

An intuitive definition of probability (also known as the classical definition) of the event $\mathcal{E}$ that adheres to the axioms is

$$\text{Prob}\{\mathcal{E}\} = \frac{M_{\mathcal{E}}}{M} \tag{2.1.4}$$

In other words, *the probability of an event equals the ratio of its favourable outcomes to the total number of outcomes provided that all outcomes are equally likely* [91].

This is typically the approach adopted when the sample space is made of the six faces of a fair die. Also in most cases the symmetric hypothesis is accepted as self evident: "if a ball is selected at random from a bowl containing $W$ white balls and $B$ black balls, the probability that we select a white one is $W/(W+B)$". Note that this number is determined without any experimentation and is based on symmetrical assumptions. But how to be sure that the symmetrical hypothesis holds? Think for instance to the probability that a newborn be a boy. Is this a symmetrical case? Moreover, how would one define the probability of an event if the symmetrical hypothesis does not hold?

### 2.1.3 Frequentist definition of probability

Let us consider a random experiment and an event $\mathcal{E}$. Suppose we repeat the experiment $N$ times and that we record the number of times $N_{\mathcal{E}}$ that the event $\mathcal{E}$ occurs. The quantity

$$\frac{N_{\mathcal{E}}}{N} \tag{2.1.5}$$

comprised between 0 and 1 is known as the *relative frequency* of $\mathcal{E}$. It can be observed that if the experiment is carried out a large number of times under exactly the same conditions, the frequency converges to a fixed value for increasing $N$. This observation led von Mises to use the notion of frequency as a foundation for the notion of probability.

**Definition 1.1** (von Mises)**.** The probability $\text{Prob}\{\mathcal{E}\}$ of an event $\mathcal{E}$ is the limit

$$\text{Prob}\{\mathcal{E}\} = \lim_{N\to\infty} \frac{N_{\mathcal{E}}}{N}$$

where $N$ is the number of observations and $N_{\mathcal{E}}$ is the number of times that $\mathcal{E}$ occurred.

This definition appears reasonable and it is compatible with the axioms in Section 2.1.1. However, in practice, in any physical experience the number $N$ is finite[1] and the limit has to be accepted as a hypothesis, not as a number that can be determined experimentally [91]. Notwithstanding, the frequentist interpretation is very important to show the links between theory and application. At the same time it appears inadequate to represent probability when it is used to model a degree-of-belief. Think for instance to the probability that your professor wins a Nobel Prize: how to define a number $N$ of repetitions?

An important alternative interpretation of the probability measure comes then from the Bayesian approach. This approach proposes a degree-of-belief interpretation of probability according to which $\text{Prob}\{\mathcal{E}\}$ measures an observer's strength of belief that $\mathcal{E}$ is or will be true. This manuscript will not cover the Bayesian approach to statistics and data analysis. Interested readers are referred to [51].

### 2.1.4 The Law of Large Numbers

A well-known justification of the frequentist approach is provided by the Weak Law of Large Numbers, proposed by Bernoulli.

**Theorem 1.2.** *Let $Prob\{\mathcal{E}\} = p$ and suppose that the event $\mathcal{E}$ occurs $N_{\mathcal{E}}$ times in $N$ trials. Then, $\frac{N_{\mathcal{E}}}{N}$ converges to $p$* in probability*, that is, for any $\epsilon > 0$,*

$$Prob\left\{\left|\frac{N_{\mathcal{E}}}{N} - p\right| \leq \epsilon\right\} \to 1 \quad as \quad N \to \infty$$

[1] As Keynes said "In the long run we are all dead".

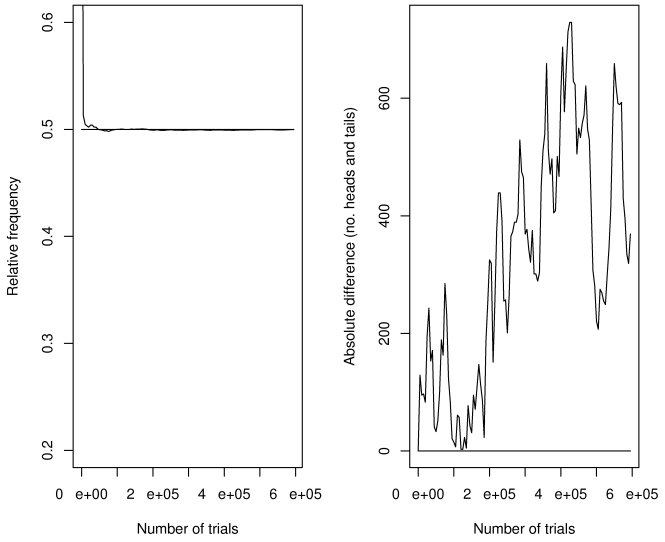



Figure 2.1: Fair coin tossing random experiment: evolution of the relative frequency (left) and of the absolute difference between the number of heads and tails (R script `freq.R`).

According to this theorem, the ratio $N_{\mathcal{E}}/N$ is close to $p$ in the sense that, for any $\epsilon > 0$, the probability that $|N_{\mathcal{E}}/N - p| \leq \epsilon$ tends to 1 as $N \to \infty$. This results justifies the widespread use of Monte Carlo simulation to solve numerically probability problems.

Note that this does NOT mean that $N_{\mathcal{E}}$ will be close to $N \cdot p$. In fact,

$$\text{Prob}\{N_{\mathcal{E}} = Np\} \approx \frac{1}{\sqrt{2\pi Np(1-p)}} \to 0, \qquad \text{as } N \to \infty \tag{2.1.6}$$

For instance, in a fair coin-tossing game this law does not imply that the absolute difference between the number of heads and tails should oscillate close to zero [113] (Figure 2.1). On the contrary it could happen that the absolute difference keeps growing (though slower than the number of tosses).

**Script**

```
### script freq.R ###

set.seed(1)  ## initialization random seed
R<-600000  ## number of trials

tosses<-sample(c("H","T"),R,replace=T)

gap<-NULL
freq<-NULL
par(mfrow=c(1,2))
trials<-NULL
for (r in seq(1,R,by=5000)){
  lH<-length(which(tosses[1:r]=="H"))
```

```
  lT<-length(which(tosses[1:r]=="T"))
  gap<-c(gap,abs(lH-lT))
  freq<-c(freq,lH/r)
  trials<-c(trials,r)
  plot(trials,freq,type="l",ylim=c(0.2,0.6),xlab="Number of trials",
       ylab="Relative frequency")
  lines(trials,0.5+numeric(length(freq)))
  plot(trials,gap,type="l",xlab="Number of trials",
       ylab="Absolute difference (no. heads and tails)")
  lines(trials,numeric(length(gap)))


}
```

•

### 2.1.5 Independence and conditional probability

Let us introduce the definition of independent events.

**Definition 1.3** (Independent events)**.** Two events $\mathcal{E}_1$ and $\mathcal{E}_2$ are *independent* if and only if

$$\text{Prob}\{\mathcal{E}_1 \cap \mathcal{E}_2\} = \text{Prob}\{\mathcal{E}_1\}\,\text{Prob}\{\mathcal{E}_2\} \tag{2.1.7}$$

and we write $\mathcal{E}_1 \perp\!\!\!\perp \mathcal{E}_2$.

Note that the quantity $\text{Prob}\{\mathcal{E}_1 \cap \mathcal{E}_2\}$ is often referred to as *joint probability* and denoted by $\text{Prob}\{\mathcal{E}_1, \mathcal{E}_2\}$.

As an example of two independent events, think of two outcomes of a roulette wheel or of two coins tossed simultaneously.

#### Exercise

Suppose that a fair die is rolled and that the number $x$ appears. Let $\mathcal{E}_1$ be the event that the number $x$ is even, $\mathcal{E}_2$ be the event that the number $x$ is greater than or equal to 3, $\mathcal{E}_3$ be the event that the number $x$ is a 4,5 or 6.

Are the events $\mathcal{E}_1$ and $\mathcal{E}_2$ independent? Are the events $\mathcal{E}_1$ and $\mathcal{E}_3$ independent?

•

#### Example

Let $\mathcal{E}_1$ and $\mathcal{E}_2$ two disjoint events with positive probability. Can they be independent? The answer is no since

$$\text{Prob}\{\mathcal{E}_1 \cap \mathcal{E}_2\} = 0 \neq \text{Prob}\{\mathcal{E}_1\}\,\text{Prob}\{\mathcal{E}_2\} > 0$$

•

Let $\mathcal{E}_1$ be an event such that $\text{Prob}\{\mathcal{E}_1\} > 0$ and $\mathcal{E}_2$ another event. We define the conditional probability of $\mathcal{E}_2$ given that $\mathcal{E}_1$ has occurred as follows:

**Definition 1.4** (Conditional probability)**.** If $\text{Prob}\{\mathcal{E}_1\} > 0$ then the conditional probability of $\mathcal{E}_2$ given $\mathcal{E}_1$ is

$$\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} = \frac{\text{Prob}\{\mathcal{E}_1 \cap \mathcal{E}_2\}}{\text{Prob}\{\mathcal{E}_1\}} \tag{2.1.8}$$

The following properties hold:

$$\text{If } \mathcal{E}_1 \subset \mathcal{E}_2 \Rightarrow \text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} = 1 \tag{2.1.9}$$

$$\text{If } \mathcal{E}_2 \subset \mathcal{E}_1 \Rightarrow \text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} \geq \text{Prob}\{\mathcal{E}_2\} \tag{2.1.10}$$

Note that for any fixed $\mathcal{E}_1$, the quantity $\text{Prob}\{\cdot|\mathcal{E}_1\}$ satisfies the axioms of probability. For instance if $\mathcal{E}_2$, $\mathcal{E}_3$ and $\mathcal{E}_4$ are disjoint events we have that

$$\text{Prob}\{\mathcal{E}_2 \cup \mathcal{E}_3 \cup \mathcal{E}_4|\mathcal{E}_1\} = \text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} + \text{Prob}\{\mathcal{E}_3|\mathcal{E}_1\} + \text{Prob}\{\mathcal{E}_4|\mathcal{E}_1\}$$

However this does NOT generally hold for $\text{Prob}\{\mathcal{E}_1|\cdot\}$, that is when we fix the term $\mathcal{E}_1$ on the left of the conditional bar. For two disjoint events $\mathcal{E}_2$ and $\mathcal{E}_3$, in general

$$\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2 \cup \mathcal{E}_3\} \neq \text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} + \text{Prob}\{\mathcal{E}_1|\mathcal{E}_3\}$$

Also it is generally NOT the case that $\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} = \text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\}$.

The following result derives from the definition of conditional probability.

**Lemma 1.** If $\mathcal{E}_1$ and $\mathcal{E}_2$ are independent events then

$$\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} = \text{Prob}\{\mathcal{E}_1\} \tag{2.1.11}$$

In qualitative terms the independence of two events means that the fact of observing (or knowing) that one of these events (e.g. $\mathcal{E}_1$) occurred, does not change the probability that the other (e.g. $\mathcal{E}_2$) will occur.

### 2.1.6 Combined experiments

Note that so far we assumed that all the events belong to the same sample space. The most interesting use of probability concerns however *combined* random experiments whose sample space

$$\Omega = \Omega_1 \times \Omega_2 \times \ldots \Omega_n$$

is the Cartesian product of several spaces $\Omega_i$, $i = 1, \ldots, n$.

For instance if we want to study the probabilistic dependence between the height and the weight of a child we have to define a joint sample space

$$\Omega = \{(w, h) : w \in \Omega^w, h \in \Omega^h\}$$

made of all pairs $(w, h)$ where $\Omega^w$ is the sample space of the random experiment describing the weight and $\Omega^h$ is the sample space of the random experiment describing the height.

Note that all the properties studied so far holds also for events which do not belong to the same sample space. For instance, given a combined experiment $\Omega = \Omega_1 \times \Omega_2$ two events $\mathcal{E}_1 \in \Omega_1$ and $\mathcal{E}_2 \in \Omega_2$ are independent iff $\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} = \text{Prob}\{\mathcal{E}_1\}$.

Some examples of real problems modelled by random combined experiments are presented in the following.

**Gambler's fallacy**

Consider a fair coin-tossing game. The outcome of two consecutive tosses can be considered as independent. Now, suppose that we assisted to a sequence of 10 consecutive tails. We could be tempted to think that the chances that the next toss will be head are now very large. This is known as the *gambler's fallacy* [113]. In fact, the fact that we assisted to a very rare event does not imply that the probability of the next event will change or rather that it will be dependent on the past.

•

**Exemple [119]**

Let us consider a medical study about the relationship between the outcome of a medical test and the presence of a disease. We model this study as the combination of two random experiments:

1. the random experiment which models the state of the patient. Its sample space is $\Omega^s = \{H, S\}$ where $H$ and $S$ stand for healthy and sick patient, respectively.
2. the random experiment which models the outcome of the medical test. Its sample space is $\Omega^o = \{+, -\}$ where $+$ and $-$ stand for positive and negative outcome of the test, respectively.

The dependency between the state of the patient and the outcome of the test can be studied in terms of conditional probability.

Suppose that out of 1000 patients, 108 respond positively to the test and that among them 9 result to be affected by the disease. Also, among the 892 patients who responded negatively to the test, only 1 is sick. According to the frequentist interpretation the probabilities of the joint events $\text{Prob}\{\mathcal{E}^s, \mathcal{E}^o\}$ can be approximated according to expression (2.1.5) by

| | $\mathcal{E}^s = S$ | $\mathcal{E}^s = H$ |
|---|---|---|
| $\mathcal{E}^o = +$ | $\frac{9}{1000} = .009$ | $\frac{108-9}{1000} = .099$ |
| $\mathcal{E}^o = -$ | $\frac{1}{1000} = .001$ | $\frac{892-1}{1000} = .891$ |

Doctors are interested in answering the following questions. What is the probability of having a positive (negative) test outcome when the patient is sick (healthy)? What is the probability of being in front of a sick (healthy) patient when a positive (negative) outcome is obtained? From the definition of conditional probability we derive

$$\text{Prob}\{\mathcal{E}^o = +|\mathcal{E}^s = S\} = \frac{\text{Prob}\{\mathcal{E}^o = +, \mathcal{E}^s = S\}}{\text{Prob}\{\mathcal{E}^s = S\}} = \frac{.009}{.009 + .001} = .9$$

$$\text{Prob}\{\mathcal{E}^o = -|\mathcal{E}^s = H\} = \frac{\text{Prob}\{\mathcal{E}^o = -, \mathcal{E}^s = H\}}{\text{Prob}\{\mathcal{E}^s = H\}} = \frac{.891}{.891 + .099} = .9$$

According to these figures, the test appears to be accurate. Do we have to expect a high probability of being sick when the test is positive? The answer is NO as shown by

$$\text{Prob}\{\mathcal{E}^s = S|\mathcal{E}^o = +\} = \frac{\text{Prob}\{\mathcal{E}^o = +, \mathcal{E}^s = S\}}{\text{Prob}\{\mathcal{E}^o = +\}} = \frac{.009}{.009 + .099} \approx .08$$

This example shows that sometimes humans tend to confound $\text{Prob}\{\mathcal{E}^s|\mathcal{E}^o\}$ with $\text{Prob}\{\mathcal{E}^o|\mathcal{E}^s\}$ and that the most intuitive response is not always the right one.

•

### 2.1.7 The law of total probability and the Bayes' theorem

Let us consider an indeterminate practical situation where a set of events $\mathcal{E}_1$, $\mathcal{E}_2$,..., $\mathcal{E}_k$ may occur. Suppose that no two such events may occur simultaneously, but at least one of them must occur. This means that $\mathcal{E}_1$, $\mathcal{E}_2$,..., $\mathcal{E}_k$ are mutually exclusive and exhaustive or, in other terms, that they form a partition of $\Omega$. The following two theorems can be shown

**Theorem 1.5** (Law of total probability)**.** *Let Prob* $\{\mathcal{E}_i\}$, $i = 1, \ldots, k$ *denote the probabilities of the* $i$*th event* $\mathcal{E}_i$ *and Prob* $\{\mathcal{E}|\mathcal{E}_i\}$, $i = 1, \ldots, k$ *the conditional probability of a generic event* $\mathcal{E}$ *given that* $\mathcal{E}_i$ *has occurred. It can be shown that*

$$\text{Prob}\{\mathcal{E}\} = \sum_{i=1}^{k} \text{Prob}\{\mathcal{E}|\mathcal{E}_i\}\,\text{Prob}\{\mathcal{E}_i\} = \sum_{i=1}^{k} \text{Prob}\{\mathcal{E} \cap \mathcal{E}_i\} \tag{2.1.12}$$

In this case the quantity Prob $\{\mathcal{E}\}$ is referred to as *marginal probability.*

**Theorem 1.6** (Bayes' theorem)**.** *The conditional ("inverse") probability of any* $\mathcal{E}_i$, $i = 1, \ldots, k$ *given that* $\mathcal{E}$ *has occurred is given by*

$$Prob\{\mathcal{E}_i|\mathcal{E}\} = \frac{Prob\{\mathcal{E}|\mathcal{E}_i\}\,Prob\{\mathcal{E}_i\}}{\sum_{j=1}^{k} Prob\{\mathcal{E}|\mathcal{E}_j\}\,Prob\{\mathcal{E}_j\}} = \frac{Prob\{\mathcal{E}, \mathcal{E}_i\}}{Prob\{\mathcal{E}\}} \qquad i = 1, \ldots, k \tag{2.1.13}$$

**Example**

Suppose that $k = 2$ and

- $\mathcal{E}_1$ is the event: "Tomorrow is going to rain".
- $\mathcal{E}_2$ is the event: "Tomorrow is not going to rain".
- $\mathcal{E}$ is the event: "Tonight is chilly and windy".

The knowledge of Prob $\{\mathcal{E}_1\}$, Prob $\{\mathcal{E}_2\}$ and Prob $\{\mathcal{E}|\mathcal{E}_k\}$, $k = 1, 2$ makes possible the computation of Prob $\{\mathcal{E}_k|\mathcal{E}\}$.

•

### 2.1.8 Array of joint/marginal probabilities

Let us consider the combination of two random experiments whose sample spaces are $\Omega^A = \{A_1, \cdots, A_n\}$ and $\Omega^B = \{B_1, \cdots, B_m\}$ respectively. Assume that for each pairs of events $(A_i, B_j)$, $i = 1, \ldots, n$, $j = 1, \ldots, m$ we know the joint probability value Prob $\{A_i, B_j\}$. The joint probability array contains all the necessary information for computing **all** marginal and conditional probabilities by means of expressions (2.1.12) and (2.1.8).

| | $B_1$ | $B_2$ | $\cdots$ | $B_m$ | Marginal |
|---|---|---|---|---|---|
| $A_1$ | Prob $\{A_1, B_1\}$ | Prob $\{A_1, B_2\}$ | $\cdots$ | Prob $\{A_1, B_m\}$ | Prob $\{A_1\}$ |
| $A_2$ | Prob $\{A_2, B_1\}$ | Prob $\{A_2, B_2\}$ | $\cdots$ | Prob $\{A_1, B_m\}$ | Prob $\{A_2\}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $A_n$ | Prob $\{A_n, B_1\}$ | Prob $\{A_n, B_2\}$ | $\cdots$ | Prob $\{A_n, B_m\}$ | Prob $\{A_n\}$ |
| Marginal | Prob $\{B_1\}$ | Prob $\{B_2\}$ | $\cdots$ | Prob $\{B_m\}$ | Sum=1 |

where Prob $\{A_i\} = \sum_{j=1,\ldots,m}$ Prob $\{A_i, B_j\}$ and Prob $\{B_j\} = \sum_{i=1,\ldots,n}$ Prob $\{A_i, B_j\}$. Using an entry of the joint probability matrix and the sum of the corresponding row/column, we may use expression (2.1.8) to compute the conditional probability as shown in the following example.

**Example: dependent/independent variables**

Let us model the commute time to go back home for an ULB student living in St. Gilles as a random experiment. Suppose that its sample space is $\Omega$={LOW, MEDIUM, HIGH}. Consider also an (extremely:-) random experiment representing the weather in Brussels, whose sample space is $\Omega$={G=GOOD, B=BAD}. Suppose that the array of joint probabilities is

| | G (in Bxl) | B (in Bxl) | Marginal |
|---|---|---|---|
| LOW | 0.15 | 0.05 | Prob {LOW} = 0.2 |
| MEDIUM | 0.1 | 0.4 | Prob {MEDIUM} = 0.5 |
| HIGH | 0.05 | 0.25 | Prob {HIGH} = 0.3 |
| | Prob {G} = 0.3 | Prob {B} = 0.7 | Sum=1 |

According to the above probability function, is the commute time dependent on the weather in Bxl? Note that if weather is good

| | LOW | MEDIUM | HIGH |
|---|---|---|---|
| Prob {·\|G} | 0.15/0.3=0.5 | 0.1/0.3=0.33 | 0.05/0.3=0.16 |

Else if weather is bad

| | LOW | MEDIUM | HIGH |
|---|---|---|---|
| Prob {·\|B} | 0.05/0.7=0.07 | 0.4/0.7=0.57 | 0.25/0.7=0.35 |

Since Prob {·|G} $\neq$ Prob {·|B}, i.e. the probability of having a certain commute time changes according to the value of the weather, the relation (2.1.11) is not satisfied.

Consider now the dependency between an event representing the commute time and an event describing the weather in Rome.

| | G (in Rome) | B (in Rome) | Marginal |
|---|---|---|---|
| LOW | 0.18 | 0.02 | Prob {LOW} = 0.2 |
| MEDIUM | 0.45 | 0.05 | Prob {MEDIUM} = 0.5 |
| HIGH | 0.27 | 0.03 | Prob {HIGH} = 0.3 |
| | Prob {G} = 0.9 | Prob {B} = 0.1 | Sum=1 |

Our question now is: is the commute time dependent on the weather in Rome?

If the weather in Rome is good we obtain

| | LOW | MEDIUM | HIGH |
|---|---|---|---|
| Prob {·\|G} | 0.18/0.9=0.2 | 0.45/0.9=0.5 | 0.27/0.9=0.3 |

while if the weather in Rome is bad

| | LOW | MEDIUM | HIGH |
|---|---|---|---|
| Prob {·\|B} | 0.02/0.1=0.2 | 0.05/0.1=0.5 | 0.03/0.1=0.3 |

Note that the probability of a commute time event does NOT change according to the value of the weather in Rome, e.g. Prob {LOW|B} = Prob {LOW}.

Try to answer now the following question: if you would like to predict the commute time in Brussels, which event would return more information on it: the weather in Rome or in Brussels?

•

| $\mathcal{E}_1$ | $\mathcal{E}_2$ | $\mathcal{E}_3$ | P($\mathcal{E}_1$,$\mathcal{E}_2$,$\mathcal{E}_3$) |
|---|---|---|---|
| CLEAR | RISING | DRY | 0.4 |
| CLEAR | RISING | WET | 0.07 |
| CLEAR | FALLING | DRY | 0.08 |
| CLEAR | FALLING | WET | 0.10 |
| CLOUDY | RISING | DRY | 0.09 |
| CLOUDY | RISING | WET | 0.11 |
| CLOUDY | FALLING | DRY | 0.03 |
| CLOUDY | FALLING | WET | 0.12 |

**Example: Marginal/conditional probability function**

Consider a probabilistic model of the day's weather based on the combination of the following random descriptors where

1. the first represents the sky condition and its sample space is $\Omega$ ={CLEAR, CLOUDY}.
2. the second represents the barometer trend and its sample space is $\Omega$ ={RISING, FALLING}.
3. the third represents the humidity in the afternoon and its sample space is $\Omega$ ={DRY,WET}.

Let the joint probability values be given by the table

From the joint values we can calculate the probabilities $P(CLEAR, RISING) = 0.47$ and $P(CLOUDY) = 0.35$ and the conditional probability value

$$\begin{aligned} & P(DRY|CLEAR, RISING) \\ &= \frac{P(DRY, CLEAR, RISING)}{P(CLEAR, RISING)} \\ &= \frac{0.40}{0.47} \approx 0.85 \end{aligned}$$

•

## 2.2 Random variables

Machine learning and statistics is concerned with data. What is then the link between the notion of random experiments and data? The answer is provided by the concept of random variable.

Consider a random experiment and the associated triple $(\Omega, \{\mathcal{E}\}, \text{Prob}\{\cdot\})$. Suppose that we have a mapping rule $\Omega \to \mathcal{Z} \subset \mathbb{R}$ such that we can associate with each experimental outcome $\omega$ a real value $z(\omega)$ in the domain $\mathcal{Z}$. We say that $z$ is the value taken by the random variable $\mathbf{z}$ when the outcome of the random experiment is $\omega$. Henceforth, in order to clarify the distinction between a random variable and its value, we will use the boldface notation for denoting a random variable (as in $\mathbf{z}$) and the normal face notation for the eventually observed value (as in $z = 11$).

Since there is a probability associated to each event $\mathcal{E}$ and we have a mapping from events to real values, a probability distribution can be associated to $\mathbf{z}$.

**Definition 2.1** (Random variable)**.** Given a random experiment $(\Omega, \{\mathcal{E}\}, \text{Prob}\{\cdot\})$, a random variable $\mathbf{z}$ is the result of a mapping $\Omega \to \mathcal{Z}$ that assigns a number $z$ to every outcome $\omega$. This mapping must satisfy the following two conditions:

- the set $\{\mathbf{z} \leq z\}$ is an event for every $z$.
- the probabilities

$$\text{Prob}\{\mathbf{z} = \infty\} = 0 \qquad \text{Prob}\{\mathbf{z} = -\infty\} = 0$$

Given a random variable $\mathbf{z} \in \mathcal{Z}$ and a subset $I \subset \mathcal{Z}$ we define the inverse mapping

$$z^{-1}(I) = \{\omega \in \Omega | z(\omega) \in I\} \tag{2.2.14}$$

where $z^{-1}(I) \in \{\mathcal{E}\}$ is an event. On the basis of the above relation we can associate a probability measure to $\mathbf{z}$ according to

$$\text{Prob}\{\mathbf{z} \in I\} = \text{Prob}\left\{z^{-1}(I)\right\} = \text{Prob}\{\omega \in \Omega | z(\omega) \in I\} \tag{2.2.15}$$

$$\text{Prob}\{\mathbf{z} = z\} = \text{Prob}\left\{z^{-1}(x)\right\} = \text{Prob}\{\omega \in \Omega | z(\omega) = z\} \tag{2.2.16}$$

In other words, a *random variable* is a numerical quantity, linked to some experiment involving some degree of randomness, which takes its value from some set $\mathcal{Z}$ of possible real values. For example the experiment might be the rolling of two six-sided dice and the r.v. $\mathbf{z}$ might be the sum (or the maximum) of the two numbers showing in the dice. In this case, the set of possible values is $\mathcal{Z} = \{2, \ldots, 12\}$ (or $\mathcal{Z} = \{1, \ldots, 6\}$ ).

**Example**

Suppose that we want to decide when to leave ULB to go home for watching Fiorentina playing the Champion's League final match. In order to take a decision, a quantity of interest is the (random) commute time $\mathbf{z}$ for getting from ULB to home. Our personal experience is that this time is a positive number which is not constant: for example $z_1 = 10$ minutes, $z_2 = 23$ minutes, $z_3 = 17$ minutes, where $z_i$ is the the time taken on the $i$th day of the week. The variability of this quantity is related to a complex random process with a large sample space $\Omega$ (depending for example on the weather condition, the weekday, the sport events in town, and so on). The probabilistic approach proposes to use a random variable to represent this uncertainty and to assume each measure $z_i$ to be the realization of a random outcome $\omega_i$ and the result of a mapping $z_i = z(\omega_i)$. The use of a random variable $\mathbf{z}$ to represent our commute time becomes then, a **compact (and approximate)** way of modelling the disparate set of causes underlying the uncertainty of this phenomenon. For example, this representation will allow us to compute when to leave ULB if we want the probability of missing the beginning of the match to be less than 5 percent.

•

## 2.3 Discrete random variables

The *probability (mass) function* of a discrete r.v. $\mathbf{z}$ is the combination of

1. the discrete set $\mathcal{Z}$ of values that the r.v. can take (also called *range*)
2. the set of probabilities associated to each value of $\mathcal{Z}$.

This means that we can attach to the random variable some specific mathematical function $P_{\mathbf{z}}(z)$ that gives for each $z \in \mathcal{Z}$ the probability that $\mathbf{z}$ assumes the value $z$

$$P_{\mathbf{z}}(z) = \text{Prob}\{\mathbf{z} = z\} \tag{2.3.17}$$

This function is called *probability function* or *probability mass function*.

As depicted in the following example, the probability function can be tabulated for a few sample values of $\mathbf{z}$. If we toss a fair coin twice, and the random variable $\mathbf{z}$ is the number of heads that eventually turn up, the probability function can be tabulated as follows

| Values of the random variable $\mathbf{z}$ | 0 | 1 | 2 |
|---|---|---|---|
| Associated probabilities | 0.25 | 0.50 | 0.25 |

### 2.3.1 Parametric probability function

Sometimes the probability function is not precisely known but can be expressed as a function of $z$ and a quantity $\theta$. An example is the discrete r.v. $\mathbf{z}$ that takes its value from $\mathcal{Z} = \{1, 2, 3\}$ and whose probability function is

$$P_{\mathbf{z}}(z) = \frac{\theta^{2z}}{\theta^2 + \theta^4 + \theta^6}$$

where $\theta$ is some fixed nonzero real number.

Whatever the value of $\theta$, $P_{\mathbf{z}}(z) > 0$ for $z = 1, 2, 3$ and $P_{\mathbf{z}}(1) + P_{\mathbf{z}}(2) + P_{\mathbf{z}}(3) = 1$. Therefore $\mathbf{z}$ is a well-defined random variable, even if the value of $\theta$ is unknown. We call $\theta$ a **parameter**, that is some constant, usually unknown, involved in the analytical expression of a probability function. We will see in the following that the parametric form is a convenient way to formalize a family of probabilistic models and that the problem of estimation can be seen as a parameter identification task.

### 2.3.2 Expected value, variance and standard deviation of a discrete r.v.

Though the probability function $P_{\mathbf{z}}$ provides a complete description of the uncertainty of $\mathbf{z}$, it is often not practical to use since it demands the definition of a set of values equal to the size of $\mathcal{Z}$. Therefore, it is often more convenient to deal with some compact representation of $P_{\mathbf{z}}$ obtained by computing a functional (i.e. a function of a function) of $P_{\mathbf{z}}$.

The most common single-number summary of the distribution $P_{\mathbf{z}}$ is the expected value which is a measure of central tendency.

**Definition 3.1** (Expected value)**.** The expected value of a discrete random variable $\mathbf{z}$ is defined to be

$$E[\mathbf{z}] = \mu = \sum_{z \in \mathcal{Z}} z P_{\mathbf{z}}(z) \tag{2.3.18}$$

assuming that the sum is well-defined.

Note that the expected value is not necessarily a value that belongs to the domain $\mathcal{Z}$ of the random variable. It is important also to remark that while the term *mean* is used as a synonym of *expected value*, this is not the case for the term *average*. We will add more on this difference in Section 3.3.2.

The concept of expected value was first introduced in the 17th century by C. Huygens in order to study the games of chance.

#### Example [113]

Let us consider a European roulette player who places a 1 $ bet on a single number where the roulette uses the numbers $0, 1, \ldots, 36$ and the number 0 is considered as winning for the house. The gain of the player is a random variable $\mathbf{z}$ whose sample

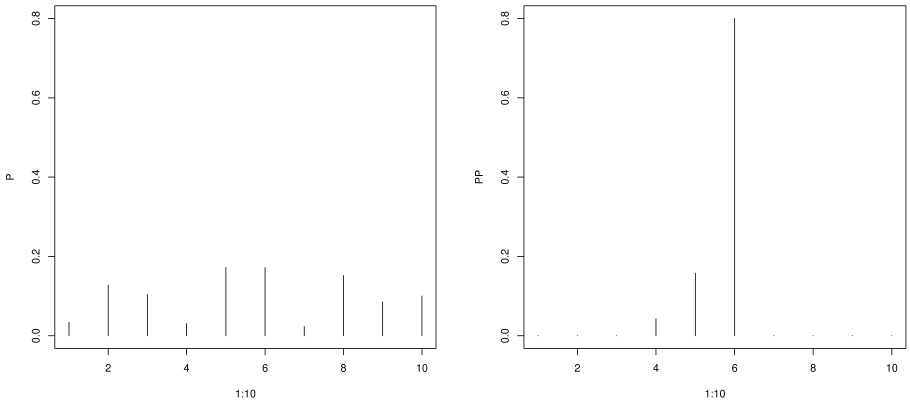



Figure 2.2: Two discrete probability functions with the same mean and different variance

space is $\mathcal{Z} = \{-1, 35\}$. In other words only two outcomes are possible: either the player wins $z_1 = -1\$$ (i.e. loses 1\$) with probability $p_1 = 36/37$ or he wins $z_2 = 35\$$ with probability $p_2 = 1/37$. The expected gain is then

$$E[\mathbf{z}] = p_1 z_1 + p_2 z_2 = p_1 * (-1) + p_2 * 35 = -36/37 + 35/37 = -1/37 = -0.027$$

In other words while casinos gain on average 2.7 cents for every staked dollar, players on average are giving away 2.7 cents (whatever sophisticated their betting strategy is).

•

A common way to summarize the spread of a distribution is provided by the variance.

**Definition 3.2** (Variance)**.** The variance of a discrete random variable $\mathbf{z}$ is defined as

$$\begin{aligned} \text{Var}[\mathbf{z}] &= \sigma^2 && (2.3.19)\\ &= E[(\mathbf{z} - E[\mathbf{z}])^2] && (2.3.20)\\ &= \sum_{z \in \mathcal{Z}} (z - E[\mathbf{z}])^2 P_{\mathbf{z}}(z) && (2.3.21) \end{aligned}$$

The variance is a measure of the dispersion of the probability function of the random variable around its mean.

Note that the following relation holds

$$\begin{aligned} \sigma^2 &= E[(\mathbf{z} - E[\mathbf{z}])^2] && (2.3.22)\\ &= E[\mathbf{z}^2 - 2\mathbf{z}E[\mathbf{z}] + (E[\mathbf{z}])^2] && (2.3.23)\\ &= E[\mathbf{z}^2] - (E[\mathbf{z}])^2 && (2.3.24)\\ &= E[\mathbf{z}^2] - \mu^2 && (2.3.25) \end{aligned}$$

whatever is the probability function of $\mathbf{z}$. Figure 2.2 illustrate two example discrete r.v. probability functions which have the same mean but different variance.

Note that an alternative measure of spread could be represented by $E[|\mathbf{z} - \mu|]$. However this quantity is much more difficult to be analytically manipulated than the variance.

The variance $\mathrm{Var}[\mathbf{z}]$ does not have the same dimension as the values of $\mathbf{z}$. For instance if $\mathbf{z}$ is measured in $m$, $\mathrm{Var}[\mathbf{z}]$ is expressed in $m^2$. A measure for the spread that has the same dimension as the r.v. $\mathbf{z}$ is the standard deviation.

**Definition 3.3** (Standard deviation)**.** The standard deviation of a discrete random variable $\mathbf{z}$ is defined as the positive square root of the variance.

$$\mathrm{Std}[\mathbf{z}] = \sqrt{\mathrm{Var}[\mathbf{z}]} = \sigma$$

**Example**

Let us consider a binary random variable $\mathbf{z} \in \mathcal{Z} = \{0, 1\}$ where $P_{\mathbf{z}}(1) = p$ and $P_{\mathbf{z}}(0) = 1 - p$. In this case

$$E[\mathbf{z}] = p * 1 + 0 * (1 - p) = p \tag{2.3.26}$$
$$E[\mathbf{z}^2] = p * 1 + 0 * (1 - p) = p \tag{2.3.27}$$
$$\mathrm{Var}[\mathbf{z}] = E[\mathbf{z}^2] - (E[\mathbf{z}])^2 \tag{2.3.28}$$
$$= p - p^2 = p(1 - p) \tag{2.3.29}$$

•

### 2.3.3 Moments of a discrete r.v.

**Definition 3.4** (Moment)**.** For any positive integer $r$, the $r$th moment of the probability function is

$$\mu_r = E[\mathbf{z}^r] = \sum_{z \in \mathcal{Z}} z^r P_{\mathbf{z}}(z) \tag{2.3.30}$$

Note that the first moment coincides with the mean $\mu$, while the second moment is related to the variance according to Equation (2.3.22). Higher-order moments provide additional information, other than the mean and the spread, about the shape of the probability function.

**Definition 3.5** (Skewness)**.** The skewness of a discrete random variable $\mathbf{z}$ is defined as

$$\gamma = \frac{E[(\mathbf{z} - \mu)^3]}{\sigma^3} \tag{2.3.31}$$

Skewness is a parameter that describes asymmetry in a random variable's probability function. Probability functions with positive skewness have long tails to the right, and functions with negative skewness have long tails to the left (Figure 2.3).

### 2.3.4 Entropy and relative entropy

**Definition 3.6** (Entropy)**.** Given a discrete r.v. $\mathbf{z}$, the *entropy* of the probability function $P_{\mathbf{z}}(z)$ is defined by

$$H(\mathbf{z}) = -\sum_{z \in \mathcal{Z}} P_{\mathbf{z}}(z) \log P_{\mathbf{z}}(z)$$

$H(\mathbf{z})$ is a measure of the unpredictability of the r.v. $\mathbf{z}$. Suppose that there are $M$ possible values for the r.v. $\mathbf{z}$. The entropy is maximized (and takes the value $\log M$) if $P_{\mathbf{z}}(z) = 1/M$ for all $z$. It is minimized iff $P(z) = 1$ for a value of $\mathbf{z}$ (i.e. all others probability values are null).

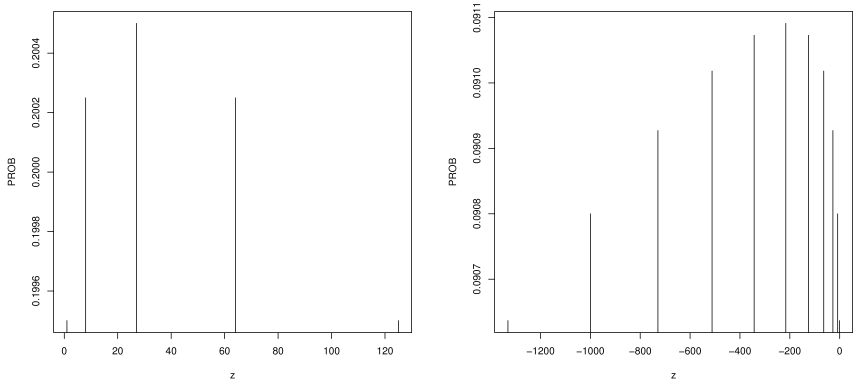



Figure 2.3: A discrete probability function with positive skewness (left) and one with a negative skewness (right).

Although entropy measures as well as variance the uncertainty of a r.v., it differs from the variance since it depends only on the probabilities of the different values and not on the values themselves. In other terms, $H$ can be thought as a function of the probability function $P_{\mathbf{z}}$ rather than of $\mathbf{z}$.

Let us now consider two different discrete probability functions on the same set of values

$$P_0 = P_{\mathbf{z}_0}(z), \qquad P_1 = P_{\mathbf{z}_1}(z)$$

where $P_0(z) > 0$ if and only if $P_1(z) > 0$.

The *relative entropies* (or the *Kullback-Leibler divergences*) associated with these two functions are

$$H(P_0||P_1) = \sum_z P_0(z) \log \frac{P_0(z)}{P_1(z)}, \qquad H(P_1||P_0) = \sum_z P_1(z) \log \frac{P_1(z)}{P_0(z)}$$

These asymmetric quantities measure the dissimilarity between the two functions. A symmetric formulation of the dissimilarity is provided by the *divergence* quantity

$$J(P_0, P_1) = H(P_0||P_1) + H(P_1||P_0).$$

## 2.4 Continuous random variable

An r.v. $\mathbf{z}$ is said to be a *continuous random variable* if it can assume any of the infinite values within a range of real numbers. The following quantities can be defined:

**Definition 4.1** (Cumulative distribution function)**.** The *(cumulative) distribution function* of $\mathbf{z}$ is the function

$$F_{\mathbf{z}}(z) = \text{Prob}\{\mathbf{z} \leq z\} \tag{2.4.32}$$

This function satisfies the following two conditions:

1. it is non-decreasing: $z_1 < z_2$ implies $F_{\mathbf{z}}(z_1) \leq F_{\mathbf{z}}(z_2)$.
2. is is normalized, i.e

$$\lim_{z \to -\infty} F_{\mathbf{z}} = 0, \qquad \lim_{z \to \infty} F_{\mathbf{z}} = 1$$

**Definition 4.2** (Density function)**.** The *density function* of a real random variable $\mathbf{z}$ is the derivative of the distribution function

$$p_{\mathbf{z}}(z) = \frac{dF_{\mathbf{z}}(z)}{dz} \tag{2.4.33}$$

at all points where $F_{\mathbf{z}}(\cdot)$ is differentiable.

Probabilities of continuous r.v. are not allocated to specific values but rather to interval of values. Specifically

$$\text{Prob}\{a \leq z \leq b\} = \int_a^b p_{\mathbf{z}}(z)dz, \qquad \int_{\mathcal{Z}} p_{\mathbf{z}}(z)dz = 1$$

Some considerations about continuous r.v. are worthy to be mentioned:

- the quantity $\text{Prob}\{\mathbf{z} = z\} = 0$ for all $z$.
- the quantity $p_{\mathbf{z}}(z)$ can be bigger than one and also unbounded.
- two r.v.s $\mathbf{z}$ and $\mathbf{x}$ are equal in distribution if $F_{\mathbf{z}}(z) = F_{\mathbf{x}}(z)$ for all $z$.

### 2.4.1 Mean, variance, moments of a continuous r.v.

Consider a continuous scalar r.v. whose range $\mathcal{Z} = (l, h)$ and density function $p(z)$. We have the following definitions

**Definition 4.3** (Expectation or mean)**.** The mean of a continuous scalar r.v. $\mathbf{z}$ is the scalar value

$$\mu = E[\mathbf{z}] = \int_l^h zp(z)dz \tag{2.4.34}$$

**Definition 4.4** (Variance)**.** The variance of a continuous scalar r.v. $\mathbf{z}$ is the scalar value

$$\sigma^2 = E[(\mathbf{z} - \mu)^2] = \int_l^h (z - \mu)^2 p(z)dz \tag{2.4.35}$$

**Definition 4.5** (Moments)**.** The r-th moment of a continuous scalar r.v. $\mathbf{z}$ is the scalar value

$$\mu_r = E[\mathbf{z}^r] = \int_l^h z^r p(z)dz \tag{2.4.36}$$

Note that the moment of order $r = 1$ coincides with the *mean* of $\mathbf{z}$.

**Definition 4.6** (Upper critical point)**.** For a given $0 \leq \alpha \leq 1$ the *upper critical point* of a continuous r.v. $\mathbf{z}$ is the number $z_\alpha$ such that

$$1 - \alpha = \text{Prob}\{\mathbf{z} \leq z_\alpha\} = F(z_\alpha) \Leftrightarrow z_\alpha = F^{-1}(1 - \alpha)$$

Figure 2.4 shows an example of cumulative distribution together with the upper critical point.

## 2.5 Joint probability

So far, we considered scalar random variables only. However, the most interesting probabilistic applications are multivariate, i.e. they concern a number of variables bigger than one.

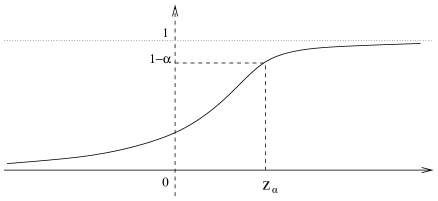



Figure 2.4: Cumulative distribution function and upper critical point.

Let us consider a probabilistic model described by $n$ discrete random variables. A fully-specified probabilistic model gives the *joint probability* for every combination of the values of the $n$ r.v..

In the discrete case the model is specified by the values of the probabilities

$$\text{Prob}\{\mathbf{z}_1 = z_1, \mathbf{z}_2 = z_2, \ldots, \mathbf{z}_n = z_n\} = P(z_1, z_2, \ldots, z_n) \tag{2.5.37}$$

for every possible assignment of values $z_1, \ldots, z_n$ to the variables.

**Spam mail example**

Let us consider a bivariate probabilistic model describing the relation between the validity of a received email and the presence of the word *Viagra* in the text. Let $\mathbf{z}_1$ be the random variable describing the validity of the email ($\mathbf{z}_1 = 0$ for no-spam and $\mathbf{z}_1 = 1$ for spam) and $\mathbf{z}_2$ the r.v. describing the presence ($\mathbf{z}_2 = 1$) or the absence ($\mathbf{z}_2 = 0$) of the word *Viagra*. The stochastic relationship between these two variables can be defined by the joint probability distribution

| | $\mathbf{z}_1 = 0$ | $\mathbf{z}_1 = 1$ | $P_{\mathbf{z}_2}$ |
|---|---|---|---|
| $\mathbf{z}_2 = 0$ | 0.8 | 0.08 | 0.88 |
| $\mathbf{z}_2 = 1$ | 0.01 | 0.11 | 0.12 |
| $P_{\mathbf{z}_1}$ | 0.81 | 0.19 | 1 |

•

In the case of $n$ continuous random variables, the model is specified by the joint distribution function

$$\text{Prob}\{\mathbf{z}_1 \leq z_1, \mathbf{z}_2 \leq z_2, \ldots, \mathbf{z}_n \leq z_n\} = F(z_1, z_2, \ldots, z_n)$$

which returns a value for every possible assignment of values $z_1, \ldots, z_n$ to the variables.

Having defined the joint probability, we can now go on to define the case when two random variables are independent.

**Definition 5.1** (Independent variables)**.** Let $\mathbf{x}$ and $\mathbf{y}$ be two discrete random variables. Two variables $\mathbf{x}$ and $\mathbf{y}$ are defined to be *statistically independent* (written as $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$) if the joint probability

$$\text{Prob}\{\mathbf{x} = x, \mathbf{y} = y\} = \text{Prob}\{\mathbf{x} = x\}\,\text{Prob}\{\mathbf{y} = y\} \tag{2.5.38}$$

The definition can be easily extended to the continuous case. In qualitative terms, this means that we do not expect that the observed outcome of one variable will affect the other. Note that if $\mathbf{x}$ and $\mathbf{y}$ are independent and $\mathbf{y}$ and $\mathbf{z}$ are independent, then $\mathbf{x}$ and $\mathbf{z}$ need not be independent. In other words, the relation of independence is not a transitive relation.

**Exercise**

Check whether the variable $\mathbf{z}_1$ and $\mathbf{z}_2$ of the spam mail example are independent.

•

### 2.5.1 Marginal and conditional probability

Let $\{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ be a subset of size $m$ of the $n$ discrete r.v. for which a joint probability function is defined (see (2.5.37)). The *marginal probabilities* for the subset can be derived from expression (2.5.37) by summing over all possible combinations of values for the remaining variables.

$$P(z_1, \ldots, z_m) = \sum_{\tilde{z}_{m+1}} \cdots \sum_{\tilde{z}_n} P(z_1, \ldots, z_m, \tilde{z}_{m+1}, \ldots, \tilde{z}_n)$$

**Exercise**

Compute the marginal probabilities $P(\mathbf{z}_1 = 0)$ and $P(\mathbf{z}_1 = 1)$ from the joint probability of the spam mail example.

•

For continuous random variables the marginal density is

$$p(z_1, \ldots, z_m) = \int p(z_1, \ldots, z_m, z_{m+1}, \ldots, z_n) dz_{m+1} \ldots dz_n \tag{2.5.39}$$

The following definition for r.v. derives directly from Equation (2.1.8).

**Definition 5.2** (Conditional probability function)**.** The *conditional probability function* for one subset of discrete variables $\{\mathbf{z}_i : i \in S_1\}$ given values for another disjoint subset $\{\mathbf{z}_j : j \in S_2\}$ where $S_1 \cap S_2 = \emptyset$, is defined as the ratio

$$P(\{\mathbf{z}_i : i \in S_1\} | \{\mathbf{z}_j : j \in S_2\}) = \frac{P(\{\mathbf{z}_i : i \in S_1\}, \{\mathbf{z}_j : j \in S_2\})}{P(\{\mathbf{z}_j : j \in S_2\})}$$

Note that if $\mathbf{x}$ and $\mathbf{y}$ are independent then

$$P_{\mathbf{x}}(x | \mathbf{y} = y) = P_{\mathbf{x}}(x) \tag{2.5.40}$$

Also, since independence is symmetric, this is equivalent to say that $P_{\mathbf{y}}(y | \mathbf{x} = x) = P_{\mathbf{y}}(y)$.

**Definition 5.3** (Conditional density function)**.** The *conditional density function* for one subset of continuous variables $\{\mathbf{z}_i : i \in S_1\}$ given values for another disjoint subset $\{\mathbf{z}_j : j \in S_2\}$ where $S_1 \cap S_2 = \emptyset$, is defined as the ratio

$$p(\{\mathbf{z}_i : i \in S_1\} | \{\mathbf{z}_j : j \in S_2\}) = \frac{p(\{\mathbf{z}_i : i \in S_1\}, \{\mathbf{z}_j : j \in S_2\})}{p(\{\mathbf{z}_j : j \in S_2\})}$$

where $p(\{\mathbf{z}_j : j \in S_2\})$ is the marginal density of the set $S_2$ of variables.

### 2.5.2 Entropy in the continuous case

Consider a continuous r.v. $\mathbf{y}$. The *(differential) entropy* of $\mathbf{y}$ is defined by

$$H(\mathbf{y}) = -\int \log(p(y))p(y)dy = E_{\mathbf{y}}[-\log(p(y))] = E_{\mathbf{y}}\left[\log \frac{1}{p(y)}\right]$$

with the convention that $0 \log 0 = 0$.

Entropy if a functional of the distribution of $\mathbf{y}$and is a measure of the predictability of a r.v. $\mathbf{y}$. The higher the entropy, the less reliable are our predictions about $\mathbf{y}$.

For a scalar normal r.v. $\mathbf{y} \sim \mathcal{N}(0, \sigma^2)$

$$H(\mathbf{y}) = \frac{1}{2}\left(1 + \ln 2\pi\sigma^2\right) = \frac{1}{2}\left(\ln 2\pi e\sigma^2\right)$$

In the case of a normal random vector $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \sim \mathcal{N}(0, \Sigma)$

$$H(\mathbf{Y}) = \frac{1}{2}\left(\ln(2\pi e)^n \det(\Sigma)\right)$$

#### 2.5.2.1 Joint and conditional entropy

Consider two continuous r.v.s $\mathbf{x}$ and $\mathbf{y}$ and their joint density $p(x, y)$. The *joint entropy* of $\mathbf{x}$ and $\mathbf{y}$ is defined by

$$H(\mathbf{x}, \mathbf{y}) = -\int\int \log(p(x, y))p(x, y)dxdy = \\ = E_{\mathbf{x},\mathbf{y}}[-\log(p(x, y))] = E_{\mathbf{x},\mathbf{y}}\left[\log \frac{1}{p(x, y)}\right]$$

The *conditional entropy* is defined as

$$H(\mathbf{y}|\mathbf{x}) = -\int\int \log(p(y|x))p(x, y)dxdy = E_{\mathbf{x},\mathbf{y}}[-\log(p(y|x))] = \\ = E_{\mathbf{x},\mathbf{y}}\left[\log \frac{1}{p(y|x)}\right] = E_{\mathbf{x}}[H(\mathbf{y}|x)]$$

This quantity quantifies the remaining uncertainty of $\mathbf{y}$ once $\mathbf{x}$ is known.

Note that in general $H(\mathbf{y}|\mathbf{x}) \neq H(\mathbf{x}|\mathbf{y})$, $H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x}) = H(\mathbf{x}) - H(\mathbf{x}|\mathbf{y})$ and that the *chain rule* holds

$$H(\mathbf{y}, \mathbf{x}) = H(\mathbf{y}|\mathbf{x}) + H(\mathbf{x})$$

Also conditioning reduces entropy

$$H(\mathbf{y}|\mathbf{x}) \leq H(\mathbf{y})$$

with equality if $\mathbf{x}$ and $\mathbf{y}$ are independent, i.e. $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$,

Another interesting property is the *independence bound*

$$H(\mathbf{y}, \mathbf{x}) \leq H(\mathbf{y}) + H(\mathbf{x})$$

with equality if $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$.

## 2.6 Common discrete probability functions

### 2.6.1 The Bernoulli trial

A *Bernoulli trial* is a random experiment with two possible outcomes, often called "success" and "failure". The probability of success is denoted by $p$ and the probability of failure by $(1-p)$. A *Bernoulli random variable* $\mathbf{z}$ is a binary discrete r.v. associated with the Bernoulli trial. It takes $z=0$ with probability $(1-p)$ and $z=1$ with probability $p$.

The probability function of $\mathbf{z}$ can be written in the form

$$\text{Prob}\{\mathbf{z}=z\} = P_{\mathbf{z}}(z) = p^z(1-p)^{1-z}, \qquad z=0,1$$

Note that $E[\mathbf{z}]=p$ and $\text{Var}[\mathbf{z}]=p(1-p)$.

### 2.6.2 The Binomial probability function

A *binomial random variable* represents the number of successes in a *fixed number* $N$ of *independent Bernoulli* trials with the *same probability* of success for each trial. A typical example is the number $\mathbf{z}$ of heads in $N$ tosses of a coin.

The probability function of $\mathbf{z} \sim \text{Bin}(N,p)$ is given by

$$\text{Prob}\{\mathbf{z}=z\} = P_{\mathbf{z}}(z) = \binom{N}{z} p^z(1-p)^{N-z}, \qquad z=0,1,\ldots,N \tag{2.6.41}$$

The mean of the probability function is $\mu = Np$. Note that:

- the Bernoulli probability function is a special case ($N=1$) of the binomial function,
- for small $p$, the probability of having at least 1 success in $N$ trials is proportional to $N$, as long as $Np$ is small,
- if $\mathbf{z}_1 \sim \text{Bin}(N_1,p)$ and $\mathbf{z}_2 \sim \text{Bin}(N_1,p)$ are independent then $\mathbf{z}_1+\mathbf{z}_2 \sim \text{Bin}(N_1+N_2,p)$

### 2.6.3 The Geometric probability function

A r.v. $\mathbf{z}$ has a *geometric* probability function if it represents the number of successes before the first failure in a sequence of independent Bernoulli trials with probability of success $p$. Its probability function is

$$P_{\mathbf{z}}(z) = (1-p)p^z, \quad z=0,1,2,\ldots$$

The geometric probability function has an important property, known as the *memoryless* or *Markov* property. According to this property, given two integers $z_1 \geq 0$, $z_2 \geq 0$,

$$P_{\mathbf{z}}(\mathbf{z}=z_1+z_2|\mathbf{z}>z_1) = P_{\mathbf{z}}(z_2)$$

Note that it is the only discrete probability function with this property.

A r.v. $\mathbf{z}$ has a *generalized geometric* probability function if it represents the number of Bernoulli trials preceding but not including the $k+1$th failure. Its function is

$$P_{\mathbf{z}}(z) = \binom{z}{k} p^{z-k}(1-p^{k+1}), \qquad z=k,k+1,k+2,\ldots$$

where $k$ is supposed to be given. The reader may verify that these functions are, indeed, probability functions by showing that $\sum_{z\in\mathcal{Z}} P_{\mathbf{z}}(z) = 1$ by using the formula for the sum of a geometric series.

The geometric (and generalized geometric probability function) are encountered in some problems in queuing theory and risk analysis.

### 2.6.4 The Poisson probability function

A r.v. $\mathbf{z}$ has a *Poisson* probability function with parameter $\lambda > 0$ and is denoted by $\mathbf{z} \sim \text{Poisson}(\lambda)$ if

$$P_{\mathbf{z}}(z) = \frac{e^{-\lambda}\lambda^z}{z!}, \quad z = 0, 1, \ldots \tag{2.6.42}$$

The Poisson probability function is a limiting form of the binomial function. If the number of trials $N$ is large, the probability of success $p$ of each trial is small and the product $Np = \lambda$ is moderate, then the probability of $z$ successes according to the Binomial function is very close to the probability that a Poisson random variable with parameter $\lambda$ takes the value $z$. Note also that

- $\text{Prob}\{\mathbf{z} \geq 1\} = 1 - \text{Prob}\{\mathbf{z} = 0\} = 1 - e^{-\lambda}$.
- $E[\mathbf{z}] = \lambda$, $\text{Var}[\mathbf{z}] = \lambda$.
- if $\mathbf{z}_1 \sim \text{Poisson}(\lambda_1)$ and $\mathbf{z}_2 \sim \text{Poisson}(\lambda_2)$ are independent then $\mathbf{z}_1 + \mathbf{z}_2 \sim \text{Poisson}(\lambda_1 + \lambda_2)$.

**Exemple**

The French football player Trezeguet has one percent probability of missing a penalty shot. Let us compute the probability that he misses no penalty in 100 shots.

We have to compute the probability of $z = 0$ realizations of an event with probability $p = 0.01$ out of $N = 100$ trials. By using (2.6.41), we obtain:

$$\text{Prob}\{\mathbf{z} = 0\} = \binom{N}{0}(1-p)^N = \binom{100}{0}0.99^{100} = 0.366$$

Let us consider now the Poisson approximation. Since $\lambda = Np = 1$, according to (2.6.42) we have

$$\text{Prob}\{\mathbf{z} = 0\} = \frac{e^{-\lambda}\lambda^0}{0!} = e^{-1} = 0.3679$$

•

Table 2.1 summarizes the expression of mean and variance for the above-mentioned probability functions.

| Probability function | Mean | Variance |
|---|---|---|
| Bernoulli | $p$ | $p(1-p)$ |
| Binomial | $Np$ | $Np(1-p)$ |
| Geometric | $p/(1-p)$ | $p/(1-p)^2$ |
| Poisson | $\lambda$ | $\lambda$ |

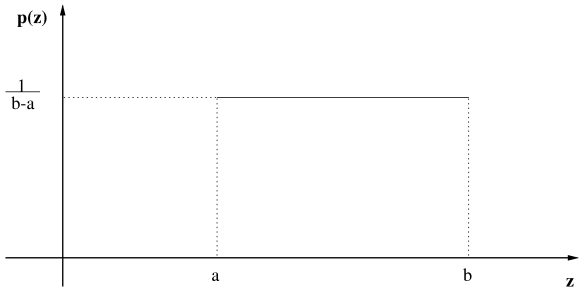
Table 2.1: Common discrete probability functions

## 2.7 Common continuous distributions

### 2.7.1 Uniform distribution

A random variable $\mathbf{z}$ is said to be uniformly distributed on the interval $(a, b)$ (written as $\mathbf{z} \sim \mathcal{U}(a, b)$) if its probability density function is given by

$$p(z) = \begin{cases} \frac{1}{b-a} & \text{if } a < z < b \\ 0, & \text{otherwise} \end{cases}$$




**Exercise**

Compute the variance of $\mathcal{U}(a, b)$.

•

### 2.7.2 Exponential distribution

A continuous random variable $\mathbf{z}$ is said to be *exponentially distributed* with rate $\lambda > 0$ (written as $\mathbf{z} \sim \mathcal{E}(\lambda)$) if its probability density function is given by

$$p_{\mathbf{z}}(z) = \begin{cases} \lambda e^{-\lambda z} & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

This distribution is commonly used to describe physical phenomena (e.g. radioactive decay time or failure time) and can be considered as a continuous approximation to the geometric distribution (Section 2.6.3). Note that

- the mean of $\mathbf{z}$ is $1/\lambda$ and its variance is $1/\lambda^2$.
- just like the geometric distribution it satisfies the *memoryless property*

$$\text{Prob}\{\mathbf{z} \geq z_1 + z_2 | \mathbf{z} = z_1\} = \text{Prob}\{\mathbf{z} \geq z_2\}$$

### 2.7.3 The Gamma distribution

We say that $\mathbf{z}$ has a *gamma distribution* with parameters $(n,\lambda)$ if its density function is

$$p_{\mathbf{z}}(z) = \begin{cases} \frac{\lambda^n z^{n-1} e^{-\lambda z}}{(n-1)!} & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$

Note that

- $E[\mathbf{z}] = n\lambda^{-1}$, $\text{Var}[\mathbf{z}] = n\lambda^{-2}$.
- it is the distribution of the sum of $n$ i.i.d.[2] r.v. having exponential distribution with rate $\lambda$.

[2] i.i.d. stands for identically and independently distributed (see also Section 3.1)

| $\alpha$ | 0.1 | 0.075 | 0.05 | 0.025 | 0.01 | 0.005 | 0.001 | 0.0005 |
|---|---|---|---|---|---|---|---|---|
| $z_\alpha$ | 1.282 | 1.440 | 1.645 | 1.967 | 2.326 | 2.576 | 3.090 | 3.291 |

Table 2.2: Upper critical points for a standard distribution

- the exponential distribution is a special case ($n = 1$) of the gamma distribution.

### 2.7.4 Normal distribution: the scalar case

A continuous scalar random variable $\mathbf{x}$ is said to be *normally distributed* with parameters $\mu$ and $\sigma^2$ (written as $\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2)$) if its probability density function is given by

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Note that

- the normal distribution is also widely known as the *Gaussian distribution*.
- the mean of $\mathbf{x}$ is $\mu$; the variance of $\mathbf{x}$ is $\sigma^2$.
- the coefficient in front of the exponential ensures that $\int_{\mathbf{X}} p(x)dx = 1$.
- the probability that an observation $x$ from a normal r.v. is within 2 standard deviations from the mean is 0.95.

If $\mu = 0$ and $\sigma^2 = 1$ the distribution is defined *standard normal*. We will denote its distribution function $F_{\mathbf{z}}(z) = \Phi(z)$. Given a normal r.v. $\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2)$, the r.v.

$$\mathbf{z} = (\mathbf{x} - \mu)/\sigma \tag{2.7.43}$$

has a standard normal distribution. Also if $\mathbf{z} \sim \mathcal{N}(0, 1)$ then

$$\mathbf{x} = \mu + \sigma\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$$

If we denote by $z_\alpha$ the upper critical points (Definition 4.6) of the standard normal density

$$1 - \alpha = \text{Prob}\{\mathbf{z} \leq z_\alpha\} = F(z_\alpha) = \Phi(z_\alpha)$$

the following relations hold

$$z_\alpha = \Phi^{-1}(1 - \alpha), \quad z_{1-\alpha} = -z_\alpha, \quad 1 - \alpha = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z_\alpha} e^{-z^2/2} dz$$

Table 2.2 lists the most commonly used values of $z_\alpha$.

In Table 2.3 we list some important relationships holding in the normal case

#### R script

The following R code can be used to test the first and the third relation in Table 2.3 by using random sampling and simulation.

```
### script norm.R ###
set.seed(0)
N<-1000000
mu<-1
sigma<-2
```

| $\mathbf{x} \in \mathcal{N}(\mu, \sigma^2)$ |
|---|
| $\text{Prob}\{\mu - \sigma \leq \mathbf{x} \leq \mu + \sigma\} = 0.683$ |
| $\text{Prob}\{\mu - 1.282\sigma \leq \mathbf{x} \leq \mu + 1.282\sigma\} = 0.8$ |
| $\text{Prob}\{\mu - 1.645\sigma \leq \mathbf{x} \leq \mu + 1.645\sigma\} = 0.9$ |
| $\text{Prob}\{\mu - 1.96\sigma \leq \mathbf{x} \leq \mu + 1.96\sigma\} = 0.95$ |
| $\text{Prob}\{\mu - 2\sigma \leq \mathbf{x} \leq \mu + 2\sigma\} = 0.954$ |
| $\text{Prob}\{\mu - 3\sigma \leq \mathbf{x} \leq \mu + 3\sigma\} = 0.997$ |

Table 2.3: Important relationships in the normal case

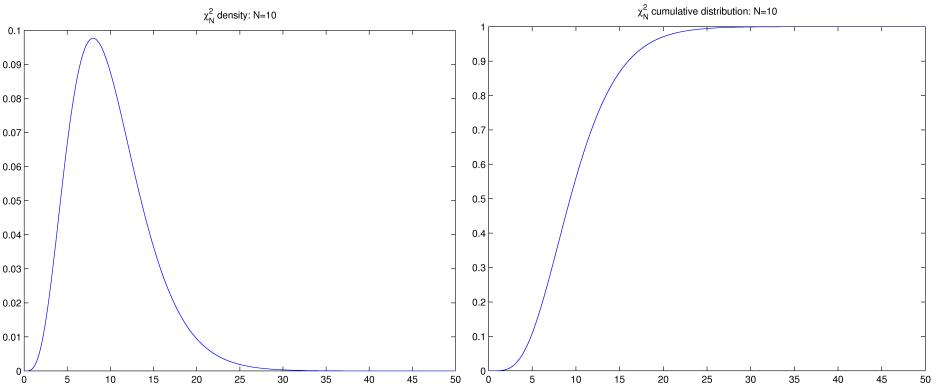



Figure 2.5: $\chi^2_N$ probability distribution ($N = 10$)

```
DN<-rnorm(N,mu,sigma)
print(sum(DN<=(mu+sigma) & DN>=(mu-sigma))/N)
#[1] 0.683213
print(sum(DN<=(mu+1.645*sigma) & DN>=(mu-1.645*sigma))/N)
#[1] 0.900165
```

Note that `rnorm` is the normal random number generator provided by the R language.

•

### 2.7.5 The chi-squared distribution

An r.v. $\mathbf{z}$ has a $\chi^2_N$ distribution if

$$\mathbf{z} = \mathbf{x}_1^2 + \cdots + \mathbf{x}_N^2$$

where $N \in \mathbb{N}$ and $\mathbf{x}_1$,$\mathbf{x}_2$,...,$\mathbf{x}_N$ are i.i.d. standard normal random variables $\mathcal{N}(0, 1)$. The distribution is called *a chi-squared distribution with $N$ degrees of freedom.* Note also that

- The probability distribution is a gamma distribution with parameters $(\frac{1}{2}N, \frac{1}{2})$.
- $E[\mathbf{z}] = N$ and $\text{Var}[\mathbf{z}] = 2N$.

The $\chi^2_N$ density and distribution function for $N = 10$ are plotted in Figure 2.5 (R script `chisq.R`).

```
### script chisq.R ###
N<-10
```

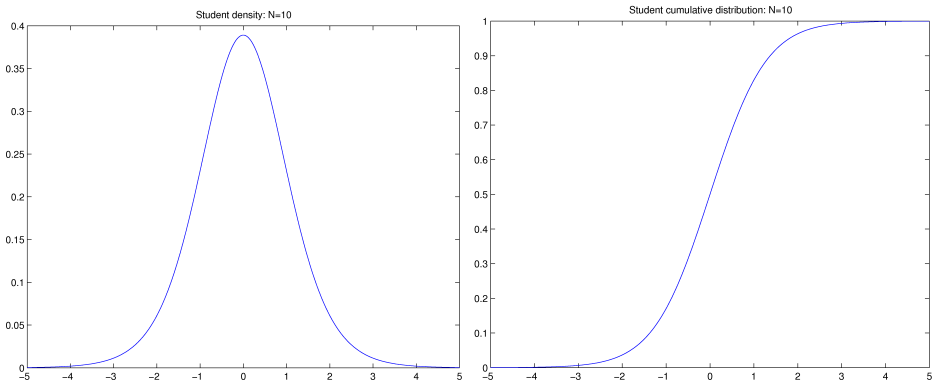



Figure 2.6: Student probability distribution ($N = 10$)

```
x<-seq(0,50,by=.1)
plot(x,dchisq(x,N),main=paste("chi-squared (N=" ,N,") density"))
plot(x,pchisq(x,N),main=paste("chi-squared (N=" ,N,") cumulative distribution"))
```

### 2.7.6 Student's $t$-distribution

If $\mathbf{x} \sim \mathcal{N}(0,1)$ and $\mathbf{y} \sim \chi^2_N$ are independent then the *Student's t-distribution* with $N$ degrees of freedom is the distribution of the r.v.

$$\mathbf{z} = \frac{\mathbf{x}}{\sqrt{\mathbf{y}/N}} \tag{2.7.44}$$

We denote this with $\mathbf{z} \sim \mathcal{T}_N$. Note that $E[\mathbf{z}] = 0$ and $Var[\mathbf{z}] = N/(N-2)$ if $N > 2$.

The Student density and distribution function for $N = 10$ are plotted in Figure 2.6.

```
# script stu.R
N<-10
x<-seq(-5,5,by=.1)
plot(x,dt(x,N),main=paste("Student (N=" ,N,") density"))
plot(x,pt(x,N),main=paste("Student (N=" ,N,") cumulative distribution"))
```

### 2.7.7 F-distribution

Let $\mathbf{x} \sim \chi^2_M$ and $\mathbf{y} \sim \chi^2_N$ be two independent r.v.. An r.v. $\mathbf{z}$ has a *F-distribution* with $M$ and $N$ degrees of freedom (written as $\mathbf{z} \sim F_{M,N}$) if

$$\mathbf{z} = \frac{\mathbf{x}/M}{\mathbf{y}/N} \tag{2.7.45}$$

Note that if $\mathbf{z} \sim F_{M,N}$ then $1/\mathbf{z} \sim F_{N,M}$, while if $\mathbf{z} \sim \mathcal{T}_N$ then $\mathbf{z}^2 \sim F_{1,N}$. The F-density and distribution function are plotted in Figure 2.7.

```
# script s_f.R
N1<-10
N2<-20
```

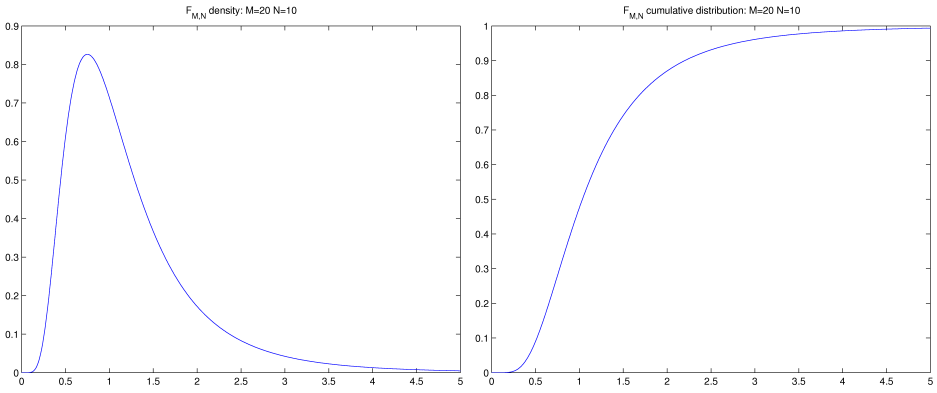



Figure 2.7: F probability distribution ($N = 10$)

```
x<-seq(-.1,5,by=.1)
plot(x,df(x,N1,N2),main=paste("F (N1=" ,N1,",N2=",N2,") density"),type="l")
plot(x,pf(x,N1,N2),main=paste("F (N1=" ,N1,",N2=",N2,") cumulative distribution"),
     type="l")
```

### 2.7.8 Bivariate continuous distribution

Let us consider two continuous r.v. $\mathbf{x}$ and $\mathbf{y}$ and their bivariate joint density function $p_{\mathbf{x},\mathbf{y}}(x, y)$. From (2.5.39), we define *marginal density* the quantity

$$p_{\mathbf{x}}(x) = \int_{-\infty}^{\infty} p_{\mathbf{x},\mathbf{y}}(x, y)dy$$

and *conditional density* the quantity

$$p_{\mathbf{y}|\mathbf{x}}(y|x) = \frac{p(x, y)}{p(x)} \tag{2.7.46}$$

which is, in loose terms, the probability that $\mathbf{y}$ belongs to an interval $dy$ about $y$ assuming that $\mathbf{x} = x$. Note that, if $\mathbf{x}$ and $\mathbf{y}$ are independent

$$p_{\mathbf{x},\mathbf{y}}(x, y) = p_{\mathbf{x}}(x)p_{\mathbf{y}}(y), \qquad p(y|x) = p_{\mathbf{y}}(y)$$

The definition of *conditional expectation* is obtained from 2.7.46 and (2.4.34).

**Definition 7.1** (Conditional expectation)**.** The conditional expectation of $\mathbf{y}$ given $\mathbf{x} = x$ is

$$E_{\mathbf{y}}[\mathbf{y}|\mathbf{x} = x] = \int yp_{\mathbf{y}|\mathbf{x}}(y|x)dy = \mu_{\mathbf{y}|\mathbf{x}}(x) \tag{2.7.47}$$

The definition of *conditional variance* derives from (2.7.46) and (2.4.35)

**Definition 7.2** (Conditional variance)**.**

$$\text{Var}\,[\mathbf{y}|\mathbf{x} = x] = \int (y - \mu_{\mathbf{y}|\mathbf{x}}(x))^2 p_{\mathbf{y}|\mathbf{x}}(y|x)dy \tag{2.7.48}$$

Note that both these quantities are function of $x$. If $\mathbf{x}$ is a random variable it follows that the conditional expectation and the conditional variance are random, too.

Some important results on their expectation are contained in the following theorems [119].

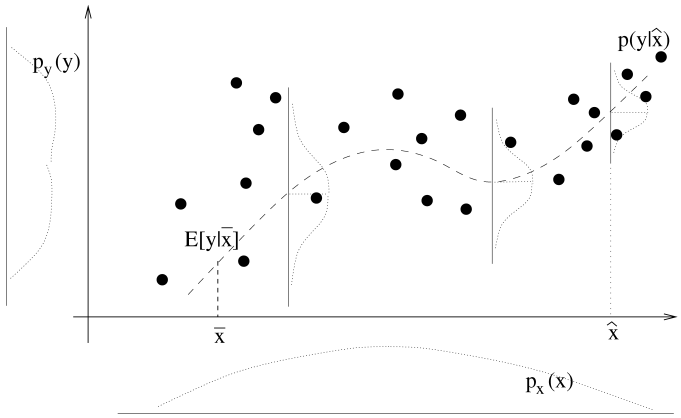



Figure 2.8: Bivariate distribution: the figure shows the two marginal distributions (beside the axis), the conditional expectation function (dashed line) and some condistional distributions (dotted).

**Theorem 7.3.** *For two r.v.* $\mathbf{x}$ *and* $\mathbf{y}$*, assuming the expectations exist, we have that*

$$E_{\mathbf{x}}[E_{\mathbf{y}}[\mathbf{y}|\mathbf{x}]] = E_{\mathbf{y}}[\mathbf{y}] \tag{2.7.49}$$

*and*

$$Var[\mathbf{y}] = E_{\mathbf{x}}[Var[\mathbf{y}|\mathbf{x}]] + Var[E_{\mathbf{y}}[\mathbf{y}|\mathbf{x}]] \tag{2.7.50}$$

*where* $Var[\mathbf{y}|\mathbf{x}]$ *and* $E_{\mathbf{y}}[\mathbf{y}|\mathbf{x}]$ *are functions of* $x$*.*

We remind that for a bivariate function $f(x, y)$

$$E_{\mathbf{y}}[f(\mathbf{x}, \mathbf{y})] = \int f(x, y) p_{\mathbf{y}}(y) dy, \quad E_{\mathbf{x}}[f(\mathbf{x}, \mathbf{y})] = \int f(x, y) p_{\mathbf{x}}(x) dx.$$

An example of bivariate continuous distribution is illustrated in Figure 2.8. It is worthy noting that, although the conditional distribution are bell-shaped this is not the case for the marginal distributions.

### 2.7.9 Correlation

Consider two random variables $\mathbf{x}$ and $\mathbf{y}$ with means $\mu_{\mathbf{x}}$ and $\mu_{\mathbf{y}}$ and standard deviations $\sigma_{\mathbf{x}}$ and $\sigma_{\mathbf{y}}$.

**Definition 7.4** (Covariance)**.** The *covariance* between $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$\text{Cov}[\mathbf{x}, \mathbf{y}] = E[(\mathbf{x} - \mu_{\mathbf{x}})(\mathbf{y} - \mu_{\mathbf{y}})] \tag{2.7.51}$$

**Definition 7.5** (Correlation)**.** The *correlation coefficient* is defined as

$$\rho(\mathbf{x}, \mathbf{y}) = \frac{\text{Cov}[\mathbf{x}, \mathbf{y}]}{\sqrt{\text{Var}\,[\mathbf{x}]\,\text{Var}\,[\mathbf{y}]}} \tag{2.7.52}$$

It is easily shown that $-1 \leq \rho(\mathbf{x}, \mathbf{y}) \leq 1$.

**Definition 7.6** (Uncorrelated variables)**.** Two r.v. are said to be *uncorrelated* if

$$E[\mathbf{xy}] = E[\mathbf{x}]E[\mathbf{y}] \tag{2.7.53}$$

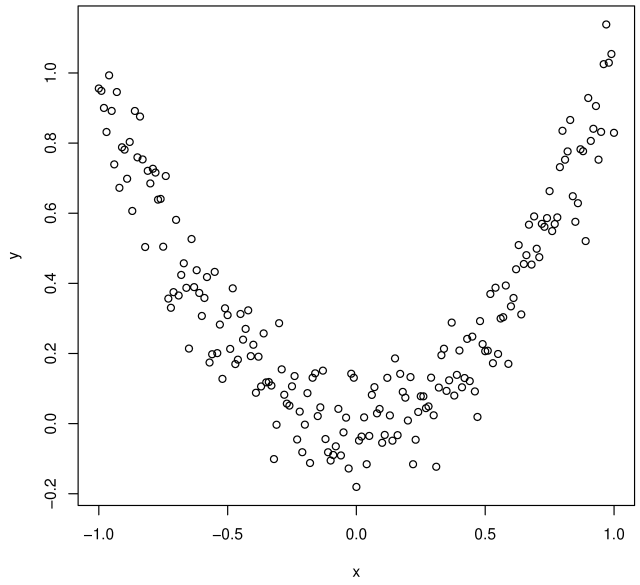



Figure 2.9: Dependent but uncorrelated random variables

Note that if $\mathbf{x}$ and $\mathbf{y}$ are two independent random variables then

$$E[\mathbf{xy}] = \int xyp(x,y)dxdy = \int xyp(x)p(y)dxdy = \int xp(x)dx \int yp(y)dy = E[\mathbf{x}]E[\mathbf{y}]$$

This means that independence implies uncorrelation. However the contrary does not hold for a generic distribution. The equivalence between independence and uncorrelation holds only if $\mathbf{x}$ and $\mathbf{y}$ are jointly Gaussian. See Figure 2.9 for an example of uncorrelated but dependent variables.

**Exercises**

- Let $\mathbf{x}$ and $\mathbf{y}$ two discrete independent r.v. such that

  $$P_{\mathbf{x}}(-1) = 0.1, \quad P_{\mathbf{x}}(0) = 0.8, \quad P_{\mathbf{x}}(1) = 0.1$$

  and

  $$P_{\mathbf{y}}(1) = 0.1, \quad P_{\mathbf{y}}(2) = 0.8, \quad P_{\mathbf{y}}(3) = 0.1$$

  If $\mathbf{z} = \mathbf{x} + \mathbf{y}$ show that $E[\mathbf{z}] = E[\mathbf{x}] + E[\mathbf{y}]$

- Let $\mathbf{x}$ be a discrete r.v. which assumes $\{-1, 0, 1\}$ with probability $1/3$ and $\mathbf{y} = \mathbf{x}^2$. Let $\mathbf{z} = \mathbf{x} + \mathbf{y}$. Show that $E[\mathbf{z}] = E[\mathbf{x}] + E[\mathbf{y}]$. Demonstrate that $\mathbf{x}$ and $\mathbf{y}$ are uncorrelated but dependent random variables.

•

### 2.7.10 Mutual information

Mutual information is one of the widely used measures to define dependency of variables. It is a measure of the amount of information that one random variable

contains about another random variable. It can also be considered as the *distance from independence* between the two variables. This quantity is always non negative and zero if and only if the two variables are stochastically independent.

Given two random variables $\mathbf{x}$ and $\mathbf{y}$, their *mutual information* is defined in terms of their probabilistic marginal density functions $p_{\mathbf{x}}(x)$, $p_{\mathbf{y}}(y)$ and the joint $p_{(\mathbf{x},\mathbf{y})}(x,y)$:

$$I(\mathbf{x};\mathbf{y}) = \int\int \log \frac{p(x,y)}{p(x)p(y)} p(x,y)dxdy = H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x}) = H(\mathbf{x}) - H(\mathbf{x}|\mathbf{y}) \tag{2.7.54}$$

with the convention that $0 \log \frac{0}{0} = 0$.

In the normal case an analytical link between correlation and mutual information exists. Let $(\mathbf{x},\mathbf{y})$ a normally distributed random vector with correlation coefficient $\rho$. The mutual information between $\mathbf{x}$ and $\mathbf{y}$ is given by

$$I(\mathbf{x};\mathbf{y}) = -\frac{1}{2}\log(1-\rho^2)$$

Equivalently the normal correlation coefficient can be written as

$$\rho = \sqrt{1 - \exp(-2I(\mathbf{x};\mathbf{y}))}$$

Note that $\rho = 0$ when $I(\mathbf{x};\mathbf{y})) = 0$ or equivalently $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$.

## 2.8 Normal distribution: the multivariate case

Let $\mathbf{z}$ be a random vector $(n \times 1)$. The vector is said to be *normally distributed* with parameters $\mu_{(n\times 1)}$ and $\Sigma_{(n\times n)}$ (also $\mathbf{z} \sim \mathcal{N}(\mu, \Sigma)$) if its probability density function is given by

$$p_{\mathbf{z}}(z) = \frac{1}{(\sqrt{2\pi})^n \sqrt{\det(\Sigma)}} \exp\left\{-\frac{1}{2}(z-\mu)^T \Sigma^{-1}(z-\mu)\right\}$$

where $\det(\Sigma)$ denotes the determinant of the matrix $\Sigma$. It follows that

- the mean $E[\mathbf{z}] = \mu$ is an $n$-dimensional vector,
- the matrix $\Sigma_{(n\times n)} = E[(\mathbf{z}-\mu)(\mathbf{z}-\mu)^T]$ is the covariance matrix. This matrix is squared and symmetric. It has $n(n+1)/2$ parameters.

The quantity

$$\Delta = (z-\mu)^T \Sigma^{-1}(z-\mu)$$

which appears in the exponent of $p_{\mathbf{z}}$ is called the *Mahalanobis distance* from $z$ to $\mu$.

