

Statistical foundations of machine learning: the handbook

Gianluca Bontempi

Machine Learning Group
Computer Science Department
ULB, Université Libre de Bruxelles,
Brussels, Belgium
mlg.ulb.ac.be

February 10, 2023

And indeed all things that are known have number. For it is not possible that anything whatsoever be understood or known without this.

Philolaus, 400 BC

Not everything that can be counted counts, and not everything that counts can be counted.

W. B. Cameron, 1963

Preface to the handbook edition

The handbook¹ is dedicated to all students interested in machine learning who are not content with running some lines of (deep learning) code but who are eager to learn about this discipline's assumptions, limitations, and perspectives. When I was a student, my dream was to become an AI researcher and save humankind with intelligent robots. For several reasons, I abandoned such ambitions (but you never know). In exchange, I really got into machine learning by realising that this discipline is intimately related to the scientific process transforming observations into knowledge.

The first version of this handbook was made publicly available in 2004 with two objectives and one ambition. The first objective was to provide a handbook to ULB students since I was (and still am) strongly convinced that a decent course should come with a decent handbook. The second objective was to group together all the material that I consider fundamental (or at least essential) for a Ph.D. student to undertake a thesis in my lab. At that time, there were already plenty of excellent machine learning reference books. However, most of the existing work did not sufficiently acknowledge what machine learning owes to statistics and concealed (or did not make explicit enough, notably because of incomplete or implicit notation) important assumptions underlying the process of inferring models from data.

The ambition was to make a free academic reference on the foundations of machine learning available on the web. There are several reasons for providing free access to this work: I am a civil servant in an institution that already takes care of my salary; most of the material is not original (though its organisation, notation definition, exercises, code and structure represent the primary added value of the author); a frequent update is important in a book on such a hot topic; in many parts of the world access to expensive textbooks or reference material is still difficult for the majority of students; most of the knowledge underlying this book was obtained by the author thanks to free (or at least non-charged) references and, last but not least, education seems to be the last societal domain where a communist approach may be as effective as rewarding. Personally, I would be delighted if this book could be used to facilitate the access of underfunded educational and research communities to state-of-the-art scientific notions.

Though machine learning was already a hot topic at the end of the 20th century, nowadays, it is definitely surrounded by a lot of hype and excitement. The number of publications describing or using a machine learning approach in the last decades is countless, making it impossible to address the heterogeneity of the domain in a single book. Therefore, it is interesting to check how much material from the first edition is still useful: reassuringly enough, the more the nature of the content is fundamental, the less it is prone to obsolescence. Nevertheless, a lot of new

¹An extended book version is available at <https://leanpub.com/statisticalfoundationsofmachinelearning>

things (not only deep learning) happened in the domain, and, more specifically, I realised the importance of some fundamental concepts that were neglected in the first edition.

In particular, I added a number of exercises, R scripts, and Shiny dashboards to visualise and illustrate (sometimes too abstract) probabilistic and estimation notions. In this sense, I am convinced that the adoption of Monte Carlo simulation to introduce probabilistic concepts should be a more common habit in introductory statistics classes.

For sure, I am strongly indebted to a lot of authors and their publications. I hope I acknowledged them adequately in the bibliography. If I did not give enough credit to some of the existing works, please do not hesitate to contact me. Last but not least, the book is dedicated to all my ULB students and MLG researchers in whom I have tried for many years to inculcate complex concepts of statistical learning. Their eyes staring at my hand-waving, while I was trying to elucidate some abstruse notions, were the best indicators of how to adapt, select and improve the book's content.

To all those who want to send a note or continue to follow my machine learning journey, see you on my blog <https://datascience741.wordpress.com>.

Software

Several scripts and dashboards are used in the main text to illustrate statistical and machine learning notions. All the scripts have been implemented in R, Shiny and are available in the R github package `gbcode`. Information about how to install and use the companion package are available in Appendix G and H.

Notes for instructors

The handbook is used as a basis for a semester Master course in ULB (Brussels, Belgium) addressed to computer scientists, computer engineers, bioinformaticians and data scientists. The handbook contains R scripts, exercises and programming home-works. The adoption of such textbook for teaching is warmly encouraged (once acknowledged): the related class material (e.g. slides) can be asked by contacting the author.

Acknowledgements

Though the book is not peer-reviewed, the added value of writing a handbook for students and researchers is that they are typically very careful readers and willing to pinpoint mistakes, inconsistencies, bad English and (a lot of) typos. First, I would like to thank (in random order) the MLG researchers who sent me very useful comments: Abhilash Miranda, Yann-ael Le Borgne, Souhaib Ben Taieb, Jacopo De Stefani, Patrick Meyer, Olivier Caelen, Liran Lerman. Thanks as well to the following students and readers (in random order) for their comments and remarks: Robin de Haes, Mourad Akandouch, Zheng Liangliang, Olga Ibanez Solé, Maud Destree, Wolf De Wulf, Dieter Vandesande, Miro-Manuel Matagne, Henry Morgan, Pascal Tribel. A big thank to all of you! And do not hesitate to fill the form <https://forms.gle/YzPK3zvNp3rXByCQA> if you have comments or remarks!

Contents

Index	4
1 Introduction	13
1.1 Notations	20
2 Foundations of probability	25
2.1 The random model of uncertainty	25
2.1.1 Axiomatic definition of probability	27
2.1.2 Sample space and random variables	27
2.1.3 Visualisation of probability measures	28
2.2 The definition of probability	29
2.2.1 Symmetrical definition of probability	29
2.2.2 Frequentist definition of probability	30
2.2.2.1 The Law of Large Numbers	30
2.2.3 Degree of belief and probability	31
2.2.4 A pragmatic approach to probability	32
2.3 Independence and conditional probability	32
2.3.1 Independent variables	35
2.4 The chain rule	35
2.5 The law of total probability	36
2.6 The Bayes' theorem	36
2.7 Direct and inverse conditional probability	38
2.8 Probability as Odds	42
2.9 Logics and probabilistic reasoning	43
2.10 Random numeric variables	44
2.11 Discrete random variables	45
2.11.1 Parametric probability function	46
2.11.2 Expected value, variance and standard deviation of a discrete r.v.	46
2.11.3 Entropy of a discrete r.v.	49
2.11.4 Kullback-Leibler divergence and cross-entropy	49
2.12 Continuous random numeric variables	50
2.12.1 Mean, variance, moments of a continuous r.v.	51
2.12.2 Univariate Normal (or Gaussian) distribution	52
2.13 Joint probability	53
2.13.1 Marginal and conditional probability	54
2.13.2 Bayes' theorem for discrete and continuous variables	55
2.13.3 Independence	55
2.13.4 Chain rule	57
2.13.5 Conditional independence	57
2.14 Bivariate continuous distribution	59
2.14.1 Covariance and correlation	61

2.15	Normal distribution: the multivariate case	63
2.15.1	Bivariate normal distribution	64
2.15.2	Conditional distribution in the normal multivariate distribution	65
2.15.3	Gaussian mixture distribution	65
2.15.4	Linear transformations of Gaussian variables	66
2.15.5	Entropy in the continuous case	66
2.15.5.1	Kullback-Leibler divergence	67
2.15.5.2	Joint and conditional entropy	67
2.15.6	Mutual information	68
2.15.6.1	Conditional mutual information	69
2.15.6.2	Joint mutual information	69
2.15.7	Partial correlation coefficient	70
2.16	Functions of random variables	70
2.17	Monte Carlo approximation of expectation	71
2.18	Linear combinations of r.v.	71
2.18.1	The sum of i.i.d. random variables	72
2.19	Conclusion	72
2.20	Exercises	73
3	Parametric estimation and testing	79
3.1	Classical approach	79
3.1.1	Point estimation	81
3.2	Empirical distributions	81
3.3	Plug-in principle to define an estimator	82
3.3.1	Sample average	83
3.3.2	Sample variance	83
3.4	Sampling distribution	83
3.4.1	Monte Carlo simulation of the sampling distribution	84
3.5	The assessment of an estimator	85
3.5.1	Bias and variance	85
3.5.2	Estimation and the game of darts	86
3.5.3	Bias and variance of $\hat{\mu}$	86
3.5.4	Bias of the estimator $\hat{\sigma}^2$	87
3.5.5	A tongue-twister exercise	89
3.5.6	Bias/variance decomposition of MSE	89
3.5.7	Consistency	90
3.5.8	Efficiency	90
3.6	The Hoeffding's inequality	91
3.7	Sampling distributions for Gaussian r.v.s	91
3.8	The principle of maximum likelihood	92
3.8.1	Maximum likelihood computation	93
3.8.2	Maximum likelihood in the Gaussian case	94
3.8.3	Cramer-Rao lower bound	95
3.8.4	Properties of m.l. estimators	96
3.9	Interval estimation	97
3.9.1	Confidence interval of μ	97
3.10	Combination of two estimators	100
3.10.1	Combination of m estimators	100
3.10.1.1	Linear constrained combination	101
3.11	Regularisation of an estimator	101
3.12	Testing hypothesis	102
3.12.1	Types of hypothesis	103
3.12.2	Types of statistical test	103
3.12.3	Pure significance test	103

3.12.4	Tests of significance	104
3.12.5	Hypothesis testing	105
3.12.6	The hypothesis testing procedure	106
3.12.7	Choice of test	106
3.12.8	UMP level- α test	107
3.12.9	Likelihood ratio test	108
3.13	Parametric tests	108
3.13.1	z-test (single and one-sided)	109
3.13.2	t-test: single sample and two-sided	110
3.13.3	Statistical significance and effect size	111
3.14	A posteriori assessment of a test	111
3.15	Conclusion	113
3.16	Exercises	114
4	Nonparametric estimation and testing	119
4.1	Nonparametric methods	119
4.2	Estimation of arbitrary statistics	120
4.3	Jackknife	121
4.3.1	Jackknife estimation	121
4.4	Bootstrap	123
4.4.1	Bootstrap sampling	123
4.4.2	Bootstrap estimate of the variance	123
4.4.3	Bootstrap estimate of bias	125
4.4.4	Bootstrap confidence interval	125
4.4.5	The bootstrap principle	126
4.5	Randomisation tests	127
4.5.1	Randomisation and bootstrap	128
4.6	Permutation test	128
4.7	Considerations on nonparametric tests	129
4.8	Exercises	130
5	Statistical supervised learning	131
5.1	Introduction	131
5.2	Estimating dependencies	133
5.3	Dependency and classification	136
5.3.1	The Bayes classifier	138
5.3.2	Inverse conditional distribution	139
5.4	Dependency and regression	142
5.5	Assessment of a learning machine	142
5.5.1	An illustrative example	144
5.6	Generalisation error	148
5.6.1	The decomposition of the generalisation error in regression .	148
5.6.2	The decomposition of the generalisation error in classification	151
5.6.3	Stability and generalisation	152
5.7	The hypothesis-based vs the algorithm-based approach	153
5.8	The supervised learning procedure	154
5.9	Validation techniques	155
5.9.1	The resampling methods	156
5.10	Concluding remarks	158
5.11	Exercises	159

6 The machine learning procedure	161
6.1 Introduction	161
6.2 Problem formulation	162
6.3 Experimental design	162
6.4 Data pre-processing	163
6.5 The dataset	164
6.6 Parametric identification	165
6.6.1 Error functions	165
6.6.2 Parameter estimation	166
6.6.2.1 The linear least-squares method	166
6.6.2.2 Iterative search methods	166
6.6.2.3 Gradient-based methods	167
6.6.2.4 Gradient descent	167
6.6.2.5 The Newton method	169
6.6.2.6 The Levenberg-Marquardt algorithm	170
6.6.3 Online gradient-based algorithms	172
6.6.4 Alternatives to gradient-based methods	172
6.7 Regularisation	173
6.8 Structural identification	174
6.8.1 Model generation	175
6.8.2 Validation	175
6.8.2.1 Testing	176
6.8.2.2 Holdout	176
6.8.2.3 Cross-validation in practice	176
6.8.2.4 Bootstrap in practice	177
6.8.2.5 Complexity based criteria	177
6.8.2.6 A comparison of validation methods	179
6.8.2.7 Residual diagnostic	179
6.8.3 Model selection criteria	180
6.8.3.1 The winner-takes-all approach	180
6.8.3.2 The combination of estimators approach	181
6.9 Partition of dataset in training, validation and test	181
6.10 Evaluation of a regression model	182
6.11 Evaluation of a binary classifier	183
6.11.1 Balanced Error Rate	183
6.11.2 Specificity and sensitivity	184
6.11.3 Additional assessment quantities	184
6.11.4 Random classifier	185
6.11.5 Receiver Operating Characteristic curve	185
6.11.6 Precision-recall and lift curves	186
6.12 Multi-class problems	187
6.13 Concluding remarks	188
6.14 Exercises	190
7 Linear approaches	193
7.1 Linear regression	193
7.1.1 The univariate linear model	193
7.1.2 Least-squares estimation	194
7.1.3 Maximum likelihood estimation	196
7.1.4 Partitioning the variability	196
7.1.5 Test of hypotheses on the regression model	197
7.1.5.1 The t-test	197
7.1.6 Interval of confidence	198
7.1.7 Variance of the response	198

CONTENTS	9
7.1.8 Coefficient of determination	199
7.1.9 Multiple linear dependence	199
7.1.10 The multiple linear regression model	199
7.1.11 The least-squares solution	200
7.1.12 Least-squares and non full-rank configurations	201
7.1.13 Properties of least-squares estimators	201
7.1.14 Variance of the prediction	202
7.1.15 The HAT matrix	202
7.1.16 Generalisation error of the linear model	203
7.1.16.1 The expected empirical error	203
7.1.16.2 The PSE and the FPE	205
7.1.17 The PRESS statistic	208
7.1.18 Dual linear formulation	209
7.1.19 The weighted least-squares	210
7.1.20 Recursive least-squares	210
7.1.20.1 1st Recursive formulation	211
7.1.20.2 2nd Recursive formulation	212
7.1.20.3 RLS initialisation	212
7.1.20.4 RLS with forgetting factor	212
7.2 Linear approaches to classification	213
7.2.1 Linear discriminant analysis	214
7.2.1.1 Discriminant functions in the Gaussian case	215
7.2.1.2 Uniform prior case	216
7.2.1.3 LDA parameter identification	218
7.2.2 Perceptrons	218
7.2.3 Support vector machines	220
7.3 Generalised linear models	224
7.4 Conclusion	225
7.5 Exercises	225
8 Nonlinear approaches	229
8.1 Nonlinear regression	231
8.1.1 Artificial neural networks	232
8.1.1.1 Feed-forward architecture	232
8.1.1.2 Back-propagation	234
8.1.1.3 Back-propagation in matrix notation	236
8.1.1.4 Approximation properties	239
8.1.2 From shallow to deep learning architectures	240
8.1.3 From global modelling to divide-and-conquer	242
8.1.4 Classification and Regression Trees	243
8.1.4.1 Learning in Regression Trees: parameter identification	244
8.1.4.2 Learning in Regression Trees: structural identification	244
8.1.5 Basis Function Networks	248
8.1.6 Radial Basis Functions	248
8.1.7 Local Model Networks	248
8.1.8 Neuro-Fuzzy Inference Systems	249
8.1.9 Learning in Basis Function Networks	251
8.1.9.1 Parametric identification: basis functions	252
8.1.9.2 Parametric identification: local models	252
8.1.9.3 Structural identification	254
8.1.10 From modular techniques to local modelling	254
8.1.11 Local modelling	256
8.1.11.1 Nadaraya-Watson estimators	256
8.1.11.2 Higher order local regression	258

8.1.11.3	Parametric identification in local regression	258
8.1.11.4	Structural identification in local regression	261
8.1.11.5	The kernel function	261
8.1.11.6	The local polynomial order	261
8.1.11.7	The bandwidth	262
8.1.11.8	The distance function	263
8.1.11.9	The selection of local parameters	264
8.1.11.10	Bias/variance decomposition of the local constant model	265
8.2	Nonlinear classification	267
8.2.1	Direct estimation via regression techniques	267
8.2.1.1	The nearest-neighbour classifier	267
8.2.2	Direct estimation via cross-entropy	270
8.2.3	Density estimation via the Bayes theorem	272
8.2.3.1	Naive Bayes classifier	272
8.2.3.2	SVM for nonlinear classification	274
8.3	Is there a best learner?	275
8.4	Conclusions	278
8.5	Exercises	278
9	Model averaging approaches	295
9.1	Stacked regression	295
9.2	Bagging	296
9.3	Boosting	298
9.3.1	The Ada Boost algorithm	298
9.3.2	The arcing algorithm	300
9.3.3	Bagging and boosting	301
9.4	Random Forests	301
9.4.1	Why are Random Forests successful?	302
9.5	Gradient boosting trees	302
9.6	Conclusion	303
9.7	Exercises	303
10	Feature selection	305
10.1	Curse of dimensionality	306
10.2	Approaches to feature selection	311
10.3	Filter methods	311
10.3.1	Principal component analysis	311
10.3.1.1	PCA: the algorithm	312
10.3.2	Clustering	314
10.3.3	Ranking methods	315
10.4	Wrapping methods	316
10.4.1	Wrapping search strategies	317
10.4.2	The Cover and van Campenhout theorem	317
10.5	Embedded methods	318
10.5.1	Shrinkage methods	318
10.5.1.1	Ridge regression	319
10.5.1.2	Lasso	320
10.5.2	Kernel methods	321
10.5.3	Dual ridge regression	322
10.5.4	Kernel function	322
10.6	Similarity matrix and non numeric data	324
10.7	Averaging and feature selection	325
10.8	Information-theoretic perspective	325

CONTENTS	11
10.8.1 Relevance, redundancy and interaction	326
10.8.2 Information-theoretic filters	327
10.8.3 Information-theoretic notions and generalisation	329
10.9 Assessment of feature selection	329
10.10 Conclusion	330
10.11 Exercises	331
11 Conclusions	333
11.1 About ML limitations	333
11.2 A bit of ethics	334
11.3 Take-home notions	335
11.4 Recommendations	335
A Unsupervised learning	337
A.1 Probability density estimation	337
A.1.1 Nonparametric density estimation	337
A.1.1.1 Kernel-based methods	338
A.1.1.2 k-Nearest Neighbors methods	339
A.1.2 Semi-parametric density estimation	339
A.1.2.1 Mixture models	339
A.1.2.2 The EM algorithm	340
A.1.2.3 The EM algorithm for the mixture model	340
A.2 K-means clustering	342
B Linear algebra notions	343
B.1 Rank of a matrix	343
B.2 Inner product	343
B.3 Gramian matrix	344
B.4 Diagonalisation	344
B.5 QR decomposition	344
B.6 Singular Value Decomposition	345
B.7 Chain rules of differential calculus	345
B.8 Quadratic norm	346
B.9 The matrix inversion formula	346
C Optimisation notions	347
C.1 Least-squares	347
C.2 Linear programming	348
C.3 Quadratic programming	348
D Probabilistic notions	349
D.1 Common univariate discrete probability functions	349
D.1.1 The Bernoulli trial	349
D.1.2 The Binomial probability function	349
D.2 Common univariate continuous distributions	350
D.2.1 Uniform distribution	350
D.2.2 The Beta distribution	350
D.2.3 The chi-squared distribution	350
D.2.4 Student's <i>t</i> -distribution	350
D.2.5 F-distribution	351
D.3 Exponential family of distributions	351
D.4 Common statistical hypothesis tests	352
D.4.1 χ^2 -test: single sample and two-sided	352
D.4.2 t-test: two samples, two sided	352

D.4.3 F-test: two samples, two sided	352
D.5 Transformation of random variables and vectors	353
D.6 Correlation and covariance matrices	353
D.7 Conditioning reduce entropy	353
D.8 Convergence of random variables	354
D.8.1 Example	355
D.9 The central limit theorem	355
D.10 The Chebyshev's inequality	355
D.11 Empirical distribution properties	356
D.12 Useful relations	356
D.13 Minimum of expectation vs. expectation of minimum	356
D.14 Taylor expansion of function	357
D.15 Proof of Eq. (5.5.28)	357
D.16 Optimism of the empirical risk	358
D.16.1 Bias of the quadratic empirical risk	358
E Plug-in estimators	361
F Kernel functions	363
F.1 Distance-based kernel functions	363
G Companion R package	365
H Companion R Shiny dashboards	367
H.1 List of Shiny dashboards	367

Chapter 1

Introduction

Over the last decades, a growing number of organisations have been allocating a 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, second, computer technology enables effective data storage and processing 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. In the artificial intelligence community, the most common name is *machine learning* [73]. More recently, the set of computational techniques and tools to support the modelling of a large amount of data is grouped under the more general label of *data science*.

The need for programs that can learn was stressed by Alan Turing, who argued that it might be too ambitious to write from scratch programs for tasks that even humans 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 relationship¹ between two observed 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 .

¹Note by relation we refer here to a probabilistic association between the two variables without any causal connotation.

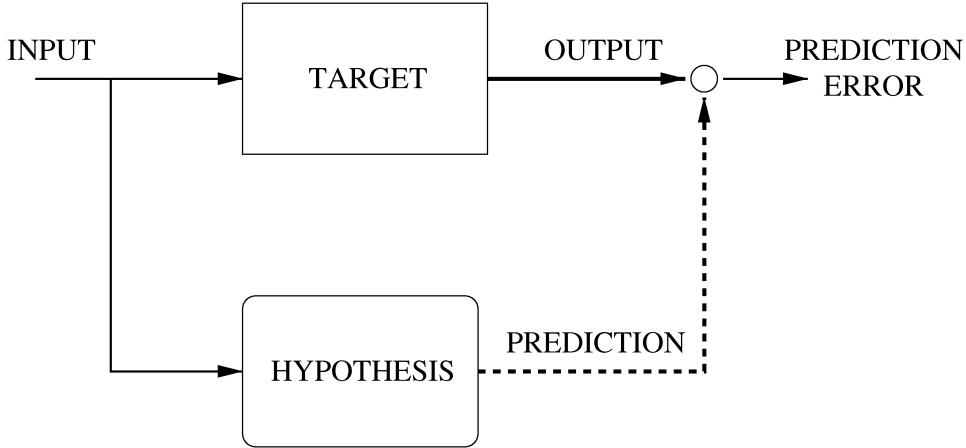


Figure 1.1: The supervised learning setting. Machine learning aims to infer from observed data the best model of the stochastic input/output dependency.

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 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 she can take advantage of existing knowledge.

Though machine learning is becoming a central component in many (so-called) intelligent applications, we deem that simply considering it as a powerful computational technology would be utterly reductive. The process of extracting knowledge from observations lies at the root of the modern scientific process, and the most challenging issues in machine learning relate to well-established philosophical and epistemological problems, notably induction or the notion of truth [30].

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 modelling process consists of a *preliminary* phase that brings the data from their original form to a structured configuration and a *learning* phase that 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, a number of descriptive variables and hypothesises the existence of a (stochastic) relation (or dependency) between the measurable variables. The definition of the input variables (and where necessary their transformations) is a very crucial step and is called *feature engineering*. It is important to stress here the proactive role played by the human (in contrast to a *tabula rasa* approach), and that this role is a necessary condition for any knowledge process.

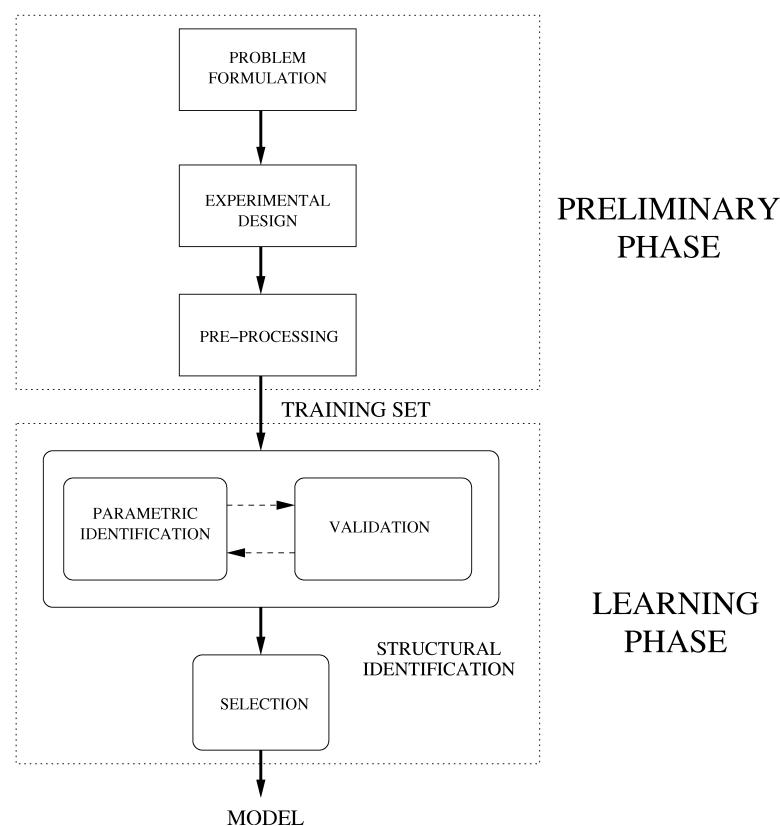


Figure 1.2: The modelling process and its decomposition in the preliminary phase and learning phase.

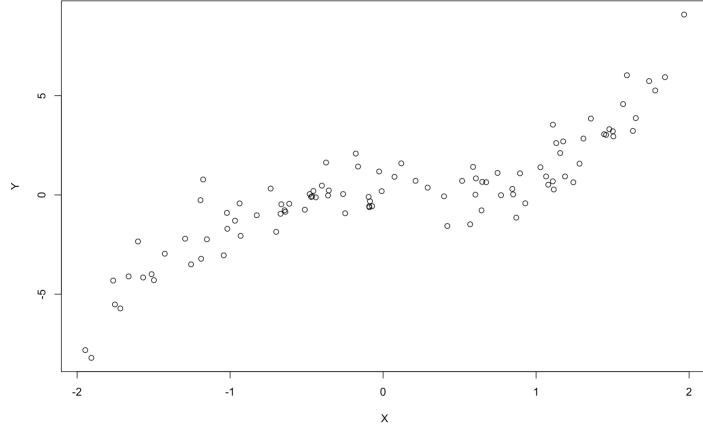


Figure 1.3: A training set for a simple supervised learning problem with one input variable x and one output variable y . The dots represent the observed samples.

Experimental design. This step aims to return a dataset which, ideally, should be made of observations that are well-representative of the phenomenon in order to maximise the performance of the modelling process [56].

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 [118], 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 will mostly focus 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. The knowledge of this dependency could shed light on the observed phenomenon and let us predict the value of the output y for a given input (e.g. what is the expected weight of a child who 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) that is not identical to the one in Figure 1.3 though both originate from the same measurable phenomenon. This simple fact suggests that a simple interpolation of the observed data would not produce an accurate model of the data.

The goal of machine learning is to formalise and optimise 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* that 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 capacity (or representation power). For our introductory purposes, it is

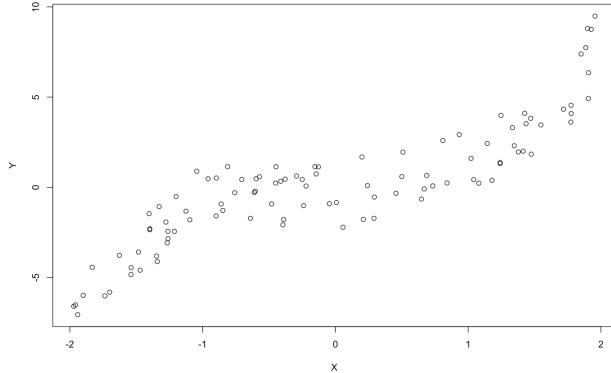


Figure 1.4: A second realisation of the training set for the same phenomenon observed in Figure 1.3. The dots represent the observed examples.

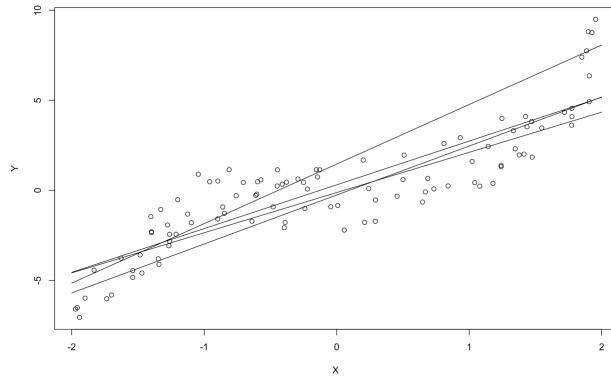


Figure 1.5: Training set and three parametric models which belong to the class of first order polynomials.

sufficient to consider here a *class* as a set of input/output models (e.g. the set of polynomial models) with the same *model structure* (e.g. second-order degree) and the *capacity* of the class as a measure of the set of input/output mappings which can be 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 belong to the class of second-order polynomials.