It can be shown that the $n$-dimensional surfaces of constant probability density

- are hyper-ellipsoids on which $\Delta^2$ is constant;
- their *principal axes* are given by the eigenvectors $u_i$, $i = 1, \ldots, n$ of $\Sigma$ which satisfy

  $$\Sigma u_i = \lambda_i u_i \qquad i = 1, \ldots, n$$

  where $\lambda_i$ are the corresponding eigenvalues.
- the eigenvalues $\lambda_i$ give the variances along the principal directions (Figure 2.10).

If the covariance matrix $\Sigma$ is *diagonal* then

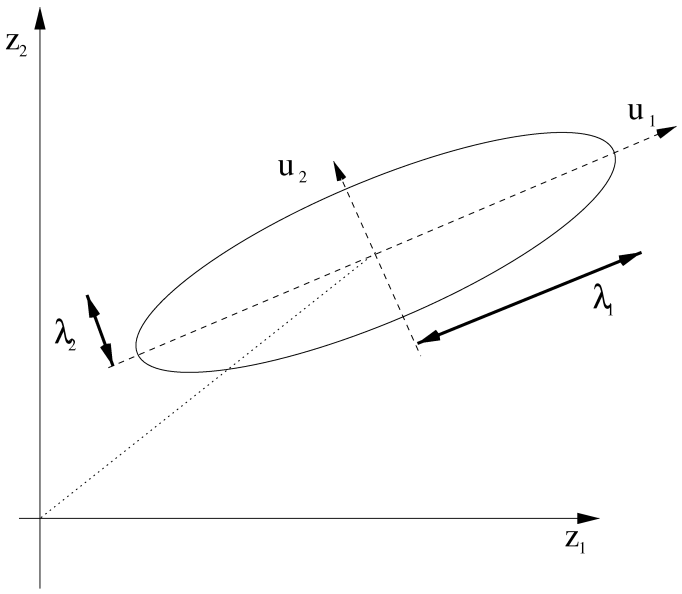



Figure 2.10: Contour curves of normal distribution for $n = 2$

- the contours of constant density are hyper-ellipsoids with the principal directions aligned with the coordinate axes.
- the components of $\mathbf{z}$ are then *statistically independent* since the distribution of $\mathbf{z}$ can be written as the product of the distributions for each of the components separately in the form

$$p_{\mathbf{z}}(z) = \prod_{i=1}^{n} p(z_i)$$

- the total number of independent parameters in the distribution is $2n$ ($n$ for the mean vector and $n$ for the diagonal covariance matrix).
- if $\sigma_i = \sigma$ for all $i$, the contours of constant density are hyper-spheres.

### 2.8.1 Bivariate normal distribution

Let us consider a bivariate ($n = 2$) normal density whose mean is $\mu = [\mu_1, \mu_2]^T$ and the covariance matrix is

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}$$

The correlation coefficient is

$$\rho = \frac{\sigma_{12}}{\sigma_1 \sigma_2}$$

It can be shown that the general bivariate normal density has the form

$$p(z_1, z_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left[\left(\frac{z_1-\mu_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{z_1-\mu_1}{\sigma_1}\right)\left(\frac{z_2-\mu_2}{\sigma_2}\right) + \left(\frac{z_2-\mu_2}{\sigma_2}\right)^2\right]\right]$$

A plot of a bivariate normal density with $\mu = [0, 0]$ and $\Sigma = [1.2919, 0.4546; 0.4546, 1.7081]$ and a corresponding contour curve are traced in Figure 2.11.

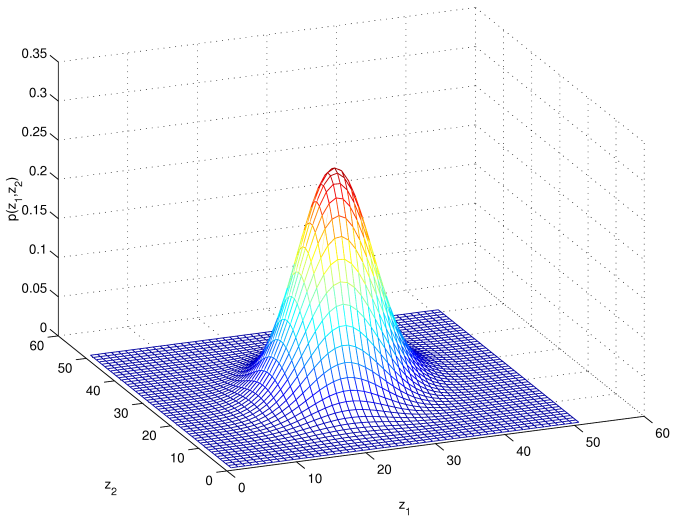



Figure 2.11: Bivariate normal density function

```
# script gaussXYZ.R
# It visualizes different bivariate gaussians with different
# orientation and axis' length

library(mvtnorm)
x <- seq(-10, 10, by= .5)
y <- x

z<-array(0,dim=c(length(x),length(y)))

#th : rotation angle of the first principal axis
#ax1: length principal axis 1
#ax2: length principal axis 2

ax1<-1

for (th in seq(0,pi,by=pi/8)){
  for (ax2 in c(4,12)){
    Rot<-array(c(cos(th), -sin(th), sin(th), cos(th)),dim=c(2,2)); #rotation matrix
    A<-array(c(ax1, 0, 0, ax2),dim=c(2,2))
    Sigma<-(Rot%*%A)%*%t(Rot)
    E<-eigen(Sigma)
    print(paste("Eigenvalue of the Variance matrix=",E$values))
    print(E$vectors)
    for (i in 1:length(x)){
      for (j in 1:length(y)){
        z[i,j]<-dmvnorm(c(x[i],y[j]),sigma=Sigma)
      }
    }
    z[is.na(z)] <- 1


    op <- par(bg = "white")
```

```
    prob.z<-z
    par(ask=TRUE)
    persp(x, y, prob.z, theta = 30, phi = 30, expand = 0.5, col = "lightblue")
    #           contour(x,y,z)
    title (paste("BIVARIATE GAUSSIAN; Rot. angle=",
                 round(th,digits=3),"; Length axis 1=", ax1, "; Length axis 2=", ax2))
  }
}
```

One of the important properties of the multivariate normal density is that all conditional and marginal probabilities are also normal. Using the relation

$$p(z_2|z_1) = \frac{p(z_1, z_2)}{p(z_1)}$$

we find that $p(z_2|z_1)$ is a normal distribution $\mathcal{N}(\mu_{2|1}, \sigma^2_{2|1})$, where

$$\mu_{2|1} = \mu_2 + \rho\frac{\sigma_2}{\sigma_1}(z_1 - \mu_1)$$
$$\sigma^2_{2|1} = \sigma^2_2(1 - \rho^2)$$

Note that

- $\mu_{2|1}$ is a linear function of $z_1$: if the correlation coefficient $\rho$ is positive, the larger $z_1$, the larger $\mu_{2|1}$.
- if there is no correlation between $z_1$ and $z_2$, the two variables are independent, i.e. we can ignore the value of $z_1$ to estimate $\mu_2$.

## 2.9 Linear combinations of r.v.

The expected value of a linear combination of r.v. is simply the linear combination of their respective expectation values

$$E[a\mathbf{x} + b\mathbf{y}] = aE[\mathbf{x}] + bE[\mathbf{y}], \qquad a \in \mathbb{R}, b \in \mathbb{R}$$

i.e., expectation is a linear statistic. On the contrary, the variance is not a linear statistic. We have

$$\begin{aligned} \mathrm{Var}\left[a\mathbf{x} + b\mathbf{y}\right] &= a^2\mathrm{Var}\left[\mathbf{x}\right] + b^2\mathrm{Var}\left[\mathbf{y}\right] + 2ab\left(E[\mathbf{xy}] - E[\mathbf{x}]E[\mathbf{y}]\right) && (2.9.55)\\ &= a^2\mathrm{Var}\left[\mathbf{x}\right] + b^2\mathrm{Var}\left[\mathbf{y}\right] + 2ab\mathrm{Cov}[\mathbf{x}, \mathbf{y}] && (2.9.56)\end{aligned}$$

where the quantity $\mathrm{Cov}[\mathbf{x}, \mathbf{y}]$ is defined in (2.7.51).

Note that, if the random variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ are independent, then

$$\mathrm{Var}\left[\sum_{i=1}^{k} c_i\mathbf{x}_i\right] = \sum_{i=1}^{k} c_i\sigma_i^2$$

### 2.9.1 The sum of i.i.d. random variables

Suppose that $\mathbf{z}_1$,$\mathbf{z}_2$,...,$\mathbf{z}_N$ are i.i.d. (identically and independently distributed) random variables, discrete or continuous, each having a probability distribution with mean $\mu$ and variance $\sigma^2$. Let us consider the two derived r.v., that is the sum

$$\mathbf{S}_N = \mathbf{z}_1 + \mathbf{z}_2 + \cdots + \mathbf{z}_N$$

and the average

$$\bar{\mathbf{z}} = \frac{\mathbf{z}_1 + \mathbf{z}_2 + \cdots + \mathbf{z}_N}{N} \tag{2.9.57}$$

The following relations hold

$$E[\mathbf{S}_N] = N\mu, \qquad \text{Var}\,[\mathbf{S}_N] = N\sigma^2 \tag{2.9.58}$$

$$E[\bar{\mathbf{z}}] = \mu, \qquad \text{Var}\,[\bar{\mathbf{z}}] = \frac{\sigma^2}{N} \tag{2.9.59}$$

An illustration of these relations by simulation can be obtained by running the R script `sum_rv.R`.

## 2.10 Transformation of random variables

Let $\mathbf{x}$ a continuous r.v. and $\mathbf{y} = f(\mathbf{x})$ a function of $\mathbf{x}$, for example $f(\mathbf{x}) = \mathbf{x}^2$. Note that $E[\mathbf{y}|x] = f(x)$. But what about $E[\mathbf{y}]$?

It can be shown that

$$E[\mathbf{y}] = E[f(\mathbf{x})] = \int f(x) dF_{\mathbf{x}}(x) \tag{2.10.60}$$

Note that if $\mathcal{E}$ is an event (e.g. a subset of $\mathcal{X}$) and $f(x) = I_{\mathcal{E}}(x)$ is the indicator function defined in (2.1.1) then

$$E[I_{\mathcal{E}}(\mathbf{x})] = \int_{\mathcal{E}} dF_{\mathbf{x}}(x) = \text{Prob}\,\{\mathbf{x} \in \mathcal{E}\} \tag{2.10.61}$$

Another important result is the Jensen's inequality.

**Theorem 10.1** (Jensen's inequality)**.** *If $f$ is a convex function then $E[f(\mathbf{x})] \geq f(E[\mathbf{x}])$ while if $f$ is concave then $E[f(\mathbf{x})] \leq f(E[\mathbf{x}])$*

## 2.11 The central limit theorem

**Theorem 11.1.** *Assume that $\mathbf{z}_1$,$\mathbf{z}_2$,...,$\mathbf{z}_N$ are i.i.d. random variables, discrete or continuous, each having the same probability distribution with finite mean $\mu$ and finite variance $\sigma^2$. As $N \to \infty$, the standardized random variable*

$$\frac{(\bar{\mathbf{z}} - \mu)\sqrt{N}}{\sigma}$$

*which is identical to*

$$\frac{(\mathbf{S}_N - N\mu)}{\sqrt{N}\sigma}$$

*converges* in distribution *(Definition 2.6) to a r.v. having the standardized* normal distribution $\mathcal{N}(0, 1)$.

This theorem, which holds regardless of the common distribution of $\mathbf{z}_i$, justifies the importance of the normal distribution, since many r.v. of interest are either sums or averages. Think for example to the commute time of the example in Section 2.2 which can be considered as the combined effect of several causes.

An ilustration of the theorem by simulation is obtained by running the R script `central.R`.

## 2.12 The Chebyshev's inequality

Let $\mathbf{z}$ be a *generic* random variable, discrete or continuous , having mean $\mu$ and variance $\sigma^2$. The Chebyshev's inequality states that for any positive constant $d$

$$\text{Prob}\left\{|\mathbf{z}-\mu| \geq d\right\} \leq \frac{\sigma^2}{d^2} \tag{2.12.62}$$

An ilustration of the Chebyshev's inequality by simulation can be found in the R script `cheby.R`.

Note that if we put $\mathbf{z}$ equal to the quantity in (2.9.57), then from (2.9.58) and (2.12.62) we find

$$\text{Prob}\left\{|\bar{\mathbf{z}}-\mu| \geq d\right\} \leq \frac{\sigma^2}{Nd^2} \tag{2.12.63}$$

i.e. the *weak law* of large numbers (Section 2.1.4). This law states that the average of a large sample converges in probability to the mean of the distribution.

# Chapter 3

# Classical approach to parametric estimation and testing

In the previous chapter we reviewed some basic notions of probability. Under the assumption that a probabilistic model is correct, logical deduction through the rules of mathematical probability leads to a description of the properties of data that might arise from the real situation. This deductive approach requires however the existence and the availability of a probabilistic model. The theory of statistics is designed to reverse the deductive process. It takes data that have arisen from a practical situation and uses the data to suggest a probabilistic model, to estimate its parameters and eventually to validate it. This chapter will focus on the estimation methodology, intended as the inductive process which leads from observed data to a probabilistic description of reality. We will focus here on the parametric approach, which assumes that some probabilistic model is available apart from some unknown parameters. Parametric estimation algorithms build estimates from data and, more important, statistical measures to assess their quality.

There are two main approaches to parametric estimation

**Classical or frequentist:** it is based on the idea that sample data are the sole quantifiable form of relevant information and that the parameters are *fixed but unknown*. It is related to the frequency view of probability (Section 2.1.3).

**Bayesian approach:** the parameters are supposed to be *random variables*, having a distribution *prior* to data observation and a distribution *posterior* to data observation. This approach assumes that there exists something beyond data, (i.e. a subjective degree of belief), and that this belief can be described in probabilistic form.

For reasons of space, we will limit here to consider the classical approach. It is important however not to neglect the important role of the Bayesian approach which led recently to a large amount of research in Bayesian data analysis and relative applications in machine learning [51].

## 3.1 Classical approach

The classical approach to parameter estimation dates back to the period 1920-35 when J. Neyman and E.S. Pearson, stimulated by problems in biology and industry,

concentrated on the principles for testing hypothesis and R.A. Fisher who was interested in agricultural issues gave attention to the estimation problem.

Parametric estimation can be presented as follows. Consider a continuous r.v. $\mathbf{z}$ and suppose that

1. we do not know completely the distribution $F_{\mathbf{z}}(z)$ but that we can write it in a parametric form (Section 2.3.1)

$$F_{\mathbf{z}}(z) = F_{\mathbf{z}}(z, \theta)$$

where the analytical form of the function $F_{\mathbf{z}}(\cdot)$ is known but $\theta \in \Theta$ is an unknown parameter vector,

2. we have access to a set $D_N$ of $N$ i.i.d. measurements of $\mathbf{z}$, called *sample data*.

An example of parametric distribution function is the Normal distribution (Section (2.7.4)) where the parameter vector $\theta = [\mu, \sigma]$ is unknown. The goal of the estimation procedure is to find a value $\hat{\theta}$ of the parameter $\theta$ so that the parameterized distribution $F_{\mathbf{z}}(z, \hat{\theta})$ closely matches the distribution of data.

The notation *i.i.d.* stands for **i**dentically and **i**ndependently **d**istributed. Identically distributed means that all the observations have been sampled from the same distribution, that is

$$\text{Prob}\{\mathbf{z}_i = z\} = \text{Prob}\{\mathbf{z}_j = z\} \qquad \text{for all } i, j = 1, \ldots, N \text{ and } z \in \mathcal{Z}$$

Independently distributed means that the fact that we have observed a certain value $\mathbf{z}_i$ does not influence the probability of observing the value $\mathbf{z}_j$, that is

$$\text{Prob}\{\mathbf{z}_j = z | \mathbf{z}_i = z_i\} = \text{Prob}\{\mathbf{z}_j = z\}$$

**Example**

Here you find some examples of estimation problems:

1. Let $D_N = \{20, 31, 14, 11, 19, \ldots\}$ be the times in minutes spent the last 2 weeks to go home. What is the mean time to reach my house from ULB?

2. Consider the car traffic in the boulevard Jacques. Suppose that the measures of the inter-arrival times are $D_N = \{10, 11, 1, 21, 2, \ldots\}$ seconds. What does this imply about the mean inter-arrival time?

3. Consider the students of the last year of Computer Science. What is the variance of their grades?

4. Let $\mathbf{z}$ be the r.v. denoting tomorrow's temperature. How can I estimate its mean value on the basis of past observations?

•

Parametric estimation is a *mapping* from the space of the sample data to the space of parameters $\Theta$. The two possible outcomes are:

1. some specific value of $\Theta$. In this case we have the so-called *point estimation*.

2. some particular region of $\Theta$. In this case we obtain an *interval of confidence* on the value of the parameter.

### 3.1.1 Point estimation

Consider a random variable $\mathbf{z}$ with a parametric distribution $F_{\mathbf{z}}(z, \theta)$, $\theta \in \Theta$. The unknown parameter can be written as a function(al) of $F$

$$\theta = t(F)$$

This corresponds to the fact that $\theta$ is a characteristic of the population described by $F_{\mathbf{z}}(\cdot)$. For instance the expected value parameter $\mu = t(F) = \int z dF(z)$ is a functional of $F$.

Suppose now that we have observed a set of $N$ i.i.d. values $D_N = \{z_1, z_2, \ldots, z_N\}$. A *point estimate* is an example of *statistic*, where by statistic it is generally meant any function of the sample data $D_N$. In other terms a *point estimate* is a function

$$\hat{\theta} = g(D_N)$$

of the sample dataset $D_N$, where $g(\cdot)$ stands for the estimation algorithm, that is the procedure which returns the estimation starting from a dataset $D_N$.

There are two main issues in estimation and more generally in data analysis, statistics and machine learning: how to construct an estimator (i.e. which form should $g$ take) and how to assess the quality of the returned estimation $\hat{\theta}$. In Sections 3.3 and 3.8 we will discuss two strategies for defining an estimator; the *plug-in principle* and the *maximum likelihood*. In Section 3.5 we will present the statistical measures most commonly adopted to assess an estimator accuracy.

Before introducing the plug-in principle we need, however, to present the notion of empirical distribution.

## 3.2 Empirical distributions

Suppose we have observed a i.i.d. random sample of size $N$ from a probability distribution $F_{\mathbf{z}}(\cdot)$

$$F_{\mathbf{z}} \rightarrow \{z_1, z_2, \ldots, z_N\}$$

The *empirical distribution probability* $\hat{F}$ is defined as the *discrete distribution* that assigns probability $1/N$ to each value $z_i$, $i = 1, \ldots, N$. In other words, $\hat{F}$ assigns to a set $A$ in the sample space of $\mathbf{z}$ its empirical probability

$$\text{Prob}\{\mathbf{z} \in A\} \approx \frac{\#z_i \in A}{N}$$

that is the proportion of the observed samples in $D_N$ which occur in $A$.

It can be proved that the vector of observed frequencies in $\hat{F}$ is a *sufficient* statistic for the true distribution $F(\cdot)$, i.e. all the information about $F(\cdot)$ contained in $D_N$ is also contained in $\hat{F}(\cdot)$.

Consider now the distribution function $F_{\mathbf{z}}(z)$ of a continuous rv $\mathbf{z}$ and a set of $N$ observed samples $D_N = \{z_1, \ldots, z_N\}$. Since

$$F_{\mathbf{z}}(z) = \text{Prob}\{\mathbf{z} \leq z\}$$

we define $N(z)$ as the number of samples in $D_N$ that do not exceed $z$. We obtain then the empirical estimate of $F(\cdot)$

$$\hat{F}_{\mathbf{z}}(z) = \frac{N(z)}{N} = \frac{\#z_i \leq z}{N} \tag{3.2.1}$$

This function is a staircase function with discontinuities at the points $z_i$ (Figure 3.1).

Note that, being $\mathbf{D}_N$ a random vector, the function $\hat{\mathbf{F}}_{\mathbf{z}}(\cdot)$ is random, too. The following two properties (unbiasedness and consistency) can be shown

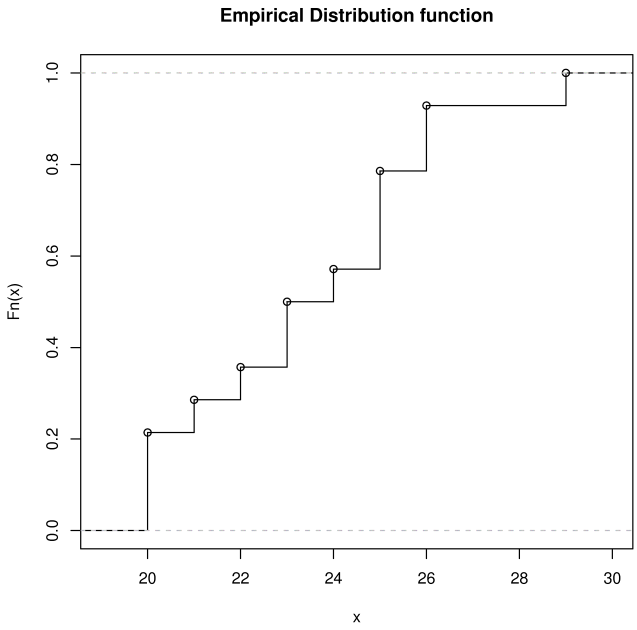



Figure 3.1: Empirical distribution.

**Theorem 2.1.** *For any fixed $z$*

$$E_{\mathbf{D}_N}[\hat{\mathbf{F}}_{\mathbf{z}}(z)] = F_{\mathbf{z}}(z) \tag{3.2.2}$$

$$Var\left[\hat{\mathbf{F}}_{\mathbf{z}}(z)\right] = \frac{F_{\mathbf{z}}(z)(1 - F_{\mathbf{z}}(z))}{N} \tag{3.2.3}$$

**Theorem 2.2** (Glivenko-Cantelli theorem)**.**

$$\sup_{-\infty<z<\infty} |\hat{\mathbf{F}}_{\mathbf{z}}(z) - F_{\mathbf{z}}(z)| \overset{N\to\infty}{\to} 0 \quad \textit{almost surely} \tag{3.2.4}$$

*where the definition of almost sure convergence is in Appendix (Def. 2.2).*

The two theoretical results can be simulated by running the R scripts `cumdis_2.R` and `cumdis_1.R`, respectively.

**Example**

Suppose that our dataset of observations is made of the following $N = 14$ samples

$$D_N = \{20, 21, 22, 20, 23, 25, 26, 25, 20, 23, 24, 25, 26, 29\}$$

The empirical distribution function $\hat{F}_{\mathbf{z}}$ (which can be traced by running the script `cumdis.R`) is plotted in Figure 3.1.

## 3.3 Plug-in principle to define an estimator

Consider an r.v. $\mathbf{z}$ and sample dataset $D_N$ drawn from the parametric distribution $F_{\mathbf{z}}(z, \theta)$. The main issue of estimation is how to define an estimate of $\theta$. A possible

solution is given by the *plug-in principle*, that is a simple method of estimating parameters from samples. The *plug-in estimate* of a parameter $\theta$ is defined to be:

$$\hat{\theta} = t(\hat{F}(z)) \tag{3.3.5}$$

obtained by replacing the distribution function with the empirical distribution in the analytical expression of the parameter.

If $\theta$ is the mean of $\mathbf{z}$, the sample average is an example of plug-in estimate

$$\hat{\mu} = \int z d\hat{F}(z) = \frac{1}{N}\sum_{i=1}^{N} z_i$$

The following section will discuss the plug-in estimators of the first two moments of a probability distribution.

### 3.3.1 Sample average

Consider an r.v. $\mathbf{z} \sim F_{\mathbf{z}}(\cdot)$ such that

$$\theta = E[\mathbf{z}] = \int z dF(z)$$

with $\theta$ unknown. Suppose we have available the sample $F_{\mathbf{z}} \rightarrow D_N$, made of $N$ observations. The *plug-in* point estimate of $\theta$ is given by the *sample average*

$$\hat{\theta} = \frac{1}{N}\sum_{i=1}^{N} z_i = \hat{\mu} \tag{3.3.6}$$

which is indeed a statistic, i.e. a function of the data set.

### 3.3.2 Sample variance

Consider a r.v. $\mathbf{z} \sim F_{\mathbf{z}}(\cdot)$ where the mean $\mu$ and the variance $\sigma^2$ are unknown. Suppose we have available the sample $F_{\mathbf{z}} \rightarrow D_N$. Once we have the sample average $\hat{\mu}$, the *plug-in* estimate of $\sigma^2$ is given by the *sample variance*

$$\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(z_i - \hat{\mu})^2 \tag{3.3.7}$$

The presence of $N-1$ instead of $N$ at the denominator will be explained later. Note also that the following relation holds for all $z_i$

$$\frac{1}{N}\sum_{i=1}^{N}(z_i - \hat{\mu})^2 = \left(\frac{1}{N}\sum_{i=1}^{N} z_i^2\right) - \hat{\mu}^2$$

## 3.4 Sampling distribution

Given a dataset $D_N$, let us consider a point estimate

$$\hat{\theta} = g(D_N)$$

Note that since $D_N$ is the outcome of $N$ realizations of a r.v. $\mathbf{z}$, the vector $D_N$ can be considered as the realization of a random vector $\mathbf{D}_N$. By applying the transformation $g$ to the random variable $\mathbf{D}_N$ we obtain the random variable

$$\hat{\boldsymbol{\theta}} = g(\mathbf{D}_N)$$

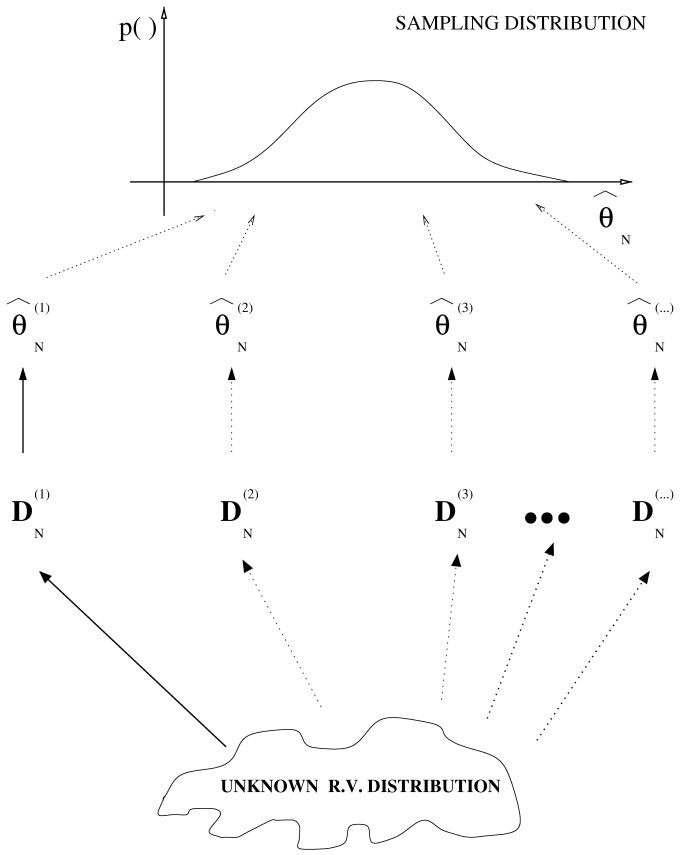



Figure 3.2: From the r.v. distribution to the sampling distribution.

which is called the *point estimator* of $\theta$. A key point is the following: *while $\theta$ is an (unknown) fixed value, its estimator $\hat{\boldsymbol{\theta}}$ is a random variable.* If the goal of the estimation procedure is to estimate $\theta = \mu$, this means also that while the mean is an unknown fixed value the average $\hat{\boldsymbol{\mu}}$ is a random variable.

The probability distribution of the r.v. $\hat{\boldsymbol{\theta}}$ is called the *sampling distribution.* An example of sampling distribution is plotted in Figure 3.2.

### Script

Suppose we observe $N$ realizations of a Normal random variable with mean $\mu = 0$ and standard deviation $\sigma = 10$. The following script illustrates the sampling distribution of the sample average estimator of $\mu$ for different number $N$ of samples.

```
# script sam_dis.R
# it visualizes the distribution of the estimator
# of the mean of a gaussian random variable
par(ask=TRUE)
for (N in seq(20,100,by=10)){
  ## N: number of samples in D_N
  mu<-0
  sdev<-10
  R<-10000  ## number of repetitions
```

```
  I<-seq(-50,50,by=.5)
  p<-dnorm(I,mean=mu,sd=sdev)
  plot(I,p,type="l",
       main=paste("Distribution of  r.v. z: var=",sdev^2))

  mu.hat<-array(0,dim=c(R,1))
  for (r in 1:R){

    D<-rnorm(N,mean=mu,sd=sdev)

    mu.hat[r,1]<-mean(D)
  }

  hist(mu.hat,freq=FALSE,
       main= paste("Mean estimator on samples of size N=",N, ": var=",var(mu.hat)),xlim=c(min(I)
  p.mu.hat<-dnorm(I,mean=mean(mu.hat),sd=sqrt(var(mu.hat)))
  lines(I,p.mu.hat,type="l")

  var(mu.hat)
  sdev^2/N
}
```

## 3.5 The assessment of an estimator

This section introduces the most used measure to assess the quality of an estimator from a statistical perspective.

### 3.5.1 Bias and variance

Once defined an estimator $\hat{\boldsymbol{\theta}}$ the next important thing is to assess how it is accurate. This leads us to the definition of bias, variance and standard error of an estimator.

**Definition 5.1** (Bias of an estimator)**.** An estimator $\hat{\boldsymbol{\theta}}$ of $\theta$ is said to be *unbiased* if and only if

$$E_{\mathbf{D}_N}[\hat{\boldsymbol{\theta}}] = \theta$$

Otherwise, it is said to be *biased* with bias

$$\text{Bias}[\hat{\boldsymbol{\theta}}] = E_{\mathbf{D}_N}[\hat{\boldsymbol{\theta}}] - \theta \tag{3.5.8}$$

**Definition 5.2** (Variance of an estimator)**.** The variance of an estimator $\hat{\boldsymbol{\theta}}$ of $\theta$ is the variance of its sampling distribution

$$\text{Var}\left[\hat{\boldsymbol{\theta}}\right] = E_{\mathbf{D}_N}[(\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}])^2]$$

**Definition 5.3** (Standard error)**.** The square root of the variance

$$\hat{\boldsymbol{\sigma}} = \sqrt{\text{Var}\left[\hat{\boldsymbol{\theta}}\right]}$$

is called the *standard error* of the estimator $\hat{\boldsymbol{\theta}}$.

Note that an unbiased estimator is an estimator that takes on average the right value. At the same time, many unbiased estimators may exist for a parameter $\theta$. Other important considerations are:

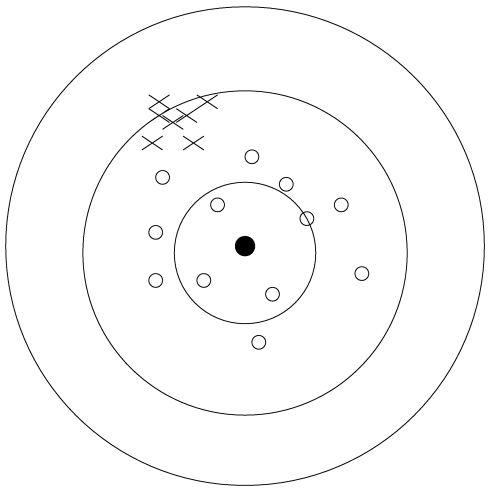


Figure 3.3: The dart analogy: the target is the unknown parameter, the round dots represent some realizations of the estimator R, while the crosses represents some realizations of the estimator C.

- Given a generic transformation $f(\cdot)$, if $\hat{\boldsymbol{\theta}}$ is unbiased for $\theta$ does not imply that that $f(\hat{\boldsymbol{\theta}})$ is unbiased for $f(\theta)$.
- A biased estimator with a known bias (not depending on $\theta$) is equivalent to an unbiased estimator since we can easily compensate for the bias.
- In general the standard error $\hat{\boldsymbol{\sigma}}$ is not an unbiased estimator of $\sigma$ even if $\hat{\boldsymbol{\sigma}}^2$ is an unbiased estimator of $\sigma^2$.

**Example**

An intuitive manner of visualizing the notion of sampling distribution of an estimator and the related concepts of bias and variance is to use the analogy of the dart game. The unknown parameter $\theta$ can be seen as the target and the estimator $\hat{\boldsymbol{\theta}}$ as a player. Figure 3.3 shows the target (black dot) together with the distribution of the draws of two different players: the C (cross) player and the R (round) player. In terms of our analogy the cross player/estimator has small variance but large bias, while the round one has small bias and large variance. Which is the best one?

•

The following section will show that for a generic r.v. $\mathbf{z}$ and an i.i.d. dataset $D_N$, the sample average $\hat{\boldsymbol{\mu}}$ and the sample variance $\hat{\boldsymbol{\sigma}}^2$ are unbiased estimators of the mean $E[\mathbf{z}]$ and the variance $\mathrm{Var}[\mathbf{z}]$, respectively.

### 3.5.2 Bias and variance of $\hat{\boldsymbol{\mu}}$

Consider a random variable $\mathbf{z} \sim F_{\mathbf{z}}(\cdot)$. Let $\mu$ and $\sigma^2$ the mean and the variance of $F_{\mathbf{z}}(\cdot)$, respectively. Suppose we have observed the i.i.d. sample $D_N \leftarrow F_{\mathbf{z}}$. From (3.3.6) we obtain

$$E_{\mathbf{D}_N}[\hat{\boldsymbol{\mu}}] = E_{\mathbf{D}_N}\left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{z}_i\right] = \frac{\sum_{i=1}^{N}E[\mathbf{z}_i]}{N} = \frac{N\mu}{N} = \mu \tag{3.5.9}$$

This means that the *sample average estimator* is not biased, whatever the distribution $F_{\mathbf{z}}(\cdot)$ is. And what about its variance? Since according to the i.i.d. assumption $\text{Cov}[\mathbf{z}_i, \mathbf{z}_j] = 0$, for $i \neq j$, from (2.9.55) we obtain that the variance of the *sample average estimator* is

$$\text{Var}\,[\hat{\boldsymbol{\mu}}] = \text{Var}\left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{z}_i\right] = \frac{1}{N^2}\text{Var}\left[\sum_{i=1}^{N}\mathbf{z}_i\right] = \frac{1}{N^2}N\sigma^2 = \frac{\sigma^2}{N}. \tag{3.5.10}$$

In fact, $\hat{\boldsymbol{\mu}}$ is like the "round player" in dart game with some variance but no bias.

You can visualize the bias and variance of the sample average estimator by running the R script presented in Section 3.4.

### 3.5.3 Bias of the estimator $\hat{\boldsymbol{\sigma}}^2$

Let us study now the bias of the estimator of the variance of $\mathbf{z}$.

$$E_{\mathbf{D}_N}[\hat{\boldsymbol{\sigma}}^2] = E_{\mathbf{D}_N}\left[\frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{z}_i - \hat{\boldsymbol{\mu}})^2\right] \tag{3.5.11}$$

$$= \frac{N}{N-1}E_{\mathbf{D}_N}\left[\frac{1}{N}\sum_{i=1}^{N}(\mathbf{z}_i - \hat{\boldsymbol{\mu}})^2\right] \tag{3.5.12}$$

$$= \frac{N}{N-1}E_{\mathbf{D}_N}\left[\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{z}_i^2\right) - \hat{\boldsymbol{\mu}}^2\right] \tag{3.5.13}$$

Since $E[\mathbf{z}^2] = \mu^2 + \sigma^2$ and $\text{Cov}[\mathbf{z}_i, \mathbf{z}_j] = 0$, the first term inside the $E[\cdot]$ is

$$E_{\mathbf{D}_N}\left[\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{z}_i^2\right)\right] = \frac{1}{N}\sum_{i=1}^{N}E_{\mathbf{D}_N}\left[\mathbf{z}_i^2\right] = \frac{1}{N}N(\mu^2+\sigma^2)$$

Since $E\left[\left(\sum_{i=1}^{N}\mathbf{z}_i\right)^2\right] = N^2\mu^2 + N\sigma^2$ the 2nd term is

$$E_{\mathbf{D}_N}[\hat{\boldsymbol{\mu}}^2] = \frac{1}{N^2}E_{\mathbf{D}_N}\left[\left(\sum_{i=1}^{N}\mathbf{z}_i\right)^2\right] = \frac{1}{N^2}(N^2\mu^2 + N\sigma^2) = \mu^2 + \sigma^2/N$$

It follows that

$$E_{\mathbf{D}_N}[\hat{\boldsymbol{\sigma}}^2] \;=\; \frac{N}{N-1}\left((\mu^2+\sigma^2) - (\mu^2+\sigma^2/N)\right) \;=\; \frac{N}{N-1}\left(\frac{N-1}{N}\sigma^2\right) \;=\; \sigma^2$$

This result justifies our definition (3.3.7). Once the term $N-1$ is inserted at the denominator, the sample variance estimator is not biased.

Some points are worth considering:

- The results (3.5.9),(3.5.10) and (3.5.11) are *independent* of the family of the distribution $F(\cdot)$.
- According to (3.5.10), the variance of $\hat{\boldsymbol{\mu}}$ is $1/N$ times the variance of $\mathbf{z}$. This is a formal justification of the reason why taking averages on a large number of samples is recommended: the larger $N$, the smaller is $\text{Var}\,[\hat{\boldsymbol{\mu}}]$, so a bigger $N$ for a given $\sigma^2$ implies a better estimate of $\mu$.

- According to the central limit theorem (Section 2.11), under quite general conditions on the distribution $F_{\mathbf{z}}$, the distribution of $\hat{\boldsymbol{\mu}}$ will be approximately normal as $N$ gets large, which we can write as

$$\hat{\boldsymbol{\mu}} \sim \mathcal{N}(\mu, \sigma^2/N) \quad \text{for } N \to \infty$$

- The *standard error* $\sqrt{\text{Var}\,[\hat{\boldsymbol{\mu}}]}$ is a common way of indicating statistical accuracy. Roughly speaking, if the estimator is not biased and the conditions of the central limit theorem apply, we expect $\hat{\boldsymbol{\mu}}$ to be less than one standard error away from $\mu$ about 68% of the time, and less than two standard errors away from $\mu$ about 95% of the time (see Table 2.3) .

**Script**

You can visualize the bias and variance of the sample variance estimator by running the following R script

```
# script sam_dis2.R
# it visualizes the distribution of the estimator
# of the variance of a gaussian random variable


par(ask=TRUE)
N<-10
mu<-0
sdev<-10
R<-10000

I<-seq(-50,50,by=.5)
p<-dnorm(I,mean=mu,sd=sdev)
plot(I,p,type="l",
     main=paste("Distribution of  r.v. z: var=",sdev^2))

var.hat<-array(0,dim=c(R,1))
std.hat<-array(0,dim=c(R,1))
for (r in 1:R){

  D<-rnorm(N,mean=mu,sd=sdev)

  var.hat[r,1]<-var(D)
  std.hat[r,1]<-sd(D)
}

I2<-seq(0,2*sdev^2,by=.5)
hist(var.hat,freq=FALSE,
     main= paste("Variance estimator on N=",N, " samples: mean=",mean(var.hat))) #,xlim=range


ch<-(var.hat*(N-1))/(sdev^2)
hist(ch,freq=FALSE)
p.var.hat<-dchisq(I2,df=N-1)
lines(I2,p.var.hat,type="l")
```

### 3.5.4 Bias/variance decomposition of MSE

Bias and variance are two independent criteria to assess the quality of an estimator. As shown in Figure 3.3 we could have two estimators behaving in an opposite ways: the first has large bias and low variance, while the second has large variance and small bias. How can we choose among them? We need a measure able to combine or merge the two to a single criteria. This is the role of the *mean-square error* (MSE) measure.

When $\hat{\boldsymbol{\theta}}$ is a biased estimator of $\theta$, its accuracy is usually assessed by its MSE rather than simply by its variance. The MSE is defined by

$$\text{MSE} = E_{\mathbf{D}_N}[(\theta - \hat{\boldsymbol{\theta}})^2]$$

For a generic estimator it can be shown that

$$\text{MSE} = (E[\hat{\boldsymbol{\theta}}] - \theta)^2 + \text{Var}\left[\hat{\boldsymbol{\theta}}\right] = \left[\text{Bias}[\hat{\boldsymbol{\theta}}]\right]^2 + \text{Var}\left[\hat{\boldsymbol{\theta}}\right] \tag{3.5.14}$$

i.e., the mean-square error is equal to the sum of the variance and the squared bias of the estimator . Here it is the analytical derivation

$$\text{MSE} = E_{\mathbf{D}_N}[(\theta - \hat{\boldsymbol{\theta}})^2] = E_{\mathbf{D}_N}[(\theta - E[\hat{\boldsymbol{\theta}}] + E[\hat{\boldsymbol{\theta}}] - \hat{\boldsymbol{\theta}})^2] = \tag{3.5.15}$$

$$= E_{\mathbf{D}_N}[(\theta - E[\hat{\boldsymbol{\theta}}])^2] + E_{\mathbf{D}_N}[(E[\hat{\boldsymbol{\theta}}] - \hat{\boldsymbol{\theta}})^2] + E_{\mathbf{D}_N}[2(\theta - E[\hat{\boldsymbol{\theta}}])(E[\hat{\boldsymbol{\theta}}] - \hat{\boldsymbol{\theta}})] = \tag{3.5.16}$$

$$= E_{\mathbf{D}_N}[(\theta - E[\hat{\boldsymbol{\theta}}])^2] + E_{\mathbf{D}_N}[(E[\hat{\boldsymbol{\theta}}] - \hat{\boldsymbol{\theta}})^2] + 2(\theta - E[\hat{\boldsymbol{\theta}}])(E[\hat{\boldsymbol{\theta}}] - E[\hat{\boldsymbol{\theta}}]) = \tag{3.5.17}$$

$$= (E[\hat{\boldsymbol{\theta}}] - \theta)^2 + \text{Var}\left[\hat{\boldsymbol{\theta}}\right] \tag{3.5.18}$$

This decomposition is typically called the *bias-variance* decomposition. Note that, if an estimator is unbiased then its MSE is equal to its variance.

### 3.5.5 Consistency

Suppose that the sample data contains $N$ independent observations $z_1, \ldots, z_N$ of a univariate random variable. Let the estimator of $\theta$ based on $N$ samples be denoted $\hat{\boldsymbol{\theta}}_N$.

As $N$ becomes larger, we might reasonably expect that $\hat{\boldsymbol{\theta}}_N$ improves as estimator of $\theta$ (in other terms it gets closer to $\theta$). The notion of consistency introduces this concept.

**Definition 5.4.** The estimator $\hat{\boldsymbol{\theta}}_N$ is said to be *weakly consistent* if $\hat{\boldsymbol{\theta}}_N$ converge to $\theta$ in probability, that is

$$\forall \epsilon > 0 \qquad \lim_{N\to\infty} \text{Prob}\left\{|\hat{\boldsymbol{\theta}}_N - \theta| \leq \epsilon\right\} = 1$$

**Definition 5.5.** The estimator $\hat{\boldsymbol{\theta}}_N$ is said *strongly consistent* if $\hat{\boldsymbol{\theta}}_N$ converge to $\theta$ with probability 1 (or almost surely).

$$\text{Prob}\left\{\lim_{N\to\infty} \hat{\boldsymbol{\theta}}_N = \theta\right\} = 1$$

For a scalar $\theta$ the property of convergence guarantees that the sampling distribution of $\hat{\boldsymbol{\theta}}_N$ becomes less disperse as $N \to \infty$. In other terms a consistent estimator is asymptotically unbiased. It can be shown that a sufficient condition for weak consistency of unbiased estimators $\hat{\boldsymbol{\theta}}_N$ is that $\text{Var}\left[\hat{\boldsymbol{\theta}}_N\right] \to 0$ as $N \to \infty$.

It is important to remark also that the property of unbiasedness (for finite samples) and consistency are largely unrelated.

**Exercise**

Suppose an estimator of the mean that takes into consideration only the first 10 samples, whatever is the number $N > 10$ of samples. Is this estimator consistent?

•

### 3.5.6 Efficiency

Suppose we have two *unbiased and consistent* estimators. How to choose between them?

**Definition 5.6** (Relative efficiency)**.** Let us consider two unbiased estimators $\hat{\boldsymbol{\theta}}_1$ and $\hat{\boldsymbol{\theta}}_2$. If

$$\text{Var}\left[\hat{\boldsymbol{\theta}}_1\right] < \text{Var}\left[\hat{\boldsymbol{\theta}}_2\right]$$

we say that $\hat{\boldsymbol{\theta}}_1$ is *more efficient* than $\hat{\boldsymbol{\theta}}_2$.

If the estimators are biased, typically the comparison is done on the basis of the mean square error.

**Exercise**

Suppose $z_1, \ldots, z_N$ is a random sample of observations from a distribution with mean $\theta$ and variance $\sigma^2$. Study the unbiasedness and the consistency of the three estimators of the mean $\mu$:

$$\hat{\theta}_1 = \hat{\mu} = \frac{\sum_{i=1}^{N} z_i}{N}$$

$$\hat{\theta}_2 = \frac{N\hat{\theta}_1}{N+1}$$

$$\hat{\theta}_3 = z_1$$

•

### 3.5.7 Sufficiency

**Definition 5.7** (Sufficiency)**.** An estimator $\hat{\boldsymbol{\theta}}$ is said to be *sufficient* for $\theta$ if the conditional distribution of $\mathbf{z}$ given $\hat{\boldsymbol{\theta}}$ does not depend on $\theta$.

**Property 1** (Fisher-Neyman factorization criterion)**.** An estimator $\hat{\boldsymbol{\theta}}$ is sufficient for $\theta$ if and only if

$$P_{\mathbf{z}}(z, \theta) = g(\hat{\theta}, \theta)h(z)$$

**Definition 5.8** (Minimal sufficiency)**.** If $\hat{\boldsymbol{\theta}}$ is sufficient and no statistic of lower dimension is sufficient then $\hat{\boldsymbol{\theta}}$ is said to be *minimal sufficient*

Some considerations:

- If an estimator $\hat{\boldsymbol{\theta}}$ is sufficient for $\theta$, this means that all the information about $\theta$ contained in the data $D_N$ is obtained from $\hat{\boldsymbol{\theta}}$ alone.
- Usually $\hat{\boldsymbol{\theta}}$ will be of lower dimension than $D_N$.

Example: consider a sample data $D_N$ from $\mathcal{N}(\mu, \sigma^2)$ where $\sigma^2$ is known. The estimator $\hat{\boldsymbol{\mu}}$ is a sufficient estimator. This means that only $\hat{\boldsymbol{\mu}}$ and no other element of the data provides information about $\mu$.

## 3.6 The Hoeffding's inequality

A probabilistic measure of the discrepancy between the estimator $\hat{\boldsymbol{\mu}}$ and the quantity $\mu = E[\mathbf{z}]$ to be estimated is returned by the Hoeffding's inequality.

**Theorem 6.1.** *[64] Let $\mathbf{z}_1, \ldots, \mathbf{z}_N$ be independent bounded random variables such that $\mathbf{z}_i$ falls in the interval $[a_i, b_i]$ with probability one. Let their sum be $\mathbf{S}_N = \sum_{i=1}^N \mathbf{z}_i$. Then for any $\varepsilon > 0$ we have*

$$P\left\{|S_N - E[\mathbf{S}_N]| > \varepsilon\right\} \leq \exp\left\{-2\varepsilon^2 / \sum_{i=1}^N (b_i - a_i)^2\right\}$$

**Corollary 6.2.** *If the variables $\mathbf{z}_1, \ldots, \mathbf{z}_N$ are independent and identically distributed, the following bound on the discrepancy between the sample mean $\hat{\mu} = \frac{\sum_{i=1}^N z_i}{N}$ and the expected value $E[\mathbf{z}]$ holds*

$$P\left\{|\hat{\boldsymbol{\mu}} - E[\mathbf{z}]| > \varepsilon\right\} \leq \exp\left\{-2N\varepsilon^2/(b-a)^2\right\}$$

Assume that $\delta$ is a confidence parameter, that is we are $100(1-\delta)\%$ confident that the estimate $\hat{\mu}$ is within the accuracy $\varepsilon$ of the true expectation. It is possible to derive the expression

$$\varepsilon(N) = \sqrt{\frac{(b-a)^2 \log(2/\delta)}{2N}}$$

which measures with confidence $1-\delta$ how the sample mean $\hat{\mu}$, estimated on the basis of $N$ points, is close to the expectation $E[\mathbf{z}]$. We can also determine the number of samples $N$ necessary to obtain an accuracy $\varepsilon$ and a confidence $\delta$ by using the relation

$$N > \frac{(b-a)^2 \log(2/\delta)}{2\varepsilon^2}$$

Hoeffding's bound is a general bound that only relies on the assumption that samples are drawn independently. Bayesian bounds are another example of statistical bounds which give tighter results under the assumption that samples are drawn from a normal distribution.

## 3.7 Sampling distributions for Gaussian r.v.s

The results in Section 3.5 are independent of the type of distribution function $F_{\mathbf{z}}$. Additional results are available in the specific case of a normal random variable.

Let $\mathbf{z}_1, \ldots, \mathbf{z}_N$ be i.i.d. realization of $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$ and let us consider the following sample statistics

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^N \mathbf{z}_i, \quad \widehat{\mathbf{SS}} = \sum_{i=1}^N (\mathbf{z}_i - \hat{\boldsymbol{\mu}})^2, \quad \hat{\boldsymbol{\sigma}}^2 = \frac{\widehat{\mathbf{SS}}}{N-1}$$

It can be shown that the following relations hold

1. $\hat{\boldsymbol{\mu}} \sim \mathcal{N}(\mu, \sigma^2/N)$ and $N(\hat{\boldsymbol{\mu}} - \mu)^2 \sim \sigma^2 \chi_1^2$.
2. $\mathbf{z}_i - \mu \sim \mathcal{N}(0, \sigma^2)$, so $\sum_{i=1}^N (\mathbf{z}_i - \mu)^2 \sim \sigma^2 \chi_N^2$
3. $\sum_{i=1}^N (\mathbf{z}_i - \mu)^2 = \widehat{\mathbf{SS}} + N(\hat{\mu} - \mu)^2$
4. $\widehat{\mathbf{SS}} \sim \sigma^2 \chi_{N-1}^2$ or equivalently $\frac{(N-1)\hat{\boldsymbol{\sigma}}^2}{\sigma^2} \sim \chi_{N-1}^2$. See R script `sam_dis2.R`.

5. $\sqrt{N}(\hat{\boldsymbol{\mu}} - \mu)/\hat{\boldsymbol{\sigma}} \sim \mathcal{T}_{N-1}$ where $\mathcal{T}$ stands for the Student distribution (Section 2.7.6)

6. if $E[|\mathbf{z} - \mu|^4] = \mu_4$ then $\text{Var}\left[\hat{\boldsymbol{\sigma}}^2\right] = \frac{1}{N}\left(\mu_4 - \frac{N-3}{N-1}\sigma^4\right)$.

## 3.8 The principle of maximum likelihood

Let us consider

1. a density distribution $p_{\mathbf{z}}(z, \theta)$ which depends on a parameter $\theta$

2. a sample data $D_N = \{z_1, z_2, \ldots, z_N\}$ i.i.d. drawn from this distribution.

According to (2.5.38), the joint probability density of the sample data is the product

$$p_{\mathbf{D}_N}(D_N, \theta) = \prod_{i=1}^{N} p_{\mathbf{z}}(z_i, \theta) = L_N(\theta) \tag{3.8.19}$$

where for a fixed $D_N$, $L_N(\cdot)$ is a function of $\theta$ and is called the *empirical likelihood* of $\theta$ given $D_N$.

The principle of maximum likelihood was first used by Lambert around 1760 and by D. Bernoulli about 13 years later. It was detailed by Fisher in 1920. The idea is simple: given an unknown parameter $\theta$ and a sample data $D_N$, the maximum likelihood estimate $\hat{\theta}$ is the value for which the empirical likelihood $L_N(\theta)$ has a maximum

$$\hat{\theta}_{\text{ml}} = \arg\max_{\theta \in \Theta} L_N(\theta)$$

The estimator $\hat{\boldsymbol{\theta}}_{\text{ml}}$ is called the maximum likelihood estimator (m.l.e.). In practice, it is usual to consider the log-likelihood $l_N(\theta)$ instead of $L_N(\theta)$ since, being $\log(\cdot)$ a monotone function, we have

$$\hat{\theta}_{\text{ml}} = \arg\max_{\theta \in \Theta} L_N(\theta) = \arg\max_{\theta \in \Theta} \log(L_N(\theta)) = \arg\max_{\theta \in \Theta} l_N(\theta) \tag{3.8.20}$$

The likelihood function quantifies the relative abilities of the various parameter values to *explain* the observed data. The principle of m.l. is that the value of the parameter under which the obtained data would have had highest probability of arising must be intuitively our best estimator of $\theta$. In other terms the likelihood can be considered a measure of how plausible the parameter values are in light of the data. Note however that the likelihood function is NOT a probability function: for instance, in general, it does not integrate to 1 (with respect to $\theta$).

### Example

Consider a binary variable (e.g. a coin tossing) which takes $z = 15$ times the value 1 (e.g. "Tail") in $N = 40$ trials. Suppose that the probabilistic model underlying the data is Binomial (Section 2.6.2) with an unknown probability $\theta = p$. We want to estimate the unknown parameter $\theta = p \in [0, 1]$ on the basis of the empirical evidence from the $N$ trials. The likelihood $L(p)$ is a function of (only) the unknown parameter $p$. By applying the maximum likelihood technique we have

$$\hat{\theta}_{\text{ml}} = \hat{p} = \arg\max_{p} L(p) = \arg\max_{p} \binom{N}{z} p^z (1-p)^{(N-z)} = \arg\max_{p} \binom{40}{15} p^1 5 (1-p)^{(25)}$$

Figure 3.4 plots $L(p)$ versus $p \in [0, 1]$ (R script `ml_bin.R`). The most likely value

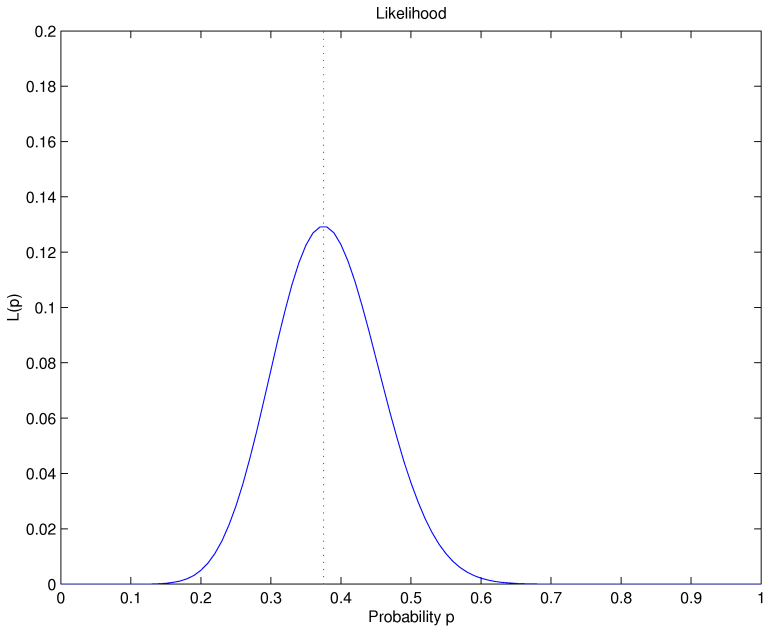



Figure 3.4: Likelihood function

of $p$ is the value where $L(\cdot)$ attains its maximum. According to Figure 3.4 this value is $\hat{p} \approx 0.4$. Note that in this case $\hat{p} = z/N$.

The log-likelihood for this model is

$$\log L(p) = \log \binom{N}{z} + z \log(p) - (N - z) \log(1 - p) =$$
$$= \log \binom{40}{15} + 15 \log p - 25 \log(1 - p)$$

•

### 3.8.1 Maximum likelihood computation

In many situations the log-likelihood $l_N(\theta)$ is particularly well behaved in being continuous with a single maximum away from the extremes of the range of variation of $\theta$. Then $\hat{\theta}_{\text{ml}}$ is obtained simply as the solution of

$$\frac{\partial l_N(\theta)}{\partial \theta} = 0$$

subject to

$$\frac{\partial^2 l_N(\theta)}{\partial \theta^2}|_{\hat{\theta}_{\text{ml}}} < 0$$

to ensure that the identified stationary point is a maximum.

An example is the Gaussian case where it is possible to derive analytically the expression of the maximum likelihood estimators of the mean and variance of $\mathbf{z}$. Let $D_N$ be a random sample from the r.v. $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$. According to (3.8.19), the likelihood of the $N$ samples is given by

$$L_N(\mu, \sigma^2) = \prod_{i=1}^{N} p_{\mathbf{z}}(z_i, \mu, \sigma^2) = \prod_{i=1}^{N} \left( \frac{1}{\sqrt{2\pi}\sigma} \right) \exp \left[ \frac{-(z_i - \mu)^2}{2\sigma^2} \right]$$

and the log-likelihood is

$$l_N(\mu,\sigma^2) = \log L_N(\mu,\sigma^2) = \log\left[\prod_{i=1}^{N} p_{\mathbf{z}}(z_i,\mu,\sigma^2)\right] = \\ = \sum_{i=1}^{N} \log p_{\mathbf{z}}(z_i,\mu,\sigma^2) = -\frac{\sum_{i=1}^{N}(z_i-\mu)^2}{2\sigma^2} + N\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right)$$

Note that, for a given $\sigma$, maximizing the log-likelihood is equivalent to minimize the sum of squares of the difference between $z_i$ and the mean. Taking the derivatives with respect to $\mu$ and $\sigma^2$ and setting them equal to zero, we obtain

$$\hat{\mu}_{\mathrm{ml}} = \frac{\sum_{i=1}^{N} z_i}{N} = \hat{\mu} \tag{3.8.21}$$

$$\hat{\sigma}^2_{\mathrm{ml}} = \frac{\sum_{i=1}^{N}(z_i-\hat{\mu}_{\mathrm{ml}})^2}{N} \neq \hat{\sigma}^2 \tag{3.8.22}$$

Note that the m.l. estimator (3.8.21) of the mean coincides with the sample average (3.3.6) but that the m.l. estimator (3.8.22) of the variance differs from the sample variance (3.3.7) in terms of the denominator.

**Exercise**

- Let $\mathbf{z} \sim \mathcal{U}(0,M)$ and $F_{\mathbf{z}} \to D_N = \{z_1,\ldots,z_N\}$. Find the maximum likelihood estimator of $M$.
- Let $\mathbf{z}$ have a Poisson distribution, i.e.

  $$p_{\mathbf{z}}(z,\lambda) = \frac{e^{-\lambda}\lambda^z}{z!}$$

  If $F_{\mathbf{z}}(z,\lambda) \to D_N = \{z_1,\ldots,z_N\}$, find the m.l.e. of $\lambda$

•

In case of generic distributions $F_{\mathbf{z}}$ computational difficulties may arise. For example no explicit solution might exist for $\partial l_N(\theta)/\partial\theta = 0$. Iterative numerical methods must be used in this case. The computational cost becomes heavier if we consider a vector of parameters instead of a scalar $\theta$ or when there are several relative maxima of the function $l_N$.

Another complex situation occurs when $l_N(\theta)$ is discontinuous, or have a discontinuous first derivative, or a maximum at an extremal point.

**R script**

Suppose we know the analytical form of a one dimensional function $f(x) : I \to \mathbb{R}$ but not the analytical expression of its extreme points. In this case numerical optimization methods can be applied. The implementation of some continuous optimization routines is available in the R statistical tool.

Consider for example the function $f(x) = (x-1/3)^2$ and $I = [0,1]$. The value of the point $x$ where $f$ takes a minimum value can be approximated numerically by this set of R commands

```
f <- function (x,a) (x-a)^2
xmin <- optimize(f, c(0, 1), tol = 0.0001, a = 1/3)
xmin
```

These routines may be applied to solve the problem of maximum likelihood estimation which is nothing more than a particular case of optimization problem. Let $D_N$ be a random sample from the r.v. $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$. The minus log-likelihood function of the $N$ samples can be written in R by

```
eml <- function(m,D,var) {
         N<- length(D)
         Lik<-1
         for (i in 1:N)
          Lik<-Lik*dnorm(D[i],m,sqrt(var))
         -log(Lik)
     }
```

and the numerical minimization of $-l_N(\mu, s^2)$ for a given $\sigma = s$ in the interval $I = [-10, 10]$ can be written in R as

```
xmin<-optimize( eml,c(-10,10),D=DN,var=s)
```

In order to run the above code and compute numerically the m.l. solution we invite the reader to run the R script `emp_ml.R`.

```
# script emp_ml.R
# Script: shows the use of maximum likelihood for parameter estimation


eml <- function(m,D,var) {   ## empirical likelihood function (1 argument)

         N<- length(D)
         Lik<-1
         for (i in 1:N)
           {
          Lik<-Lik*dnorm(D[i],m,sqrt(var))

        }
         -log(Lik)
     }

eml2 <- function(m,D) {   ## empirical likelihood function (2 arguments)
         N<- length(D)
         Lik<-1
         for (i in 1:N)
          Lik<-Lik*dnorm(D[i],m[1],sqrt(max(0,m[2])))
         -log(Lik)
     }

N <-10

DN<-rnorm(N) # data generation

xmin<-optimize( eml,c(-10,10),D=DN,var=1,lower=-1,upper=1)
# maximization of log likelihood function (1 argument)
xmin

xmin2<-optim( c(-10,10),eml2, D=DN)
# maximization of log likelihood function (2 arguments)
```

```
mean(DN)
var(DN)
```

•

### 3.8.2 Properties of m.l. estimators

*Under the (strong) assumption that the probabilistic model structure is known*, the maximum likelihood technique features the following properties:

- $\hat{\boldsymbol{\theta}}_{\rm ml}$ is *asymptotically unbiased* but usually biased in small samples (e.g. $\hat{\sigma}^2_{\rm ml}$).
- $\hat{\boldsymbol{\theta}}_{\rm ml}$ is consistent.
- If $\hat{\boldsymbol{\theta}}_{\rm ml}$ is the m.l.e. of $\theta$ and $\gamma(\cdot)$ is a *monotone* function then $\gamma(\hat{\boldsymbol{\theta}}_{\rm ml})$ is the m.l.e. of $\gamma(\theta)$.
- If $\gamma(\cdot)$ is a non monotonic function, then even if $\hat{\boldsymbol{\theta}}_{\rm ml}$ is an unbiased estimator of $\theta$, the m.l.e. $\gamma(\hat{\boldsymbol{\theta}}_{\rm ml})$ of $\gamma(\theta)$ is usually biased.
- the variance of $\hat{\boldsymbol{\theta}}_{\rm ml}$ is often difficult to determine. For large samples we can use as approximation

$$\left(-E\left[\frac{\partial^2 \boldsymbol{l}_N}{\partial\theta^2}\right]\right)^{-1} \text{ or } \left(-\frac{\partial^2 l_N}{\partial\theta^2}|_{\hat{\theta}_{\rm ml}}\right)^{-1}$$

  A lower bound to the variance of an estimator is established by the Cramer-Rao theorem (see next section).

- $\hat{\boldsymbol{\theta}}_{\rm ml}$ is asymptotically fully efficient, that is

$$\text{Var}\left[\hat{\boldsymbol{\theta}}_{\rm ml}\right] \to [I_N(\theta)]^{-1} = [NI(\theta)]^{-1}, \qquad N \to \infty$$

- $\hat{\boldsymbol{\theta}}_{\rm ml}$ is asymptotically normally distributed, that is

$$\hat{\boldsymbol{\theta}}_{\rm ml} \sim \mathcal{N}(\theta, [I_N(\theta)]^{-1}), \qquad N \to \infty$$

- the score is asymptotically normally distributed

$$\frac{\partial \boldsymbol{l}_N}{\partial\theta} \sim \mathcal{N}(0, I_N(\theta)), \qquad N \to \infty$$

### 3.8.3 Cramer-Rao lower bound

Assume that $\theta$ is a scalar parameter, that the first two derivatives of $L_N(\theta)$ with respect to $\theta$ exist for all $\theta$ and that certain operations of integration and differentiation may be interchanged.

Let $\hat{\boldsymbol{\theta}}$ be an unbiased estimator of $\theta$ and $l_N(\theta) = \log_e[L_N(\theta)]$. Suppose that the regularity condition

$$E\left[\frac{\partial \boldsymbol{l}_N(\theta)}{\partial\theta}\right] = 0 \tag{3.8.23}$$

holds where the quantity $\partial l(\theta)/\partial\theta$ is called *score*. The Cramer-Rao bound is a lower bound to the variance of the estimator $\hat{\boldsymbol{\gamma}}$ which states that

$$\text{Var}\left[\hat{\boldsymbol{\theta}}\right] \geq \frac{1}{E\left[\left(\frac{\partial \boldsymbol{l}_N(\theta)}{\partial\theta}\right)^2\right]} = -\frac{1}{NE\left[\left(\frac{\partial^2 \log p(z,\theta)}{\partial\theta^2}\right)\right]} = \frac{1}{I_N}$$

where the denominator term $I_N$ is known as the Fisher information. An estimator having as variance the right term is called the *Minimum Variance Bound (MVB) estimator*.

**Example**

Consider a r.v. $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$ where $\sigma^2$ is known and the unknown parameter is $\theta = \mu$. Let us consider the bound on the variance of the estimator (3.8.21). Since

$$\frac{\partial \log p(z,\theta)}{\partial \theta} = \frac{z-\theta}{\sigma^2}$$

$$\frac{\partial^2 \log p(z,\theta)}{\partial \theta^2} = -\frac{1}{\sigma^2}$$

It follows that

$$\text{Var}\left[\hat{\boldsymbol{\theta}}\right] \geq \frac{1}{\frac{N}{\sigma^2}} = \frac{\sigma^2}{N}$$

From (3.5.10) it derives then that the m.l. estimator (3.8.21) of the mean $\mu$ is optimal.

•

## 3.9 Interval estimation

Unlike point estimation which is based on a one-to-one mapping from the space of data to the space of parameters, interval estimation maps $D_N$ to an interval of $\Theta$. A point estimator is a function which, given a dataset $D_N$ generated from $F_{\mathbf{z}}(z,\theta)$, returns an estimate of $\theta$. An *interval estimator* is a transformation which, given a dataset $D_N$, returns an interval estimate of $\theta$. While an estimator is a random variable, an interval estimator is a random interval. Let $\underline{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{\theta}}$ be the random lower and the upper bounds respectively. While an interval either contains or not a certain value, a random interval has a certain probability of containing a value. Suppose that

$$\text{Prob}\left\{\underline{\boldsymbol{\theta}} \leq \theta \leq \bar{\boldsymbol{\theta}}\right\} = 1-\alpha \qquad \alpha \in [0,1] \tag{3.9.24}$$

then the random interval $[\underline{\boldsymbol{\theta}}, \bar{\boldsymbol{\theta}}]$ is called a $100(1-\alpha)\%$ confidence interval of $\theta$. If (3.9.24) holds, we expect that by repeating the sampling of $D_N$ and the construction of the confidence interval many times, our confidence interval will contain the true $\theta$ at least $100(1-\alpha)\%$ of the time. Notice, however, that $\theta$ is a fixed unknown value and that at each realization $D_N$ the interval either may or may not contain the true $\theta$. While an estimator is characterized by bias and variance, an interval estimator is characterized by its **endpoints** $\underline{\boldsymbol{\theta}}$ and $\bar{\boldsymbol{\theta}}$ (or its width) and by its *confidence* $\alpha$.

### 3.9.1 Confidence interval of $\mu$

Consider a random sample $D_N$ of a r.v. $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$ where $\sigma^2$ is known. Suppose we want to estimate $\mu$ with the estimator $\hat{\boldsymbol{\mu}}$. From Section 3.7 we have that $\hat{\boldsymbol{\mu}} \sim \mathcal{N}(\mu, \sigma^2/N)$ is Gaussian distributed. From (2.7.43) it follows that

$$\frac{\hat{\boldsymbol{\mu}} - \mu}{\sigma/\sqrt{N}} \sim \mathcal{N}(0,1)$$

and consequently, according to the Definition 4.6

$$\text{Prob}\left\{-z_{\alpha/2} \leq \frac{\hat{\boldsymbol{\mu}} - \mu}{\sigma/\sqrt{N}} \leq z_{\alpha/2}\right\} = 1-\alpha \tag{3.9.25}$$

$$\text{Prob}\left\{\hat{\boldsymbol{\mu}} - z_{\alpha/2}\frac{\sigma}{\sqrt{N}} \leq \mu \leq \hat{\boldsymbol{\mu}} + z_{\alpha/2}\frac{\sigma}{\sqrt{N}}\right\} = 1-\alpha \tag{3.9.26}$$

where $z_\alpha$ is the upper critical point of the standard Gaussian distribution.

It follows that $\underline{\boldsymbol{\theta}} = \hat{\boldsymbol{\mu}} - z_\alpha \sigma/\sqrt{N}$ is a lower $1-\alpha$ confidence bound for $\mu$ while $\bar{\boldsymbol{\theta}} = \hat{\boldsymbol{\mu}} + z_\alpha \sigma/\sqrt{N}$ is an upper $1-\alpha$ confidence bound for $\mu$.

By varying $\alpha$ we can vary the width and the confidence of the interval.

### Example

Let $\mathbf{z} \sim \mathcal{N}(\mu, 0.01)$ and $D_N = \{10, 11, 12, 13, 14, 15\}$. We want to estimate the confidence interval of $\mu$ with level $\alpha = 0.1$. Since $N = 6$, $\hat{\mu} = 12.5$, and

$$\epsilon = z_{\alpha/2}\sigma/\sqrt{N} = 1.645 \cdot 0.01/\sqrt{6} = 0.0672$$

the 90% confidence interval for the given $D_N$ is

$$\{\mu : |\hat{\boldsymbol{\mu}} - \mu| \leq \epsilon\} = \{12.5 - 0.0672 \leq \mu \leq 12.5 + 0.0672\}$$

•

### R script

The R script `confidence.R` allows the test of the formula (3.9.25) by simulation. The user sets $\mu$, $\sigma$, $N$, $\alpha$ and a number of iterations $N_{\text{iter}}$.

```
# script confidence.R

cnt<-1
perc<-NULL
mu<-10  ## mean
sigma<-1 ## stdev
N<-100 ## number of samples
alpha<-0.1
z.alpha<-qnorm(alpha/2, lower=FALSE)

for (N.iter in seq(100,1000,by=10)){
  mu.hat<-array(0,dim=c(N.iter,1))
  ins<-mu.hat
  for ( i in seq(1,N.iter)){
    D<-rnorm(N,mean=mu,sd=sigma);
    mu.hat[i,1]<-mean(D)
    ins[i,1]<-
      (mu.hat[i,1]>(mu-z.alpha*sigma/sqrt(N)))& (mu.hat[i,1]<(mu+z.alpha*sigma/sqrt(N)))
  }

  perc<-cbind(perc,sum(ins)/N.iter)
  cnt<-cnt+1
}

one<-array(1-alpha,dim=c(length(perc),1))

plot(1:length(perc)),one);

lines(1:length(perc),perc)
```

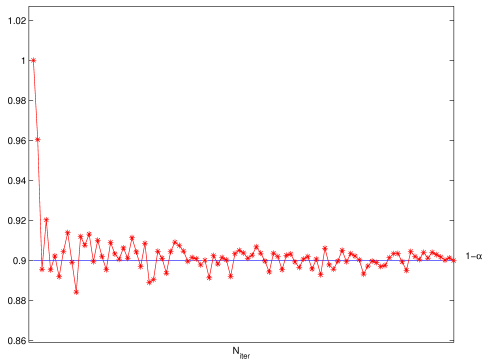



Figure 3.5: Fraction of times that the interval of confidence contains the parameter $\mu$ vs. the number of repetitions for $\alpha = 0.1$

The script generates $N_{\text{iter}}$ times $D_N \sim \mathcal{N}(\mu, \sigma^2)$ and computes $\hat{\mu}$. The script returns the percentage of times that

$$\hat{\mu} - \frac{z_{\alpha/2}\sigma}{\sqrt{N}} < \mu < \hat{\mu} + \frac{z_{\alpha/2}\sigma}{\sqrt{N}}$$

This percentage versus the number of iterations is plotted in Figure 3.5 (R script `confidence.R`). We can easily check that this percentage converges to $100(1-\alpha)\%$ for $N_{\text{iter}} \to \infty$.

•

Consider now the interval of confidence of $\mu$ when the variance $\sigma^2$ is not known. Let $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\sigma}}^2$ be the estimators of $\mu$ and $\sigma^2$ computed on the basis of the i.i.d. dataset $D_N$. From Section 3.7, it follows that

$$\frac{\hat{\mu} - \mu}{\sqrt{\frac{\hat{\sigma}^2}{N}}} \sim \mathcal{T}_{N-1}$$

Analogously to (3.9.26) we have

$$\text{Prob}\left\{\hat{\boldsymbol{\mu}} - t_{\alpha/2}\frac{\sigma}{\sqrt{N}} \leq \mu \leq \hat{\boldsymbol{\mu}} + t_{\alpha/2}\frac{\sigma}{\sqrt{N}}\right\} = 1 - \alpha \tag{3.9.27}$$

where $t_\alpha$ is the upper critical point of the Student distribution.

**Example**

Let $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$, with $\sigma^2$ unknown and $D_N = \{10, 11, 12, 13, 14, 15\}$. We want to estimate the confidence region of $\mu$ with level $\alpha = 0.1$. We have $\hat{\mu} = 12.5$, $\hat{\sigma}^2 = 3.5$. According to (3.9.27) we have

$$\epsilon = t_{\{\alpha/2, N-1\}}\hat{\sigma}/\sqrt{N} = 2.015 * 1.87/\sqrt{6} = 1.53$$

The $(1-\alpha)$ confidence interval of $\mu$ is

$$\hat{\mu} - \epsilon < \mu < \hat{\mu} + \epsilon$$

•

**Example**

We want to estimate $\theta$, the proportion of people who support the politics of Mr. Berlusconi amongst a very large population. We want to define how many interviews are necessary to have a confidence interval of 3% with a significance of 5%. We interview $N$ people and estimate $\theta$ as

$$\hat{\theta} = \frac{x_1 + \cdots + x_N}{N} = \frac{S}{N}$$

where $x_i = 1$ if the $i$th person supports Berlusconi and $x_i = 0$ otherwise. Note that **S** is a binomial variable. We have

$$E[\hat{\boldsymbol{\theta}}] = \theta, \qquad \text{Var}\left[\hat{\boldsymbol{\theta}}\right] = \text{Var}\left[S/N\right] = \frac{N(\theta)(1-\theta)}{N^2} = \frac{\theta(1-\theta)}{N} \leq \frac{1}{4N}$$

If we approximate the distribution of $\hat{\boldsymbol{\theta}}$ by $\mathcal{N}(\theta, \frac{\theta(1-\theta)}{N})$ it follows that $\frac{\hat{\boldsymbol{\theta}}-\theta}{\sqrt{\theta(\theta-1)N}} \sim \mathcal{N}(0,1)$.

The following relation holds

$$\begin{aligned}
&\text{Prob}\left\{\hat{\boldsymbol{\theta}} - 0.03 \leq \theta \leq \hat{\boldsymbol{\theta}} + 0.03\right\} = \\
&\qquad \text{Prob}\left\{-\frac{0.03}{\sqrt{\theta(1-\theta)/N}} \leq \frac{\hat{\boldsymbol{\theta}} - \theta}{\sqrt{\theta(1-\theta)/N}} \leq \frac{0.03}{\sqrt{\theta(1-\theta)/N}}\right\} = \\
&\qquad\qquad \Phi\left(\frac{0.03}{\sqrt{\theta(1-\theta)/N}}\right) - \Phi\left(-\frac{0.03}{\sqrt{\theta(1-\theta)/N}}\right) \geq \\
&\qquad\qquad\qquad \geq \Phi(0.03\sqrt{4N}) - \Phi(-0.03\sqrt{4N})
\end{aligned}$$

In order to have this probability to be at least 0.95 we need $0.03\sqrt{4N} \geq 1.96$ or equivalently $N \geq 1068$.

•

## 3.10 Combination of two estimators

Consider two unbiased estimators $\hat{\boldsymbol{\theta}}_1$ and $\hat{\boldsymbol{\theta}}_2$ of the same parameter $\theta$

$$E[\hat{\boldsymbol{\theta}}_1] = \theta \qquad E[\hat{\boldsymbol{\theta}}_2] = \theta$$

having equal and non zero variance

$$\text{Var}\left[\hat{\boldsymbol{\theta}}_1\right] = \text{Var}\left[\hat{\boldsymbol{\theta}}_2\right] = v$$

and being uncorrelated, i.e. $\text{Cov}[\hat{\boldsymbol{\theta}}_1, \hat{\boldsymbol{\theta}}_2] = 0$. Let $\hat{\boldsymbol{\theta}}_{\text{cm}}$ be the combined estimator

$$\hat{\boldsymbol{\theta}}_{\text{cm}} = \frac{\hat{\boldsymbol{\theta}}_1 + \hat{\boldsymbol{\theta}}_2}{2}$$

This estimator has the nice properties of being unbiased

$$E[\hat{\boldsymbol{\theta}}_{\text{cm}}] = \frac{E[\hat{\boldsymbol{\theta}}_1] + E[\hat{\boldsymbol{\theta}}_2]}{2} = \theta$$

and with a smaller variance than the original estimators

$$\text{Var}\left[\hat{\boldsymbol{\theta}}_{\text{cm}}\right] = \frac{1}{4}\text{Var}\left[\hat{\boldsymbol{\theta}}_1 + \hat{\boldsymbol{\theta}}_2\right] = \frac{\text{Var}\left[\hat{\boldsymbol{\theta}}_1\right] + \text{Var}\left[\hat{\boldsymbol{\theta}}_2\right]}{4} = \frac{v}{2}$$

This trivial computation shows that the simple average of two unbiased estimators with a non zero variance returns a combined estimator with reduced variance.

### 3.10.1 Combination of $m$ estimators

Here, we report the general formula of the linear combination of a number $m$ of estimators. Assume we want to estimate the unknown parameter $\theta$ by combining a set of $m$ estimators $\{\hat{\boldsymbol{\theta}}_j\}$, $j = 1, \ldots, m$. Let

$$E[\hat{\boldsymbol{\theta}}_j] = \mu_j \quad \text{Var}\left[\hat{\boldsymbol{\theta}}_j\right] = v_j \quad \text{Bias}[\hat{\boldsymbol{\theta}}_j] = b_j$$

be the expected values, the variances and the bias of the $m$ estimators, respectively.

We are interested in estimating $\theta$ by forming a linear combination

$$\hat{\boldsymbol{\theta}}_{\text{cm}} = \sum_{j=1}^{m} w_j \hat{\boldsymbol{\theta}}_j = w^T \hat{\boldsymbol{\theta}} \tag{3.10.28}$$

where $\hat{\boldsymbol{\theta}} = [\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_m]^T$ is the vector of estimators and $w = [w_1, \ldots, w_m]^T$ is the weighting vector.

The mean-squared error of the combined system is

$$\begin{aligned}\text{MSE} = E[(\hat{\boldsymbol{\theta}}_{\text{cm}} - \theta)^2] &= E[(w^T\hat{\boldsymbol{\theta}} - E[w^T\hat{\boldsymbol{\theta}}])^2] + (E[w^T\hat{\boldsymbol{\theta}}] - \theta)^2 \\ &= E[(w^T(\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}]))^2] + (w^T\mu - \theta)^2 = \\ &= w^T\Omega w + (w^T\mu - \theta)^2\end{aligned}$$

where $\Omega$ is a $[m \times m]$ covariance matrix whose $ij^{\text{th}}$ term is

$$\Omega_{ij} = E[(\hat{\boldsymbol{\theta}}_i - \mu_i)(\hat{\boldsymbol{\theta}}_j - \mu_j)]$$

and $\mu = (\mu_1, \ldots, \mu_m)^T$ is the vector of expected values. Note that the MSE error has a variance term (dependent on the covariance of the single estimators) and a bias term (dependent on the bias of the single estimators).

#### 3.10.1.1 Linear constrained combination

A commonly used constraint is

$$\sum_{j=1}^{m} w_j = 1, \quad w_j \geq 0, \quad j = 1, \ldots, m \tag{3.10.29}$$

This means that the combined estimator is unbiased if the individual estimators are unbiased. Let us write $w$ as

$$w = (u^T g)^{-1} g$$

where $u = (1, \ldots, 1)^T$ is an $m$-dimensional vector of ones, $g = (g_1, \ldots, g_m)^T$ and $g_j > 0, \forall j = 1, \ldots, m$.

The constraint can be enforced in minimizing the MSE by using the Lagrangian function

$$L = w^T\Omega w + (w^T\mu - \theta)^2 + \lambda(w^T u - 1)$$

with $\lambda$ Lagrange multiplier.

The optimum is achieved if we set

$$g^* = [\Omega + (\mu - \theta u)(\mu - \theta u)^T]^{-1} u$$

With unbiased estimators ($\mu = \theta$) we obtain

$$g^* = \Omega^{-1} u$$

and with uncorrelated estimators

$$g_j^* = \frac{1}{\text{v}_j} \qquad j = 1, \ldots, m \tag{3.10.30}$$

This means that the optimal term $g_j^*$ of each estimator is inversely proportional to its own variance.

## 3.11 Testing hypothesis

*Hypothesis testing* is together with estimation a major area of statistical inference. A *statistical hypothesis* is an assertion or conjecture about the distribution of one or more random variables. A *test* of a statistical hypothesis is a rule or procedure for deciding whether to reject the assertion on the basis of the observed data. The basic idea is formulate some statistical hypothesis and look to see whether the data provides any evidence to reject the hypothesis. Examples of hypothesis tests follow:

- Consider the model of the traffic in the boulevard. Suppose that the measures of the inter-arrival times are $D_N = \{10, 11, 1, 21, 2, \dots\}$ seconds. Can we say that the mean inter-arrival time $\theta$ is different from 10?
- We want to know the effect of a drug on rats' survival to cancer. We randomly divide some rats in two groups and we administrate a drug only to one of them. Is the survival rate of the groups the same?
- Consider the grades of two different school sections. Section A had $\{15, 10, 12, 19, 5, 7\}$. Section B had $\{14, 11, 11, 12, 6, 7\}$. Can we say that Section A had better grades than Section B?
- Consider two protein coding genes and their expression levels in a cell. Are the two genes *differentially expressed* ?

A statistical test is a procedure that aims to answer such questions.

### 3.11.1 Types of hypothesis

We start by declaring the *working (basic, null) hypothesis* $H$ to be tested, in the form $\theta = \theta_0$ or $\theta \in \omega \subset \Theta$, where $\theta_0$ or $\omega$ are *given*.

The hypothesis can be

**simple:** this means that it fully specifies the distribution of the r.v. $\mathbf{z}$.

**composite:** this means that it partially specifies the distribution of $\mathbf{z}$.

For example if $D_N$ is a random sample of size $N$ drawn from $\mathcal{N}(\mu, \sigma^2)$ the hypothesis $H : \mu = \mu_0, \sigma = \sigma_0$, (with $\mu_0$ and $\sigma_0$ known values) is simple while the hypothesis $H : \mu = \mu_0$ is composite since it leaves open the value of $\sigma$ in $(0, \infty)$.

### 3.11.2 Types of statistical test

Suppose we have collected $N$ samples $D_N = \{z_1, \dots, z_N\}$ from a distribution $F_{\mathbf{z}}$ and we have declared a null hypothesis $H$ about $F$. The three most common types of statistical test are:

**Pure significance test:** data $D_N$ are used to assess the inferential evidence against $H$.

**Significance test:** the inferential evidence against $H$ is used to judge whether $H$ is inappropriate. In other words this test produces a decision rule for rejecting or not rejecting $H$.

**Hypothesis test:** data $D_N$ are used to assess the hypothesis $H$ against a specific alternative hypothesis $\bar{H}$. In other words this test produces a rule for rejecting $H$ in favour of $\bar{H}$.

The three tests will be discussed in the following sections.

### 3.11.3 Pure significance test

Consider a null simple hypothesis $H$. Let $t(\mathbf{D}_N)$ be a statistic (i.e. a function of the dataset) such that the larger its value the more it casts doubt on $H$. The quantity $t(\mathbf{D}_N)$ is called *test statistic* or *discrepancy measure*. Suppose that the distribution of $t(\mathbf{D}_N)$ under $H$ is known. This is possible since the function $t(\cdot)$ is fixed by the user and the simple hypothesis $H$ entirely specifies the distribution of $\mathbf{z}$ and consequently the distribution of $t(\mathbf{D}_N)$. Let $t_N = t(D_N)$ the observed value of $t$ calculated on the basis of the sample data $D_N$. Let us define the *p-value* quantity as

$$p = \text{Prob}\{t(\mathbf{D}_N) > t_N | H\} \tag{3.11.31}$$

i.e. the probability of observing a statistic greater than $t_N$ if the hypothesis $H$ were true. Note that in the expression (3.11.31), the term $t(\mathbf{D}_N)$ is a random variable having a known distribution while $t_N$ is a value computed on the basis of the observed dataset.

If the $p$ quantity is small then the sample data $D_N$ are highly inconsistent with $H$ and $p$ (*significance probability or significance level* ) is the measure of such inconsistency. If $p$ is small then either a rare event has occured or perhaps $H$ is not true. In other terms, if $H$ were true, the quantity $p$ would be the proportion of situations where we would observe a degree of inconsistency at least to the extent represented by $t_N$.

Note that $p$ depends on $D_N$ since different $D_N$ would yield different values of $t_N$ and consequently different values of $p \in [0, 1]$. Also, in a frequentist perspective, we cannot say that $p$ is the probability that $H$ is true but rather that $p$ *is the probability that the dataset* $D_N$ *is observed given that* $H$ *is true.*

### 3.11.4 Tests of significance

The test of significance proposes the following decision rule: if $p$ is less than some stated value $\alpha$, we reject $H$. Once a *critical level* $\alpha$ is chosen and the dataset $D_N$ is observed, the rule rejects $H$ at level $\alpha$ if

$$P\{t(\mathbf{D}_N) > t_\alpha | H) = \alpha \tag{3.11.32}$$

This is equivalent to choosing some *critical value* $t_\alpha$ and to reject $H$ if $t_N > t_\alpha$. This implies the existence of two regions in the space of sample data:

**critical region:** this is the set of values of $D_N$

$$S_0 = \{D_N : t(D_N) > t_\alpha\}$$

such that if $D_N \in S_0$ we reject the null hypothesis $H$.

**non-critical region:** this is the set of values of $D_N$ such that there is no reason to reject $H$ on the basis of the level-$\alpha$ test.

The principle is that we will accept $H$ unless we witness some event that has sufficiently small probability of arising when $H$ is true. Note that, given the stochastic setting, for a true $H$ we could still obtain data in $S_0$ and consequently wrongly reject $H$. This can happen with probability

$$\text{Prob}\{\mathbf{D}_N \in S_0\} = \text{Prob}\{t(\mathbf{D}_N) > t_\alpha | H\} \leq \alpha$$

The significance level $\alpha$ provides an upper bound to the maximum probability of incorrectly rejecting $H$. The p-value is the probability that the test statistic is more extreme than its observed value. Note that the p-value changes with the observed data (i.e. it is a random variable) while $\alpha$ is a level fixed by the user.

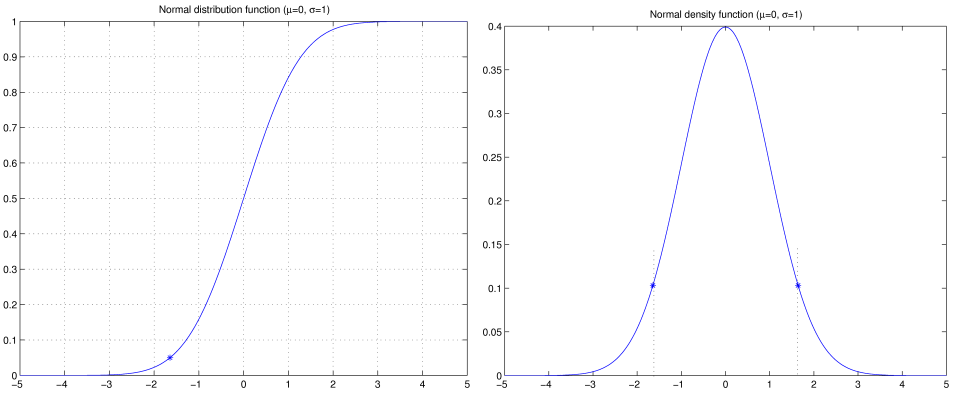



Figure 3.6: Standard normal distribution

**Example**

Let $D_N$ consist of $N$ independent observations of $\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2)$, with known variance $\sigma^2$. We want to test the hypothesis $H : \mu = \mu_0$ with $\mu_0$ known. Consider as test statistic the quantity $t(\mathbf{D}_N) = |\hat{\boldsymbol{\mu}} - \mu_0|$ where $\hat{\boldsymbol{\mu}}$ is the sample average estimator . If $H$ is true we know from Section 3.4 that $\hat{\boldsymbol{\mu}} \sim \mathcal{N}(\mu_0, \sigma^2/N)$. Let us calculate the value $t(D_N) = |\hat{\mu} - \mu_0|$ and fix a significance level $\alpha = 10\%$. This means that the decision rule needs the definition of the value $t_\alpha$ such that

$$\text{Prob}\{t(\mathbf{D}_N) > t_\alpha | H\} = \text{Prob}\{|\hat{\boldsymbol{\mu}} - \mu_0| > t_\alpha | H\} = \\ \text{Prob}\{(\hat{\boldsymbol{\mu}} - \mu_0 > t_\alpha) \cup (\hat{\boldsymbol{\mu}} - \mu_0 < -t_\alpha) | H\} = 0.1$$

For a generic normal variable $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$, from Table 2.2 we have that

$$\text{Prob}\{|\mathbf{z} - \mu| > 1.645\sigma\} = \text{Prob}\left\{\frac{|\mathbf{z} - \mu|}{\sigma} > 1.645\right\} = 2 * 0.05$$

It follows that being $\hat{\boldsymbol{\mu}} \sim \mathcal{N}(\mu_0, \sigma^2/N)$

$$\text{Prob}\left\{|\hat{\boldsymbol{\mu}} - \mu_0| > 1.645\sigma/\sqrt{N}\right\} = 0.05 + 0.05 = 0.1$$

and consequently

$$t_\alpha = 1.645\sigma/\sqrt{N} \tag{3.11.33}$$

The critical region is

$$S_0 = \left\{D_N : |\hat{\mu} - \mu_0| > 1.645\sigma/\sqrt{N}\right\}$$

•

**Example**

Suppose that $\sigma = 0.1$ and that we want to test if $\mu = \mu_0 = 10$ with a significance level 10%. After $N = 6$ observations we have $D_N = \{10, 11, 12, 13, 14, 15\}$. On the basis of the dataset we compute

$$\hat{\mu} = \frac{10 + 11 + 12 + 13 + 14 + 15}{6} = 12.5$$

and

$$t(D_N) = |\hat{\mu} - \mu_0| = 2.5$$

Since according to (3.11.33) $t_\alpha = 1.645 * 0.1/\sqrt{6} = 0.0672$, and $t(D_N) > t_\alpha$, the observations $D_N$ are in the critical region. The conclusion is: *the hypothesis* $H : \mu = 10$ *is rejected* and the probability that we are making an error by rejecting $H$ is smaller than 0.1.

•

### 3.11.5 Hypothesis testing

So far we have dealt with single hypothesis tests. Let us now consider two alternative hypothesis: $H$ and $\bar{H}$. Suppose we have a dataset $\{z_1, \ldots, z_N\} \sim F$ drawn from a distribution $F$. Let $H$ and $\bar{H}$ represent two hypotheses about $F$. On the basis of the dataset, one will be *accepted* and the other one **rejected**. In this case, given the stochastic setting, two type of errors are possible.

**Type I error.** This is the kind of error we make when *we reject $H$ and $H$ is true*. For a given critical level $t_\alpha$ the probability of making this error is

$$\text{Prob}\{t(\mathbf{D}_N) > t_\alpha | H\} \tag{3.11.34}$$

According to (3.11.32) this probability is upper bounded by $\alpha$.

**Type II error.** This is the kind of error we make when *we accept $H$ and $H$ is false*. In order to define this error, we are forced to declare an alternative hypothesis $\bar{H}$ as a formal definition of what is meant by $H$ being "false". The probability of type II error is

$$\text{Prob}\left\{t(\mathbf{D}_N) \leq t_\alpha | \bar{H}\right\} \tag{3.11.35}$$

that is the probability that the test leads to acceptance of $H$ when in fact $\bar{H}$ holds.

Note that

- when the alternative hypothesis is composite, there could be no unique Type II error.
- although $H$ and $\bar{H}$ are complementary events, the quantity (3.11.35) cannot be derived from (3.11.34) (Section 2.1.5)

**Example**

In order to better illustrate these notions, let us consider the analogy with a murder trial, where the suspect is Mr. Bean. The null hypothesis $H$ is "Mr. Bean is innocent". The dataset is the amount of evidence collected by the police against Mr. Bean. The Type I error is the error that we make if, Mr. Bean being innocent, we send him to death-penalty. The Type II error is the error that we make if, being Mr. Bean guilty, we acquit him. Note that the two hypotheses have different philosophical status (asymmetry). $H$ is a conservative hypothesis, not to be rejected unless evidence against Mr Bean's innocence is clear. This means that a type I error is *more serious* than a type II error (*benefit of the doubt*).

•

**Example**

Let us consider a professor who has to decide on the basis of empirical evidence whether a student copied or not during a class test. The null hypothesis $H$ is that the student is honest. The alternative hypothesis $\bar{H}$ is that the student has cheated. Let the empirical evidence $t_N$ be represented by the number of lines of the classwork that a student shares with one of his classmates.

Suppose that a student pass if the professor thinks he has not copied ($t_N < 2$) while he fails otherwise.

•

In general terms a hypothesis testing procedure can be decomposed in the following steps:

1. Declare the null and the alternative hypothesis
2. Choose the numeric value $\alpha$ of the type I error (e.g. the maximal risk I want to run when I reject the null hypothesis).
3. Choose a procedure to obtain test statistic.
4. Determine the critical value of the test statistic that leads to a rejection of $H$ which ensures the Type I error defined in Step 2.
5. Among the set of tests of level $\alpha$, choose the test that minimizes the probability of type II error.
6. Obtain the data and determine whether the observed value of the test statistic leads to an acceptation or rejection of $H$.

Note that a number of tests, having a different type II error, can guarantee the same type I error. An appropriate choice of test as a function of the type II error is therefore required and will be discussed in the following section.

### 3.11.6 Choice of test

The choice of test and consequently the choice of the partition $\{S_0, S_1\}$ is based on two steps

1. Define a significance level $\alpha$, that is the probability of type I error

$$\text{Prob}\{\text{reject } H|H\} = \text{Prob}\{\mathbf{D}_N \in S_0|H\} \leq \alpha$$

   that is the probability of incorrectly rejecting $H$.

2. Among the set of tests $\{S_0, S_1\}$ of level $\alpha$, choose the test that minimizes the probability of type II error

$$\text{Prob}\left\{\text{accept } H|\bar{H}\right\} = \text{Prob}\left\{\mathbf{D}_N \in S_1|\bar{H}\right\}$$

   that is the probability of incorrectly accepting $H$. This is equivalent to maximizing the *power of the test*

$$\text{Prob}\left\{\text{reject } H|\bar{H}\right\} = \text{Prob}\left\{\mathbf{D}_N \in S_0|\bar{H}\right\} = 1 - \text{Prob}\left\{\mathbf{D}_N \in S_1|\bar{H}\right\}$$

   which is the probability of *correctly* rejecting $H$. Note that for a given significance level, the higher the power, the better !

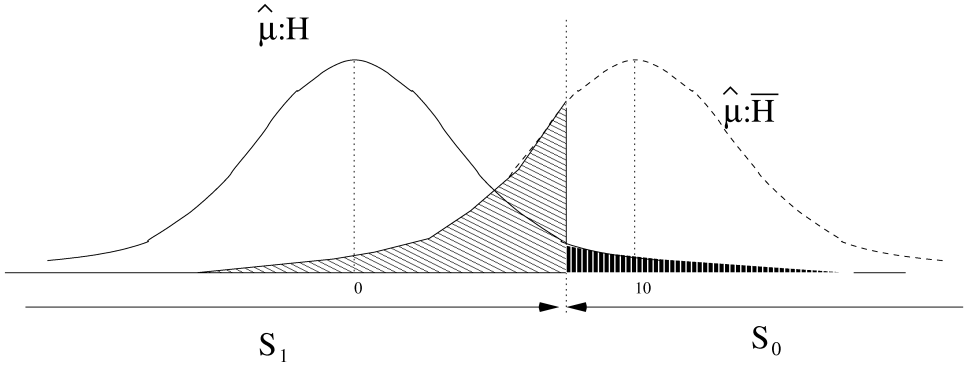



Figure 3.7: On the left: distribution of the test statistic $\hat{\boldsymbol{\mu}}$ if $H : \mu_0 = 0$ is true. On the right: distribution of the test statistic $\hat{\boldsymbol{\mu}}$ if $\bar{H} : \mu_1 = 10$ is true. The interval marked by $S_1$ denotes the set of observed $\hat{\mu}$ values for which $H$ is accepted (non-critical region). The interval marked by $S_0$ denotes the set of observed $\hat{\mu}$ values for which $H$ is rejected (critical region). The area of the black pattern region on the right equals Prob $\{\mathbf{D}_N \in S_0|H\}$, i.e. the probability of rejecting $H$ when $H$ is true (Type I error). The area of the grey shaded region on the left equals the probability of accepting $H$ when $H$ is false (Type II error).

**Example**

In order to reason about the Type II error, let us consider an r.v. $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$, where $\sigma$ is known and a set of $N$ iid observations are given. We want to test the null hypothesis $\mu = \mu_0 = 0$, with $\alpha = 0.1$ Consider three different tests and the associated critical regions $S_0$

1. $|\hat{\mu} - \mu_0| > 1.645\sigma/\sqrt{N}$
2. $\hat{\mu} - \mu_0 > 1.282\sigma/\sqrt{N}$ (Figure 3.7)
3. $|\hat{\mu} - \mu_0| < 0.126\sigma/\sqrt{N}$ (Figure 3.8)

Assume that the area blackened in Figure (3.7) equals the area blackened in Figure (3.8). For all these tests Prob $\{\mathbf{D}_N \in S_0|H\} \leq \alpha$, hence the significance level (i.e. Type I error) is the same. However if $\bar{H} : \mu_1 = 10$ the type II error of the three tests is significantly different. Which test is the best one, that is the one which guarantees the lowest Type II error?

•

### 3.11.7 UMP level-$\alpha$ test

Given a significance level $\alpha$ we denote by *uniformly most powerful (UMP)* test, the test

1. which satisfies
$$\text{Prob}\{\text{reject H}|H\} = \text{Prob}\{\mathbf{D}_N \in S_0|H\} \leq \alpha$$
2. for which
$$\text{Prob}\left\{\text{reject H}|\bar{H}\right\} = \text{Prob}\left\{\mathbf{D}_N \in S_0|\bar{H}\right\}$$
is maximed simultaneously for all $\theta \in \Theta_{\bar{H}}$.

How is it possible to find UMP tests? In a simple case, an answer is given by the Neyman-Pearson lemma.

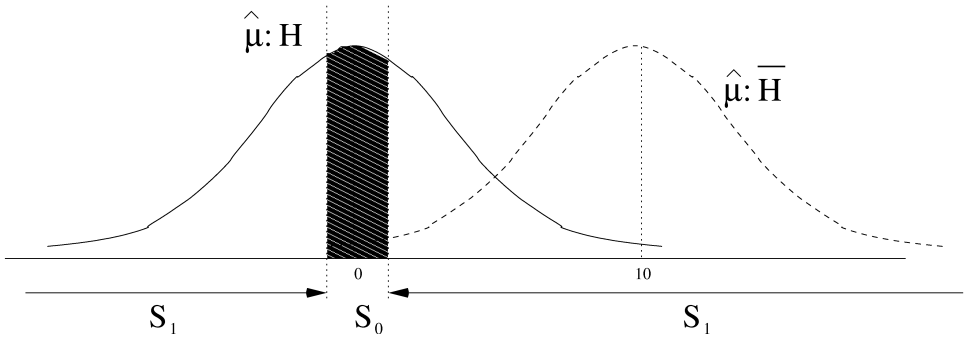



Figure 3.8: On the left: distribution of the test statistic $\hat{\boldsymbol{\mu}}$ if $H : \mu_0 = 0$ is true. On the right: distribution of the test statistic $\hat{\boldsymbol{\mu}}$ if $\bar{H} : \mu_1 = 10$ is true. The two intervals marked by $S_1$ denote the set of observed $\hat{\mu}$ values for which $H$ is accepted (non-critical region). The interval marked by $S_0$ denotes the set of observed $\hat{\mu}$ values for which $H$ is rejected (critical region). The area of the pattern region equals Prob $\{\mathbf{D}_N \in S_0|H\}$, i.e. the probability of rejecting $H$ when $H$ is true (Type I error). Which area corresponds to the probability of the Type II error?

### 3.11.8 Likelihood ratio test

Consider the simplest case $\Theta = \{\theta_0, \theta_1\}$, where $H : \theta = \theta_0$ and $\bar{H} : \theta = \theta_1$ and $\theta_0, \theta_1$ are two different values of the parameter of a r.v. $\mathbf{z}$. Let us denote the two likelihoods by $L_0(\theta)$ and $L_1(\theta)$, respectively.

The idea of Neyman and Pearson was to base the acceptance/rejection of $H$ on the relative values $L(\theta_0)$ and $L(\theta_1)$. In other terms we reject $H$ if the *likelihood ratio*

$$\frac{L(\theta_1)}{L(\theta_0)}$$

is sufficiently big.

We reject $H$ only if the sample data $D_N$ are sufficiently more probable when $\theta = \theta_1$ than when $\theta = \theta_0$.

**Lemma 2** (Neyman-Pearson lemma)**.** Let $H : \theta = \theta_0$ and $\bar{H} : \theta = \theta_1$. If a partition $\{S_0, S_1\}$ of the sample space $\mathcal{D}$ is defined by

$$S_0 = \{D_N : L(\theta_1) > kL(\theta_0)\} \qquad S_1 = \{D_N : L(\theta_1) < kL(\theta_0)\}$$

with $\int_{S_0} p(D_N, \theta_0) dD_N = \alpha$, then $\{S_0, S_1\}$ is the most powerful level-$\alpha$ test of $H$ against $\bar{H}$.

This lemma demonstrates that among all tests of level $\leq \alpha$, the likelihood ratio test is the optimal procedure, i.e. it has the smallest probability of type II error.

Although, for a generic distribution, the definition of an optimum test is very difficult, all the tests that will be described in the following are optimal in the UMP sense.

## 3.12 Parametric tests

Suppose we want to test an assertion about a random variable with a known parametric distribution $F(\cdot, \theta)$. Besides the distinction between simple and composite tests presented in Section 3.11.1, there are two more ways of classifying hypothesis tests:

**One-sample vs. two-sample:** one-sample tests concern an hypothesis about the properties of a single r.v. $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$ while two-sample test concern the relationship between two r.v. $\mathbf{z}_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathbf{z}_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$.

**Single-sided (one-tailed) vs. Two-sided (two-tailed):** in single-sided tests the region of rejection concerns only one tail of the distribution of the null distribution. This means that $\bar{H}$ indicates the predicted direction of the difference. In two-sided tests the region of rejection concerns both tails of the null distribution. This means that $\bar{H}$ does not indicate the predicted direction of the difference.

The most common parametric tests rely on hypothesis of normality. A non-exhaustive list of conventional parametric test is available in the following table:

| Name | single/two sample | known | $H$ | $\bar{H}$ |
|---|---|---|---|---|
| z-test | single | $\sigma^2$ | $\mu = \mu_0$ | $\mu \neq \mu_0$ |
| z-test | two | $\sigma_1^2 = \sigma_2^2$ | $\mu_1 = \mu_2$ | $\mu_1 \neq \mu_2$ |
| t-test | single | | $\mu = \mu_0$ | $\mu \neq \mu_0$ |
| t-test | two | | $\mu_1 = \mu_2$ | $\mu_1 \neq \mu_2$ |
| $\chi^2$-test | single | $\mu$ | $\sigma^2 = \sigma_0^2$ | $\sigma^2 \neq \sigma_0^2$ |
| $\chi^2$-test | single | | $\sigma^2 = \sigma_0^2$ | $\sigma^2 \neq \sigma_0^2$ |
| F-test | two | | $\sigma_1^2 = \sigma_2^2$ | $\sigma_1^2 \neq \sigma_2^2$ |

The columns $H$ and $\bar{H}$ contain the parameter taken into consideration by the test.

All the parametric test procedures can be decomposed into five main steps:

1. Define the null hypothesis and the alternative one.
2. Fix the probability $\alpha$ of having a Type I error.
3. Choose a test statistic $t(D_N)$.
4. Define the critical value $t_\alpha$ that satisfies the Type I error constraint.
5. Collect the dataset $D_N$, compute $t(D_N)$ and decide if the hypothesis is either accepted or rejected.

Note that the first 4 steps are independent of the data and should be carried out *before* the collection of the dataset. A more detailed description of some of these tests is contained in the following sections.

### 3.12.1 z-test (single and one-sided)

Consider a random sample $D_N \leftarrow \mathbf{x} \sim \mathcal{N}(\mu, \sigma^2)$ with $\mu$ unknown and $\sigma^2$ known. Let us see in detail how the five steps of the testing procedure are instantiated in this case.

*STEP 1:*

Consider the null hypothesis and the alternative (composite and one-sided)

$$H : \mu = \mu_0; \qquad \bar{H} : \mu > \mu_0$$

*STEP 2:* fix the value $\alpha$ of the type I error.

*STEP 3:* If $H$ is true then the distribution of $\hat{\boldsymbol{\mu}}$ is $\mathcal{N}(\mu_0, \sigma^2/N)$. This means that the test statistic $t(D_N)$ is

$$t_N = t(D_N) = \frac{(\hat{\boldsymbol{\mu}} - \mu_0)\sqrt{N}}{\sigma} \sim \mathcal{N}(0, 1)$$

*STEP 4:* determine the critical value $t_\alpha$.

We reject the hypothesis $H$ if $t_N > t_\alpha = z_\alpha$ where $z_\alpha$ is such that Prob $\{\mathcal{N}(0,1) > z_\alpha\} = \alpha$.

Example: for $\alpha = 0.05$ we would take $z_\alpha = 1.645$ since 5% of the standard normal distribution lies to the right of 1.645. Note that the value $z_\alpha$ for a given $\alpha$ can be obtained by the R command `qnorm(alpha,lower.tail=FALSE)`.

*STEP 5:* Once the dataset $D_N$ is measured, the value of the test statistic is

$$t_N = \frac{(\hat{\mu} - \mu_0)\sqrt{N}}{\sigma}$$

and the hypothesis is either accepted ($t_N \leq z_\alpha$) or rejected.

**Example z-test**

Consider a r.v. $\mathbf{z} \sim \mathcal{N}(\mu, 1)$. We want to test $H : \mu = 5$ against $\bar{H} : \mu > 5$ with significance level 0.05. Suppose that the dataset is $D_N = \{5.1, 5.5, 4.9, 5.3\}$. Then $\hat{\mu} = 5.2$ and $z_N = (5.2 - 5) * 2/1 = 0.4$. Since this is less than 1.645, we do not reject the null hypothesis.

•

### 3.12.2 t-test: single sample and two-sided

Consider a random sample from $\mathcal{N}(\mu, \sigma^2)$ with $\sigma^2$ *unknown.* Let

$$H : \mu = \mu_0; \qquad \bar{H} : \mu \neq \mu_0$$

Let

$$t(D_N) = t_N = \frac{\sqrt{N}(\hat{\mu} - \mu_0)}{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(z_i - \hat{\mu})^2}} = \frac{(\hat{\mu} - \mu_0)}{\sqrt{\frac{\hat{\sigma}^2}{N}}}$$

a statistic computed using the data set $D_N$.

If the hypothesis $H$ holds, from Sections 2.7.6 and 3.7 it follows that $t(\mathbf{D}_N) \sim \mathcal{T}_{N-1}$ is a r.v. with a Student distribution with $N-1$ degrees of freedom. The size $\alpha$ t-test consists in rejecting $H$ if

$$|t_N| > k = t_{\alpha/2,N-1}$$

where $t_{\alpha/2,N-1}$ is the upper $\alpha$ point of a $\mathcal{T}$-distribution on $N-1$ degrees of freedom, i.e.

$$\text{Prob}\left\{\boldsymbol{t}_{N-1} > t_{\alpha/2,N-1}\right\} = \alpha/2, \quad \text{Prob}\left\{|\boldsymbol{t}_{N-1}| > t_{\alpha/2,N-1}\right\} = \alpha.$$

where $\boldsymbol{t}_{N-1} \sim \mathcal{T}_{N-1}$. In other terms $H$ is rejected when $t_N$ is too large.

Note that the value $t_{\alpha/2,N-1}$ for a given $N$ and $\alpha$ can be obtained by the R command `qt(alpha/2,N-1,lower.tail=TRUE)`.

**Example [41]**

Suppose we want an answer to the following question: *Does jogging lead to a reduction in pulse rate?*. Let us engage eight non jogging volunteers in a one-month jogging programme and let us take their pulses before and after the programme

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| pulse rate before | 74 | 86 | 98 | 102 | 78 | 84 | 79 | 70 |
| pulse rate after | 70 | 85 | 90 | 110 | 71 | 80 | 69 | 74 |
| decrease | 4 | 1 | 8 | -8 | 7 | 4 | 10 | -4 |

Let us assume that the decreases are samples randomly drawn from $\mathcal{N}(\mu, \sigma^2)$ where $\sigma^2$ is unknown. We want to test $H : \mu = \mu_0 = 0$ against $\bar{H} : \mu \neq 0$ with a

significance $\alpha = 0.05$. We have $N = 8$, $\hat{\mu} = 2.75$, $T = 1.263$, $t_{\alpha/2,N-1} = 2.365$ Since $|T| \leq t_{\alpha/2,N-1}$, the data is not sufficient to reject the hypothesis $H$. In other terms the experiment does not provide enough evidence that jogging leads to reduction in pulse rate.

•

### 3.12.3 $\chi^2$-test: single sample and two-sided

Consider a random sample from $\mathcal{N}(\mu, \sigma^2)$ with $\mu$ known. Let

$$H : \sigma^2 = \sigma_0^2; \qquad \bar{H} : \sigma^2 \neq \sigma_0^2$$

Let $\widehat{SS} = \sum_i (z_i - \mu)^2$. From Section 3.7 it follows that if $H$ is true then $\widehat{\mathbf{SS}}/\sigma_0^2 \sim \chi^2_N$ (Section 2.7.5 )

The level $\alpha$ $\chi^2$-test rejects $H$ if $\widehat{SS}/\sigma_0^2 < a_1$ or $\widehat{SS}/\sigma_0^2 > a_2$ where

$$\text{Prob}\left\{\frac{\widehat{\mathbf{SS}}}{\sigma_0^2} < a_1\right\} + \text{Prob}\left\{\frac{\widehat{\mathbf{SS}}}{\sigma_0^2} > a_2\right\} = \alpha$$

A slight modification is necessary if $\mu$ is unknown. In this case you must replace $\mu$ with $\hat{\mu}$ in the quantity $\widehat{SS}$ and use a $\chi^2_{N-1}$ distribution.

### 3.12.4 t-test: two samples, two sided

Consider two r.v.s $\mathbf{x} \sim \mathcal{N}(\mu_1, \sigma^2)$ and $\mathbf{y} \sim \mathcal{N}(\mu_2, \sigma^2)$ with the same variance. Let $D_N^x \leftarrow \mathbf{x}$ and $D_M^y \leftarrow \mathbf{y}$ two independent sets of samples of size $N$ and $M$, respectively..

We want to test $H : \mu_1 = \mu_2$ against $\bar{H} : \mu_1 \neq \mu_2$.

Let

$$\hat{\mu}_x = \frac{\sum_{i=1}^N x_i}{N}, \quad \widehat{SS}_x = \sum_{i=1}^N (x_i - \hat{\mu}_x)^2, \quad \hat{\mu}_y = \frac{\sum_{i=1}^M y_i}{M}, \quad \widehat{SS}_y = \sum_{i=1}^M (y_i - \hat{\mu}_y)^2$$

It can be shown that if $H$ is true then the statistic

$$t(\mathbf{D}_N) = \frac{\hat{\boldsymbol{\mu}}_x - \hat{\boldsymbol{\mu}}_y}{\sqrt{\left(\frac{1}{M} + \frac{1}{N}\right)\left(\frac{\widehat{\mathbf{SS}}_x + \widehat{\mathbf{SS}}_y}{M+N-2}\right)}} \sim \mathcal{T}_{M+N-2}$$

It follows that the test of size $\alpha$ rejects $H$ if

$$|t(D_N)| > t_{\alpha/2,M+N-2}$$

### 3.12.5 F-test: two samples, two sided

Consider a random sample $\{x_1, \ldots, x_M\} \leftarrow \mathbf{x} \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and a random sample $\{y_1, \ldots, y_N\} \leftarrow \mathbf{y} \sim \mathcal{N}(\mu_2, \sigma_2^2)$ with $\mu_1$ and $\mu_2$ unknown. Suppose we want to test

$$H : \sigma_1^2 = \sigma_2^2; \qquad \bar{H} : \sigma_1^2 \neq \sigma_2^2$$

Let us consider the statistic

$$\mathbf{f} = \frac{\hat{\boldsymbol{\sigma}}_1^2}{\hat{\boldsymbol{\sigma}}_2^2} = \frac{\widehat{\mathbf{SS}}_1/(M-1)}{\widehat{\mathbf{SS}}_2/(N-1)} \sim \frac{\sigma_1^2 \chi^2_{M-1}/(M-1)}{\sigma_2^2 \chi^2_{N-1}/(N-1)} = \frac{\sigma_1^2}{\sigma_2^2} F_{M-1,N-1}$$

It can be shown that if $H$ is true, the ratio $\mathbf{f}$ has a F-distribution $F_{M-1,N-1}$ (Section 2.7.7) The F-test rejects $H$ if the ratio $f$ is large, i.e. $f > F_{\alpha,M-1,N-1}$ where

$$\text{Prob}\left\{\mathbf{f} > F_{\alpha,M-1,N-1}\right\} = \alpha$$

if $\mathbf{f} \sim F_{M-1,N-1}$.

So far we assumed that the distribution of the test statistic is known under the null hypothesis. In this case it is possible to fix a priori the Type I error. But what about if we do not know anything about the distribution? Is it possible to assess a posteriori the quality (in terms of errors of Type I or II) of a certain test (e.g. using a certain threshold) ?

## 3.13 A posteriori assessment of a test

Let us consider the professor example and the hypothesis test strategy which leads to the refusal of a student when $t_N < 2$. Suppose that $N$ students took part in the exam and that $N_0$ did not copy while $N_1$ copied. According to the professor $\hat{N}_N$ were considered not guilty and passed the exam, while $\hat{N}_P$ were considered guilty and rejected. It happens that $F_P$ honest students failed and $F_N$ dishonest students passed. Note that the honest students who failed indeed did not copy but they had by chance more than one line in common with a classmate. At the same time there are dishonest student who succeeded by copying but who were clever enough to avoid identical lines.

The resulting situation can be summarized in this table where we associated the null hypothesis $H$ to the minus sign (lack of guilty) and the hypothesis $\bar{H}$ to the plus sign.

| | Passed | Failed | |
|---|---|---|---|
| $H$: Not guilty student (-) | $T_N$ | $F_P$ | $N_N = T_N + F_P$ |
| $\bar{H}$: Guilty student (+) | $F_N$ | $T_P$ | $N_P = F_N + T_P$ |
| | $\hat{N}_N = T_N + F_N$ | $\hat{N}_P = F_P + T_P$ | $N$ |

In this table $F_P$ is the number of False Positives, i.e. the number of times that the professor predicted the student as guilty (+) but in reality he was innocent (-). The ratio $F_P/N$ represents an estimate of the type I error (probability of rejecting the null hypothesis when it is true). The term $F_N$ represents the number of False Negatives, i.e. the number of times that the professor predicted the student as honest (-) but in reality he had copied (+). The ratio $F_N/N$ is an estimation of the type II error (probability of accepting the null hypothesis when it is false).

The previous figures make possible the definition of several quantities concerning the quality of the test.

#### 3.13.0.1 Specificity and sensitivity

**Specificity:** the ratio (to be maximized)

$$SP = \frac{T_N}{F_P + T_N} = \frac{T_N}{N_N} = \frac{N_N - F_P}{N_N}, \qquad 0 \leq SP \leq 1$$

It increases by reducing the number of false positive.

**Sensitivity:** the ratio (to be maximized)

$$SE = \frac{T_P}{T_P + F_N} = \frac{T_P}{N_P} = \frac{N_P - F_N}{N_P}, \quad 0 \leq SE \leq 1$$

This quantity is also known as the True Positive Rate (TPR) and increases by reducing the number of false negatives.

There exists a trade-off between these two quantities.

- In the case of a test who return always $H$ (e.g. very kind professor) we have $\hat{N}_P = 0$,$\hat{N}_N = N$, $F_P = 0$, $T_N = N_N$ and $SP = 1$ but $SE = 0$.
- In the case of a test who return always $\bar{H}$ (e.g. very suspicious professor) we have $\hat{N}_P = N$,$\hat{N}_N = 0$, $F_N = 0$, $T_P = N_P$ and $SE = 1$ but $SP = 0$.

Othere related quantities are:

**Positive Predictive value:** the ratio(to be maximized)

$$PPV = \frac{T_P}{T_P + F_P} = \frac{T_P}{\hat{N}_P}, \qquad 0 \leq PPV \leq 1$$

**Negative Predictive value:** the ratio (to be maximized)

$$PNV = \frac{T_N}{T_N + F_N} = \frac{T_N}{\hat{N}_N}, \qquad 0 \leq PNV \leq 1$$

**False Discovery Rate:** the ratio (to be minimized)

$$FDR = \frac{F_P}{T_P + F_P} = \frac{F_P}{\hat{N}_P} = 1 - PPV, \qquad 0 \leq FDR \leq 1$$

**False Positive Rate:**

$$FPR = 1 - SP = 1 - \frac{T_N}{F_P + T_N} == \frac{F_P}{F_P + T_N}, \qquad 0 \leq FPR \leq 1$$

It decreases by reducing the number of false positive.

**False Negative Rate**

$$FNR = 1 - SE = 1 - \frac{T_P}{T_P + F_N} = \frac{F_N}{T_P + F_N} \quad 0 \leq FPR \leq 1$$

It decreases by reducing the number of false negative.

### 3.13.1 Receiver Operating Characteristic curve

The values $FP$, $TP$,$FN$ and $TN$ are dependent on the threshold used in the decision strategy. It is important then to monitor how these values change according to the value of the threshold . The Receiver Operating Characteristic (also known as ROC curve) is a plot of the true positive rate (i.e. sensitivity) against the false positive rate for the different possible decision thresholds of a test (Figure 3.9).

Consider an example where $t^+ \sim \mathcal{N}(1,1)$ and $t^- \sim \mathcal{N}(-1,1)$. Suppose that the examples are classed as positive if $t > THR$ and negative if $t < THR$, where THR is a threshold.

- If $THR = -\infty$, all the examples are classed as positive: $TN = FN = 0$ which implies $SE = 1$ and $FPR = 1$.
- If $THR = \infty$, all the examples are classed as negative: $TP = FP = 0$ which implies $SE = 0$ and $FPR = 0$.

The relative ROC curve is plotted in Figure 3.9.

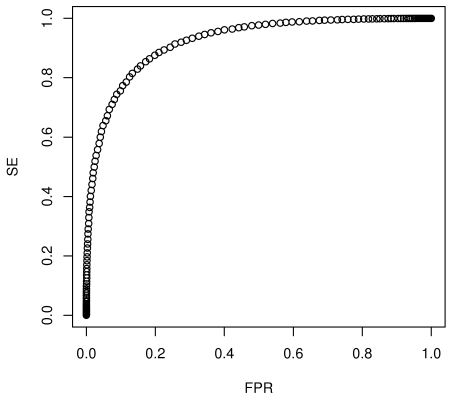



Figure 3.9: ROC curve (see the R script `roc.R`)

# Chapter 4

# Nonparametric approaches to estimation and testing

## 4.1 Nonparametric methods

In the previous two chapters we considered estimation and testing problems concerning random variables with some known probability distribution (e.g. normal) but where one or more parameters were unknown (e.g. mean and/or variance). The estimation and testing methods we used are called **parametric methods**. The meaningfulness of the results of a parametric test depends entirely on the validity of the assumptions made about the analytical form of the distribution. However, in real configurations, it is not uncommon for the experimenter to question paramatric assumptions.

Consider a random sample $D_N \leftarrow \mathbf{z}$ collected through some experimental observation and for which no hint about the underlying probability distribution $F_{\mathbf{z}}(\cdot)$ is available. Suppose we want to estimate a parameter of interest $\theta$ of the distribution of $\mathbf{z}$ by using the estimate $\hat{\theta} = t(D_N)$. What can we say about the accuracy of the estimator $\hat{\boldsymbol{\theta}}$? As shown in Section 3.5.2, for some specific parameters (e.g. mean and variance) the accuracy can be estimated independently of the parametric distribution. In most cases, however, the assessment of the estimator is not possible unless we know the underlying distribution. What to do, hence, if the distribution is not available? A solution is provided by the so-called **nonparametric** or **distribution-free** methods that work independently on any specific assumption about the probability distribution.

The adoption of these methods recently enjoyed a considerable success thanks to the big steps forward made by computers in terms of processing power. In fact, most techniques for nonparametric estimation and testing are based on *resampling procedures* which require a large number of repeated (and almost similar) computations on the data.

This chapter will deal with two resampling strategies for estimation and two resampling strategies for hypothesis testing, respectively.

**Jacknife:** this approach to nonparametric estimation relies on repeated computations of the statistic of interest for all the combinations of the data where one or more of the original samples are removed. It will be presented in Section 4.3.

**Bootstrap:** this approach to nonparametric estimation attempts to estimate the sampling distribution of a population by generating new samples by drawing

(with replacement) from the original data. It will be introduced in Section 4.4.

**Randomization:** This is a resampling without replacement testing procedure. It consists in taking the original data and either scrambling the order or the association of the original data. It will be discussed in Section 4.6

**Permutation:** This is a resampling two-sample hypothesis testing procedure based on repeated permutations of the dataset. It will be presented in Section 4.7.

## 4.2 Estimation of arbitrary statistics

Consider a set $D_N$ of $N$ data points sampled from a scalar r.v. $\mathbf{z}$. Suppose that our quantity of interest is the mean of $\mathbf{z}$. It is straightforward to derive the bias and the variance of the estimator $\hat{\boldsymbol{\mu}}$ (Section 3.3.1):

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} z_i, \qquad \text{Bias}[\hat{\boldsymbol{\mu}}] = 0, \qquad \text{Var}\left[\hat{\boldsymbol{\mu}}\right] = \frac{\sigma^2}{N}$$

Consider now another quantity of interest, for example the median or a mode of the distribution. While it is easy to design a plug-in estimate of these quantities, their accuracy is difficult to be computed. In other terms, *given an arbitrary estimator* $\hat{\boldsymbol{\theta}}$*, the analytical form of the variance* $Var\left[\hat{\boldsymbol{\theta}}\right]$ *and the bias* $Bias[\hat{\boldsymbol{\theta}}]$ *is typically not available.*

**Example**

Suppose we want to estimate the skewness (2.3.31) of a random variable $\mathbf{z}$ on the basis of a collected dataset $D_N$. According to Section 3.3 a plug-in estimator could be

$$\hat{\gamma} = \frac{\frac{1}{N}\sum_i (\mathbf{z}_i - \hat{\mu})^3}{\hat{\sigma}^3}$$

What about the accuracy (e.g. bias, variance) of this estimator?

•

**Example**

Let us consider an example of estimation taken from a medical experimental study [41]. The goal of the study is to show bioequivalence between an old and a new version of a patch designed to infuse a certain hormone in the blood. Eight subjects take part to the study. Each subject has his hormone levels measured after wearing three different patches: a placebo, an "old" patch and a "new" patch. It is established by the Food and Drug Administration (FDA) that the new patch will be approved for sale only if the new patch is bioequivalent to the old one according to the following criterion:

$$\theta = \frac{|E(\text{new}) - E(\text{old})|}{E(\text{old}) - E(\text{placebo})} \leq 0.2 \tag{4.2.1}$$

Let us consider the following plug-in estimator of (4.2.1)

$$\hat{\boldsymbol{\theta}} = \frac{|\hat{\boldsymbol{\mu}}_{\text{new}} - \hat{\boldsymbol{\mu}}_{\text{old}}|}{\hat{\boldsymbol{\mu}}_{\text{old}} - \hat{\boldsymbol{\mu}}_{\text{placebo}}}$$

Suppose we have collected the following data (details in [41])

| subj | plac | old | new | z=old-plac | y=new-old |
|---|---|---|---|---|---|
| 1 | 9243 | 17649 | 16449 | 8406 | -1200 |
| 2 | 9671 | 12013 | 14614 | 2342 | 2601 |
| 3 | 11792 | 19979 | 17274 | 8187 | -2705 |
| ... | ... | ... | ... | ... | ... |
| 8 | 18806 | 29044 | 26325 | 10238 | -2719 |
| mean: | | | | 6342 | -452.3 |

The estimate is

$$\hat{\theta} = t(\hat{F}) = \frac{|\hat{\mu}_{\text{new}} - \hat{\mu}_{\text{old}}|}{\hat{\mu}_{\text{old}} - \hat{\mu}_{\text{placebo}}} = \frac{|\hat{\mu}_y|}{\hat{\mu}_z} = \frac{452.3}{6342} = 0.07$$

Can we say on the basis of this value that the new patch satisfies the FDA criterion in (4.2.1)? What about the accuracy, bias or variance of the estimator? The techniques introduced in the following sections may provide an answer to these questions.

•

## 4.3 Jacknife

The **jacknife** (or **leave-one-out**) resampling technique aims at providing a computational procedure to estimate the variance and the bias of a generic estimator $\hat{\boldsymbol{\theta}}$. The technique was first proposed by Quenouille in 1949 and is based on removing samples from the available dataset and recalculating the estimator. It is a general-purpose tool which is easy to implement and to solve a number of estimation problems.

### 4.3.1 Jacknife estimation

In order to show the theoretical foundation of the jacknife, we first apply this technique to the estimator $\hat{\boldsymbol{\mu}}$ of the mean. Let $D_N = \{z_1, \ldots, z_N\}$ be the available dataset. Let us remove the $i$th sample from $D_N$ and let us calculate the **leave-one-out (l-o-o) mean estimate** from the $N-1$ remaining samples

$$\hat{\mu}_{(i)} = \frac{1}{N-1} \sum_{j \neq i}^{N} z_j = \frac{N\hat{\mu} - z_i}{N-1}$$

Observe from above that the following relation holds

$$z_i = N\hat{\mu} - (N-1)\hat{\mu}_{(i)} \tag{4.3.2}$$

that is, we can calculate the i-th sample $z_i$, $i = 1, \ldots, N$ if we know both $\hat{\mu}$ and $\hat{\mu}_{(i)}$. Suppose now we wish to estimate some parameter $\theta$ by using as estimator some complex statistic of the $N$ data points

$$\hat{\theta} = f(D_N) = f(z_1, z_2, \ldots, z_N)$$

The jacknife procedure consists in first computing

$$\hat{\theta}_{(i)} = f(z_1, z_2, \ldots, z_{i-1}, z_{i+1}, \ldots, z_N), \quad i = 1, \ldots, N$$

which is called the $i$th **jacknife replication** of $\hat{\theta}$. Then by analogy with the relation (4.3.2) holding for the mean estimator, we define the $i$-th **pseudovalue** by

$$\eta_{(i)} = N\hat{\theta} - (N-1)\hat{\theta}_{(i)}. \tag{4.3.3}$$

These pseudovalues assume the same role as the $z_i$ in calculating the sample average (3.3.6). Hence the **jacknife estimate** of $\theta$ is given by

$$\hat{\theta}_{\text{jk}} = \frac{1}{N}\sum_{i=1}^{N}\eta_{(i)} = \frac{1}{N}\sum_{i=1}^{N}\left(N\hat{\theta} - (N-1)\hat{\theta}_{(i)}\right) = N\hat{\theta} - (N-1)\hat{\theta}_{(\cdot)} \tag{4.3.4}$$

where

$$\hat{\theta}_{(\cdot)} = \frac{\sum_{i=1}^{N}\hat{\theta}_{(i)}}{N}.$$

The rationale of the jacknife technique is to use the quantity (4.3.4) in order to estimate the bias of the estimator. Since, according to (3.5.8), $\theta = E[\hat{\boldsymbol{\theta}}] - \text{Bias}[\hat{\boldsymbol{\theta}}]$, the jacknife approach consists in replacing $\theta$ by $\hat{\theta}_{\text{jk}}$ and $E[\hat{\boldsymbol{\theta}}]$ by $\hat{\theta}$, thus obtaining

$$\hat{\theta}_{\text{jk}} = \hat{\theta} - \text{Bias}_{\text{jk}}[\hat{\boldsymbol{\theta}}].$$

It follows that the **jacknife estimate of the bias of $\hat{\boldsymbol{\theta}}$** is

$$\text{Bias}_{\text{jk}}[\hat{\boldsymbol{\theta}}] = \hat{\theta} - \hat{\theta}_{\text{jk}} = \hat{\theta} - N\hat{\theta} + (N-1)\hat{\theta}_{(\cdot)} = (N-1)(\hat{\theta}_{(\cdot)} - \hat{\theta}).$$

Note that in the particular case of a mean estimator (i.e. $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\mu}}$), we see that we obtain, as expected, $\text{Bias}_{\text{jk}}[\hat{\boldsymbol{\mu}}] = 0$.

A jacknife estimate of the variance of $\hat{\boldsymbol{\theta}}$ can be obtained from the sample variance of the pseudo-values. We define the **jacknife estimate of the variance of $\hat{\boldsymbol{\theta}}$** as

$$\text{Var}_{\text{jk}}[\hat{\boldsymbol{\theta}}] = \text{Var}\left[\hat{\boldsymbol{\theta}}_{\text{jk}}\right] \tag{4.3.5}$$

Under the hypothesis of i.i.d. $\eta_{(i)}$

$$\text{Var}\left[\hat{\boldsymbol{\theta}}_{\text{jk}}\right] = \text{Var}\left[\frac{\sum_{i=1}^{N}\boldsymbol{\eta}_{(i)}}{N}\right] = \frac{\text{Var}\left[\boldsymbol{\eta}_{(i)}\right]}{N}$$

From (4.3.3) we have

$$\frac{\sum_{i=1}^{N}\eta_{(i)}}{N} = N\hat{\theta} - \frac{(N-1)}{N}\sum_{i=1}^{N}\hat{\theta}_{(i)}$$

Since

$$\eta_{(i)} = N\hat{\theta} - (N-1)\hat{\theta}_{(i)} \Leftrightarrow \eta_{(i)} - \frac{\sum_{i=1}^{N}\eta_{(i)}}{N} = -(N-1)\left(\hat{\theta}_{(i)} - \frac{\sum_{i=1}^{N}\hat{\theta}_{(i)}}{N}\right)$$

from (4.3.5) and (4.3.4) we obtain

$$\text{Var}_{\text{jk}}[\hat{\boldsymbol{\theta}}] = \frac{\sum_{i=1}^{N}\left(\eta_{(i)} - \hat{\theta}_{\text{jk}}\right)^2}{N(N-1)} = \left(\frac{N-1}{N}\sum_{i=1}^{N}\left(\hat{\theta}_{(i)} - \hat{\theta}_{(\cdot)}\right)^2\right)$$

Note that in the case of the estimator of the mean (i.e. $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\mu}}$), since $\eta_{(i)} = z_i$ and $\hat{\theta}_{\text{jk}} = \hat{\mu}$, we find again the result (3.5.10)

$$\text{Var}_{\text{jk}}[\hat{\boldsymbol{\theta}}] = \frac{\sum_{i=1}^{N}(z_i - \hat{\mu})^2}{N(N-1)} = \frac{\hat{\sigma}^2}{N} = \text{Var}\left[\hat{\boldsymbol{\mu}}\right] \tag{4.3.6}$$

The major motivation for jacknife estimates is that they reduce bias. Also, it can be shown that under suitable conditions on the type of estimator $\hat{\boldsymbol{\theta}}$, the quantity (4.3.6) converges in probability to $\text{Var}\left[\hat{\boldsymbol{\theta}}\right]$. However, the jacknife can fail if the statistic $\hat{\boldsymbol{\theta}}$ is not smooth (i.e. small changes in data cause small changes in the statistic). An example of non-smooth statistic for which the jacknife works badly is the median.

## 4.4 Bootstrap

The method of **bootstrap** was proposed by Efron [39] as a computer-based technique to estimate the accuracy of a generic estimator $\hat{\boldsymbol{\theta}}$.

Bootstrap relies on a data-based simulation method for statistical inference. The use of the term *bootstrap* derives from the phrase *to pull oneself up by one's bootstrap*, widely thought to be based on one of the eighteenth century Adventures of Baron Munchausen, by R.E. Raspe. The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps[1].

The idea of bootstrap is very simple, namely that in absence of any other information, the sample itself offers the best guide of the sampling distribution. The method is completely automatic, requires no theoretical calculation, and is available no matter how mathematically complicated the estimator $\hat{\boldsymbol{\theta}}$ is. By resampling with replacement from $D_N$ we can build a set of $B$ datasets $D_{(b)}, b = 1, \ldots, B$. From the empirical distribution of the statistics $t(D_{(b)})$ we can construct confidence intervals and tests for significance.

### 4.4.1 Bootstrap sampling

Consider a data set $D_N$. A **bootstrap data set** $D_{(b)}$, $b = 1, \ldots, B$ is created by randomly selecting $N$ points from the original set $D_N$ **with replacement** (Figure 4.1).

Since $D_N$ itself contains $N$ points there is nearly always duplication of individual points in a bootstrap data set. Each point has equal probability $1/N$ of being chosen on each draw. Hence, the probability that a point is chosen exactly $k$ times is given by the binomial distribution (Section 2.6.2)

$$\text{Prob}\{k\} = \frac{N!}{k!(N-k)!}\left(\frac{1}{N}\right)^k\left(\frac{N-1}{N}\right)^{N-k} \qquad 0 \le k \le N$$

Given a set of $N$ distinct values, there is a total of $\binom{2N-1}{N}$ distinct bootstrap datasets. The number is quite large already for $N > 10$. For example, if $N = 3$ and $D_N = \{a, b, c\}$, we have 10 different bootstrap sets: {a,b,c},{a,a,b},{a,a,c},{b,b,a},{b,b,c}, {c,c,a},{c,c,b},{a,a,a},{b,b,b},{c,c,c}.

Under **balanced bootstrap sampling** the $B$ bootstrap sets are generated in such a way that each original data point is present exactly $B$ times in the entire collection of bootstrap samples.

### 4.4.2 Bootstrap estimate of the variance

For each bootstrap dataset $D_{(b)}$, $b = 1, \ldots, B$, we can define a **bootstrap replication**

$$\hat{\theta}_{(b)} = t(D_{(b)}) \qquad b = 1, \ldots, B$$

that is the value of the statistic for the specific bootstrap sample. The bootstrap approach computes the variance of the estimator $\hat{\boldsymbol{\theta}}$ through the variance of the set $\hat{\theta}_{(b)}$, $b = 1, \ldots, B$, given by

$$\text{Var}_{\text{bs}}[\hat{\boldsymbol{\theta}}] = \frac{\sum_{b=1}^{B}(\hat{\theta}_{(b)} - \hat{\theta}_{(\cdot)})^2}{(B-1)} \quad \text{where} \quad \hat{\theta}_{(\cdot)} = \frac{\sum_{b=1}^{B}\hat{\theta}_{(b)}}{B} \tag{4.4.7}$$

[1] This term has not the same meaning (though the derivation is similar) as the one used in computer operating systems where bootstrap stands for starting a computer from a set of core instructions

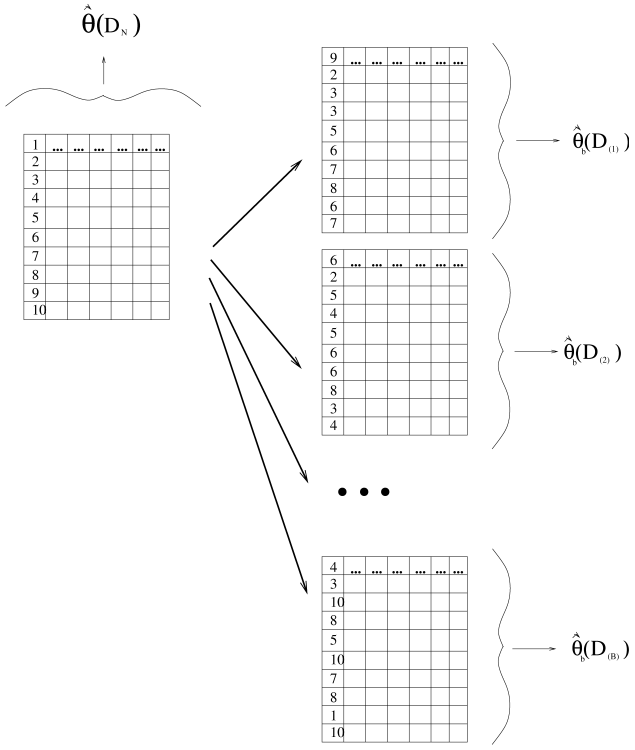



Figure 4.1: Bootstrap replications of a dataset and boostrap statistic computation

It can be shown that if $\hat{\theta} = \hat{\mu}$, then for $B \to \infty$, the bootstrap estimate $\mathrm{Var}_{\mathrm{bs}}[\hat{\boldsymbol{\theta}}]$ converges to the variance $\mathrm{Var}\,[\hat{\boldsymbol{\mu}}]$.

### 4.4.3 Bootstrap estimate of bias

Let $\hat{\boldsymbol{\theta}}$ be the estimator based on the original sample $D_N$ and

$$\hat{\theta}_{(\cdot)} = \frac{\sum_{b=1}^{B} \hat{\theta}_{(b)}}{B}$$

Since $\mathrm{Bias}[\hat{\boldsymbol{\theta}}] = E[\hat{\boldsymbol{\theta}}] - \theta$, the **bootstrap estimate of the bias of the estimator $\hat{\boldsymbol{\theta}}$** is obtained by replacing $E[\hat{\boldsymbol{\theta}}]$ with $\hat{\theta}_{(\cdot)}$ and $\theta$ with $\hat{\theta}$:

$$\mathrm{Bias}_{\mathrm{bs}}[\hat{\boldsymbol{\theta}}] = \hat{\theta}_{(\cdot)} - \hat{\theta}$$

Then, since

$$\theta = E[\hat{\boldsymbol{\theta}}] - \mathrm{Bias}[\hat{\boldsymbol{\theta}}]$$

the **bootstrap bias corrected** estimate is

$$\hat{\theta}_{\mathrm{bs}} = \hat{\theta} - \mathrm{Bias}_{\mathrm{bs}}[\hat{\boldsymbol{\theta}}] = \hat{\theta} - (\hat{\theta}_{(\cdot)} - \hat{\theta}) = 2\hat{\theta} - \hat{\theta}_{(\cdot)}$$

**R script**

See R file `patch.R` for the estimation of bias and variance in the case of the patch data example.

```
## script patch.R

placebo<-c(9243,9671,11792,13357,9055,6290,12412,18806)
oldpatch<-c(17649,12013,19979,21816,13850,9806,17208,29044)
newpatch<-c(16449,14614,17274,23798,12560,10157,16570,26325)

data<-data.frame(placebo,oldpatch,newpatch)

N<-nrow(data)

B<-400  ## number of bootstrap repetitions

theta.hat<-abs(mean(data[,"newpatch"])-mean(data[,"oldpatch"]))/
  (mean(data[,"oldpatch"])-mean(data[,"placebo"]))

thetaB<-numeric(B)
for (b in 1:B){
  Ib<-sample(N,N,replace=TRUE)  ## sampling with replacement
  Db<-data[Ib,]
  thetaB[b]<-(mean(Db[,"newpatch"])-mean(Db[,"oldpatch"]))/
    (mean(Db[,"oldpatch"])-mean(Db[,"placebo"]))
}



hist(thetaB,
     main=paste("Bias=", round(abs(theta.hat-mean(thetaB)),2),
       "; Stdev=", round(sd(thetaB),2)))
abline(v=theta.hat,col="red")

print(paste("Probability that theta.hat >0.2=",sum(thetaB>0.2)/B))
```

•

## 4.5 Bootstrap confidence interval

Standard bootstrap confidence limits are based on the assumption that the estimator $\hat{\boldsymbol{\theta}}$ is normally distributed with mean $\theta$ and variance $\sigma^2$. Taking the bootstrap estimate of variance, an approximate $100(1-\alpha)\%$ confidence interval is given by

$$\hat{\theta} \pm z_{\alpha/2}\sqrt{\mathrm{Var}_{\mathrm{bs}}[\hat{\boldsymbol{\theta}}]} = \hat{\theta} \pm z_{\alpha/2}\sqrt{\frac{\sum_{b=1}^{B}(\hat{\theta}_{(b)} - \hat{\theta}_{(\cdot)})}{(B-1)}} \tag{4.5.8}$$

An improved interval is given by using the bootstrap correction for bias

$$2\hat{\theta} - \hat{\theta}_{(\cdot)} \pm z_{\alpha/2}\sqrt{\frac{\sum_{b=1}^{B}(\hat{\theta}_{(b)} - \hat{\theta}_{(\cdot)})}{(B-1)}} \tag{4.5.9}$$

Another bootstrap approach for constructing a $100(1-\alpha)\%$ confidence interval is to use the upper and lower $\alpha/2$ values of the bootstrap distribution. This approach is referred to as *bootstrap percentile confidence interval*. If $\hat{\theta}_{L,\alpha/2}$ denotes the value such that only a fraction $\alpha/2$ of all bootstrap estimates are inferior to it, and

likewise $\hat{\theta}_{H,\alpha/2}$ is the value exceeded by only $\alpha/2$ of all bootstrap estimates, then the confidence interval is given by

$$[\hat{\theta}_{L,\alpha/2}, \hat{\theta}_{H,\alpha/2}] \tag{4.5.10}$$

where the two extremes are also called the *Efron's percentile confidence limits*.

### 4.5.1 The bootstrap principle

Given an unknown parameter $\theta$ of a distribution $F_{\mathbf{z}}$ and an estimator $\hat{\boldsymbol{\theta}}$, the goal of any estimation procedure is to derive or approximate the distribution of $\hat{\boldsymbol{\theta}} - \theta$. For example the calculus of the variance of $\hat{\boldsymbol{\theta}}$ requires the knowledge of $F_{\mathbf{z}}$ and the computation of $E_{\mathbf{D}_N}[(\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}])^2]$. Now, in practical contexts, $F_{\mathbf{z}}$ is unknown and the calculus of $E_{\mathbf{D}_N}[(\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}])^2]$ is not possible in an analytical way. The rationale of the bootstrap approach is (i) to replace $F_{\mathbf{z}}$ by the empirical counterpart (3.2.1) and (ii) to compute $E_{\mathbf{D}_N}[(\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}])^2]$ by a Monte Carlo simulation approach where several samples of size $N$ are generated by resampling $D_N$.

The outcome of a bootstrap technique is a Monte Carlo approximation of the distribution $\hat{\boldsymbol{\theta}}_{(b)} - \hat{\theta}$. In other terms the variability of $\hat{\boldsymbol{\theta}}_{(b)}$ (based on the empirical distribution) around $\hat{\theta}$ is expected to be similar (or mimic) the variability of $\hat{\boldsymbol{\theta}}$ (based on the true distribution) around $\theta$.

The bootstrap principle relies on the two following properties (i) as $N$ gets larger and larger the empirical distribution $\hat{F}_{\mathbf{z}}(\cdot)$ converges (almost surely) to $F_{\mathbf{z}}(\cdot)$ (see Equation (3.2.4)) and (ii) as $B$ gets larger the quantity (4.4.7) converges (in probability) to the variance of the estimator $\hat{\boldsymbol{\theta}}$ based on the empirical distribution (as stated in (2.12.63)). In other terms

$$\mathrm{Var}_{\mathrm{bs}}[\hat{\boldsymbol{\theta}}] \overset{B\to\infty}{\to} E_{\widehat{\mathbf{D}_N}}[(\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}])^2] \overset{N\to\infty}{\to} E_{\mathbf{D}_N}[(\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}])^2] \tag{4.5.11}$$

where $E_{\widehat{\mathbf{D}_N}}[(\hat{\boldsymbol{\theta}} - E[\hat{\boldsymbol{\theta}}])^2]$ stands for the plug-in estimate of the variance of $\hat{\boldsymbol{\theta}}$ based on the empirical distribution.

In practice, for a small finite $N$, bootstrap estimation inevitably returns some error. This error is a combination of a *statistical error* and *simulation error*. The statistical error component is due to the difference between the underlying distribution $F_{\mathbf{z}}(\cdot)$ and the empirical distribution $\hat{F}_{\mathbf{z}}(\cdot)$. The magnitude of this error depends on the choice of the estimator $\hat{\boldsymbol{\theta}}(\mathbf{D}_N)$ and decreases by increasing the number $N$ of observations.

The simulation error component is due to the use of empirical (Monte Carlo) properties of $\hat{\boldsymbol{\theta}}(\mathbf{D}_N)$ rather than exact properties. Simulation error decreases by increasing the number $B$ of bootstrap replications.

Unlike the jacknife method, in the bootstrap the number of replicates $B$ can be adjusted to the computer resources. In practice two "rules of thumb" are typically used:

1. Even a small number of bootstrap replications, e.g. $B = 25$, is usually informative. $B = 50$ is often enough to give a good estimate of $\mathrm{Var}\left[\hat{\boldsymbol{\theta}}\right]$.

2. Very seldom are more than $B = 200$ replications needed for estimating $\mathrm{Var}\left[\hat{\boldsymbol{\theta}}\right]$. Much bigger values of $B$ are required for bootstrap confidence intervals.

Note that the use of rough statistics $\hat{\boldsymbol{\theta}}$ (e.g. unsmooth or unstable) can make the resampling approach behave wildly. Example of nonsmooth statistics are sample quantiles and the median.

In general terms, for i.i.d. observations, the following conditions are required for the convergence of the bootstrap estimate

1. the convergence of $\hat{F}$ to $F$ (satisfied by the Glivenko-Cantelli theorem) for $N \to \infty$;

2. an estimator such that the estimate $\hat{\theta}$ is the corresponding functional of the empirical distribution.
$$\theta = t(F) \to \hat{\theta} = t(\hat{F})$$
This is satisfied for sample means, standard deviations, variances, medians and other sample quantiles.

3. a smoothness condition on the functional. This is not true for extreme order statistics such as the minimum and the maximum values.

But what happens when the dataset $D_N$ is not i.i.d. sampled from a distribution $F$? In such non conventional configurations, the most basic version of bootstrap might fail. Examples are: incomplete data (survival data, missing data), dependent data (e.g. variance of a correlated time series) and dirty data (outliers) configurations. In these cases specific adaptations of the bootstrap procedure are required. For reason of space, we will not discuss them here. However, for a more exhaustive view on bootstrap, we invite the reader to see the publication *Exploring the limits of bootstrap* edited by Le Page and Billard which is a compilation of the papers presented at a special conference of the Institute of Mathematical Statistics held in Ann Arbor, Michigan, 1990.

## 4.6 Randomization tests

Randomization tests were introduced by R.A. Fisher in 1935. The goal of a randomization test is to help discovering some regularity (e.g. a **non random** property or pattern) in a *complicated* data set. A classic example is to take a pack of poker play-cards and check whether they were well shuffled by our poker opponent. According to the hypothesis testing terminology, randomization tests make the null hypothesis of randomness and test this hypothesis against data. In order to test the randomness hypothesis, several random transformation of data are generated.

Suppose we are interested in some property which is related to the **order** of data. Let the original data set $D_N = \{x_1, \ldots, x_N\}$ and $t(D_N)$ some statistic which is a function of the order in the data $D_N$. We want to test if the value of $t(D_N)$ is due only to randomness.

- An empirical distribution is generated by scrambling (or **shuffling**) $R$ times the $N$ elements at random. For example the $j$th, $j = 1, \ldots, R$ scrambled data set could be $D_N^{(j)} = \{x_{23}, x_4, x_{343}, \ldots\}$
- For each of the $j$th scrambled sets we compute a statistic $t^{(i)}$. The resulting distribution is called the **resampling distribution**.
- Suppose that the value of $t(D_N)$ is only exceeded by $k$ of the $R$ values of the resampling distribution.
- The probability of observing $t(D_N)$ under the null hypothesis (i.e. randomness) is only $p_t = k/R$. The null hypothesis can be accepted/rejected on the basis of $p_t$.

The quantity $p_t$ plays the role of nonparametric p-value (Section 3.11.3) and it can be used, like its parametric counterpart, both to assess the evidence of the null hypothesis and to perform a decision test (e.g. refuse to play if we think cards were not sufficiently shuffled).

### A bioinformatics example

Suppose we have a DNA sequence and we think that the number of repeated sequences (e.g. AGTAGTAGT) in the sample is greater than expected by chance. Let $t = 17$ be the number of repetitions. How to test this hypothesis? Let us formulate the null hypothesis that the base order is random. We can construct an empirical distribution under the null hypothesis by taking the original sample and randomly scrambling the bases $R = 1000$ times. This creates a sample with the same base frequencies as the original sample but where the order of bases is assigned at random. Suppose that only 5 of the 1000 randomized samples has a number of repetition higher or equal than 17. The p-value (i.e. the probability of seeing $t = 17$ under the null hypothesis) which is returned by the randomization test amounts to 0.005. You can run the randomization test by using the R script file `randomiz.R`.

```
## script randomiz.R

#>YAL068C 6319249 unspliced OF sequence, from start to stop, size 363
seq<-"ATGGTCAAATTAACTTCAATCGCCGCTGGTGTCGCTGCCATCGCTGCTACTGCTTCT
GCAACCACCACTCTAGCTCAATCTGACGAAAGAGTCAACTTGGTGGAATTGGGTGTCTACGTCT
CTGATATCAGAGCTCACTTAGCCCAATACTACATGTTCCAAGCCGCCCACCCAACTGAAACCT
ACCCAGTCGAAGTTGCTGAAGCCGTTTTCAACTACGGTGACTTCACCACCATGTTGACCGGTATT
GCTCCAGACCAAGTGACCAGAATGATCACCGGTGTTCCAAGTGGTACTCCAGCAGATTAAAGCCAG
CCATCTCCAGTGCTCTAAGTCCAAGGACGGTATCTACACTATCGCAAACTAAG"

count.motif<-function(seq,mot){
  N<-nchar(seq)
  M<-nchar(mot)
  cnt<-0
  for (i in 1:N){
    cnt<-cnt+length(grep(mot,substr(seq,i,i+M-1)))
  }
  cnt
}

R<-2000
alpha<-0.05
l<-unlist(strsplit(seq,split=""))
mot<-"AAG" ## test also the motif AA
count<-count.motif(seq,mot)
print(paste ("Motif ", mot, " encountered ", count," times"))
count.perm<-array(0,c(R,1))
for (r in 1:R){
  permut.l<-sample(l)
  count.perm[r]<-count.motif(paste(permut.l,collapse=""),mot)
}

p<- sum(count.perm>=count)/R
hist(count.perm,main="Histogram of counting in randomization")
if (p<alpha){
  print(paste ("Motif ", mot, " significantly represented"))} else
  {
    print(paste("Motif ", mot, " NOT significantly represented"))
  }
```

•

### 4.6.1 Randomization and bootstrap

Both bootstrap and randomization rely on resampling. But what are their peculiarities? A randomized sample is generated by scrambling the existing data (sampling without replacement) while a bootstrap sample is generated by sampling with replacement from the original sample. Also, randomization tests are appropriate when the order or association between parts of data are assumed to convey important information. They test the null hypothesis that the order or the association is random. On the other side, bootstrap sampling aims to characterize the statistical distribution of some statistics $t(D_N)$ where the order makes no difference in the statistics (e.g. mean). Randomization would be useless in that case since $t(D_N^{(1)}) = t(D_N^{(2)})$ if $D_N^{(1)}$ and $D_N^{(2)}$ are obtained by resampling $D_N$ without replacement.

## 4.7 Permutation test

Permutation test is used to perform a nonparametric two-sample test. Consider a random sample $\{z_1, \ldots, z_M\}$ drawn from an unknown distribution $\mathbf{z} \sim F_{\mathbf{z}}(\cdot)$ and a random sample $\{y_1, \ldots, y_N\}$ from an unknown distribution $\mathbf{y} \sim F_{\mathbf{y}}(\cdot)$ . Let the null hypothesis be that the two distributions are the same regardless of the analytical forms of the distributions.

Consider a (order-independent) test statistic for the observed data and call it $t(D_N, D_M)$. The rationale of the permutation test is to locate the statistic $t(D_N, D_M)$ with respect to the distribution which could be obtained if the null hypothesis were true. In order to build the null hypothesis distribution, all the possible $R = \binom{M+N}{M}$ partitionings of the $N + M$ observations in two subsets of size $N$ and $M$ are considered. If the null hypothesis were true all the partitionings would be equally likely. Then for each $i$-th permutation ($i = 1, \ldots, R$) the permutation test computes the $t^{(i)}$ statistic. Eventually, the value $t(D_N, D_M)$ is compared with the set of values $t^{(i)}$. If the the value $t(D_N, D_M)$ falls in the $\alpha/2$ tails of the $t^{(i)}$ distribution, the null hypothesis is rejected with type I error $\alpha$.

The permutation procedure will involve substantial computation unless $M$ and $N$ are small. When the number of permutations is too large a random sample of a large number $R$ of permutations can be taken.

Note that when observations are drawn according to a normal distribution, it can be shown that the use of a permutation test gives results close to those obtained using the $t$ test.

#### Example

Let us consider $D_4 = [74, 86, 98, 102, 89]$ and $D_3 = [10, 25, 80]$. We run a permutation test ($R = \binom{8}{4} = 70$ permutations) to test the hypothesis that the two sets belong to the same distribution (R script `s_perm.R`).

Let $t(D_N) = \hat{\mu}(D_4) - \hat{\mu}(D_3) = 51.46$. Figure 4.2 shows the position of $t(D_N)$ with respect to the null sampling distribution.

```
## script s_prem.R

rm(list=ls())
library(e1071)

dispo<-function(v,d){
```

```
  dis<-NULL
  n<-length(v)
  B<-bincombinations(n)
  IB<-apply(B,1,sum)
  BB<-B[which(IB==d),]

  for (i in 1:NROW(BB)){
    dis<-rbind(dis,v[which(BB[i,]>0)])
  }
  dis
}

D1<-c(74,86,98,102,89)
D2<-c(10,25,80)

alpha<-0.1

M<-length(D1)
N<-length(D2)
D<-c(D1, D2)

t<-mean(D[1:M])-mean(D[(M+1):(M+N)])

Dp<-dispo(D,M)
tp<-numeric(nrow(Dp))

for (p in 1:nrow(Dp))
  tp[p]<-mean(Dp[p,])-mean(setdiff(D,Dp[p,]))

tp<-sort(tp)
q.inf<-sum(t<tp)/length(tp)
q.sup<-sum(t>tp)/length(tp)

hist(tp,main="")
abline(v=t,col="red")

if ((q.inf<alpha/2) | (q.sup<alpha/2)){
  title(paste("Hypothesis D1=D2 rejected: p-value=",
              round(min(c(q.inf,q.sup)),2)," alpha=", alpha))
} else {
  title(paste("Hypothesis D1=D2 not rejected: p-value=",
              round(min(c(q.inf,q.sup)),2)," alpha=", alpha))
}
```

•

## 4.8 Considerations on nonparametric tests

Nonparametric tests are a worthy alternative to parametric approaches when no assumptions about the probability distribution may be made. It is risky, however, to consider them as a panacea and a critical attitude towards them has to be preferred. In short terms, here you find some of the major advantages and disadvantages concerning the use of a nonparametric approach.

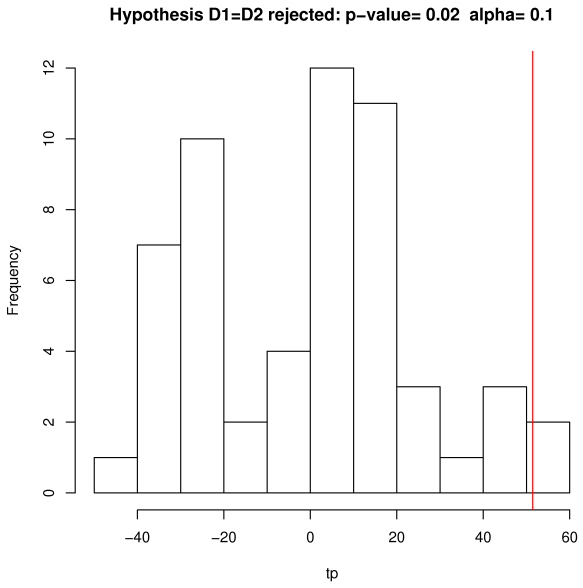



Figure 4.2: Null distribution returned by the permutation test and position (vertical red line) of the observed statistic

Advantages:

- If the sample size is very small, there may be no alternative to using a nonparametric test unless the nature of the population distribution is **known exactly**.
- Nonparametric tests make fewer assumptions about the data.
- Nonparametric tests are available to analyze data which are inherently in ranks (e.g. taste of food), classificatory or categorical.
- Nonparametric tests are typically more intuitive and easier and to implement.

Disadvantages:

- They involve high computational costs.
- The larger availability of statistical software makes possible the potential misuse of statistical measures.
- A nonparametric test is less powerful than a parametric one when the assumptions of the parametric test are met.
- Assumptions are associated with most nonparametric statistical tests, namely, that the observations are independent.

# Chapter 5

# A statistical framework of supervised learning

## 5.1 Introduction

A supervised learning problem can be described in statistical terms by the following elements:

1. A vector of $n$ random input variables $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$ , whose values are distributed according to an unknown probabilistic distribution $F_{\mathbf{x}}(\cdot)$.
2. A *target* operator which transforms the input values into *outputs* $\mathbf{y} \in \mathcal{Y}$ according to an unknown probability conditional distribution $F_{\mathbf{y}}(y|\mathbf{x} = x)$.
3. A collection $D_N$ of $N$ input/output values, called the *training set* drawn according to the joint input/output density $F_{\mathbf{x},\mathbf{y}}(x, y)$.
4. A *learning machine* or learning algorithm which, on the basis of the training set $D_N$, returns an estimation (or prediction) of the target for an input $x$. The input/output function estimated by the learning machine is called *hypothesis* or *model*.

Note that in this definition we find most of the notions presented in the previous chapters: probability distribution, conditional distribution, estimation.

#### Examples

Several practical problems can be seen as instances of a supervised learning problem:

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack, on the basis of demographic, diet and clinical measurements.
- Predict the price of a stock in 6 months from now, on the basis of company performance measures and economic data.
- Identify the risk factors for breast cancer, based on clinical, demographic and genetic variables.
- Classify the category of a text email (spam or not) on the basis of its text content.
- Characterize the mechanical property of a steel plate on the basis of its physical and chemical composition.

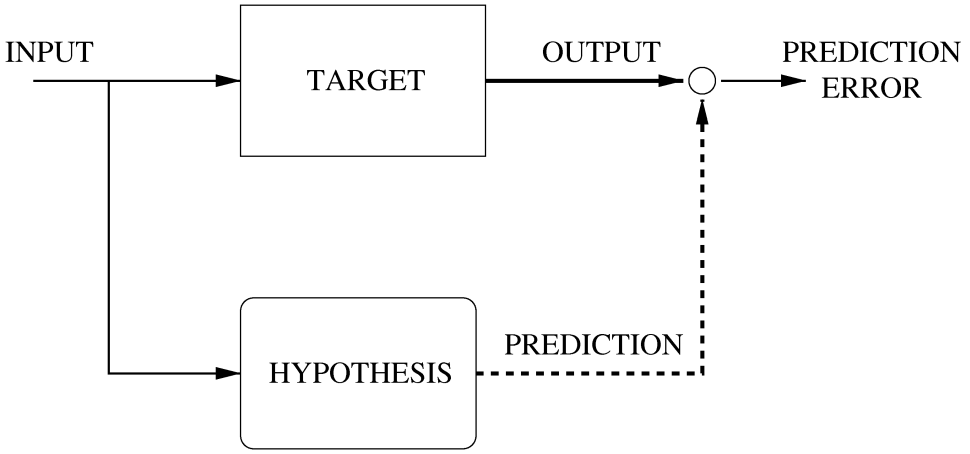



Figure 5.1: The supervised learning setting. The target operator returns an output for each input according to a fixed but unknown probabilistic law. The hypothesis predicts the value of the target output when entered with the same input.

In the case of the spam categorization problem, the input vector may be a vector of size $n$ where $n$ is the number of the most used English words and the $i$th component of $\mathbf{x}$ represents the frequency of the ith word in the email text. The output $\mathbf{y}$ is a binary class which takes two values: {SPAM,NO.SPAM}. The training set is a set of emails previously labeled by the user as SPAM and NO.SPAM. The goal of the learning machine is to create a classification function which, once a vector $x$ of word frequencies is presented, should be able to classify correctly the nature of the email.

•

A learning machine is nothing more than a particular instance of an estimator whose goal is to estimate the parameters of the joint distribution $F_{\mathbf{x},\mathbf{y}}(y,x)$ (or sometimes of the conditional distribution $F_{\mathbf{y}}(y|\mathbf{x}=x)$) on the basis of a training setb$D_N$, which is a set of random realizations of the pair of random variables $\mathbf{x}$ and $\mathbf{y}$. The goal of a *learning machine* is to return a hypothesis with low prediction error, i.e. a hypothesis which computes an accurate estimate of the output of the target when the same value is an input to the target and the predictor (Fig. 5.1). The prediction error is also usually called *generalization error*, since it measures the capacity of the hypothesis to generalize, i.e. to return a good prediction of the output for input values not contained in the training set.

We will consider only hypotheses in the form $h(\cdot,\alpha)$ where $\alpha \in \Lambda^*$ is a vector of model parameters. Therefore, henceforth, we will denote an hypothesis $h(\cdot,\alpha)$ by the corresponding vector $\alpha \in \Lambda^*$. As we will see later, examples of hypothesis are linear models $h(x,\alpha) = \alpha x$ (Section 7.1) where $\alpha$ represents the coefficients of the model, or feed-forward neural networks (Section 8.1.1) where $\alpha$ is the set of values taken by the weights of the architecture.

Let $\alpha_N$ be the hypothesis returned by the learning machine on the basis of the training set, and define $G_N$ its generalization error. The goal of the learning machine is then to seek the hypothesis $\alpha_N$ which minimizes the value $G_N$.

In these terms, the learning problem could appear as a simple problem of optimization which consists of searching the hypothesis $\alpha$ which yields the lowest generalization error. Unfortunately the reality is not that simple, since the learning machine cannot measure directly $G_N$ but only return an estimate of this quantity,

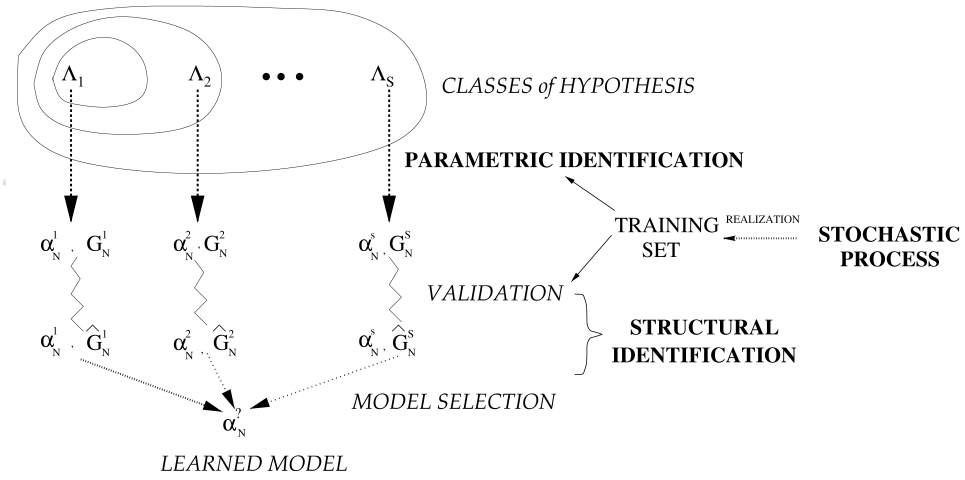



Figure 5.2: The learning problem and its decomposition in parametric and structural identification.

denoted by $\hat{G}_N$. Moreover, what makes the problem still more complex is that the same finite training set is employed both to select $\alpha_N$ and to estimate $G_N$, thus inducing a strong correlation between these two quantities.