The reader could visually decide whether the class of second-order models is more suitable 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 that produces the best fitting.

In real high-dimensional settings, however, a visual assessment of the quality of a model is neither possible nor sufficient. Data-driven quantitative criteria are therefore required. We will assume that the goal of learning is to achieve a good *statistical generalisation*. This means that the learned model is expected to return an accurate prediction of the dependent (output) variable for new (unseen) values

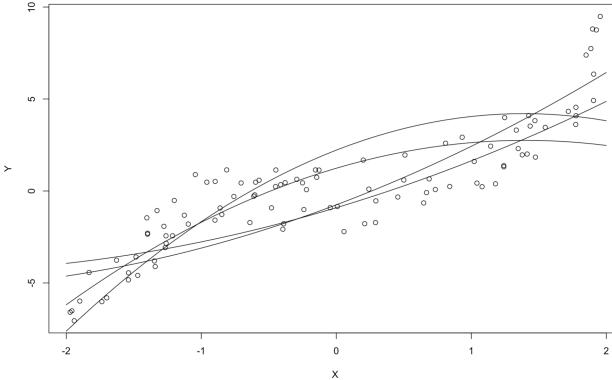


Figure 1.6: Training set and three parametric models which belong to the class of second-order polynomials.

of the independent (input) variables. By new values we intend values which are not part of the training set but are generated by the same stochastic process.

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 that seeks the model structure which is expected to have the best accuracy. 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 that 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 optimisation. Unfortunately, the reality is far more complex. In fact, because of the finite amount of data and their random nature, there is a strong tie 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 are 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 generalisation with an effective learning procedure.

However, the ambition of this handbook is to present machine learning as a sci-

entific domain that 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, optimisation and analysis of massive datasets.

It is important to remark, also, that the recent adoption of machine learning models is showing the limitation of pure black-box approaches, targeting accuracy at the cost of interpretability. This is made evident by the embedding of automatic approaches in decision-making processes with impact on ethical, social, political, or juridical aspects. While we are personally skeptical about gaining any interpretability from a large number of parameters and hyperparameters underlying a supervised learner, we are confident that human insight can be obtained by techniques able to reduce or modularise large variate tasks. In this direction, feature selection and causal inference techniques are promising approaches to master the complexity of data-driven modelling and return human accessible descriptions (e.g. in the form of mechanisms).

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 [147] and made available by the companion package `gbcod` (Appendix G). We provide as well a number of Shiny dashboards (Appendix H) to give the reader a more tangible idea of somewhat abstract concepts. For an introduction to R we refer the reader to [54, 172]. 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 summarises the relevant background material in probability. Chapter 3 introduces the classical 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 that 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 generalisation accuracy by combining several learning machines. Chapter 10 deals with the problem of dimensionality reduction and in particular with feature selection strategies. Although the handbook focuses on supervised learning, some related notions of unsupervised learning and density estimation are presented in Appendix A.

We invite the readers who are interested in additional topics (philosophical foundations, graphical modeling, Bayesian estimation, time series forecasting and causal inference) to refer to the extended version of the handbook [30].

1.1 Notations

Throughout this manuscript, boldface denotes random variables (e.g. \mathbf{x}) and normal font is used for instances (realisations) of random variables (e.g. x). Strictly speaking, one should always distinguish in notation between a random variable and its realisation. However, we will adopt this extra notational burden only when the meaning is not clear from the context. Then we will use $\text{Prob}\{z\}$ (or $(p(z))$) as a shorthand for $\text{Prob}\{\mathbf{z} = z\}$ ($(p_{\mathbf{z}}(z))$) when the identity of the random variable is 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.

Warning: a typical ambiguity in multivariate data notations comes from the use of a subscript to identify the observation of dataset, e.g. the use of x_i to identify the i th realisation of the random vector $\mathbf{x} \in \mathbb{R}^n$. Now x_i could be used as well to refer to the realisation of the scalar random variable $\mathbf{x}_i \in \mathbb{R}$. This book is no exception in this sense but we are confident that the context will be of help to the reader to distinguish between the two cases.

Gender-neutral pronouns: computer sciences suffer from the gender issue definitely more than other sciences. Of course, you won't find any solution in this book but the author (a man) felt odd in referring to a generic reader by using a masculine pronoun exclusively. He then decided to use as much as possible a "(s)he" notation or, alternatively, a (balanced) random gender choice.

Generic notation

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

Probability Theory notation

- Ω : Set of possible outcomes.
- ω : 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} .
- $L_N(\theta)$: Likelihood of a parameter θ given the dataset D_N .
- $l_N(\theta)$: Log-Likelihood of a parameter θ given the dataset D_N .
- $\mathcal{U}(a, b)$: univariate uniform probability density between a and $b \geq a$.
- $\mathcal{N}(\mu, \sigma^2)$: univariate Normal probability density with mean μ and variance σ^2 (Section 2.12.2).
- $\mathbf{z} \sim p_{\mathbf{z}}(z)$: random variable \mathbf{z} with probability density $p_{\mathbf{z}}(z)$.
- $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$: random variable \mathbf{z} with Normal density with mean μ and variance σ^2 .

Learning Theory notation

- \mathbf{x} : Multidimensional random input vector.
- \mathbf{x}_j : j th component of the multidimensional input variable.
- $\mathcal{X} \subset \mathbb{R}^n$: Input space.
- \mathbf{y} : output (or target) variable.
- $\mathcal{Y} \subset \mathbb{R}$: Output space.
- x_i : i th observation of the random vector \mathbf{x} (but also realisation of the variable \mathbf{x}_i . See warning above)
- x_{ij} : i th observation of the j th component 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 example (also observation or data point): i^{th} case in training set.
- N : Number of observed examples in the training set.

- $D_N = \{z_1, z_2, \dots, z_N\}$: Training set.
- Λ : Class of hypothesis.
- α : Hypothesis parameter vector.
- $h(x, \alpha)$: Hypothesis function.
- Λ_s : Hypothesis class of capacity (or complexity) s .
- $L(y, f(x, \alpha))$: Loss function.
- $R(\alpha)$: Functional risk.
- $\alpha_0 = \arg \min_{\alpha \in \Lambda} R(\alpha)$: lowest risk hypothesis in the class Λ .
- $R_{\text{emp}}(\alpha)$: Empirical functional risk.
- α_N : Parameter which minimises the empirical risk of D_N
- G_N : Mean integrated squared error (MISE).
- l : Number of folds in cross-validation.
- \hat{G}_{cv} : Cross-validation estimate of G_N .
- \hat{G}_{loo} : Leave-one-out estimate of G_N .
- N_{tr} : Number of examples used for training in cross-validation.
- N_{ts} : Number of examples used for test in cross-validation.
- $D_{(i)}$: Training set with the i th example set aside.
- $\alpha_{N(i)}$: Parameter which minimises the empirical risk of $D_{(i)}$.
- \hat{G}_{bs} : Bootstrap estimate of G_N .
- $D_{(b)}$: Bootstrap training set of size N generated by D_N with replacement.
- $\alpha_{(b)}$: Parameter which minimises the empirical risk of the bootstrap set $D_{(b)}$.
- B : Number of bootstrap examples.

Data analysis notation

- x_i or $X[i, :]$: i th row of matrix X .
- x_j or $X[:, j]$: j th column of matrix X .
- x_{ij} : j th element of vector x_i .
- X_{ij} or $X[i, j]$: ij 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^{th} example set aside.
- $e_j^{\text{loo}} = y_j - \hat{y}_j^{-j}$: Leave-one-out error with the j^{th} example set aside.
- $K(\cdot, \cdot)$: Kernel function.
- B : Bandwidth.
- β : Linear coefficients vector.
- $\hat{\beta}$: Least-squares parameters vector.
- $\hat{\beta}^{-j}$: Least-squares parameters vector with the j^{th} example set aside.
- $h_j(x, \alpha)$: j^{th} , $j = 1, \dots, m$, local model in a modular architecture.
- ρ_j : Activation or basis function.
- η_j : Set of parameters of the activation function.

Chapter 2

Foundations of probability

Uncertainty is inescapable in the real world. Even without resort to indeterminism, its pervasiveness is due to the complexity of reality and the limitations of human observational skills and modelling capabilities. According to [113] *uncertainty arises because of limitations in our ability to observe the world, limitations in our ability to model it, and possibly even because of innate nondeterminism*. Probability theory is one of many disciplines [135] concerned with the study of uncertain (or random) phenomena. It is also, according to the author, one of the most successful ones in terms of formalisation, theoretical and algorithmic developments and practical applications. For this reason, in this book, we will adopt probability as the mathematical language to describe and quantify uncertainty. Uncertain phenomena, although not predictable in a deterministic fashion, may present some regularities and consequently be described mathematically by idealised probabilistic models. These models consist of a list of all possible outcomes together with the respective probabilities. The theory of probability makes it possible to infer from these models the patterns of future behaviour.

This chapter presents the basic notions of probability which serve 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 a random variable as a compact representation of uncertain knowledge and the use of probability as an effective formal tool to manipulate and process such 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 characterised 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, passing an exam or measuring the time to reach home.

A random experiment is then characterised by a *sample space* Ω that is a (finite or infinite) set of all the possible outcomes (or configurations) ω of the experiment. The elements of the set Ω are called *experimental outcomes* or *realisations*. For example, in the dice experiment, $\Omega = \{\omega_1, \omega_2, \dots, \omega_6\}$ and ω_i stands for the outcome

corresponding to getting the face with the number i . If ω 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. Alternatively, we could be interested in representing the uncertainty of two consecutive tosses. In that case, the outcome would be the pair $(\omega_{(t)}, \omega_{(t+1)})$ where $\omega_{(t)}$ is the outcome at time t .

Uncertainty stems from variability. Each time we observe a random phenomenon, we may observe different outcomes. In probabilistic jargon, observing a random phenomenon is interpreted as the realisation 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 that generated the outcome ω_i : we say that an event \mathcal{E} occurred during the trial if the set \mathcal{E} contains the element ω_i . For example, in the die experiment, an event (denoted *odd number*) is the set of odd values $\mathcal{E} = \{\omega_1, \omega_3, \omega_5\}$. This means that when we observe the outcome ω_5 the event *odd 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:

- Ω 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 *disjoint* if

$$\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset \tag{2.1.1}$$

that is each time that \mathcal{E}_1 occurs, \mathcal{E}_2 does not occur as well.

- a partition of Ω is a set of disjoint sets \mathcal{E}_j , $j = 1, \dots, J$ (i.e. $\mathcal{E}_{j_1} \cap \mathcal{E}_{j_2} = \emptyset \forall j_1, j_2 \in J$) such that

$$\bigcup_{j=1}^J \mathcal{E}_j = \Omega$$

- given an event \mathcal{E} we define the *indicator function* of \mathcal{E} by

$$I_{\mathcal{E}}(\omega) = \begin{cases} 1 & \text{if } \omega \in \mathcal{E} \\ 0 & \text{if } \omega \notin \mathcal{E} \end{cases} \tag{2.1.2}$$

Let us now consider the notion of *class of events*. An arbitrary collection of subsets of Ω is not a class of events. We require that if \mathcal{E}_1 and \mathcal{E}_2 are events, the same also holds 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* [134]. 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, this measure associates to each possible outcome ω a number between 0 and 1. It follows that we can assign to each event \mathcal{E} a real number $\text{Prob}\{\mathcal{E}\} \in [0, 1]$ which denotes the *probability of the event \mathcal{E}* . The measure associated with the event including all possibilities is 1. The function $\text{Prob}\{\cdot\} : 2^\Omega \rightarrow [0, 1]$ 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. $\text{Prob}\{\mathcal{E}_1 \cup \mathcal{E}_2\} = \text{Prob}\{\mathcal{E}_1\} + \text{Prob}\{\mathcal{E}_2\} - \text{Prob}\{\mathcal{E}_1 \cap \mathcal{E}_2\}$.

These conditions are known as the *axioms of the theory of probability* [114]. The first axiom states that all the probabilities are nonnegative real numbers. The second axiom attributes a probability of unity to the universal event Ω , thus providing a normalisation of the probability measure. If \mathcal{E}_1 and \mathcal{E}_2 are mutually exclusive (Equation (2.1.1)) the third axiom boils down to $\text{Prob}\{\mathcal{E}_1 \cup \mathcal{E}_2\} = \text{Prob}\{\mathcal{E}_1\} + \text{Prob}\{\mathcal{E}_2\}$. It then states that the probability function must be additive for disjoint events, consistently with the intuitive idea of how probabilities behave.

So from a mathematician perspective, probability is easy to define: it is a countably additive set function defined on a *Borel field*, with a total mass of one. Every probabilistic property, for instance $\mathcal{E}_1 \subset \mathcal{E}_2 \Rightarrow \text{Prob}\{\mathcal{E}_1\} \leq \text{Prob}\{\mathcal{E}_2\}$ or $\text{Prob}\{\mathcal{E}^c\} = 1 - \text{Prob}\{\mathcal{E}\}$, can be derived directly or indirectly from the axioms (and only the axioms).

There are many interpretations and justifications of these axioms, and we discuss the frequentist and the Bayesian interpretation in Section 2.2.2 briefly. What is relevant here is that the probability function is a formalisation of uncertainty and that most of its properties and results appear to be coherent with the human perception of uncertainty [105].

2.1.2 Sample space and random variables

The *sample space* Ω is the (finite or infinite) set of all the possible outcomes (or configurations) ω of the experiment. In most probabilistic models, it is convenient to provide a structure to such space, for instance by defining it as the Cartesian product (or product set) of the domains of several variables. A variable is any property or descriptor (categorical or numerical) of a phenomenon that can take multiple values. Suppose for instance that we want to model, in a very compact way, the atmospheric weather: we could use three categorical random variables where

1. the first represents the sky condition and takes value in the finite set {CLEAR, CLOUDY}.
2. the second represents the barometer trend and takes value in the finite set {RISING, FALLING},
3. the third represents the humidity in the afternoon and takes value in {DRY, WET}.