The common supervised learning practice to minimize the quantity $G_N$ consists in first decomposing the set of hypothesis $\Lambda^*$ into a nested sequence of hypothesis classes (or *model structures*) $\Lambda_1 \subset \Lambda_2 \subset \cdots \subset \Lambda_S$, where $\Lambda^* = \cup_{s=1}^{S} \Lambda_s$, and then implementing a search procedure at two nested levels [79] (Fig. 5.2).

The inner level, also known as *parametric identification*, considers a single class of hypotheses $\Lambda_s$, with $s = 1, \ldots, S$, and uses a method or *algorithm* to select a hypothesis $h(\cdot, \alpha_N^s)$ from this class. The algorithm typically implements a procedure of multivariate optimization in the space of parameters of the class $\Lambda_s$, which can be solved by (conventional) optimization techniques. Examples of parametric identification procedures which will be presented in subsequent chapters are linear least-squares for linear models or backpropagated gradient-descent for feedforward neural networks [104].

The outer level, also called *structural identification*, ranges over different classes of hypotheses $\Lambda_s$, $s = 1, \ldots, S$, and calls for each of them the parametric routine which returns the vector $\alpha_N^s$. The outcome of the parametric identification is used to assess the class $\Lambda_s$ through a *validation* procedure which returns the estimate $\hat{G}_N^s$ on the basis of the finite training set. It is common to use nonparametric techniques to assess the quality of a predictor like the bootstrap (Section 4.4) or cross-validation [110] based on the jacknife strategy (Section 4.3).

Finally, the best hypothesis is selected in the set $\{\alpha_N^s\}$, with $s = 1, \ldots, S$, according to the assessments $\left\{\hat{G}_N^s\right\}$ produced by the validation step. This final step, which returns the model to be used for prediction, is usually referred to as the *model selection* procedure. Instances of model selection include the problem choosing the degree of a polynomial model or the problem of determining the best number of hidden nodes in a neural network [20].

The outline of the chapter is as follows. Section 5.2 introduces the supervised learning problem in statistical terms. We will show that classification (Section 5.3) and regression (Section 5.4) can be easily cast in this framework. Section 5.5 discusses the notion of generalization error and its bias/variance decomposition. Section 5.6 introduces the supervised learning procedure and its decomposition in

structural and parametric identification. Model validation and in particular cross validation, a technique for estimating the generalization error on the basis of a finite number of data, are introduced in Section 5.7.

## 5.2 Estimating dependencies

This section details the main actors of the supervised learning problem:

- A data generator of random input vectors $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$ independently and identically distributed (i.i.d) according to some unknown (but fixed) probability distribution $F_{\mathbf{x}}(x)$. The variable $\mathbf{x}$ is called the *independent variable*. It is helpful to distinguish between cases in which the experimenter has a complete control over the values of $\mathbf{x}$ and those cases in which he does not. When the nature of inputs is completely random, we consider $x$ as a realization of the random variable $\mathbf{x}$ having probability law $F_{\mathbf{x}}(\cdot)$. When the experimenter's control is complete, we can regard $F_{\mathbf{x}}(\cdot)$ as describing the relative frequencies with which different values for $x$ are set.

- A *target* operator, which transforms the input $\mathbf{x}$ into the output value $\mathbf{y} \in \mathcal{Y}$ according to some unknown (but fixed) conditional distribution

$$F_{\mathbf{y}}(y|\mathbf{x} = x) \tag{5.2.1}$$

  (this includes the simplest case where the target implements some deterministic function $\mathbf{y} = f(\mathbf{x})$). The conditional distribution (5.2.1) formalizes the stochastic dependency between inputs and output.

- A *training set* $D_N = \{\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \ldots, \langle x_N, y_N\rangle\}$ made of $N$ pairs (or training examples) $\langle x_i, y_i\rangle \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ independent and identically distributed (i.i.d) according to the joint distribution

$$F_{\mathbf{z}}(z) = F_{\mathbf{x},\mathbf{y}}(\langle x, y\rangle) \tag{5.2.2}$$

  Note that, as in Section 3.4, the observed training set $D_N \in \mathcal{Z}^N = (\mathcal{X} \times \mathcal{Y})^N$ is considered here as the realization of a random variable $\mathbf{D}_N$.

- A *learning machine* having three components:

  1. A class of *hypothesis* functions $h(\cdot, \alpha)$ with $\alpha \in \Lambda$. We consider only the case where the functions $h(\cdot, \alpha) \in \mathcal{Y}$ are single valued mappings.
  2. A *loss* function $L(\cdot, \cdot)$ associated with a particular $y$ and a particular $h(x)$, whose value $L(y, h(x))$ measures the discrepancy between the output $y$ and the prediction $h(x)$. For each hypothesis $h(\cdot, \alpha)$ we define the *functional risk* as the loss average over the $\mathcal{X}\mathcal{Y}$-domain

$$R(\alpha) = E_{\mathbf{xy}}[\mathbf{L}] = \int_{\mathcal{X},\mathcal{Y}} L(y, h(x, \alpha)) dF_{\mathbf{x},\mathbf{y}}(x, y) \tag{5.2.3}$$

     For the class $\Lambda$ of hypothesis we define

$$\alpha_0 = \arg\min_{\alpha \in \Lambda} R(\alpha) \tag{5.2.4}$$

     as the hypothesis in the class $\Lambda$ which has the lowest functional risk. Here, we assume for simplicity that there exists a minimum value of $R(\alpha)$ achievable by a function in the class $\Lambda$. We define with $R(\alpha_0)$ the *functional risk of the class of hypotheses.*

3. An *algorithm* $\mathcal{L}$ of parametric identification which takes as input the training set $D_N$ and returns as output one hypothesis function $h(\cdot, \alpha_N)$ with $\alpha_N \in \Lambda$. Here, we will consider only the case of *deterministic* and *symmetric* algorithms. This means respectively that they always give the same $h(\cdot, \alpha_N)$ for the same data set $D_N$ and that they are insensitive to the ordering of the examples in $D_N$.

   The parametric identification of the hypothesis is done according to ERM (Empirical Risk Minimization) principle where

$$\alpha_N = \alpha(D_N) = \arg\min_{\alpha \in \Lambda} R_{\text{emp}}(\alpha) \tag{5.2.5}$$

   minimizes the *empirical risk* (also know as *training error* or *apparent error*)

$$R_{\text{emp}}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, h(x_i, \alpha)) \tag{5.2.6}$$

   constructed on the basis of the data set $D_N$.

- A learning machine works well if it exhibits good generalization, i.e. if it is able to perform good predictions for unseen input values, which are not part of the training set but that are generated by the same input/output distribution (5.2.2) underlying the training set. This ability is commonly assessed by the amount of bad predictions, measured by the *generalization error*. The generalization error of a learning machine can be evaluated at two levels:

  **Hypothesis:** Let $\alpha_N$ be the hypothesis generated by the algorithm for a given training set $D_N$ according to Eq. (5.2.5). According to (5.2.3) we define with $R(\alpha_N)$ the *generalization error of the hypothesis* $\alpha_N$. This quantity is also known as *conditional error rate*.

  **Algorithm:** Let us define first the average of the loss $L$ for a given input $x$ over the ensemble of training sets of size $N$ as

$$g_N(x) = E_{\mathbf{D}_N, \mathbf{y}}[\mathbf{L} | \mathbf{x} = x] = \int_{\mathcal{Z}^N, \mathcal{Y}} L(y, h(x, \alpha_N)) dF_{\mathbf{y}}(y|x) dF_{\mathbf{z}}^N(D_N) \tag{5.2.7}$$

   where $F_{\mathbf{z}}^N(D_N)$ is the distribution of the i.i.d. dataset $D_N$. In this expression $\mathbf{L}$ is a function of the random variables $\mathbf{D}_N$ (through $h$) and $\mathbf{y}$. In the case of a quadratic loss function, this quantity corresponds to the *mean squared error* (MSE) defined in Section 3.5.4. By averaging the quantity (5.2.7) over the $\mathcal{X}$ domain we have

$$G_N = \int_{\mathcal{X}} g_N(x) dF_{\mathbf{x}}(x) = E_{\mathbf{D}_N} E_{\mathbf{x}, \mathbf{y}}[L(\mathbf{y}, h(\mathbf{x}, \boldsymbol{\alpha}_N))] \tag{5.2.8}$$

   that is the *generalization error of the algorithm* $\mathcal{L}$ (also known as *expected error rate* [42]). In the case of a quadratic loss function, the quantity

$$\text{MISE} = E_{\mathbf{D}_N} E_{\mathbf{x}, \mathbf{y}}[(\mathbf{y} - h(\mathbf{x}, \boldsymbol{\alpha}_N))^2] \tag{5.2.9}$$

   takes the name of *mean integrated squared error* (MISE).

The two criteria correspond to two different ways of assessing the learning machine: the first is a measure to assess the specific hypothesis (5.2.5) chosen by ERM, the second assesses the average performance of the algorithm over training sets with $N$ samples. Note that both quantities would require the knowledge of $F_{\mathbf{x}, \mathbf{y}}$ which is unfortunately unknown in real situations.

This formulation of a supervised learning problem is quite general, given that it includes two basic statistical problems:

1. The problem of classification (also known as pattern recognition)
2. The problem of regression estimation

These two problems and their link with supervised learning will be discussed in the following sections.

## 5.3 The problem of classification

*Classification* is one of the most common problem in statistics. It consists in exploring the association between categorical dependent variables and independent variables which can take either continuous or discrete values. The problem of *classification* is formulated as follows: consider an input/output stochastic dependence which can be described by a joint distribution $F_{\mathbf{x},\mathbf{y}}$, such that once an input vector $x$ is given, $\mathbf{y}$ takes a value among $K$ different classes. In other terms the output variable $\mathbf{y} \in \mathcal{Y} = \{c_1, \ldots, c_K\}$. In the example of spam email classification, $K = 2$ and $c_1$ =SPAM, $c_2$ =NO.SPAM. We assume that the dependence is described by a conditional discrete probability distribution Prob $\{\mathbf{y} = c_k | \mathbf{x} = x\}$ that satisfies

$$\sum_{k=1}^{K} \text{Prob}\{\mathbf{y} = c_k | x\} = 1$$

This means that observations are noisy and follow a probability distribution. In other terms, given an input $x$, $\mathbf{y}$ does not always take the same value.

**Example**

Consider a stochastic dependence where $\mathbf{x}$ represents a year's month and $\mathbf{y}$ is a categorical variable representing the weather situation in Brussels. Suppose that $\mathbf{y}$ may take only the two values {RAIN,NO.RAIN}. The setting is stochastic since you might have rainy August and some rare sunny December days. Suppose that the conditional probability distribution of $\mathbf{y}$ is represented in Figure 5.3. This figure plots Prob $\{\mathbf{y} = \text{RAIN} | \mathbf{x} = \text{month}\}$ and Prob $\{\mathbf{y} = \text{NO.RAIN} | x = \text{month}\}$ for each month. Note that for each month the probability constraint is respected:

$$\text{Prob}\{\mathbf{y} = \text{RAIN} | \mathbf{x} = \text{month}\} + \text{Prob}\{\mathbf{y} = \text{NO.RAIN} | \mathbf{x} = \text{month}\} = 1$$

•

A classifier is a particular instance of estimator which for a given $x$ is expected to return an estimate $\hat{y} = \hat{c} = h(x, \alpha)$ which takes a value in $\{c_1, \ldots, c_K\}$. Once a cost function is defined, the problem of classification can be expressed in terms of the formalism introduced in the previous section. An example of cost function is the 0/1 loss

$$L(c, \hat{c}) = \begin{cases} 0 & \text{if } c = \hat{c} \\ 1 & \text{if } c \neq \hat{c} \end{cases} \tag{5.3.10}$$

However, we can imagine situations where some misclassifications are worse than others. In this case, it is better to introduce a *loss matrix* $L_{(K \times K)}$ where the element $L_{(jk)} = L_{(c_j, c_k)}$ denotes the cost of the misclassification when the predicted class is $\hat{c}(x) = c_j$ and the correct class is $c_k$. This matrix must be null on the diagonal and non negative elsewhere. Note that in the case of the 0-1 loss function (Equation 5.3.10) all the elements outside the diagonal are equal to one.

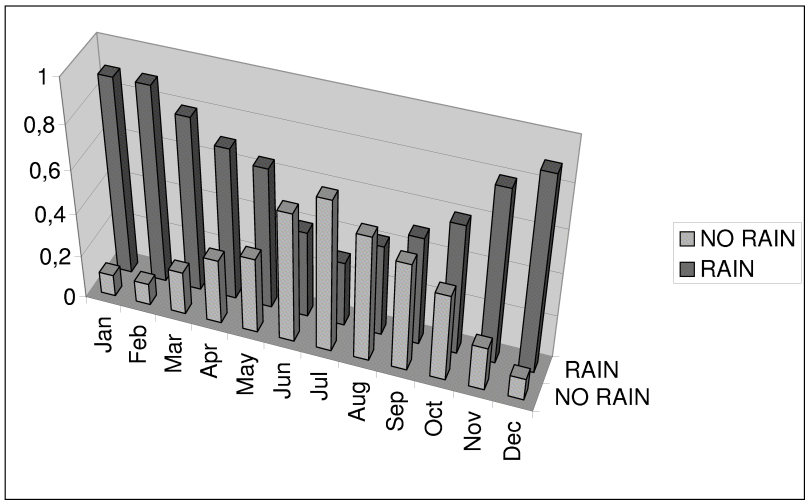



Figure 5.3: Conditional distribution Prob $\{\mathbf{y}|x\}$ where $x$ is the current month and $\mathbf{y}$ is the random weather state. For example the column corresponding to $x$ =Dec and $y$ =RAIN returns the conditional probability of RAIN in December.

The goal of the classification procedure for a given $x$ is to find the predictor $\hat{c}(x) = h(x, \alpha)$ that minimizes the quantity

$$\sum_{k=1}^{K} L_{(\hat{c}(x),c_k)} \text{Prob}\{\mathbf{y} = c_k|x\} \tag{5.3.11}$$

which is an average of the $\hat{c}(x)$ row of the loss matrix weighted by the conditional probabilities of observing $\mathbf{y} = c_k$. Note that the average of the above quantity over the $\mathcal{X}$ domain

$$\int_{\mathcal{X}} \sum_{k=1}^{K} L_{(\hat{c}(x),c_k)} \text{Prob}\{\mathbf{y} = c_k|x\}\, dF_{\mathbf{x}} = \int_{\mathcal{X},\mathcal{Y}} L(y, h(x, \alpha)) dF_{\mathbf{x},\mathbf{y}} = R(\alpha) \tag{5.3.12}$$

corresponds to the functional risk (5.2.3).

The problem of classification can then be seen as a particular instance of the more general supervised learning problem described in Section 5.2.

It can be shown that the optimal classifier $h(, \alpha_0)$ where $\alpha_0$ is defined as in (5.2.4) is the one that returns for all $x$

$$c^*(x) = h(x, \alpha_0) = \arg \min_{c_j \in \{c_1,\ldots,c_K\}} \sum_{k=1}^{K} L_{(j,k)} \text{Prob}\{\mathbf{y} = c_k|x\} \tag{5.3.13}$$

The optimal classifier is also known as the *Bayes classifier*. In the case of a 0-1 loss function the optimal classifier returns

$$c^*(x) = \arg \min_{c_j \in \{c_1,\ldots,c_K\}} \sum_{k=1:K,k\neq j} \text{Prob}\{\mathbf{y} = c_k|x\} \tag{5.3.14}$$

$$= \arg \min_{c_j \in \{c_1,\ldots,c_K\}} (1 - \text{Prob}\{\mathbf{y} = c_j|x\}) \tag{5.3.15}$$

$$= \arg \min_{c_j \in \{c_1,\ldots,c_K\}} \text{Prob}\{\mathbf{y} \neq c_j|x\} = \arg \max_{c_j \in \{c_1,\ldots,c_K\}} \text{Prob}\{\mathbf{y} = c_j|x\} \tag{5.3.16}$$

The Bayes decision rule selects the $j$, $j = 1, \ldots, K$, that *maximizes* the posterior probability $\text{Prob}\{\mathbf{y} = c_j | x\}$.

**Example**

Consider a classification task where $\mathcal{X} = \{1, 2, 3, 4, 5\}$, $\mathcal{Y} = \{c_1, c_2, c_3\}$ and the loss matrix and the conditional probability values are given in the following figures.

| REAL / PRED | C1 | C2 | C3 |
|---|---|---|---|
| C1 | 0 | 1 | 5 |
| C2 | 20 | 0 | 10 |
| C3 | 2 | 1 | 0 |

LOSS MATRIX

| x | C1 | C2 | C3 |
|---|---|---|---|
| 1 | 0.1 | 0.6 | 0.3 |
| 2 | 0.2 | 0.8 | 0.0 |
| 3 | 0.9 | 0.04 | 0.06 |
| 4 | 0.5 | 0.25 | 0.25 |
| 5 | 0.3 | 0.1 | 0.6 |

CONDITIONAL PROBABILITY

Let us focus on the optimal classification for $x = 2$. According to (5.3.13) the Bayes classification rule for $x = 2$ returns

$$
\begin{aligned}
c^*(2) = \arg \min_{k=1,2,3} \{ & L_{11}\text{Prob}\{\mathbf{y} = c_1 | \mathbf{x} = 2\} + L_{12}\text{Prob}\{\mathbf{y} = c_2 | \mathbf{x} = 2\} + L_{13}\text{Prob}\{\mathbf{y} = c_3 | \mathbf{x} = 2\}, \\
& L_{21}\text{Prob}\{\mathbf{y} = c_1 | \mathbf{x} = 2\} + L_{22}\text{Prob}\{\mathbf{y} = c_2 | \mathbf{x} = 2\} + L_{23}\text{Prob}\{\mathbf{y} = c_3 | \mathbf{x} = 2\}, \\
& L_{31}\text{Prob}\{\mathbf{y} = c_1 | \mathbf{x} = 2\} + L_{32}\text{Prob}\{\mathbf{y} = c_2 | \mathbf{x} = 2\} + L_{33}\text{Prob}\{\mathbf{y} = c_3 | \mathbf{x} = 2\}\} \\
= \arg \min_{k=1,2,3} \{ & 0 * 0.2 + 1 * 0.8 + 5 * 0.0, 20 * 0.2 + 0 * 0.8 + 10 * 0.0, \\
& 2 * 0.2 + 1 * 0.8 + 0.0 * 0\} = \arg \min_{k=1,2,3} \{1, 4, 1.2\} = 1
\end{aligned}
$$

What would have been the Bayes classification in the 0-1 case?

•

### 5.3.1 Inverse conditional distribution

An important quantity, often used in classification algorithms, is the *inverse conditional distribution*. According to the Bayes theorem we have that

$$\text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\} = \frac{\text{Prob}\{\mathbf{x} = x | \mathbf{y} = c_k\}\,\text{Prob}\{\mathbf{y} = c_k\}}{\sum_{k=1}^{K} \text{Prob}\{\mathbf{x} = x | \mathbf{y} = c_k\}\,\text{Prob}\{\mathbf{y} = c_k\}}$$

and that

$$\text{Prob}\{\mathbf{x} = x | \mathbf{y} = c_k\} = \frac{\text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\}\,\text{Prob}\{\mathbf{x} = x\}}{\sum_{x} \text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\}\,\text{Prob}\{\mathbf{x} = x\}}. \tag{5.3.17}$$

The above relation means that by knowing the conditional distribution $\text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\}$ and the *a priori* distribution $\text{Prob}\{\mathbf{x} = x\}$, we can derive the conditional distribution $\text{Prob}\{\mathbf{x} = x | \mathbf{y} = c_k\}$. This distribution characterizes the values of the inputs $x$ for a given class $c_k$.

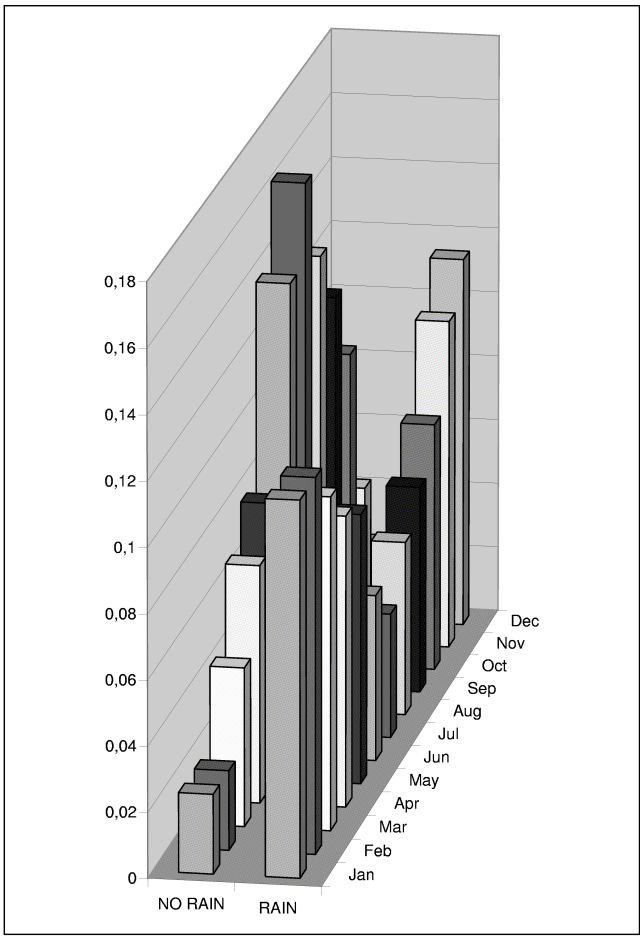



Figure 5.4: Inverse conditional distribution of the distribution in Figure 5.3

**Example**

Suppose we want to know during which months it is most probable to have rain. This boils down to have the distribution of $x$ for $\mathbf{y} = RAIN$. Figure 5.4 plots the inverse conditional distributions $\text{Prob}\{\mathbf{x} = \text{month}|\mathbf{y} = \text{RAIN}\}$ and $\text{Prob}\{\mathbf{x} = \text{month}|\mathbf{y} = \text{NO.RAIN}\}$ according to (5.3.17) when we assume that the a priori distribution is uniform (i.e. $\text{Prob}\{\mathbf{x} = x\} = 1/12$ for all $x$).

Note that

$$\sum_{\text{month}} \text{Prob}\{\mathbf{x} = \text{month}|\mathbf{y} = \text{NO.RAIN}\} = \sum_{\text{month}} \text{Prob}\{\mathbf{x} = \text{month}|\mathbf{y} = \text{RAIN}\} = 1$$

•

## 5.4 The problem of regression estimation

Consider the stochastic relationship between two continuous random variables $\mathbf{x}$ and $\mathbf{y}$ following a joint distribution

$$F_{\mathbf{x},\mathbf{y}}(x, y) \tag{5.4.18}$$

This means that to each vector $x$ sampled according to the $F_x(x)$ there corresponds a scalar $y$ sampled from $F_{\mathbf{y}}(\mathbf{y} = y|\mathbf{x} = x)$. Assume that a set of $N$ input/output samples is available. The estimation of the stochastic dependence on the basis of the empirical dataset requires the estimation of the conditional distribution $F_{\mathbf{y}}(y|x)$. This is known to be a difficult problem but for prediction purposes, most of the time, it is sufficient to estimate the conditional expectation

$$f(x) = E_{\mathbf{y}}[\mathbf{y}|x] = \int_{\mathcal{Y}} y dF_{\mathbf{y}}(y|x) \tag{5.4.19}$$

also known as the *regression function*. Note that it can be shown that the minimum of the functional risk for the quadratic loss

$$R(\alpha) = \int L(y, h(x, \alpha)) dF_{\mathbf{x},\mathbf{y}}(x, y) = \int (y - h(x, \alpha))^2 dF_{\mathbf{x},\mathbf{y}}(x, y) \tag{5.4.20}$$

is attained by the regression function $h(\cdot, \alpha_0) = f(\cdot)$ if the function $f$ belongs to the set $h(x, \alpha)$, $\alpha \in \Lambda$.

The problem of estimating the regression function (5.4.19) is then a particular instance of the supervised learning problem described in Section 5.2, where the learning machine is assessed by a quadratic cost function. Examples of learning algorithms for regression will be discussed in Section 7.1 and Section 8.1.

### 5.4.1 An illustrative example

The notation introduced in Section 5.2 is rigorous but it may appear hostile to the practitioner. In order to make the statistical concepts more affordable we present a simple example to illustrate these concepts.

We consider a supervised learning regression problem where :

- The input is a scalar random variable $\mathbf{x} \in \mathbb{R}$ with a uniform probability distribution over the interval $[-2, 2]$.
- The target is distributed according to a conditional Gaussian distribution $F_{\mathbf{y}}(\mathbf{y}|\mathbf{x} = x) = \mathcal{N}(x^3, 1)$ where the expected value is a function of $x$
- The training set $D_N = \{\langle x_i, y_i \rangle\}, i = 1, \ldots, N$ consists of $N = 100$ pairs (Figure 5.5). Note that this training set can be easily generated with the following R commands

```
## script regr.R

N<-100
X<-runif(N,-2,2)
Y=X^3+rnorm(N)
plot(X,Y)
```

- The learning machine is characterized by the following three components:

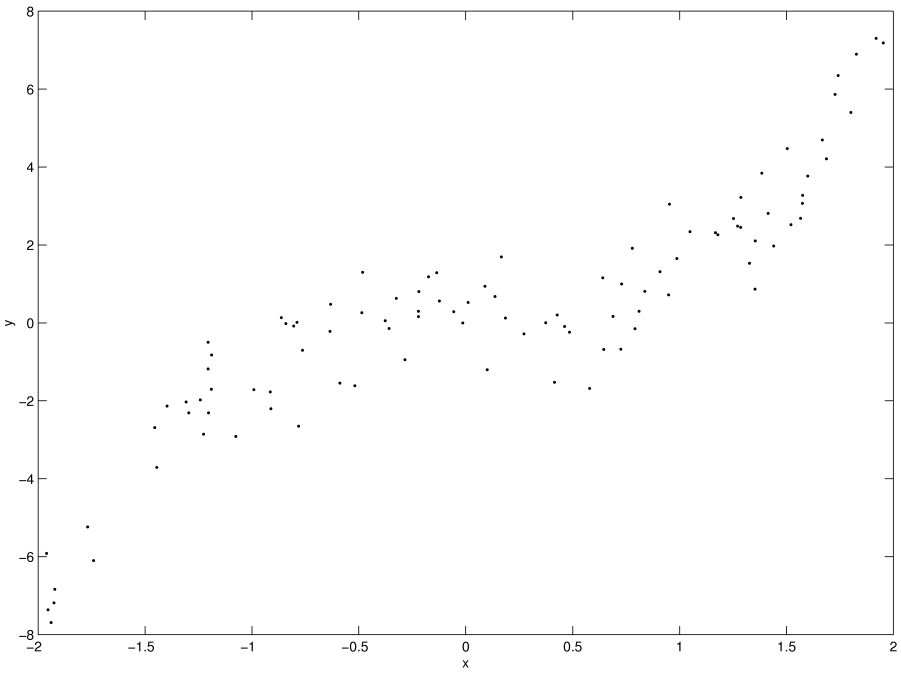



Figure 5.5: Training set (dots) obtained by sampling uniformly in the interval $[-2, 2]$ an input/output distribution with regression function $f(x) = x^3$ and unit variance.

1. A class of hypothesis functions $h(x, \alpha) = \alpha x$ consisting of all the linear models passing through the origin. The class $\Lambda$ is then the set of real numbers.
2. A quadratic loss $L(y, h(x)) = (y - h(x))^2$.
3. An algorithm of parametric identification based on the least-squares technique, which will be detailed later in Section 7.1.2. The empirical risk is the quantity

$$R_{\text{emp}}(\alpha) = \frac{1}{100} \sum_{i=1}^{100} (y_i - \alpha x_i)^2$$

The empirical risk is a function of $\alpha$ and the training set. For the given training set $D_N$, the empirical risk as a function of $\alpha$ is plotted in Fig. 5.6.

For the dataset in Fig. 5.5, the least-squares computation returns

$$\alpha_N = \arg\min_{\alpha \in \Lambda} R_{\text{emp}}(\alpha) = \arg\min_{\alpha \in \Lambda} \frac{1}{100} \sum_{i=1}^{100} (y_i - \alpha x_i)^2 = 2.3272 \tag{5.4.21}$$

The selected hypothesis is plotted in the input/output domain in Fig. 5.7.

If the knowledge on the joint distribution was available, it would be possible to compute also the risk functional (5.2.3) as

$$R(\alpha) = \frac{1}{4} \int_{-2}^{2} (x^3 - \alpha x)^2 dx + 1 \tag{5.4.22}$$

where the derivation of the equality is sketched in Appendix B.6. For the given joint distribution, the quantity $R(\alpha)$ is plotted as a function of $\alpha$ in Fig. 5.8.

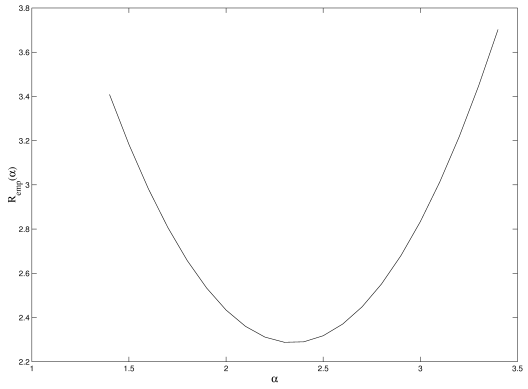



Figure 5.6: The empirical risk for the training set $D_N$ vs. the parameter value (x-axis). The minimum of the empirical risk is attained in $\alpha = 2.3272$.

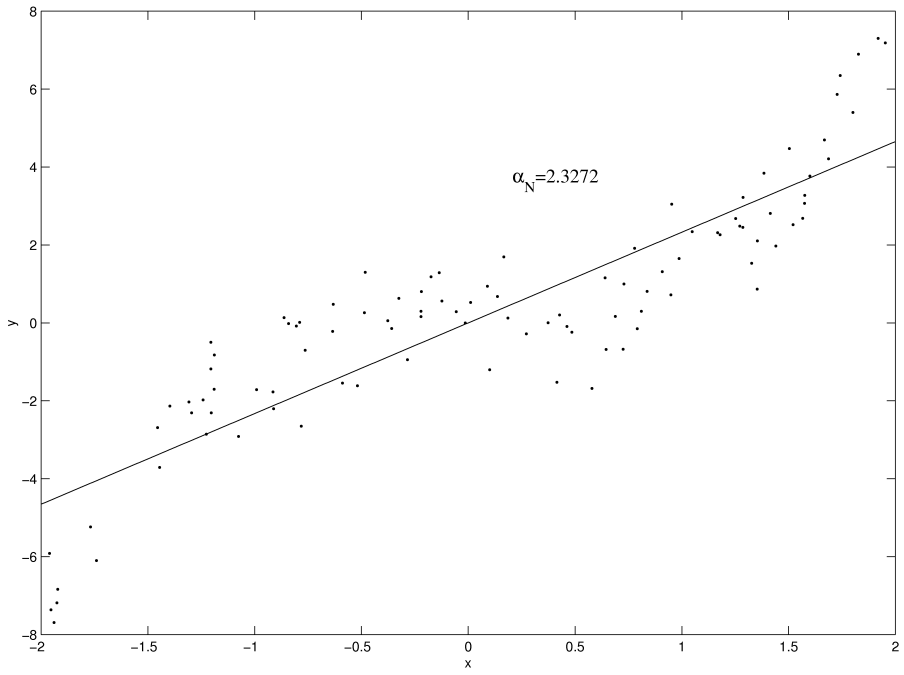



Figure 5.7: Training set (dotted points) and the linear hypothesis function $h(\cdot, \alpha_N)$ (straight line). The quantity $\alpha_N$, which represents the slope of the straight line, is the value of the parameter $\alpha$ which minimizes the empirical risk.

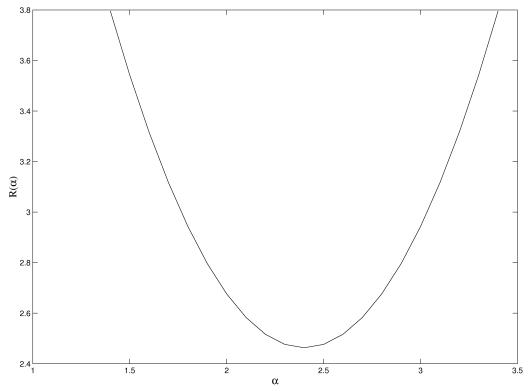



Figure 5.8: The functional risk (5.4.22) vs. the value of parameter $\alpha$ (x-axis). The minimum of the functional risk is attained in $\alpha = 2.4$.

The computation of the quantity (5.2.7) requires however an average over all the possible realizations of the random variable $\boldsymbol{\alpha}_N$ for datasets of $N = 100$ points. Figure 5.9 shows 6 different realizations of the training set for the same joint distribution and the corresponding 6 values of $\alpha_N$.

It is important to remark that both the quantities (5.2.3) and (5.2.7) are easy to compute once an a priori knowledge over the joint distribution underlying the data is available. Unfortunately, in a large number of real cases this knowledge is not accessible and the goal of learning theory is to study the problem of estimating these quantities from a finite set of data.

## 5.5 Generalization error

This section studies in detail the algorithm-based criterion $G_N$ (Equation (5.2.9)) as a measure of the generalization error of the learning machine. Note that a different approach is proposed by Vapnik [114, 115, 116] in his formalization of the statistical learning theory where the accuracy of the learning machine is evaluated on the basis of the quantity $R(\alpha_N)$.

In particular we will study how the generalization error can be decomposed in the regression formulation and in the classification formulation.

### 5.5.1 The decomposition of the generalization error in regression

Let us focus now on of the $g_N$ measure (Equation (5.2.7)) of the generalization error in the case of regression. In the case of a quadratic loss

$$L(y(x), h(x, \alpha)) = (y(x) - h(x, \alpha))^2 \tag{5.5.23}$$

the quantity $g_N$ is often referred to as the *mean squared error* (MSE) and its marginal (5.2.9) as the *mean integrated squared error* (MISE). Suppose that the conditional target density can be put in the *regression plus noise* form[1]

$$p_{\mathbf{y}}(y - f(x)|x) = p_{\mathbf{y}}(y - E_{\mathbf{y}}[\mathbf{y}|x]|x) = p_{\mathbf{w}}(w) \tag{5.5.24}$$

[1]Notice that the assumption of an additive noise independent of $x$ is common in statistical literature and is not overly restrictive. In fact, many other conceivable signal/noise models can be transformed into this form.

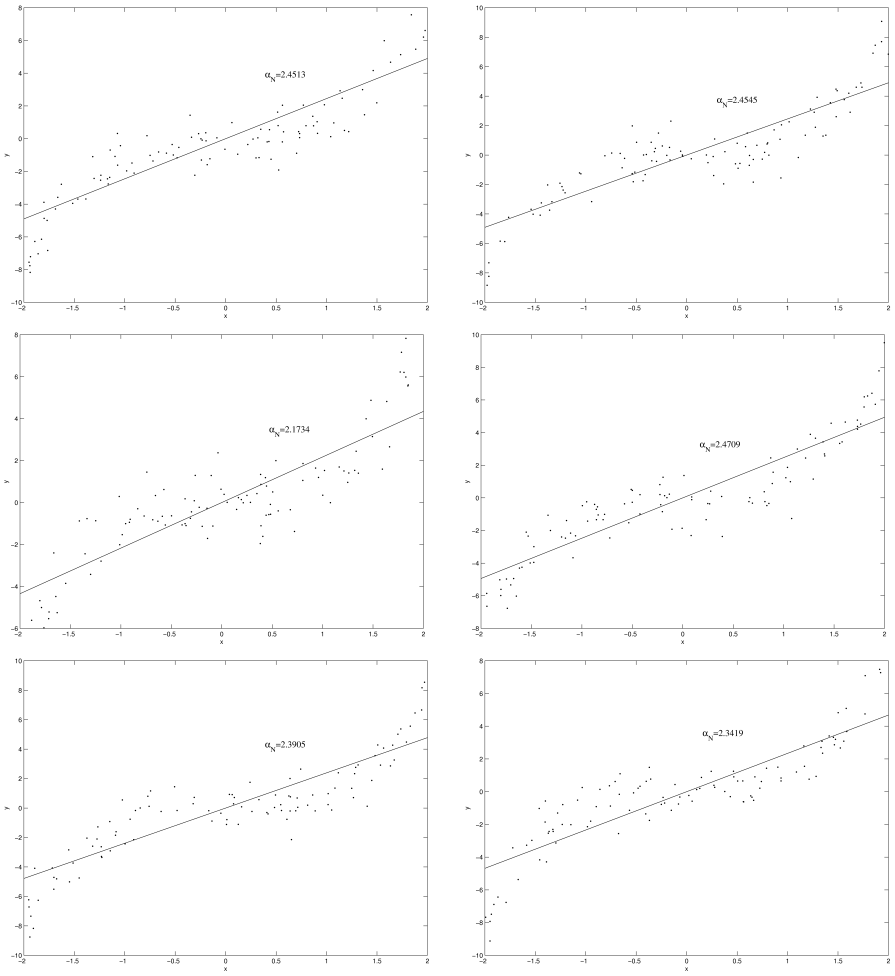



Figure 5.9: Six different realizations of a training set with $N = 100$ points (dots) and the relative hypotheses (solid straight lines) chosen according to the ERM principle (5.4.21).

where $\mathbf{w}$ is a noisy random variable with zero mean and variance $\sigma^2_{\mathbf{w}}$.

This supervised learning problem can be seen as a particular instance of the estimation problem discussed in Chapter 3, where, for a given $x$, the unknown parameter $\theta$ to be estimated is the quantity $f(x)$ and the estimator based on the training set is $\hat{\boldsymbol{\theta}} = h(x, \boldsymbol{\alpha}_N)$. The MSE quantity, defined in (3.5.14) coincides, apart from an additional term, with the term (5.2.7) since

$$
\begin{aligned}
g_N(x) &= E_{\mathbf{D}_N,\mathbf{y}}[\mathbf{L}|x] = && (5.5.25)\\
&= E_{\mathbf{D}_N,\mathbf{y}}\left[(\mathbf{y} - h(x,\boldsymbol{\alpha}_N))^2\right] = && (5.5.26)\\
&= E_{\mathbf{y}}\left[(\mathbf{y} - E_{\mathbf{y}}[\mathbf{y}|x])^2\right] + E_{\mathbf{D}_N}\left[(h(x,\boldsymbol{\alpha}_N) - E_{\mathbf{y}}[\mathbf{y}|x])^2\right] = && (5.5.27)\\
&= \sigma^2_{\mathbf{w}} + E_{\mathbf{D}_N}[(f(x) - h(x,\boldsymbol{\alpha}_N)^2] = \sigma^2_{\mathbf{w}} + E_{\mathbf{D}_N}[(\theta - \hat{\boldsymbol{\theta}}^2)^2] = && (5.5.28)\\
&= \sigma^2_{\mathbf{w}} + \text{MSE} && (5.5.29)
\end{aligned}
$$

We can then apply *bias/variance* decomposition (5.5.31) to the regression problem.

$$
\begin{aligned}
g_N(x) &= E_{\mathbf{D}_N,\mathbf{y}}[\mathbf{L}(x,y)] = \\
&= \sigma^2_{\mathbf{w}} + E_{\mathbf{D}_N}\left[(h(x,\boldsymbol{\alpha}_N) - E_{\mathbf{y}}[\mathbf{y}|x])^2\right] = \\
&= \sigma^2_{\mathbf{w}} + && \textit{noise}\\
&+ (E_{\mathbf{D}_N}[h(x,\boldsymbol{\alpha}_N)] - E_{\mathbf{y}}[\mathbf{y}|x])^2 + && \textit{squared bias}\\
&+ E_{\mathbf{D}_N}\left[(h(x,\boldsymbol{\alpha}_N) - E_{\mathbf{D}_N}[h(x,\boldsymbol{\alpha}_N)])^2\right] = && \textit{variance}\\
&= \sigma^2_{\mathbf{w}} + B^2(x) + V(x)
\end{aligned}
\qquad (5.5.30)
$$

In a regression task, the bias $B(x)$ measures the difference in $x$ between the average of the outputs of the hypothesis functions over the set of possible $D_N$ and the regression function value $E_{\mathbf{y}}[\mathbf{y}|x]$. The variance $V(x)$ reflects the variability of the guessed $h(x, \boldsymbol{\alpha}_N)$ as one varies over training sets of fixed dimension $N$. This quantity measures how sensitive the algorithm is to changes in the data set, regardless to the target. So by Eq. (5.2.9) by averaging over $\mathcal{X}$ we obtain

$$
\text{MISE} = G_N = \sigma^2_{\mathbf{w}} + \int_{\mathcal{X}} B^2(x) dF_{\mathbf{x}} + \int_{\mathcal{X}} V(x) dF_{\mathbf{x}} = \text{squared bias+variance} \qquad (5.5.31)
$$

where the intrinsic noise term reflects the target alone, the bias reflects the target's relation with the learning algorithm and the variance term reflects the learning algorithm alone.

As the goal of a learning machine is to minimize the entire quantity $G_N$, this decomposition could appear as a useless theoretical exercise. In reality, the designer of a learning machine has not access to the term $G_N$ but can only estimate it on the basis of the training set. Hence, the bias/variance decomposition is relevant in practical learning since it provides a useful hint about the features to control in order to make the error $G_N$ small. In particular, the bias term measures the lack of representational power of the class of hypotheses. This means that to reduce the bias term of the generalization error we should consider large classes of hypotheses, or in other words hypotheses which can approximate a large number of input/output mappings. On the other side, the variance term warns us against an excessive complexity of the approximator. This means that a class of too powerful hypotheses runs the risk of being excessively sensitive to the noise affecting the training set; therefore, our class could contain the target but it could be practically impossible to find it out on the basis of the available dataset.

In other terms, it is commonly said that an hypothesis with large bias but low variance *underfits* the data while an hypothesis with low bias but large variance *overfits* the data. In both cases, the hypothesis gives a poor representation of the

target and a reasonable trade-off needs to be found. The task of the model designer is to search for the optimal trade-off between the variance and the bias terms, on the basis of the available training set. Section 5.6 will illustrate how this search proceeds.

### 5.5.2 The decomposition of the generalization error in classification

Let us consider a classification task with $K$ output classes and a loss function $L$. For a given input $x$, we denote by $\hat{\mathbf{y}}$ the class predicted by the classifier $h(x, \boldsymbol{\alpha}_N)$ trained with a dataset $D_N$. We derive the analytical expression of $g_N(x)$, usually referred to as the *mean misclassification error* (MME).

$$\text{MME}(x) = E_{\mathbf{y},\mathbf{D}_N}[L(\mathbf{y}, h(x, \boldsymbol{\alpha}_N))|x] = E_{\mathbf{y},\mathbf{D}_N}[L(\mathbf{y}, \hat{\mathbf{y}})] = \tag{5.5.32}$$

$$= E_{\mathbf{y},\mathbf{D}_N}[\sum_{k,j=1}^{K} L_{(j,k)} 1(\hat{\mathbf{y}} = c_j|x) 1(\mathbf{y} = c_k|x)] = \tag{5.5.33}$$

$$= \sum_{k,j=1}^{K} L_{(j,k)} E_{\mathbf{D}_N}[1(\hat{\mathbf{y}} = c_j|x)] E_{\mathbf{y}}[1(\mathbf{y} = c_k|x)]] = \tag{5.5.34}$$

$$= \sum_{k,j=1}^{K} L_{(j,k)} \text{Prob}\{\hat{\mathbf{y}} = c_j|x\} \text{Prob}\{\mathbf{y} = c_k|x\} \tag{5.5.35}$$

where $1(\cdot)$ is the indicator function which returns zero when the argument is false and one elsewhere. Note that the distribution of $\hat{\mathbf{y}}$ depend on the training set $\mathbf{D}_N$ while the distribution of $\mathbf{y}$ is the distribution of a test set (independent of $\mathbf{D}_N$). For zero-one loss function, since $\mathbf{y}$ and $\hat{\mathbf{y}}$ are independent, the MME expression simplifies to

$$\begin{aligned}
\text{MME}(x) &= \sum_{k,j=1}^{K} 1(c_j \neq c_k) \text{Prob}\{\hat{\mathbf{y}} = c_j|x\} \text{Prob}\{\mathbf{y} = c_k|x\} = \\
&= 1 - \sum_{k,j=1}^{K} 1(c_j = c_k) \text{Prob}\{\hat{\mathbf{y}} = c_j|x\} \text{Prob}\{\mathbf{y} = c_k|x\} = \\
&= 1 - \sum_{k}^{K} \text{Prob}\{\hat{\mathbf{y}} = c_k|x\} \text{Prob}\{\mathbf{y} = c_k|x\} = \\
&= \text{Prob}\{\mathbf{y} \neq \hat{\mathbf{y}}\}
\end{aligned} \tag{5.5.36}$$

A decomposition of a related quantity was proposed in [122]. Let us consider the squared sum:

$$\frac{1}{2} \sum_{j=1}^{K} \left(\text{Prob}\{\mathbf{y} = c_j\} - \text{Prob}\{\hat{\mathbf{y}} = c_j\}\right)^2 =$$

$$\frac{1}{2} \left( \sum_{j=1}^{K} \text{Prob}\{\mathbf{y} = c_j\}^2 \right) + \frac{1}{2} \left( \sum_{j=1}^{K} \text{Prob}\{\hat{\mathbf{y}} = c_j\}^2 \right) - \sum_{j=1}^{K} \text{Prob}\{\mathbf{y} = c_j\} \text{Prob}\{\hat{\mathbf{y}} = c_j\}$$

By adding one to both members and by using (5.5.32) we obtain a decomposition

analogous to the one in (5.5.30)

$$
\begin{aligned}
g_N(x) &= \mathrm{MME}(x) = \\
&= \frac{1}{2}\left(1 - \left(\sum_{j=1}^{K} \mathrm{Prob}\left\{\mathbf{y} = c_j\right\}^2\right)\right) + \qquad \textit{“noise”} \\
&+ \frac{1}{2}\sum_{j=1}^{K}\left(\mathrm{Prob}\left\{\mathbf{y} = c_j\right\} - \mathrm{Prob}\left\{\hat{\mathbf{y}} = c_j\right\}\right)^2 + \qquad \textit{“squared bias”} \\
&+ \frac{1}{2}\left(1 - \left(\sum_{j=1}^{K} \mathrm{Prob}\left\{\hat{\mathbf{y}} = c_j\right\}^2\right)\right) \qquad \textit{“variance”}
\end{aligned}
\tag{5.5.37}
$$

The noise terms measures the degree of uncertainty of $\mathbf{y}$ and consequently the degree of stochasticity of the dependance. It equals zero only and only if there exists a class $c$ such that $\mathrm{Prob}\left\{\mathbf{y} = c|x\right\} = 1$ and zero elsewhere. Note that this quantity does not depend on the learning algorithm nor on the training set.

The variance term measures how variant the classifier prediction $\hat{\mathbf{y}} = h(x, \boldsymbol{\alpha}_N)$ is. This quantity is zero if the predicted class is always the same regardless of the training set.

The squared bias term measures the squared difference between the $\mathbf{y}$ and the $\hat{\mathbf{y}}$ probability distributions on the domain $\mathcal{Y}$.

## 5.6 The supervised learning procedure

The goal of supervised learning is to return the hypothesis with the lowest generalization error. Since we assume that data samples are generated in a random way, there is no hypothesis which gives a null generalization error. Therefore, the generalization error $G_N$ of the hypothesis returned by a learning machine has to be compared to the minimal generalization error that can be attained by the best single-valued mapping. Let us define by $\Lambda^*$ the set of all possible single valued mappings $f : \mathcal{X} \to \mathcal{Y}$ and consider the hypothesis

$$\alpha^* = \arg\min_{\alpha \in \Lambda^*} R(\alpha) \tag{5.6.38}$$

Thus, $R(\alpha^*)$ represents the absolute minimum rate of error obtainable by a single valued approximator of the unknown target. For maintaining a simple notation, we put $G^* = R(\alpha^*)$. For instance, in our illustrative example in Section 5.4.1, $\alpha^*$ denotes the parameters of the cubic function and $G^*$ amounts to the unit variance of the Gaussian noise.

In theoretical terms, a relevant issue is to demonstrate that the generalization error $G_N$ of the outcome of the learning machine converges to the minimum $G^*$ for $N$ going to infinity.

Unfortunately, in real learning settings, two problems must be dealt with. The first is that the error $G_N$ cannot be computed directly but has to be estimated from data. The second is that a single class $\Lambda$ could not be large enough to contain the hypothesis $\alpha^*$.

A common practice to handle these problems is to decompose the learning procedure in the following sequence of steps:

1. A nested sequence of classes of hypotheses

$$\Lambda_1 \subseteq \cdots \subseteq \Lambda_s \subseteq \ldots \Lambda_S \tag{5.6.39}$$

is defined so that $\Lambda^* = \cup_{s=1}^{S} \Lambda_s$. This guarantees that the set of hypotheses taken into consideration will necessarily contain the best hypothesis $\alpha^*$.

A priori informations as well as considerations related to the bias/variance dilemma can help in the design of this sequence.

2. For each class in the sequence, a hypothesis $\alpha_N^s$, $s = 1, \ldots, S$, is selected by minimizing the empirical risk (5.2.6). This step is defined as the *parametric identification* step of the learning procedure.

3. For each class in the sequence, a validation procedure returns $\hat{G}_N^s$ which estimates the generalization error $G_N^s$ of the hypothesis $\alpha_N^s$. This step is called the *validation* step of the learning procedure.

4. The hypothesis $\alpha_N^{\bar{s}}$ with
$$\bar{s} = \arg\min_s \hat{G}_N^s \tag{5.6.40}$$
is returned as the final outcome. This final step is called the *model selection* step.

However, the learning procedure, as defined above, has still a degree of freedom, represented by the validation method. This element is determinant for the performance of the supervised learning procedure since the selection of the final outcome requires the estimates returned by the validation step (see Eq. (5.6.40)).

## 5.7 Validation techniques

In this section we introduce some validation methods which address the estimation of the quantity $G_N$ from a finite set of data.

The empirical risk or apparent error $R_{\text{emp}}(\alpha_N)$ introduced in (5.2.5) is seen as the most obvious estimate $\hat{G}_N$ of $G_N$. However, it is generally known that the empirical risk is a biased estimate of $G_N$ and that $R_{\text{emp}}(\alpha_N)$ tends to be smaller than $G_N$, because the same data have been used both to construct and to evaluate $h(\cdot, \alpha_N)$. A demonstration of the biasedness of the empirical risk for a quadratic loss function in a regression setting is available in Appendix B.7. In Section 7.1.14 we will analytically derive the biasedness of the empirical risk in case of linear regression models.

The study of error estimates other than the apparent error is of significant importance if we wish to obtain results applicable to practical learning scenarios. There are two main ways to obtain better, i.e. unbiased, estimates of $G_N$: the first requires some knowledge on the distribution underlying the data set, the second makes no assumptions on the data. As we will see later, example of the first approach is the FPE criterion (presented in Section 7.1.16) while examples of the second approach are the resampling procedures.

### 5.7.1 The resampling methods

*Cross-validation* [110] is a well-known method in sampling statistics to circumvent the limits of the apparent error estimate. The basic idea of cross-validation is that one builds a model from one part of the data and then uses that model to predict the rest of the data. The dataset $D_N$ is split $l$ times in a training and a test subset, the first containing $N_{tr}$ samples, the second containing $N_{ts} = N - N_{tr}$ samples. Each time, $N_{tr}$ samples are used by the parametric identification algorithm $\mathcal{L}$ to select a hypothesis $\alpha_{N_{tr}}^i$, $i = 1, \ldots, l$, from $\Lambda$ and the remaining $N_{ts}$ samples are used to estimate the error of $h(\cdot, \alpha_{N_{tr}}^i)$ (Fig. 5.10)

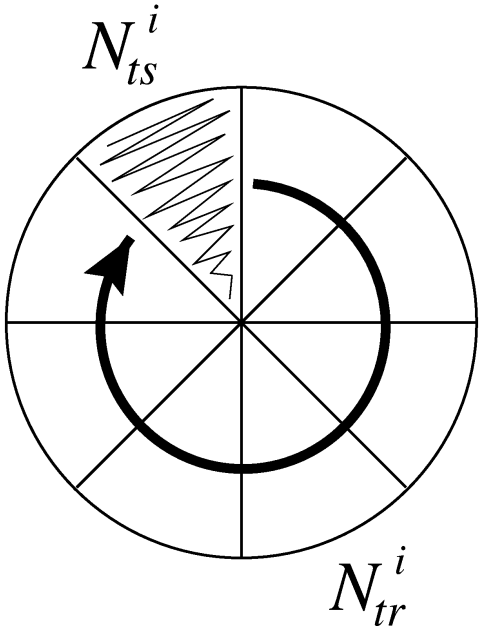



Figure 5.10: Partition of the training dataset in the $i^{\text{th}}$ fold of cross-validation. The quantity $N_{tr}^i$ is the number of training samples while $N_{ts}^i$ is the number of test samples.

$$\hat{R}_{ts}(\alpha^i_{N_{tr}}) = \sum_{j=1}^{N_{ts}} L(y_j, h(x_j, \alpha^i_{N_{tr}})) \tag{5.7.41}$$

The resulting average of the $l$ errors $\hat{R}_{ts}(\alpha^i_{N_{tr}})$, $i = 1, \ldots, l$, is the cross-validation estimate

$$\hat{G}_{\text{cv}} = \frac{1}{l} \sum_{i=1}^{l} R_{ts}(\alpha^i_{N_{tr}}) \tag{5.7.42}$$

A common form of cross-validation is the "leave-one-out" (l-o-o). Let $D_{(i)}$ be the training set with $z_i$ removed, and $h(x, \alpha_{N(i)})$ be the corresponding prediction rule. The l-o-o cross-validated error estimate is

$$\hat{G}_{\text{loo}} = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, h(x_i, \alpha_{N(i)})\right) \tag{5.7.43}$$

In this case $l$ equals the number of training samples and $N_{ts} = 1$.

*Bootstrap* (Section 4.4) is also used to return a nonparametric estimate of $G_N$, by repeatedly sampling the training cases *with replacement*.

A bootstrap sample $D_N^*$ is a "fake" dataset $\{z_1^*, z_2^*, \ldots, z_N^*\}$ randomly selected from the training set $\{z_1, z_2, \ldots, z_N\}$ with replacement. The bootstrap estimate of $G_N$, also called the E0 estimate in the bootstrap literature, is [40]

$$\hat{G}_{\text{boot}} = E_{\mathbf{D}_N^*} \left[ \sum_{i=1}^{N} P_i L\left(y_i, h(x_i, \alpha_{\text{boot}})\right) \right] \tag{5.7.44}$$

where $P_i$ indicates the proportion of the bootstrap sample on $z_i$,

$$P_i = \frac{\#_{j=1}^{N}(z_j^* = z_i)}{N} \tag{5.7.45}$$

and $\alpha_{\text{boot}}$ is the output of the parametric identification performed on the set $D_N^*$.

Usually $\hat{G}_{\text{boot}}$ must be evaluated by a Monte Carlo strategy. This means that independent bootstrap training sets $D_N^{*\,1}, D_N^{*\,2}, \ldots, D_N^{*\,N_b}$ are generated and for each of them the prediction rule $h(\cdot, \alpha_{\text{boot}}^j)$ is calculated. Eventually, the bootstrap estimate of $G_N$ is returned by

$$\hat{G}_{\text{boot}} = \frac{1}{N_b} \sum_{j=1}^{N_b} \sum_{i=1}^{N} P_i^j L\left(y_i, h(x_i, \alpha_{\text{boot}}^j)\right) \tag{5.7.46}$$

## 5.8 Concluding remarks

The goal of a learning procedure is to return a hypothesis which is able to predict accurately the outcome of an input/output probabilistic mapping on the basis of past observations. In order to achieve this goal, the learning procedure has to deal with three major difficulties.

**Minimization of the empirical risk:** in a general case finding the global minimum of the empirical risk as in (5.2.5) demands the resolution of a multivariate and nonlinear optimization problem for which no analytical solution could exist. Some heuristics to address this issue are discussed in Section 6.6.

**Finite number of data:** in real problems, only one random realization of the statistical process, made of a finite number of input/output samples, is accessible to the learning machine. This means, that the hypothesis generated by a learning algorithm is a random variable as well. In theory, it would be required to have access to the underlying process and to generate several times the training set, in order to have a reliable assessment of the learning algorithm. In practice, the use of repeated realizations is not viable in a real learning problem.

The validation procedure copes with this problem by trying to assess a random variable on the basis of a single realization. In particular we focused on cross-validation, a resampling method which works by simulating the stochastic process underlying the data.

**No a priori knowledge:** we consider a setting where no knowledge about the process underlying the data is available. This lack of a priori knowledge puts no constraints on the complexity of the class of hypotheses to consider, with the consequent risk of using an inadequate type of approximator. The model selection deals with this issue by considering classes of hypotheses of increasing complexity and selecting the one which behaves the best according to the validation criteria. This strategy ensures the covering of the whole spectrum of approximators, ranging from low bias/high variance to high bias/low variance models, making easier the selection of a good trade-off on the basis of the available data.

So far, the learning problem has been introduced and discussed for a generic class of hypotheses, and we did not distinguish on purpose between different learning machines. The following chapter will show the parametric and the structural identification procedure as well as the validation phase for some specific learning approaches.

# Chapter 6

# The machine learning procedure

## 6.1 Introduction

Raw data is rarely of direct benefit. Its true value resides in the amount of information that a model designer can extract from it. Modeling from data is often viewed as an art form, mixing the insight of the expert with the information contained in the observations. Typically, a modeling process is not a sequential process but is better represented as a sort of loop with a lot of feedbacks and a lot of interactions with the designer. Different steps are repeated several times aiming to reach, through continuous refinements, a good model description of the phenomenon underlying the data.

This chapter reviews the practical steps constituting the process of constructing models for accurate prediction from data. Note that the overview is presented with the aim of not distinguishing between the different families of approximators and of showing which procedures are common to the majority of modeling approaches. The following chapters will be instead devoted to the discussion of the peculiarities of specific learning approaches.

We partition the data modeling process in two phases: a *preliminary phase* which leads from the raw data to a structured training set and a *learning phase*, which leads from the training set to the final model. The preliminary phase is made of a *problem formulation* step (Section 6.2) where the designer selects the phenomenon of interest and defines the relevant input/output features, an *experimental design* step (Section 6.3) where input/output data are collected and a data *preprocessing* step (Section 6.4) where a preliminary filtering of data is performed.

Once the dataset has been formatted, the learning procedure begins. In qualitative terms, this procedure can be described as follows. First the designer defines a set of parametric models (e.g. polynomial models, neural networks) and a complexity index (or hyper parameter) which controls the approximation power of the model (e.g. degree of polynomial, number of neurons). By means of the complexity index, the set of parametric models is consequently decomposed in a nested sequence of classes of models (e.g. classes of polynomials with increasing degree). Hence, a structural identification procedure loops over the set of classes, first by identifying a parametric model for each class (*parametric identification*), and then by assessing the prediction error of the identified model on the basis of the finite set of samples (*validation*). Finally a *model selection* procedure selects the final model to be used for future predictions.

The problem of parametric identification is typically a problem of multivariate

optimization. Section 6.6 introduces the most common optimization algorithms for linear and nonlinear configurations. Structural identification is discussed in Section 6.7 which focuses on the existing methods for model generation, model validation and model selection. The last section concludes and resumes the whole modeling process with the support of a diagram.

## 6.2 Problem formulation

The problem formulation is the preliminary and somewhat the most important step of a learning procedure. The model designer chooses a particular application domain, a phenomenon to be studied and hypothesizes the existence of an unknown dependency which is to be estimated from experimental data. At this step, the modeler specifies a set of potentially relevant input and output variables and the sources of observations. Here domain-specific knowledge and experience are the most important requirements to come up with a meaningful problem formulation. Note that the time spent in this phase is usually very rewarding and can save vast amounts of modeling time. There is still no substitute for physical intuition and human analytical skills.

## 6.3 Experimental design

The most precious thing in data modeling is the data itself. No matter how powerful a learning method is, the resulting model would be ineffective if the data are not informative enough. Hence, it is necessary to devote a great deal of attention to the process of observing and collecting the samples. In input/output modeling it is essential that the training set be a representative sample of the phenomenon and cover adequately the input space. To this aim it is relevant to consider the relative importance of the various areas of the input space. Some regions are more relevant than others, as in the case of a dynamical system whose state has to be regulated about some specified operating point.

The discipline of creating an optimal sampling of the input space is called *experimental design* [47]. The study of experimental design is concerned with locating training input data in the space of input variables so that the performance of the modeling process is maximized. However, in some cases the designer cannot manipulate the process of collecting data and the model process has to deal with what is available. This configuration, which is common to many real problems, is called the *observational setting* [26], and this is the kind of situation we will assume in the rest of the chapter.

## 6.4 Data pre-processing

Once data have been recorded, it is common practice to preprocess them. The idea is that every treatment that can transform data to make learning easier will lead to a significant improvement in the final performance.

*Pre-processing* includes a large set of actions on the observed data but among them three operations are worth to be put into evidence:

**The missing data treatment.** In real applications it often happens that some input values are missing. If the quantity of data is sufficiently large, the simplest solution is to discard the samples having missing features. When the amount of data is too restricted or there are too many partial samples, it

becomes important to adopt some specific technique to deal with the problem. Various heuristics [58] as well as methods based on the Expectation Maximization (EM) algorithm [52] have been proposed in literature.

**Feature selection.** The feature selection problem consists in selecting a relevant subset of input variables in order to maximize the performance of the learning machine. This approach is useful if there are inputs which carry only little useful information or in case they are strongly correlated. In these situations a dimensionality reduction improves the performance reducing the variance of the estimator at the cost of a slight increase in the bias.

Several techniques exist for feature selection, such as conventional methods in linear statistical analysis [37], principal component analysis [89] and the general wrapper approach [75].

**Outliers removal.** Outliers are unusual data values which are not consistent with the most part of observations. Commonly, outliers are due to wrong measurement procedures, storage errors and coding malfunctioning. Two are the most common strategies to deal with outliers: the first is performed at the preprocessing stage [46] and consists in their detection and consequent removal, the second is to delay their treatment at the model identification level, by adopting robust methodologies [65] that are by design insensitive to outliers.

Other common preprocessing operations are *pre-filtering* to remove noise effects, *anti-aliasing* to deal with sampled signals, *variable scaling*, *compensation* of nonlinearities or the integration of some domain-specific information to reduce the distorting effects of measurable disturbances.

## 6.5 The dataset

The outcome of the preliminary data treatment is a training set

$$D_N = \{z_1, z_2, \ldots, z_N\}$$

where the $i^{\text{th}}$ sample is an input/output pair $z_i = \langle x_i, y_i \rangle$, $x_i$ is a $[n \times 1]$ input vector and $y_i$ is a scalar output.

Note that hereafter, for the sake of simplicity we will restrict ourselves to a regression setting. We will assume the input/output data to be generated as follows:

$$\mathbf{y}_i = f(\mathbf{x}_i) + \mathbf{w}_i, \tag{6.5.1}$$

where $\forall i$, $\mathbf{w}_i$ is a random variable such that $E[\mathbf{w}_i] = 0$ and $E[\mathbf{w}_i \mathbf{w}_j] = 0$, $\forall j \neq i$.

The assumption underlying Equation (6.5.1), i.e. that the noise enters additively to the output, implies some restrictions. Sometimes the measurements of the inputs to the system may also be noise corrupted; in system identification literature this is what is called the *error-in-variable* configuration [6]. As far as this problem is concerned, we adopt the pragmatic approach proposed by Ljung [79] which assumes that the measured input values are the actual inputs and that their deviations from the correct values propagate through $f$ and lump into the noise $\mathbf{w}$.

In the following we will refer to the set of vectors $x_i$ and $y_i$ through the following matrices:

1. the *input matrix* $X$ of dimension $[N \times n]$ whose $i^{\text{th}}$ row is the vector $x_i^T$,
2. the *output vector* $Y$ of dimension $[N \times 1]$ whose $i^{\text{th}}$ component is the scalar $y_i$.

## 6.6 Parametric identification

Assume that a class of hypotheses $h(\cdot, \alpha)$ with $\alpha \in \Lambda$ has been fixed. The problem of *parametric identification* from a finite set of data consists in seeking the hypothesis whose vector of parameters $\alpha_N \in \Lambda$ minimizes the loss function

$$R_{\text{emp}}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, h(x_i, \alpha)\right) \tag{6.6.2}$$

This phase of the learning procedure requires the resolution of the optimization task (5.2.5). In this section we will review some of the most common algorithms that address the problem of parametric identification in linear and nonlinear cases.

To make the notation more readable, henceforth we will define the *error function*

$$J(\alpha) = R_{\text{emp}}(\alpha)$$

and we will formulate the optimization problem (5.2.5) as

$$\alpha_N = \arg\min_{\alpha \in \Lambda} J(\alpha). \tag{6.6.3}$$

Also, we will use the term *model* as a synonymous of *hypothesis*.

### 6.6.1 Error functions

The choice of an optimization algorithm is strictly dependent on the form of the error function $J(\alpha)$. The function $J(\alpha)$ is directly determined by two factors

1. the form of the model $h(\cdot, \alpha)$ with $\alpha \in \Lambda$,
2. the loss function $L(y, h(x, \alpha))$ for a generic $x$.

As far as the cost function is concerned, there are many possible choices depending on the type of data analysis problem. In regression problems the goal is to model the conditional distribution of the output variable conditioned on the input variable (see Section 5.4) whose mean is the value minimizing the mean squared error. This motivates the use of a quadratic function

$$L\left(y, h(x, \alpha)\right) = \left(y - h(x, \alpha)\right)^2 \tag{6.6.4}$$

which gives to $J(\alpha)$ the form of a sum-of-squares.

For classification problems the goal is to model the posterior probabilities of class membership, again conditioned on the input variables. Although the sum-of-squares $J$ can be used also for classification, there are perhaps other more appropriate error functions to be considered [38].

Since this chapter will focus exclusively on regression problems, we will limit to consider the case of a quadratic loss function.

### 6.6.2 Parameter estimation

#### 6.6.2.1 The linear least-squares method

The parametric identification of a linear model

$$h(x, \alpha) = \alpha^T x$$

in the case of a quadratic cost, can be obtained through the well-known *linear least-squares method*. In this case $J(\alpha)$ is a quadratic function of the parameters and

there is a single minimum which can be obtained analytically. In Chapter 7 we will see in detail the linear least-squares minimization. Here we just report the final result

$$\alpha_N = (X^T X)^{-1} X^T Y \tag{6.6.5}$$

which holds in the case of non singularity of the matrix $(X^T X)$.

#### 6.6.2.2 Iterative search methods

In general cases, when either the model is not linear or the cost function is not quadratic, $J(\alpha)$ can be a highly nonlinear function of the parameters $\alpha$, and there may exist many minima all of which satisfy

$$\nabla J = 0 \tag{6.6.6}$$

where $\nabla$ denotes the gradient of $J$ in parameter space. We will define as *stationary points* all the points which satisfy condition (6.6.6). They include local maxima, saddle points and minima. The minimum for which the value of the error function is the smallest is called the *global* minimum while other minima are called *local* minima. As a consequence of the non-linearity of the error function $J(\alpha)$, it is not in general possible to find closed-form solutions for the minima. We will consider *iterative* algorithms which involve a search through the parameter space consisting of a succession of steps of the form

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} + \Delta\alpha^{(\tau)} \tag{6.6.7}$$

where $\tau$ labels the iteration step.

Iterative algorithms differ for the choice of the increment $\Delta\alpha^{(\tau)}$. In the following we will present some gradient-based and non gradient-based iterative algorithms. Note that each algorithm has a preferred domain of application and that it is not possible, or at least fair, to recommend a single universal optimization algorithm. We consider it much more interesting to highlight the relative advantages and limitations of the different approaches.

#### 6.6.2.3 Gradient-based methods

In some cases the analytic form of the error function makes it possible to evaluate the gradient of the cost function $J$ with respect to the parameters $\alpha$, increasing the rate of convergence of the iterative algorithm. Some examples of gradient-based methods are reported in the following.

#### 6.6.2.4 Gradient descent

It is the simplest of the gradient-based optimization algorithms, also known as *steepest descent*. This algorithm starts with some initial guess $\alpha^{(0)}$ for the parameter vector (often chosen at random). Then, it iteratively updates the parameter vector such that, at the $\tau^{\text{th}}$ step, the estimate is updated by moving a short distance in the direction of the negative gradient evaluated in $\alpha^{(\tau)}$:

$$\Delta\alpha^{(\tau)} = -\mu \nabla J(\alpha^{(\tau)}) \tag{6.6.8}$$

where $\mu$ is called the *learning rate*. If its value is sufficiently small, it is expected that the value of $J(\alpha^{(\tau)})$ will decrease at each successive step, eventually leading to a parameter vector at which the condition (6.6.6) is satisfied.

The gradient descent method is known to be a very inefficient procedure. One drawback is the need for a suitable value of the learning parameter $\mu$. Further,

at most points in the parameter space, the local gradient does not point directly towards the minimum: gradient descent then needs many small steps to reach a stationarity point.

In alternative to the simplest gradient descent there are many iterative methods in literature, as the *momentum* based method [96], the *enhanced gradient descent* method [118] and the *conjugate gradients* techniques [98], which make implicit use of second-order derivatives of the error function.

In the following section we present instead a class of algorithms which make explicit use of second-order information.

#### 6.6.2.5 The Newton method

The Newton's method is a well-known example in optimization literature. It is an iterative algorithm which uses at the $\tau^{\text{th}}$ step a local quadratic approximation in the neighborhood of $\alpha^{(\tau)}$

$$\hat{J}(\alpha) = J\left(\alpha^{(\tau)}\right) + \left(\alpha - \alpha^{(\tau)}\right)^T \nabla J\left(\alpha^{(\tau)}\right) + \frac{1}{2}\left(\alpha - \alpha^{(\tau)}\right)^T H\left(\alpha - \alpha^{(\tau)}\right) \quad (6.6.9)$$

where $H$ is the Hessian matrix of $J(\alpha)$ computed in $\alpha^{(\tau)}$. The minimum of (6.6.9) satisfies

$$\alpha^{min} = \alpha^{(\tau)} - H^{-1}\nabla J\left(\alpha^{(\tau)}\right) \quad (6.6.10)$$

where the vector $H^{-1}\nabla J\left(\alpha^{(\tau)}\right)$ is denoted as the Newton direction or the Newton step and forms the basis for the iterative strategy

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - H^{-1}\nabla J\left(\alpha^{(\tau)}\right) \quad (6.6.11)$$

There are several difficulties with such an approach, mainly related to the prohibitive computational demand. Alternative approaches, known as *quasi-Newton* or *variable metric* methods are based on (6.6.10) but instead of calculating the Hessian directly, and then evaluating its inverse, they build up an approximation to the inverse Hessian. The two most commonly used update formulae are the *Davidson-Fletcher-Powell* (DFP) and the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) procedures [80].

#### 6.6.2.6 The Levenberg-Marquardt algorithm

This algorithm is designed specifically for minimizing a sum-of-squares error function

$$J(\alpha) = \frac{1}{2}\sum_{i=1}^{N} L\left(y_i, h(x_i, \alpha)\right) = \frac{1}{2}\sum_{i=1}^{N} e_i^2 = \frac{1}{2}\|e\|^2 \quad (6.6.12)$$

where $e_i$ is the error for the $i^{\text{th}}$ training case, $e$ is the $[N \times 1]$ vector of errors and $\|\cdot\|$ is a 2-norm. Let us consider an iterative step in the parameter space

$$\alpha^{(\tau)} \rightarrow \alpha^{(\tau+1)} \quad (6.6.13)$$

If the quantity (6.6.13) is sufficiently small, the error vector $e$ can ban expanded in a first order Taylor series form:

$$e\left(\alpha^{(\tau+1)}\right) = e\left(\alpha^{(\tau)}\right) + E\left(\alpha^{(\tau+1)} - \alpha^{(\tau)}\right) \quad (6.6.14)$$

where the generic element of the matrix $E$ is in the form

$$E_{ij} = \frac{\partial e_i}{\partial \alpha_j} \quad (6.6.15)$$

and $\alpha_j$ is the $j^{\text{th}}$ element of the vector $\alpha$. The error function can then be approximated by

$$J\left(\alpha^{(\tau+1)}\right) = \frac{1}{2}\left\|e\left(\alpha^{(\tau)}\right) + E\left(\alpha^{(\tau+1)} - \alpha^{(\tau)}\right)\right\|^2 \tag{6.6.16}$$

If we minimize with respect to $\alpha^{(\tau+1)}$ we obtain:

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - (E^T E)^{-1} E^T e(\alpha^{(\tau)}) \tag{6.6.17}$$

where $(E^T E)^{-1} E^T$ is the pseudo-inverse of the matrix $E$. For the sum-of-squares error function (6.6.12), the elements of the Hessian take the form

$$H_{jk} = \frac{\partial^2 E}{\partial \alpha_j \alpha_k} = \sum_{i=1}^{N} \left\{ \frac{\partial e_i}{\partial \alpha_j} \frac{\partial e_i}{\partial \alpha_k} + e_i \frac{\partial^2 e_i}{\partial \alpha_j \alpha_k} \right\} \tag{6.6.18}$$

Neglecting the second term, the Hessian can be written in the form

$$H \simeq E^T E \tag{6.6.19}$$

This relation is exact in the case of linear models while in the case of nonlinearities it represents an approximation which holds exactly only at the global minimum of the function [20]. The update formula (6.6.17) could be used as the step of an iterative algorithm. However, the problem with such an approach could be a too large step size returned by (6.6.17), making the linear approximation no longer valid.

The idea of the Levenberg-Marquardt algorithm is to use the iterative step, at the same time trying to keep the step size small so as to guarantee the validity of the linear approximation. This is achieved by modifying the error function in the form

$$J_{lm} = \frac{1}{2}\left\|e(\alpha^{(\tau)}) + E(\alpha^{(\tau+1)} - \alpha^{(\tau)})\right\|^2 + \lambda\left\|\alpha^{(\tau+1)} - \alpha^{(\tau)}\right\|^2 \tag{6.6.20}$$

where $\lambda$ is a parameter which governs the step size. The minimization of the error function (6.6.20) ensures at the same time the minimization of the sum-of-square cost and a small step size. Minimizing (6.6.20) with respect to $\alpha^{(\tau+1)}$ we obtain

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - (E^T E + \lambda I)^{-1} E^T e(\alpha^{(\tau)}) \tag{6.6.21}$$

where $I$ is the unit matrix. For very small values of $\lambda$ we have the Newton formula, while for large values of $\lambda$ we recover the standard gradient descent.

A common approach for setting $\lambda$ is to begin with some arbitrary low value (e.g. $\lambda = 0.1$) and at each step (6.6.21) check the change in $J$. If $J$ decreases the new parameter is retained, $\lambda$ is decreased (e.g. by a factor of 10), and the process repeated. Otherwise, if $J$ increased after the step (6.6.21), the old parameter is restored, $\lambda$ decreased and a new step performed. The procedure is iterated until a decrease in $J$ is obtained [20].

#### 6.6.2.7 Alternatives to gradient-based methods

Virtually no gradient-based method is guaranteed to find the global optimum of a complex nonlinear error function. Additionally, all descent methods are deterministic in the sense that they inevitably lead to convergence to the nearest local minimum. As a consequence, the way a deterministic method is initialized, is decisive for the final result.

Further, in many practical situations the gradient-based computation is time consuming or extremely difficult due to the complexity of the objective function. For these reasons, a lot of derivative-free and stochastic alternatives to gradient-based methods have been explored in literature. We will limit to cite the most common solutions:

**Random search methods.** They are iterative methods which are primarily used for continuous optimization problems. Random search methods explore the parameter space of the error function sequentially in a random fashion to find the global minimum. Their strength lies mainly in their simplicity, which makes these methods easily understood and conveniently customized for specific applications. Moreover, it has been demonstrated that they converge to the global optimum with probability one on a compact set. However, the theoretical result of convergence to the minimum is not really important here since the optimization process could take a prohibitively long time.

**Genetic algorithms.** They are derivative-free stochastic optimization methods based loosely on the concepts of natural selection and evolutionary processes [53]. Important properties are the strong parallelism and the possibility to be applied to both continuous and discrete optimization problems. Typically, Genetic Algorithms (GA) encode each parameter solution into a binary bit string (*chromosome*) and associate each solution with a *fitness* value. GAs usually keep a set of solutions (*population*) which is evolved repeatedly toward a better overall fitness value. In each *generation*, the GA constructs a new population using *genetic operators* such as crossover or mutation; members with higher fitness values are more likely to survive and to participate in future operations. After a number of generations, the population is expected to contain members with better fitness values and to converge, under particular conditions, to the optimum.

**Simulated annealing.** It is another derivative-free method suitable for continuous and discrete optimization problems. In Simulated Annealing (SA), the value of the cost function $J(\alpha)$ to be minimized is put in analogy to the energy in a thermodynamic system at a certain temperature $T$ [73]. At high temperatures $T^{(\tau)}$, the SA technique allows function evaluations at points far away from $\alpha^{(\tau)}$ and it is likely to accept a new parameter value with higher function value. The decision whether to accept or reject a new parameter value $\alpha^{(\tau+1)}$ is based on the value of an acceptance function, generally shaped as the *Boltzmann probability distribution*. At low temperatures SA evaluates the objective function at more local points and the likelihood of accepting a new point with an higher cost is much lower. An *annealing schedule* regulates how rapidly the temperature $T$ goes from high values to low values, as a function of time or iteration counts.

## 6.7 Structural identification

Once a class of models $\Lambda$ is given, the identification procedure, described above, returns a model $h(, \cdot, \alpha_N)$ defined by the set of parameters $\alpha_N \in \Lambda$ .

The choice of an appropriate class, or *model structure* [79] is, however, the most crucial aspect for a successful modeling application. The procedure of selecting the best model structure, is called *structural identification*. The structure of a model is made of a series of features which influence the generalization power of the model itself. Among others, there are:

- The type of the model. We can distinguish, for example, between nonlinear and linear models, between physically based and black box representations, between continuous-time or discrete-time systems.

- The size of the model. This is related to features like the number of inputs (regressors), the number of parameters, the degree of the polynomials in a

polynomial model, the number of neurons in a neural network, the number of nodes in a classification tree, etc.

In general terms, structural identification requires (i) a procedure for proposing a series of alternative model structures, (ii) a method for assessing each of these alternatives and (iii) a technique for choosing among the available candidates.

We denote the first issue as *model generation*. Some techniques for obtaining different candidates to the final model structure are presented in Section 6.7.1.

The second issue concerns the important problem of *model validation*, and will be extensively dealt with in Section 6.7.2.

Once models have been generated and validated, the last step is the *model selection*, which will be discussed in Section 6.7.3

It is important to remark that a selected model structure should never be accepted as a final and true description of the phenomenon. Rather, it should be regarded as a good enough description, given the available dataset.

### 6.7.1 Model generation

Goal of the model generation procedure is to generate a set of candidate model structures among which the best one is to be selected. The more this procedure is effective, the easier will be the selection of a powerful structure at the end of the whole identification. Traditionally there have been a number of popular ways to search through a large collection of model structures. Maron and Moore [83] distinguish between two main methods of model generation:

**Brute force.** This is the exhaustive approach. Every possible model structure is generated in order to be evaluated.

Consider for instance the problem of selecting the best structure in a 3-layer Feed Forward Neural Network architecture. The brute force approach consists in enumerating all the possible configurations in terms of number of neurons.

The exhaustive algorithm runs in a time which is generally unacceptably slow for complex architecture with a large number of structural parameters. The only advantage, however, is that this method is guaranteed to return the best learner according to a specified assessment measure.

**Search methods.** These methods treats the collection of models as a continuous and differentiable surface. They start at some point on the surface and search for the model structure that corresponds to the minimum of the generalization error, until some stop condition is met. This procedure is much faster than brute force since it does not need to explore all the space. It only needs to validate those models that are on the search path. Gradient-based techniques and/or non-gradient based methods can be used for the search in the model space. Besides the well-known problem related to local minima in the gradient base case, a more serious issues derives from the structure of the model selection procedure. At every step of the search algorithm we need to find a collection of models that are near or related to the current model. Both gradient based and non gradient based techniques require some metric in the search space. This implies a notion of model distance, difficult to define in a general model selection problem. Examples of search methods in model generation are the growing and pruning techniques in Neural Networks structural identification [12].

### 6.7.2 Validation

The output of the model generation procedure is a set of model structures $\Lambda_s$, $s = 1, \ldots, S$. Once the parametric identification is performed on each of these model structures, we have a set of models $h(\cdot, \alpha_N^s)$ identified according to the Empirical Risk Minimization principle.

Now, the prediction quality of each one of the model structures $\Lambda_s$, $s = 1, \ldots, S$, has to be assessed on the basis of the available data. In principle, the assessment procedure, known as *model validation*, could measure the goodness of a structure in many different ways: how the model relates to our a priori knowledge, how the model is easy to be used, to be implemented or to be interpreted. In this book, as stated in the introduction, we will focus only on criteria of accuracy, neglecting any other criterion of quality.

In the following, we will present the most common techniques to assess a model on the basis of a finite set of observations.

#### 6.7.2.1 Testing

An obvious way to assess the quality of a learned model is by using a *testing sequence*

$$D_{ts} = (\langle x_{N+1}, y_{N+1} \rangle, \ldots, \langle x_{N+N_{ts}}, y_{N+N_{ts}} \rangle) \tag{6.7.22}$$

that is a sequence of i.i.d. pairs, independent of $D_N$ and distributed according to the probability distribution $P(x, y)$ defined in (5.2.2). The testing estimator is defined by the sample mean

$$\hat{R}_{ts}(\alpha_N^s) = \frac{1}{N_{ts}} \sum_{j=1}^{N_{ts}} \left(y_j - h(x_j, \alpha_N^s)\right)^2 \tag{6.7.23}$$

This estimator is clearly unbiased in the sense that

$$E_{\mathbf{D}_{ts}}[\hat{\mathbf{R}}_{ts}(\alpha_N^s)] = R(\alpha_N^s) \tag{6.7.24}$$

When the number of available examples is sufficiently high, the testing technique is an effective validation technique at low computational cost. A serious problem concerning the practical applicability of this estimate is that it requires a large, independent testing sequence. In practice, unfortunately, an additional set of input/output observations is rarely available.