	Sky	Barometer	Humidity
ω_1	CLEAR	RISING	DRY
ω_2	CLEAR	RISING	WET
ω_3	CLEAR	FALLING	DRY
ω_4	CLEAR	FALLING	WET
ω_5	CLOUDY	RISING	DRY
ω_6	CLOUDY	RISING	WET
ω_7	CLOUDY	FALLING	DRY
ω_8	CLOUDY	FALLING	WET

Table 2.1: Sample space as the product set of three binary categorical variables

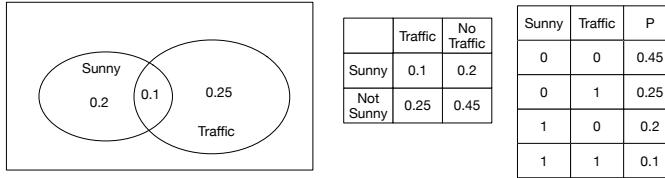


Figure 2.1: Visualisation of two events and probability measures: Venn diagram (left), two-way table (centre), probability distribution table (right)

The associated sample space is made of 8 outcomes in Table 2.1. In this representation the definition of an event as a subset of experimental outcomes coincides with any declarative statement made using variables. For instance, the statement "*the sky is clear*" corresponds to the event $\{\omega_1, \omega_2, \omega_3, \omega_4\}$ while the statement "*the weather is dry and the barometer is falling*" corresponds to the event $\{\omega_3, \omega_7\}$. In what follows we will use the product set representation of the sample space every time it is more interesting to reason in terms of relationships between variables than in terms of relationships between events.

2.1.3 Visualisation of probability measures

Since probabilistic events are sets of outcomes, Venn diagrams could be a convenient manner to illustrate the relations between events and the notion of probability measure. Suppose that you are a biker and you are interested in representing the variability of weather and traffic conditions in your town in the morning. In particular, you are interested in the probability that the morning will be sunny (or not) and the road busy (or not). In order to formalise your practical issue, you could define the uncertainty about the morning state by defining a sample space which is the set of all possible morning conditions. Two notions are of interest here: sunny mornings and traffic conditions. We could then define two categorical random variables denoting the presence/absence of sun and of traffic. What is the relationship and probability of the events in such sample space? Figure 2.1 illustrates the sample space, the two events, and the (hypothetical) probability measures by means of a Venn diagram and two different tabular representations. The three representations in Figure 2.1 convey the same information in different manners. Notwithstanding, they do not necessarily scale-up in the same manner if we take into consideration a larger number of variables. For instance, for n binary variables the Venn diagram should contain all 2^n hypothetically possible outcomes¹.

¹see Wikipedia https://en.wikipedia.org/wiki/Venn_diagram

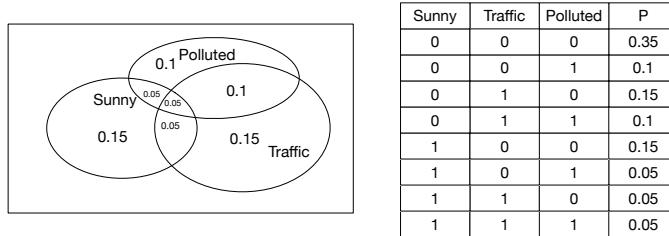


Figure 2.2: Visualisation of three events and related probability measures: Venn diagram (left), probability distribution table (right)

Suppose that you are also interested in another variable, i.e. the air quality. Adding such a variable to your probability representation would make your Venn representation more complicated and the two-way table inadequate (Figure 2.2). The visualisation will still be more difficult to handle and interpret if we deal with more than three variables.

Given their difficulty of encoding information in realistic probabilistic settings, Venn diagrams are a pedagogical yet very limited tool for representing uncertainty. This is the reason, why, apart from some exceptions, we will prefer to represent the probability distribution by using the tabular format (Figure 2.2 right or Table 2.1).

2.2 The definition of probability

Once introduced the notion of probability, 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 actual probability values, 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.

2.2.1 Symmetrical definition of probability

Consider a random experiment where the sample space is made of a finite number M of symmetric outcomes (i.e., they are equally likely to occur). Let the number of outcomes that are favourable to the event \mathcal{E} (i.e. the event \mathcal{E} takes place if one of them occurs) 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

$$\text{Prob}\{\mathcal{E}\} = \frac{M_{\mathcal{E}}}{M} \quad (2.2.3)$$

In other words, according to the principle of indifference (a term popularised by J.M. Keynes in 1921), we have that *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* [134]. The computation of this quantity requires combinatorial methods for counting the favourable outcomes. This is typically the approach adopted for 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 and finite space assumptions. But how to be sure that the symmetrical hypothesis holds? and that is invariant? Think, for instance, to the probability that a newborn is a boy. Is this a symmetric case? More generally, how would one define the probability of an event if the symmetrical hypothesis does not necessarily hold or the space is not finite?

2.2.2 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

$$\frac{N_{\mathcal{E}}}{N} \tag{2.2.4}$$

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

$$\text{Prob}\{\mathcal{E}\} = \lim_{N \rightarrow \infty} \frac{N_{\mathcal{E}}}{N} \tag{2.2.5}$$

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 finite², and the limit has to be accepted as a hypothesis, not as a number that can be determined experimentally [134].

Moreover, the assumption *under exactly the same conditions* is not as innocuous as it seems. How could you ensure that two experiments occur under exactly the same conditions? And what do those conditions refer to? Temperature, humidity, obsolescence of the equipment? Are humans really able to control exactly all of them? Would you be able to reproduce the exact same conditions of an experiment?

Notwithstanding, the frequentist interpretation is very important to show the links between theory and application as shown by the theoretical result in the following section.

2.2.2.1 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 2.2. Let $\text{Prob}\{\mathcal{E}\} = p$ 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 $\epsilon > 0$,

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

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 $N \rightarrow \infty$. This result suggests the use of the frequency as estimation of p and justifies the widespread use

²As Keynes said "In the long run we are all dead".

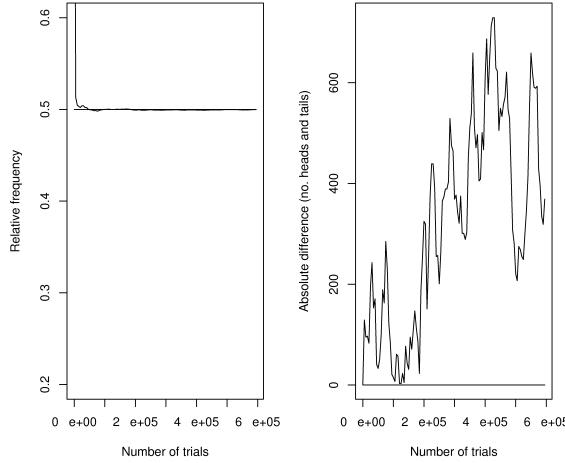


Figure 2.3: Fair coin-tossing random experiment: evolution of the relative frequency (left) and of the absolute difference (right) between the number of heads and tails (R script `Probability/freq.R` in `gbcodes`).

of the frequentist approach (e.g. in Monte Carlo simulation) to illustrate or numerically solve probability problems. The relation between frequency and probability is illustrated by the Shiny dashboard `lawlarge.R` (package `gbcodes`).

Note that such a result does not imply that the number $N_{\mathcal{E}}$ will be close to Np as one could naively infer from (2.2.4). In fact,

$$\text{Prob}\{N_{\mathcal{E}} = Np\} \approx \frac{1}{\sqrt{2\pi Np(1-p)}} \rightarrow 0, \quad \text{as } N \rightarrow \infty \quad (2.2.6)$$

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 [165] (Figure 2.3). On the contrary, it could happen that the absolute difference keeps growing (though at a slower rate than the number of tosses) as shown in the R script `Probability/freq.R` and the Shiny dashboard `lawlarge.R`.

2.2.3 Degree of belief and probability

The frequentist interpretation of probability relies on the availability of a number N of observations. As such, it appears inadequate to represent the notion of probability when it is used to model a subjective degree of belief. Think, for instance, to the probability that your professor wins a Nobel Prize: how to define in such case 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 [174]: think for instance of the subjective degree of conviction of jurors who must judge a defendant on the basis of uncertain evidence³.

³Note that the subjective interpretation of probability relies on the strong assumption that human belief and reasoning (i.e. belief manipulation) are coherent with the probabilistic calculus. Now several works challenge this assumption [108]. Nevertheless, the "Dutch book" argument shows that a reasoning procedure that does not respect the probability rules would necessarily lead to contradictions (and then irrationality). In other words, though probability is hardly the way humans think, it is definitely the way they should if they want to be coherent.

It follows that the Bayesian vision of probability is wider than the frequentist one since it allows the integration of subjective beliefs and observed data. This is made possible by the Bayes' theorem (Section 2.6) which provide a formal way to revise a priori belief in the light of new evidence (e.g. data). Though this is a powerful reasoning engine, its weakness derives from the need of encoding in probabilistic terms vague human information.

This manuscript will not cover the Bayesian approach to statistics and data analysis for the sake of compactness, though the author is well aware that Bayesian machine learning approaches are more and more common and successful. Readers interested in the foundations of the Bayesian interpretation of probability are referred to [105]. Readers interested in introductions to Bayesian machine learning are referred to [80, 13].

2.2.4 A pragmatic approach to probability

The history of probability has been characterised by a never-ending controversy between frequentist and Bayesian supporters [14] about the interpretation of probability⁴.

In this book we will take a pragmatic approach where probability is considered as a mathematical model describing uncertainty in the real world for the sake of performing reliable predictions from observations. In this perspective, more than a philosophical definition of probability, what is relevant here is (i) to consider observed data as the result of the sampling process of an underlying distribution, (ii) how to estimate such distribution from real observations, (iii) how the accuracy of such estimation is related to observable outcomes (e.g. predictions) and (iv) how useful is such estimation/prediction for our purposes.

In this sense, it is interesting to mention the analogy in [134] between the notion of probability and physical notions, like the one of resistance in electrical circuits. Resistance (like probability) is just a useful abstraction of the behaviour of the real resistor object. Arguing about its meaning is less relevant than stressing the added value it provides when we reason about electricity and circuits configurations.

2.3 Independence and conditional probability

Let us consider two different events. We have already introduced the notions of complementary and disjoint events. Another important definition is the definition of independent events and the related notion of conditional probability. This notion is essential in machine learning since supervised learning aims to detect and model (in)dependencies by estimating conditional probabilities.

Definition 3.1 (Independent events). Two events \mathcal{E}_1 and \mathcal{E}_2 are *independent* if and only if

$$\text{Prob}\{\mathcal{E}_1 \cap \mathcal{E}_2\} = \text{Prob}\{\mathcal{E}_1\} \text{Prob}\{\mathcal{E}_2\} \quad (2.3.7)$$

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

The probability $\text{Prob}\{\mathcal{E}_1 \cap \mathcal{E}_2\}$ of seeing two events occurring together is also known as *joint probability* and often noted as $\text{Prob}\{\mathcal{E}_1, \mathcal{E}_2\}$. If two events are independent, the joint probability depends only on the two individual probabilities. As an example of two independent events, think of two outcomes of a roulette wheel or of two coins tossed simultaneously.

⁴We recommend this enlightening lecture on this topic http://videolectures.net/mlss09uk_jordan_bfway/.

From an uncertain reasoning perspective, independence is a very simplistic assumption since the occurrence (or the observation) of one event has no influence on the occurrence of the other, or similarly that the second event has no memory of the first. In other words, independence considers the uncertainty of a complex joint event as a function of the uncertainties of its components⁵. This makes the reasoning much simpler but, at the same time, too rough.

Exercise

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

•

Let \mathcal{E}_1 be an event such that $\text{Prob}\{\mathcal{E}_1\} > 0$ and \mathcal{E}_2 a second event. We define the conditional probability of \mathcal{E}_2 , given that \mathcal{E}_1 has occurred, the revised probability of \mathcal{E}_2 after we learn about \mathcal{E}_1 occurrence:

Definition 3.2 (Conditional probability). If $\text{Prob}\{\mathcal{E}_1\} > 0$ then the conditional probability of \mathcal{E}_2 given \mathcal{E}_1 is

$$\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} = \frac{\text{Prob}\{\mathcal{E}_1 \cap \mathcal{E}_2\}}{\text{Prob}\{\mathcal{E}_1\}} \quad (2.3.8)$$

The following result derives from the definition of conditional probability.

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

$$\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} = \text{Prob}\{\mathcal{E}_1\} \quad (2.3.9)$$

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.

Example

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

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

or equivalently $\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} = 0$. We can interpret this result by noting that if two events are disjoint, the realisation of one of them is highly informative about the realisation of the other. For instance, though it is very probable that Italy will win the next football World Cup ($\text{Prob}\{\mathcal{E}_1\} \gg 0$), this probability goes to zero if the (rare yet possible) event \mathcal{E}_2 ("World cup won by Belgium") occurs ($\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} = 0$). The two events are then dependent.

•

⁵We refer the interested reader to the distinction between extensional and intensional reasoning in [138]. Extensional reasoning (e.g. logics) always makes an assumption of independence, while intensional reasoning (e.g. probability) consider independence as an exception.

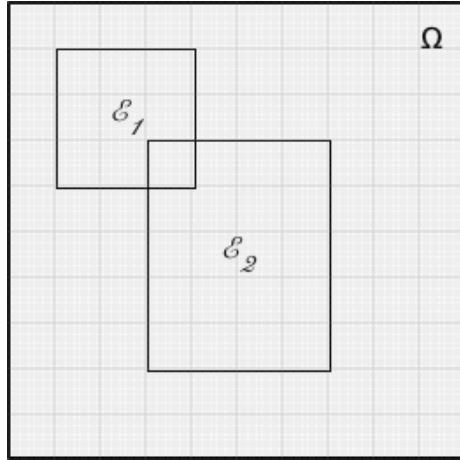


Figure 2.4: Events in a sample space.

Exercise

Let \mathcal{E}_1 and \mathcal{E}_2 be two independent events, and \mathcal{E}_1^c the complement of \mathcal{E}_1 . Are \mathcal{E}_1^c and \mathcal{E}_2 independent?

•

Example

Consider the sample space Ω and the two events \mathcal{E}_1 and \mathcal{E}_2 in Figure 2.4. Suppose that the probability of the two events is proportional to the surface of the regions. From the Figure 2.4 we derive

$$\text{Prob}\{\mathcal{E}_1\} = \frac{9}{100} = 0.09 \quad (2.3.10)$$

$$\text{Prob}\{\mathcal{E}_2\} = \frac{20}{100} = 0.2 \quad (2.3.11)$$

$$\text{Prob}\{\mathcal{E}_1 \cap \mathcal{E}_2\} = \frac{1}{100} = 0.01 \neq \text{Prob}\{\mathcal{E}_1\} \text{Prob}\{\mathcal{E}_2\} \quad (2.3.12)$$

$$\text{Prob}\{\mathcal{E}_1 \cup \mathcal{E}_2\} = 0.28 = \text{Prob}\{\mathcal{E}_1\} + \text{Prob}\{\mathcal{E}_2\} - \text{Prob}\{\mathcal{E}_1 \cap \mathcal{E}_2\} \quad (2.3.13)$$

$$\text{Prob}\{\mathcal{E}_1 | \mathcal{E}_2\} = \frac{1}{20} = 0.05 \neq \text{Prob}\{\mathcal{E}_1\} \quad (2.3.14)$$

$$\text{Prob}\{\mathcal{E}_2 | \mathcal{E}_1\} = \frac{1}{9} \neq \text{Prob}\{\mathcal{E}_2\} \quad (2.3.15)$$

and then derive the following conclusions: the events \mathcal{E}_1 and \mathcal{E}_2 are neither disjoint nor independent. Also, it is more probable that \mathcal{E}_2 occurs given that \mathcal{E}_1 occurred rather than the opposite.