#### 6.7.2.2 Holdout

The *holdout* method, sometimes called test sample estimation, partitions the data $D_N$ into two mutually exclusive subsets, the training set $D_{tr}$ and the holdout or test set $D_{N_{ts}}$. It is common to design 2/3 of the data as training set and the remaining 1/3 as test set. However, when the training set has a reduced number of cases, the method can present a series of shortcomings, mainly due to the strong dependence of the prediction accuracy on the repartition of the data between training and validation set. Assuming that the error $R(\alpha_{N_{tr}})$ decreases as more cases are inserted in the training set, the holdout method is a pessimistic estimator since only a reduced amount of data is used for training. The more samples used for test set, the higher the bias of the estimate $\alpha_{N_{tr}}$; at the same time, fewer test samples implies a larger confidence interval of the estimate of the generalization error.

#### 6.7.2.3 Cross-validation in practice

In Chapter 5 we focused on the theoretical properties of cross-validation and bootstrap. Here we will see some more practical details on these validation procedures, commonly grouped under the name of *computer intensive statistical methods* (CISM) [63].

Consider a learning problem with a training set made of $N$ samples.

In $l$-fold cross-validation the available cases are randomly divided into $l$ mutually exclusive test partitions of approximately equal size. The cases not found in each test partition are independently used for selecting the hypothesis which will be tested on the partition itself (Fig. 5.10).

The average error over all the $l$ partitions is the cross-validated error rate.

A special case is the leave-one-out (l-o-o). For a given algorithm and a data set $D_N$, an hypothesis is generated using $N-1$ cases and tested on the single remaining case. This is repeated $N$ times, each time designing an hypothesis by leaving-one-out. Thus, each case in the sample is used as a test case, and each time nearly all the cases are used to design a hypothesis. The error estimate is the average over the $N$ repetitions.

In a general nonlinear case leave-one-out is a computationally quite expensive procedure. This is not true for a linear model where the PRESS l-o-o statistic is computed as a by-product of the least-squares regression (Section 7.2).

#### 6.7.2.4 Bootstrap in practice

Bootstrap is a resampling technique which samples the training set with replacement to return a nonparametric estimate of a desired statistic.

There are many bootstrap estimators, but two are the most commonly used in model validation: the E0 and the E632 bootstrap.

The E0 bootstrap estimator, denoted by $\hat{G}_{boot}$ in Section 5.7.1, samples with replacement from the original training set $N_b$ bootstrap training sets, each consisting of $N$ cases. The cases not found in the training group form the test groups. The average of the error rate on the $N_b$ test groups is the E0 estimate of the generalization error.

The rationale for the E632 technique is given by Efron [40]. He argues that while the resubstitution error $R_{\text{emp}}$ is the error rate for patterns which are "zero" distance from the training set, patterns contributing to the E0 estimate can be considered as "too far out" from the training set. Since the asymptotic probability that a pattern will not be included in a bootstrap sample is

$$(1 - 1/N)^N \approx e^{-1} \approx 0.368$$

the weighted average of $R_{\text{emp}}$ and E0 should involve patterns at the "right" distance from the training set in estimating the error rate:

$$\hat{G}_{\text{E632}} = 0.368 * R_{\text{emp}} + 0.632 * \hat{G}_{boot} \tag{6.7.25}$$

where the quantity $\hat{G}_{boot}$ is defined in (5.7.44). The choice of $N_b$ is not critical as long as it exceeds 100. Efron [40] suggests, however, that $N_b$ need not be greater than 200.

There a lot of experimental results on comparison between cross-validation and bootstrap methods for assessing models [66], [74]. In general terms only some guidelines can be given to the practitioner [121]:

- For training set size greater than 100 use cross-validation, either 10-fold cross-validation or leave-one-out is acceptable.

- For training set sizes less than 100 use leave-one-out.
- For very small training set (fewer than 50 samples) in addition to the leave-one-out estimator, the $\hat{G}_{\text{E632}}$ and the $\hat{G}_{boot}$ estimates may be useful measures.

#### 6.7.2.5 Complexity based criteria

In conventional statistics, various criteria have been developed, often in the context of linear models, for assessing the generalization performance of the learned hypothesis without the use of further validation data. Such criteria aim to understand the relationship between the generalization performance and the training error. Generally, they take the form of a prediction error which consists of the sum of two terms

$$\hat{G}_{\text{PE}} = R_{\text{emp}} + \text{complexity term} \tag{6.7.26}$$

where the complexity term represents a penalty which grows as the number of free parameters in the model grows.

This expression quantifies the qualitative consideration that simple models return high empirical risk with a reduced complexity term while complex models have a low empirical risk thanks to the high number of parameters. The minimum for the criterion (6.7.26) represents a trade-off between performance on the training set and complexity. Let us consider a quadratic loss function and the quantity

$$\widehat{\text{MISE}}_{\text{emp}} = R_{\text{emp}}(\alpha_N) = \min_{\alpha} \frac{\sum_{i=1}^{N}(y_i - h(x_i, \alpha))^2}{N}$$

If the input/output relation is linear and $n$ is the number of input variables, well-known examples of complexity based criteria are:

1. the Final Prediction Error (FPE) (see Section 7.1.16 and [4])

$$\text{FPE} = \widehat{\text{MISE}}_{\text{emp}} \frac{1 + p/N}{1 - p/N} \tag{6.7.27}$$

   with $p = n + 1$,

2. the Predicted Squared Error (PSE) (see Section 7.1.16)

$$\text{PSE} = \widehat{\text{MISE}}_{\text{emp}} + 2\hat{\sigma}_w^2 \frac{p}{N} \tag{6.7.28}$$

   where $\hat{\sigma}_w^2$ is an estimate of the variance of noise. This quantity is also known as the Mallows' $C_p$ statistics [82]

3. the Generalized Cross-Validation (GCV) [31]

$$\text{GCV} = \widehat{\text{MISE}}_{\text{emp}} \frac{1}{(1 - \frac{p}{N})^2} \tag{6.7.29}$$

A comparative analysis of these different measures is reported in [10].

These estimates are computed assuming a linear model underlying the data. Moody [84] introduced the Generalized Prediction Error (GPE) as an estimate of the prediction risk for generic biased nonlinear models. The algebraic form is:

$$\text{GPE} = \widehat{\text{MISE}}_{\text{emp}} + \frac{2}{N} \text{tr} \hat{V} \hat{R} \tag{6.7.30}$$

where $\mathrm{tr}(\cdot)$ denotes the trace, $\hat{V}$ is a nonlinear generalization of the estimated *noise covariance* matrix of the target and $\hat{R}$ is the estimated *generalized influence matrix*. GPE can be expressed in an equivalent form as:

$$\mathrm{GPE} = \widehat{\mathrm{MISE}}_{\mathrm{emp}} + 2\hat{\sigma}_{\mathrm{eff}}\frac{\hat{p}_{\mathrm{eff}}}{N} \tag{6.7.31}$$

where $\hat{p}_{\mathrm{eff}} = \mathrm{tr}\hat{R}$ is the estimated effective number of model parameters, and $\hat{\sigma}_{\mathrm{eff}} = \frac{\mathrm{tr}\hat{V}\hat{R}}{\mathrm{tr}\hat{R}}$ is the estimated effective noise variance in the data. For nonlinear models, $\hat{p}_{\mathrm{eff}}$ is generally not equal to the number of parameters (e.g. number of weights in a neural network). When the noise in the target variables is assumed to be independent with uniform variance and the squared error loss function is used, (6.7.31) simplifies to:

$$\mathrm{GPE} = \widehat{\mathrm{MISE}}_{\mathrm{emp}} + 2\hat{\sigma}_w^2\frac{\hat{p}_{\mathrm{eff}}}{N} \tag{6.7.32}$$

In neural network literature another well-known form of complexity-based criterion is the *weight decay technique*

$$U(\lambda, \alpha, D_N) = \sum_{i=1}^{N}(y_i - h(x_i, \alpha))^2 + \lambda g(\alpha) \tag{6.7.33}$$

where the second term penalizes either small, medium or large weights of the neurons depending on the form of $g(\cdot)$. Two common examples of weight decay functions are the *ridge regression* form $g(\alpha) = \alpha^2$ which penalizes large weights, and the Rumelhart form $g(\alpha) = \frac{\alpha^2}{\alpha_0+\alpha^2}$ which penalizes weights of intermediate values near $\alpha_0$.

Another important method for model validation is the *minimum-description length* principle proposed by Rissanen [102]. This method proposes to choose the model which induces the shortest description for the data available. Rissanen and Barron [10] have each shown a qualitative similarity between this principle and the complexity based approaches. For further details refer to the cited works.

#### 6.7.2.6 A comparison between validation methods

Computer intensive statistical methods are relatively new and must be measured against more established statistical methods, as the complexity based criteria. In the following we summarize some practical arguments on behalf of one or the other method. The benefits of a CISM method are:

- All the assumptions of prior knowledge on the process underlying the data are discarded.
- The validation technique replaces theoretical analysis by computation.
- Results are generally much easier to grasp for non-theorist.
- No assumption on the statistical properties of noise is required.
- They return an estimate of the model precision and an interval of confidence.

Arguments on behalf of complexity criteria are:

- The whole dataset can be used for estimating the prediction performance and no partitioning is required.

- Results valid for linear models remain valid to the extent that the nonlinear model can be approximated by some first-order Taylor expansion in the parameters.

Some results in literature show the relation existing between resampling and complexity based methods. For example, an asymptotic relation between a kind of cross-validation and the Akaike's measure was derived by Stone [111], under the assumptions that the real model $\alpha^*$ is contained in the class of hypothesis $\Lambda$ and that there is a unique minimum for the log-likelihood.

Here we will make the assumption that no a priori information about the correct structure or the quasi-linearity of the process is available. This will lead us to consider computer intensive methods as the preferred method to validate the learning algorithms.

### 6.7.3 Model selection criteria

Model selection concerns the final choice of the model structure in the set that has been proposed by model generation and assessed by model validation. In real problems, this choice is typically a subjective issue and is often the result of a compromise between different factors, like the quantitative measures, the personal experience of the designer and the effort required to implement a particular model in practice.

Here we will reduce the subjectivity factors to zero, focusing only on a quantitative criterion of choice. This means that the structure selection procedure will be based only on the indices returned by the methods of Section 6.7.2. We distinguish between two possible quantitative approaches: the *winner-takes-all* and the *combination of estimators* approach.

#### 6.7.3.1 The winner-takes-all approach

This approach chooses the model structure that minimizes the generalization error according to one of the criteria described in Section 6.7.2.

Consider a set of candidate model structures $\Lambda_s$, $s = 1, \ldots, S$, and an associated measure $\hat{G}(\Lambda_s)$ which quantifies the generalization error.

The winner-takes-all method simply picks the structure

$$\bar{s} = \arg\min \hat{G}(\Lambda_s) \tag{6.7.34}$$

that minimizes the generalization error.

The model which is returned as final outcome of the learning process is then $h(\cdot, \alpha_N^{\bar{s}})$.

#### 6.7.3.2 The combination of estimators approach

The winner-takes-all approach is intuitively the approach which should work the best. However, recent results in machine learning [95] show that the performance of the final model can be improved not by choosing the model structure which is expected to predict the best but by creating a model whose output is the combination of the output of models having different structures.

The reason for that non intuitive result is that in reality any chosen hypothesis $h(\cdot, \alpha_N)$ is only an estimate of the real target and, like any estimate, is affected by a bias and a variance term.

Section 3.10 presented some results on the combination of estimators. The extension of these results to supervised learning is the idea which underlies the first results in combination [11] and that has led later to more enhanced forms of averaging different models.

Consider $m$ different models $h(\cdot, \alpha_j)$ and assume they are unbiased and uncorrelated. By (3.10.28), (3.10.29) and (3.10.30) we have that the combined model is

$$h(\cdot) = \frac{\sum_{j=1}^{m} \frac{1}{\hat{v}_j} h(\cdot, \alpha_j)}{\sum_{j=1}^{m} \frac{1}{\hat{v}_j}} \tag{6.7.35}$$

where $\hat{v_j}$ is an estimation of the variance of $h(\cdot, \alpha_j)$. This is an example of the *generalized ensemble method* (GEM) [95].

More advanced applications of the combination principle to supervised learning will be discussed in Chapter 9.

## 6.8 Concluding remarks

The chapter presented the steps to extract a model from a finite set of input/output data (Figure 6.1). They were presented in sequence but it should be kept in mind that each modeling process is in reality a loop procedure with a lot of feedbacks and adjustments to be made each time some information is returned at a certain stage. An example is the identification procedure which is made of two nested loops where the inner one computes the optimal parameter for a fixed model structure, while the external one searches for the best configuration.

All the steps described in this chapter are equally important for the success of a learning procedure. However, the steps described in Section 6.2-6.4 are strongly domain-dependent and they are inevitably beyond the scope of the book. In the following chapters we will discuss extensively the model identification procedures of specific linear and nonlinear learning approaches.

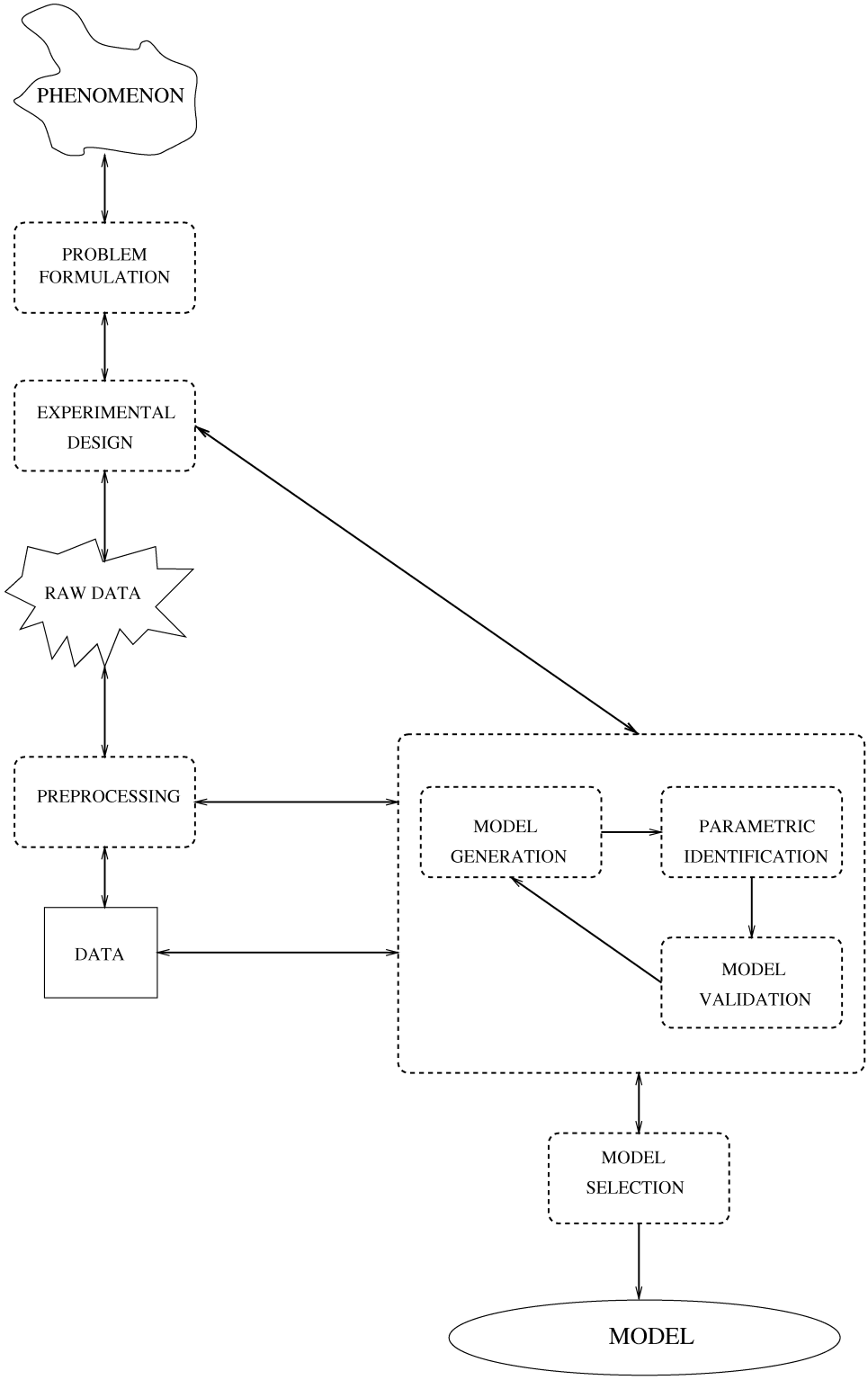



Figure 6.1: From the phenomenon to the prediction model: overview of the steps constituting the modeling process.

# Chapter 7

# Linear approaches

The previous chapters distinguished between two types of supervised learning tasks according to the type of output:

**Regression** when we predict quantitative outputs, e.g. real or integer numbers. Predicting the weight of an animal on the basis of its age and height is an example of regression problem.

**Classification (or pattern recognition)** where we predict qualitative or categorical outputs which assume values in a finite set of classes (e.g. black, white and red) where there is no explicit ordering. Qualitative variables are also referred to as **factors**. Predicting the class of an email on the basis of English words frequency is an example of classification task.

This chapter will consider learning approaches to classification and regression where the hypothesis functions are linear combinations of the input variables.

## 7.1 Linear regression

Linear regression is a very old technique in statistics and traces back to the work of Gauss.

### 7.1.1 The univariate linear model

The simplest regression model is the univariate linear regression model where the input is supposed to be a scalar variable and the stochastic dependency between input and output is described by

$$\mathbf{y} = \beta_0 + \beta_1 x + \mathbf{w} \tag{7.1.1}$$

where $x$ is the regressor (or independent) variable, $\mathbf{y}$ is the measured response (or dependent) variable, $\beta_0$ is the intercept, $\beta_1$ is the slope and $\mathbf{w}$ is called *noise* or *model error*. We will assume that $E[\mathbf{w}] = 0$ and that its variance $\sigma^2_{\mathbf{w}}$ is independent of the $x$ value. The assumption of constant variance is often referred to as *homostedasticity*.

From (7.1.1) we obtain

$$\text{Prob}\{\mathbf{y} = y|x\} = \text{Prob}\{\mathbf{w} = y - \beta_0 - \beta_1 x\}, \qquad E[\mathbf{y}|x] = f(x) = \beta_0 + \beta_1 x$$

The function $f(x) = E[\mathbf{y}|x]$, also known as *regression function*, is a linear function in the parameters $\beta_0$ and $\beta_1$ (Figure 7.1).

In the following we will intend as *linear model* each input/output relationship which is *linear in the parameters* but not necessarily in the dependent variables. This means that

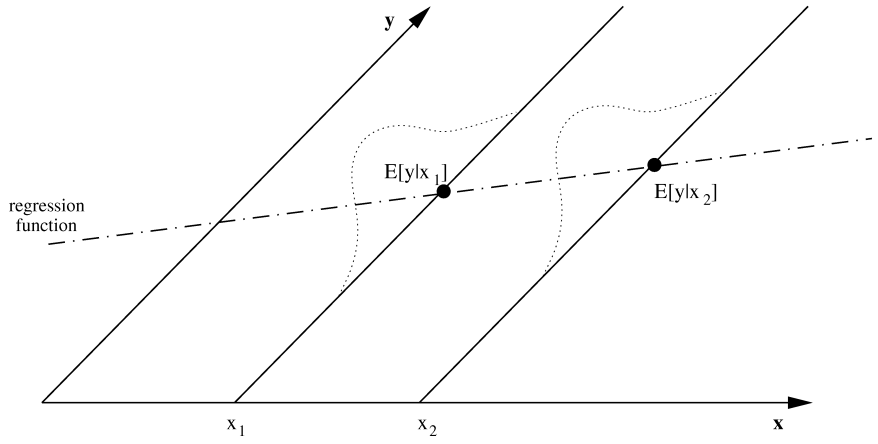



Figure 7.1: Conditional distribution and regression function for stochastic linear dependence

1. any value of the response variable $y$ is described by a linear combination of a series of parameters (regression slopes, intercept)
2. no parameter appears as an exponent or is multiplied or divided by another parameter.

According to our definition of linear model, then

- $y = \beta_0 + \beta_1 x$ is a linear model
- $y = \beta_0 + \beta_1 x^2$ is again a linear model. Simply by making the transformation $X = x^2$, the dependency can be put in in the linear form (7.1.1).
- $y = \beta_0 x^{\beta_1}$ can be studied as a linear model between $Y = \log(y)$ and $X = \log(x)$ thanks to the equality

$$\log(y) = \beta_0 + \beta_1 \log(x) \Leftrightarrow Y = \beta_0 + \beta_1 X$$

- the relationship $y = \beta_0 + \beta_1 \beta_2^x$ is not linear since there is no way to linearize it.

### 7.1.2 Least-squares estimation

Suppose that $N$ pairs of observations $(x_i, y_i)$, $i = 1, \ldots, N$ are available. Let us assume that data are generated by the following stochastic dependency

$$y_i = \beta_0 + \beta_1 x_i + w_i, \qquad i = 1, \ldots, N \tag{7.1.2}$$

where

1. the $\mathbf{w}_i$ are iid realizations of the r.v. $\mathbf{w}$ having mean zero and constant variance $\sigma^2_{\mathbf{w}}$ (homoscedasticity),
2. the $x_i$ are non random and observed with negligible error.

The unknown parameters (also known as *regression coefficients*) $\beta_0$ and $\beta_1$ can be estimated by the *least-squares method*. The method of least squares is designed to provide

1. the estimations $\hat{\beta}_0$ and $\hat{\beta}_1$ of $\beta_0$ and $\beta_1$, respectively
2. the fitted values of the response $y$

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i, \quad i = 1, \ldots, N$$

so that the *residual sum of squares* (which is $N$ times the *empirical risk*)

$$\mathrm{SSE}_{\mathrm{emp}} = N \cdot \widehat{\mathrm{MISE}}_{\mathrm{emp}} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{N} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$$

is minimized. In other terms

$$\{\hat{\beta}_0, \hat{\beta}_1\} = \arg \min_{\{b_0, b_1\}} \sum_{i=1}^{N} (y_i - b_0 - b_1 x_i)^2$$

It can be shown that the least-squares solution is

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}} \qquad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

where

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}, \quad \bar{y} = \frac{\sum_{i=1}^{N} y_i}{N}$$

and

$$\begin{aligned} S_{xy} &= \sum_{i=1}^{N} (x_i - \bar{x}) y_i \\ S_{xx} &= \sum_{i=1}^{N} (x_i - \bar{x})^2 = \sum_{i=1}^{N} (x_i^2 - 2 x_i \bar{x} + \bar{x}^2) = \sum_{i=1}^{N} (x_i^2 - x_i \bar{x} - x_i \bar{x} + \bar{x}^2) = \\ &= \sum_{i=1}^{N} [(x_i - \bar{x}) x_i] + \sum_{i=1}^{N} [\bar{x}(\bar{x} - x_i)] = \sum_{i=1}^{N} (x_i - \bar{x}) x_i \end{aligned}$$

**R script**

The script `lin_uni.R` computes and plots the least-squares solution for $N = 100$ samples generated according to the dependency (7.1.2) where $\beta_0 = 2$ and $\beta_1 = -2$.

```
## script lin_uni.R
set.seed(0)
N<-100
sigma.w<-0.5
X<-rnorm(N)
W<-rnorm(N,sd=0.5)

beta0 <- 2
beta1 <- -2
Y<- beta0+beta1*X+W

plot(X,Y)

xhat<-mean(X)
yhat<-mean(Y)
```

```
Sxy<- sum((X-xhat)*Y)
Sxx<- sum((X-xhat)*X)

betahat1<-Sxy/Sxx
betahat0<-yhat-betahat1*xhat

lines(X,betahat0+X*betahat1)
```

•

If the dependency underlying the data is linear then the estimators are unbiased. We show this property for $\hat{\boldsymbol{\beta}}_1$:

$$E_{\mathbf{D}_N}[\hat{\boldsymbol{\beta}}_1] = E_{\mathbf{D}_N}\left[\frac{S_{xy}}{S_{xx}}\right] = \sum_{i=1}^{N}\frac{(x_i-\bar{x})E[\mathbf{y}_i]}{S_{xx}} = \frac{1}{S_{xx}}\sum_{i=1}^{N}(x_i-\bar{x})(\beta_0+\beta_1 x_i) =$$
$$= \frac{1}{S_{xx}}\left[\sum_{i=1}^{N}[(x_i-\bar{x})\beta_0] + \sum_{i=1}^{N}[(x_i-\bar{x})\beta_1 x_i]\right] = \frac{\beta_1 S_{xx}}{S_{xx}} = \beta_1$$

Note that the analytical derivation relies on the relation $\sum_{i=1}^{N}(x_i-\bar{x})=0$ and the fact that $x$ is not a random variable. Also it can be shown [89] that

$$\text{Var}\left[\hat{\boldsymbol{\beta}}_1\right] = \frac{\sigma^2}{S_{xx}} \tag{7.1.3}$$
$$E[\hat{\boldsymbol{\beta}}_0] = \beta_0 \tag{7.1.4}$$
$$\text{Var}\left[\hat{\boldsymbol{\beta}}_0\right] = \sigma^2\left(\frac{1}{N}+\frac{\bar{x}^2}{S_{xx}}\right) \tag{7.1.5}$$

Another important result in linear regression is that the quantity

$$\hat{\sigma}^2_{\mathbf{w}} = \frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{N-2}$$

is an unbiased estimator of $\sigma^2_{\mathbf{w}}$ *under the (strong) assumption that samples have been generated according to (7.1.1)*. The denominator is often referred to as the *residual degrees of freedom*, also denoted by df. The degree of freedom can be be seen as the number $N$ of samples reduced by the numbers of parameters estimated (slope and intercept). The estimate of the variance $\sigma^2_{\mathbf{w}}$ can be used in Equations (7.1.5) and (7.1.3) to derive an estimation of the variance of the intercept and slope, respectively.

### 7.1.3 Maximum likelihood estimation

The properties of least-squares estimators rely on the only assumption that the

$$w_i = y_i - \beta_0 - \beta_1 x_i \tag{7.1.6}$$

are iid realizations with mean zero and constant variance $\sigma^2_{\mathbf{w}}$. Therefore, no assumption is made concerning the probability distribution of $\mathbf{w}$ (e.g. Gaussian or uniform). On the contrary, if we want to use the maximum likelihood approach (Section 3.8), we have to define the distribution of $\mathbf{w}$. Suppose that $\mathbf{w} \sim \mathcal{N}(0, \sigma^2_{\mathbf{w}})$. By using (7.1.6), the likelihood function can be written as

$$L_N(\beta_0,\beta_1) = \prod_{i=1}^{N} p_{\mathbf{w}}(w_i) = \frac{1}{(2\pi)^{N/2}\sigma^N}\exp\left\{\frac{-\sum_{i=1}^{N}(y_i-\beta_0-\beta_1 x_i)^2}{2\sigma^2_{\mathbf{w}}}\right\}$$

It can be shown that the estimates of $\beta_0$ and $\beta_1$ obtained by maximizing $L_N(\cdot)$ under the normal assumption are identical to the ones obtained by least squares estimation.

### 7.1.4 Partitioning the variability

An interesting way of assessing the quality of a linear model is to evaluate which part of the output variability the model is able to explain. We can use the following relation

$$\sum_{i=1}^{N}(y_i - \bar{y})^2 = \sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

i.e.

$$SS_{\text{Tot}} = SS_{\text{Mod}} + SS_{\text{Res}}$$

where $SS_{\text{Tot}}$ (which is also $N$ times the sample variance of $\mathbf{y}$) represents the total variability of the response, $SS_{\text{Mod}}$ is the variability explained by the model and $SS_{\text{Res}}$ is the variability left unexplained. This partition aids in the determination of whether the variation explained the regression model is real or is no more than chance variation. It will be used in the following section to perform hypothesis test on the quantities estimated by the regression model.

### 7.1.5 Test of hypotheses on the regression model

Suppose that we want to answer the question whether the regression variable $x$ truly influences the distribution $F_{\mathbf{y}}(\cdot)$ of the response $\mathbf{y}$ or, in other words, that they are linearly dependent. We can formulate the problem as an hypothesis testing problem on the slope $\beta_1$ where

$$H : \beta_1 = 0, \qquad \bar{H} : \beta_1 \neq 0$$

If $H$ is true this means that the regressor variable does not influence the response (at least not through a linear relationship). Rejection of $H$ in favor of $\bar{H}$ leads to the conclusion that $x$ significantly influences the response in a linear fashion. It can be shown that, under the assumption that $\mathbf{w}$ is normally distributed, if the null hypothesis $H$ (null correlation) is true then

$$\frac{SS_{\text{Mod}}}{SS_{\text{Res}}/(N-2)} \sim F_{1,N-2}.$$

Large values of the $F$ statistic (Section 2.7.7) provide evidence in favor of $\bar{H}$ (i.e. a linear trend exists). The test is a two-sided test. In order to perform a single-sided test, typically $\mathcal{T}$-statistics are used.

#### 7.1.5.1 The t-test

We want to test whether the value of the slope is equal to a predefined value $\bar{\beta}$:

$$H : \beta_1 = \bar{\beta}, \qquad \bar{H} : \beta_1 \neq \bar{\beta}$$

*Under the assumption of normal distribution* of $\mathbf{w}$, the following relation holds

$$\hat{\boldsymbol{\beta}}_1 \sim \mathcal{N}(\beta_1, \frac{\sigma^2}{S_{xx}}) \tag{7.1.7}$$

It follows that

$$\frac{(\hat{\boldsymbol{\beta}}_1 - \beta_1)}{\hat{\sigma}}\sqrt{S_{xx}} \sim \mathcal{T}_{N-2}$$

where $\hat{\sigma}^2$ is the estimation of the variance $\sigma^2_{\mathbf{w}}$. This is a typical t-test applied to the regression case.

Note that this statistic can be used also to test a one-sided hypothesis, e.g.

$$H : \beta_1 = \bar{\beta}, \qquad \bar{H} : \beta_1 > \bar{\beta}$$

### 7.1.6 Interval of confidence

*Under the assumption of normal distribution*, according to (7.1.7)

$$\text{Prob}\left\{-t_{\alpha/2,N-2} < \frac{(\hat{\boldsymbol{\beta}}_1 - \beta_1)}{\hat{\sigma}}\sqrt{S_{xx}} < t_{\alpha/2,N-2}\right\} = 1-\alpha$$

where $t_{\alpha/2,N-2}$ is the upper $\alpha/2$ critical point of the $\mathcal{T}$-distribution with $N-2$ degrees of freedom. Equivalently we can say that with probability $1-\alpha$, the real parameter $\beta_1$ is covered by the interval described by

$$\hat{\beta}_1 \pm t_{\alpha/2,N-2}\sqrt{\frac{\hat{\sigma}^2}{S_{xx}}}$$

Similarly from (7.1.5) we obtain that the $100(1-\alpha)\%$ confidence interval of $\beta_0$ is

$$\hat{\beta}_0 \pm t_{\alpha/2,N-2}\hat{\sigma}\sqrt{\frac{1}{N} + \frac{\bar{x}^2}{S_{xx}}}$$

### 7.1.7 Variance of the response

Let

$$\hat{\mathbf{y}} = \hat{\boldsymbol{\beta}}_0 + \hat{\boldsymbol{\beta}}_1 x$$

be the estimator of the regression function value in $x$.

If the linear dependence (7.1.1) holds, we have for a specific $x = x_0$

$$E[\hat{\mathbf{y}}|x_0] = E[\hat{\boldsymbol{\beta}}_0] + E[\hat{\boldsymbol{\beta}}_1]x_0 = \beta_0 + \beta_1 x_0 = E[\mathbf{y}|x_0]$$

This means that $\hat{\mathbf{y}}$ is an unbiased estimator of the value of the regression function in $x_0$.

*Under the assumption of normal distribution of* $\mathbf{w}$, the variance of $\hat{\mathbf{y}}$ in $x_0$

$$\text{Var}\,[\hat{\mathbf{y}}|x_0] = \sigma^2\left[\frac{1}{N} + \frac{(x_0 - \bar{x})^2}{S_{xx}}\right]$$

where $\bar{x} = \frac{\sum_{i=1}^N x_i}{N}$. This quantity measures how the prediction $\hat{\mathbf{y}}$ would vary if repeated data collection from (7.1.1) and least-squares estimations were conducted.

#### R script

Let us consider a data set $D_N = \{x_i, y_i\}_{i=1,\ldots,N}$ where

$$y_i = \beta_0 + \beta_1 x_i + w_i$$

where $\beta_0$ and $\beta_1$ are known and $\mathbf{w} \sim \mathcal{N}(0, \sigma^2)$ with $\sigma^2$ known. The R script `bv.R` may be used to:

- Study experimentally the bias and variance of the estimators $\hat{\boldsymbol{\beta}}_0$, $\hat{\boldsymbol{\beta}}_1$ and $\hat{\boldsymbol{\sigma}}$ when data are generated according to the linear dependency (7.1.2) with $\beta_0 = -1$, $\beta_1 = 1$ and $\sigma_w = 4$.

- Compare the experimental values with the theoretical results.
- Study experimentally the bias and the variance of the response prediction.
- Compare the experimental results with the theoretical ones.

```
## script bv.R
## Simulated study of bias and variance of estimators of beta_0 and beta_1 in
## univariate linear regression

rm(list=ls())
par(ask=TRUE)
X<-seq(-10,10,by=0.4)
beta0<--1  # y_i = -1 + x_i + Normal(0,4)
beta1<-1
sd.w<-4
N<-length(X)
R<-10000  # number of simulated trials

# Analytical computation and simulation of variance of estimators
beta.hat.1<-numeric(R)
beta.hat.0<-numeric(R)
var.hat.w<-numeric(R)
Y.hat<-array(NA,c(R,N))
x.hat<-mean(X)
S.xx<-sum((X-x.hat)^2)
for (r in 1:R){
  Y<-beta0+beta1*X+rnorm(N,sd=sd.w)
  y.hat<-mean(Y)
  S.xy<-sum((X-x.hat)*Y)

  beta.hat.1[r]<-S.xy/S.xx
  beta.hat.0[r]<-y.hat-beta.hat.1[r]*x.hat

  Y.hat[r,]<-beta.hat.0[r]+beta.hat.1[r]*X
  var.hat.w[r]<-sum((Y-Y.hat[r,])^2)/(N-2)
}

#  beta_1
var.beta.hat.1<-(sd.w^2)/S.xx
print(paste("Theoretical var beta1=", var.beta.hat.1, "; Observed =",
            var(beta.hat.1) ))
hist(beta.hat.1, main=paste("Distribution of beta.hat.1: beta1=", beta1))

#  beta_0
var.beta.hat.0<-(sd.w^2)*(1/N+(x.hat^2)/S.xx)
print(paste("Theoretical var beta0=", var.beta.hat.0, "; Observed =",
            var(beta.hat.0) ))
hist(beta.hat.0,main=paste("Distribution of beta.hat.0: beta0=", beta0))

#  sigma_w
hist(var.hat.w,main=paste("Distribution of var.hat.w: var w=", sd.w^2))

# Plot of results
plot(X,Y.hat[1,],type="l",
```

```
     main=paste("Variance of the prediction:",R, " repetitions"),
     ylim=c(-20,20))
for (r in 2:R)
  lines(X,Y.hat[r,])

# ================================================================
lines(X,apply(Y.hat,2,mean),col="red")

# Plotting of regression line f(x)= beta0 + beta1 x
lines(X,beta0+beta1*X,col="green")

# Study of variance of prediction
for (i in 1:N){
  var.y.hat<-var(Y.hat[,i])

  th.var.y.hat<-sd.w^2*(1/N+((X[i]-x.hat)^2)/S.xx)
  print(paste("Theoretical var predic=",th.var.y.hat, "; Observed =",
              var.y.hat ))
}
```

•

**R script**

Consider the medical dataset available in the R script `medical.R`. This script may be used to

- Estimate the intercept and slope of the linear model fitting the dataset.
- Plot the fitted linear model.
- Estimate the variance of the estimator of the slope.
- Test the hypothesis $\beta_1 = 0$.
- Compute the confidence interval of $\beta_1$.
- Compare your results with the output of the R command `lm()`.

```
## script medical.R
## Medical dataset from package  ISwR.

library(ISwR)
data(thuesen)
par(ask=TRUE)

## Data formatting

I<-! is.na(thuesen[,"short.velocity"])
Y<-thuesen[I,"short.velocity"] ### ventricular shortening velocity"
X<-thuesen[I,"blood.glucose"] ## sanguin (blood.glucose)


## Data visualization

thuesen
hist(Y)
```

```
plot(X,Y)


## Least-squares estimation and plotting of fitting line

N<-length(Y)  ## 24 patients
x.hat<-mean(X)
y.hat<-mean(Y)
S.xy<-sum((X-x.hat)*Y)
S.xx<-sum((X-x.hat)^2)
beta.hat.1<-S.xy/S.xx
beta.hat.0<-y.hat-beta.hat.1*x.hat
print(paste("beta.hat.0 = ", beta.hat.0))
print(paste("beta.hat.1 = ", beta.hat.1))
Y.hat<-beta.hat.0+beta.hat.1*X
sigma.hat = sqrt( sum( (Y-Y.hat)^2 ) / (N-2) )
print(paste("sigma.hat = ", sigma.hat))
x<-seq(min(X),max(X),by=.1)
lines(x,beta.hat.0+beta.hat.1*x)

## Test of hypothesis "beta1=0" by F-test.

SS.mod <- sum((Y.hat-mean(Y))^2)
SS.res <- sum((Y-Y.hat)^2)
F.value<-SS.mod/(SS.res/(N-2))
F.pr<-(1-pf(F.value,df1=1,df2=N-2))
print(paste(" F-test: F.value= ", F.value, "; Pr[F >= F.value]= ", F.pr))


## Test of hypothesis "beta1=0" by t-test.
var.hat.w<-sum((Y-Y.hat)^2)/(N-2)
beta.bar<-0
t.value<-(beta.hat.1-beta.bar)*sqrt(S.xx)/(sqrt(var.hat.w))
t.pr<-(1-pt(t.value,df=N-2))*2
print(paste("t-test: t.value= ",t.value, "; Pr[|T| >= t.value]= ", t.pr))


# Confidence interval computation for beta_1
alpha<-0.05
conf.interval.min<-beta.hat.1-qt(alpha/2,df=N-2,lower.tail=FALSE)*sqrt(var.hat.w/S.xx)
conf.interval.max<-beta.hat.1+qt(alpha/2,df=N-2,lower.tail=FALSE)*sqrt(var.hat.w/S.xx)
print(paste("Confidence interval beta1=(", conf.interval.min, ",", conf.interval.max, ")"))



## Test of R lm command
summary(lm(Y~X))
```

•

### 7.1.8 Coefficient of determination

The coefficient of determination, also known as $R^2$,

$$R^2 = \frac{SS_{\text{Mod}}}{SS_{\text{Tot}}} = \frac{\sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} = 1 - \frac{SS_{\text{Res}}}{SS_{\text{Tot}}}$$

is often used as a measure of the fit of the regression line.

This quantity, which satisfies the inequality $0 \leq R^2 \leq 1$, represents the proportion of variation in the response data that is explained by the model. The coefficient of determination is easy to interpret and can be understood by most experimenters regardless of their training in statistics. However, it is a dangerous criterion for comparison of candidate models because any additional model terms (e.g. a quadratic term) will decrease $SS_{\text{Res}}$ and thus increase $R^2$. In other terms $R^2$ can be made artificially high by a practice of overfitting (Section 5.5) since it is not merely the quality of fit which influences $R^2$.

### 7.1.9 Multiple linear dependence

Consider a linear relation between an independent vector $x \in \mathcal{X} \subset \mathbb{R}^n$ and a dependent random variable $\mathbf{y} \in \mathcal{Y} \subset \mathbb{R}$

$$\mathbf{y} = \beta_0 + \beta_1 x_{\cdot 1} + \beta_2 x_{\cdot 2} + \cdots + \beta_n x_{\cdot n} + \mathbf{w}$$

where $\mathbf{w}$ represents a random variable with mean zero and constant variance $\sigma^2_{\mathbf{w}}$. In matrix notation the equation can be written as:

$$\mathbf{y} = x^T\beta + \mathbf{w} \tag{7.1.8}$$

where $x$ stands for the $[p \times 1]$ vector $x = [1, x_{\cdot 1}, x_{\cdot 2}, \ldots, x_{\cdot n}]^T$ and $p = n + 1$ is the total number of model parameters.

Henceforth, in order to avoid misunderstanding, $x_{\cdot i}$ will denote the $i$th variable of the vector $x$, while $x_i = [1, x_{i1}, x_{i2}, \ldots, x_{in}]^T$ will denote the $i$th observation of the vector $x$.

### 7.1.10 The multiple linear regression model

Consider $N$ observations $D_N = \{\langle x_i, y_i\rangle : i = 1, \ldots, N\}$ generated according to the stochastic dependence (7.1.8) where $x_i = [1, x_{i1}, \ldots, x_{in}]^T$. We suppose that the following multiple linear relation holds

$$Y = X\beta + W$$

where $Y$ is the $[N \times 1]$ response vector, $X$ is the $[N \times p]$ *data matrix*, whose $j^{\text{th}}$ column of $X$ contains readings on the $j^{\text{th}}$ regressor, $\beta$ is the $[p \times 1]$ vector of parameters

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1n} \\ 1 & x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{N1} & x_{N2} & \ldots & x_{Nn} \end{bmatrix} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} \quad W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}$$

Here $w_i$ are assumed uncorrelated, with mean zero and constant variance $\sigma^2_{\mathbf{w}}$ (homogeneous variance). Then $\text{Var}[\mathbf{w}_1, \ldots, \mathbf{w}_N] = \sigma^2_{\mathbf{w}} I_N$.

### 7.1.11 The least-squares solution

We seek the **the least-squares estimator $\hat{\boldsymbol{\beta}}$** such that

$$\hat{\beta} = \arg\min_b \sum_{i=1}^{N} (y_i - x_i^T b)^2 = \arg\min_b \left((Y - Xb)^T (Y - Xb)\right)$$

where we call the quantity

$$\text{SSE}_{\text{emp}} = N \cdot \widehat{\text{MISE}}_{\text{emp}} = \left((Y - Xb)^T (Y - Xb)\right) = e^T e \tag{7.1.9}$$

the *residual sum of squares* (or $N$ times the empirical risk) and

$$e = Y - Xb$$

the $[N \times 1]$ vector of residuals. In order to minimize the residual sum of squares the vector $\hat{\beta}$ must satisfy

$$\frac{\partial}{\partial \hat{\beta}} [(Y - X\hat{\beta})^T (Y - X\hat{\beta})] = 0 \tag{7.1.10}$$

Differentiating the residual sum of squares we obtain the *least-squares normal equations*

$$(X^T X)\hat{\beta} = X^T Y$$

As a result, assuming $X$ is of full column rank

$$\hat{\beta} = (X^T X)^{-1} X^T Y \tag{7.1.11}$$

where the $X^T X$ matrix is a symmetric $[p \times p]$ matrix which plays an important role in multiple linear regression. Note that the computation of $\hat{\beta}$ represents the parametric identification step (Section 5.6) when the class of hypothesis is linear.

During the presentation we assumed that the variables $x_{\cdot i}$ are non random and observed with a negligible error. Of course, in most practical situations the variables $x_{\cdot i}$ do experience some random variation. We assume that here any random variation of $x$ is negligible compared to the range in which it is measured. In addition any error in the measurement of $x$ is assumed to be small compared to the range.

Under the condition that the stochastic dependence (7.1.8) holds it can be shown [89] that:

- If $E[\mathbf{w}] = 0$ then $\hat{\boldsymbol{\beta}}$ is an unbiased estimator of $\beta$.
- The *residual mean square* estimator

  $$\hat{\boldsymbol{\sigma}}^2 = \frac{(Y - X\hat{\boldsymbol{\beta}})^T (Y - X\hat{\boldsymbol{\beta}})}{N - p}$$

  is an unbiased estimator of the error variance $\sigma^2_{\mathbf{w}}$.
- If the $\mathbf{w}_i$ are uncorrelated and have common variance, the variance-covariance matrix of $\hat{\boldsymbol{\beta}}$ is given by

  $$\text{Var}[\hat{\boldsymbol{\beta}}] = \sigma^2_{\mathbf{w}} (X^T X)^{-1}$$

**R script**

The R script `bv_mult.R` may be used to validate by simulation the above-mentioned properties of a least-squares estimator.

```
## script bv_mult.R
## Simulated study of bias and variance of estimators of beta and sigma_w in
## multivariate linear regression
rm(list=ls())
library(MASS)
par(ask=TRUE)

n<-3 # number input variables
p<-n+1
N<-100 # number training data
X<-array(runif(N*n,min=-20,max=20),c(N,n))
X<-cbind(array(1,c(N,1)),X)
beta<-seq(2,p+1)

R<-10000
sd.w<-5


beta.hat<-array(0,c(p,R))
var.hat.w<-numeric(R)
Y.hat<-array(NA,c(R,N))
for (r in 1:R){
  Y<-X%*%beta+rnorm(N,sd=sd.w)


  beta.hat[,r]<-ginv(t(X)%*%X)%*%t(X)%*%Y


  Y.hat[r,]<-X%*%beta.hat[,r]
  e<-Y-Y.hat[r,]
  var.hat.w[r]<-(t(e)%*%e)/(N-p)
}

hist(var.hat.w,main=paste("Distribution of var.hat.w: var w=", sd.w^2))
for (i in 1:p){
  hist(beta.hat[i,], main=paste("Distribution of beta.hat.",i,": beta",i,"=", beta[i]))
}


# graphical test unbiasedness prediction
plot(apply(Y.hat,2,mean),X%*%beta,ylab="Regression function",xlab="Expected prediction")
# test unbiasedness prediction
print(paste("Bias prediction", apply(Y.hat,2,mean)-X%*%beta))

# comparison analytical and simulated variance of the prediction
for (i in 1:N){
  print(var(Y.hat[,i])-sd.w^2*(t(X[i,])%*%ginv(t(X)%*%X)%*%X[i,]))

}
```

•

### 7.1.12 Variance of the prediction

The variance of the prediction $\hat{\mathbf{y}}$ for a generic input value $x = x_0$ is given by

$$\text{Var}\,[\hat{\mathbf{y}}|x_0] = \sigma^2_{\mathbf{w}} x_0^T (X^T X)^{-1} x_0 \tag{7.1.12}$$

Assuming that $\mathbf{w}$ is normally distributed, the $100(1-\alpha)\%$ confidence bound for the regression value in $x_0$ is given by

$$\hat{y}(x_0) \pm t_{\alpha/2,N-p}\hat{\sigma}\sqrt{x_0^T (X^T X)^{-1} x_0}$$

where $t_{\alpha/2,N-p}$ is the upper $\alpha/2$ percent point of the t-distribution with $N-p$ degrees of freedom and the quantity $\hat{\sigma}\sqrt{x_0^T (X^T X)^{-1} x_0}$, obtained from (7.1.12), is the *standard error of prediction* for multiple regression.

### 7.1.13 The HAT matrix

The Hat matrix is defined as

$$H = X(X^T X)^{-1} X^T$$

It is a symmetric, idempotent $[N \times N]$ matrix that transforms the output values $Y$ of the training set in the regression predictions $\hat{Y}$:

$$\hat{Y} = X\hat{\beta} = X(X^T X)^{-1} X^T Y = HY$$

Using the above relation, the vector of residuals can be written as:

$$e = Y - X\hat{\beta} = Y - X(X^T X)^{-1} X^T Y = [I - H]Y$$

and the residual sum of squares as

$$e^T e = Y^T [I - H]^2 Y = Y^T [I - H]Y = Y^T PY \tag{7.1.13}$$

where $P$ is a $[N \times N]$ matrix, called the *projection* matrix.

### 7.1.14 Generalization error of the linear model

The linear predictor

$$\hat{y} = x^T \hat{\beta}$$

has been estimated by using the training dataset $D_N = \{\langle x_i, y_i\rangle : i = 1, \ldots, N\}$. Then the estimator $\hat{\boldsymbol{\beta}}$ is a r.v..

Given an input $x_i$ and the corresponding output $\mathbf{y}(x_i)$, let us consider the least-squares prediction $\hat{y}_i = x_i^T \hat{\beta}$. Which precision can we expect from $\hat{\mathbf{y}}(x_i) = x_i^T \hat{\boldsymbol{\beta}}$ on average?

A quantitative measure of the quality of the linear predictor on the whole domain $\mathcal{X}$ is the Mean Integrated Squared Error (MISE) defined in 5.2.9. But how can we estimate this quantity? Also, is the empirical risk (apparent error) a good estimate of MISE?

### 7.1.15 The expected empirical error

This section shows that the empirical risk $\widehat{\text{MISE}}_{\text{emp}}$ defined in (7.1.9) is a bad estimate of the MISE generalization error.

First we compute the expectation of the residual sum of squares which is equal to $N$ times the empirical risk. According to (7.1.13) and Theorem 4.1 the expectation can be written as

$$E_{\mathbf{D}_N}[\text{SSE}_{\text{emp}}] = E_{\mathbf{D}_N}[\mathbf{e}^T\mathbf{e}] = E_{\mathbf{D}_N}[\mathbf{Y}^T P \mathbf{Y}] = \sigma^2_{\mathbf{w}}\text{tr}(P) + E[\mathbf{Y}^T]\ P\ E[\mathbf{Y}]$$

Since $\text{tr}(ABC) = \text{tr}(CAB)$

$$\begin{aligned}\text{tr}(P) &= \text{tr}(I - H)\\ &= N - \text{tr}(X(X^TX)^{-1}X^T)\\ &= N - \text{tr}(X^TX(X^TX)^{-1})\\ &= N - \text{tr}(I_p) = N - p\end{aligned}$$

we have

$$\begin{aligned}E[\mathbf{e}^T\mathbf{e}] &= (N-p)\sigma^2_{\mathbf{w}} + (X\beta)^T P (X\beta)\\ &= (N-p)\sigma^2_{\mathbf{w}} + \beta^T X^T (I - X(X^TX)^{-1}X^T)X\beta\\ &= (N-p)\sigma^2_{\mathbf{w}}\end{aligned}$$

It follows that

$$E[\widehat{\text{MISE}}_{\text{emp}}] = E\left[\frac{\text{SSE}_{\text{emp}}}{N}\right] = (1 - p/N)\sigma^2_{\mathbf{w}}$$

is the expectation of the error made by a linear model trained on $D_N$ to predict the value of the output in the same dataset $D_N$.

In order to obtain the MISE term we derive analytically the expected sum of squared errors of a linear model trained on $D_N$ and used to predict for the same training inputs $X$ a set of outputs $Y_{ts}$ distributed according to the same linear law (7.1.1) but independent of the training output $Y$.

$$\begin{aligned}&E_{\mathbf{D}_N,\mathbf{Y}_{ts}}[(\mathbf{Y}_{ts} - X\hat{\boldsymbol{\beta}})^T(\mathbf{Y}_{ts} - X\hat{\boldsymbol{\beta}})]\\ &= E_{\mathbf{D}_N,\mathbf{Y}_{ts}}[(\mathbf{Y}_{ts} - X\beta + X\beta - X\hat{\boldsymbol{\beta}})^T(\mathbf{Y}_{ts} - X\beta + X\beta - X\hat{\boldsymbol{\beta}})]\\ &= E_{\mathbf{D}_N,\mathbf{Y}_{ts}}[(\mathbf{W}_{ts} + X\beta - X\hat{\boldsymbol{\beta}})^T(\mathbf{W}_{ts} + X\beta - X\hat{\boldsymbol{\beta}})]\\ &= N\sigma^2_{\mathbf{w}} + E_{\mathbf{D}_N}[(X\beta - X\hat{\boldsymbol{\beta}})^T(X\beta - X\hat{\boldsymbol{\beta}})]\end{aligned}$$

Since

$$\begin{aligned}X\beta - X\hat{\beta} &= X\beta - X(X^TX)^{-1}X^TY\\ &= X\beta - X(X^TX)^{-1}X^T(X\beta + W)\\ &= -X(X^TX)^{-1}X^TW\end{aligned}$$

we obtain

$$\begin{aligned}&N\sigma^2_{\mathbf{w}} + E_{\mathbf{D}_N}[(X\beta - X\hat{\boldsymbol{\beta}})^T(X\beta - X\hat{\boldsymbol{\beta}})]\\ &= N\sigma^2_{\mathbf{w}} + E_{\mathbf{D}_N}[(\mathbf{W}^TX(X^TX)^{-1}X^T)(X(X^TX)^{-1}X^T\mathbf{W})]\\ &= N\sigma^2_{\mathbf{w}} + E_{\mathbf{D}_N}[\mathbf{W}^TX(X^TX)^{-1}X^T\mathbf{W}]\\ &= N\sigma^2_{\mathbf{w}} + \sigma^2_{\mathbf{w}}\text{tr}(X(X^TX)^{-1}X^T)\\ &= N\sigma^2_{\mathbf{w}} + \sigma^2_{\mathbf{w}}\text{tr}(X^TX(X^TX)^{-1})\\ &= N\sigma^2_{\mathbf{w}} + \sigma^2_{\mathbf{w}}\text{tr}(I_p) = \sigma^2_{\mathbf{w}}(N+p)\end{aligned}$$

By dividing the above quantity by $N$ we obtain that

$$\text{MISE} = (1 + p/N)\sigma^2_{\mathbf{w}}$$

This means that that the empirical error $\widehat{\text{MISE}}_{\text{emp}}$ is a biased estimate of MISE, that is

$$E_{\mathbf{D}_N}[\widehat{\text{MISE}}_{\text{emp}}] = E_{\mathbf{D}_N}\left[\frac{\mathbf{e}^T\mathbf{e}}{N}\right] \neq \text{MISE}$$

As a consequence, if we replace $\widehat{\text{MISE}}_{\text{emp}}$ with

$$\frac{\mathbf{e}^T\mathbf{e}}{N} + 2\sigma^2_{\mathbf{w}}p/N \tag{7.1.14}$$

we obtain an unbiased estimator of the generalization error measured by the MISE. (see R file `ee.R`). Nevertheless, this estimator requires an estimate of the noise variance.

**Example**

Let $\{y_1, \ldots, y_N\} \leftarrow F_{\mathbf{y}}$ be the training set. Consider the simplest linear predictor of the output variable: the average $\hat{\boldsymbol{\mu}}_y$ (i.e. $p = 1$). This means that

$$\hat{y}_i = \frac{\sum_{i=1}^N y_i}{N} = \hat{\mu}_y, \qquad i = 1, \ldots, N$$

We want to show that, even for this simple estimator, the empirical error is a biased estimator of the quality of this predictor. Let $\mu$ be the mean of the r.v. $\mathbf{y}$. Let us write $\mathbf{y}$ as

$$\mathbf{y} = \mu + \mathbf{w}$$

where $E[\mathbf{w}] = 0$ and $\text{Var}[\mathbf{w}] = \sigma^2$. Let $\{z_1, \ldots, z_N\} \leftarrow F_{\mathbf{y}}$ a test set coming from the same distribution underlying $D_N$. Let us compute the expected empirical error and the mean integrated square error.

Since $E[\hat{\boldsymbol{\mu}}_y] = \mu$ and $\text{Var}\left[\hat{\boldsymbol{\mu}}_y\right] = \sigma^2/N$

$$\begin{aligned} N \cdot \text{MISE} = E_{\mathbf{D}_N, \mathbf{Y}_{ts}}[\sum_{i=1}^N (z_i - \hat{\boldsymbol{\mu}}_y)^2] &= E_{\mathbf{D}_N, \mathbf{w}}[\sum_{i=1}^N (\mu + \mathbf{w}_i - \hat{\boldsymbol{\mu}}_y)^2] \\ &= N\sigma^2 + \sum_{i=1}^N E_{\mathbf{D}_N}[(\mu - \hat{\boldsymbol{\mu}}_y)^2] \\ &= N\sigma^2 + N(\sigma^2/N) = (N+1)\sigma^2 \end{aligned}$$

Instead, since $\hat{\sigma}_y = (\sum_{i=1}^N (y_i - \mu_y)^2)/(N-1)$ and $E[\hat{\boldsymbol{\sigma}}^2_y] = \sigma^2$

$$E_{\mathbf{D}_N}[\sum_{i=1}^N (y_i - \hat{\boldsymbol{\mu}}_y)^2] = E_{\mathbf{D}_N}[(N-1)\hat{\boldsymbol{\sigma}}^2_y] = (N-1)\sigma^2 \neq N \cdot \text{MISE}$$

It follows that the empirical error is a biased estimate of the accuracy of the average predictor (see R file `ee_mean.R`).

•

### 7.1.16 The PSE and the FPE

In the previous section we derived that $\widehat{\text{MISE}}_{\text{emp}}$ is a biased estimate of MISE and that the addition of the correction term $2\sigma^2_{\mathbf{w}}p/N$ makes it unbiased.

Suppose we have an estimate $\hat{\sigma}^2_{\mathbf{w}}$ of $\sigma_{\mathbf{w}}$. By replacing it into the expression (7.1.14) we obtain the so-called *Predicted Square Error (PSE)* criterion

$$\text{PSE} = \widehat{\text{MISE}}_{\text{emp}} + 2\hat{\sigma}^2_{\mathbf{w}}p/N$$

In particular, if we take as estimate of $\sigma^2_{\mathbf{w}}$ the quantity

$$\hat{\sigma}^2_{\mathbf{w}} = \frac{1}{N-p}\text{SSE}_{\text{emp}} = \frac{N}{N-p}\widehat{\text{MISE}}_{\text{emp}} \tag{7.1.15}$$

we obtain the so-called *Final Prediction Error (FPE)*

$$\text{FPE} = \frac{1+p/N}{1-p/N}\widehat{\text{MISE}}_{\text{emp}}$$

The PSE and the FPE criteria allows to replace the empirical risk with a more accurate estimate of the generalization error of a linear model. Although their expression is easy to compute, it is worthy to remind that their derivation relies on the assumption that the stochastic input/output dependence has the linear form 7.1.8.

#### R script

Let us consider an input/output dependence

$$\mathbf{y} = f(x) + \mathbf{w} = 1 + x + x^2 + x^3 + \mathbf{w} \tag{7.1.16}$$

where $\mathbf{w} \sim \mathcal{N}(0,1)$ and $\mathbf{x} \sim \mathcal{U}(-1,1)$. Suppose that a dataset $D_N$ of $N = 100$ input/output observations is drawn from the joint distribution of $\langle \mathbf{x}, \mathbf{y} \rangle$. The R script `fpe.R` assesses the prediction accuracy of 7 different models having the form

$$h_m(x) = \hat{\beta}_0 + \sum_{j=1}^{m} \hat{\beta}_j x^j \tag{7.1.17}$$

by using the empirical risk and the FPE measure. These results are compared with the generalization error estimated by

$$\text{MISE}_m = \frac{1}{N}\sum_{i=1}^{N}(h_m(x_i) - f(x_i))^2 \tag{7.1.18}$$

The empirical risk and the FPE values for $m = 2, \ldots, 7$ are plotted in Figure 7.2 and 7.3, respectively. The values $\text{MISE}_m$ are plotted in Figure 7.4. It is evident, as confirmed by Figure 7.4, that the best model should be $h_3(x)$ since it has the same analytical structure as $f(x)$. However, the empirical risk is not able to detect this and returns as the best model the one with the highest complexity ($m = 7$). This is not the case for FPE which, by properly correcting the $\widehat{\text{MISE}}_{\text{emp}}$ value, is able to select the optimal model.

•

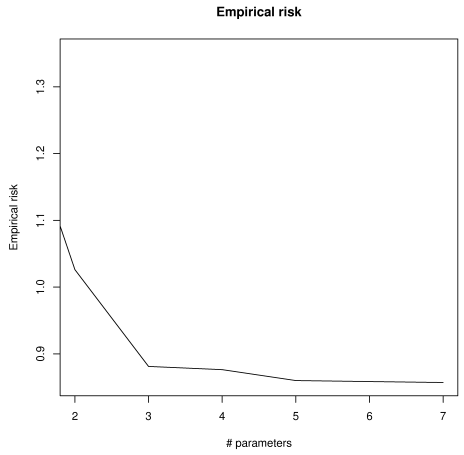



Figure 7.2: Estimation of the generalization accuracy of $h_m$, $m = 2, \ldots, 7$ returned by the empirical error.

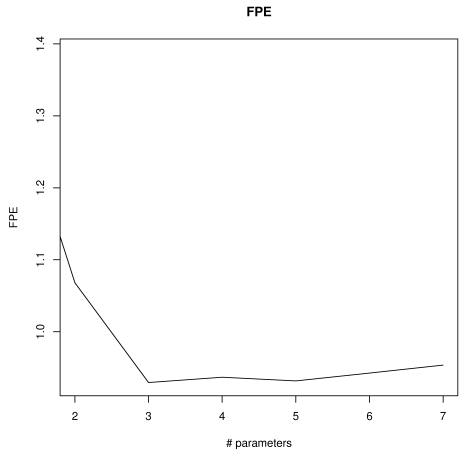



Figure 7.3: Estimation of the generalization accuracy of $h_m$, $m = 2, \ldots, 7$ returned by the FPE.

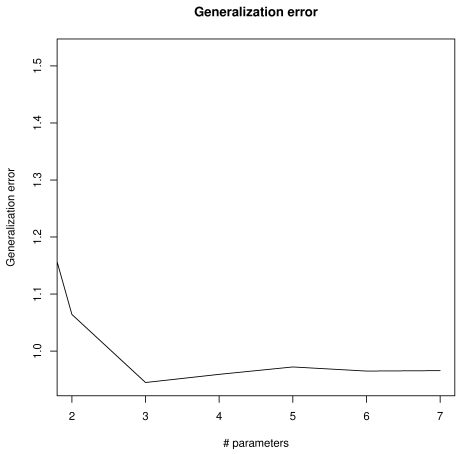



Figure 7.4: Estimation of the generalization accuracy of $h_m$, $m = 2, \ldots, 7$ returned by (7.1.18).

## 7.2 The PRESS statistic

We mentioned in Section 5.7.1 that cross-validation can provide a reliable estimate of the algorithm generalization error $G_N$. The disadvantage of such an approach is that it requires the training process to be repeated $l$ times, which sometimes means a large computational effort. However, in the case of linear models there exists a powerful statistical procedure to compute the leave-one-out cross-validation measure at a reduced computational cost (Fig. 7.5). It is the PRESS (Prediction Sum of Squares) statistic [5], a simple formula which returns the leave-one-out (l-o-o) as a by-product of the parametric identification of $\hat{\beta}$ in Eq. (8.1.39). Consider a training set $D_N$ in which for $N$ times

1. we set aside the $j^{\text{th}}$ observation $\langle x_j, y_j \rangle$ from the training set,
2. we use the remaining $N - 1$ observations to estimate the linear regression coefficients $\hat{\beta}^{-j}$,
3. we use $\hat{\beta}^{-j}$ to predict the target in $x_j$.

The leave-one-out residual is

$$e_j^{\text{loo}} = y_j - \hat{y}_j^{-j} = y_j - x_j^T \hat{\beta}^{-j} \tag{7.2.19}$$

The PRESS statistic is an efficient way to compute the l-o-o residuals on the basis of the simple regression performed on the whole training set. This allows a fast cross-validation without repeating $N$ times the leave-one-out procedure. The PRESS procedure can be described as follows:

1. we use the whole training set to estimate the linear regression coefficients *ebe*. This procedure is performed only once on the $N$ samples and returns as by product the Hat matrix (see Section 7.1.13)

$$H = X(X^T X)^{-1} X^T \tag{7.2.20}$$

2. we compute the residual vector $e$, whose $j^{\text{th}}$ term is $e_j = y_j - x_j^T \hat{\beta}$,

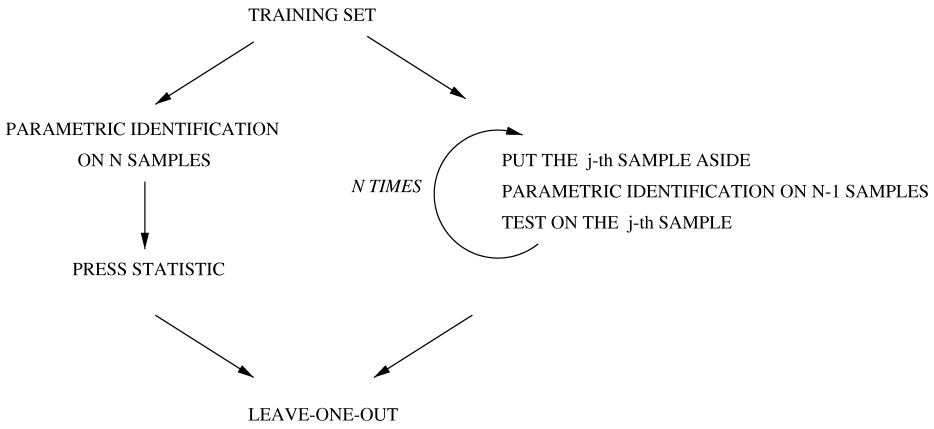



Figure 7.5: Leave-one-out for linear models. The leave-one-out error can be computed in two equivalent ways: the slowest way (on the right) which repeats $N$ times the training and the test procedure; the fastest way (on the left) which performs only once the parametric identification and the computation of the PRESS statistic.

3. we use the PRESS statistic to compute $e_j^{\text{loo}}$ as

$$e_j^{\text{loo}} = \frac{e_j}{1 - H_{jj}} \tag{7.2.21}$$

where $H_{jj}$ is the $j^{\text{th}}$ diagonal term of the matrix $H$.

Note that (7.2.21) is not an approximation of (7.2.19) but simply a faster way of computing the leave-one-out residual $e_j^{\text{loo}}$.

Let us now derive the formula of the PRESS statistic.

Matrix manipulations show that

$$X^T X - x_j x_j^T = X_{-j}^T X_{-j} \tag{7.2.22}$$

where $X_{-j}^T X_{-j}$ is the $X^T X$ matrix obtained by putting the $j^{\text{th}}$ row aside.

Using the relation (B.5.9) we have

$$(X_{-j}^T X_{-j})^{-1} = (X^T X - x_j x_j^T)^{-1} = (X^T X)^{-1} + \frac{(X^T X)^{-1} x_j x_j^T (X^T X)^{-1}}{1 - H_{jj}} \tag{7.2.23}$$

and

$$\hat{\beta}^{-j} = (X_{-j}^T X_{-j})^{-1} X'_{-j} y_{-j} = \left[ (X^T X)^{-1} + \frac{(X^T X)^{-1} x_j x_j^T (X^T X)^{-1}}{1 - H_{jj}} \right] X_{-j}^T y_{-j} \tag{7.2.24}$$

where $y_{-j}$ is the target vector with the $j^{\text{th}}$ sample set aside.

From (7.2.19) and (7.2.24) we have

$$\begin{aligned} e_j^{\text{loo}} &= y_j - x_j^T \hat{\beta}^{-j} \\ &= y_j - x_j^T \left[ (X^T X)^{-1} + \frac{(X^T X)^{-1} x_j x_j^T (X^T X)^{-1}}{1 - H_{jj}} \right] X_{-j}^T y_{-j} \\ &= y_j - x_j^T (X^T X)^{-1} X_{-j}^T y_{-j} - \frac{H_{jj} x_j^T (X^T X)^{-1} X_{-j}^T y_{-j}}{1 - H_{jj}} \\ &= \frac{(1 - H_{jj}) y_j - x_j^T (X^T X)^{-1} X_{-j}^T y_{-j}}{1 - H_{jj}} \\ &= \frac{(1 - H_{jj}) y_j - x_j^T (X^T X)^{-1} (X^T y - x_j y_j)}{1 - H_{jj}} \\ &= \frac{(1 - H_{jj}) y_j - \hat{y}_j + H_{jj} y_j}{1 - H_{jj}} \\ &= \frac{y_j - \hat{y}_j}{1 - H_{jj}} = \frac{e_j}{1 - H_{jj}} \end{aligned} \tag{7.2.25}$$

where $X_{-j}^T y_{-j} + x_j y_j = X^T y$ and $x_j^T (X^T X)^{-1} X^T y = \hat{y}_j$.

Thus, the leave-one-out estimate of the local mean integrated squared error is:

$$\hat{G}_{\text{loo}} = \frac{1}{N} \sum_{i=1}^{N} \left\{ \frac{y_i - \hat{y}_i}{1 - H_{ii}} \right\}^2 \tag{7.2.26}$$

## 7.3 The weighted least-squares

The assumption of homogeneous error variance made in Eq. (7.1.8) is often violated in practical situations. Suppose we relax the assumption that $\text{Var}(\mathbf{w}) = \sigma_{\mathbf{w}}^2 I_N$ with $I_N$ identity matrix and assume instead that there is a positive definite matrix $V$ for which $\text{Var}(\mathbf{w}) = V$. We may wish to consider

$$V = \text{diag}[\sigma_1^2, \sigma_2^2, \ldots, \sigma_N^2] \tag{7.3.27}$$

in which case we are assuming uncorrelated errors with error variances that vary from observation to observation. As a result it would seem reasonable that the estimator of $\beta$ should take this into account by weighting the observations in some way that allows for the differences in the precision of the results. Then the function being minimized is no more (7.1.9) but depends on $V$ and is given by

$$(y - X\hat{\beta})^T V^{-1} (y - X\hat{\beta}) \tag{7.3.28}$$

The estimate of $\beta$ is then

$$\hat{\beta} = (X^T V^{-1} X)^{-1} X^T V^{-1} y \tag{7.3.29}$$

The corresponding estimator is called the *generalized least-squares estimator* and has the following properties:

- It is unbiased, that is $E[\hat{\boldsymbol{\beta}}] = \beta$
- Under the assumption $\mathbf{w} \sim N(0, V)$ it is the minimum variance estimator among all the unbiased estimators.

### 7.3.1 Recursive least-squares

In many data analysis tasks data are not available all at once but arrive sequentially. Think for example to data collected from sensors, financial time series or adaptive control applications. In this cases it is useful not to restart from scratch the model estimation but simply to update the model on the basis of the newly collected data. One appealing feature of least-squares estimates is that they can be updated at a lower cost than their batch counterpart.

Let us rewrite the least-squares estimator (7.1.11) for a training set of $N$ samples as:

$$\hat{\beta}_{(N)} = (X_{(N)}^T X_{(N)})^{-1} X_{(N)}^T Y_{(N)}$$

where the subscript $(N)$ is added to denote the number of samples used for the estimation. Suppose that a new sample $\langle x_{N+1}, y_{N+1} \rangle$ becomes available. Instead of recomputing the estimate $\hat{\beta}_{(N+1)}$ by using all the $N+1$ available data we want to derive $\hat{\beta}_{(N+1)}$ as an update of $\hat{\beta}_{(N)}$. This problem is solved by the so-called *recursive least-squares (RLS) estimation.*

If a single new example $\langle x_{N+1}, y_{N+1} \rangle$, with $x_{N+1}$ a $[1, p]$ vector, is added to the training set the $X$ matrix acquires a new row and $\hat{\beta}_{(N+1)}$ can be written as:

$$\hat{\beta}_{(N+1)} = \left( \begin{bmatrix} X_{(N)} \\ x_{N+1} \end{bmatrix}^T \begin{bmatrix} X_{(N)} \\ x_{N+1} \end{bmatrix} \right)^{-1} \begin{bmatrix} X_{(N)} \\ x_{N+1} \end{bmatrix}^T \begin{bmatrix} Y_{(N)} \\ y_{N+1} \end{bmatrix}$$

By defining the $[p, p]$ matrix

$$S_{(N)} = (X_{(N)}^T X_{(N)})$$

we have

$$\begin{aligned} S_{(N+1)} = (X_{(N+1)}^T X_{(N+1)}) &= \left( \begin{bmatrix} X_{(N)}^T x_{N+1}^T \end{bmatrix} \begin{bmatrix} X_{(N)} \\ x_{N+1} \end{bmatrix} \right) \\ &= \left( X_{(N)}^T \; X_{(N)} + x_{N+1}^T x_{N+1} \right) = S_{(N)} + x_{N+1}^T x_{N+1} \end{aligned} \tag{7.3.30}$$

Since

$$\begin{bmatrix} X_{(N)} \\ x_{N+1} \end{bmatrix}^T \begin{bmatrix} Y_{(N)} \\ y_{N+1} \end{bmatrix} = X_{(N)}^T Y_{(N)} + x_{N+1}^T y_{N+1}$$

and

$$S_{(N)} \hat{\beta}_{(N)} = (X_{(N)}^T X_{(N)}) \left[ (X_{(N)}^T X_{(N)})^{-1} X_{(N)}^T Y_{(N)} \right] = X_{(N)}^T Y_{(N)}$$

we obtain

$$\begin{aligned} S_{(N+1)} \hat{\beta}_{(N+1)} = \begin{bmatrix} X_{(N)} \\ x_{N+1} \end{bmatrix}^T \begin{bmatrix} Y_{(N)} \\ y_{N+1} \end{bmatrix} &= S_{(N)} \hat{\beta}_{(N)} + x_{N+1}^T y_{N+1} \\ &= \left( S_{(N+1)} - x_{N+1}^T x_{N+1} \right) \hat{\beta}_{(N)} + x_{N+1}^T y_{N+1} \\ &= S_{(N+1)} \hat{\beta}_{(N)} - x_{N+1}^T x_{N+1} \hat{\beta}_{(N)} + x_{N+1}^T y_{N+1} \end{aligned}$$

or equivalently

$$\hat{\beta}_{(N+1)} = \hat{\beta}_{(N)} + S_{(N+1)}^{-1} x_{N+1}^T (y_{N+1} - x_{N+1} \hat{\beta}_{(N)}) \tag{7.3.31}$$

#### 7.3.1.1 1st Recursive formulation

From (7.3.30) and (7.3.31) we obtain the following recursive formulation

$$\begin{cases} S_{(N+1)} & = S_{(N)} + x_{N+1}^T x_{N+1} \\ \gamma_{(N+1)} & = S_{(N+1)}^{-1} x_{N+1}^T \\ e & = y_{N+1} - x_{N+1}\hat{\beta}_{(N)} \\ \hat{\beta}_{(N+1)} & = \hat{\beta}_{(N)} + \gamma_{(N+1)} e \end{cases}$$

where the term $\hat{\beta}_{(N+1)}$ can be expressed as a function of the old estimate $\hat{\beta}_{(N)}$ and the new sample $\langle x_{N+1}, y_{N+1}\rangle$. This formulation requires the inversion of the $[p \times p]$ matrix $S_{(N+1)}$.This operation is computationally expensive but, fortunately, using a matrix inversion theorem, an incremental formula for $S^{-1}$ can be found.

#### 7.3.1.2 2nd Recursive formulation

Once defined

$$V_{(N)} = S_{(N)}^{-1} = (X_{(N)}^T X_{(N)})^{-1}$$

we have $(S_{(N+1)})^{-1} = (S_{(N)} + x_{N+1}^T x_{N+1})^{-1}$ and

$$V_{(N+1)} = V_{(N)} - V_{(N)} x_{(N+1)}^T (I + x_{N+1} V_{(N)} x_{N+1}^T)^{-1} x_{N+1} V_{(N)} \tag{7.3.32}$$

$$= V_{(N)} - \frac{V_{(N)} x_{N+1}^T x_{N+1} V_{(N)}}{1 + x_{N+1} V_{(N)} x_{N+1}^T} \tag{7.3.33}$$

From (7.3.32) and (7.3.31) we obtain a second recursive formulation:

$$\begin{cases} V_{(N+1)} & = V_{(N)} - \frac{V_{(N)} x_{N+1}^T x_{N+1} V_{(N)}}{1 + x_{N+1} V_{(N)} x_{N+1}^T} \\ \gamma_{(N+1)} & = V_{(N+1)} x_{N+1}^T \\ e & = y_{N+1} - x_{N+1}\hat{\beta}_{(N)} \\ \hat{\beta}_{(N+1)} & = \hat{\beta}_{(N)} + \gamma_{(N+1)} e \end{cases} \tag{7.3.34}$$

#### 7.3.1.3 RLS initialization

Both recursive formulations presented above require the initialization values $\hat{\beta}_{(0)}$ and $V_{(0)}$. One way to avoid choosing these initial values is to collect the first $N$ data points, to solve $\hat{\beta}_{(N)}$ and $V_{(N)}$ directly from

$$V_{(N)} = (X_{(N)}^T X_{(N)})^{-1}$$
$$\hat{\beta}_{(N)} = V_{(N)} X_{(N)}^T Y_{(N)}$$

and to start iterating from the $N + 1^{\text{th}}$ point. Otherwise, in case of a generic initialization $\hat{\beta}_{(0)}$ and $V_{(0)}$ we have the following relations

$$V_{(N)} = (V_{(0)} + X_{(N)}^T X_{(N)})^{-1}$$
$$\hat{\beta}_{(N)} = V_{(N)}(X_{(N)}^T Y_{(N)} + V_{(0)}^{-1}\hat{\beta}_{(0)})$$

A common choice is to put

$$V_{(0)} = aI, \qquad a > 0$$

Since $V_{(0)}$ represents the variance of the estimator to choose a very large $a$ is equivalent to consider the initial estimation of $\beta$ as very uncertain. By setting $a$ equal to a large number the RLS algorithm will diverge very rapidly from the initialization $\hat{\beta}_{(0)}$. Therefore, we can force the RLS variance and parameters to be arbitrarily close to the ordinary least-squares values, regardless of $\hat{\beta}_{(0)}$.

In any case, in absence of further information, the initial value $\hat{\beta}_{(0)}$ is usually put equal to a zero vector.

#### 7.3.1.4 RLS with forgetting factor

In some adaptive configurations it can be useful not to give equal importance to all the historical data but to assign higher weights to the most recent data (and then to forget the oldest one). This may happen when the phenomenon underlying the data is non stationary or when we want to approximate a nonlinear dependence by using a linear model which is *local in time*. Both these situations are common in adaptive control problems.

RLS techniques can deal with these situations by a modification of the formulation (7.3.34) obtained by adding a forgetting factor $\mu < 1$.

$$\begin{cases} V_{(N+1)} & = \frac{1}{\mu}\left(V_{(N)} - \frac{V_{(N)}x_{N+1}^T x_{N+1} V_{(N)}}{1+x_{N+1}V_{(N)}x_{N+1}^T}\right) \\ \gamma_{(N+1)} & = V_{(N+1)}x_{N+1}^T \\ e & = y_{N+1} - x_{N+1}\hat{\beta}_{(N)} \\ \hat{\beta}_{(N+1)} & = \hat{\beta}_{(N)} + \gamma_{(N+1)}e \end{cases}$$

Note that: (i) the smaller $\mu$, the higher the forgetting, (ii) for $\mu = 1$ we have the conventional RLS formulation.

#### R script

The R script `lin_rls.R` considers the RLS fitting of a nonlinear univariate function. The simulation shows how the fitting evolves as long as data $x_i, y_i$, $i = 1, \ldots, N$ are collected. Note that the sequence of $x_i$, $i = 1, \ldots, N$ is increasing. This means that $\mathbf{x}$ is not random and that the oldest collected values are the ones with the lowest $x_i$.

The final fitting for a forgetting factor $\mu = 0.9$ is shown in Figure 7.6. Note that the linear fitting concerns only the rightmost samples since the values on the left, which are also the oldest ones, are forgotten.

```
## script lin_rls.R
##
rm(list=ls())
par(ask=TRUE)
n<-1;

rls<-function(x,y,t,P,mu=1){

P.new <-(P-(P%*%x%*%x%*%P)/as.numeric(1+x%*%P%*%x))/mu
ga <- P.new%*%x
epsi <- y-x%*%t
t.new<-t+ga*as.numeric(epsi)
list(t.new,P.new)
}
```

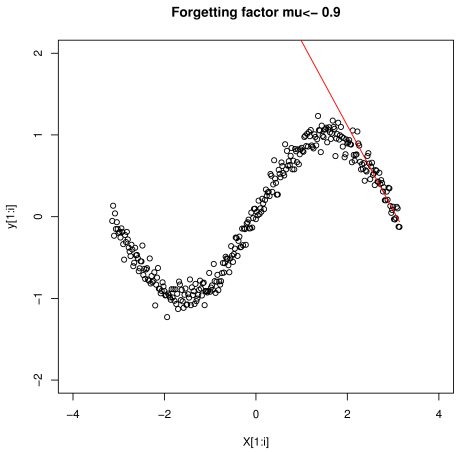



Figure 7.6: RLS fitting of a nonlinear function where the arrival order of data is from left to right

```
X<-seq(-pi,pi,by=.02)
N<-length(X)

y<-sin(X)+0.1*rnorm(N)
t<-numeric(2)
P<-500*diag(n+1)
mu<-0.9
for (i in 1:N){
    rls.step<-rls(c(1, X[i]),y[i],t,P,mu)
    t<-rls.step[[1]]
    P<-rls.step[[2]]
    plot(X[1:i],y[1:i],
         xlim=c(-4,4),
         ylim=c(-2,2),
         main=paste("Forgetting factor mu<-",mu))

    lines(X[1:i],cbind(array(1,c(i,1)), X[1:i])%*%t,
         col="red",
         )
}
```

•

## 7.4 Discriminant functions for classification

Let $\mathbf{x} \in \mathbb{R}^n$ denote a real valued random input vector and $\mathbf{y}$ a categorical random output variable that takes values in the set $\{c_1, \ldots, c_K\}$ such that

$$\sum_{k=1}^{K} \text{Prob}\{\mathbf{y} = c_k | x\} = 1$$

A classifier can be represented in terms of a set of $K$ discriminant functions $g_k(x)$, $k = 1, \ldots, K$ such that the classifier applies the following decision rule: *assigns a*

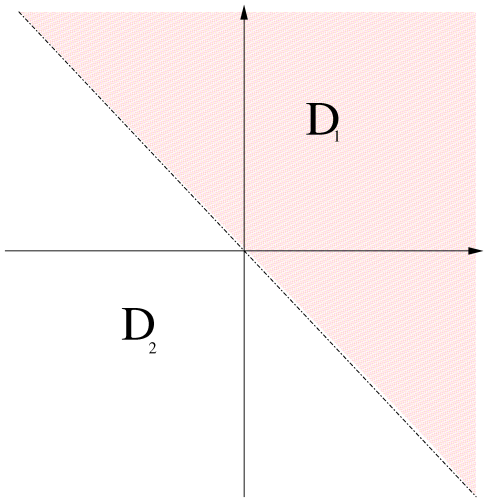



Figure 7.7: Decision boundary and decision regions for the binary discrimination functions $g_1(x) = 3x_1 + x_2 + 2$ and $g_2(x) = 2x_1 + 2$

*feature vector $x$ to a class $\hat{y}(x) = c_k$ if*

$$g_k(x) > g_j(x) \quad \text{for all } j \neq k \tag{7.4.35}$$

Section 5.3 showed that in the case of a zero-one loss function (Equation (5.3.14)), the optimal classifier corresponds to a *maximum a posteriori* discriminant function $g_k(x) = \text{Prob}\{\mathbf{y} = c_k|x\}$. This means that if we are able to define the $K$ functions $g_k(\cdot)$, $k = 1, \ldots, K$ and we apply the classification rule (7.4.35) to an input $x$, we obtain a classifier which is equivalent to the Bayes one.