•

From (2.3.8) we derive

$$\text{Prob}\{\mathcal{E}_1, \mathcal{E}_2\} = \text{Prob}\{\mathcal{E}_1\} \text{Prob}\{\mathcal{E}_2 | \mathcal{E}_1\} \quad (2.3.16)$$

If we replace the event \mathcal{E}_2 with the intersection of two events \mathcal{E}_2 and \mathcal{E}_3 , from (2.3.16) we obtain

$$\begin{aligned} \text{Prob}\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3\} &= \text{Prob}\{\mathcal{E}_1\} \text{Prob}\{\mathcal{E}_2, \mathcal{E}_3 | \mathcal{E}_1\} = \\ &= \text{Prob}\{\mathcal{E}_1\} \text{Prob}\{\mathcal{E}_2 | \mathcal{E}_3, \mathcal{E}_1\} \text{Prob}\{\mathcal{E}_3 | \mathcal{E}_1\} = \text{Prob}\{\mathcal{E}_1, \mathcal{E}_3\} \text{Prob}\{\mathcal{E}_2 | \mathcal{E}_3, \mathcal{E}_1\} \end{aligned}$$

z_1	z_2	$P(\mathbf{z}_1 = z_1, \mathbf{z}_2 = z_2)$
Neg	Neg	0.1
Neg	Pos	0.2
Zero	Neg	0.2
Zero	Pos	0.1
Pos	Neg	0.2
Pos	Pos	0.2

Table 2.2: The events \mathcal{E}_1 (\mathbf{z}_1 positive) and \mathcal{E}_2 (\mathbf{z}_2 negative) are independent though the two variables are dependent.

If we divide both terms by $\text{Prob}\{\mathcal{E}_3\}$ we obtain

$$\text{Prob}\{\mathcal{E}_1, \mathcal{E}_2 | \mathcal{E}_3\} = \text{Prob}\{\mathcal{E}_1 | \mathcal{E}_3\} \text{Prob}\{\mathcal{E}_2 | \mathcal{E}_1, \mathcal{E}_3\} \quad (2.3.17)$$

which is the conditioned version of (2.3.16).

2.3.1 Independent variables

If we define the sample space in terms of variables (Section 2.1.2), it is interesting to extend the notion of independence to variables. Let \mathbf{x} and \mathbf{y} be two random variables taking categorical values in \mathcal{X} and \mathcal{Y} . The two variables \mathbf{x} and \mathbf{y} are *probabilistically independent* if for all values of \mathbf{x} and \mathbf{y}

$$\text{Prob}\{\mathbf{x} = x, \mathbf{y} = y\} = \text{Prob}\{\mathbf{x} = x\} \text{Prob}\{\mathbf{y} = y\}, \quad \forall x \in \mathcal{X}, \quad y \in \mathcal{Y}$$

This means that we do not expect that the observation of a certain value of one variable will affect the probability of observing a certain value of the other. The independence between two variables \mathbf{x} and \mathbf{y} is a symmetric relation and is denoted by $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$.

Note that in the sample space defined by two dependent variables we could nevertheless find two independent events. Consider for instance the sample space in Table 2.2, where the two events \mathcal{E}_1 (\mathbf{z}_1 positive) and \mathcal{E}_2 (\mathbf{z}_2 negative) are independent.

2.4 The chain rule

The equation (2.3.16) shows that a joint probability can be *factorised* as the product of a conditional and an unconditional probability. In more general terms, the following rule holds.

Definition 4.1 (Chain rule). For any sequence of events $\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n$,

$$\begin{aligned} \text{Prob}\{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n\} &= \\ \text{Prob}\{\mathcal{E}_1\} \text{Prob}\{\mathcal{E}_2 | \mathcal{E}_1\} \text{Prob}\{\mathcal{E}_3 | \mathcal{E}_1, \mathcal{E}_2\} \dots \text{Prob}\{\mathcal{E}_n | \mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_{n-1}\} \end{aligned} \quad (2.4.18)$$

It is also possible to write a conditioning version of (2.4.18) by considering an event \mathcal{E}' :

$$\begin{aligned} \text{Prob}\{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n | \mathcal{E}'\} &= \\ \text{Prob}\{\mathcal{E}_1 | \mathcal{E}'\} \text{Prob}\{\mathcal{E}_2 | \mathcal{E}_1, \mathcal{E}'\} \text{Prob}\{\mathcal{E}_3 | \mathcal{E}_1, \mathcal{E}_2, \mathcal{E}'\} \dots \text{Prob}\{\mathcal{E}_n | \mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_{n-1}, \mathcal{E}'\} \end{aligned} \quad (2.4.19)$$

It follows that, given three events $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ it is possible to write down the *multiplication law of probability*

$$\text{Prob}\{\mathcal{E}_1, \mathcal{E}_2 | \mathcal{E}_3\} = \text{Prob}\{\mathcal{E}_1 | \mathcal{E}_3\} \text{Prob}\{\mathcal{E}_2 | \mathcal{E}_1, \mathcal{E}_3\} = \text{Prob}\{\mathcal{E}_2 | \mathcal{E}_3\} \text{Prob}\{\mathcal{E}_1 | \mathcal{E}_2, \mathcal{E}_3\}$$

which satisfies commutativity.

2.5 The law of total probability

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

Theorem 5.1 (Law of total probability). *Let $\text{Prob}\{\mathcal{E}_i\}$, $i = 1, \dots, k$ denote the probability of the i th event \mathcal{E}_i and $\text{Prob}\{\mathcal{E}|\mathcal{E}_i\}$, $i = 1, \dots, k$ the conditional probability of a generic event \mathcal{E} given that \mathcal{E}_i has occurred. It can be shown that*

$$\text{Prob}\{\mathcal{E}\} = \sum_{i=1}^k \text{Prob}\{\mathcal{E}|\mathcal{E}_i\} \text{Prob}\{\mathcal{E}_i\} = \sum_{i=1}^k \text{Prob}\{\mathcal{E} \cap \mathcal{E}_i\} \quad (2.5.20)$$

The quantity $\text{Prob}\{\mathcal{E}\}$ is referred to as *marginal probability* and denotes the probability of the event \mathcal{E} irrespective of the occurrence of other events. A commonsense interpretation of this theorem is that if an event \mathcal{E} (e.g. an effect) depends on the realisation of k disjoint events (e.g. causes), the probability of observing \mathcal{E} , is a weighted average of each single conditional probability $\text{Prob}\{\mathcal{E}|\mathcal{E}_i\}$ where the weights are given by the marginal probabilities of each event $\mathcal{E}_i, i = 1, \dots, k$.

In its simplest version, given a generic event \mathcal{E}_1 , we can compute the probability of another event \mathcal{E} as

$$\text{Prob}\{\mathcal{E}\} = \text{Prob}\{\mathcal{E}|\mathcal{E}_1\} \text{Prob}\{\mathcal{E}_1\} + \text{Prob}\{\mathcal{E}|\mathcal{E}_1^c\} \text{Prob}\{\mathcal{E}_1^c\} \quad (2.5.21)$$

where \mathcal{E}_1^c is the complement of \mathcal{E}_1 ⁶. For instance, we can compute the probability that the highway is busy (event \mathcal{E}) once we know the probability that an accident occurred (event \mathcal{E}_1) or not (complement \mathcal{E}_1^c) and the conditional probabilities of traffic given the occurrence (or not) of an accident.

It may be useful also to write an *extra-conditioning* [27] version of the total probability. Given an event \mathcal{E}' and the set $\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_k$ of mutually exclusive events:

$$\text{Prob}\{\mathcal{E}|\mathcal{E}'\} = \sum_{i=1}^k \text{Prob}\{\mathcal{E}|\mathcal{E}_i, \mathcal{E}'\} \text{Prob}\{\mathcal{E}_i|\mathcal{E}'\} \quad (2.5.22)$$

2.6 The Bayes' theorem

The Bayes' theorem is the mathematical version of the formula first proved by the Reverend Thomas Bayes in 1763 and represents the cornerstone of the probabilistic calculus since it formalises the relationship between a direct conditional probability $\text{Prob}\{\mathcal{E}|\mathcal{E}_i\}$ and its inverse $\text{Prob}\{\mathcal{E}_i|\mathcal{E}\}$.

⁶Note that from (2.5.21) it follows that $\text{Prob}\{\mathcal{E}\} = \text{Prob}\{\mathcal{E}|\mathcal{E}_1\} = \text{Prob}\{\mathcal{E}|\mathcal{E}_1^c\}$ (i.e. \mathcal{E} is independent of \mathcal{E}_1) if and only if $\text{Prob}\{\mathcal{E}|\mathcal{E}_1\} = \text{Prob}\{\mathcal{E}|\mathcal{E}_1^c\}$.

Theorem 6.1 (Bayes' theorem). *The conditional “inverse” probability of any \mathcal{E}_i , $i = 1, \dots, k$ given that \mathcal{E} has occurred is given by*

$$\text{Prob}\{\mathcal{E}_i|\mathcal{E}\} = \frac{\text{Prob}\{\mathcal{E}|\mathcal{E}_i\} \text{Prob}\{\mathcal{E}_i\}}{\sum_{j=1}^k \text{Prob}\{\mathcal{E}|\mathcal{E}_j\} \text{Prob}\{\mathcal{E}_j\}} = \frac{\text{Prob}\{\mathcal{E}, \mathcal{E}_i\}}{\text{Prob}\{\mathcal{E}\}} \quad i = 1, \dots, k \quad (2.6.23)$$

It follows that the Bayes' theorem is the only sound way to derive from a conditional probability $\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\}$ its inverse

$$\begin{aligned} \text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} &= \frac{\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} \text{Prob}\{\mathcal{E}_1\}}{\text{Prob}\{\mathcal{E}_2\}} = \\ &= \frac{\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} \text{Prob}\{\mathcal{E}_1\}}{\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} \text{Prob}\{\mathcal{E}_1\} + \text{Prob}\{\mathcal{E}_2|\mathcal{E}_1^c\} \text{Prob}\{\mathcal{E}_1^c\}} \end{aligned} \quad (2.6.24)$$

Any alternative derivation (or shortcut) will lead inevitably to fallacious reasoning and inconsistent results (see the Prosecutor fallacy discussion in Section 2.7). Note that the Bayes' theorem can also be interpreted as a manner of updating the marginal probability $\text{Prob}\{\mathcal{E}_1\}$ to the conditional version $\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\}$ that is the probability of \mathcal{E}_1 once the event \mathcal{E}_2 occurred. In subjectivist terms the Bayes' theorem is a rule for updating the degree of belief of \mathcal{E}_1 once new evidence (i.e. the event \mathcal{E}_2) is obtained.

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

•

From (2.6.24) and by conditioning on a third event \mathcal{E}_3 , we obtain an *extra-conditioning* [27] version of the Bayes' theorem

$$\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2, \mathcal{E}_3\} = \frac{\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1, \mathcal{E}_3\} \text{Prob}\{\mathcal{E}_1|\mathcal{E}_3\}}{\text{Prob}\{\mathcal{E}_2|\mathcal{E}_3\}} = \frac{\text{Prob}\{\mathcal{E}_3|\mathcal{E}_1, \mathcal{E}_2\} \text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\}}{\text{Prob}\{\mathcal{E}_3|\mathcal{E}_2\}} \quad (2.6.25)$$

as long as $\text{Prob}\{\mathcal{E}_2|\mathcal{E}_3\} > 0$ and $\text{Prob}\{\mathcal{E}_3|\mathcal{E}_2\} > 0$.

•

Exercise

Verify the validity of the law of total probability and of the Bayes' theorem for the problem in Figure 2.5.

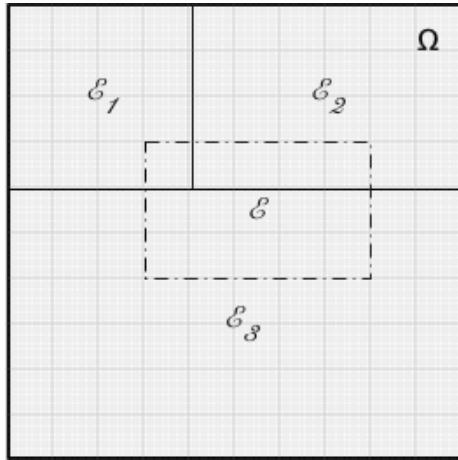


Figure 2.5: Events in a sample space

2.7 Direct and inverse conditional probability

The notion of conditional probability is central in probability and machine learning, but it is often prone to dangerous misunderstanding, for instance, when inappropriately used in domains like medical sciences or law. The most common error consists of taking a conditional probability $\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\}$ for its inverse $\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\}$. This is also known as the *prosecutor fallacy*, as discussed in an example later.

The first important element to keep in mind is that for any fixed \mathcal{E}_1 , the quantity $\text{Prob}\{\cdot|\mathcal{E}_1\}$ still satisfies the axioms of probability, i.e. the function $\text{Prob}\{\cdot|\mathcal{E}_1\}$ is itself a probability measure. *Conditional probabilities are probabilities* [27]. However, this does not generally hold for $\text{Prob}\{\mathcal{E}_1|\cdot\}$, which corresponds to fix the term \mathcal{E}_1 on the left of the conditional bar. For instance, if \mathcal{E}_2 , \mathcal{E}_3 and \mathcal{E}_4 are disjoint events we have

$$\text{Prob}\{\mathcal{E}_2 \cup \mathcal{E}_3 \cup \mathcal{E}_4|\mathcal{E}_1\} = \text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} + \text{Prob}\{\mathcal{E}_3|\mathcal{E}_1\} + \text{Prob}\{\mathcal{E}_4|\mathcal{E}_1\} \quad (2.7.26)$$

in agreement with the third axiom (Section 2.1.1) but

$$\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2 \cup \mathcal{E}_3 \cup \mathcal{E}_4\} \neq \text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} + \text{Prob}\{\mathcal{E}_1|\mathcal{E}_3\} + \text{Prob}\{\mathcal{E}_1|\mathcal{E}_4\} \quad (2.7.27)$$

Also it is generally not the case that $\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} = \text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\}$. As a consequence if \mathcal{E}_1 and \mathcal{E}_2 are not independent then

$$\text{Prob}\{\mathcal{E}_1^c|\mathcal{E}_2\} = 1 - \text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\}$$

but

$$\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2^c\} \neq 1 - \text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} \quad (2.7.28)$$

where \mathcal{E}^c denotes the complement of \mathcal{E} .