The discriminant functions divide the feature space into $K$ *decision regions* $D_k$, where a decision region $D_k$ is a region of the input space $\mathcal{X}$ where the discriminant classifier returns the class $c_k$ for each $x \in D_k$. The regions are separated by *decision boundaries*, i.e. surfaces in the domain of $x$ where ties occur among the largest discriminant functions.

**Example**

Consider a binary classification problem where $\mathbf{y}$ can take values in $\{c_1, c_2\}$ and $x \in \mathbb{R}^2$. Let $g_1(x) = 3x_1 + x_2 + 2$ and $g_2(x) = 2x_1 + 2$ the two discriminant functions associated to the class $x_1$ and $x_2$, respectively. The classifier will return the class $c_1$ if

$$3x_1 + x_2 + 2 > 2x_1 + 2 \Leftrightarrow x_1 > -x_2$$

The decision regions $D_1$ and $D_2$ are depicted in Figure 7.7.

•

We can multiply all the discriminant functions by the same positive constant or shift them by the same additive constant without influencing the decision. More generally, if we replace every $g_k(z)$ by $f(g_k(z))$, where $f(\cdot)$ is a monotonically increasing function, the resulting classification is unchanged.

For example in the case of a zero/one loss function, any of the following choices

gives identical classification result:

$$g_k(x) = \text{Prob}\{\mathbf{y} = c_k|x\} = \frac{p(x|\mathbf{y} = c_k)P(\mathbf{y} = c_k)}{\sum_{k=1}^{K} p(x|\mathbf{y} = c_k)P(\mathbf{y} = c_k)} \tag{7.4.36}$$

$$g_k(x) = p(x|\mathbf{y} = c_k)P(\mathbf{y} = c_k) \tag{7.4.37}$$

$$g_k(x) = \ln p(x|\mathbf{y} = c_k) + \ln P(\mathbf{y} = c_k) \tag{7.4.38}$$

and returns a minimum-error-rate classification by implementing a Bayes classifier.

#### 7.4.0.5 Discriminant functions in the Gaussian case

Let us consider a binary classification task where the inverse conditional densities are multivariate normal, i.e. $p(\mathbf{x} = x|\mathbf{y} = c_k) \sim \mathcal{N}(\mu_k, \Sigma_k)$ where $x \in \mathbb{R}^n$, $\mu_k$ is a $[n, 1]$ vector and $\Sigma_k$ is a $[n, n]$ covariance matrix. Since

$$p(\mathbf{x} = x|\mathbf{y} = c_k) = \frac{1}{(\sqrt{2\pi})^n \sqrt{\det(\Sigma_k)}} \exp\left\{-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k)\right\}$$

from (7.4.38) we obtain

$$g_k(x) = \ln p(x|\mathbf{y} = c_k) + \ln P(\mathbf{y} = c_k) \tag{7.4.39}$$

$$= -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) - \frac{n}{2}\ln 2\pi - \frac{1}{2}\ln\det(\Sigma_k) + \ln P(\mathbf{y} = c_k) \tag{7.4.40}$$

Let us consider the simplest case, that is all the distributions have the same diagonal covariance matrix $\Sigma_k = \sigma^2 I$ where $I$ is the $[n, n]$ identity matrix.

In geometrical terms, this means that, for each class, the $x$ samples fall in equal-size spherical clusters which are parallel to the axes. In analytical terms, this means that the two quantities in (7.4.39)

$$\det(\Sigma_k) = \sigma^{2n}, \qquad \Sigma_k^{-1} = (1/\sigma^2)I$$

are independent of $k$, i.e. they are unimportant additive constants that can be ignored by the decision rule (7.4.35). From (7.4.39), we obtain the simpler discriminant function

$$\begin{aligned} g_k(x) &= -\frac{\|x - \mu_k\|^2}{2\sigma^2} + \ln P(\mathbf{y} = c_k) \\ &= -\frac{(x - \mu_k)^T(x - \mu_k)}{2\sigma^2} + \ln P(\mathbf{y} = c_k) \\ &= -\frac{1}{2\sigma^2}[x^T x - 2\mu_k^T x + \mu_k^T \mu_k] + \ln P(\mathbf{y} = c_k) \end{aligned}$$

However, since the quadratic term $x^T x$ is the same for all $k$, making it an ignorable additive constant, this is equivalent to a linear discriminant function

$$g_k(x) = w_k^T x + w_{k0}$$

where $w_k$ is a $[n, 1]$ vector

$$w_k = \frac{1}{\sigma^2}\mu_k$$

and the term $w_{k0}$

$$w_{k0} = -\frac{1}{2\sigma^2}\mu_k^T \mu_k + \ln P(\mathbf{y} = c_k)$$

is called the *bias* or threshold.

In the two-classes problem, the decision boundary (i.e. the set of points where $g_1(x) = g_2(x)$) can be obtained by solving the identity

$$w_1^T x + w_{10} = w_2^T x + w_{20} \Leftrightarrow (w_1 - w_2)^T x - (w_{20} - w_{10}) = 0$$

We obtain an hyperplane having equation

$$w^T(x - x_0) = 0 \tag{7.4.41}$$

where

$$w = \frac{\mu_1 - \mu_2}{\sigma^2}$$

and

$$x_0 = \frac{1}{2}(\mu_1 + \mu_2) - \frac{\sigma^2}{\|\mu_1 - \mu_2\|^2} \ln \frac{\text{Prob}\{\mathbf{y} = c_1\}}{\text{Prob}\{\mathbf{y} = c_2\}}(\mu_1 - \mu_2)$$

This can be verified by the fact that $w^T x_0 = w_{20} - w_{10}$. The equation (7.4.41) defines a hyperplane through the point $x_0$ and orthogonal to the vector $w$.

#### 7.4.0.6 Uniform prior case

If the prior probabilities $P(\mathbf{y} = c_k)$ are the same for all the $K$ classes, then the term $\ln P(\mathbf{y} = c_k)$ is an unimportant additive constant that can be ignored. In this case, it can be shown that the optimum decision rule is a *minimum distance classifier*. This means that in order to classify an input $x$, it measures the Euclidean distance $\|x - \mu_k\|^2$ from $x$ to each of the $K$ mean vectors, and assign $x$ to the category of the nearest mean. It can be shown that for the more generic case $\Sigma_k = \Sigma$, the discriminant rule is based on minimizing the Mahalanobis distance

$$\hat{c}(x) = \arg\min_k (x - \mu_k)^T \Sigma^{-1} (x - \mu_k)$$

#### R script

The R script `discri.R` considers a binary classification task where $\mathbf{x} \in \mathbb{R}^2$ and the inverse conditional distributions of the two classes are $\mathcal{N}(\mu_1, \sigma^2 I)$ and $\mathcal{N}(\mu_2, \sigma^2 I)$, respectively. Suppose that the two a priori probabilities are identical, that $\sigma = 1$, $\mu_1 = [-1, -2]^T$ and $\mu_2 = [2, 5]^T$. The positions of 100 points randomly drawn from $\mathcal{N}(\mu_1, \sigma^2 I)$, of 100 points drawn from $\mathcal{N}(\mu_2, \sigma^2 I)$ together with the optimal decision boundary computed by (7.4.41) are plotted in Figure 7.8.

```
## script discri.R

rm(list=ls())
norm<-function(x){
  sqrt(sum(x^2))
}

N1<-100  ## number samples class1
N2<-100  ## number samples class2
P<-c(N1,N2)/(N1+N2)
sigma2<-1

mu.1 <- c(-1,-2)  ## mean of cluster 1
mu.2<-c(2,5)    ## mean of cluster 2
```

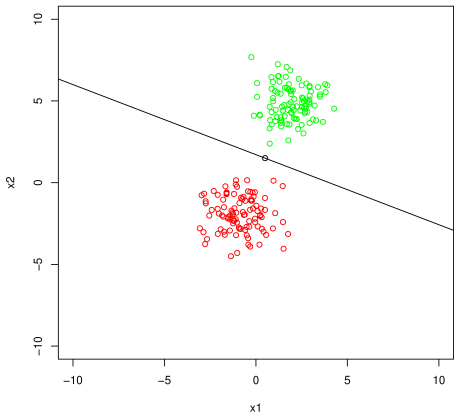



Figure 7.8: Binary classification problem: distribution of inputs and linear decision boundary

```
## Data generation
X1<-cbind(rnorm(N1,mu.1[1],sqrt(sigma2)),rnorm(N1,mu.1[2],sqrt(sigma2)))
X2<-cbind(rnorm(N2,mu.2[1],sqrt(sigma2)),rnorm(N2,mu.2[2],sqrt(sigma2)))


plot(X1[,1],X1[,2],col="red",xlim=c(-10,10),ylim=c(-10,10),xlab="x1",ylab="x2")
points(X2[,1],X2[,2],col="green")

Xd<--10:10
w<-mu.1-mu.2
x0<-0.5*(mu.1+mu.2)-sigma2/(norm(mu.1-mu.2)^2)*(mu.1-mu.2)*log(P[1]/P[2])

m<--w[1]/w[2]
intc<-w[1]/w[2]*x0[1]+x0[2]

abline(a=intc,b=m)
points(x0[1],x0[2])
```

•

### 7.4.1 Perceptrons

Consider a binary classification task (Figure 7.9). The previous section presented a technique to separate input data by a linear boundary. In a generic configuration, however, the problem is ill-posed and there are infinitely many possible *separating hyperplanes* (Figure 7.9) characterized by the equation

$$\beta_0 + x^T\beta = 0 \tag{7.4.42}$$

If $x \in \mathbb{R}^2$, this equation represents a line. In a generic case ($x \in \mathbb{R}^n$) some properties hold for all hyperplanes

- Since for any two points $x_1$ and $x_2$ lying on the hyperplane we have ($x_1 -$

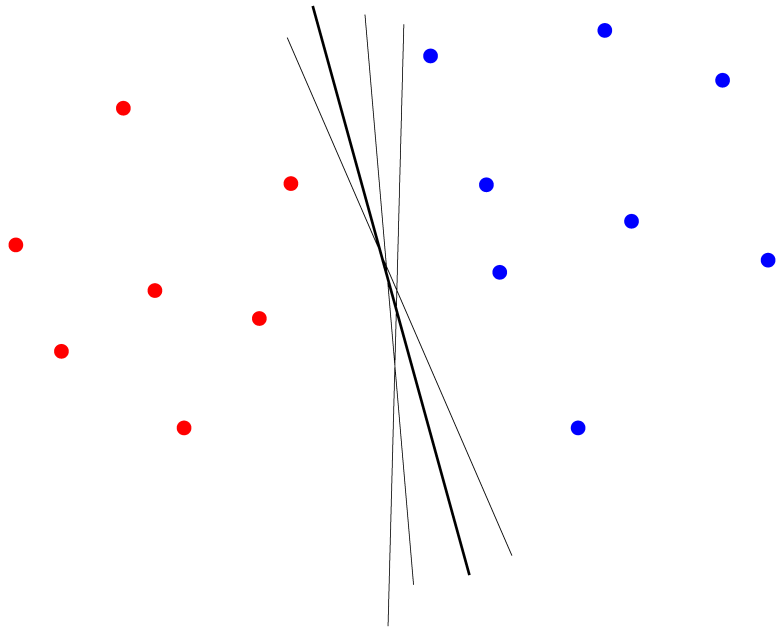


Figure 7.9: Separating hyperplane

$x_2)^T\beta = 0$, the vector normal to the hyperplane (Figure 7.10) is given by

$$\beta^* = \frac{\beta}{\|\beta\|}$$

- The signed distance of a point $x$ to the hyperplane (Figure 7.10) is called the *geometric margin* and is given by

$$\beta^{*T}(x - x_0) = \frac{x^T\beta - \beta x_0^T}{\|\beta\|} = \frac{1}{\|\beta\|}(x^T\beta + \beta_0)$$

Classifiers that use the sign of the linear combination $h(x, \beta) = \beta_0 + \beta^T x$ to perform classification were called *perceptrons* in the engineering literature in the late 1950. The class returned by a perceptron for a given input $x_q$ is

$$\begin{cases} 1 & \text{if } \beta_0 + x_q^T\beta = \beta_0 + \sum_{j=1}^n x_{qj}\beta_q > 0 \\ -1 & \text{if } \beta_0 + x_q^T\beta = \beta_0 + \sum_{j=1}^n x_{qj}\beta_q < 0 \end{cases}$$

In other terms the decision rule is given by

$$h(x) = \text{sgn}(\beta_0 + x^T) \tag{7.4.43}$$

For all well classified points in the training set the following relation hold

$$\gamma_i = y_i(x_i^T\beta + \beta_0) > 0$$

where the quantity $\gamma_i$ is called the *functional margin* of the pair $\langle x_i, y_i\rangle$ with respect to the hyperplane (7.4.42).

Misclassifications in the training set occur when

$$\begin{cases} y_i = 1 & \text{but } \beta_0 + \beta^T x_i < 0 \\ y_i = -1 & \text{but } \beta_0 + \beta^T x_i > 0 \end{cases} \Leftrightarrow y_i(\beta_0 + \beta^T x_i) < 0$$

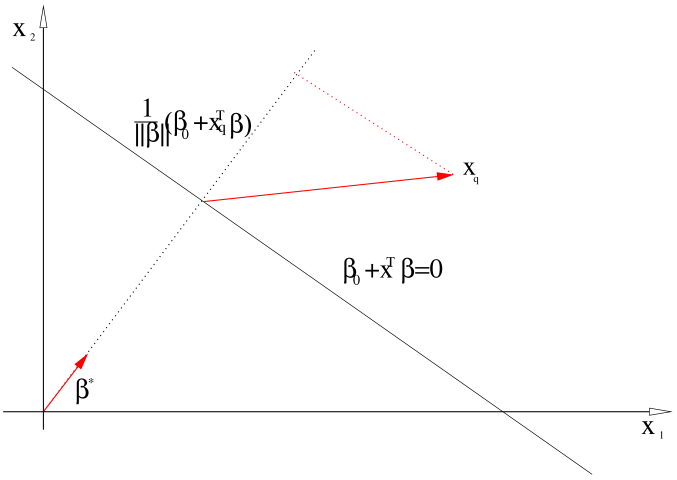



Figure 7.10: Bidimensional space ($n = 2$): vector $\beta^*$ normal to the hyperplane and distance of a point from an hyperplane

The parametric identification step of a perceptron learning procedure aims at finding the values $\{\beta, \beta_0\}$ that minimize the quantity

$$\text{SSE}_{\text{emp}}(\beta, \beta_0) = -\sum_{i \in \mathcal{M}} y_i (x_i^T \beta + \beta_0)$$

where $\mathcal{M}$ is the subset of misclassified points in the training set.

Note that this quantity is non-negative and proportional to the distance of the misclassified points to the hyperplane. Since the gradients are

$$\frac{\partial \text{SSE}_{\text{emp}}(\beta, \beta_0)}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i x_i, \qquad \frac{\partial \text{SSE}_{\text{emp}}(\beta, \beta_0)}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i$$

a gradient descent minimization procedure (Section 6.6.2.3) can be adopted. This procedure is guaranteed to converge provided there exists a hyperplane that correctly classifies the data: this configuration is called *linearly separable*.

Although the perceptron set the foundations for much of the following research in machine learning, a number of problems with this algorithm have to be mentioned:

- When the data are separable, there are many possible solutions, and which one is found depends on the initialization of the gradient method.
- When the data are not separable, the algorithm will not converge.
- Also for a separable problem the convergence of the gradient minimization can be very slow.

A possible solution to the separating hyperplane problem has been proposed by the SVM technique.

### 7.4.2 Support vector machines

This technique relies on an optimization approach to the computation of separating hyperplanes.

Let us define as *geometric margin of a hyperplane with respect to a training dataset* the minimum of the geometric margin of the training points. Also, the

*margin of a training set* is the maximum geometric margin over all hyperplanes. The hyperplane realizing this maximum is known as a *maximal margin hyperplane*.

The SVM approach computes the maximal margin hyperplane for a training set. In other words, the SVM optimal separating hyperplane is the one which separates the two classes by maximizing the distance to the closest point from either class. This approach provides a unique solution to the separating hyperplane problem and was shown to lead to good classification performance on real data. The search for the optimal hyperplane is modeled as the optimization problem

$$\max_{\beta,\beta_0} C \tag{7.4.44}$$

$$\text{subject to } \frac{1}{\|\beta\|} y_i(x_i^T\beta + \beta_0) \geq C \quad \text{for } i = 1, \ldots, N \tag{7.4.45}$$

where the constraint ensures that all the points are at least a distance $C$ from the decision boundary defined by $\beta$ and $\beta_0$. The SVM parametric identification step seeks the largest $C$ that satisfies the constraints and the associated parameters.

Since the hyperplane (7.4.42) is equivalent to the original hyperplane where the parameters $\beta_0$ and $\beta$ have been multiplied by a constant, we can set $\|\beta\| = 1/C$. The maximization problem can be reformulated in a minimization form

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 \tag{7.4.46}$$

$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1 \quad \text{for } i = 1, \ldots, N \tag{7.4.47}$$

where the constraints impose a margin around the linear decision of thickness $1/\|\beta\|$. This optimization problem is a convex optimization problem (quadratic criterion with linear inequality constraints) where the primal Lagrangian is

$$L_P(\beta, \beta_0) = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{N} \alpha_i[y_i(x_i^T\beta + \beta_0) - 1] \tag{7.4.48}$$

and $\alpha_i \geq 0$ are the Lagrangian multipliers.

Setting the derivatives wrt $\beta$ and $\beta_0$ to zero we obtain:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i, \qquad 0 = \sum_{i=1}^{N} \alpha_i y_i \tag{7.4.49}$$

Substituting these in the primal form (7.4.48) we obtain the Wolfe dual

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N} \alpha_i\alpha_k y_i y_k x_i^T x_k \tag{7.4.50}$$

subject to $\alpha_i \geq 0$.

The dual optimization problem is now

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N} \alpha_i\alpha_k y_i y_k x_i^T x_k = \max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N} \alpha_i\alpha_k y_i y_k \langle x_i, x_k\rangle \tag{7.4.51}$$

$$\text{subject to } 0 = \sum_{i=1}^{N} \alpha_i y_i, \tag{7.4.52}$$

$$\alpha_i \geq 0, \quad i = 1, \ldots, N \tag{7.4.53}$$

where $\langle x_i, x_k \rangle$ is the inner product of $x_i$ and $x_k$.

Note that the problem formulation requires the computation of all the inner products $\langle x_i, x_k \rangle, i = 1, \ldots, N, k = 1, \ldots, N$. This boils down to the computation of the Gram matrix

$$G = XX^T \tag{7.4.54}$$

It can be shown that the optimal solution must satisfy the Karush-Kuhn-Tucker (KKT) condition

$$\alpha_i[y_i(x_i^T\beta + \beta_0) - 1] = 0, \qquad \forall i$$

The above condition means that we are in either of these two situations

1. $y_i(x_i^T\beta + \beta_0) = 1$, i.e. the point is on the boundary of the margin, then $\alpha_i > 0$
2. $y_i(x_i^T\beta + \beta_0) > 1$, i.e. the point is not on the boundary of the margin, then $\alpha_i = 0$

The training points having an index $i$ such that $\alpha_i > 0$ are called the *support vectors*

Given the solution $\alpha$ and obtained $\beta$ from (7.4.49), the term $\beta_0$ is obtained by

$$\beta_0 = -\frac{1}{2}[\beta x^*(1) + \beta x^*(-1)]$$

where we denote by $x^*(1)$ some (any) support vector belonging to the first class and we denote by $x^*(-1)$ a support vector belonging to the second class.

Now, the decision function can be written as

$$h(x, \beta, \beta_0) = \text{sign}[x^T\beta + \beta_0]$$

or equivalently

$$h(x, \beta, \beta_0) = \text{sign}[\sum_{\text{support vectors}} y_i\alpha_i\langle x_i x\rangle + \beta_0] \tag{7.4.55}$$

This is an attractive property of support vector machines: the classifier can be expressed as a function of a limited number of points of the training set, the so called *support vectors* which are on the boundaries. This means that in SVM all the points far from the class boundary do not play a major role, unlike the linear discriminant rule where the mean and the variance of the class distributions determine the separating hyperplane (see Equation (7.4.41)). It can be shown, also, that in the separable case

$$C = \frac{1}{\|\beta\|} = \frac{1}{\sqrt{\sum_{i=1}^N \alpha_i}} \tag{7.4.56}$$

**R script**

The R script `svm.R` considers a binary classification problem. It generates sets of separable data and builds a separating hyperplane by solving the problem (7.4.48). The training points belonging to the two classes (in red and blue), the separating hyperplane, the boundary of the margin and the support vectors (in black) are plotted for each training set (see Figure 7.11 ).

•

A modification of the formulation (7.4.44) occurs when we suppose that the classes are nonlinearly separable. In this case the dual problem (7.4.51) is unbounded. The idea is still to maximize the margin but by allowing some points to

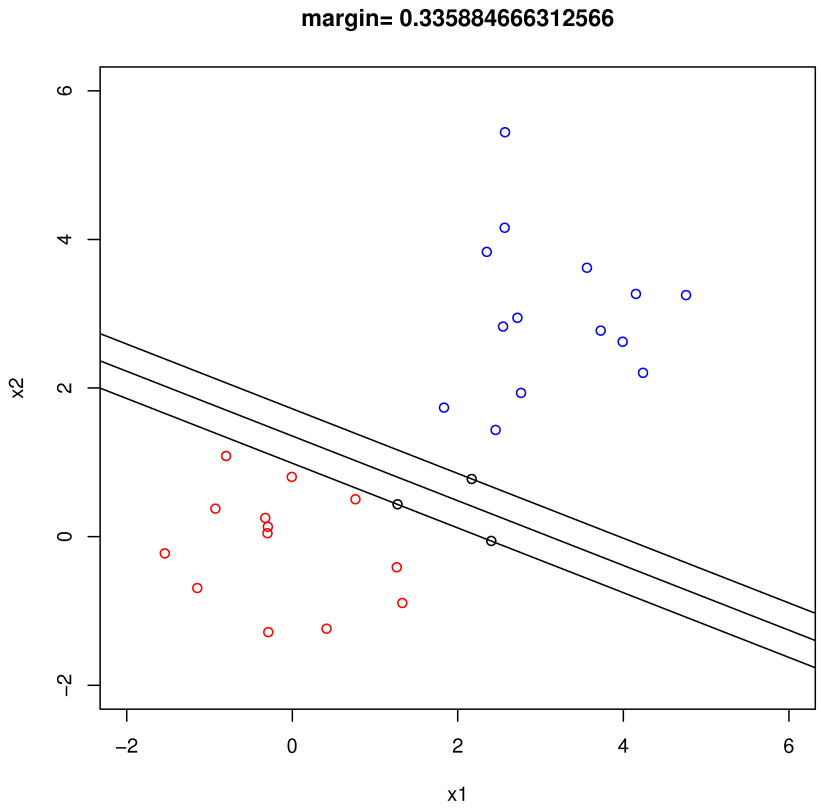



Figure 7.11: Maximal margin hyperplane for a binary classification task with the support vectors in black

be misclassified. For each example $\langle x_i, y_i \rangle$ we define the *slack variable* $\xi_i$ and we relax the constraints (7.4.45) into

$$\frac{1}{\|\beta\|} y_i(x_i^T \beta + \beta_0) \geq C(1 - \xi_i) \quad \text{for } i = 1, \ldots, N \tag{7.4.57}$$

$$\xi_i \geq 0 \tag{7.4.58}$$

$$\sum_{i=1}^{N} \xi_i \leq \gamma \tag{7.4.59}$$

The value $\xi_i$ represents the proportional amount by which the quantity $y_i(x_i^T \beta + \beta_0)$ can be lower than $C$ and the norm $\|\xi\|$ measures how much the training set fails to have a margin $C$. Note that since misclassifications occur when $\xi_i > 1$, the upper bound $\gamma$ of $\sum_{i=1}^{N} \xi_i$ represents the maximum number of allowed misclassifications in the training set. It can be shown [62] that the maximization (7.4.44) with the above constraints can be put in the equivalent quadratic form

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k \tag{7.4.60}$$

$$\text{subject to } 0 = \sum_{i=1}^{N} \alpha_i y_i, \tag{7.4.61}$$

$$0 \leq \alpha_i \leq \gamma, \quad i = 1, \ldots, N \tag{7.4.62}$$

The decision function takes again the form (7.4.55) where $\beta_0$ is chosen so that $y_i h(x_i) = 1$ for any $i$ such that $0 < \alpha_i < \gamma$. The geometric margin takes the value

$$C = \left( \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k \right)^{-1/2} \tag{7.4.63}$$

Note that the set of points for which the corresponding slack variables satisfy $\xi_i > 0$ are also the points for which $\alpha_i = \gamma$.

**R script**

The R script `svm.R` solves a non separable problem by setting the boolean variable `separable` to FALSE. Figure 7.12 plots: the training points belonging to the two classes (in red and blue), the separating hyperplane, the boundary of the margin, the support vectors (in black), the points of the red class for which the slack variable is positive (in yellow) and the points of the blue class for which the slack variable is positive (in green).

•

Once the value $\gamma$ is fixed, the parametric identification in the SVM approach boils down to a quadratic optimization problem for which a large amount of methods and numerical software exists. The value $\gamma$ plays the role of complexity parameter which bounds the total proportional amount by which classifications fall on the wrong side of the margin. In practice, the choice of this parameter requires a structural identification loop where the parameter $\gamma$ is varied through a wide range of values and assessed through a validation strategy.

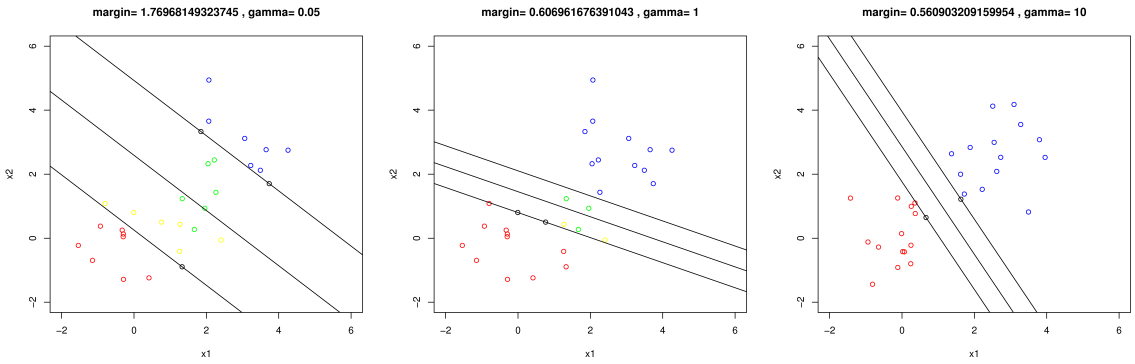



Figure 7.12: Maximal margin hyperplane for a non separable binary classification task for different values of $C$: support vectors are in black, the slack points of the red class are in yellow and the slack points of the blue class are in green

# Chapter 8

# Nonlinear approaches

In the previous chapter we have considered input/output regression problems where the relation between input and output is linear and classification problems where the optimal decision boundaries are linear.

The advantage of linear models are numerous:

- the least-squares $\hat{\beta}$ estimate can be expressed in an analytical form and can be easily calculated through matrix computation.
- statistical properties of the estimator can be easily defined.
- recursive formulation for sequential updating are available.

Unfortunately in real problems it is extremely unlikely that the input and output variables are linked by a linear relation. Moreover, the form of the relation is often unknown and only a limited amount of samples is available. Along the years statisticians and machine learning researchers have proposed a number of nonlinear approaches, with the aim of finding approximators able to combine high generalization with effective learning procedures. The presentation of these techniques could be organized according to several criteria and principles. In this chapter we will focus on the distinction between global and divide-and-conquer approaches.

A family of models traditionally used in supervised learning is the family of *global models* which describes the relationship between the input and the output values as a single analytical function over the whole input domain (Fig. 8.1). In general, this makes sense when it is reasonable to believe that a physical-like law describes the data over the whole set of operating conditions. Examples of well-known global parametric models in literature are the linear models discussed in the previous chapter, generalized linear models and neural networks which will be presented in Section 8.1.1.

A nice property of global modeling is that, even for huge datasets, a parametric model can be stored in a small memory. Moreover, the evaluation of the model requires a short program that can be executed in a reduced amount of time. These features have undoubtedly contributed to the success of the global approach in years when most computing systems imposed severe limitations on users.

However, for a generic global model, the parametric identification (Section 5.2) consists in a nonlinear optimization problem (see Equation 5.2.5) which is not analytically tractable due to the numerous local minima and for which only a suboptimal solution can be found through a slow iterative procedure. Similarly, the problem of selecting the best model structure in a generic nonlinear case cannot be handled in analytical form and requires time consuming validation procedures.

For these reasons, in recent years, alternatives to global modeling techniques, as the *divide-and-conquer* approach, gained popularity in the modeling community.

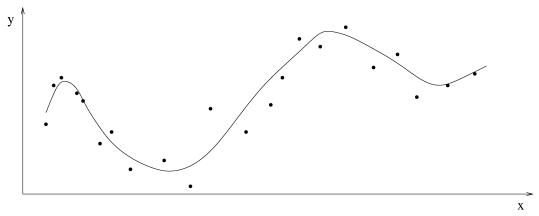



Figure 8.1: A global model (solid line) which fits the training set (dotted points) for a learning problem with one input variable (x-axis) and one output variable (y-axis).

The *divide-and-conquer* principle consists in attacking a complex problem by dividing it into simpler problems whose solutions can be combined to yield a solution to the original problem. This principle presents two main advantages. The first is that simpler problems can be solved with simpler estimation techniques: in statistical language this means to adopt linear techniques, well studied and developed over the years. The second is that the learning method can better adjust to the properties of the available dataset. Training data are rarely distributed uniformly in the input space. Whenever the distribution of patterns in the input space is uneven, a proper local adjustment of the learning algorithm can significantly improve the overall performance.

We will focus on two main instances of the divide-and-conquer principle: the modular approach, which originated in the field of system identification, and the local modeling approach, which was first proposed in the statistical nonparametric literature.

*Modular* architectures are input/output approximators composed of a number of modules which cover different regions of the input space. This is the idea of *operating regimes* which propose a partitioning of the operating range of the system as a more effective way to solve modeling problems (Section 8.1.2).

Although these architectures are a modular combination of local models, their learning procedure is still performed on the basis of the whole dataset. Hence, learning in modular architectures remains a functional estimation problem, with the advantage that the parametric identification can be made simpler by the adoption of local linear modules. However, in terms of structural identification the problem is still nonlinear and requires the same procedures used for generic global models.

A second example of divide-and-conquer methods are *local modeling* techniques (Section 8.1.10), which turn the problem of function estimation in a problem of value estimation. The goal is not to model the whole statistical phenomenon but to return the best output for a given test input, hereafter called the *query*. The motivation is simple: why should the problem of estimating the values of an unknown function at given points of interest be solved in two stages? Global modeling techniques first estimate the function (*induction*) and second estimate the values of the function using the estimated function (*deduction*). In this two-stage scheme one actually tries to solve a relatively simple problem (estimating the values of a function at given points of interest) by first solving, as an intermediate problem, a much more difficult one (estimating the function).

Local modeling techniques take an alternative approach, defined as *transduction* by Vapnik [115] (Fig. 8.2). They focus on approximating the function only in the neighborhood of the point to be predicted. This approach requires to keep in memory the dataset for each prediction, instead of discarding it as in the global

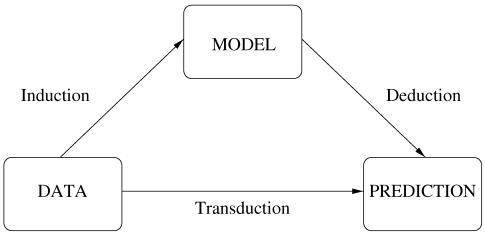



Figure 8.2: Function estimation (model induction + model evaluation) vs. value estimation (direct prediction from data).

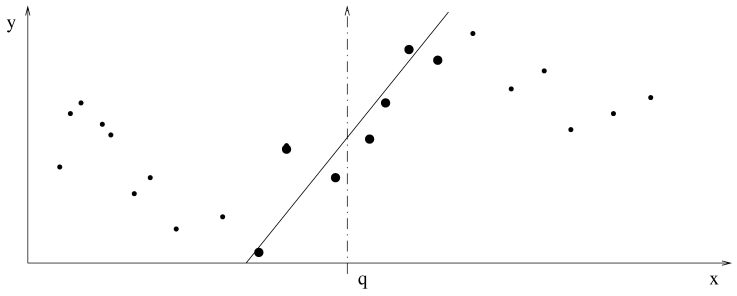



Figure 8.3: Local modeling of the input/output relationship between the input variable $x$ and the output variable $y$, on the basis of a finite set of observations (dots). The value of the variable $y$ for $x = q$ is returned by a linear model (solid line) which fits the samples in a neighborhood of the query point (bigger dots).

modeling case. At the same time, local modeling requires only simple approximators, e.g. constant and/or linear, to model the dataset in a neighborhood of the query point. An example of local linear modeling in the case of a single-input single-output mapping is presented in Fig. 8.3.

Many names have been used in the past to label variations of the local modeling approach: memory-based reasoning [108], case-based reasoning [77], local weighted regression [27], nearest neighbor [30], just-in-time [32], lazy learning [3], exemplar-based, instance based [2],... These approaches are also called *nonparametric* in the literature [60, 107], since they relax the assumptions on the form of a regression function, and let the data search for a suitable function that describes well the available data.

In the following we will present in detail some machine learning techniques for nonlinear regression and classification.

## 8.1 Nonlinear regression

A general way of representing the unknown input/output relation in a regression setting is the **regression plus noise form**

$$\mathbf{y} = f(x) + \mathbf{w}$$

where $f(\cdot)$ is a deterministic function and the term $\mathbf{w}$ represents the noise or random error. It is typically assumed that $\mathbf{w}$ is independent of $\mathbf{x}$ and $E[\mathbf{w}] = 0$.

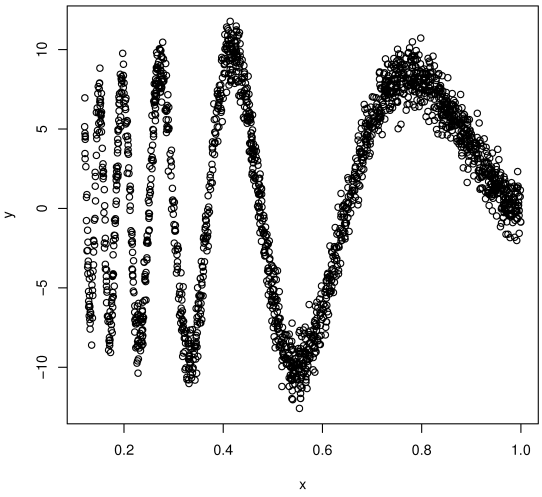



Figure 8.4: Training set obtained by sampling uniformly in the input domain a Dopler function distorted with Gaussian noise.

Suppose that we have available a **training set** $\{\langle x_i, y_i\rangle : i = 1, \ldots, N\}$, where $x_i = (x_{i1}, \ldots, x_{in})$, generated according to the previous model. The goal of a learning procedure is to find a model $h(x)$ which is able to give a good approximation of the unknown function $f(x)$.

**Example**

Consider an input/output mapping represented by the *Dopler* function

$$f(x) = 20\sqrt{x(1-x)}\sin(2\pi\frac{1.05}{x+0.05}) \tag{8.1.1}$$

distorted by additive Gaussian noise $\mathbf{w}$ with unit variance.

The training set is made of $N = 2000$ points obtained by sampling the input domain $\mathcal{X} = [0.12, 1]$ through a random uniform distribution (Fig. 8.4). This stochastic dependency and the related training dataset (see R script `dopler.R`) will be used to assess the performance of the techniques we are going to present.

•

### 8.1.1 Artificial neural networks

Artificial neural networks (ANN) (aka *neural nets*) are parallel, distributed information processing computational models which draw their inspiration from neurons in the brain. However, one of the most important trends in recent neural computing has been to move away from a biologically inspired interpretation of neural networks to a more rigorous and statistically founded interpretation based on results deriving from statistical pattern recognition theory.

The main class of neural network used in supervised learning for classification and regression is the *feed-forward network*, aka as multi-layer perceptron (MLP). Feed-forward ANN (FNN) have been applied to a wide range of prediction tasks in such diverse fields as speech recognition, financial prediction, image compression, adaptive industrial control.

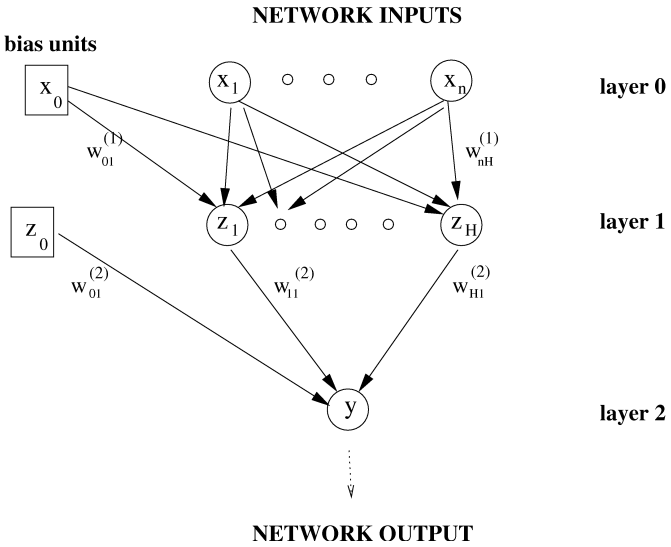



Figure 8.5: Two-layer feed-forward NN

#### 8.1.1.1 Feed-forward architecture

Feed-forward NN have a *layered* architecture, with each layer comprising one or more simple processing units called artificial neurons or *nodes* (Figure 8.5). Each node is connected to one or more other nodes by real valued *weights* (in the following we will refer to them as parameters) but not to nodes in the same layer. All FNN have an input layer and an output layer. FNN are generally implemented with an additional node, called the *bias unit*, in all layers except the output layer. This node plays the role of the intercept term $\beta_0$ in linear models.

For simplicity, henceforth, we will consider only FNN with one single output. Let

- $n$ be the number of inputs,
- $L$ the number of layers,
- $H^{(l)}$ the number of hidden units of the $l$th layer ($l = 1, \ldots, L$) of the FNN,
- $w_{kv}^{(l)}$ denote the weight of the link connecting the $k$th node in the $l-1$ layer and the $v$th node in the $l$ layer,
- $z_v^{(l)}$, $v = 1, \ldots, H^{(l)}$ the output of the $v$th hidden node of the $l$th layer,
- $z_0^{(l)}$ denote the bias for the $l$, $l = 1, \ldots, L$ layer.
- Let $H^{(0)} = n$ and $z_v^{(0)} = x_v$, $v = 0, \ldots, n$.

For $l \geq 1$ the output of the $v$th, $v = 1, \ldots, H^{(l)}$, hidden unit of the $l$th layer, is obtained by first forming a weighted linear combination of the $H^{(l-1)}$ outputs of the lower level

$$a_v^{(l)} = \sum_{k=1}^{H^{(l-1)}} w_{kv}^{(l)} z_k^{(l-1)} + w_{0v}^{(l)} z_0^{(l-1)}, \qquad v = 1, \ldots, H^{(l)}$$

and then by transforming the sum using an *activation* function to give

$$z_v^{(l)} = g^{(l)}(a_v^{(l)}), \qquad v = 1, \ldots, H^{(l)}$$

The activation function $g^{(l)}(\cdot)$ is typically a nonlinear transformation like the *logistic* or *sigmoid* function

$$g^{(l)}(z) = \frac{1}{1+e^{-z}}$$

For $L = 2$ (i.e. single hidden layer or two-layer feed-forward NN), the input/output relation is given by

$$\hat{y} = h(x, \alpha_N) = g^{(2)}(a_1^{(2)}) = g^{(2)}\left(\sum_{k=1}^{H} w_{k1}^{(2)} z_k + w_{01}^{(2)} z_0\right)$$

where

$$z_k = g^{(1)}\left(\sum_{j=1}^{n} w_{jk}^{(1)} x_j + w_{0k}^{(1)} x_0\right), \qquad k = 1, \ldots, H$$

Note that if $g^{(1)}(\cdot)$ and $g^{(2)}(\cdot)$ are linear mappings, this functional form becomes linear.

Once given the number of inputs and the form of the function $g(\cdot)$ two are the parameters which remain to be chosen: the value of weights $w^{(l)}$, $l = 1, 2$ and the number of hidden nodes $H$. Note that the set of weights of a FNN represents the set of parameters $\alpha_N$ introduced in Section 5.1 when the hypothesis function $h(\cdot)$ is modeled by a FNN. The calibration procedure of the weights on the basis of a training dataset represents the parametric identification procedure in neural networks. This procedure is normally carried out by a backpropagation algorithm which will be discussed in the following section.

The number $H$ of hidden nodes represents the complexity $s$ in Equation (5.6.39). By increasing the value $H$ we increase the class of input/output functions that can be represented by the FNN. In other terms, the choice of the number of hidden nodes affects the representation power of the FNN approximator and constitutes the structural identification procedure in FNN (Section 8.1.1.3) .

#### 8.1.1.2 Backpropagation

Backpropagation is an algorithm which, once the number of hidden nodes $H$ is given, estimates the weights $\alpha_N = \{w^{(l)}, l = 1, 2\}$ on the basis of the training set $D_N$. It is a gradient-based algorithm which aims to minimize the cost function

$$\mathrm{SSE}_{\mathrm{emp}}(\alpha_N) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{N} (y_i - h(x_i, \alpha_N))^2$$

where $\alpha_N = \{w^{(l)}, l = 1, 2\}$ is the set of weights.

The backpropagation algorithm exploits the network structure in order to compute recursively the gradient.

The simplest (and least effective) backprop algorithm is an iterative gradient descent which is based on the iterative formula

$$\alpha_N(k+1) = \alpha_N(k) - \eta \frac{\partial \mathrm{SSE}_{\mathrm{emp}}(\alpha_N(k))}{\partial \alpha_N(k)} \tag{8.1.2}$$

where $\alpha_N(k)$ is the weight vector at the $k$th iteration and $\eta$ is the learning rate which indicates the relative size of the change in weights.

The weights are initialized with random values and are changed in a direction that will reduce the error. Some convergence criterion is used to terminate the algorithm. This method is known to be inefficient since many steps are needed to reach a stationary point and no monotone decrease of $\mathrm{SSE}_{\mathrm{emp}}$ is guaranteed. More

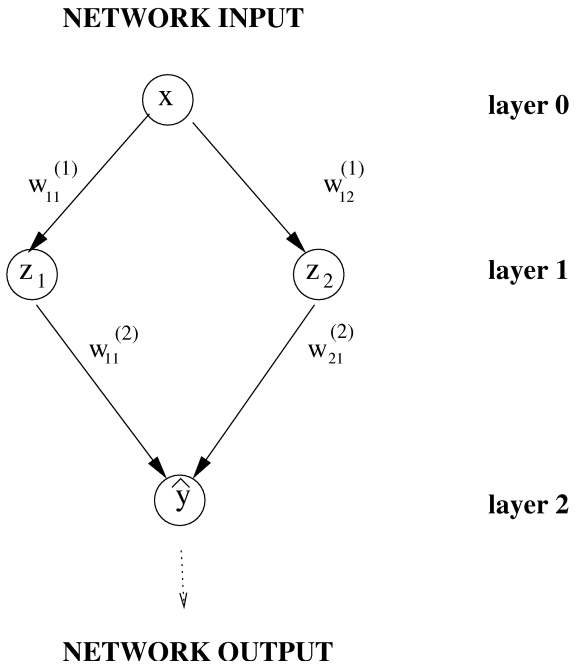



Figure 8.6: Single-input single-output neural network with one hidden layer, two hidden nodes and no bias units.

effective versions of the algorithm are based on the Levenberg-Marquardt algorithm (Section 6.6.2.6).

In order to better illustrate how the derivatives are computed in (8.1.2), let us consider a simple single-input (i.e. $n = 1$) single-output neural network with one hidden layer, two hidden nodes and no bias units (Figure 8.6). Since

$$a_1(x) = w_{11}^{(2)} z_1 + w_{21}^{(2)} z_2$$

the FNN predictor takes the form

$$\hat{y}(x) = h(x, \alpha_N) = g(a_1(x)) = g(w_{11}^{(2)} z_1 + w_{21}^{(2)} z_2) = g(w_{11}^{(2)} g(w_{11}^{(1)} x) + w_{21}^{(2)} g(w_{12}^{(1)} x))$$

where $\alpha_N = [w_{11}^{(1)}, w_{12}^{(1)}, w_{11}^{(2)}, w_{21}^{(2)}]$ The backprop algorithm needs the derivatives of $\text{SSE}_{\text{emp}}$ wrt to each weight $w \in \alpha_N$. Since for each $w \in \alpha_N$

$$\frac{\partial \text{SSE}_{\text{emp}}}{\partial w} = -2 \sum_{i=1}^{N} (y_i - \hat{y}(x_i)) \frac{\partial \hat{y}(x_i)}{\partial w}$$

and the terms $(y_i - \hat{y}(x_i))$ are easy to be computed, we focus on $\frac{\partial \hat{y}}{\partial w}$.

As far as the weights $\{w_{11}^{(2)}, w_{21}^{(2)}\}$ of the hidden/output layer are concerned, we have

$$\frac{\partial \hat{y}(x)}{\partial w_{v1}^{(2)}} = \frac{\partial g}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial w_{v1}^{(2)}} = g'(a_1^{(2)}(x)) z_v(x), \quad v = 1, \ldots, 2 \tag{8.1.3}$$

where

$$g'(z) = \frac{e^{-z}}{(1 + e^{-z})^2}$$

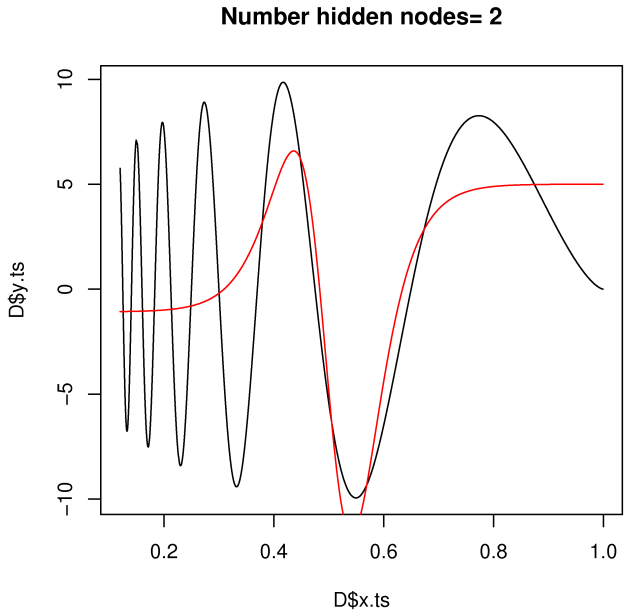



Figure 8.7: Neural network fitting with $s = 2$ hidden nodes. The red continuous line represents the neural network estimation of the Dopler function.

As far as the weights $\{w_{11}^{(1)}, w_{12}^{(1)}\}$ of the input/hidden layer

$$\frac{\partial \hat{y}(x)}{\partial w_{1v}^{(1)}} = \frac{\partial g}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_v} \frac{\partial z_v}{\partial a_v^{(1)}} \frac{\partial a_v^{(1)}}{\partial w_{1v}^{(1)}} = g'(a_1^{(2)}(x)) w_{v1}^{(2)} g'(a_v^{(1)}(x)) x$$

where the term $g'(a_1(x))$ has been already obtained during the computation of (8.1.3). This shows how the computation of the derivatives with respect to the weights of the lower layers relies on some terms which have been used in the computation of the derivatives with respect to the weights of the upper layers. In other terms, there is a sort of backpropagation of numerical terms from the upper layer to the lower layers, that justifies the name of the procedure.

Note that this algorithm presents all the typical drawbacks of the gradient-based procedures discussed in Section 6.6.2.7, like slow convergence, local minima convergence, sensitivity to the weights initialization.

**R implementation**

The FNN learning algorithm for a single-hidden layer architecture is implemented by the R library `nnet`. The script `nnet.R` shows the prediction accuracy for different number of hidden nodes (Figure 8.7 and Figure 8.8).

•

#### 8.1.1.3 Approximation properties

Let us consider a two-layer FNN with sigmoidal hidden units. This has proven to be an important class of network for practical applications. It can be shown that such networks can approximate arbitrarily well any functional (one-one or many-one) continuous mapping from one finite-dimensional space to another, provided

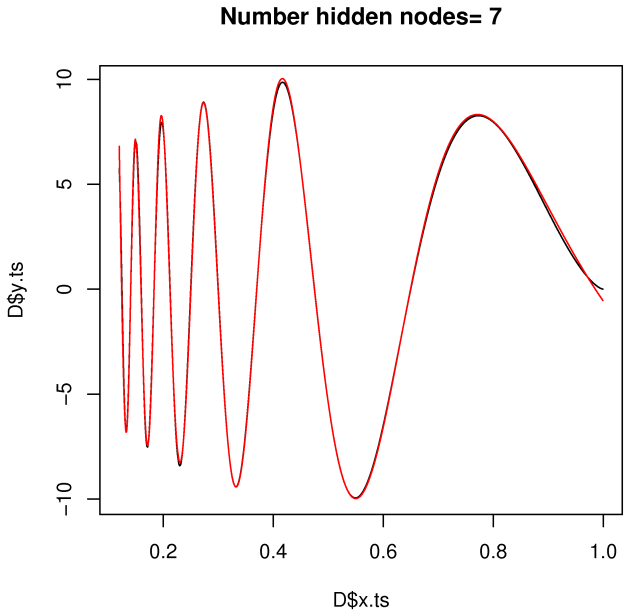



Figure 8.8: Neural network fitting with $s = 7$ hidden nodes. The red continuous line represents the neural network estimation of the Dopler function.

the number $H$ of hidden units is sufficiently large. Note that although this result is remarkable, it is of no practical use. No indication is given about the number of hidden nodes to choose for a finite number of samples and a generic nonlinear mapping.

In practice, the choice of the number of hidden nodes requires a structural identification procedure (Section 6.7) which assesses and compares several different FNN architectures before choosing the ones expected to be the closest to the optimum. Cross-validation techniques or regularization strategies based on complexity based criteria (Section 6.7.2.5) are commonly used for this purpose.

### Example

This example presents the risk of overfitting when the structural identification of a neural network is carried out on the basis of the empirical risk and not on less biased estimates of the generalization error.

Consider a dataset $D_N = \{x_i, y_i\}$, $i = 1, \ldots, N$ where $N = 50$ and

$$\mathbf{x} \in \mathcal{N}\left([0,0,0], \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right)$$

is a 3-dimensional vector. Suppose that $\mathbf{y}$ is linked to $\mathbf{x}$ by the input/output relation

$$y = x_1^2 + 4\log(|x_2|) + 5x_3$$

where $x_i$ is the $i$th component of the vector $x$. Consider as non-linear model a single-hidden-layer neural network (implemented by the R package `nnet`) with $s = 15$ hidden neurons. We want to estimate the prediction accuracy on a new i.i.d dataset

of $N_{\text{ts}} = 50$ samples. Let us train the neural network on the whole training set by using the R script `cv.R`. The empirical prediction MISE error is

$$\widehat{\text{MISE}}_{\text{emp}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - h(x_i, \alpha_N))^2 = 1.6 * 10^{-6}$$

where $\alpha_N$ is obtained by the parametric identification step. However, if we test $h(\cdot, \alpha_N)$ on the test set we obtain

$$\widehat{\text{MISE}}_{\text{ts}} = \frac{1}{N_{\text{ts}}}\sum_{i=1}^{N_{\text{ts}}}(y_i - h(x_i, \alpha_N))^2 = 22.41$$

This neural network is **seriously overfitting** the dataset. The empirical error is a **very bad** estimate of the MISE.

We perform now a $K$-fold cross-validation in order to have a better estimate of MISE, where $K = 10$. The $K = 10$ cross-validated estimate of MISE is

$$\widehat{\text{MISE}}_{\text{CV}} = 24.84$$

This figure is a much more reliable estimation of the prediction accuracy.

The leave-one-out estimate $K = N = 50$ is

$$\widehat{\text{MISE}}_{\text{loo}} = 19.47$$

It follows that the cross-validated estimate could be used to select a more appropriate number of hidden neurons.

```
## script cv.R

library(nnet)
N<- 100
n<-3

set.seed(555)
X <- rnorm(N*n)
X<-array(X,c(N,n))
Y<-X[,1]^2+5*X[,3]+4*log(abs(X[,2]))
N.tr <- N/2
X.tr <-X[1:N.tr,]
Y.tr <- Y[1:N.tr]

## 10-fold cross-validation
K<-10
N.k<-N/K
cv<-rep(0,K)

for (k in 1:K)
{
  I.k.ts<-((k-1)*N.k+1):(N.k*k)
  I.k.tr<-setdiff(1:N.tr,I.k.ts)
  X.k.tr <-X[I.k.tr,]
  Y.k.tr <- Y[I.k.tr]
  set.seed(555)
  model.nn<- nnet (X.k.tr,Y.k.tr,size=15, maxit=10000,linout=T,trace=F)
```

```
  X.k.ts <-X[I.k.ts,]
  Y.k.ts <- Y[I.k.ts]
  Y.k.hat.ts <- predict(model.nn,X.k.ts)
  test.k.MSE <- mean((Y.k.ts-Y.k.hat.ts)^2)
  cv[k]<-test.k.MSE
}


## leave-one-out
loo<-rep(0,N)

for (k in 1:N)
{
  X.k.tr <-X[-k,]
  Y.k.tr <- Y[-k]
  set.seed(555)
  model.nn<- nnet (X.k.tr,Y.k.tr,size=15, maxit=10000,linout=T,trace=F)
  X.k.ts <-X[k,]
  Y.k.ts <- Y[k]
  Y.k.hat.ts <- predict(model.nn,X.k.ts)
  test.k.MSE <- mean((Y.k.ts-Y.k.hat.ts)^2)
  loo[k]<-test.k.MSE
}

## test
I.ts<-(N.tr+1):N
X.ts <-X[I.ts,]
Y.ts <- Y[I.ts]
Y.hat.ts <- predict(model.nn,X.ts)
test.MSE <- mean((Y.ts-Y.hat.ts)^2)

## empirical risk
set.seed(555)
model.nn<- nnet (X.tr,Y.tr,size=15, maxit=10000,linout=T,trace=T)
Y.hat.tr <- predict(model.nn,X.tr)
empirical.MSE <- mean((Y.tr-Y.hat.tr)^2)


print(paste("Empirical MSE=",empirical.MSE))
print(paste("Test MSE=",round(test.MSE,2)))
print(paste("10-fold cross-validation MSE=",round(mean(cv),2)))
print(paste("Leave-one-out cross-validation MSE=",round(mean(loo),2)))
```

•

### 8.1.2 From global modeling to divide-and-conquer

FNN are a typical example of *global* modeling. Global models have essentially two main properties. First, they make the assumption that the relationship between the inputs and the output values can be described by an analytical function over the whole input domain. Second, they solve the problem of learning as a problem of function estimation: given a set of data, they extract the hypothesis which is expected to approximate the best the whole data distribution (Chapter 5).

The *divide-and-conquer* paradigm originates from the idea of relaxing the global

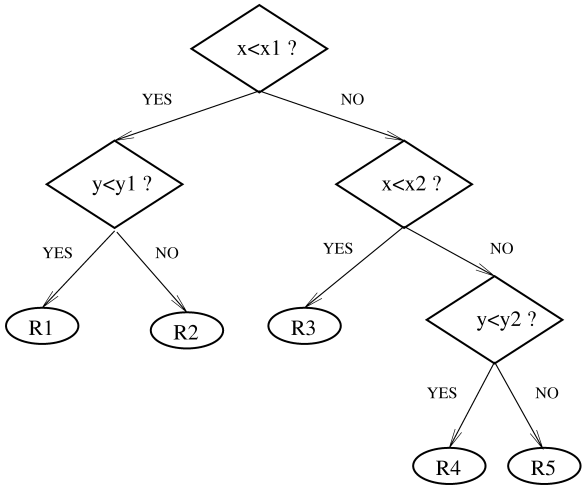



Figure 8.9: A binary decision tree.

modeling assumptions. It attacks a complex problem by dividing it into simpler problems whose solutions can be combined to yield a solution to the original problem. This principle presents two main advantages. The first is that simpler problems can be solved with simpler estimation techniques; in statistics this means to adopt linear techniques, well studied and developed over the years. The second is that the learning method can better adjust to the properties of the available dataset.

The divide-and-conquer idea evolved in two different paradigms: the *modular architectures* and the *local modeling* approach.

Modular techniques replace a global model with a modular architecture where the modules cover different parts of the input space. This is the idea of *operating regimes* which assume a partitioning of the operating range of the system in order to solve modeling and control problems [70]. The following sections will introduce some examples of modular techniques.

### 8.1.3 Classification and Regression Trees

The use of tree-based classification and regression dates back to the work of Morgan and Sonquist in 1963. Since then, methods of tree induction from samples have been an active topic in the machine learning and the statistics community. In machine learning the most representative methods of decision-tree induction are the ID3 [99] and the C4 [100] algorithms. Similar techniques were introduced in statistics by Breiman *et al.* [24], whose methodology is often referred to as the CART (Classification and Regression Trees) algorithm.

A decision tree (see Fig. 8.9) partitions the input space into mutually exclusive regions, each of which is assigned a procedure to characterize its data points (see Fig. 8.10)

The nodes of a decision tree can be classified in internal nodes and terminal nodes. An *internal node* is a decision-making unit that evaluates a decision function to determine which child node to visit next. A *terminal node* or *leaf* has no child nodes and is associated with one of the partitions of the input space. Note that each terminal node has a unique path that leads from the root to itself.

In *classification trees* each terminal node contains a label that indicates the class for the associated input region. In *regression trees* the terminal node contains a model that specifies the input/output mapping for the corresponding input

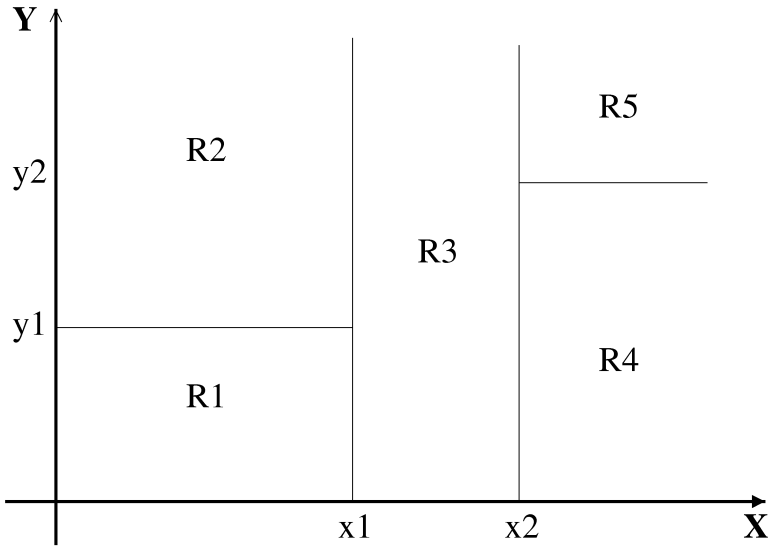



Figure 8.10: Input space partitioning induced on the input space by the binary tree in Fig. 8.9

partition.

Hereafter we will focus only on the regression case. Let $m$ be the number of leaves and $h_j(\cdot, \alpha_j)$ the input/output model associated with the $j^{\text{th}}$ leaf. Once a prediction in a query point $q$ is required, the output evaluation proceeds as follows. First the query is presented to the root node of the decision tree; depending on the result of the associated decision function, the tree will branch to one of the root's children. The procedure is iterated recursively until a leaf is reached and a input/output model is selected. The returned output will be the value $h_j(q, \alpha_j)$.

Consider for example the regression trees in Fig. 8.9, and a query point $q = (x_q, y_q)$ so that $x_q < x1$ and $y_q > y1$. The predicted output will be $y_q = h_2(q, \alpha_2)$ where $\alpha_2$ is the vector of parameters of the model localized in region $R2$.

When the terminal nodes contain only constant models, the input/output mapping results in a combination of several constant-height planes put together with crisp boundaries. In the case of linear terms, the resulting approximator is instead a piecewise linear model.

#### 8.1.3.1 Learning in Regression Trees

#### 8.1.3.2 Parameter identification

A regression tree partitions the input space into mutually exclusive regions. In terms of parametric identification this requires a two-step procedure. First, the training dataset is partitioned into $m$ disjoint sets $D_{Nj}$; second, a local model $h_j(\cdot, \alpha_j)$ is fitted to each subset $D_{Nj}$. The nature of the local model determines the kind of procedure (linear or nonlinear) to be adopted for the parameter identification (see Section 6.6).

#### R implementation

A regression tree with constant local models is implemented by the R library `tree`. The script `tree.R` shows the prediction accuracy for different minimum number of

observations per leaf. (Figure 8.11 and Figure 8.12).

```
## tree
## it studies the impact of the number of leaves on the smoothness (bias/variance) of the pre

rm(list=ls())
source("dopler.R")
library(tree)

N<-2000
D<-dataset.dopler(2000)
plot(D$x,D$y,type="l",main="Dataset",
     xlim=c(0.1,1),ylim=c(-10,10))

d<-data.frame(D$y,D$x)
names(d)<-c("Y","X")

for (number.nodes in seq(10,250,by=10)){

  mod.tree<-tree(Y~.,d,minsize=number.nodes,mindev=0)
  d.ts<-data.frame(D$y.ts,D$x.ts)
  names(d.ts)<-c("Y","X")
  p<-predict(mod.tree,d.ts)

  plot(D$x.ts,D$y.ts,type="l",
       main=paste("Min number points per leaf=",number.nodes),
       xlim=c(0.1,1),ylim=c(-10,10))
  lines(D$x.ts,p,col="red")

}
```

•

#### 8.1.3.3 Structural identification

This section presents a summary of the CART procedure [24] for structural identification in binary regression trees. In this case the structural identification procedure addresses the problem of choosing the optimal partitioning of the input space.

To construct an appropriate decision tree, CART first grows the tree on the basis of the training set, and then prunes the tree back based on a minimum cost-complexity principle. This is an example of the exploratory approach to model generation described in Section 6.7.1.

Let us see in detail the two steps of the procedure:

**Tree growing.** CART makes a succession of splits that partition the training data into disjoint subsets. Starting from the root node that contains the whole dataset, an exhaustive search is performed to find the split that best reduces a certain cost function.

Let us consider a certain node $t$ and let $D(t)$ be the corresponding subset of the original $D_N$. Consider the empirical error of the local model fitting the $N(t)$ data contained in the node $t$ :

$$R_{emp}(t) = \min_{\alpha_t} \sum_{i=1}^{N(t)} L(y_i, h_t(x_i, \alpha_t)) \tag{8.1.4}$$

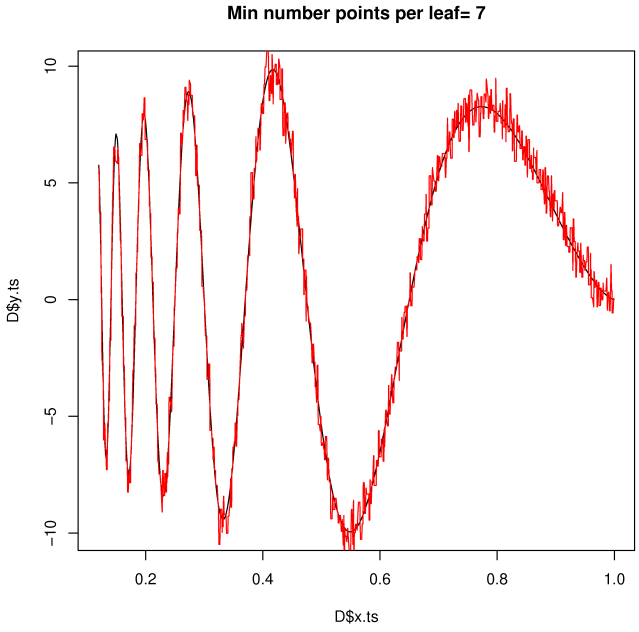



Figure 8.11: Regression tree fitting with a minimum number of points per leaf equal to $s = 7$.

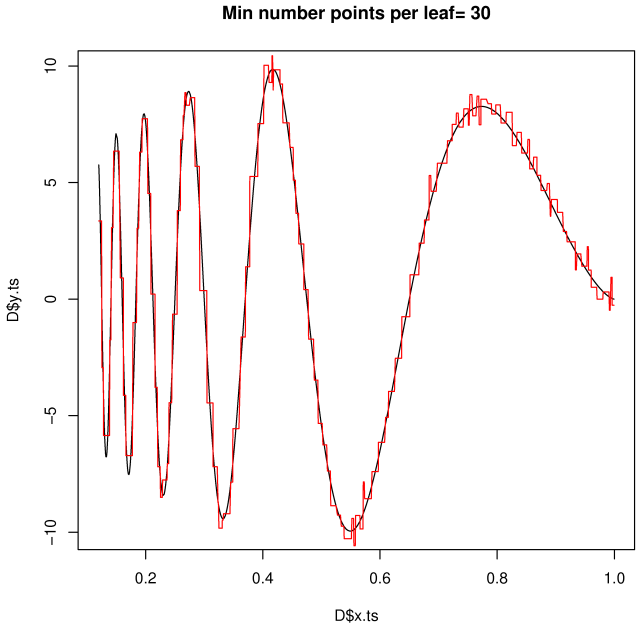



Figure 8.12: Regression tree fitting with a minimum number of points per leaf equal to $s = 30$.

For any possible split $s$ of node $t$ into the two children $t_r$ and $t_l$, we define the quantity

$$\Delta E(s,t) = R_{emp}(t) - (R_{emp}(t_l) + R_{emp}(t_r)) \quad \text{with } N(t_r) + N(t_l) = N(t) \tag{8.1.5}$$

that represents the change in the empirical error due to a further partition of the dataset. The best split is the one that maximizes the decrease $\Delta E$

$$s^* = \arg\max_s \Delta E(s,t) \tag{8.1.6}$$

Once the best split is attained, the dataset is partitioned into the two disjoint subsets of length $N(t_r)$ and $N(t_l)$, respectively. The same method is recursively applied to all the leaves. The procedure terminates either when the error measure associated with a node falls below a certain tolerance level, or when the error reduction $\Delta E$ resulting from further splitting does not exceed a threshold value.

The tree that the growing procedure yields is typically too large and presents a serious risk of overfitting the dataset (Section 5.5). For that reason a pruning procedure is often adopted.

**Tree pruning.** Consider a fully expanded tree $T_{max}$ characterized by $L$ terminal nodes.

Let us introduce a complexity based measure of the tree performance

$$R_\lambda(T) = R_{emp}(T) + \lambda|T| \tag{8.1.7}$$

where $\lambda$ is a parameter that accounts for the tree's complexity and $|T|$ is the number of terminal nodes of the tree $T$. For a fixed $\lambda$ we define with $T(\lambda)$ the tree structure which minimizes the quantity (8.1.7).

The parameter $\lambda$ is gradually increased in order to generate a sequence of tree configurations with decreasing complexity

$$T_L = T_{max} \supset T_{L-1} \supset \cdots \supset T_2 \supset T_1 \tag{8.1.8}$$

where $T_i$ has $i$ terminal nodes. In practice, this requires a sequence of shrinking steps where for each step we select the value of $\lambda$ leading from a tree to a tree of inferior complexity. When we have a tree $T$ the next inferior tree is found by computing for each admissible subtree $T_t \subset T$ the value $\lambda_t$ which makes of it the minimizer of (8.1.7). For a generic subtree $T_t$ this value must satisfy

$$R_{\lambda_t}(T_t) \leq R_{\lambda_t}(T) \tag{8.1.9}$$

that is

$$R_{emp}(T_t) + \lambda_t|T_t| \leq R_{emp}(T) + \lambda_t|T|$$

which means

$$\lambda_t \geq \frac{R_{emp}(T) - R_{emp}(T_t)}{|T| - |T_t|} \tag{8.1.10}$$

Hence, $\lambda_t = \frac{R_{emp}(T) - R_{emp}(T_t)}{|T| - |T_t|}$ makes of $T_t$ the minimizing tree. Therefore we choose among all the admissible subtrees $T_t$ the one with the smallest right-hand term in Eq. (8.1.10). This implies a minimal increase in $\lambda$ toward the next minimizing tree.

At the end of the shrinking process we have a sequence of candidate trees which have to be properly assessed in order to perform the structural selection. As far as validation is concerned, either a procedure of cross-validation or of independent testing can be used. The final structure is then obtained through one of the selection procedures described in Section 6.7.3.

Regression trees are a very easy-to-interpret representation of a nonlinear input/output mapping. However, these methods are characterized by rough discontinuity at the decision boundaries which might bring undesired effects to the overall generalization. Dividing the data by partitioning the input space shows typically small estimator bias but at the cost of an increased variance. This is particularly problematic in high-dimensional spaces where data become sparse. One response to the problem is the adoption of simple local models (e.g. constant or linear). These simple functions minimize variance at the cost of an increased bias.

Another trick is to make use of soft splits, allowing data to lie simultaneously in multiple regions. This is the approach taken by BFN.

### 8.1.4 Basis Function Networks

*Basis Function Networks* (BFN) are a family of modular architectures which are described by the weighted linear combination

$$y = \sum_{j=1}^{m} \rho_j(x) h_j \tag{8.1.11}$$

where the weights are returned by the activations of $m$ local nonlinear *basis functions* $\rho_j$ and where the term $h_j$ is the output of a generic module of the architecture.

The *basis* or *activation* function $\rho_j$ is a function

$$\rho_j : \mathcal{X} \to [0, 1] \tag{8.1.12}$$

usually designed so that its value monotonically decreases towards zero as the input point moves away from its center $c_j$.

The basis function idea arose almost at the same time in different fields and led to similar approaches, often denoted with different names. Examples are the Radial Basis Function in machine learning, the Local Model Networks in system identification and the Neuro-Fuzzy Inference Systems in fuzzy logic. These three architectures are described in the following sections.

### 8.1.5 Radial Basis Functions

A well-known example of basis functions are the *Radial Basis Functions* (RBF) [97]. Each basis function in RBF takes the form of a kernel

$$\rho_j = K(x, c_j, B_j) \tag{8.1.13}$$

where $c_j$ is the center of the kernel and $B_j$ is the bandwidth. An example of kernel functions is illustrated in Fig. 8.13. Other examples of kernel functions are available in Appendix C. Once we define with $\eta_j$ the set $\{c_j, B_j\}$ of parameters of the basis function, we have

$$\rho_j = \rho_j(\cdot, \eta_j) \tag{8.1.14}$$

If the basis $\rho_j$ have localized receptive fields and a limited degree of overlap with their neighbors, the weights $h_j$ in Eq. (8.1.11) can be interpreted as locally piecewise constant models, whose validity for a given input is indicated by the corresponding activation function for a given input.

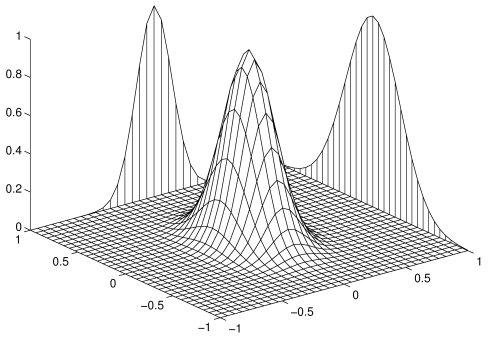



Figure 8.13: A Gaussian kernel function in a two dimensional input space.

### 8.1.6 Local Model Networks

*Local Model Networks* (LMN) were first introduced by Johansen and Foss [70]. They are a generalized form of Basis Function Network in the sense that the constant weights $h_j$ associated with the basis functions are replaced by local models $h_j(\cdot, \alpha_j)$. The typical form of a LMN is then

$$y = \sum_{j=1}^{m} \rho_j(x, \eta_j) h_j(x, \alpha_j) \tag{8.1.15}$$

where the $\rho_j$ are constrained to satisfy

$$\sum_{j=1}^{m} \rho_j(x, \eta_j) = 1 \quad \forall x \in \mathcal{X} \tag{8.1.16}$$

This means that the basis functions form a *partition of unity* [87]. This ensures that every point in the input space has equal weight, so that any variation in the output over the input space is due only to the models $h_j$.

The smooth combination provided by the LMN formalism enables the representation of complex nonlinear mappings on the basis of simpler modules. See the example in Fig. 8.14 which shows the combination in a two dimensional input space of three local linear models whose validity regions is represented by Gaussian basis functions.

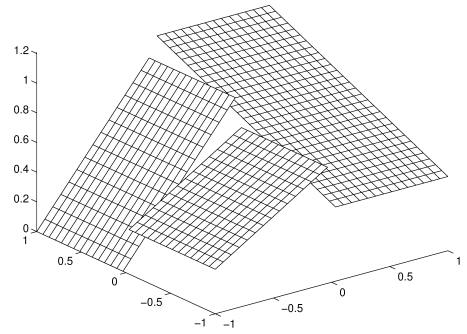
In general, the local models $h_j(\cdot, \alpha)$ in Eq. (8.1.15) can be of any form: linear, nonlinear, physical models or black-box parametric models.

Note that, in the case of local linear models

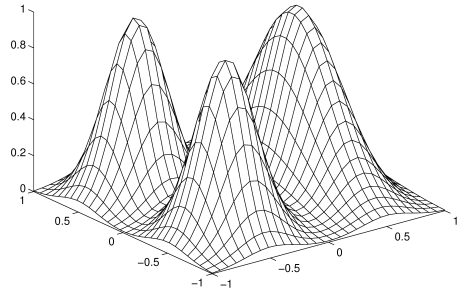
$$h_j(x, \alpha_j) = \sum_{i=1}^{n} a_{ji} x_i + b_j \tag{8.1.17}$$

where the vector of parameters of the local model is $\alpha_j = [a_{j1}, \ldots, a_{jn}, b_j]$ and $x_i$ is the $i^{\text{th}}$ term of the vector $x$, a LMN architecture returns one further information about the input/output phenomenon: the local linear approximation $h_{lin}$ of the input/output mapping about a generic point $x$

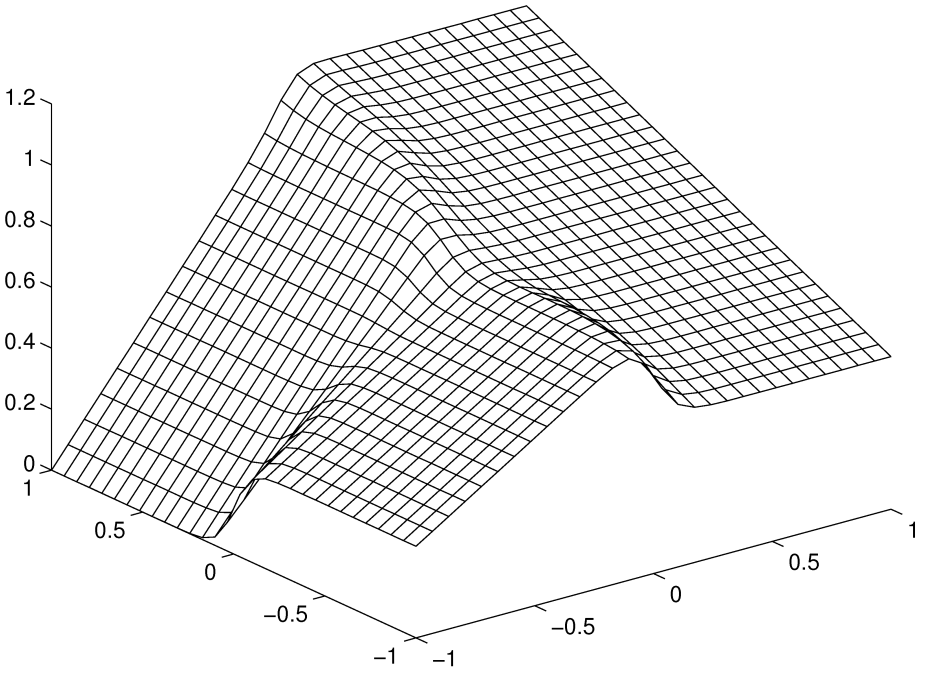
$$h_{lin}(x) = \sum_{j=1}^{m} \rho_j(x, \eta_j) \left( \sum_{i=1}^{n} a_{ji} x_i + b_j \right) \tag{8.1.18}$$




(a)




(b)




(c)

Figure 8.14: A Local Model Network with $m = 3$ local models: the nonlinear input/output approximator in (c) is obtained by combining the three local linear models in (a) according to the three basis functions in (b).

### 8.1.7 Neuro-Fuzzy Inference Systems

Fuzzy modeling consists in describing relationships between variables by means of *if-then* rules, such as

$$\text{If } x \text{ is high then } y \text{ is low}$$

where the linguistic terms, as *high* and *low*, are described by fuzzy sets [123].

The first part of each rule is called the *antecedent* while the second part is called the *consequent*. Depending on the particular form of the consequent proposition, different types of rule-based fuzzy models can be distinguished [8].

Here we will focus on the fuzzy architecture for nonlinear modeling introduced by Takagi and Sugeno [112]. A Takagi-Sugeno (TS) *fuzzy inference system* is a set of $m$ fuzzy if-then rules having the form:

$$\begin{cases} \text{If } x_1 \text{ is } A_{11} \text{ and } x_2 \text{ is } A_{21} \dots \text{ and } x_n \text{ is } A_{n1} \text{ then } y = h_1(x_1, x_2, ..., x_n, \alpha_1) \\ \dots \\ \text{If } x_1 \text{ is } A_{1m} \text{ and } x_2 \text{ is } A_{2m} \dots \text{ and } x_n \text{ is } A_{nm} \text{ then } y = h_m(x_1, x_2, ..., x_n, \alpha_m) \end{cases} \tag{8.1.19}$$

The *antecedent* is defined as a fuzzy AND proposition where $A_{kj}$ is a fuzzy set on the $k^{\text{th}}$ premise variable defined by the membership function $\mu_{kj} \colon \Re \rightarrow [0, 1]$. The *consequent* is a function $h_j(\cdot, \alpha_j)$, $j = 1, \dots, m$, of the input vector $[x_1, x_2, \dots, x_n]$.

By means of the fuzzy sets $A_{kj}$, the input domain is softly partitioned into $m$ regions where the mapping is locally approximated by the models $h_j(\cdot, \alpha_j)$.

If the TS inference system uses the weighted mean criterion to combine the local representations, the model output for a generic query $x$ is computed as

$$y = \frac{\sum_{j=1}^{m} \mu_j(x) h_j(x, \alpha_j)}{\sum_{j=1}^{m} \mu_j(x)} \tag{8.1.20}$$

where $\mu_j$ is the degree of fulfillment of the $j^{\text{th}}$ rule, commonly obtained by

$$\mu_j(x) = \prod_{k=1}^{n} \mu_{kj}(x)$$

This formulation makes of a TS fuzzy system a particular example of LMN where

$$\rho_j(x) = \frac{\mu_j(x)}{\sum_{j=1}^{m} \mu_j(x)} \tag{8.1.21}$$

is the basis function and $h_j(\cdot, \alpha_j)$ is the local model of the LMN architecture.

In a conventional fuzzy approach the membership functions and the consequent models are fixed by the model designer according to a priori knowledge. In many cases this knowledge is not available, however a set of input/output data has been observed. Once we put the components of the fuzzy system (memberships and consequent models) in a parametric form, the TS inference system becomes a parametric model which can be tuned by a learning procedure. In this case the fuzzy system turns into a *Neuro-Fuzzy* approximator [67]. For a thorough introduction to Neuro-Fuzzy architecture see [68] and the references therein. Further work on this subject was presented by the author in [14, 15, 21, 13].

### 8.1.8 Learning in Basis Function Networks

Given the strong similarities between the three instances of BFN discussed above, our discussion on the BFN learning procedure does not distinguish between these approaches.

The learning process in BFN is divided in structural (see Section 6.7) and parametric identification (see Section 6.6). The structural identification aims to find the optimal number and shape of the basis functions $\rho_j(\cdot)$. Once the structure of the network is defined, the parametric identification searches for the optimal set of parameters $\eta_j$ of the basis functions (e.g. center and width in the Gaussian case) and the optimal set of parameters $\alpha_j$ of the local models (e.g. linear coefficients in the case of local linear models).

Hence, the classes of parameters to be identified are two: the parameters of the basis function and the parameters of the local model.

#### 8.1.8.1 Parametric identification: basis functions

The relationship between the model output and the parameters $\eta_j$ of the basis function is typically nonlinear, hence, methods for nonlinear optimization are currently employed. A typical approach consists in decomposing the identification procedure into two steps: first an *initialization* step, which computes the initial location and width of the basis functions, then a *nonlinear optimization* procedure which uses the outcome $\eta_j^{(0)}$ of the previous step as initial value.

Since the methods for nonlinear optimization have already been discussed in Section 6.6.2.2, here we will focus on the different initialization techniques for Basis Function Networks.

One method for placing the centers of the basis functions is to locate them at the interstices of some coarse lattice defined over the input space [25]. If we assume the lattice to be uniform with $d$ divisions along each dimension, and the dimensionality of the input space to be $n$, a uniform lattice requires $d^n$ basis functions. This exponential growth makes the use of such a uniform lattice impractical for high dimensional space.

Moody and Darken [85] suggested a K-means clustering procedure in the input space to position the basis functions. The K-means method, described into detail in Appendix A.2, takes as input the training set and returns $m$ groups of input vectors each parameterized by a center $c_j$ and a width $\sigma_j$. This method generally requires a much smaller number of basis functions than the uniform partition, nevertheless the basis location concerns only that part of the input space actually covered by data. The assumption underlying this method is that similar inputs should produce similar outputs and that these similar input pairs should be bundled together into clusters in the training set. This assumption is reasonable but not necessarily true in real problems. Therefore, the adoption of K-means clustering techniques for supervised learning is essentially an heuristic technique and finding a dataset to which this technique cannot be applied satisfactorily is not uncommon.