Another remarkable property of conditional probability, which is also a distinctive aspect of probabilistic reasoning, is its non-monotonic property. Given a non-conditional probability $\text{Prob}\{\mathcal{E}_1\} > 0$ a priori, we cannot say anything about the conditional term $\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\}$. This term can be larger, equal or smaller than $\text{Prob}\{\mathcal{E}_1\}$. For instance, if observing the event \mathcal{E}_2 makes the event more (less) probable then $\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} > \text{Prob}\{\mathcal{E}_1\}$ ($\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} < \text{Prob}\{\mathcal{E}_1\}$). If the two events are independent, then the probability of \mathcal{E}_1 does not change by conditioning. It follows that the degree of belief of an event (or statement) depends on the context.

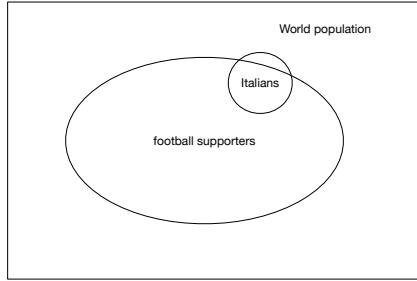


Figure 2.6: Italians and football fans.

Note that this does not apply to conventional logical reasoning where the validity of a statement is context-independent.

In more general terms, it is possible to say that any probability statement is conditional since it has been formulated on the basis of an often implicit background knowledge \mathcal{K} . For instance, if we say that the probability of the event \mathcal{E} = "rain tomorrow" is $\text{Prob}\{\mathcal{E}\} = 0.9$, we are implicitly taking into consideration the season, our location and probably the weather today. So we should better note it as $\text{Prob}\{\mathcal{E}|\mathcal{K}\} = 0.9$. As succinctly stated in [27] *all probabilities are conditional, and conditional probabilities are probabilities*.

Exercise

Consider as sample space Ω the set of all human beings. Let us define two events: the set \mathcal{E}_1 of Italians and the set \mathcal{E}_2 of football supporters. Suppose that the probability of the two events is proportional to the surface of the regions in Figure 2.6. Are these events disjoint? Are they independent? What about $\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\}$ and $\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\}$? Are they equal? If not, which one is the largest?

•

The prosecutor fallacy

Consider the following story: *A crime occurs in a big city (1M of inhabitants), and a deteriorated DNA trace of the murderer is collected. The DNA profile matches the profile of a person in a police database. A geneticist is contacted, and she states that the probability of finding a person with the same DNA profile is one out of 100 thousand (i.e. $1e - 5$). The prosecution lawyer asks for condemnation with the following argument: "since the chance of finding an innocent man with such characteristics is so tiny, then the probability that he is innocent will be tiny as well". The jury is impressed and ready to proceed with a life sentence. Then the defendant replies: "Do you know that the population of the city is 1M? So the average number of persons matching such DNA profile is 10. His chance of being innocent is not so tiny since it is $9/10$ and not one in 100000" Lacking any additional evidence, the suspect is acquitted.*

This short story is inspired by a number of real cases in court that were confronted with the serious error of confounding direct and inverse conditional probability [155]. The impact of such false reasoning is so relevant in law that it is known as

the *Prosecutor's fallacy*, a common default in reasoning when the collected evidence is tiny if the accused was innocent.

Let us analyse in probabilistic terms the fallacious reasoning that occurred in the example above. Let consider a criminal case for which we have 10 suspects, i.e. the responsible and 9 innocent persons (out of a 1 million population) matching the DNA profile. The probability of matching evidence (M) given that someone is innocent (I) is very low

$$\text{Prob}\{M|I\} = \frac{9}{999999} \approx 1e-5$$

However, what is relevant here is not the probability of the evidence given that he is innocent ($\text{Prob}\{M|I\}$) but the probability that is innocent given the evidence

$$\text{Prob}\{I|M\} = \frac{\text{Prob}\{M|I\} \text{Prob}\{I\}}{\text{Prob}\{M\}} = \frac{9/999999 \times 999999/1000000}{10/1000000} = 9/10.$$

We can rephrase the issue in the following frequentist terms. Given N inhabitants, m persons with DNA matching profiles and a single murderer, the following table shows the distribution of persons

	Match	No match
Innocent	$m-1$	$N-m$
Guilty	1	0

From the table above, it is easy to derive the inconsistency of the prosecutor fallacy reasoning since

$$\begin{aligned} \text{Prob}\{M|I\} &= \frac{m-1}{N-1} \approx \text{Prob}\{M\} = \frac{m}{N} \\ \text{Prob}\{I|M\} &= \frac{m-1}{m} >> \text{Prob}\{M|I\} \end{aligned}$$

•

Example: dependent/independent scenarios

Let us study the dependence between an ULB student commute time (denoted by the categorical variable \mathbf{z}_1) and the weather in Brussels (denoted by the categorical variable \mathbf{z}_2). Suppose that \mathbf{z}_1 takes value in $\{\text{LOW}, \text{MEDIUM}, \text{HIGH}\}$. and \mathbf{z}_2 in $\{G = \text{GOOD}, B = \text{BAD}\}$. Suppose that the array of joint probabilities is

	$\mathbf{z}_2 = G$ (in Bxl)	$\mathbf{z}_2 = B$ (in Bxl)	Marginal
$\mathbf{z}_1 = \text{LOW}$	0.15	0.05	$\text{Prob}\{\text{LOW}\} = 0.2$
$\mathbf{z}_1 = \text{MEDIUM}$	0.1	0.4	$\text{Prob}\{\text{MEDIUM}\} = 0.5$
$\mathbf{z}_1 = \text{HIGH}$	0.05	0.25	$\text{Prob}\{\text{HIGH}\} = 0.3$
$\text{Prob}\{G\} = 0.3 \quad \text{Prob}\{B\} = 0.7$			Sum=1

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

	LOW	MEDIUM	HIGH
$\text{Prob}\{\cdot G\}$	$0.15/0.3=0.5$	$0.1/0.3=0.33$	$0.05/0.3=0.16$

Else if the weather is bad

	LOW	MEDIUM	HIGH
$\text{Prob}\{\cdot B\}$	$0.05/0.7=0.07$	$0.4/0.7=0.57$	$0.25/0.7=0.35$

Since $\text{Prob}\{\cdot|G\} \neq \text{Prob}\{\cdot|B\}$, i.e. the probability of having a certain commute time changes according to the value of the weather, the relation (2.3.9) is not satisfied.

Now consider the dependency between the commute time in Brussels and the weather in Rome (denoted by the categorical variable \mathbf{z}_3).

	$\mathbf{z}_3 = G$ (in Rome)	$\mathbf{z}_3 = B$ (in Rome)	Marginal
$\mathbf{z}_1 = \text{LOW}$	0.18	0.02	$\text{Prob}\{\text{LOW}\} = 0.2$
$\mathbf{z}_1 = \text{MEDIUM}$	0.45	0.05	$\text{Prob}\{\text{MEDIUM}\} = 0.5$
$\mathbf{z}_1 = \text{HIGH}$	0.27	0.03	$\text{Prob}\{\text{HIGH}\} = 0.3$
	$\text{Prob}\{G\} = 0.9$	$\text{Prob}\{B\} = 0.1$	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
$\text{Prob}\{\cdot G\}$	$0.18/0.9=0.2$	$0.45/0.9=0.5$	$0.27/0.9=0.3$

while if the weather in Rome is bad

	LOW	MEDIUM	HIGH
$\text{Prob}\{\cdot B\}$	$0.02/0.1=0.2$	$0.05/0.1=0.5$	$0.03/0.1=0.3$

Note that the probability of the commute time does NOT change according to the value of the weather in Rome, e.g. $\text{Prob}\{\text{LOW}|B\} = \text{Prob}\{\text{LOW}\}$. Now try to answer the following question. If you would like to predict the commute time in Brussels, which variable would return more information on it: the weather in Rome or in Brussels?

•

Example: a medical study [174]

Let us consider a medical study about the relationship between the outcome of a medical test and the presence of a disease. We model the sample space as the product set of two categorical random variables:

1. the state of the patient \mathbf{z}_1 taking value in $\{H, S\}$ where H and S stand for a healthy and a sick patient, respectively.
2. the outcome of the medical test \mathbf{z}_2 taking value in $\{+, -\}$, where $+$ and $-$ stand for a positive and a 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 $\text{Prob}\{\mathbf{z}_1 = z_1, \mathbf{z}_2 = z_2\}$ can be approximated according to the expression (2.2.4) by

	$\mathbf{z}_1 = S$	$\mathbf{z}_1 = H$
$\mathbf{z}_2 = +$	$\frac{9}{1000} = .009$	$\frac{108-9}{1000} = .099$
$\mathbf{z}_2 = -$	$\frac{1}{1000} = .001$	$\frac{892-1}{1000} = .891$

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

$$\text{Prob}\{\mathbf{z}_2 = +|\mathbf{z}_1 = S\} = \frac{\text{Prob}\{\mathbf{z}_2 = +, \mathbf{z}_1 = S\}}{\text{Prob}\{\mathbf{z}_1 = S\}} = \frac{.009}{.009 + .001} = .9$$

$$\text{Prob}\{\mathbf{z}_2 = -|\mathbf{z}_1 = H\} = \frac{\text{Prob}\{\mathbf{z}_2 = -, \mathbf{z}_1 = H\}}{\text{Prob}\{\mathbf{z}_1 = H\}} = \frac{.891}{.891 + .099} = .9$$

\mathbf{z}_1	\mathbf{z}_2	\mathbf{z}_3	$P(\mathbf{z}_1 = z_1, \mathbf{z}_2 = z_2, \mathbf{z}_3 = z_3)$
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

Table 2.3: Joint probability distribution of the three-variable probabilistic model of the weather

According to these figures, the test appears to be accurate. Does this mean that we should be scared if we test positive? Though the test is accurate, the answer is negative, as shown by the quantity

$$\text{Prob}\{\mathbf{z}_1 = S | \mathbf{z}_2 = +\} = \frac{\text{Prob}\{\mathbf{z}_2 = +, \mathbf{z}_1 = S\}}{\text{Prob}\{\mathbf{z}_2 = +\}} = \frac{.009}{.009 + .099} \approx .08$$

This example confirms that sometimes humans tend to confound $\text{Prob}\{z_1|z_2\}$ with $\text{Prob}\{z_2|z_1\}$ and that the most intuitive response is not always the right one (see the example in Section 2.7).

•

Example: probabilistic weather

Consider a probabilistic model of the day's weather based on the sample space in Table 2.1. Let the joint probability values be given by Table 2.3. From the joint values, we can calculate the probabilities $P(CLEAR, RISING) = 0.47$ and $P(CLOUDY) = 0.35$ and the conditional probability value

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

Take the time now to compute yourself other probabilities: for instance what is the probability of having a cloudy sky in wet conditions? Does a rising barometer increase or not this probability? Is the event "clear sky and falling barometer" independent from the event "dry weather"?

•

2.8 Probability as Odds

Though probability is a conventional way to represent uncertainty or risk, humans (e.g. gamblers) often prefer to reason about risks in terms of odds rather than probability. Given the event \mathcal{E} , its *odds ratio* can be written as a function of the probability $\text{Prob}\{\mathcal{E}\}$ in the following manner

$$\text{OR}\{\mathcal{E}\} = \frac{\text{Prob}\{\mathcal{E}\}}{\text{Prob}\{\mathcal{E}^c\}} = \frac{\text{Prob}\{\mathcal{E}\}}{1 - \text{Prob}\{\mathcal{E}\}} \quad (2.8.29)$$

Note that if $\text{OR}\{\mathcal{E}\} = a/b$ then $\text{Prob}\{\mathcal{E}\} = a/(a+b)$. For instance, in betting jargon, if an event (e.g. Italy's victory at the next World Cup) is given "10:1", this

means that it is ten times more likely to happen than not and its probability is $10/11$.

Using odds-ratio can be convenient also to formalise the impact of an observed evidence E on the probability (or degree of belief) that an hypothesis H (e.g. a defendant is innocent) be true. From the Bayes' theorem (2.6.24), it is easy to derive

$$\frac{\text{Prob}\{H|E\}}{\text{Prob}\{\bar{H}|E\}} = \frac{\text{Prob}\{E|H\}}{\text{Prob}\{E|\bar{H}\}} \frac{\text{Prob}\{H\}}{\text{Prob}\{\bar{H}\}} \Rightarrow \text{OR}\{H|E\} = \frac{\text{Prob}\{E|H\}}{\text{Prob}\{E|\bar{H}\}} \text{OR}\{H\}$$

where \bar{H} is the complement of H and

- $\frac{\text{Prob}\{H|E\}}{\text{Prob}\{\bar{H}|E\}}$ is the *posterior odds ratio* (i.e. the odds in favour of the hypothesis H after the evidence has been presented),
- $\frac{\text{Prob}\{H\}}{\text{Prob}\{\bar{H}\}}$ is the *prior odds ratio* (i.e. the odds in favour of the hypothesis H before the evidence has been presented) and
- $\frac{\text{Prob}\{E|H\}}{\text{Prob}\{E|\bar{H}\}}$ is also known as the *likelihood ratio*.

Exercise

Suppose for instance that you want to assess the hypothesis H that if your professor is Italian and that you suspect already something on the basis of his accent ($\text{Prob}\{H\} = 0.6$).

You also know that the love of Italians for football (E) is higher than in the rest of the world. and you translate that into the probabilistic relation $\text{Prob}\{E|H\} = 0.9$ and $\text{Prob}\{E|\bar{H}\} = 0.5$. How would your belief change if you realise that he is a supporter of Fiorentina?

•

2.9 Logics and probabilistic reasoning

This section aims to present some interesting relationships between logic deduction and probabilistic reasoning.

First, we show that we can write down a probabilistic version of the deductive *modus ponens* rule of propositional logic :

If $\mathcal{E}_1 \Rightarrow \mathcal{E}_2$ and \mathcal{E}_1 is true, then \mathcal{E}_2 is true as well.