An alternative to K-means clustering for initialization has been proposed in the Neuro-Fuzzy literature [8, 9]. The initialization of the architecture is provided by a *hyperellipsoidal fuzzy clustering* procedure (see Appendix A.4). This procedure clusters the data in the input/output domain, obtaining a set of hyperellipsoids which are a preliminary rough representation of the mapping. The parameters of the ellipsoids (eigenvalues) are used to initialize the parameters $\alpha_j$ of the consequent models, while the projection of their barycenters on the input domain determines the initial positions of the membership functions (see Fig. 8.15).

#### 8.1.8.2 Parametric identification: local models

A common approach to the optimization of the parameters $\alpha_j$ of local models is the least-squares method (see Eq. (6.6.2) and (6.6.4)).

If the local models are nonlinear, some nonlinear optimization technique is required (Section 6.6.2.2). Such a procedure is typically computationally expensive

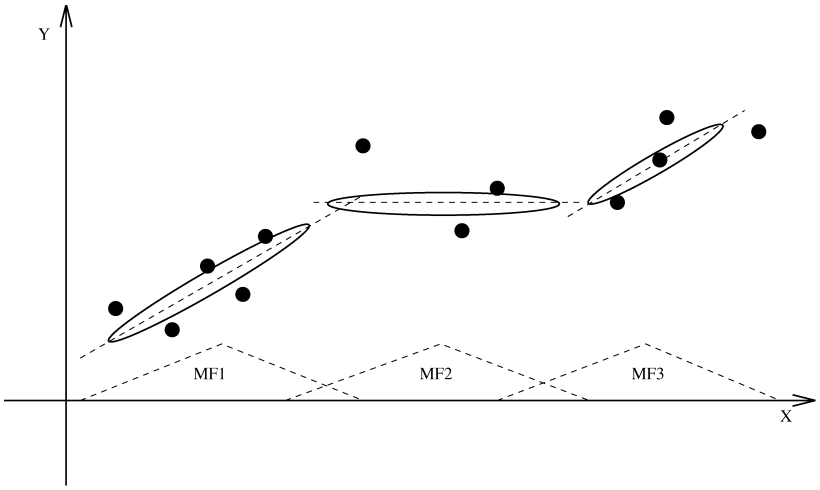



Figure 8.15: The hyperellipsoidal clustering initialization procedure for a single-input single-output mapping. The training samples (dots) are grouped in three ellipsoidal clusters after a procedure of fuzzy clustering in the input/output domain. The projection of the resulting clusters in the input domain ($x$-axis) determines the center and the width of the triangular membership functions.

and does not guarantee the convergence to the global minimum.

However, in the case of local linear models (Eq. 8.1.17), the parametric identification can take advantage of linear techniques. Assume that the local models are linear, i.e.

$$h_j(x, \alpha_j) = h_j(x, \beta_j) = x^T \beta_j \tag{8.1.22}$$

There are two possible variants for the parameter identification [87, 88]:

**Local optimization.** The parameters of each local model are estimated independently.

A weighted least squares optimization criterion can be defined for each local model, where the weighting factor is the current activation of the corresponding basis function. The parameters of each model $h_j(\cdot, \beta_j)$, $j = 1, \ldots, m$, are then estimated using a set of locally weighted estimation criteria

$$J_j(\beta_j) = \frac{1}{N}(y - X\beta_j)^T Q_j (y - X\beta_j) \tag{8.1.23}$$

where $Q_j$ is a $[N \times N]$ diagonal weighting matrix, having as diagonal elements the weights $\rho_j(x_1, \eta_j), \ldots, \rho_j(x_N, \eta_j)$. The weight $\rho_j(x_i, \eta_j)$ represents the relevance of the $i^{\text{th}}$ sample of the training set in the definition of the $j^{\text{th}}$ local model.

The locally weighted least squares estimate $\hat{\beta}_j$ of the local model parameter vector $\beta_j$ is

$$\hat{\beta}_j = (X^T Q_j X)^{-1} X^T Q_j y \tag{8.1.24}$$

**Global optimization.** The parameters of the local models are all estimated at the same time. If the local models are assumed to be linear in the parameters, the optimization is a simple least-squares problem. We get the following regression model:

$$y = \sum_{j=1}^{m} \rho_j(x, \eta_j) x^T \beta_j = \Phi\Theta \tag{8.1.25}$$

where $\Phi$ is a matrix $[N \times (n+1)m]$

$$\Phi = \begin{bmatrix} \Phi_1 \\ \vdots \\ \Phi_N \end{bmatrix} \tag{8.1.26}$$

with

$$\Phi_i = [\rho_1(x_i, \eta_j)x_i^T, \ldots, \rho_m(x_i, \eta_j)x_i^T] \tag{8.1.27}$$

and $\Theta$ is a matrix $[(n+1)m \times 1]$

$$\Theta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} \tag{8.1.28}$$

The least-squares estimate $\hat{\Theta}$ returns the totality of parameters of the local models.

Note that the two approaches differ both in terms of predictive accuracy and final interpretation of the local models. While the first approach aims to obtain local linear models $h_j$ somewhat representative of the local behavior of the target in the region described by $\rho_j$ [88], the second approach disregards any qualitative interpretation by pursuing only a global prediction accuracy of the modular architecture.

#### 8.1.8.3 Structural identification

The structure of a BFN is characterized by many factors: the shape of the basis functions, the number $m$ of modules and the structure of the local models. Here, for simplicity, we will consider a structural identification procedure which deals exclusively with the number $m$ of local models.

The structural identification procedure consists in adapting the number of modules to the complexity of the process underlying the data. According to the process described in Section 6.7, different BFN architectures with different number of models are first generated, then validated and finally selected.

Analogously to Neural Networks and Regression Trees, two are the possible approaches to the generation of BFN architectures:

**Forward:** the number of local models increases from a minimum $m_{min}$ to a maximum value $m_{max}$.

**Backward:** we start with a large number of models and we proceed gradually by merging basis functions. The initial number must be set sufficiently high such that the nonlinearity can be captured accurately enough.

Once a set of BFN architectures has been generated, first a validation measure is used to assess the generalization error of the different architectures and then a selection of the best structure is performed. An example of structural identification of Neuro-Fuzzy Inference Systems based on cross-validation is presented in [14].

Note that BFN structural identification, unlike the parametric procedure described in Section 8.1.8.2, cannot employ any linear validation technique. This is due to the fact that a BFN architecture, even if composed of local linear modules, behaves globally as a nonlinear approximator. The resulting learning procedure is then characterized by an iterative loop over different model structures as illustrated in the flow chart of Fig. 8.16.

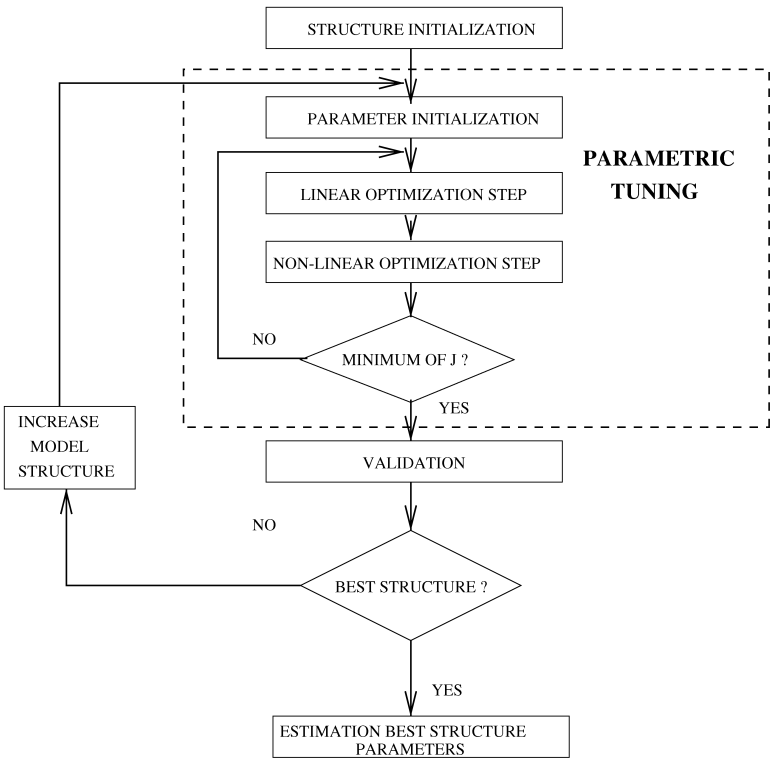



Figure 8.16: Flow-chart of the BFN learning procedure. The learning procedure is made of two nested loops: the inner one (made of a linear and nonlinear step) is the parametric identification loop which minimizes the empirical error $J$, the outer one searches for the model structure which minimizes the validation criterion.

### 8.1.9 From modular techniques to local modeling

Modular techniques are powerful engines but leave still some problems unsolved. While these architectures have efficient parametric identification algorithms, they are inefficient in terms of structural optimization. If the parametric identification takes advantage of the adoption of local linear models, the validation of the global architecture remains a nonlinear problem which can be addressed only by computationally expensive procedures.

The learning problem for modular architectures is still a problem of function estimation formulated in terms of minimization of the empirical risk over the whole training set. The modular configuration makes the minimization simpler but in theoretical terms the problem appears to be at the same level of complexity as in a generic nonlinear estimator.

Once also the constraint of a global optimization is relaxed, the divide-and-conquer idea leads to the local modeling approach.

### 8.1.10 Local modeling

Local modeling is a popular nonparametric technique, which combines excellent theoretical properties with a simple and flexible learning procedure.

This section will focus on the application of local modeling to the regression problem. The idea of local regression as a natural extension of parametric fitting arose independently at different points in time and in different countries in the $19^{\text{th}}$ century. The early literature on smoothing by local fitting focused on one independent variable with equally spaced values. For an historical review of early work on local regression see [29].

The modern view of smoothing by local regression has origins in the 1950's and 1960's in the kernel methods introduced in the density estimation setting . As far as regression is concerned, the first modern works on local regression were proposed by Nadaraya [90] and Watson [120].

#### 8.1.10.1 Nadaraya-Watson estimators

Let $K(x, q, B)$ be a nonnegative *kernel* function that embodies the concept of vicinity. This function depends on the query point $q$, where the prediction of the target value is required, and on a parameter $B \in (0, \infty)$, called *bandwidth*, which represents the radius of the neighborhood. The function $K$ satisfies two conditions:

$$0 \leq K(x, q, B) \leq 1 \tag{8.1.29}$$

$$K(q, q, B) = 1 \tag{8.1.30}$$

For example, in the simplest one-dimensional case (dimension $n = 1$ of the input space) both the rectangular vicinity function (also called *uniform* kernel) (Fig. 8.17)

$$K(x, q, B) = \begin{cases} 1 & \text{if } \|x - q\| < \frac{B}{2} \\ 0 & \text{otherwise} \end{cases} \tag{8.1.31}$$

and the soft threshold vicinity function (Fig. 8.18)

$$K(x, q, B) = \exp\left\{-\frac{(x - q)^2}{B^2}\right\} \tag{8.1.32}$$

satisfy these requirements. Other examples of kernel functions are reported in Appendix C.

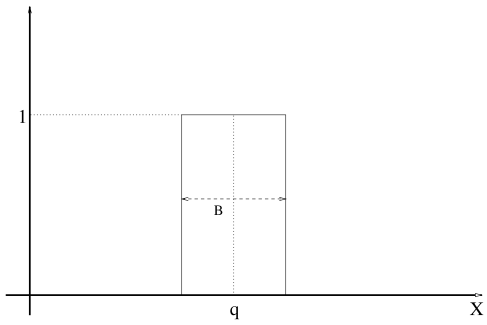



Figure 8.17: Hard-threshold kernel function.

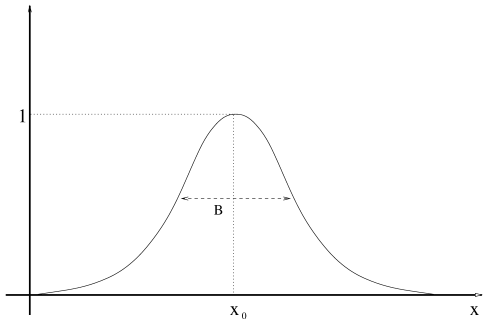



Figure 8.18: Soft-threshold kernel function.

The Nadaraya-Watson kernel regression estimator is given by

$$h(q) = \frac{\sum_{i=1}^{N} K(x_i, q, B) y_i}{\sum_{i=1}^{N} K(x_i, q, B)} \tag{8.1.33}$$

where $N$ is the number of samples in the training set. The idea of kernel estimation is simple. Consider the case of a rectangular kernel in one dimension ($n = 1$). In this case the estimator (8.1.33) is a simple moving average with equal weights: the estimate at point $q$ is the average of observations $y_i$ corresponding to the $x_i$'s belonging to the window $[q - B, q + B]$.

If $B \to \infty$ then the estimator tends to the average $h = \frac{\sum_{i=1}^{N} y_i}{N}$ and thus for mappings $f(\cdot)$ which are far from constant the bias become large.

If $B$ is smaller than the pairwise distance between the sample points $x_i$ then the estimator reproduces the observations $h(x_i) = y_i$. In this extremal case the bias tends to zero at the cost of high variance. In general terms, by increasing $B$ we increase the bias of the estimator, while by reducing $B$ we obtain a larger variance. The optimal choice for $B$ corresponds to an equal balance between bias and variance (Section 5.5).

From a function approximation point of view, the Nadaraya-Watson estimator is a least-squares constant approximator. Suppose we want to approximated locally the unknown function $f(\cdot)$ by a constant $\theta$. The local weighted least-squares estimate is

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{N} w_i (y_i - \theta)^2 = \frac{\sum_{i=1}^{N} w_i y_i}{\sum_{i=1}^{N} w_i} \tag{8.1.34}$$

It follows that the kernel estimator is an example of locally weighted constant

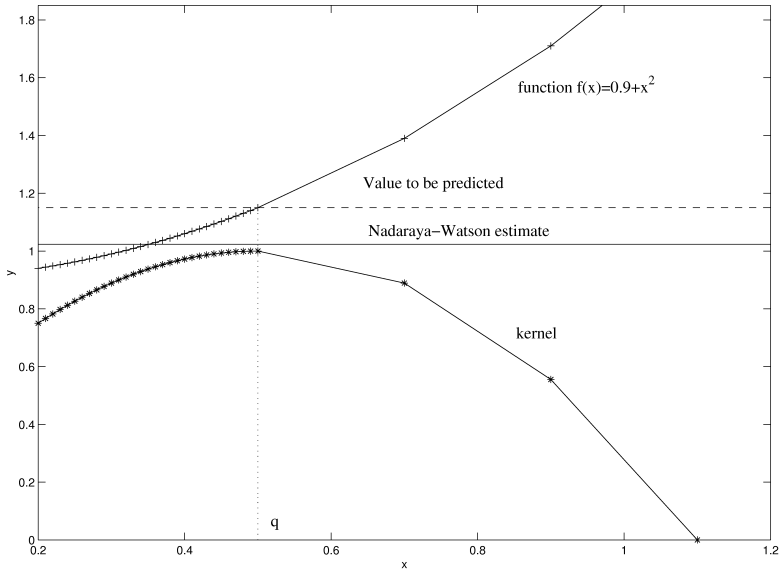



Figure 8.19: Effect of an asymmetric data distribution on the Nadaraya-Watson estimator: the plot reports in the input/output domain the function $f = 0.9 + x^2$ to be estimated, the samples available (crosses), the values of the kernel function (stars), the value to be predicted in $q = 0.5$ (dotted horizontal line) and the value predicted by the NW estimator (solid horizontal line).

approximator with $w_i = K(x_i, q, B)$.

The Nadaraya-Watson estimator suffers from a series of shortcomings: it has large bias particularly in regions where the derivative of the regression function $f(x)$ or of the density $\pi(x)$ is large. Further, it does not adapt easily to nonuniform $\pi(x)$.

An example is given in Fig. 8.19 where the Nadaraya-Watson estimator is used to predict the value of the function $f(x) = 0.9 + x^2$ in $q = 0.5$. Since most of the observations (crosses) are on the left of $q$, the estimate is biased downwards.

A more severe problem is the large bias which occurs when estimating at a boundary region. In Fig. 8.20 we wish to estimate the value of $f(x) = 0.9 + x^2$ at $q = 0.5$. Here the regression function has positive slope and hence the Nadaraya-Watson estimate has substantial positive bias.

#### 8.1.10.2 Higher order local regression

Once the weakness of the local constant approximation was recognized, a more general local regression appeared in the late 1970's [27, 109, 72]. Work on local regression continued throughout the 1980's and 1990's, focusing on the application of smoothing to multidimensional problems [28].

Local regression is an attractive method both from the theoretical and the practical point of view. It adapts easily to various kinds of input distributions $\pi$ (e.g. random, fixed, highly clustered or nearly uniform). See in Fig. 8.21 the local regression estimation in $q = 0.5$ for a function $f(x) = 0.9 + x^2$ and an asymmetric data configuration.

Moreover, there are almost no boundary effects: the bias at the boundary stays at the same order as in the interior, without use of specific boundary kernels (compare Fig. 8.20 and Fig. 8.22).

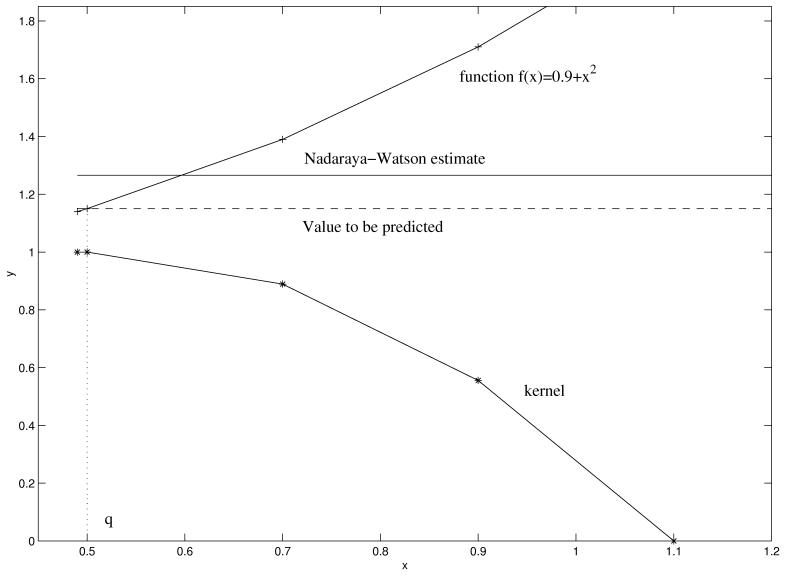



Figure 8.20: Effect of a boundary on the Nadaraya-Watson estimator: the plot reports in the input/output domain the function $f = 0.9 + x^2$ to be estimated, the samples available (crosses), the values of the kernel function (stars), the value to be predicted in $q = 0.5$ (dotted horizontal line) and the value predicted by the NW estimator (solid horizontal line).

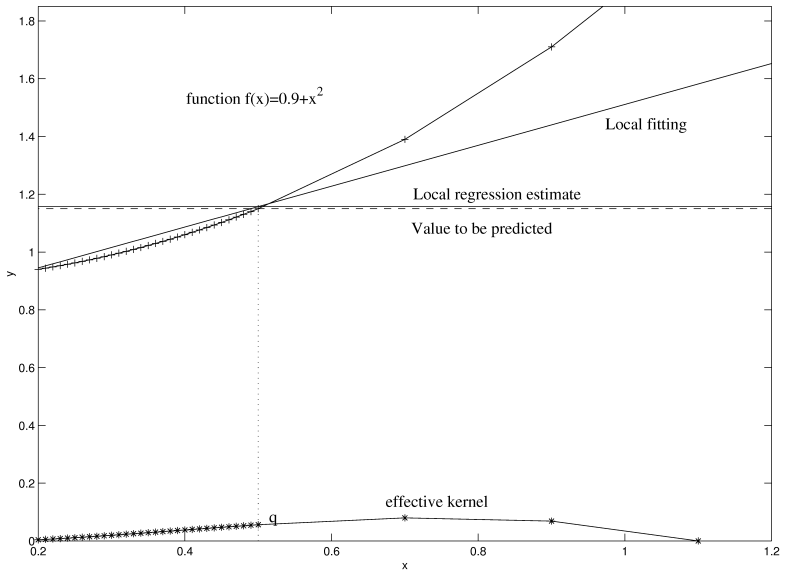



Figure 8.21: Local linear regression in asymmetric data configuration: the plot reports in the input/output domain the function $f = 0.9 + x^2$ to be estimated, the samples available (crosses), the values of the effective kernel (stars), the local linear fitting, the value to be predicted in $q = 0.5$ (dotted horizontal line) and the value predicted by the local regression (solid horizontal line).

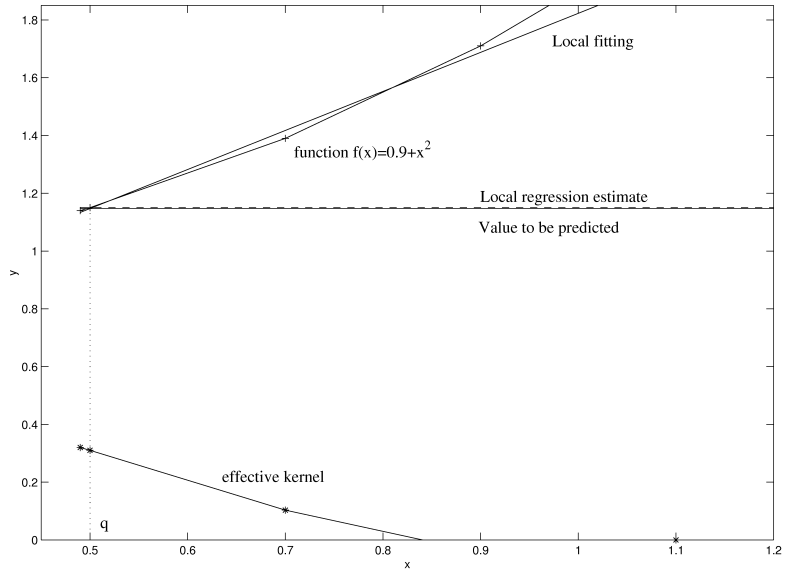



Figure 8.22: Local linear regression in boundary configuration: the plot reports in the input/output domain the function $f = 0.9 + x^2$ to be estimated, the samples available (crosses), the values of the effective kernel (stars), the local linear fitting, the value to be predicted in $q = 0.5$ (dotted horizontal line) and the value predicted by the local regression (solid horizontal line).

#### 8.1.10.3 Parametric identification in local regression

Given two variables $x \in \mathcal{X} \subset \Re^n$ and $y \in \mathcal{Y} \subset \Re$, let us consider the mapping $f\colon \mathbb{R}^n \to \mathbb{R}$, known only through a set of $N$ examples $\{\langle x_i, y_i\rangle\}_{i=1}^N$ obtained as follows:

$$y_i = f(x_i) + w_i, \tag{8.1.35}$$

where $\forall i$, $w_i$ is a random variable such that $E_{\mathbf{w}}[w_i] = 0$ and $E_{\mathbf{w}}[w_i w_j] = 0$, $\forall j \neq i$, and such that $E_{\mathbf{w}}[w_i^m] = \mu_m(x_i)$, $\forall m \geq 2$, where $\mu_m(\cdot)$ is the unknown $m^{\text{th}}$ moment (Eq. (2.3.30)) of the distribution of $w_i$ and is defined as a function of $x_i$. In particular for $m = 2$, the last of the above mentioned properties implies that no assumption of global constant variance (*homoscedasticity*) is made.

The problem of local regression can be stated as the problem of estimating the value that the regression function $f(x) = E_{\mathbf{y}}[\mathbf{y}|x]$ takes for a specific query point $q$, using information pertaining only to a neighborhood of $q$.

By using the Taylor's expansion truncated to the order $p$, a generic smooth regression function $f(\cdot)$ can be approximated by

$$f(x) \approx \sum_{j=0}^{p} \frac{f^{(j)}(q)}{j!}(x-q)^j \tag{8.1.36}$$

for $x$ in a neighborhood of $q$. Given a query point $q$, and under the hypothesis of a local homoscedasticity of $w_i$, the parameter vector $\hat{\beta}$ of a local linear approximation of $f(\cdot)$ in a neighborhood of $q$ can be obtained by solving the *locally weighted regression* (LWR)

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} \left\{ \left(y_i - x_i^T \beta\right)^2 K(x_i, q, B) \right\}, \tag{8.1.37}$$

where $K(\cdot)$ is a kernel function, $B$ is the bandwidth, and a constant value 1 has been appended to each vector $x_i$ in order to consider a constant term in the regression. In matrix notation, the weighted least squares problem (8.1.37) can be written as

$$\hat{\beta} = \arg\min_{\beta}(y - X\beta)^T W(y - X\beta) \tag{8.1.38}$$

where $X$ denotes the $[N \times (n+1)]$ input matrix whose $i^{\text{th}}$ row is $x_i^T$, $y$ is a $[N \times 1]$ vector whose $i^{\text{th}}$ element is $y_i$ and $W$ is a $[N \times N]$ diagonal matrix whose $i^{\text{th}}$ diagonal element is $w_{ii} = \sqrt{K(x_i, q, B)}$. From least-squares theory, the solution of the above stated weighted least squares problem is given by the $[(n+1) \times 1]$ vector:

$$\hat{\beta} = (X^T W^T W X)^{-1} X^T W^T W y = (Z^T Z)^{-1} Z^T v = P Z^T v, \tag{8.1.39}$$

where $Z = WX$, $v = Wy$, and the matrix $X^T W^T W X = Z^T Z$ is assumed to be non-singular so that its inverse $P = (Z^T Z)^{-1}$ is defined.

Once obtained the local linear polynomial approximation, a prediction of $f(q)$, is finally given by:

$$\hat{y}_q = q^T \hat{\beta}. \tag{8.1.40}$$

#### 8.1.10.4 Structural identification in local regression

While the parametric identification in a local regression problem is quite simple and reduces to a weighted least-squares, there are several choices to be made in terms of model structure. These are the most relevant parameters in local structure identification:

- the kernel function $K$,
- the order of the local polynomial,
- the bandwidth parameter,
- the distance function,
- the localized global structural parameters.

In the following sections, we will present in detail the importance of these structural parameters and finally we will discuss the existing methods for tuning and selecting them.

#### 8.1.10.5 The kernel function

Under the assumption that the data to be analyzed are generated by a continuous mapping $f(\cdot)$, we want to consider positive kernel functions $K(\cdot, \cdot, B)$ that are peaked at $x = q$ and that decay smoothly to 0 as the distance between $x$ and $q$ increases. Examples of different kernel functions are reported in Appendix C.

Some considerations can be made on how relevant is the kernel shape for the final accuracy of the prediction. First, it is evident that a smooth weight function results in a smoother estimate. On the other side, for hard-threshold kernels (8.1.31), as $q$ changes, available observations abruptly switch in and out of the smoothing window. Second, it is relevant to have kernel functions with nonzero values on a compact bounded support rather than simply approaching zero for $|x - q| \to \infty$. This allows faster implementations, since points further from the query than the bandwidth can be ignored with no error.

#### 8.1.10.6 The local polynomial order

The choice of the local polynomial degree is a bias/variance trade-off. Generally speaking, a higher degree will generally produce a less biased, but a more variable estimate than a lower degree one.

Some asymptotic results in literature assert that a good practice in local polynomial regression is to adopt a polynomial order which differs of an odd degree from the order of the terms to be estimated [45]. In practice, this means that if the goal of local polynomial regression is to estimate the value of the function in the query point (degree zero in the Taylor expansion (8.1.36)), it is advisable to use orders of odd degree; otherwise, if the purpose is to estimate the derivatives in the query point it is better to fit with even degrees. However, others suggest in practical applications not to rule out any type of degree [29].

In the previous sections, we already introduced some consideration on degree zero fitting. This choice very rarely appears to be the best choice in terms of prediction, even if it presents a strong advantage in computational terms. By using a polynomial degree greater than zero we can typically increase the bandwidth by a large amount without introducing intolerable bias. Despite the increased number of parameters the final result is smoother thanks to an increased neighborhood size.

A degree having an integer value is generally assumed to be the only possible choice for the local order. However, the accuracy of the prediction results to be highly sensitive to discrete changes of the degree.

A possible alternative is *polynomial mixing*, proposed in global parametric fitting by Mallows [81] and in local regression by Cleveland and Loader [29]. Polynomial mixings are polynomials of fractional degree $p = m + c$ where $m$ is an integer and $0 < c < 1$. The mixed fit is a weighted average of the local polynomial fits of degree $m$ and $m + 1$ with weight $1 - c$ for the former and weight $c$ for the latter

$$f_p(\cdot) = (1 - c)f_m(\cdot) + cf_{m+1}(\cdot) \tag{8.1.41}$$

We can choose a single mixing degree for all $x$ or we can use an adaptive method by letting $p$ vary with $x$.

#### 8.1.10.7 The bandwidth

A natural question is how wide the local neighborhood should be so that the local approximation (8.1.36) holds. This is equivalent to asking how large the bandwidth parameter should be in (8.1.31). If we take a small bandwidth $B$, we are able to cope with the eventual nonlinearity of the mapping, that is, in other terms, we keep the modeling bias small . However, since the number of data points falling in this local neighborhood is also small, we cannot average out the noise from the samples and the variance of our prediction will be consequently large (Fig. 8.23).

On the other hand, if the bandwidth is too large, we could smooth excessively the data, then introducing a large modeling bias (Fig. 8.24). In the limit case of an infinite bandwidth, for example, a local linear model turns to be a global linear fitting which, by definition, cannot take into account any type of nonlinearity.

A vast amount of literature has been devoted to the bandwidth selection problem. Various techniques for selecting smoothing parameters have been proposed during the last decades in different setups, mainly in kernel density estimation [71] and kernel regression.

Two are the main strategies for the bandwidth selection:

**Constant bandwidth selection.** The bandwidth $B$ is independent of the training set $D_N$ and the query point $q$.

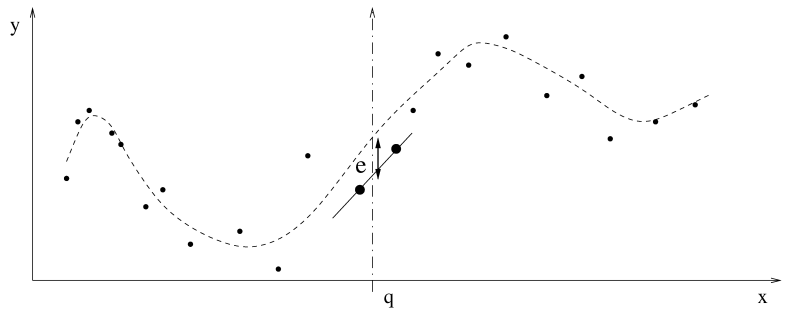



Figure 8.23: Too narrow bandwidth ⇒ overfitting ⇒ large prediction error $e$.

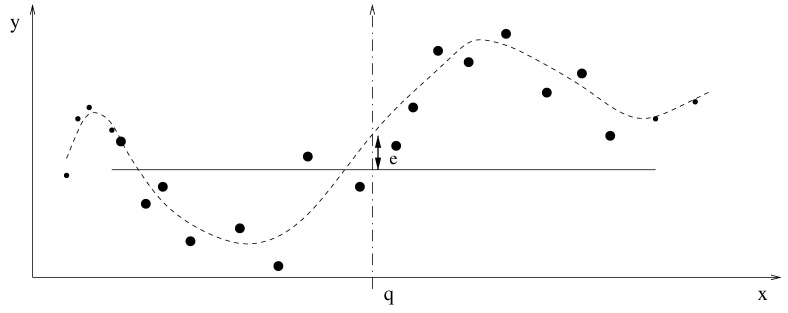



Figure 8.24: Too large bandwidth ⇒ underfitting ⇒ large prediction error $e$)

**Variable bandwidth selection.** The bandwidth is a function $B(D_N)$ of the dataset $D_N$. For a variable bandwidth a further distinction should be made between the local and global approach.

1. A *local variable* bandwidth $B(D_N, q)$ is not only function of the training data $D_N$ but also changes with the query point $q$. An example is the nearest neighbor bandwidth selection where the bandwidth is set to be the distance between the query point and the $k^{\text{th}}$ nearest sample [109].
2. A *global variable bandwidth* is a function $B(D_N)$ of the data set but is the same for all the queries. However, a further degree of distinction should be made between a *point-based* case where the bandwidth $B(x_i)$ changes with the samples in the training set and an *uniform* case where $B$ is the same for all the samples contained in $D_N$.

A constant bandwidth is easy to interpret and can be sufficient if the unknown curve is not too wiggly, i.e. has an high smoothness. Such a bandwidth, however, fails to do a good job when the unknown curve has a rather complicate structure. To capture the complexity of such a curve a variable bandwidth is needed. A variable bandwidth allows for different degrees of smoothing, resulting in a possible reduction of the bias at peaked regions and of the variance at flat regions. Further a variable local bandwidth can adapt to the data distribution, to different level of noise and to changes in the smoothness of the function. Fan and Gijbels [43] argue for point-based in favor of query-based local bandwidth selection mainly for computational efficiency reasons.

#### 8.1.10.8 The distance function

The performance of any local method depends critically on the choice of the distance function $d : \Re^n \times \Re^n \to \Re$. In the following we will define some distance functions for ordered inputs:

**Unweighted Euclidean distance**

$$d(x, q) = \sqrt{\sum_{j=1}^{n} (x_j - q_j)^2} = \sqrt{(x - q)^T (x - q)} \tag{8.1.42}$$

**Weighted Euclidean distance**

$$d(x, q) = \sqrt{(x - q)^T M^T M (x - q)} \tag{8.1.43}$$

The unweighted distance is a particular case of the weighted case for $M$ diagonal with $m_{jj} = 1$.

**Unweighted $L_p$ norm (Minkowski metric)**

$$d(x, q) = \left( \sum_{j=1}^{N} |x_j - q_j| \right)^{\frac{1}{p}} \tag{8.1.44}$$

**Weighted $L_p$ norm** It is computed through the unweighted norm $d(Mx, Mq)$.

It is important to remark that when an entire column of $M$ is zero, all points along the corresponding direction get the same relevance in the distance computation. Also, notice that once the bandwidth is selected, some terms in the matrix $M$ can

be redundant parameters of the local learning procedure. The redundancy can be eliminated by requiring the determinant of $M$ to be one or fixing some element of $M$.

Atkeson *et al.* [7] distinguish between three ways of using distance functions:

**Global distance function.** The same distance is used at all parts of the input space.

**Query-based distance function.** The distance measure is a function of the current query point. Examples are in [108, 61, 50].

**Point-based local distance functions.** Each sample in the training set has an associated distance metric [108]. This is typical of classification problems where each class has an associated distance metric [1, 2].

#### 8.1.10.9 The selection of local parameters

As seen in the previous sections, there are several parameters that affect the accuracy of the local prediction. Generally, they cannot be selected and/or optimized in isolation as the accuracy depends on the whole set of structural choices. At the same time, they do not all play the same role in the determination of the final estimation. It is common belief in local learning literature that the bandwidth and the distance function are the most important parameters. The shape of the weighting function, instead, plays a secondary role.

In the following we will mainly focus on the methods existing for bandwidth selection. They can be classified in

**Rule of thumb methods.** They provide a crude bandwidth selection which in some situations may result sufficient. Examples of rule of thumb is provided in [44] and in [59].

**Data-driven estimation.** It is a selection procedure which estimates the generalization error directly from data. Unlike the previous approach, this method does not rely on the asymptotic expression but it estimates the values directly from the finite data set. To this group belong methods like cross-validation, Mallow's $C_p$, Akaike's AIC and other extensions of methods used in classical parametric modeling.

There are several ways in which data-driven methods can be used for structural identification. Atkeson *et al.* [7] distinguish between

**Global tuning.** The structural parameters are tuned by optimizing a data driven assessment criterion on the whole data set. An example is the General Memory Based Learning (GMBL) described in [86].

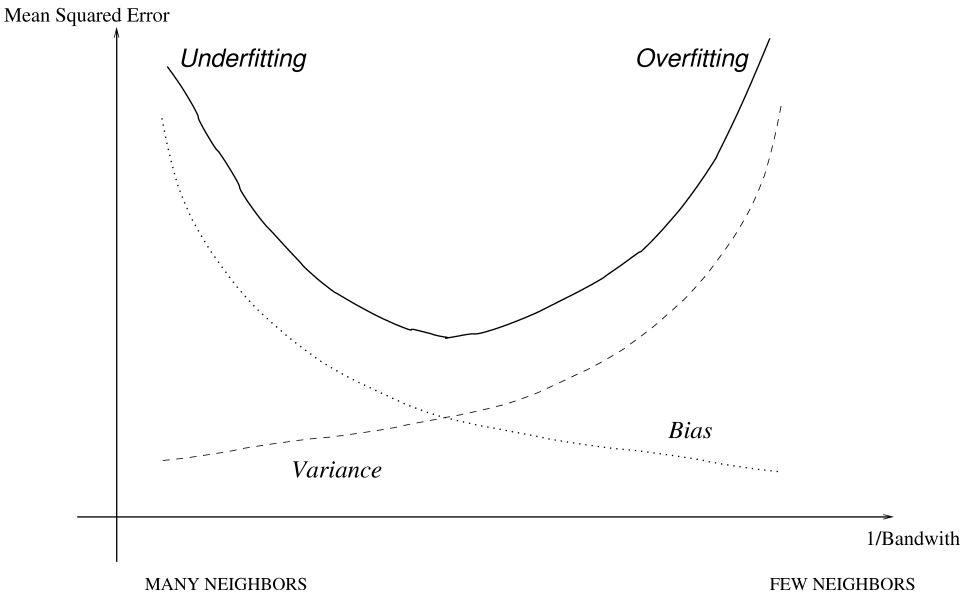
**Query-based local tuning.** The structural parameters are tuned by optimizing a data driven assessment criterion query-by-query. An example is the lazy learning algorithm proposed in [19, 23, 22].

**Point-based local tuning.** A different set of structural parameters is associated to each point of the training set.

#### R implementation

A local linear algorithm for regression is implemented by the R library `lazy`. The script `lazy.R` shows the prediction accuracy in the Dopler dataset for different number of neighbors. (Figure 8.25 and Figure 8.26).

•



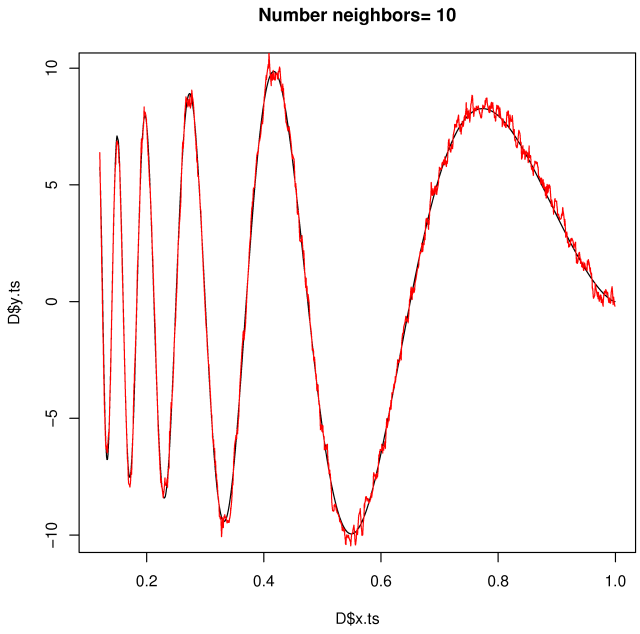




Figure 8.25: Locally linear fitting with a rectangular kernel and a bandwidth made of 10 neighbors.

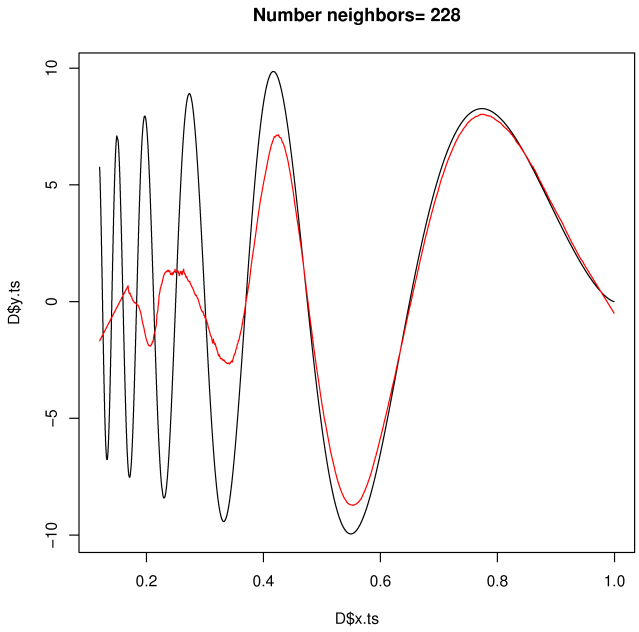



Figure 8.26: Locally linear fitting with a rectangular kernel and a bandwidth made of 228 neighbors.

## 8.2 Nonlinear classification

In Section 7.4, we have shown that optimal classification is possible only if the quantities $\text{Prob}\{\mathbf{y} = c_k|x\}$, $k = 1, \ldots, K$ are known. What happens if this is not the case? Two strategies are generally used.

**Direct estimation via regression techniques.** If the classification problem has $K = 2$ classes and if we denote them by $y = 0$ and $y = 1$

$$E[\mathbf{y}|x] = 1 \cdot \text{Prob}\{\mathbf{y} = 1|x\} + 0 \cdot \text{Prob}\{\mathbf{y} = 0|x\} = \text{Prob}\{\mathbf{y} = 1|x\}$$

Then the classification problem can be put in the form of a regression problem where the output takes value in $\{0, 1\}$. All the regression techniques presented so far can then be used to solve a classification task.

**Density estimation via the Bayes theorem.** Since

$$\text{Prob}\{\mathbf{y} = c_k|x\} = \frac{p(x|\mathbf{y} = c_k)\text{Prob}\{\mathbf{y} = c_k\}}{p(x)}$$

an estimation of $\text{Prob}\{x|\mathbf{y} = c_k\}$ allows an estimation of $\text{Prob}\{\mathbf{y} = c_k|x\}$. Several techniques exist in literature to estimate $\text{Prob}\{x|\mathbf{y} = c_k\}$.

We will present two of them in the following section. The first makes the assumption of conditional independence to make easier the estimation. The second relies on the construction of optimal separating hyperplanes to create convex regions containing set of $x$ points sharing the same class label.

### 8.2.1 Naive Bayes classifier

The Naive Bayes (NB) classifier has shown in some domains a performance comparable to that of neural networks and decision tree learning.

Consider a classification problem with $n$ inputs and a random output variable $\mathbf{y}$ that takes values in the set $\{c_1, \ldots, c_K\}$. The Bayes optimal classifier should return

$$c^*(x) = \arg \max_{j=1,\ldots,K} \text{Prob}\{\mathbf{y} = c_j | x\}$$

We can use the Bayes theorem to rewrite this expression as

$$\begin{aligned} c^*(x) &= \arg \max_{j=1,\ldots,K} \frac{\text{Prob}\{x|\mathbf{y} = c_j\}\,\text{Prob}\{\mathbf{y} = c_j\}}{\text{Prob}\{x\}} \\ &= \arg \max_{j=1,\ldots,K} \text{Prob}\{x|\mathbf{y} = c_j\}\,\text{Prob}\{\mathbf{y} = c_j\} \end{aligned}$$

How to estimate these two terms on the basis of a finite set of data? It is easy to estimate each of the a priori probabilities $\text{Prob}\{\mathbf{y} = c_j\}$ simply by counting the frequency with which each target class occurs in the training set. The estimation of $\text{Prob}\{x|\mathbf{y} = c_j\}$ is much harder. The NB classifier is based on the simplifying assumption that the input values are conditionally independent given the target value:

$$\text{Prob}\{x|\mathbf{y} = c_j\} = \text{Prob}\{x_1, \ldots, x_n|\mathbf{y} = c_j\} = \prod_{h=1}^{n} \text{Prob}\{x_h|\mathbf{y} = c_j\}$$

The NB classification is then

$$c_{NB}(x) = \arg \max_{j=1,\ldots,K} \text{Prob}\{\mathbf{y} = c_j\} \prod_{h=1}^{n} \text{Prob}\{x_h|\mathbf{y} = c_j\}$$

If the inputs $x_h$ are discrete variables the estimation of $\text{Prob}\{x_h|\mathbf{y} = c_j\}$ boils down to the counting of the frequencies of the occurrences of the different values of $x_h$ for a given class $c_j$.

**Example**

| Obs | G1 | G2 | G3 | G |
|---|---|---|---|---|
| 1 | P.LOW | P.HIGH | N.HIGH | P.HIGH |
| 2 | N.LOW | P.HIGH | P.HIGH | N.HIGH |
| 3 | P.LOW | P.LOW | N.LOW | P.LOW |
| 4 | P.HIGH | P.HIGH | N.HIGH | P.HIGH |
| 5 | N.LOW | P.HIGH | N.LOW | P.LOW |
| 6 | N.HIGH | N.LOW | P.LOW | N.LOW |
| 7 | P.LOW | N.LOW | N.HIGH | P.LOW |
| 8 | P.LOW | N.HIGH | N.LOW | P.LOW |
| 9 | P.HIGH | P.LOW | P.LOW | N.LOW |
| 10 | P.HIGH | P.LOW | P.LOW | P.LOW |

Let us compute the NB classification for the query {G1=N.LOW G2= N.HIGH G3=N.LOW Since

$\text{Prob}\{\mathbf{y} = P.HIGH\} = 2/10, \quad \text{Prob}\{\mathbf{y} = P.LOW\} = 5/10$

$\text{Prob}\{\mathbf{y} = N.HIGH\} = 1/10, \quad \text{Prob}\{\mathbf{y} = N.LOW\} = 2/10$

$\text{Prob}\{G1 = N.LOW|\mathbf{y} = P.HIGH\} = 0/2, \quad \text{Prob}\{G1 = N.LOW|\mathbf{y} = P.LOW\} = 1/5$

$\text{Prob}\{G1 = N.LOW|\mathbf{y} = N.HIGH\} = 1/1, \quad \text{Prob}\{G1 = N.LOW|\mathbf{y} = N.LOW\} = 0/2$

$\text{Prob}\{G2 = N.HIGH|\mathbf{y} = P.HIGH\} = 0/2, \quad \text{Prob}\{G2 = N.HIGH|\mathbf{y} = P.LOW\} = 1/5$

$\text{Prob}\{G2 = N.HIGH|\mathbf{y} = N.HIGH\} = 0/1, \quad \text{Prob}\{G2 = N.HIGH|\mathbf{y} = N.LOW\} = 0/2$

$\text{Prob}\{G3 = N.LOW|\mathbf{y} = P.HIGH\} = 0/2, \quad \text{Prob}\{G3 = N.LOW|\mathbf{y} = P.LOW\} = 3/5$

$\text{Prob}\{G3 = N.LOW|\mathbf{y} = N.HIGH\} = 0/1, \quad \text{Prob}\{G3 = N.LOW|\mathbf{y} = N.LOW\} = 0/2$

it follows that

$$c_{NB}(x) = \arg \max_{P.H,P.L,N.H,N.L} \{2/10 * 0 * 0 * 0, 5/10 * 1/5 * 1/5 * 3/5, 1/10 * 1 * 0 * 1, 2/10 * 0 * 0 * 0\}$$
$$== P.LOW$$

•

### 8.2.2 SVM for nonlinear classification

The extension of the Support Vector (SV) approach to nonlinear classification relies on the transformation of the input variables and the possibility of effectively adapting the SVM procedure to a transformed input spaces.

The idea of transforming the input space by using basis functions is an intuitive manner of extending linear techniques to a nonlinear setting.

Consider for example an input output regression problem where $x \in \mathcal{X} \subset \mathbb{R}^n$. Let us define $m$ new transformed variables $z_j = z_j(x)$, $j = 1, \ldots, m$, where $z_j(\cdot)$ is a pre-defined nonlinear transformation (e.g. $z_j(x) = \log x_1 + \log x_2$). This is equivalent to mapping the input space $\mathcal{X}$ into a new space, also known as *feature space*, $\mathcal{Z} = \{z = z(x) | x \in \mathcal{X}\}$. Note that, if $m < n$, this transformation boils down to a dimensionality reduction and it is an example of feature selection (Section **??**).

Let us now fit a linear model $y = \sum_{j=1}^{m} \beta_m z_m$ to the training data in the new input space $z \in \mathbb{R}^m$. By doing this, we carry out a nonlinear fitting of data simply by using a conventional linear technique.

This procedure can be adopted for every learning procedure. However, it is still more worthy to be used in a SV framework. Before discussing it, we introduce the notion of dot-product kernel.

**Definition 2.1** (Dot-product kernel)**.** A dot-product kernel is a function K, such that for all $x_i, x_j \in \mathcal{X}$

$$K(x_i, x_j) = \langle z(x_i), z(x_j) \rangle \tag{8.2.45}$$

where $\langle z_1, z_2 \rangle = z_1^T z_2$ stands for the inner product and $z(\cdot)$ is the mapping from the original to the feature space $\mathcal{Z}$.

Let us suppose now that we want to perform a binary classification by SVM in a transformed space $z \in \mathcal{Z}$. For the sake of simplicity we will consider a separable case. The parametric identification step requires the solution of a quadratic programming problem in the space $\mathcal{Z}$

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k z_i^T z_k = \tag{8.2.46}$$

$$= \max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \langle z_i, z_k \rangle = \tag{8.2.47}$$

$$= \max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k K(x_i, x_k) \tag{8.2.48}$$

$$\text{subject to } 0 = \sum_{i=1}^{N} \alpha_i y_i, \tag{8.2.49}$$

$$\alpha_i \geq 0, \quad i = 1, \ldots, N \tag{8.2.50}$$

What is interesting is that the resolution of this problem differs from the linear one (Equation (**??**)) by the replacement of the quantities $\langle x_i, x_k \rangle$ with $\langle z_i, z_k \rangle = K(x_i, x_k)$.

This means that, whatever the feature transformation $z(x)$ and whatever the dimensionality $m$, the SVM computation requires only the availability of the Gram matrix in the feature space, also referred to as the *kernel matrix* $K$. What is interesting, is that once we know how to derive the kernel matrix we do not even need to know the underlying feature transformation function $z(x)$.

The use of a kernel function is an attractive computational short-cut. In practice, the approach consists in defining a kernel function directly, hence implicitly defining the feature space.

# Chapter 9

# Model averaging approaches

All the techniques presented so far require a model selection procedure where different model structures are assessed and compared in order to attain the best representation of the data. In model selection the winner-takes-all approach is intuitively the approach which should work the best. However, recent results in machine learning show that the performance of the final model can be improved not by choosing the model structure which is expected to predict the best but by creating a model whose output is the combination of the output of models having different structures. The reason is that, in fact, every hypothesis $h(\cdot, \alpha_N)$ is only an estimate of the real target and, like any estimate, is affected by a bias and a variance term. The theoretical results of Section 3.10 show that a variance reduction can be obtained by simply combining uncorrelated estimators. This simple idea underlies some of the most effective techniques recently proposed in machine learning. This chapter will sketch some of them.

## 9.1 Stacked regression

Suppose we have $m$ distinct predictors $h_j(\cdot, \alpha_N)$, $j = 1, \ldots, m$ obtained from a given training set $D_N$. For example a predictor could be a linear model fit on some subset of the variables, a second one a neural network and a third one a regression tree. The idea of averaging models is to design an average estimator

$$\sum_{j=1}^{m} \beta_j h_j(\cdot, \alpha_N)$$

by linear combination which is expected to be more accurate than each of the estimators taken individually.

A simple way to estimate the weights $\hat{\beta}_j$ is to perform a least-squares regression of the output $y$ on the $m$ inputs $h_j(\cdot, \alpha_N)$. The training set for this regression is then made by $D_N = \{h_i, y_i\}$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad H = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_N \end{bmatrix} = \begin{bmatrix} h_1(x_1, \alpha_N) & h_2(x_1, \alpha_N) & \ldots & h_m(x_1, \alpha_N) \\ h_1(x_2, \alpha_N) & h_2(x_2, \alpha_N) & \ldots & h_m(x_2, \alpha_N) \\ \vdots & \vdots & \vdots & \vdots \\ h_1(x_N, \alpha_N) & h_2(x_N, \alpha_N) & \ldots & h_m(x_N, \alpha_N) \end{bmatrix}$$

where $h_i$, $i = 1, \ldots, N$ is a vector of $m$ terms.

Once computed the least-squares solution $\hat{\beta}$ the combined estimator is

$$h_{\text{cm}}(x) = \sum_{j=1}^{m} \hat{\beta}_j h_j(x, \alpha_N)$$

In spite of its simplicity, the least-squares approach might produce poor results since it does not take into account the correlation existing among the $h_j$ and induced by the fact that all of them are estimated on the same training set $D_N$.

Wolpert (1992) presented an interesting idea, called *s*tacked generalization for combining estimators without suffering of the correlation problem. This proposal was translated in statistical language by Breiman, in 1993, who introduced the *s*tacked regression principle.

The idea consists in estimating the $m$ parameters $\hat{\beta}_j$ by solving the following optimization task

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} \left( y_i - \sum_{j=1}^{m} \beta_j h_j^{(-i)}(x_i) \right)^2$$

where $h_j^{(-i)}(x_i)$ is the leave-one-out estimate (6.7.2.3) of the $j$th model.

In other terms the parameters are obtained by performing a least-squares regression of the output $y$ on the $m$ inputs $h_j(\cdot, \alpha_N^{(-i)})$. The training set for this regression is then made by $D_N = \{h_i^-, y_i\}$, $i = 1, \ldots, N$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad H = \begin{bmatrix} h_1^- \\ h_2^- \\ \vdots \\ h_N^- \end{bmatrix} =$$

$$\begin{bmatrix} h_1(x_1, \alpha_N^{(-1)}) & h_2(x_1, \alpha_N^{(-1)}) & \ldots & h_m(x_1, \alpha_N^{(-1)}) \\ h_1(x_2, \alpha_N^{(-2)}) & h_2(x_2, \alpha_N^{(-2)}) & \ldots & h_m(x_2, \alpha_N^{(-2)}) \\ \vdots & \vdots & \vdots & \vdots \\ h_1(x_N, \alpha_N^{(-N)}) & h_2(x_N, \alpha_N^{(-N)}) & \ldots & h_m(x_N, \alpha_N^{(-N)}) \end{bmatrix}$$

where $h_j(x_i, \alpha_N^{(-i)})$ is the predicted outcome in $x_i$ of the $j$th model trained on $D_N$ with the $i$th sample $(x_i, y_i)$ set aside.

By using the cross-validated predictions $h_j(x_i, \alpha_N^{(-i)})$ stacked regression avoids giving unfairly high weight to models with higher complexity. It was shown by Breiman, that the performance of the stacked regressor improves when the coefficients $\hat{\beta}$ are constrained to be non negative. There is a close connection between stacking and winner-takes-all model selection. If we restrict the minimization to weight vectors $w$ that have one unit weight and the rest zero, this leads to the model choice returned by the winner-takes-all based on the leave-one-out. Rather than choose a single model, stacking combines them with estimated optimal weights. This will often lead to better prediction, but less interpretability than the choice of only one of the $m$ models.

## 9.2 Bagging

A learning algorithm is informally called *unstable* if *small* changes in the training data lead to significantly different models and relatively *large* changes in accuracy. Unstable learners can have low bias but have typically high variance. Unstable

methods can have their accuracy improved by *perturbing* (i.e. generating multiple versions of the predictor by perturbing the training set or learning method) and *combining*. Breiman calls these techniques P&C methods.

Combining multiple estimator is a variance reduction technique. The *bagging* technique is a P&C technique which aims to improve accuracy for unstable learners by averaging over such discontinuities. The philosophy of bagging is to improve the accuracy by reducing the variance.

Consider a dataset $D_N$ and a learning procedure to build an hypothesis $\alpha_N$ from $D_N$. The idea of **bagging** or **bootstrap aggregating** is to imitate the stochastic process underlying the realization of $D_N$. A set of $B$ repeated bootstrap samples $D_N^{(b)}$, $b = 1, \ldots, B$ are taken from $D_N$. A model $\alpha_N^{(b)}$ is built for each $D_N^{(b)}$. A final predictor is built by aggregating the $B$ models $\alpha_N^{(b)}$. In the regression case the bagging predictor is

$$h_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} h(x, \alpha_N^{(b)})$$

In the classification case a majority vote is used.

### R script

The R script `bagging.R` shows the efficacy of bagging as a remedy against overfitting.

Consider a dataset $D_N = \{x_i, y_i\}$, $i = 1, \ldots, N$ of $N = 100$ i.i.d. normally distributed inputs $\mathbf{x} \sim \mathcal{N}([0, 0, 0], I)$. Suppose that $\mathbf{y}$ is linked to $\mathbf{x}$ by the input/output relation

$$y = x_1^2 + 4 \log(|x_2|) + 5x_3 + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 0.25)$ represents the noise. Let us train a single-hidden-layer neural network with $s = 25$ hidden neurons on the training set (Section 8.1.1). The prediction accuracy on the test set ($N_{\text{ts}} = 100$) is $\widehat{\text{MISE}}_{\text{ts}} = 70.86$. Let us apply a bagging combination with $B = 50$ (R-file. The prediction accuracy on the test set of the bagging predictor is $\widehat{\text{MISE}}_{\text{ts}} = 6.7$. This shows that the bagging combination reduces the overfitting of the single neural network. Below there is the histogram of the $\widehat{\text{MISE}}_{\text{ts}}$ accuracy of each bootstrap repetition. We can see in Figure (9.1) that the performance of the bagging predictor is much better than the average performance.

•

Tests on real and simulated datasets showed that bagging can give a substantial gain in accuracy. The vital element is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy. On the other hand it can slightly degrade the performance of stable procedures. There is a cross-over point between instability and stability at which bagging stops improving.

Bagging demands the repetition of $B$ estimations of $h(\cdot, \alpha_N^{(b)})$ but avoids the use of expensive validation techniques (e.g. cross-validation). An open question, as in bootstrap, is to decide how many bootstrap replicates to carry out. In his experiments Breiman suggests that $B \approx 50$ is a reasonable figure.

Bagging is an ideal procedure for parallel computing. Each estimation of $h(\cdot, \alpha_N^{(b)})$, $b = 1, \ldots, B$ can proceed independently of the others. At the same time bagging is a relatively easy way to improve a existing method. It simply needs adding

1. a loop that selects the bootstrap sample and sends it to the learning machine and

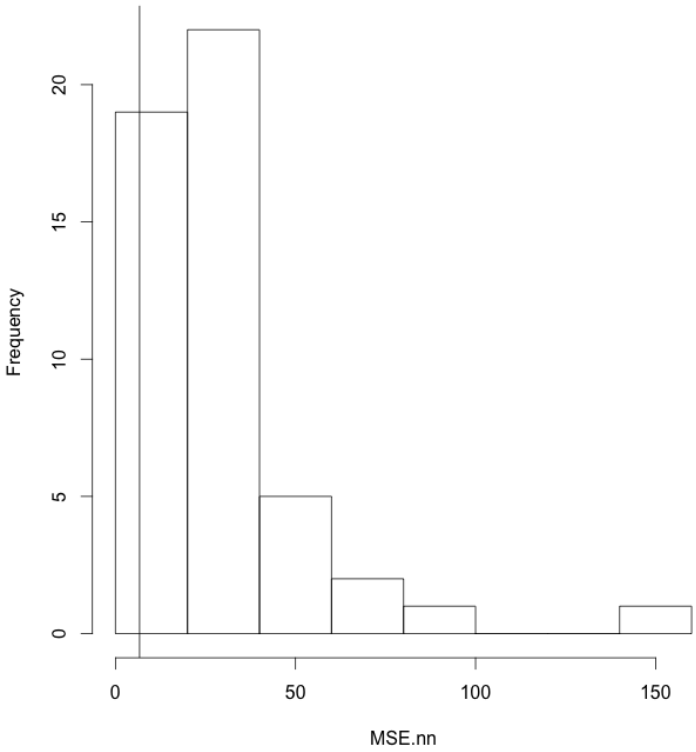



Figure 9.1: Histogram of misclassification rates of resampled trees: the vertical line represents the misclassification rate of the bagging predictor.

2. a back end that does the aggregation.

Note however that if the original learning machine has an interpretable structure (e.g. classification tree) this is lost for the sake of increased accuracy.

## 9.3 Boosting

Boosting is one of the most powerful learning ideas introduced in the last ten years.

Boosting is a general method which attempts to *boost* the accuracy of any given learning algorithm. It was originally designed for classification problems but it can profitably be extended to regression as well. Boosting [49, 106] encompasses a family of methods. The focus of boosting methods is to produce a series of "weak" learners in order to produce a powerful combination. A *weak learner* is a learner that has accuracy only slightly better than chance.

The training set used for each member of the series is chosen based on the performance of the earlier classifier(s) in the series. Examples that are incorrectly predicted by previous classifiers in the series are chosen more often than examples that were correctly predicted.

Thus Boosting attempts to produce new classifiers that are better able to predict examples for which the current ensemble's performance is poor. Unlike Bagging, the resampling of the training set is **dependent** on the performance of the earlier classifiers. The two most important types of boosting algorithms are the Ada Boost (Adaptive Boosting) algorithm (Freund, Schapire, 1997) and the Arcing algorithm (Breiman, 1996).

### 9.3.1 The Ada Boost algorithm

Consider a binary classification problem where the output take values in $\{-1, 1\}$. Let $D_N$ be the training set. A classifier is a predictor $h(\cdot)$ which given an input $x$, produces a prediction taking one of the values $\{-1, 1\}$. A *weak classifier* is one whose misclassification error rate is only slightly better than random guessing.

The purpose of boosting is to sequentially apply the weak classification algorithm to repeatedly modified versions of the data, thereby producing a sequence of classifiers $h_j(\cdot)$, $j = 1, \ldots, m$. The predictions of the $m$ *weak* classifiers are then combined through a **weighted majority vote** to produce he final prediction

$$h_{\text{boo}} = \text{sign}\left(\sum_{j=1}^{m} \alpha_j h_j(x, \alpha_N)\right)$$

The weights $\alpha_j$ of the different classifiers are computed by the algorithm. The idea is to give higher influence to the more accurate classifiers in the sequence. At each step, the boosting algorithm samples $N$ times from a distribution $w$ on the training set which put a weight $w_i$ on each sample $(x_i, y_i)$, $i = 1, \ldots, N$ of $D_N$ Initially the weights are all set to $w_i = 1/N$ so that the first step simply trains the classifier in the standard manner. For each successive iteration $j = 1, \ldots, m$ the probability weights are individually modified and the classification algorithm is re-applied to the resampled training set.

At the generic $j$th step the observations that were misclassified by the classifier $h_{j-1}(\cdot)$ trained at the previous step, have their weights $w_i$ increased, whereas the weights are decreased for those that were classified correctly. The rationale of the approach is that, as the iterations proceed, observations that are difficult to classify receive ever-increasing influence and the classifier is forced to concentrate on them. Note the presence in the algorithm of **two types of weights**: the weights $\alpha_j$,

$j = 1, \ldots, m$ that measure the importance of the classifiers and the weights $w_i$, $i = 1, \ldots, N$ of the samples.

Weak learners are added until some desired low training error has been achieved. This is the algorithm in detail:

1. Initialize the observation weights $w_i = 1/N$, $i = 1, \ldots, N$.
2. For $j = 1$ to $m$:
   (a) Fit a classifier $h_j(\cdot)$ to the training data obtained by resampling $D_N$ using weights $w_i$.
   (b) Compute the misclassification error on the training set
   $$\widehat{\text{MME}}_{\text{emp}}^{(j)} = \frac{\sum_{i=1}^{N} w_i I(y_i \neq h_j(x_i))}{\sum_{i=1}^{N} w_i}$$
   (c) Compute
   $$\alpha_j = \log((1 - \widehat{\text{MME}}_{\text{emp}}^{(j)})/\widehat{\text{MME}}_{\text{emp}}^{(j)})$$
   Note that $\alpha_j > 0$ if $\widehat{\text{MME}}_{\text{emp}}^{(j)} \leq 1/2$ (otherwise we stop or we restart) and that $\alpha_j$ gets larger as $\widehat{\text{MME}}_{\text{emp}}^{(j)}$ gets smaller.
3. (d) For $i = 1, \ldots, N$ set
   $$w_i \leftarrow w_i \begin{cases} \exp[-\alpha_j] & \text{if correctly classified} \\ \exp[\alpha_j] & \text{if incorrectly classified} \end{cases}$$
   (e) The weights are normalized to ensure that $w_i$ represents a true distribution.
4. Output of the weighted majority vote
   $$h_{\text{boo}} = \text{sign}\left(\sum_{j=1}^{m} \alpha_j h_j(x, \alpha_N)\right)$$

**R script**

The R script `boosting.R` tests the performance of the Ada Boost algorithm in a classification task. Consider the medical dataset *Pima* obtained by a statistical survey on women of Pima Indian heritage. This dataset reports the presence of diabetes in Pima Indian women together with other clinical measures (blood pressure, insulin, age,...). The classification task is to predict the presence of diabetes as a function of the clinical measures. We consider a training set of $N = 40$ and a test set of 160 samples. The classifier is a simple classification tree which returns a misclassification rate $\widehat{\text{MME}}_{\text{ts}} = 0.36$. We use a boosting procedure with $m = 15$ to improve the performance of the week classifier. The misclassification rate of the boosted classifier is $\widehat{\text{MME}}_{\text{ts}} = 0.3$.

```
## script boosting.R

library(class)
library(mlbench)
library(tree)
data(Pima.tr)
```

```
N.tr<-40
N<-nrow(Pima.tr)
Pima.tr$type<-2*as.integer(Pima.tr$type=='Yes')-1
train<-Pima.tr[1:N.tr,]
w<-rep(1/N.tr,N.tr)

test<-Pima.tr[(N.tr+1):N,]

m<-15

misc<-rep(NA,m);
alpha<-rep(NA,m)
set.seed(555)
tree.model <-tree(type ~ npreg+glu+bp+skin+bmi+ped+age, train,
                  control=tree.control(N.tr,mincut=10),weights=w*N.tr)
pred <- sign(predict(tree.model,test))
misc.tree <- sum(as.integer(test$type != sign(pred)))/length(pred)


pred.test<-rep(0,N-N.tr)
for (j in 1:m)
{
  set.seed(555)
  I<-sample(seq(1,N.tr),prob=w,replace=TRUE)
  tree.model <-tree(type ~ npreg+glu+bp+skin+bmi+ped+age, train[I,],control=tree.control(N.tr,mi

  pred.train <-sign(predict(tree.model,train))

  misc[j] <- sum(w*as.integer(train$type != pred.train))/sum(w)
  alpha[j]<-log((1-misc[j])/misc[j])
  w<- w*exp(alpha[j]*as.integer(train$type != pred.train))
  pred.test<-pred.test+alpha[j]*predict(tree.model,test)

  if (misc[j]>=0.49)
    w<-rep(1/N.tr,N.tr)
}


misc.boosting <- sum(test$type != sign(pred.test))/length(pred.test)


cat("Misclassification single tree=",misc.tree,"\n")
cat("Misclassification boosting=",misc.boosting,"\n")
```

•

Boosting has its roots in a theoretical framework for studying machine learning called the PAC learning model. Freund and Scapire proved that the empirical error of the final hypothesis $h_{\text{boo}}$ is at most

$$\prod_{j=1}^{m}\left[2\sqrt{\widehat{\text{MME}}_{\text{emp}}^{(j)}*(1-\widehat{\text{MME}}_{\text{emp}}^{(j)})}\right]$$

They showed also how to bound the generalization error.

### 9.3.2 The arcing algorithm

This algorithm was proposed as a modification of the original Ada Boost algorithms by Breiman. It is based on the idea that the success of boosting is related to the adaptive resampling property where increasing weight is placed on those samples more frequently misclassified. ARCing stays for *Adaptive Resampling and Combining*. The complex updating equations of Ada Boost are replaced by much simpler formulations. The final classifier is obtained by unweighted voting

This is the ARCing algorithm in detail:

1. Initialize the observation weights $w_i = 1/N$, $i = 1, \ldots, N$.
2. For $j = 1$ to $m$:
   (a) Fit a classifier $h_j$ to the training data obtained by resampling $D_N$ using weights $w_i$.
   (b) Let $e_i$ the number of misclassifications of the $i$th sample by the $j$ classifiers $h_1, \ldots, h_j$.
   (c) The updated weights are defined by

$$w_i = \frac{1 + e_i^4}{\sum_{i=1}^{N}(1 + e_i^4)}$$

3. The output is obtained by unweighted voting of the $m$ classifiers $h_j$.

#### R script

The R file `arcing.R` tests the performance of the ARCing algorithm in a classification task. Consider the medical dataset *Breast Cancer* obtained by Dr. William H. Wolberg (physician) at the University of Wisconsin Hospital in USA. This dataset reports the class of cancer (malignant and benign) together with other properties (clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion,...) The classification task is to predict the class of the breast cancer as a function of the clinical measures. We consider a training set of $N = 400$ and a test set of 299 samples. The classifier is a simple classification tree which returns a misclassification rate $\widehat{\text{MME}}_{\text{ts}} = 0.063$. We use an arcing procedure with $m = 15$. It gives a misclassification rate $\widehat{\text{MME}}_{\text{ts}} = 0.010$.

```
## script arcing.R

library(class)
library(mlbench)
library(tree)
data(BreastCancer)
N.tr<-400
N<-nrow(BreastCancer)
I<-BreastCancer$Class=='benign'
BreastCancer$Class2[I]<-1
BreastCancer$Class2[! I]<- -1
train<-BreastCancer[1:N.tr,]
w<-rep(1/N.tr,N.tr)

test<-BreastCancer[(N.tr+1):N,]
```

```
m<-15

misc<-rep(NA,m);
alpha<-rep(NA,m)
set.seed(555)
tree.model <-tree(Class2 ~ Cl.thickness+Cell.size+Cell.shape+
                    +Marg.adhesion+Epith.c.size+Bare.nuclei+Bl.cromatin+Normal.nucleoli+Mitoses
                  , train,control=tree.control(N.tr,mincut=50),weights=w*N.tr)

pred <- sign(predict(tree.model,test))
misc.tree <- sum(test$Class2 != pred)/length(pred)

pred.test<-rep(0,N-N.tr)
mi<-rep(0,N.tr)

for (j in 1:m)
{
  set.seed(555)
  I<-sample(seq(1,N.tr),prob=w,replace=TRUE)
  tree.model <-tree(Class2 ~ Cl.thickness+Cell.size+Cell.shape+
                      Marg.adhesion+Epith.c.size+Bare.nuclei+Bl.cromatin+Normal.nucleoli+Mitoses
                    train[I,],control=tree.control(N.tr,mincut=50))

  pred.train <-sign(predict(tree.model,train))

  mi<-mi+as.integer(train$Class2 != pred.train)

  w<- (1+mi^4)/(sum(1+mi^4))
  pred.test<-pred.test+sign(predict(tree.model,test))

}

pred.test<-pred.test/m
misc.arcing <- sum(test$Class2 != sign(pred.test))/length(pred.test)

cat("Misclassification single tree=",misc.tree,"\n")
cat("Misclassification boosting=",misc.arcing,"\n")
```

•

Boosting is a recent and promising technique which is simple and easy to program. Moreover, it has few parameters (e.g. max number of classifiers) to tune. Boosting methods advocate a shift in the attitude of the learning-system designer: instead of trying to design a learning algorithm which should be accurate over the entire space, he can instead focus on finding weak algorithms that only need to be better than random.

Furthermore, a nice property of Ada Boost is its ability to identify outliers (hard samples).

### 9.3.3 Bagging and boosting

This section makes a short comparison of bagging and boosting techniques. Like bagging, boosting avoid the cost of heavy validation procedures and, like bagging,

boosting trades accuracy for interpretability. As for bagging, the main effect of boosting is to reduce variance and it works effectively for high variance classifiers. However, unlike bagging, boosting cannot be implemented in parallel, since it is based on a sequential procedure. In terms of experimental accuracy, several research works (e.g. Breiman's work) show that boosting seems outperforms bagging

Some caveats are notwithstanding worthy to mention: the actual performance of boosting on a particular problem is clearly dependent on the data and the weak learner. Also boosting can fail to perform well given insufficient data, overly complex weak hypothesis, and weak hypothesis that are definitely too weak.

# Chapter 10

# Feature selection

Supervised learning algorithms techniques were originally not designed to cope with large amounts of irrelevant variables and are known to degrade in performance (prediction accuracy) when faced with many inputs (also known as *features*) that are not necessary for predicting the desired output.

In the feature selection problem, a learning algorithm is faced with the problem of selecting some subset of features upon which to focus its attention, while ignoring the rest. In many recent challenging problems the number of features amounts to several thousands: this is the case of microarray cancer classification tasks [105] where the number of variables (i.e. the number of genes for which expression is measured) may range from 6000 to 40000. New high-throughput measurement techniques (e.g. in biology) let easily foresee that this number could be rapidly passed by several orders of magnitude.

Using all available features in learning may negatively affect generalization performance, especially in the presence of irrelevant or redundant features. In that sense feature selection can be seen as an instance of model selection problem.

There are many potential benefits of feature selection [56, 57]:

- facilitating data visualization and data understanding,
- reducing the measurement and storage requirements,
- reducing training and utilization times of the final model,
- defying the curse of dimensionality to improve prediction performance.

At the same time we should be aware that feature selection implies additional time for learning. In fact the search for a subset of relevant features introduces an additional layer of complexity in the modelling task. The search in the model hypothesis space is augmented by another dimension: the one of finding the optimal subset of relevant features.

## 10.1 Curse of dimensionality

Feature selection addresses what is known in several scientific domains as the *curse of dimensionality*. This term, coined by R E Bellman, refers to all computational problems related to large dimensional modeling tasks.

For instance in classification, though the error of the best model (e.g. the error (5.3.13) of the Bayes classifier) decreases with $n$, the mean integrated squared error of models increases faster than linearly in $n$.

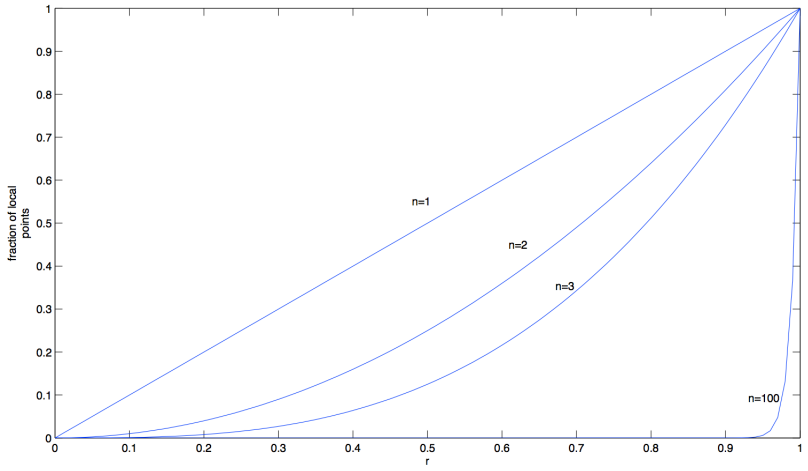



Figure 10.1: Local density of points for increasing dimensionality $n$.

This is essentially due to the fact that in high dimensions, all data sets are sparse, show multicollinearity and that the number of possible models to consider increases superexponentially in $n$.

By analyzing the degree of sparsity of data with respect to the dimensionality $n$ it is possible to show that as $n$ increases the amount of local data goes to zero.

For instance Figure 10.1 shows that for a uniform distribution around a query point $x_q$ the amount of data that are contained in a ball of radius $r < 1$ centered in $x_q$ grows like $r^n$.

As a consequence the size of the neigborhood on which we can estimate local features of the output (e.g. $E[y|x]$ in local learning) increases with dimension $n$, making the estimation coarser and coarser.

## 10.2 Approaches to feature selection

There are three main approaches to feature selection:

- **Filter methods:** they are preprocessing methods. They attempt to assess the merits of features from the data, ignoring the effects of the selected feature subset on the performance of the learning algorithm. Examples are methods that select variables by ranking them through compression techniques (like PCA or clustering) or by computing correlation with the output.

- **Wrapper methods:** these methods assess subsets of variables according to their usefulness to a given predictor. The method conducts a search for a good subset using the learning algorithm itself as part of the evaluation function. The problem boils down to a problem of stochastic state space search. Example are the stepwise methods proposed in linear regression analysis.

- **Embedded methods:** they perform variable selection as part of the learning procedure and are usually specific to given learning machines. Examples are classification trees, random forests, and methods based on regularization techniques (e.g. lasso)

## 10.3 Filter methods

Filter methods are commonly used in case of very large dimensionality (e.g. $n > 2000$) since they have some nice properties: they easily scale to very high-dimensional datasets, they are computationally simple and fast, and independent of the classification algorithm. Also feature selection needs to be performed only once, and then different classifiers can be evaluated.

However users should be aware of some limitations. Filter by definition ignore the interaction with the classifier and are often univariate or low-variate. This means that each feature is considered separately, thereby ignoring feature dependencies, which may lead to worse classification performance when compared to other types of feature selection techniques.

### 10.3.1 Principal component analysis

Principal component analysis (PCA) is one of the most popular methods for linear dimensionality reduction. It can project the data from the original space into a a lower dimensional space in an unsupervised manner. Each of the original dimensions is an axis. However, other axes can be created as linear combinations of the original ones.

PCA creates a completely new set of axes (principal components) that like the original ones are orthogonal to each other.

The first principal component (i.e. the axis $v_1$ in Figure 10.2) is the axis through the data along which there is the greatest variation amongst the observations. This corresponds to find the vector $a = [a_1, \ldots, a_n] \in \mathbb{R}^n$ such that the variable

$$\mathbf{z} = a_1 \mathbf{x}_{\cdot 1} + \cdots + a_n \mathbf{x}_{\cdot n} = a^T \mathbf{x}$$

has the largest variance. It can be shown that the optimal $a$ is the eigenvector of $\mathrm{Var}[\mathbf{x}]$ corresponding to the largest eigenvalue.

The second principal component (i.e. the axis $v_1$ in Figure 10.2) is the axis orthogonal to the first that has the greatest variation in the data associated with it; the third p.c. is the axis with the greatest variation along it that is orthogonal to both the first and the second axed; and so forth.

#### 10.3.1.1 PCA: the algorithm

Consider the training input matrix $X$ having size $[N, n]$. The PCA consists in the following steps.

1. The matrix is normalized and transformed to a matrix $\tilde{X}$ such that each column $\tilde{X}[, i]$, $i = 1, \ldots, n$, has mean 0 and variance 1.

2. The Singular Value Decomposition (SVD) of $\tilde{X}$ is computed

   $$\tilde{X} = UDV^T$$

   where $U$ is an orthogonal $[N, N]$ matrix, $D$ is a $[N, n]$ rectangular diagonal matrix with diagonal elements $d_1 \geq d_2 \geq \cdots \geq d_n$ and $V$ is an orthogonal matrix $[n, n]$.

3. The matrix $\tilde{X}$ can be transformed into a new set of coordinates $Z = \tilde{X}V = UD$ where $UD$ is a $[N, n]$ matrix, where each column is a linear combination of the original features and its importance is diminishing. The first $h < n$ columns of $Z$ (aka eigen-features) may be chosen to represent the dataset.

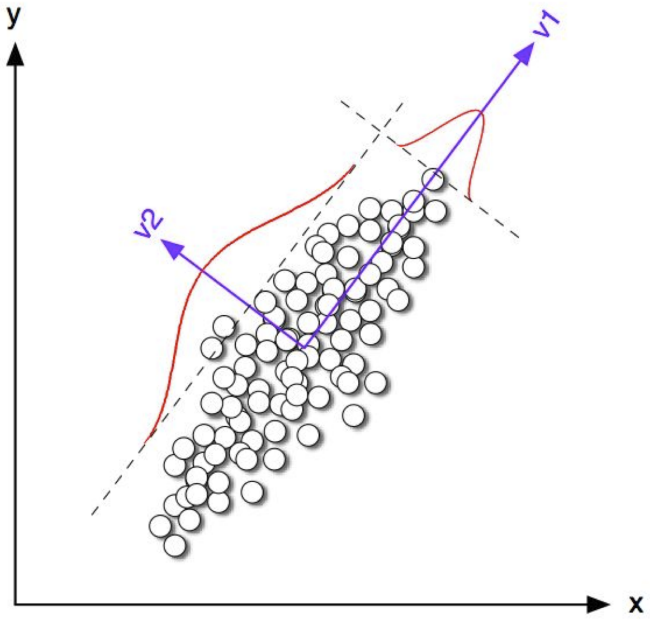



Figure 10.2: Two first principal components in case of $n = 2$.

### 10.3.2 Clustering

Clustering, also known as unsupervised learning, is presented in Appendix A. Here we will discussed how it may play a role also in dimensionality reduction by determining groups of features or observations with similar patterns (e.g. patterns of gene expressions in microarray data).

The use of a clustering method for feature selection requires the definition of a distance function between variables and the definition of a distance between clusters.

The two most common methods are

- **Nearest neighbor clustering:** the number of clusters is decided first, then each variable is assigned to each cluster. Examples are Self Organizing Maps (SOM) and K-means.
- **Agglomerative clustering:** they are bottom-up methods where clusters start as empty and variables are successively added. Example is hierarchical clustering: it begins by considering all the observations as separate clusters and starts by putting together the two samples that are nearest to each other. In subsequent stages also clusters can be merged. A measure of dissimilarity between sets of observations is required. It is based on an appropriate metric (a measure of distance between pairs of observations), and a linkage criterion which specifies the dissimilarity of sets as a function of the pairwise distances of observations in the sets. The output of hierarchical clustering is a *dendrogram*, a tree diagram used to illustrate the arrangement of the clusters (Figure 10.3).

### 10.3.3 Ranking methods

It is probably the most commonly used family of filters because of its simplicity.

Ranking methods assess the importance (or relevance) of each variable with respect to the output by using a univariate measure and select the best $k$ variables.