Since $\mathcal{E}_1 \Rightarrow \mathcal{E}_2$ is equivalent in set terms to $\mathcal{E}_1 \subset \mathcal{E}_2$ we obtain

$$\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} = \frac{\text{Prob}\{\mathcal{E}_1, \mathcal{E}_2\}}{\text{Prob}\{\mathcal{E}_1\}} = \frac{\text{Prob}\{\mathcal{E}_1\}}{\text{Prob}\{\mathcal{E}_1\}} = 1$$

i.e. a translation of the *modus ponens* argument in the probabilistic language. Interestingly enough, the probability theory provides us with a result also in the case of true \mathcal{E}_2 . It is well-known that in propositional logic if $\mathcal{E}_1 \Rightarrow \mathcal{E}_2$ and \mathcal{E}_2 is true, then nothing can be inferred about \mathcal{E}_1 . Probability theory is more informative since in this case we may derive from $\mathcal{E}_2 \subset \mathcal{E}_1$ that

$$\text{Prob}\{\mathcal{E}_1|\mathcal{E}_2\} = \frac{\text{Prob}\{\mathcal{E}_1\}}{\text{Prob}\{\mathcal{E}_2\}} \geq \text{Prob}\{\mathcal{E}_1\}$$

Note that this is a probabilistic formulation of the abduction principle . In other words, probability supports the following commonsense reasoning: if both $\mathcal{E}_1 \Rightarrow \mathcal{E}_2$

and \mathcal{E}_2 apply, then the conditional probability of \mathcal{E}_1 (i.e. the probability of \mathcal{E}_1 once we know that \mathcal{E}_2 occurred) cannot be smaller than the unconditional probability (i.e. the probability of \mathcal{E}_1 if we knew nothing about \mathcal{E}_2).

Also the properties of transitivity and inverse *modus ponens* hold in probability. Let us consider three events $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$. The transitivity principle in logic states that

$$\text{If } \mathcal{E}_1 \Rightarrow \mathcal{E}_2 \text{ and } \mathcal{E}_2 \Rightarrow \mathcal{E}_3 \text{ then } \mathcal{E}_1 \Rightarrow \mathcal{E}_3$$

In probabilistic terms we can rewrite $\mathcal{E}_1 \Rightarrow \mathcal{E}_2$ as

$$\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} = 1$$

and $\mathcal{E}_2 \Rightarrow \mathcal{E}_3$ as

$$\text{Prob}\{\mathcal{E}_3|\mathcal{E}_2\} = 1$$

respectively. From the law of total probability (Equation (2.5.20)) we obtain

$$\text{Prob}\{\mathcal{E}_3|\mathcal{E}_1\} = \text{Prob}\{\mathcal{E}_3|\mathcal{E}_2, \mathcal{E}_1\} \underbrace{\text{Prob}\{\mathcal{E}_2^c|\mathcal{E}_1\}}_0 + \underbrace{\text{Prob}\{\mathcal{E}_3|\mathcal{E}_2, \mathcal{E}_1\}}_1 \underbrace{\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\}}_1 = 1$$

Inverse *modus ponens* in logic states that

$$\text{If } \mathcal{E}_1 \Rightarrow \mathcal{E}_2 \text{ then } \neg\mathcal{E}_2 \Rightarrow \neg\mathcal{E}_1$$

In probabilistic terms from $\text{Prob}\{\mathcal{E}_2|\mathcal{E}_1\} = 1$ it follows

$$\text{Prob}\{\mathcal{E}_1^c|\mathcal{E}_2^c\} = 1 - \text{Prob}\{\mathcal{E}_1|\mathcal{E}_2^c\} = 1 - \frac{\underbrace{\text{Prob}\{\mathcal{E}_2^c|\mathcal{E}_1\}}_0 \text{Prob}\{\mathcal{E}_1\}}{\text{Prob}\{\mathcal{E}_2^c\}} = 1$$

Those results show that deductive logic rules can be seen as limiting cases of probabilistic reasoning and confirm the compatibility of probability reasoning with human common sense.

2.10 Random numeric variables

Machine learning and statistics are concerned with numeric data and measurements while so far we have mainly been dealing with events and categorical variables. What is then the link between the notion of random experiment 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 $\mathbf{z} : \Omega \rightarrow \mathcal{Z} \subset \mathbb{R}$ such that we can associate with each experimental outcome ω a real value $z = \mathbf{z}(\omega)$ 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 ω . 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 with each event \mathcal{E} and we have a mapping from events to real values, a probability distribution can be associated with \mathbf{z} .

Definition 10.1 (Random numeric variable). Given a random experiment $(\Omega, \{\mathcal{E}\}, \text{Prob}\{\cdot\})$, a random variable \mathbf{z} is the result of a mapping $\mathbf{z} : \Omega \rightarrow \mathcal{Z}$ that assigns a number z to every outcome ω . This mapping must satisfy the following two conditions:

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

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

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

$$\mathbf{z}^{-1}(I) = \{\omega \in \Omega | \mathbf{z}(\omega) \in I\} \quad (2.10.30)$$

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

$$\text{Prob}\{\mathbf{z} \in I\} = \text{Prob}\{\mathbf{z}^{-1}(I)\} = \text{Prob}\{\omega \in \Omega | \mathbf{z}(\omega) \in I\} \quad (2.10.31)$$

$$\text{Prob}\{\mathbf{z} = z\} = \text{Prob}\{\mathbf{z}^{-1}(z)\} = \text{Prob}\{\omega \in \Omega | \mathbf{z}(\omega) = z\} \quad (2.10.32)$$

In other words, a *random numeric 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. The notion of r.v. formalises the notion of numeric measurements, which is indeed a mapping between an event (e.g. your body temperature) and a number (e.g. in the range $\mathcal{Z} = \{35, \dots, 41\}$ returned by the thermometer). Another 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 $\mathcal{Z} = \{2, \dots, 12\}$ (or $\mathcal{Z} = \{1, \dots, 6\}$). In what follows, unless explicitly stated, we will use the term random variables to refer to random numeric variables.

Example

Suppose that we have to decide when to go home and watch Fiorentina AC playing the Champion's League final match against Anderlecht. In order to make such 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 that is not constant: for example, $z_1 = 10$ minutes, $z_2 = 23$ minutes, $z_3 = 17$ minutes, where z_i is 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 Ω (depending, for example, on the weather conditions, the weekday, the sports events in town, and so on). The probabilistic approach uses a random variable to represent this uncertainty and considers each measure z_i as the consequence of a random outcome ω_i . The use of a random variable \mathbf{z} to represent the commute time becomes then a *compact (and approximate)* way of modelling the disparate set of causes underlying the uncertainty of this phenomenon. Whatever its limits, the probabilistic representation provides us with a computational way to decide when to leave if we want to bound the probability of missing the start of the match.

•

2.11 Discrete random variables

A discrete random variable \mathbf{z} is an uncertain quantity that can take a discrete number of mutually exclusive and exhaustive values in \mathcal{Z} . Its *probability (mass) function* is the combination of

1. the countable 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

$$P_{\mathbf{z}}(z) = \text{Prob}\{\mathbf{z} = z\} \quad (2.11.33)$$

This function is called *probability function* or *probability mass function*. Note that hereafter we will use $P(z)$ as a shorthand for $\text{Prob}\{\mathbf{z} = z\}$ when the identity of the random variable is clear from the context.

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

2.11.1 Parametric probability function

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

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

where θ is some fixed nonzero real number.

Whatever the value of θ , $P_{\mathbf{z}}(z) > 0$ 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 θ is unknown. We call θ 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 formalise a family of probabilistic models and that the problem of estimation can be seen as a parameter identification task.

2.11.2 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} , the use of quantity is often impractical since it requires to deal with as many values as the size of \mathcal{Z} . In practice, it is more convenient to deal with some compact representations 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 tendency⁷.

Definition 11.1 (Expected value). The *expected value* of a discrete random variable \mathbf{z} is

$$E[\mathbf{z}] = \mu = \sum_{z \in \mathcal{Z}} z P_{\mathbf{z}}(z) \quad (2.11.34)$$

assuming that the sum is well-defined.

An interesting property of the expected value is that it is the value which minimises the squared deviation

$$\mu = \arg \min_c E[(\mathbf{z} - c)^2] \quad (2.11.35)$$

Note that:

- the expected value is a theoretical notion which assumes the existence of a probability function $P_{\mathbf{z}}$;

⁷This concept was first introduced in the 17th century by C. Huygens in order to study the games of chance

- the expected value is defined without reference to any particular sample;
- the expected value does not necessarily belong to the domain \mathcal{Z} of the random variable;
- in this book the term *mean* is used as a synonym of *expected value*;
- the term *average* is not a synonymous of *expected value* since it refers to a particular sample (set of observations of \mathbf{z}). In particular, given an r.v. \mathbf{z} it exists a single $E[\mathbf{z}]$ but there can be as many samples (and averages) as we like.

We will discuss in detail the difference between mean and sample average in Section 3.3.2.

Example [165]

Let us consider a European roulette with numbers $0, 1, \dots, 36$ and where the number 0 is considered as winning for the house. The gain of a player who places a 1\$ bet on a single number is a random variable \mathbf{z} whose sample space is $\mathcal{Z} = \{-1, 35\}$. In other words, only two outcomes are possible: either she wins $z_1 = -1\$$ (or better he 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

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

This means that 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 summarise the spread of a distribution is provided by the variance.

Definition 11.2 (Variance). The *variance* of a discrete random variable \mathbf{z} is

$$\text{Var}[\mathbf{z}] = \sigma^2 = E[(\mathbf{z} - E[\mathbf{z}])^2] = \sum_{z \in \mathcal{Z}} (z - E[\mathbf{z}])^2 P_{\mathbf{z}}(z)$$

The variance is a measure of the dispersion of the probability function of the random variable around its mean μ . Note that the following relation holds

$$\sigma^2 = E[(\mathbf{z} - E[\mathbf{z}])^2] = E[\mathbf{z}^2 - 2\mathbf{z}E[\mathbf{z}] + (E[\mathbf{z}])^2] \quad (2.11.36)$$

$$= E[\mathbf{z}^2] - (E[\mathbf{z}])^2 = E[\mathbf{z}^2] - \mu^2 \quad (2.11.37)$$

whatever is the probability function of \mathbf{z} . Figure 2.7 illustrates two discrete r.v. probability functions that have the same mean but different variance. Note that the variance $\text{Var}[\mathbf{z}]$ does not have the same dimension as the values of \mathbf{z} . For instance, if \mathbf{z} is measured in the unit $[m]$, $\text{Var}[\mathbf{z}]$ is expressed in the unit $[m]^2$. Standard deviation is a measure for the spread that has the same dimension as \mathbf{z} . An alternative measure of spread is $E[|\mathbf{z} - \mu|]$ but this quantity is less used since more difficult to be analytically manipulated than the variance.

Definition 11.3 (Standard deviation). The *standard deviation* of a discrete random variable \mathbf{z} is the positive square root of the variance.

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

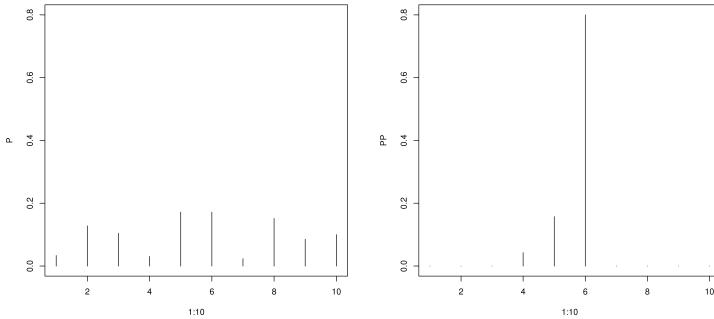


Figure 2.7: Two discrete probability functions with the same mean and different variance

Example

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

$$E[\mathbf{z}] = p * 1 + 0 * (1 - p) = p \quad (2.11.38)$$

$$E[\mathbf{z}^2] = p * 1 + 0 * (1 - p) = p \quad (2.11.39)$$

$$\text{Var} [\mathbf{z}] = E[\mathbf{z}^2] - (E[\mathbf{z}])^2 = p - p^2 = p(1 - p) \quad (2.11.40)$$

•

Definition 11.4 (Moment). For any positive integer r , the r th moment of the probability function is

$$\mu_r = E[\mathbf{z}^r] = \sum_{z \in \mathcal{Z}} z^r P_{\mathbf{z}}(z) \quad (2.11.41)$$

Note that the first moment coincides with the mean μ , while the second moment is related to the variance according to Equation (2.11.36). Higher-order moments provide additional information, other than the mean and the spread, about the shape of the probability function.

Definition 11.5 (Skewness). The skewness of a discrete random variable \mathbf{z} is defined as

$$\gamma = \frac{E[(\mathbf{z} - \mu)^3]}{\sigma^3} \quad (2.11.42)$$

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 2.8).

Definition 11.6 (Kurtosis). The kurtosis of a discrete random variable \mathbf{z} is defined as

$$\kappa = \frac{E[(\mathbf{z} - \mu)^4]}{\sigma^4} \quad (2.11.43)$$

Kurtosis is always positive. Its interpretation is that the probability function of a distribution with large kurtosis has fatter tails, compared with the probability function of a distribution with smaller kurtosis.

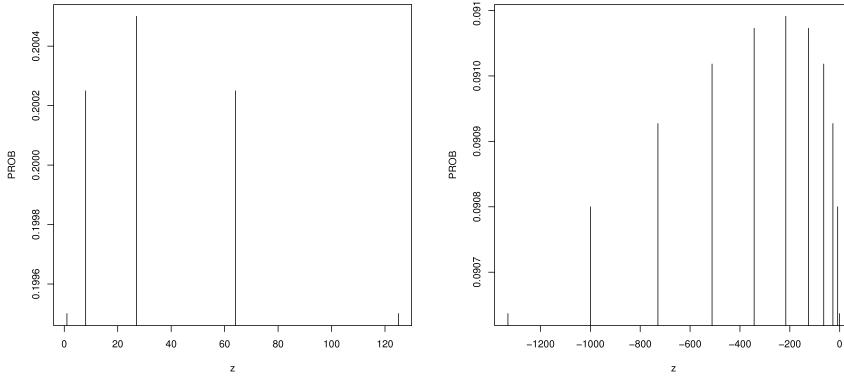


Figure 2.8: A discrete probability function with positive skewness (left) and one with a negative skewness (right).