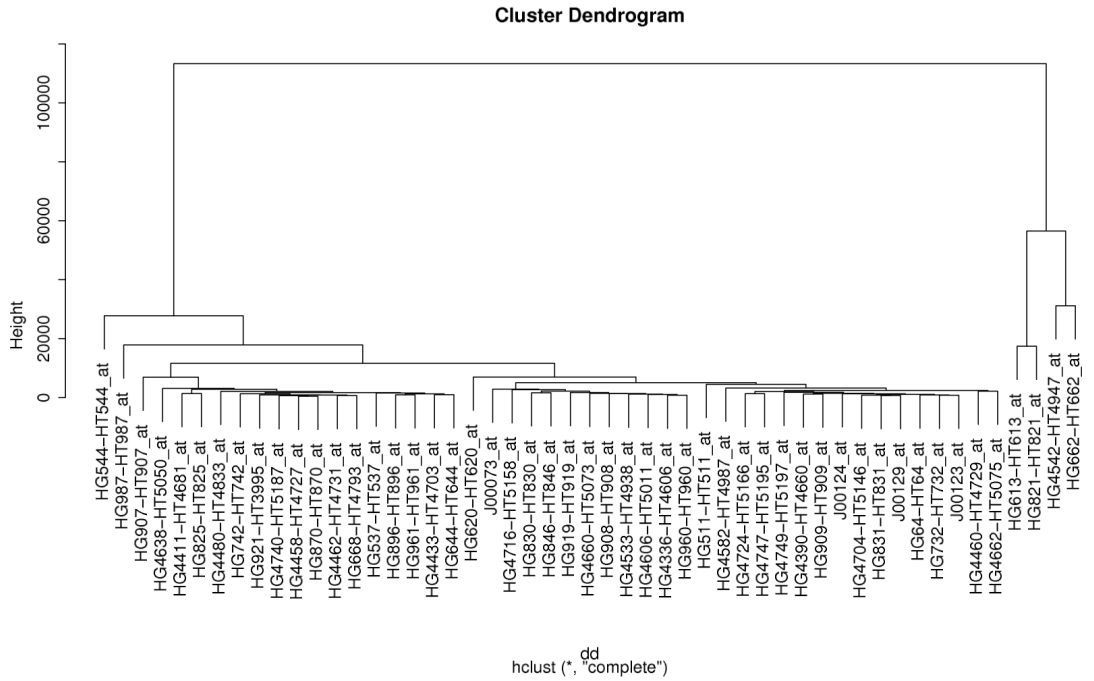



Figure 10.3: Dendrogram.

Unlike the two techniques previously presented, they are supervised since they take into account the target $\mathbf{y}$.

Measures of relevance which are commonly used in ranking are:

- Pearson correlation (the greater the more relevant) which assumes linearity;
- in case of binary classification significance p-value of an hypothesis test (the lower the more relevant) which aims at detect the features that split well the dataset. Parametric (t-test) and nonparametric (Wilcoxon) tests have been proposed in litterature;
- mutual information (Section 2.7.10) (the greater the more relevant).

After the univariate assessment the method ranks the variables in a decreasing order of relevance.

These methods are fast (complexity $O(n)$) and their output is intuitive and easy to understand. At the same time they disregard redundancies and higher order interactions between variables (e.g. genes).

The best $k$ features taken individually do not necessarily constitute the best $k$ variate vector.

### Feature selection in a gene expression dataset

A well-known classification task characterized by a very large dimensionality is the classification of gene expression data in bioinformatics. This task is characterized by a very large number of variables (e.g. large number of gene probes) with respect to the number of samples (e.g. number of patients). An example of gene expression (microaray) dataset is the one used by Golub et al. in [54]. This dataset contains the genome expressions of $n = 7129$ genes of $N = 72$ patients, for which $V = 11$

phenotype variables are measured. In the following we will focus on the dependency between the expression of the genes and one specific phenotype variable, the ALL.AML factor indicating the leukemia type: lymphoblastic leukemia (ALL) or acute myeloid leukemia (AML).

The expression matrix $X$ and the phenotype vector $Y$ are contained in the file `golub.Rdata`.

The script `featsel.R` performs ranking by correlation and shows the misclassification error for different sizes of the input set.

•

## 10.4 Wrapping methods

Wrapper methods combine a search in the space of possible feature subsets with an assessment phase relying on a learner and a validation (often cross-validation) technique. Unlike filter methods, wrappers take into consideration in a supervised manner the interaction between features.

Unfortunately this induce a much larger computational cost especially in case of expensive training phases.

The wrapper search can be seen as a search in a space $W = \{0,1\}^n$ where a generic vector $w \in W$ is such that

$$w[j] = \begin{cases} 0 & \text{if the input } j \text{ does NOT belong to the set of features} \\ 1 & \text{if the input } j \text{ belongs to the set of features} \end{cases}$$

Wrappers look for the optimal vector $w^* \in \{0,1\}^n$ such that

$$w^* = \arg\min_{w \in W} \text{MISE}_w$$

where $\text{MISE}_w$ is the generalization error of the model based on the set of variables described by $w$.

Note that the number of vectors in $W$ is equal to $2^n$. and that for moderately large $n$, the exhaustive search is no more possible. For this reason wrappers rely typically on heuristic search strategies.

### 10.4.1 Wrapping search strategies

Greedy methods have been developed for evaluating only a $O(n^2)$ number of variables by either adding or deleting one variable at a time. Three are the main categories:

- **Forward selection:** the procedure starts with no variables and progressively incorporate features. The first input selected is the one which allows the lowest generalization error. The second input selected is the one that, together with the first, has the lowest error, and so on, till when no improvement is made. R script `fs_wrap.R`.
- **Backward selection:** it works in the opposite direction of the forward approach by progressively removing feature from the full set. We begin with a model that contains all the $n$ variables. The first input to be removed is the one that allows the lowest generalization error.
- **Stepwise selection:** it combines the previous two techniques, by testing for each set of variables, first the removal of features beloning to the set, then the addition of variables not in the set.

Though these techniques are commonly used, a word of caution about their capacity of detecting the optimal solution comes from the Cover and van Campenhout theorem [36], which provides a negative result about the aim of wrapper search techniques to find the optimal subset by greedy local procedures.

Let us consider a learning problem and denote by $R^*_w$ the lowest functional risk (Section 5.2) for the subset of variables $w$. It can be shown that

$$w_2 \subset w_1 \Rightarrow R^*_{w_1} \leq R^*_{w_2}$$

that is by adding variables we reduces the minimal risk.

Cover and van Campenhout demonstrated that given $n$ features, **any** ordering of the $2^n$ subsets consistent with the above constraint is possible. This means for any possible ordering there exists a distribution of the data that is compatible with that.

In order terms if we found the the best set of three variables is

$$\{\mathbf{x}_{\cdot 1}, \mathbf{x}_{\cdot 3}, , \mathbf{x}_{\cdot 13}\}$$

there is no guarantee that the best set of four variables is a superset of $w_1$, e.g. it could be

$$\{\mathbf{x}_{\cdot 2}, \mathbf{x}_{\cdot 6}, \mathbf{x}_{\cdot 16}, \mathbf{x}_{\cdot 23}\}.$$

## 10.5 Embedded methods

They are typically less computationally intensive than wrapper methods but are specific to a learning machine (e.g. classification tree).

### 10.5.1 Shrinkage methods

Shrinkage is a general technique to improve a least-squares estimator which consists in reducing the variance by adding constraints on the value of coefficients. In what follows we present shrinkage approaches which penalize those least-squares solutions having a large number of coefficients with absolute values different from zero. The rationale is that only those variables whose impact on the empirical risk is considerable deserve to take a coefficient bigger than zero and consequently appear in the fitted linear model. Doing shrinkage is therefore an implicit embedded manner of doing feature selection since only a subset of variables contributes to the final predictor.

#### 10.5.1.1 Ridge regression

Ridge regression is an example of shrinkage method applied to least squares regression

$$\hat{\beta}_r = \arg\min_b \{\sum_{i=1}^{N}(y_i - x_i^T b)^2 + \lambda \sum_{j=1}^{p} b_j^2\} = \\ = \arg\min_b \left((Y - Xb)^T(Y - Xb) + \lambda b^T b\right)$$

where $\lambda > 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of $\lambda$ the greater the amount of shrinkage. Note that if $\lambda = 0$ the approach boils down to a conventional unconstrained least-squares.

An equivalent way to write the ridge problem is

$$\hat{\beta}_r = \arg\min_b \sum_{i=1}^{N} (y_i - x_i^T b)^2,$$
$$\text{subject to } \sum_{j=1}^{p} b_j^2 \leq L$$

where there is a one-to-one correspondence between the parameter $\lambda$ and $L$

It can be shown that the ridge regression solution is

$$\hat{\beta}_r = (X^T X + \lambda I)^{-1} X^T Y$$

where $I$ is a $[p, p]$ identity matrix.

#### 10.5.1.2 Lasso

Another well known shrinkage method is *lasso* where the estimate of the linear parameters is returned by

$$\hat{\beta}_r = \arg\min_b \sum_{i=1}^{N} (y_i - x_i^T b)^2,$$
$$\text{subject to } \sum_{j=1}^{p} |b_j| \leq L$$

If on one hand the 1-norm penalty of the lasso approach allows a stronger constraint on the coefficients, on the other hand it makes the solution nonlinear and demands the adoption a quadratic programming algorithm.

Note that if $L > \sum_{j=1}^{p} \hat{\beta}_j$ the lasso returns the common least-squares solution. The penalty factor $L$ is typically set by having recourse to cross-validation strategies.

## 10.6 Averaging and feature selection

The role of averaging methods in supervised learning has been discussed in the previous chapter.

Now, it appears that averaging may play a crucial role also in dealing with large dimensionality.

Instead of choosing one particular feature selection method, and accepting its outcome as the final subset, different feature selection methods can be combined using ensemble approaches. Since there is not an optimal feature selection technique and due to the possible existence of more than one subset of features that discriminates the data equally well, model combination approaches have been adapted to improve the robustness and stability of final, discriminative methods.

Novel ensemble techniques include averaging over multiple single feature subsets. Furthermore, methods based on a collection of decision trees (e.g. random forests) can be used in an ensemble way to assess the relevance of each feature. Although the use of ensemble approaches requires additional computational resources, this is still feasible with small sample domains.

## 10.7 Feature selection from an information-theoretic perspective

So far, we focused mainly on state-of-the-art algorithmic methods for feature selection. In this section we intend to go deeper in the study of the properties of feature

selection by using notions of information theory.

The notions of entropy and mutual information as measures of variability and dependency have been already introduced in Chapter 2.

Here we go a bit further by introducing the notion of conditional information.

**Definition 7.1** (Conditional information)**.** Consider three r.v.s $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. The *conditional mutual information* is defined by

$$I(\mathbf{y};\mathbf{x}|\mathbf{z}) = H(\mathbf{y}|\mathbf{z}) - H(\mathbf{y}|\mathbf{x},\mathbf{z})$$

The conditional mutual information is null iff $\mathbf{x}$ and $\mathbf{y}$ are conditionally independent given $\mathbf{z}$.

For a larger number $n$ of variables $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ a chain rule holds

$$I(\mathbf{X};\mathbf{y}) = I(\mathbf{X}_{-i};\mathbf{y}|\mathbf{x}_i) + I(\mathbf{x}_i;\mathbf{y}) = I(\mathbf{x}_i;\mathbf{y}|\mathbf{X}_{-i}) + I(\mathbf{X}_{-i};\mathbf{y}), \quad i = 1, \ldots, n$$

This means that for $n = 2$

$$I(\{\mathbf{x}_1,\mathbf{x}_2\};\mathbf{y}) = I(\mathbf{x}_2;\mathbf{y}|\mathbf{x}_1) + I(\mathbf{x}_1;\mathbf{y}) = I(\mathbf{x}_1;\mathbf{y}|\mathbf{x}_2) + I(\mathbf{x}_2;\mathbf{y}) \tag{10.7.1}$$

### 10.7.1 Relevance, redundancy and interaction

On the basis of the previous definition it is possible to define in information-theoretic terms what is a relevant variable in a supervised learning task where $\mathbf{X}$ is a set of $n$ input variables and $\mathbf{y}$ is the target. These definitions are obtained by interpreting in information theoretic terms the definitions made by [75].

**Definition 7.2** (Strongly relevance)**.** A variable $\mathbf{x}_i \in \mathbf{X}$ is *strongly relevant* to the target $\mathbf{y}$ if

$$I(\mathbf{X}_{-i};\mathbf{y}) < I(\mathbf{X};\mathbf{y})$$

where $\mathbf{X}_{-i}$ is the set obtained by removing the variable $\mathbf{x}_i$ from $\mathbf{X}$.

In other words, a variable is strongly relevant if it carries some information about $\mathbf{y}$ that no other variable carrries. Strong relevance indicates that the feature is always necessary for an optimal subset.

**Definition 7.3** (Weak relevance)**.** A variable is *weakly relevant* to the target $\mathbf{y}$ if it is not strongly relevant and

$$\exists \mathbf{S} \subseteq \mathbf{X}_{-i} : I(\mathbf{S};\mathbf{y}) < I(\{\mathbf{x}_i,\mathbf{S}\};\mathbf{y})$$

In other words, a variable is weakly relevant when it exists a certain context $\mathbf{S}$ in which it carries information about the target. Weak relevance suggests that the feature is not always necessary but may become necessary at certain conditions.

**Definition 7.4** (Irrelevance)**.** A variable is *irrelevant* if it is neither strongly or weakly relevant.

Irrelevance indicates that the feature is not necessary at all.

#### Example

Consider a learning problem where $n = 4$, $\mathbf{x}_3 = -\mathbf{x}_2$

$$\mathbf{y} = \begin{cases} 1 + \mathbf{w}, & \mathbf{x}_1 + \mathbf{x}_2 > 0 \\ 0, & \text{else} \end{cases}$$

Which variables are strongly, weakly relevant and irrelevant?

•

**Definition 7.5** (Markov blanket)**.** Let us consider a set $\mathbf{X}$ of $n$ r.v.s., a target variable $\mathbf{y}$ and a subset $\mathbf{M}_y \subset \mathbf{X}$. The subset $\mathbf{M}$ is said to be a *Markov blanket* of $\mathbf{y}$, $\mathbf{y} \notin \mathbf{M}$ iff

$$I(\mathbf{y}; \mathbf{X}_{-(M_y)})|\mathbf{M}_y) = 0$$

The following theorem can be shown

**Theorem 7.6** (Total conditioning)**.**

$$\mathbf{x} \in \mathbf{M}_y \Leftrightarrow I(\mathbf{x}; \mathbf{y}|\mathbf{X}_{-(x,y)}) > 0$$

This theorem proves that the Markov blanket of a target $\mathbf{y}$ is composed of the set of all the strongly relevant variables in $\mathbf{X}$.

Another useful notion to reason about how a set of variables bring information about a target $\mathbf{y}$ is the notion of interaction.

**Definition 7.7** (Interaction)**.** Given three r.v.s. $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{y}$ we define the interaction between these three variables as

$$I(\mathbf{x}_1; \mathbf{y}) - I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2)$$

An important property of interaction is the following:

$$I(\mathbf{x}_1; \mathbf{y}) - I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2) = I(\mathbf{x}_1; \mathbf{x}2) - I(\mathbf{x}_1; \mathbf{x}_2|\mathbf{y}) = I(\mathbf{x}_2; \mathbf{y}) - I(\mathbf{x}_2; \mathbf{y}|\mathbf{x}_1)$$

Note that since, because of the chain rule (10.7.1)

$$I(\mathbf{x}_2; \mathbf{y}|\mathbf{x}_1) + I(\mathbf{x}_1; \mathbf{y}) = I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2) + I(\mathbf{x}_2; \mathbf{y})$$

we obtain

$$I(\mathbf{x}_2; \mathbf{y}|\mathbf{x}_1) = I(\mathbf{x}_2; \mathbf{y}) - I(\mathbf{x}_1; \mathbf{y}) + I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2)$$

By summing $I(\mathbf{x}_1; \mathbf{y})$ to both sides, from (10.7.1) it follows that the joint information of two variables about a target $\mathbf{y}$ can be decomposed as follows:

$$\begin{aligned} I(\{\mathbf{x}_1, \mathbf{x}_2\}; \mathbf{y}) &= I(\mathbf{x}_1; \mathbf{y}) + I(\mathbf{x}_2; \mathbf{y}) - \underbrace{[I(\mathbf{x}_1; \mathbf{y}) - I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2)]}_{\text{interaction}} = \\ &= I(\mathbf{x}_1; \mathbf{y}) + I(\mathbf{x}_2; \mathbf{y}) - \underbrace{[I(\mathbf{x}_1; \mathbf{x}_2) - I(\mathbf{x}_1; \mathbf{x}_2|\mathbf{y})]}_{\text{interaction}} \end{aligned} \quad (10.7.2)$$

What emerges is that the joint information of two variables is not necessarily equal, greater or smaller than the sum of the two individual information terms. All depends on the interaction term: if the interaction term is negative the two variables are complementary, or in other terms they bring jointly more information than the sum of the univariate terms. This is typically the case of the XOR example illustrated in Figure 10.4. In this case $I(\mathbf{x}_1; \mathbf{y}) = 0$, $I(\mathbf{x}_2; \mathbf{y}) = 0$ but $I(\{\mathbf{x}_1, \mathbf{x}_2\}; \mathbf{y}) > 0$ and maximal. When they are redundant, the resulting joint information is lower than the sum $I(\mathbf{x}_1; \mathbf{y}) + I(\mathbf{x}_2; \mathbf{y})$.

Since (10.7.2) holds also when $\mathbf{x}_1$ and/or $\mathbf{x}_2$ are sets of variables, this result sheds an interesting light about the challenges of feature selection.

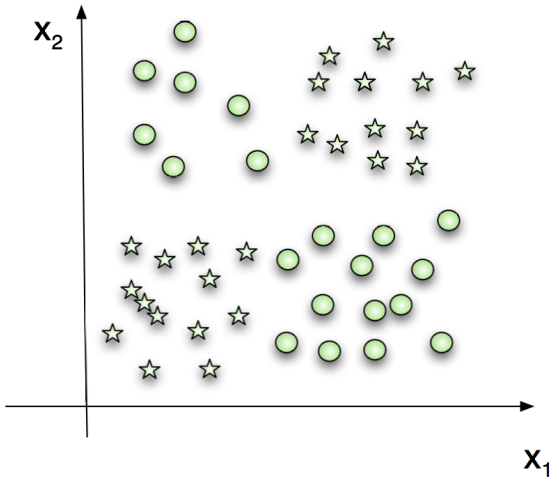



Figure 10.4: XOR classification separable task with two inputs and one binary class taking two values (stars and rounds). Two variables $\mathbf{x}_1$ and $\mathbf{x}_2$ are complementary: they bring alone no information but they bring the maximal information about $\mathbf{y}$ when considered together.

### 10.7.2 Information theoretic filters

In terms of mutual information the feature selection problem can be formulated as follows. Given an output target $\mathbf{y}$ and a set of input variables $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ selecting the optimal subset of $d$ variables is the solution of the following optimization problem

$$X^* = \arg \max_{\mathbf{X}_S \subset \mathbf{X}, |\mathbf{X}_S| = d} I(\mathbf{X}_S; \mathbf{y})$$

Thanks to the chain rule, this maximization task can be tackled by adopting an incremental approach (e.g. forward approach).

Let $\mathbf{X} = \{\mathbf{x}_i\}, i = 1, \ldots, n$ the whole set of variables and $\mathbf{X}_S$ the set of variables selected after $m$ steps. The choice of the $(m+1)$th variable $\mathbf{x}^* \in \mathbf{X} - \mathbf{X}_S$ can be done by solving

$$x^* = \arg \max_{\mathbf{x}_k \in \mathbf{X} - \mathbf{X}_S} I(\{\mathbf{X}_S, \mathbf{x}_k\}; \mathbf{y}) \tag{10.7.3}$$

This is known as the *maximal dependency* problem and requires at each step multivariate estimation of the mutual information term $I(\{\mathbf{X}_S, \mathbf{x}_k\}; \mathbf{y})$. Such estimation is often inaccurate in large variate settings (i.e. large $n$ and large $m$) because of ill-conditioning and high variance issues.

In literature several filter approaches have been proposed to solve the optimization (10.7.3) by approximating the multivariate term $I(\{\mathbf{X}_S, \mathbf{x}_k\}; \mathbf{y})$ with low variate approximations. These approximations are necessarily biased, yet much less prone to variance than their multivariate counterparts.

We mention here two of the most used information theoretic filters:

- CMIM [48]: since according to the first (chain-rule) formulation

$$\arg \max_{\mathbf{x}_k \in \mathbf{X} - \mathbf{X}_S} I(\{\mathbf{X}_S, \mathbf{x}_k\}; \mathbf{y}) = \arg \max_{\mathbf{x}_k \in \mathbf{X} - \mathbf{X}_S} I(\mathbf{x}_k; \mathbf{y} | \mathbf{X}_S)$$

  it makes the approximation

$$I(\mathbf{x}_k; \mathbf{y} | \mathbf{X}_S) \approx \min_{\mathbf{x}_j \in \mathbf{X}_S} I(\mathbf{x}_k; \mathbf{y} | \mathbf{x}_j)$$

- mRMR (minimum Redundancy Maximal Relevance) [94]: the mRMR method approximates at the $m+1$ step $I(\{\mathbf{X}_S, \mathbf{x}_k\}; \mathbf{y})$ with

$$I(\mathbf{x}_k; y) - \frac{1}{m} \sum_{\mathbf{x}_i \in X_S} I(\mathbf{x}_i; \mathbf{x}_k)$$

  where $m$ is the number of features belonging to $\mathbf{X}_S$. As a result it implements a forward selection method which selects at each step the variable which has at the same time the highest degree of relevance and the lowest redundancy with the already selected features in $\mathbf{X}_S$.

## 10.8 Conclusion

Most of the discussed techniques aim to find the best subset of features by performing a large number of comparisons and selections. Despite the use of validation procedures, it is highly possible that low misclassification or low prediction errors are found only due to chance.

A bad practice consists in using the same set of observations both to select the feature set and to assess the generalization accuracy of the classifier. The use of external validation sets is therefore strongly recommended.

If no additional data are available, an alternative consists in estimating how often random data would generate correlation or classification accuracy of the magnitude obtained in the original analysis. Typically this is done through use of permutation testing. This consists in repeating the feature selection several times with data where the dependency between the input and the output is artificially removed (for exemple by permuting the output ordering). This provides us with a distribution of the accuracy in case of random data where no information is present in the data.

Making a robust assessment of a feature selection outcome has a striking importance today because we are more and more confronted with tasks characterized by a very large feature to sample ratio, where a bad assessment procedure can give too optimistic (over fitted) results.

# Chapter 11

# Conclusions

A vast amount of literature in machine learning served the purpose of showing the superiority of some learning methods over the others. To support this claim, qualitative considerations and thousands of experimental simulations have been submitted to the scientific community.

If there was a universally best learning machine, research on machine learning would be unnecessary: we would use it all the time. Unfortunately, the theoretical results on this subject are not encouraging [36]. For any number $N$ of samples, there exists a distribution of the input/output samples for which the estimate of generalization error is arbitrarily poor. At the same time, for any learning machine LM1 there exists a data distribution and another learning machine LM2 such that for all $N$, LM2 is better than LM1.

The nonexistence of the universal best learning approach implies that a debate based on experiments would never end and that simulations should never be used to prove the superiority of one method over another.

These considerations warn the designer not to prefer unconditionally a data analysis tool but to take always in consideration a range of possible alternatives. If it is not realistic to find the approximator which works nicely for any kind of experiment, it is however advisable to have an understanding of available methods, their rationale, properties and use.

We would like then to end this manuscript not by suggesting a unique and superior way of proceeding in front of data but by proposing some golden rules for anyone who would like to adventure in the world of statistical modeling and data analysis:

- Each approach has its own assumptions! Be aware of them before using it.
- Simpler things first!
- Reality is probably most of the time nonlinear but a massive amount of (theoretical, algorithmic) results exists only for linear methods.
- Expert knowledge MATTERS...
- But data too :-)
- Better be confident with a number of alternative techniques (preferably linear and nonlinear) and use them in parallel on the same task.
- Resampling techniques make few assumptions and appear as powerful nonparametric tools.

- Resampling and combing are at the forefront of the data analysis technology. Do not forget to test them when you have a data analysis problem.
- Do not be religious about learning/modeling techniques. The best learning algorithm does NOT exist.
- Statistical dependency does not imply causality.

The latter point will be discussed in the following section.

## 11.1 Causality and dependencies

In this book we focused exclusively on the modelling of dependencies between variables. The reader should be aware that a detection of dependency between an input variable $\mathbf{x}$ and an output variable $\mathbf{y}$ does not always imply the existence of a causality relationship between input and output variables.

In fact, it can sometimes happen that we obtain from an observational study a dependency between two variables which are not causally related. It would be erroneous and fallacious to deduce a causality from the existence of a statistical dependency.

Typically this happens when a phenomenon causes two effects and the cause is not observable. In this case a spurious relation between effects is observed.

To better illustrate this notion let us consider this example.

#### The Caca-Cola study

The Caca-Cola marketing department aims to show that, unlike what feared by most parents, the famous refreshing drink is beneficial to the sport performance of its drinkers. In order to support this idea, it funds a statistical study on the relationship between the amount of Caca-Cola liters drunk per day and the time spent (in seconds) by the drinker to perform a 100 meters run.

Here it is the dataset collected by statisticians

| Liters per day | Seconds |
|---|---|
| 1.00 | 11.9 |
| 1.09 | 12.8 |
| 0.20 | 14.3 |
| 0.18 | 15.3 |
| 1.02 | 12.2 |
| 1.04 | 12.5 |
| 1.06 | 12.6 |
| 0.00 | 16.3 |
| 1.08 | 12.7 |
| 0.18 | 17.7 |
| 0.50 | 14.0 |
| 0.17 | 17.6 |

illustrated by Figure 11.1 which plots the performance (in seconds) as a function of the number of drunk liters per day.

The Caca-Cola marketing department is excited. Caca-Cola seems to have magnificent effects on the sprinter performance, as illustrated by the significant correlation between amount of liters and running time of Figure 11.1. The CEO of the company triumphly extrapolates on the basis of a sophisticated machine learning tool that any human being fed with more than 3 liters per day could easily beat the world's record.

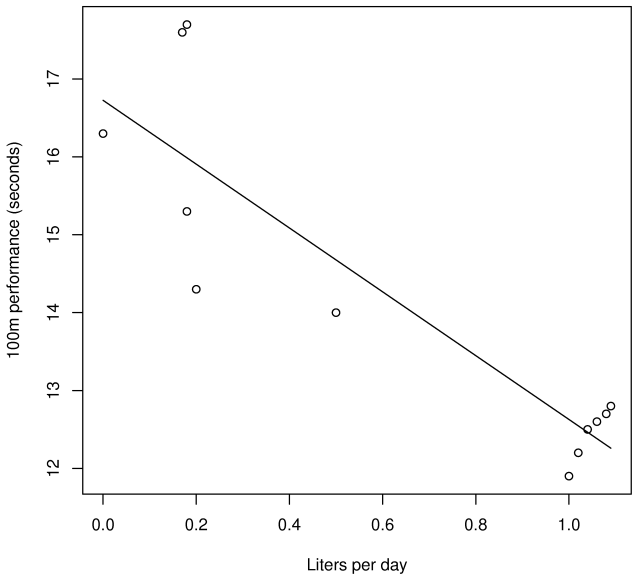



Figure 11.1: Performance improves as Caca-Cola drinking increases. It it real?

In front of such enthusiasm, the League of Parents is skeptical and ask for a simple elucidation: did the Caca-Cola statisticians record the age of the sprinter? In front of the growing public opinion pressure, Caca-Cola is forced to publish the complete dataset.

| Age | Liters per day | Seconds |
|---|---|---|
| 17 | 1.00 | 11.9 |
| 19 | 1.09 | 12.8 |
| 49 | 0.20 | 14.3 |
| 59 | 0.18 | 15.3 |
| 21 | 1.02 | 12.2 |
| 19 | 1.04 | 12.5 |
| 17 | 1.06 | 12.6 |
| 62 | 0.00 | 16.3 |
| 21 | 1.08 | 12.7 |
| 61 | 0.18 | 17.7 |
| 30 | 0.50 | 14.0 |
| 65 | 0.17 | 17.6 |

At last truth can triumph! Caca-Cola statisticains had hidden the real cause of good (or bad performance): the age of the athletes! Since youngsters tend to drink more Caca-Cola as well as having better sport performance, the first causal relationships between drunk liters and performance was fallacious. On the contrary a more detailed analysis on a homogenous group of young athletes show that Caca-Cola tend to deteriorate the performances. Figure 11.2 plots the performance (in seconds) as a function of drunk liters per day exclusively for the subgroup of young (less than 30) athletes.

Note that the Caca-Cola marjketing department was not wrong in claiming the existence of a significative relationship betwen Caca-Cola and performance. On the

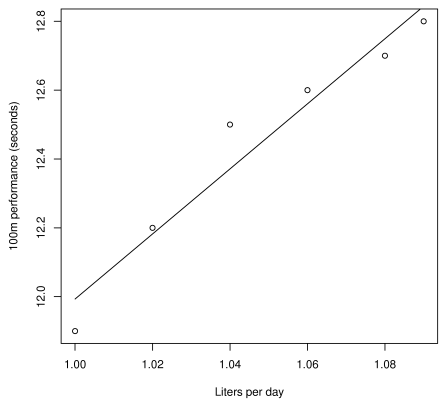



Figure 11.2: Performance deteriorates as Caca-Cola drinking increases.

contrary they were definitely wrong when they claimed the existence of a cause-effect relationship between theese two variables.

•

On purpose, this book did not discuss causality since this would have required much more pages. We suggest the interested readers to refer to [93] and the research work of J. Pearl.

# Appendix A

# Unsupervised learning

## A.1 Probability density estimation

The probability density estimation is the problem of inferring a probability density function $p_{\mathbf{z}}$, given a finite number of data points $\{z_1, z_2, \ldots, z_N\}$ drawn from that density function.

We distinguish three alternative approaches to density estimation:

**Parametric.** This approach assumes a parametric model of the unknown density probability. The parameters are estimated by fitting the parametric function to the observed dataset. This approach has been extensively discussed in Chapter 3.

**Nonparametric.** This approach does not assume any a priori form of the density model. The form of the density is entirely determined by the data and the number of parameters grows with the size of the dataset.

**Semi-parametric.** In this approach the number of parameter is not fixed a priori but is independent of the size of the dataset.

### A.1.1 Nonparametric density estimation

The term *nonparametric* is used to describe probability density functions whose functional form is not specified in advance, but is dependent on data [103, 92].

Let us consider a random variable $\mathbf{z}$ with density probability $p_{\mathbf{z}}(z)$ and a region $R$ defined on the $\mathbf{z}$ space. The probability that a value $z$ drawn according to $p_{\mathbf{z}}(z)$ falls inside $R$ is

$$\mathrm{P}_R = \mathrm{prob}\{z \in R\} = \int_R p_{\mathbf{z}}(z)dz \tag{A.1.1}$$

Let us define with $\mathbf{k}$ the random variable which represents the number of points which falls within $R$, after we have drawn $N$ points from $p_{\mathbf{z}}(z)$ independently.

From well-know results in probability theory we have that its probability distribution is

$$p_{\mathbf{k}}(k) = \frac{N!}{k!(N-k)!}\mathrm{P}_R^k(1-\mathrm{P}_R)^{(N-k)} \tag{A.1.2}$$

Moreover, the random variable $\mathbf{k}/N$ satisfies

$$E[\mathbf{k}/n] = \mathrm{P}_R \tag{A.1.3}$$

and

$$\mathrm{Var}[\mathbf{k}/N] = E[(\mathbf{k}/N - \mathrm{P}_R)^2] = \frac{\mathrm{P}_R(1-\mathrm{P}_R)}{N} \tag{A.1.4}$$

Since according to (A.1.4), the variance of $\mathbf{k}/N$ converges to zero as $N \to \infty$, it is reasonable to expect that the fraction $k/N$ return a good estimate of the probability $\mathrm{P}_R$

$$\mathrm{P}_R \cong \frac{k}{N} \tag{A.1.5}$$

At the same time, if we assume that $p_{\mathbf{z}}(z)$ is continuous and does not vary appreciably over $R$, we can approximate $\mathrm{P}_R$ with:

$$\mathrm{P}_R = \int_R p(z)dz \cong p(z)V \tag{A.1.6}$$

with $V$ volume of $R$. From (A.1.5) and (A.1.6) it follows that for values of $z$ inside $R$

$$p(z) \cong \frac{k}{NV} \tag{A.1.7}$$

In order for (A.1.5) to hold it is required to have a large $R$. This implies a sharply peaked $p_{\mathbf{z}}(z)$. In order for (A.1.6) to hold it is required to have a small $R$. This ensures a $p_{\mathbf{z}}(z)$ constant in $R$. These are two clashing requirements. We deduce that it is necessary to find an optimal trade-off for $R$ in order to guarantee a reliable estimation of $p_{\mathbf{z}}(z)$. This issue is common to all nonparametric approaches to density estimation.

In particular, we will introduce two of them

**Kernel-based.** This approach fixes $R$ and searches for the optimal number of points $k$.

**$k$-Nearest Neighbor (k-NN).** This approach fixes the value for $k$ and searches for the optimal $R$.

The two approaches are discussed in detail in the following sections.

#### A.1.1.1 Kernel-based methods

Consider a random vector $\mathbf{z}$ of dimension $[n \times 1]$ and suppose we take an hypercube region $R$ with sides of length $B$ centered on the point $z$. The volume of $R$ is

$$V = B^n$$

Let us now define a *kernel* function (or Parzen window) $K(u)$ as

$$K(u) = \begin{cases} 1 & \text{if } |u_j| < 1/2 \qquad j = 1, \ldots, n \\ 0 & \text{else} \end{cases} \tag{A.1.8}$$

where $u_j$ is the $j^{\text{th}}$ component of the vector $u$. It follows that the quantity

$$K\left(\frac{z - z_i}{B}\right)$$

is equal to unity if $z_i$ is inside the hypercube centered at $z$ with side $B$.

Therefore, given a set of $N$ points, the number of points falling inside $R$ is given by

$$k = \sum_{i=1}^{N} K\left(\frac{z - z_i}{B}\right) \tag{A.1.9}$$

From (A.1.7) and (A.1.9) it is possible to define the *kernel-based* estimate of the probability density for the kernel (A.1.8) as

$$\hat{p}(z) = \frac{\sum_{i=1}^{N} K\left(\frac{z-z_i}{B}\right)}{NB^n} \tag{A.1.10}$$

Note that the estimate (A.1.10) is discontinuous over the $z$-space. In order to smooth it we may choose alternative kernel functions, as the Gaussian kernel.

The kernel-based method is a traditional approach to density estimation. However, two are the most relevant shortcomings of this approach:

1. it returns a biased estimator [20],
2. it requires the memorization of the whole set of samples. As a consequence the estimation is very slow if there is an high number of data.

#### A.1.1.2 k-Nearest Neighbors methods

Consider an hyper sphere $R$ centered at a point $z$, and let us grow it until it contains a number of $k$ points. Using Eq. (A.1.7) we can derive the *k-Nearest Neighbor* (k-NN) density estimate

$$\hat{p}_{\mathbf{z}}(z) = \frac{k}{NV} \tag{A.1.11}$$

where $k$ is the value of a fixed a priori parameter, $N$ is the number of available observations and $V$ is the volume of the hyper sphere.

Like kernel-based methods, k-NN is a state-of-the-art technique in density estimation. However it features two main shortcomings

1. the quantity (A.1.11) is not properly a probability, since its integral over the whole Z space is not equal to one but diverges
2. as in the kernel method, it requires the storage of the whole dataset.

### A.1.2 Semi-parametric density estimation

In semi-parametric techniques the size of the model does not grow with the size of the data but with the complexity of the problem. As a consequence, the procedure for defining the structure of the model is more complex than in the approaches previously seen.

#### A.1.2.1 Mixture models

The unknown density function is represented as a linear superposition of $m$ basis functions. The distribution is called called *mixture model* and has the form

$$p_{\mathbf{z}}(z) = \sum_{j=1}^{m} p(z|j)\pi(j) \tag{A.1.12}$$

where $m$ is a parameter of the model and typically $m << N$. The coefficients $\pi(j)$ are called *mixing coefficients* and satisfy the following constraints

$$\sum_{j=1}^{m} \pi(j) = 1 \qquad 0 \leq \pi(j) \leq 1 \tag{A.1.13}$$

The quantity $\pi(j)$ is typically interpreted as the *prior* probability that a data point be generated by the $j^{\text{th}}$ component of the mixture. According to Bayes' theorem, the corresponding *posterior* probabilities is

$$p(j|z) = \frac{p(z|j)\pi(j)}{p(z)} = \frac{p(z|j)\pi(j)}{\sum_{j=1}^{m} p(z|j)\pi(j)} \tag{A.1.14}$$

Given a data point $z$, the quantity $p(j|z)$ represents the probability that the component $j$ had been responsible for generating $z$.

An important property of mixture models is that they can approximate any continuous density with arbitrary accuracy provided the model has a sufficient number of components and provided the parameters of the model are tuned correctly.

Let us consider a Gaussian mixture model with components $p(z|j) \sim N(\mu_j, \sigma_j^2)$ and suppose that a set of $N$ observations is available. Once fixed the number of basis functions, the parameters to be estimated from data are the mixing coefficients $\pi(j)$, and the terms $\mu_j$ and $\sigma_j$.

The procedure of maximum likelihood estimation of a mixture model is not simple, due to existence of local minima and singular solutions. Standard nonlinear optimization techniques can be employed, once the gradients of the log-likelihood with respect to the parameters is given. However, there exist algorithms which avoid the complexity of a nonlinear estimation procedure. One of them is the EM algorithm, which will be introduced in the following section.

#### A.1.2.2 The EM algorithm

The *expectation-maximization* or EM algorithm [35] is a simple and practical method for estimating the mixture parameters avoiding complex nonlinear optimization algorithm.

The assumption of the EM algorithm is that the available dataset is incomplete. This incompleteness can either be due to some missing measurements or because some imaginary data are introduced to simplify the mathematical form of the likelihood function.

The second situation is assumed to hold in the case of mixture models. The goal of the EM algorithm is then to maximize the likelihood of the parameters of a mixture model assuming that some data is missing in the available dataset.

The algorithm has an iterative form in which each iteration consists of two steps: an expectation calculation (E step) and a maximization (the M step). It has been shown in literature that the iteration of EM estimates converge to a local maximum of the likelihood of the incomplete data.

Assume that there exists a statistical model of our dataset $D_N$ and that it is parametrized by a real vector $\theta$. Assume also that further data, denoted by $\Xi$, exist but are not observable. The quantity $\Delta_N$ is used to denote the whole dataset, containing both the observed and unobserved data, and is usually referred to as the *complete data*.

Let us denote by $\mathbf{l}_{comp}(\theta)$ the log likelihood of the parameter $\theta$ given the complete data. This is a random variable because the values of $\Xi$ are not known. Hence, it is possible for a given value $\theta^{(\tau)}$ of the parameter vector to compute the expected value of $\mathbf{l}_{comp}(\theta^{(\tau)})$. This gives a deterministic function of the current value of the parameter, denoted by $Q(\theta^{(\tau)})$, that can be considered as an approximation to the real value of $l$, called the *incomplete likelihood*. The maximization step is expected to find the parameter value $\theta^{(\tau+1)}$ which maximize $Q$. The EM procedure in detail is the following:

1. Make an initial estimate $\theta^{(0)}$ of the parameter vector.
2. The log likelihood $\mathbf{l}_{comp}(\theta^{(\tau)})$ of the parameters $\theta^{(\tau)}$ with respect to the complete data $\Delta_N$ is calculated. This is a random function of the unknown dataset $\Xi$.
3. The E-step: the expectation $Q(\theta^{(\tau)})$ of $l_{comp}(\theta^{(\tau)})$ is calculated.
4. The M-step: a new estimate of the parameters is found by the maximization

$$\theta^{(\tau+1)} = \arg\max_{\theta} Q(\theta) \tag{A.1.15}$$

The theoretical justification comes from the following result proved in [35]: for a sequence $\theta^{(\tau)}$ generated by the EM algorithm it is always true that for the incomplete likelihood

$$l(\theta^{(\tau+1)}) \geq l(\theta^{(\tau)}) \tag{A.1.16}$$

Hence the EM algorithm is guaranteed to converge to a local maximum of the incomplete likelihood.

#### A.1.2.3 The EM algorithm for the mixture model

In the mixture model estimation problem the problem of determining the parameters (i.e. the mixing coefficients and the parameters of the density $p(z|j)$ in Eq. (A.1.12)) would be straightforward if we knew which component $j$ was responsible for generating each data point in the dataset. We therefore consider a hypothetical complete dataset in which each data point is labeled by the component which generated it. Thus, for each point $z_i$ we introduce $m$ indicator random variables $\boldsymbol{\zeta}_{ij}$, $j = 1, \ldots, m$, such that

$$\zeta_{ij} = \begin{cases} 1 & \text{if } z_i \text{ is generated by the } j^{\text{th}} \text{ basis} \\ 0 & \text{otherwise} \end{cases} \tag{A.1.17}$$

Let $\Delta_N$ be the extension of the dataset $D_N$, i.e. it represents the complete dataset, including the unobservable $\zeta_{ij}$. The probability distribution for each $(z_i, \zeta_{ij})$ is either zero or $p(z_i|j)$. If we let $\boldsymbol{\zeta}_i$ represent the set $\{\boldsymbol{\zeta}_{i1}, \boldsymbol{\zeta}_{i2}, \ldots, \boldsymbol{\zeta}_{im}\}$ then

$$p_{\boldsymbol{\zeta}_i}(\zeta_i) = \pi(j') \text{ where } j' \text{ is such that } \zeta_{ij'} = 1 \tag{A.1.18}$$

so

$$p(z_i, \zeta_i) = p(\zeta_i)p(z_i|j') = \pi(j')p(z_i|j') = \prod_{j=1}^{m} [\pi(j)p(z_i|j)]^{\zeta_{ij}} \tag{A.1.19}$$

Thus the complete log likelihood is given by

$$\mathbf{l}_{comp}(\theta) = \ln \mathbf{L}_{comp}(\theta) = \ln \prod_{i=1}^{N} \prod_{j=1}^{m} [\pi(j)p(z_i|j)]^{\boldsymbol{\zeta}_{ij}} \tag{A.1.20}$$

$$= \sum_{i=1}^{N} \ln \prod_{j=1}^{m} [\pi(j)p(z_i|j)]^{\boldsymbol{\zeta}_{ij}} \tag{A.1.21}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{m} \boldsymbol{\zeta}_{ij} \{\ln \pi(j) + \ln p(z_i|j)\} \tag{A.1.22}$$

where the vector $\theta$ includes the mixing coefficients and the parameters of the density $p(z|j)$ in Eq. (A.1.12).

Introducing the terms $\boldsymbol{\zeta}_{ij}$ the logarithm can be brought inside the summation term. The cost of this algebraic simplification is that we do not know the values of the $\boldsymbol{\zeta}_{ij}$ for the training data. At this point the EM algorithm can be used. For a value $\theta^{(\tau)}$ of the parameters the E-step is carried out:

$$Q(\theta^{(\tau)}) = E[\mathbf{l}_{comp}(\theta^{(\tau)}) = E[\sum_{i=1}^{N} \sum_{j=1}^{m} \boldsymbol{\zeta}_{ij} \{\ln \pi(j) + \ln p(z_i|j)\}] \tag{A.1.23}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{m} E[\boldsymbol{\zeta}_{ij}] \{\ln \pi(j) + \ln p(z_i|j)\} \tag{A.1.24}$$

Since

$$E[\boldsymbol{\zeta}_{ij}] = P(\zeta_{ij} = 1|z_i) = \frac{p(z_i|\zeta_{ij})P(\zeta_{ij})}{p(z_i)} = \frac{p(z_i|j)\pi(j)}{p(z_i)} = p(j|z_i) \tag{A.1.25}$$

from Eq. (A.1.14) and (A.1.18) we have

$$Q(\theta^{(\tau)}) = \sum_{i=1}^{N}\sum_{j=1}^{m} p(j|z_i)\{\ln \pi(j) + \ln p(z_i|j)\} \tag{A.1.26}$$

The M-step maximizes $Q$ with respect to the whole set of parameters $\theta$ but it is known that this can be done individually for each parameter, if we consider a Gaussian mixture model

$$p(z|j) = \frac{1}{(2\pi {\sigma_j}^2)^{n/2}} \exp\left\{-\frac{(z-\mu_j)^2}{2{\sigma_j}^2}\right\} \tag{A.1.27}$$

In this case we have:

$$Q(\theta^{(\tau)}) = \sum_{i=1}^{N}\sum_{j=1}^{m} p(j|z_i)\{\ln \pi(j) + \ln p(z_i|j)\} \tag{A.1.28}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{m} p(j|z_i)\left\{\ln \pi(j) - n \ln {\sigma_j}^{(\tau)} - \frac{(z_i - {\mu_j}^{(\tau)})^2}{2({\sigma_j}^{(\tau)})^2}\right\} + \text{constant} \tag{A.1.29}$$

We can now perform the maximization (A.1.15). For the parameters $\mu_j$ and $\sigma_j$ the maximization is straightforward:

$$\mu_j^{(\tau+1)} = \frac{\sum_{i=1}^{N} p(j|z_i) z_i}{\sum_{i=1}^{N} p(j|z_i)} \tag{A.1.30}$$

$$\left(\sigma_j^{(\tau+1)}\right)^2 = \frac{1}{n}\frac{\sum_{i=1}^{N} p(j|z_i)(z_i - \mu_j^{(\tau+1)})^2}{\sum_{i=1}^{N} p(j|z_i)} \tag{A.1.31}$$

For the mixing parameters the procedure is more complex [20] and returns:

$$\pi(j)^{(\tau+1)} = \frac{1}{N}\sum_{i=1}^{N} p(j|z_i) \tag{A.1.32}$$

where $p(j|z_i)$ is computed as in (A.1.25).

## A.2 K-means clustering

The *K-means* algorithm partitions a collection of $N$ vectors $x_i$, $i = 1, \ldots, N$, into $K$ groups $G_k$, $k = 1, \ldots, K$, and finds a cluster center in each group such that a cost function of dissimilarity (or distance) measure is minimized. When the Euclidean distance is chosen as the dissimilarity measure between a vector $x$ in the $k^{\text{th}}$ group and the corresponding cluster center $c_k$, the cost function can be defined by

$$J = \sum_{k=1}^{K} J_k = \sum_{k=1}^{K}\sum_{x \in G_k} d(x, c_k) \tag{A.2.33}$$

where $J_k$ is the cost function within group $k$ and $d$ is a generic distance function

$$d(x, c_k) = (x - c_k)^T M (x - c_k) \tag{A.2.34}$$

where $M$ is the distance matrix. The partitioned groups are typically defined by a membership $[K \times N]$ matrix $U$, where the element $u_{ki}$ is 1 if the $i^{\text{th}}$ data point $x_i$ belongs to group $k$, and 0 otherwise. The matrix $U$ satisfies the following conditions:

$$\begin{aligned} &\sum_{k=1}^{K} u_{ki} = 1 \qquad \forall i = 1, \ldots, N \\ &\sum_{k=1}^{K} \sum_{i=1}^{N} u_{ki} = N \end{aligned} \tag{A.2.35}$$

Once the cluster centers $c_k$ are fixed, the terms $u_{ki}$ which minimize Eq. (A.2.33) are:

$$u_{ki} = \begin{cases} 1 & \text{if } d(x_i, c_k) \leq d(x_i, c_j), \text{ for each } j \neq k \\ 0 & \text{otherwise} \end{cases} \tag{A.2.36}$$

This means that $x_i$ belongs to the group $k$ if $c_k$ is the closest center among all centers.

Once the terms $u_{ki}$ are fixed, the optimal center $c_k$ that minimizes Eq. (A.2.33) is the mean of all vectors in the $k$th group:

$$c_k = \frac{1}{|G_k|} \sum_{x \in G_k} x \tag{A.2.37}$$

where $|G_k|$ is the size of $G_k$.

The K-means algorithm determines iteratively the cluster centers $c_k$ and the membership matrix $U$ using the following procedure:

1. Initialize the cluster centers $c_k$, typically by randomly selecting $K$ points among all data points.

2. Evaluate the membership matrix $U$ through Eq. (A.2.36).

3. Compute the cost (A.2.33). If it is below a certain tolerance value or if the improvement is not significant, stop and return the centers and the groups.

4. Update the cluster centers according to Eq. (A.2.37). Go to step 2.

Some final remarks should be made on the $K - means$ algorithm. As many other clustering algorithms, this technique is iterative and no guarantee of convergence to an optimum solution can be found. Also, the final performance is quite sensitive to the initial position of the cluster centers and to the number $K$ of clusters, typically fixed a priori by the designer.

## A.3 Fuzzy clustering

Bezdek [16] proposed the *fuzzy c-means clustering* (FCM) algorithm as an improvement over the K-means clustering. The FCM algorithm is based on the idea that each data should belong to a cluster to a degree specified by a membership function. The conditions expressed in Eq. (A.2.35) still hold in FCM with the only difference that a given data point can belong to several clusters with a degree $u_{ki} \in [0, 1]$.

The cost function for FCM is a generalization of Eq. (A.2.33)

$$J = \sum_{k=1}^{K} J_k = \sum_{k=1}^{K} \left( \sum_{x=1}^{N} u_{ki}^m d(x_i, c_k) \right) \tag{A.3.38}$$

where $m \in [1, \infty)$ is a weighting exponent. The necessary conditions for Eq. (A.3.38) to reach the minimum are

$$c_k = \frac{\sum_{i=1}^{N} u_{ki}^m x_i}{\sum_{i=1}^{N} u_{ki}^m} \tag{A.3.39}$$

$$u_{ki} = \frac{1}{\sum_{j=1}^{K} \left( \frac{d(x_i, c_k)}{d(x_i, c_j)} \right)^{2/(m-1)}} \tag{A.3.40}$$

The FCM algorithm is an iterated procedure based on the relations (A.3.39) and (A.3.40). These are the steps of the FCM algorithm

1. Initialize the membership matrix with random values satisfying conditions in (A.2.35)
2. Calculate $K$ fuzzy cluster centers $c_k$ using (A.3.39).
3. Compute the cost function through Eq. (A.3.38). Stop the procedure if the cost function is below a certain tolerance value or its improvement over the previous iteration is below a certain threshold.
4. Compute a new $U$ using Eq. (A.3.40). Goto step 1.

There is no guarantee that FCM converges to an optimum solution. As in the K-means case, the final performance depends on the initial location of clusters and the number of clusters.

In the clustering algorithms we presented so far, a distance metric with a constant distance matrix $M$ is used and clusters are represented by their prototypical points (e.g. centers). The first restriction is relaxed by clustering algorithms where also the term $M$ is considered as an adaptive parameter. Examples are the Gustafson-Kessel algorithm [55] and the fuzzy maximum likelihood estimates clustering [17]

Linear or nonlinear prototypes are introduced in the *fuzzy c-varieties algorithms* (FCV) [18], where distances are measured no more from points to centers but from points to $n$ dimensional linear varieties, i.e. lines, planes or hyperplanes. A major drawback of the FCV algorithm is that the linear variety is not limited in size, and thus the algorithm tends to connect collinear clusters that may be well separated. A solution to the problem is provided by the algorithm described in the following section.

## A.4 Fuzzy c-ellyptotypes

The *fuzzy c-elliptotypes* algorithm [18] measures the distance as a convex combination of the distance from the center and the distance from the linear variety:

$$d_{ki} = \varsigma d(x_i, c_k) + (1 - \varsigma) D_{ki} \tag{A.4.41}$$

where $0 < \varsigma < 1$,

$$D_{ki} = \sqrt{|x_i - c_k|^2 - \sum_{j=1}^{n} [(x_i - c_k) \star s_{kj}]} \tag{A.4.42}$$

is the orthogonal distance from $x_i$ to the $k^{\text{th}}$ linear variety, $\{s_{kj}\}$ is a tuple of linearly independent vectors spanning the $k$ variety and the operator $\star$ denotes the scalar product. These are the steps of the FCE algorithm

1. Calculate $K$ fuzzy cluster centers $c_k$ using (A.3.39)

2. Compute cluster covariance matrices through

$$\Sigma_k = \frac{\sum_{i=1}^{N} u_{ki}^m (x_i - c_k)(x_i - c_k)^T}{\sum_{i=1}^{N} u_{ki}^m} \tag{A.4.43}$$

3. Extract the $n$ principal eigenvectors from each $\Sigma_k$.

4. Compute the distances through (A.4.41).

5. Compute the cost function through Eq. (A.3.38) using the distances computed in Step 4. Stop the procedure if the cost function is below a certain tolerance value or its improvement over the previous iteration is below a certain threshold.

6. Update the $U$ matrix by Eq. (A.3.40) using the distances computed in Step 4.

# Appendix B

# Some statistical notions

## B.1 Useful relations

Some relations

$$E[(\mathbf{z}-\mu)^2] = \sigma^2 \qquad = E[\mathbf{z}^2 - 2\mu\mathbf{z} + \mu^2] = E[\mathbf{z}^2] - 2\mu E[\mathbf{z}] + \mu^2$$
$$= E[\mathbf{z}^2] - 2\mu\mu + \mu^2 = E[\mathbf{z}^2] - \mu^2$$

For $N = 2$

$$\begin{aligned} E[(\mathbf{z}_1 + \mathbf{z}_2)^2] &= E[\mathbf{z}_1^2] + E[\mathbf{z}_2^2] + 2E[\mathbf{z}_1\mathbf{z}_2] \\ &= 2E[\mathbf{z}^2] + 2\mu^2 \\ &= 4\mu^2 + 2\sigma^2 \end{aligned}$$

For $N = 3$

$$\begin{aligned} E[(\mathbf{z}_1 + \mathbf{z}_2 + \mathbf{z}_3)^2] &= E[\mathbf{z}_1^2] + E[\mathbf{z}_2^2] + E[\mathbf{z}_3^2] + 2E[\mathbf{z}_1\mathbf{z}_2] + 2E[\mathbf{z}_1\mathbf{z}_3] + 2E[\mathbf{z}_2\mathbf{z}_3] \\ &= 3E[\mathbf{z}^2] + 6\mu^2 = 9\mu^2 + 3\sigma^2 \end{aligned}$$

In general for $N$ i.i.d. $\mathbf{z}_i$, $E[(\mathbf{z}_1 + \mathbf{z}_2 + \cdots + \mathbf{z}_N)^2] = N^2\mu^2 + N\sigma^2$.

## B.2 Convergence of random variables

Let $\{\mathbf{z}_N\}$, $N = 1, 2, \ldots$, be a sequence of random variables and let $\mathbf{z}$ be another random variable. Let $F_N(\cdot)$ denote the distribution function of $\mathbf{z}_i$ and $F_{\mathbf{z}}$ the distribution of $\mathbf{z}$. We introduce the following definitions:

**Definition 2.1** (Convergence in probability)**.** We say that

$$\lim_{N\to\infty} \mathbf{z}_N = \mathbf{z} \text{ in probability} \tag{B.2.1}$$

and we note $\mathbf{z}_N \xrightarrow{P} \mathbf{z}$ if for each $\varepsilon > 0$

$$\lim_{N\to\infty} P\{|\mathbf{z}_N - \mathbf{z}| \geq \varepsilon\} = 0 \tag{B.2.2}$$

**Definition 2.2** (Convergence with probability one)**.** We say that

$$\lim_{N\to\infty} \mathbf{z}_N = \mathbf{z} \text{ with probability one (or almost surely)} \tag{B.2.3}$$

and we note $\mathbf{z}_N \xrightarrow{a.s.} \mathbf{z}$ if

$$P\{\omega : \lim_{N\to\infty} \mathbf{z}_N(\omega) = \mathbf{z}(\omega)\} = 1 \tag{B.2.4}$$

**Definition 2.3** (Convergence in $L_p$)**.** For a fixed number $p \geq 1$ we say that

$$\lim_{N\to\infty} \mathbf{z}_N = \mathbf{z} \text{ in } L_p \tag{B.2.5}$$

if

$$\lim_{N\to\infty} E\left[|\mathbf{z}_N - \mathbf{z}|^p\right] = 0 \tag{B.2.6}$$

The following theorems hold:

**Theorem 2.4.** *Convergence in $L_p$ implies convergence in probability.*

**Theorem 2.5.** *Convergence with probability one implies convergence in probability.*

Note however that convergence in probability does not imply convergence in $L_2$.

**Definition 2.6** (Convergence in distribution)**.** The sequence $\mathbf{z}_N$ converges in distribution to $\mathbf{z}$ and we note $\mathbf{z}_N \xrightarrow{D} \mathbf{z}$ if

$$\lim_{N\to\infty} F_N(z) = F(z) \tag{B.2.7}$$

for all $z$ for which $F$ is continuous.

It can be shown that

**Theorem 2.7.** *Convergence with probability implies convergence in distribution.*

Note however that convergence in distribution does not imply convergence in probability.

As a summary

$$\mathbf{z}_N \xrightarrow{a.s.} \mathbf{z} \text{ implies } \mathbf{z}_N \xrightarrow{P} \mathbf{z} \text{ implies } \mathbf{z}_N \xrightarrow{D} \mathbf{z}$$

## B.3 Limits and probability

- Let $\mathbf{z} \sim \mathcal{U}(1,2)$ and $\theta = 0$.
- Consider the two estimators (stochastic processes) $\hat{\boldsymbol{\theta}}_N^{(1)}$ and $\hat{\boldsymbol{\theta}}_N^{(2)}$ for $N \to \infty$ where

$$\hat{\boldsymbol{\theta}}_N^1 = \exp^{-\mathbf{z}N}, \qquad \hat{\boldsymbol{\theta}}_N^2 = \begin{cases} \exp^{-N} \text{ with probability } 1 - 1/N \\ 1 \text{ with probability } 1/N \end{cases}$$

- For the first estimator, all the trajectories converge to $\theta$ (strongly consistent).
- For the second process, the trajectory which does not converge has a probability decreasing to zero for $N \to \infty$ (weakly consistent).

## B.4 Expected value of a quadratic form

**Theorem 4.1.** *Given a random vector $\mathbf{z}$ of dimension $[N \times 1]$ with expected value $E[\mathbf{z}] = \mu$ and covariance matrix $Var[\mathbf{z}] = \sigma^2 I$, for a generic matrix $A$ of dimension $[N \times N]$ the following relation holds*

$$E[\mathbf{z}^T A \mathbf{z}] = \sigma^2 tr(A) + \mu^T A \mu \tag{B.4.8}$$

*where $tr(A)$ is the trace of matrix $A$.*

## B.5 The matrix inversion formula

- Let us consider the four matrices $F$, $G$, $H$ and $K$ and the matrix $F+GHK$. Assume that the inverses of the matrices $F$, $G$ and $(F+GHK)$ exist. Then

$$(F+GHK)^{-1} = F^{-1} - F^{-1}G\left(H^{-1}+KF^{-1}G\right)^{-1}KF^{-1} \tag{B.5.9}$$

- Consider the case where $F$ is a $[n\times n]$ square nonsingular matrix, $G=z$ where $z$ is a $[n\times 1]$ vector, $K=z^T$ and $H=1$. Then the formula simplifies to

$$(F+zz^T)^{-1} = F^{-1} - \frac{F^{-1}zz^TF^{-1}}{1+z^TFz}$$

where the denominator in the right hand term is a scalar.

## B.6 Proof of Eq. (5.4.22)

Since the cost is quadratic, the input uniform density is $\pi(x)=\frac{1}{4}$ in the interval $[-2,2]$, the regression function is $x^3$ and the noise is i.i.d. with unit variance, from Eq. (5.2.3) we obtain

$$R(\alpha) = \int_{\mathcal{X},\mathcal{Y}} C(y,\alpha x)p_f(y|x)\pi(x)dy\ dx \tag{B.6.10}$$

$$= \int_{x=-2}^{2}\int_{\mathcal{Y}} (y-\alpha x)^2 p_f(y|x)\pi(x)dx\ dy \tag{B.6.11}$$

$$= \int_{x=-2}^{2}\int_{\mathcal{W}} (x^3+w-\alpha x)^2 p_{\mathbf{w}}(w)\frac{1}{4}dx\ dw \tag{B.6.12}$$

$$= \frac{1}{4}[\int_{\mathcal{W}} p_{\mathbf{w}}(w)dw \int_{x=-2}^{2}(x^3-\alpha x)^2dx + \tag{B.6.13}$$

$$+\int_{x=-2}^{2}dx\int_{\mathcal{W}} w^2p_{\mathbf{w}}(w)dw + \int_{\mathcal{W}}\int_{x=-2}^{2} 2w(x^3-\alpha x)p_{\mathbf{w}}(w)dwdx] \tag{B.6.14}$$

$$= \frac{1}{4}\left[\int_{-2}^{2}(x^3-\alpha x)^2dx + 4\sigma_{\mathbf{w}}^2\right] \tag{B.6.15}$$

$$= \frac{1}{4}\int_{-2}^{2}(x^3-\alpha x)^2dx + \sigma_{\mathbf{w}}^2 \tag{B.6.16}$$

## B.7 Biasedness of the quadratic empirical risk

Consider a regression framework where $\mathbf{y}=f(x)+\mathbf{w}$, with $E[\mathbf{w}]=0$ and $\mathrm{Var}[\mathbf{w}]=\sigma_{\mathbf{w}}^2$, and $h_N$ an estimation of $f$ built on the basis of a dataset $D_N\sim\mathbf{y}$.

Let us consider the quantity

$$g_N(x) = E_{\mathbf{D}_N,\mathbf{y}}[(\mathbf{y}-h(x,\alpha(\mathbf{D}_N)))^2] = E_{\mathbf{D}_N}[E_{\mathbf{y}}[(\mathbf{y}-\mathbf{h}_N)^2]]$$

where $\mathbf{h}_N$ stands for $h(x,\alpha(\mathbf{D}_N))$. Since

$$(y-h_N)^2 = (y-f+f-h_N)^2 = (y-f)^2+(f-h_N)^2+2(y-f)(f-h_N)$$

we obtain

$$(y-f)^2+(f-h_N)^2 = (y-h_N)^2+2(y-f)(h_N-f)$$

Note that since $E_{\mathbf{y}}[\mathbf{y}] = f$

$$E_{\mathbf{y}}[(\mathbf{y} - h_N)^2] = E_{\mathbf{y}}[(\mathbf{y} - f)^2 + (f - h_N)^2 + 2(\mathbf{y} - f)(f - h_N)] = \\ = E_{\mathbf{y}}[(\mathbf{y} - f)^2 + (f - h_N)^2]$$

Since $E_{\mathbf{y}}[(\mathbf{y} - f)^2] = E_{\mathbf{D}_N}[(\mathbf{y} - f)^2]$ and

$$(y - h)^2 = (y - f + f - h)^2 = (y - f)^2 + (f - h)^2 - 2(y - f)(h - f)$$

it follows

$$E_{\mathbf{D}_N}[E_{\mathbf{y}}[(\mathbf{y} - \mathbf{h}_N)^2]] = \\ = E_{\mathbf{D}_N}[E_{\mathbf{y}}[(\mathbf{y} - f)^2] + (f - \mathbf{h}_N)^2] = E_{\mathbf{D}_N}[(\mathbf{y} - f)^2 + (f - \mathbf{h}_N)^2] = \\ = E_{\mathbf{D}_N}[(\mathbf{y} - \mathbf{h}_N)^2 + 2(\mathbf{y} - f)(\mathbf{h}_N - f)]$$

By averaging over the $\mathcal{X}$ domain we obtain

$$\text{MISE} = E_{\mathbf{D}_N,\mathbf{y},\mathbf{x}}[(\mathbf{y} - \mathbf{h}(x, \alpha(\mathbf{D}_N)))^2]] = \\ = E_{\mathbf{D}_N,\mathbf{x}}[(\mathbf{y} - \mathbf{h}_N)^2] + 2E_{\mathbf{D}_N,\mathbf{x}}[(\mathbf{y} - f)(\mathbf{h}_N - f)] = E_{\mathbf{D}_N}[\widehat{\text{MISE}}_{\text{emp}}] + 2\text{Cov}[\mathbf{h}_N, \mathbf{y}]$$

where $\text{Cov}[\mathbf{h}_N, \mathbf{y}] = E_{\mathbf{D}_N,\mathbf{x}}[(\mathbf{y} - f)(\mathbf{h}_N - f)]$ and $\widehat{\text{MISE}}_{\text{emp}}$ is the quantity (5.2.5) for a quadratic error loss. This means we have to add a *covariance penalty term* to the apparent error $\widehat{\text{MISE}}_{\text{emp}}$ in order to have an unbiased estimate of MISE.

Suppose that $\mathbf{h}_N$ is a linear estimator, i.e.

$$\mathbf{h}_N = S\mathbf{y}$$

where $S$ is known as the smoother matrix. Note that in least-square regression $S$ is the Hat matrix $H = X(X^T X)^{-1} X^T$.

In the linear case, since $H^T = H$

$$\sum_{i=1}^{N} \text{Cov}[\mathbf{h}(x_i, \boldsymbol{\alpha}_N), y_i] = \text{Cov}[\mathbf{h}_N, \mathbf{y}] = E_{\mathbf{D}_N}[(\mathbf{y} - f)^T (H\mathbf{y} - f)] = \\ = E_{\mathbf{D}_N}[\mathbf{y}^T H\mathbf{y} - \mathbf{y}^T f - f^T H\mathbf{y} - \mathbf{y}^T H^T f] = E_{\mathbf{D}_N}[\mathbf{y}^T H(\mathbf{y} - f)] = \\ = \sigma^2 \text{tr}(H) + f^T Hf - f^T Hf = \sigma^2 \text{tr}(H)$$

where $\text{tr}(H)$ is the trace of the matrix $H$.

In this case we obtain the $C_p$ formula (6.7.28).

# Appendix C

# Kernel functions

A kernel function $K$ is a nonnegative function

$$K : \Re^n \times \Re^n \times \Re^+ \to \Re^+$$

where the first argument is a $n$-dimensional input, the second argument is typically called the *center* and the third argument is called *width* or *bandwidth*. Once a distance function between the input and the center

$$d : \Re^n \times \Re^n \to \Re^+ \tag{C.0.1}$$

is defined, the kernel function can be expressed as a function

$$K : \Re^+ \times \Re^+ \to \Re^+ \tag{C.0.2}$$

of the distance $d$ and the bandwidth parameter. The maximum value of a kernel function is located at zero distance and the function decays smoothly as the distance increases.

Here you have some examples of kernel functions.

**Inverse distance:**

$$K(d, B) = \frac{1}{(d/B)^p} \tag{C.0.3}$$

This function goes to infinity as the distance approaches zero.

**Corrected inverse distance:**

$$K(d, B) = \frac{1}{1 + (d/B)^p} \tag{C.0.4}$$

**Gaussian kernel:**

$$K(d, B) = \exp\left(-\frac{d^2}{B^2}\right) \tag{C.0.5}$$

**Exponential kernel:**

$$K(d, B) = \exp\left(-\left|\frac{d}{B}\right|\right) \tag{C.0.6}$$

**Quadratic or Epanechnikov kernel:**

$$K(d, B) = \begin{cases} \left(1 - \frac{d}{B}^2\right) & \text{if } |d| < B \\ 0 & \text{otherwise} \end{cases} \tag{C.0.7}$$

**Tricube kernel:**

$$K(d,B) = \begin{cases} \left(1 - \left|\frac{d}{B}\right|^3\right)^3 & \text{if } |d| < B \\ 0 & \text{otherwise} \end{cases} \tag{C.0.8}$$

**Uniform kernel:**

$$K(d,B) = \begin{cases} 1 & \text{if } |d| < B \\ 0 & \text{otherwise} \end{cases} \tag{C.0.9}$$

**Triangular kernel:**

$$K(d,B) = \begin{cases} 1 - \left|\frac{d}{B}\right| & \text{if } |d| < B \\ 0 & \text{otherwise} \end{cases} \tag{C.0.10}$$

# Appendix D

# Datasets

## D.1 USPS dataset

The dataset and this description is made available on
`http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/datasets/zip.info`.

The dataset refers to numeric data obtained from the scanning of handwritten digits from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations; the images here have been deslanted and size normalized, resulting in 16 x 16 grayscale images (Le Cun et al., 1990).

There are 7291 training observations and 2007 test observations, distributed as follows:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Train | 1194 | 1005 | 731 | 658 | 652 | 556 | 664 | 645 | 542 | 644 | 7291 |
| Test | 359 | 264 | 198 | 166 | 200 | 160 | 170 | 147 | 166 | 177 | 2007 |

or as proportions:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Train | 0.16 | 0.14 | 0.1 | 0.09 | 0.09 | 0.08 | 0.09 | 0.09 | 0.07 | 0.09 |
| Test | 0.18 | 0.13 | 0.1 | 0.08 | 0.10 | 0.08 | 0.08 | 0.07 | 0.08 | 0.09 |

The test set is notoriously "difficult", and a 2.5% error rate is excellent. This is a notorious example of multiclass classifiction task where $\mathbf{y}\ in\{0, 1, \ldots, 9\}$ and the inputs are real vectors.

## D.2 Golub dataset

This is the microarray data used by Golub et al. in [54]. It contains the genome expressions of $n = 7129$ gene probes of $N = 72$ patients, for which $V = 11$ phenotype variables are measured. The samples were assayed using Affymetrix Hgu6800 chips.

This is an example of binary classifiction task where $\mathbf{y} \in \{ALL, AML\}$ and ALL stands for lymphoblastic leukemia while AML stands for acute myeloid leukemia. The inputs are real vectors representing the gene expression of a patient.

# Bibliography


[1] D. W. Aha. Incremental, instance-based learning of independent and graded concept descriptions. In *Sixth International Machine Learning Workshop*, pages 387–391, San Mateo, CA, 1989. Morgan Kaufmann.

[2] D. W. Aha. *A Study of Instance-Based Algorithms for Supervised Learning Tasks: Mathematical, Empirical and Psychological Observations*. PhD thesis, University of California, Irvine, Department of Information and Computer Science, 1990.

[3] D. W. Aha. Editorial of special issue on lazy learning. *Artificial Intelligence Review*, 11(1–5):1–6, 1997.

[4] H. Akaike. Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mechanics*, 21:243–247, 1969.

[5] D. M. Allen. The relationship between variable and data augmentation and a method of prediction. *Technometrics*, 16:125–127, 1974.

[6] B. D. O. Anderson and M. Deistler. Identifiability in dynamic errors-in-variables models. *Journal of Time Series Analysis*, 5:1–13, 1984.

[7] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1–5):11–73, 1997.

[8] R. Babuska. *Fuzzy Modeling and Identification*. PhD thesis, Technische Universiteit Delft, 1996.

[9] R. Babuska and H. B. Verbruggen. Fuzzy set methods for local modelling and identification. In R. Murray-Smith and T. A. Johansen, editors, *Multiple Model Approaches to Modeling and Control*, pages 75–100. Taylor and Francis, 1997.

[10] A. R. Barron. Predicted squared error: a criterion for automatic model selection. In S. J. Farlow, editor, *Self-Organizing Methods in Modeling*, volume 54, pages 87–103, New York, 1984. Marcel Dekker.

[11] W. G. Baxt. Improving the accuracy of an artificial neural network using multiple differently trained networks. *Neural Computation*, 4:772–780, 1992.

[12] M. G. Bello. Enhanced training algorithms, and integrated training/architecture selection for multilayer perceptron networks. *IEEE Transactions on Neural Networks*, 3(6):864–875, 1992.

[13] H. Bersini and G. Bontempi. Fuzzy models viewed as multi-expert networks. In *IFSA '97 (7th International Fuzzy Systems Association World Congress, Prague)*, pages 354–359, Prague, 1997. Academia.

[14] H. Bersini and G. Bontempi. Now comes the time to defuzzify the neuro-fuzzy models. *Fuzzy Sets and Systems*, 90(2):161–170, 1997.

[15] H. Bersini, G. Bontempi, and C. Decaestecker. Comparing rbf and fuzzy inference systems on theoretical and practical basis. In F. Fogelman-Soulie' and P. Gallinari, editors, *ICANN '95,International Conference on Artificial Neural Networks*, pages 169–174, 1995.

[16] J. C. Bezdek. *Fuzzy Mathematics in Pattern Classification*. PhD thesis, Applied Math. Center, Cornell University, Ithaca, 1973.

[17] J. C. Bezdek and J. C. Dunn. Optimal fuzzy partition: a heuristic for estimating the parameters in a mixture of normal distributions. *IEEE Transactions on Computers*, 24:835–838, 1975.

[18] J.C. Bezdek, C. Coray, R. Gunderson, and J. Watson. Detection and characterization of cluster substructure. *SIAM Journal of Applied Mathematics*, 40(2):339–372, 1981.

[19] M. Birattari, G. Bontempi, and H. Bersini. Lazy learning meets the recursive least-squares algorithm. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *NIPS 11*, pages 375–381, Cambridge, 1999. MIT Press.

[20] C. M. Bishop. *Neural Networks for Statistical Pattern Recognition*. Oxford University Press, Oxford, UK, 1994.

[21] G. Bontempi and H. Bersini. Identification of a sensor model with hybrid neuro-fuzzy methods. In A. B. Bulsari and S. Kallio, editors, *Neural Networks in Engineering systems (Proceedings of the 1997 International Conference on Engineering Applications of Neural Networks (EANN '97), Stockolm, Sweden)*, pages 325–328, 1997.

[22] G. Bontempi, M. Birattari, and H. Bersini. Lazy learning for modeling and control design. *International Journal of Control*, 72(7/8):643–658, 1999.

[23] G. Bontempi, M. Birattari, and H. Bersini. A model selection approach for local learning. *Artificial Intelligence Communications*, 121(1), 2000.

[24] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

[25] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[26] V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory, and Methods*. Wiley, New York, 1998.

[27] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:829–836, 1979.

[28] W. S. Cleveland and S. J. Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of American Statistical Association*, 83:596–610, 1988.

[29] W. S. Cleveland and C. Loader. Smoothing by local regression: Principles and methods. *Computational Statistics*, 11, 1995.

[30] T. Cover and P. Hart. Nearest neighbor pattern classification. *Proc. IEEE Trans. Inform. Theory*, pages 21–27, 1967.

[31] P. Craven and G. Wahba. Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numer. Math.*, 31:377–403, 1979.

[32] G. Cybenko. Just-in-time learning and estimation. In S. Bittanti and G. Picci, editors, *Identification, Adaptation, Learning. The Science of Learning Models from data*, NATO ASI Series, pages 423–434. Springer, 1996.

[33] Peter Dalgaard. *Introductory statistics with R.* Springer, 2002.

[34] A. Dean and D. Voss. *Design and Analysis of Experiments.* Springer Verlag, New York, NY, USA, 1999.

[35] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, B*, 39(1):1–38, 1977.

[36] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer Verlag, 1996.

[37] N. R. Draper and H. Smith. *Applied Regression Analysis.* John Wiley and Sons, New York, 1981.

[38] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1976.

[39] B. Efron. Bootstrap methods: Another look at the jacknife. *Annals of Statistics*, pages 1–26, 1979.

[40] B. Efron. *The Jacknife, the Bootstrap and Other Resampling Plans.* SIAM, 1982. Monograph 38.

[41] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap.* Chapman and Hall, New York, NY, 1993.

[42] B. Efron and R. J. Tibshirani. Cross-validation and the bootstrap: estimating the error rate of a prediction rule. Technical report, Stanford University, 1995.

[43] J. Fan and I. Gijbels. Variable bandwidth and local linear regression smoothers. *The Annals of Statistics*, 20(4):2008–2036, 1992.

[44] J. Fan and I. Gijbels. Adaptive order polynomial fitting: bandwidth robustification and bias reduction. *J. Comp. Graph. Statist.*, 4:213–227, 1995.

[45] J. Fan and I. Gijbels. *Local Polynomial Modelling and Its Applications.* Chapman and Hall, 1996.

[46] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, November 1996.

[47] V. Fedorov. *Theory of Optimal Experiments.* Academic Press, 1972.

[48] F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, 2004.

[49] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[50] J. H. Friedman. Flexible metric nearest neighbor classification. Technical report, Stanford University, 1994.

[51] A. Gelman. *Bayesian Data Analysis*. Chapman and Hall, 2004.

[52] Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an em approach. In J. D. Cowan, G. T. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 120–127, San Mateo, CA, 1994. Morgan Kaufmann.

[53] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[54] T. R. Golub, D. K. Slonin, P. Tamayo, C. Huard, and M. Gaasenbeek. Molecular clssification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[55] D. E. Gustafson and W. C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proc. IEEE CDC*, pages 761–776, San Diego, CA,USA, 1979.

[56] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[57] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lotfi A. Zadeh. *Feature Extraction: Foundations and Applications*. Springer-Verlag New York, Inc., 2006.

[58] D. J. Hand. *Discrimination and classification*. John Wiley, New York, 1981.

[59] W. Hardle and J. S. Marron. Fast and simple scatterplot smoothing. *Comp. Statist. Data Anal.*, 20:1–17, 1995.

[60] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, London, UK, 1990.

[61] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–615, 1996.

[62] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, 2001.

[63] J. S. U. Hjorth. *Computer Intensive Statistical Methods*. Chapman and Hall, 1994.

[64] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of American Statistical Association*, 58:13–30, 1963.

[65] P. J. Huber. *Robust Statistics*. Wiley, New York, 1981.

[66] A. K. Jain, R. C. Dubes, and C. Chen. Bootstrap techniques for error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:628–633, 1987.

[67] J.-S. R. Jang. Anfis: Adaptive-network-based fuzzy inference systems. *IEEE Transactions on Fuzzy Systems*, 23(3):665–685, 1993.

[68] J. S. R. Jang, C. T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing*. Matlab Curriculum Series. Prentice Hall, 1997.

[69] E.T. Jaynes. *Probability theory : the logic of science*. Cambridge University Press, 2003.

[70] T. A. Johansen and B. A. Foss. Constructing narmax models using armax models. *International Journal of Control*, 58:1125–1153, 1993.

[71] M. C. Jones, J. S. Marron, and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *Journal of American Statistical Association*, 90, 1995.

[72] V. Y. Katkovnik. Linear and nonlinear methods of nonparametric regression analysis. *Soviet Automatic Control*, 5:25–34, 1979.

[73] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[74] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of IJCAI-95*, 1995. available at http://robotics.stanford.edu/users/ronnyk/ronnyk-bib.html.

[75] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[76] A. N. Kolmogorov. *Foundations of Probability*. Berlin, 1933.

[77] J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.

[78] R. J.A. Little and D. B. Rubin. *Statistical analysis with missing data*. Wiley, 2002.

[79] L. Ljung. *System identification: Theory for the User*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[80] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison Wesley, Reading, MA, 1984.

[81] C. Mallows. Discussion of a paper of beaton and tukey. *Technometrics*, 16:187–188, 1974.

[82] C. L. Mallows. Some comments on $c_p$. *Technometrics*, 15:661, 1973.

[83] O. Maron and A. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11(1–5):193–225, 1997.

[84] J. Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In J. Moody, Hanson, and Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 847–854, Palo Alto, 1992. Morgan Kaufmann.

[85] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[86] A. W. Moore, D. J. Hill, and M. P. Johnson. An empirical investigation of brute force to choose features, smoothers and function approximators. In S. Janson, S. Judd, and T. Petsche, editors, *Computational Learning Theory and Natural Learning Systems*, volume 3. MIT Press, Cambridge, MA, 1992.

[87] R. Murray-Smith. *A local model network approach to nonlinear modelling*. PhD thesis, Department of Computer Science, University of Strathclyde, Strathclyde, UK, 1994.

[88] R. Murray-Smith and T. A. Johansen. Local learning in local model networks. In R. Murray-Smith and T. A. Johansen, editors, *Multiple Model Approaches to Modeling and Control*, chapter 7, pages 185–210. Taylor and Francis, 1997.

[89] R. H. Myers. *Classical and Modern Regression with Applications*. PWS-KENT Publishing Company, Boston, MA, second edition, 1994.

[90] E. Nadaraya. On estimating regression. *Theory of Prob. and Appl.*, 9:141–142, 1964.

[91] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, 1991.

[92] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[93] J. Pearl. *Causality*. Cambridge University Press, 2000.

[94] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, 2005.

[95] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142. Chapman and Hall, 1993.

[96] D. Plaut, S. Nowlan, and G. E. Hinton. Experiments on learning by back propagation. Technical Report CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1986.

[97] M. J. D. Powell. *Algorithms for Approximation*, chapter Radial Basis Functions for multivariable interpolation: a review, pages 143–167. Clarendon Press, Oxford, 1987.

[98] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992. Second ed.

[99] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[100] J. R. Quinlan. Simplyfying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[101] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2004. ISBN 3-900051-07-0.

[102] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[103] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics*, 27:832–837, 1956.

[104] D. E. Rumelhart, G. E. Hinton, and R. K. Williams. Learning representations by backpropagating errors. *Nature*, 323(9):533–536, 1986.

[105] Y. Saeys, I. Inza, and P. Larranaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23:2507–2517, 2007.

[106] R. E. Schapire. *Nonlinear Estimation and Classification*, chapter The boosting approach to machine learning: An overview. Springer,.

[107] D. W. Scott. *Multivariate density estimation.* Wiley, New York, 1992.

[108] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1987.

[109] C. Stone. Consistent nonparametric regression. *The Annals of Statistics*, 5:595–645, 1977.

[110] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36(1):111–147, 1974.

[111] M. Stone. An asymptotic equivalence of choice of models by cross-validation and akaike's criterion. *Journal of Royal Statistical Society, Series B*, 39:44–47, 1977.

[112] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1):116–132, 1985.

[113] H. Tijms. *Understanding probability.* Cambridge, 2004.

[114] V. N. Vapnik. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems*, volume 4, Denver, CO, 1992.

[115] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, NY, 1995.

[116] V. N. Vapnik. *Statistical Learning Theory.* Springer, 1998.

[117] W. N. Venables and D. M. Dmith. *An Introduction to R.* Network Theory, 2002.

[118] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon. Accelerating the convergence of the back-propagation method. *Biological Cybernetics*, 59:257–263, 1988.

[119] L. Wasserman. *All of statistics.* Springer, 2004.

[120] G. Watson. Smooth regression analysis. *Sankhya, Series*, A(26):359–372, 1969.

[121] S. M. Weiss and C. A. Kulikowski. *Computer Systems that learn.* Morgan Kaufmann, San Mateo, California, 1991.

[122] D. H. Wolpert and R. Kohavi. Bias plus variance decomposition for zero-one loss functions. In *Prooceedings of the 13th International Conference on Machine Learning*, pages 275–283, 1996.

[123] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.