2.11.3 Entropy of a discrete r.v.

Definition 11.7 (Entropy). Given a discrete r.v. \mathbf{z} , the *entropy* of the probability function $P_{\mathbf{z}}(z)$ is defined by

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

Note that if we use logarithm in base 2 (e) then the entropy is said to be measured in *bits (nats)*. $H(\mathbf{z})$ 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 maximised (and takes the value $\log M$) if the distribution is uniform, i.e. $P_{\mathbf{z}}(z) = 1/M$ for all $z \in \mathcal{Z}$. It is minimised iff $P(z) = 1$ for a single value of \mathbf{z} (i.e. all other probability values are null).

Although the entropy measures (like the variance) the uncertainty of an 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 seen as a function of the probability function $P_{\mathbf{z}}$ rather than of \mathbf{z} .

2.11.4 Kullback-Leibler divergence and cross-entropy

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

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

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

$$H(P_0||P_1) = D_{KL}(P_0, P_1) = \sum_z P_0(z) \log \frac{P_0(z)}{P_1(z)} = \quad (2.11.44)$$

$$= \sum_z P_0(z) \log P_0(z) - \sum_z P_0(z) \log P_1(z) \quad (2.11.45)$$

$$H(P_1||P_0) = D_{KL}(P_1, P_0) = \sum_z P_1(z) \log \frac{P_1(z)}{P_0(z)} = \quad (2.11.46)$$

$$= \sum_z P_1(z) \log P_1(z) - \sum_z P_1(z) \log P_0(z) \quad (2.11.47)$$

where the term

$$-\sum_z P_0(z) \log P_1(z) = -E_{\mathbf{z} \sim P_0}[\log P_1] \quad (2.11.48)$$

is also called the *cross-entropy*. It follows that the Kullback-Leibler divergence between two probability distribution P_0 and P_1 is the sum of the cross-entropy and the negative of the entropy of P_0 . Note also that it can be shown the D_{KL} terms are always non-negative.

The asymmetric KL divergences measure the dissimilarity between the two probability functions. A symmetric formulation of the dissimilarity is provided by the *divergence* quantity

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

2.12 Continuous random numeric variables

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

Definition 12.1 (Cumulative distribution function). The (*cumulative*) *distribution function* of \mathbf{z} is the function $F_{\mathbf{z}} : \mathbb{R} \rightarrow [0, 1]$

$$F_{\mathbf{z}}(z) = \text{Prob}\{\mathbf{z} \leq z\} \quad (2.12.49)$$

This function satisfies the following two conditions:

1. it is right continuous: $F_{\mathbf{z}}(z) = \lim_{y \rightarrow z} F_{\mathbf{z}}(y)$,
2. it is non-decreasing: $z_1 < z_2$ implies $F_{\mathbf{z}}(z_1) \leq F_{\mathbf{z}}(z_2)$,
3. it is normalised, i.e.

$$\lim_{z \rightarrow -\infty} F_{\mathbf{z}}(z) = 0, \quad \lim_{z \rightarrow \infty} F_{\mathbf{z}}(z) = 1$$

Definition 12.2 (Density function). The *density function* of a real random variable \mathbf{z} is the derivative of the distribution function

$$p_{\mathbf{z}}(z) = \frac{dF_{\mathbf{z}}(z)}{dz} \quad (2.12.50)$$

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

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

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

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 (since it is a density and not a probability) and even unbounded,
- $p_{\mathbf{z}}(z)dz$ is the probability that \mathbf{z} takes a value in the interval $[z, z + dz]$ as $dz \rightarrow 0$,

- the *support* of a continuous r.v. is the set of values that are assigned non-zero density,
- two r.v.s \mathbf{z}_1 and \mathbf{z}_2 with the same domain \mathcal{Z} are equal in distribution if $F_{\mathbf{z}_1}(z) = F_{\mathbf{z}_2}(z)$ for all $z \in \mathcal{Z}$.

Note that hence after we will use $p(z)$ as a shorthand for $p_{\mathbf{z}}(z)$ when the identity of the random variable is clear from the context.

2.12.1 Mean, variance, moments of a continuous r.v.

Consider a continuous scalar r.v. with range $\mathcal{Z} = (l, h)$ and density function $p(z)$. We may define some quantities, which summarise key properties of the probability distribution.

Definition 12.3 (Expectation or mean). The mean (or expectation) of a continuous scalar r.v. \mathbf{z} is the scalar value

$$\mu = E[\mathbf{z}] = \int_l^h z p(z) dz \quad (2.12.51)$$

if the integral is finite.

Definition 12.4 (Variance). The variance of a continuous scalar r.v. \mathbf{z} is the scalar value

$$\sigma^2 = E[(\mathbf{z} - \mu)^2] = \int_l^h (z - \mu)^2 p(z) dz \quad (2.12.52)$$

Definition 12.5 (Moments). The r -th moment of a continuous scalar r.v. \mathbf{z} is the scalar value

$$\mu_r = E[\mathbf{z}^r] = \int_l^h z^r p(z) dz \quad (2.12.53)$$

The r -th central moment of a continuous scalar r.v. \mathbf{z} is the scalar value

$$\mu_r = E[(\mathbf{z} - \mu)^r] = \int_l^h (z - \mu)^r p(z) dz \quad (2.12.54)$$

It follows that the moment of order $r = 1$ coincides with the mean and the central moment of order $r = 2$ coincides with the variance of \mathbf{z} .

Definition 12.6 (Mode). The value z^* is called the mode of \mathbf{z} if $p_{\mathbf{z}}(z^*) \geq p_{\mathbf{z}}(z)$ for all z .

A probability density can have several modes and in this case it is called *multimodal*.

Definition 12.7 (Quantile function). Given the cumulative function $F_{\mathbf{z}}$, the quantile (or inverse cumulative) function is the function $F_{\mathbf{z}}^{-1} : [0, 1] \rightarrow \mathbb{R}$ such that

$$F_{\mathbf{z}}^{-1}(q) = \inf\{z : F_{\mathbf{z}}(z) > q\}$$

The quantities $F_{\mathbf{z}}^{-1}(1/4)$, $F_{\mathbf{z}}^{-1}(1/2)$, $F_{\mathbf{z}}^{-1}(3/4)$ are called the *lower quartile*, the *median* and the *upper quartile*, respectively. Note that the mean, median and mode are also called measures of *central tendency* [27] since they provide information about the center of the probability mass.

Also, while the mean minimizes the mean-squared term in (2.11.35), it can be shown the median m satisfies

$$m = \arg \min_c E[|\mathbf{z} - c|] \quad (2.12.55)$$

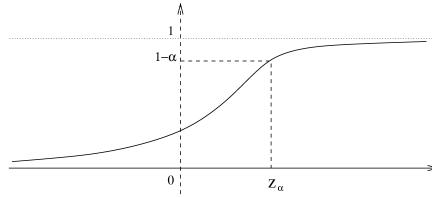


Figure 2.9: Cumulative distribution function and upper critical point.

Definition 12.8 (Upper critical point). For a given $0 \leq \alpha \leq 1$ the *upper critical point* of a continuous r.v. \mathbf{z} is the value z_α such that

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

Figure 2.9 shows an example of cumulative distribution together with the upper critical point. A compact review of univariate discrete and continuous distributions is available in Appendix D.1. In what follows we will detail only the univariate normal case.

2.12.2 Univariate Normal (or Gaussian) distribution

A continuous scalar random variable \mathbf{x} is said to be *normally distributed* with parameters μ and σ^2 (also $\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2)$) if its probability density function is Normal (or Gaussian). The analytical form of a Normal probability density function is

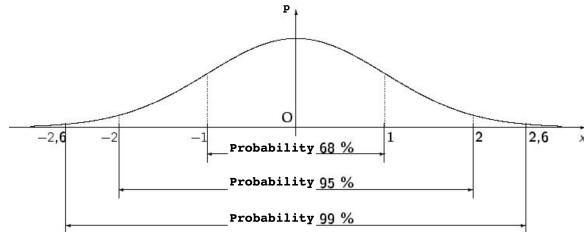
$$p_{\mathbf{x}}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2.12.56)$$

where the coefficient before the exponential ensures that $\int p_{\mathbf{x}}(x)dx = 1$. The mean of the Normal random variable \mathbf{x} is μ and its variance is σ^2 . An interesting property of a normal r.v. is that the probability that an observation x is within 1 (2) standard deviations from the mean is 0.68 (0.95). You may find more probabilistic relationships in Table 2.4. When $\mu = 0$ and $\sigma^2 = 1$ the distribution is called *standard normal* (Figure 2.10) and its distribution function is denoted $F_{\mathbf{z}}(z) = \Phi(z)$. All random variables $\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2)$ are linked to a standard variable \mathbf{z} by the following relation

$$\mathbf{z} = (\mathbf{x} - \mu)/\sigma. \quad (2.12.57)$$

It follows that $\mathbf{z} \sim \mathcal{N}(0, 1) \Rightarrow \mathbf{x} = \mu + \sigma\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$.

The practitioner might now wonder why the Normal distribution is so ubiquitous in statistics books and literature. There are plenty of reasons both from the theoretical and the practical side. From a theoretical perspective, the adoption of a Normal distribution is justified by the Central Limit theorem (Appendix D.9) which states that, under conditions almost always satisfied in practice, a linear combination of random variables converges to a Normal distribution. This is particularly useful if we wish to represent in a compact lumped form the variability that escapes to a modelling effort (e.g. the regression plus noise form in Section 8.1). Another relevant property of Gaussian distributions is that they are invariant to linear transformations, i.e. a linear transformation of a Gaussian r.v. is still Gaussian, and its mean (variance) depends on the mean (variance) of the original r.v.. Also, it can be shown that the normal distribution is the one with the highest differential entropy (Section 2.15.5) over all continuous distributions with a given mean and variance and a support from $-\infty$ to $+\infty$: this means that if we only know the mean and variance of a distribution (and nothing more), the most adequate (and

Figure 2.10: Density of a standard r.v. $\mathcal{N}(0, 1)$

$\text{Prob}\{\mu - \sigma \leq x \leq \mu + \sigma\} \approx 0.683$
$\text{Prob}\{\mu - 1.282\sigma \leq x \leq \mu + 1.282\sigma\} \approx 0.8$
$\text{Prob}\{\mu - 1.645\sigma \leq x \leq \mu + 1.645\sigma\} \approx 0.9$
$\text{Prob}\{\mu - 1.96\sigma \leq x \leq \mu + 1.96\sigma\} \approx 0.95$
$\text{Prob}\{\mu - 2\sigma \leq x \leq \mu + 2\sigma\} \approx 0.954$
$\text{Prob}\{\mu - 2.57\sigma \leq x \leq \mu + 2.57\sigma\} \approx 0.99$
$\text{Prob}\{\mu - 3\sigma \leq x \leq \mu + 3\sigma\} \approx 0.997$

Table 2.4: Some probabilistic relations holding for $x \in \mathcal{N}(\mu, \sigma^2)$

less constraining) distribution to assume is the normal one. From a more pragmatic perspective, an evident asset of a Gaussian representation is that only a finite number of parameters (two in the univariate case) are sufficient to characterise the entire distribution. However, it is worth to mention that it has also some limitations, like assigning non-zero probability to large positive and negative values, which may not be necessarily appropriate for the quantity under consideration.

Exercise

Test yourself the relations in Table 2.4 by random sampling and simulation using the script `Probability/norm.R`.

•

2.13 Joint probability

So far, we considered scalar random variables only. However, the most interesting probabilistic (and machine learning) applications are multivariate, i.e. concerning a number of variables larger than one. 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.s. In other terms, the joint probability contains all the information about the random variables.

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

$$\text{Prob}\{\mathbf{z}_1 = z_1, \mathbf{z}_2 = z_2, \dots, \mathbf{z}_n = z_n\} = P(z_1, z_2, \dots, z_n) \quad (2.13.58)$$

for every possible assignment of values z_1, \dots, z_n to the variables. The joint distribution can be represented by a table like Table 2.5 listing the probabilities for all possible assignments of values.

Spam mail example

Let us consider a bivariate probabilistic model to describe 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 given by the table

	$\mathbf{z}_1 = 0$	$\mathbf{z}_1 = 1$	$P_{\mathbf{z}_2}$
$\mathbf{z}_2 = 0$	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

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

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

2.13.1 Marginal and conditional probability

Let $\{\mathbf{z}_1, \dots, \mathbf{z}_m\}$ be a subset of size m of the n discrete r.v.s for which a joint probability function (2.13.58) is defined. The *marginal probabilities* for the subset can be obtained according to (2.5.20) by summing over all possible combinations of values for the remaining variables.

$$P(z_1, \dots, z_m) = \sum_{\tilde{z}_{m+1}} \dots \sum_{\tilde{z}_n} P(z_1, \dots, z_m, \tilde{z}_{m+1}, \dots, \tilde{z}_n) \quad (2.13.59)$$

The operation of deriving a marginal⁸ probability from a joint distribution is also known as *marginalisation* and relies on the third axiom of probability (Section 2.1.1) in the case of disjoint events.

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

$$p(z_1, \dots, z_m) = \int p(z_1, \dots, z_m, z_{m+1}, \dots, z_n) dz_{m+1} \dots dz_n \quad (2.13.60)$$

This is also known as the *sum rule* or the *marginalisation property*.

The following definition for r.v. derives directly from Equation (2.3.8).

Definition 13.1 (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

$$P(\{z_i : i \in S_1\} | \{z_j : j \in S_2\}) = \frac{P(\{z_i : i \in S_1\}, \{z_j : j \in S_2\})}{P(\{z_j : j \in S_2\})}$$

⁸Note that the notion of marginal has opposite meanings in statistics and economics: in statistics it refers to a sum (or integral) while in economics (e.g. marginal cost) it refers to a change (or derivative). See also https://statmodeling.stat.columbia.edu/2006/05/01/marginal_and_ma/.

Definition 13.2 (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

$$p(\{z_i : i \in S_1\} | \{z_j : j \in S_2\}) = \frac{p(\{z_i : i \in S_1\}, \{z_j : j \in S_2\})}{p(\{z_j : j \in S_2\})} \quad (2.13.61)$$

where $p(\{z_j : j \in S_2\})$ is the marginal density of the set S_2 of variables. When $p(\{z_j : j \in S_2\}) = 0$ this quantity is not defined.

The simplified version of (2.13.61) for two r.v.s \mathbf{z}_1 and \mathbf{z}_2 is

$$\begin{aligned} p(\mathbf{z}_1 = z_1, \mathbf{z}_2 = z_2) &= \\ &= p(\mathbf{z}_2 = z_2 | \mathbf{z}_1 = z_1) p(\mathbf{z}_1 = z_1) = p(\mathbf{z}_1 = z_1 | \mathbf{z}_2 = z_2) p(\mathbf{z}_2 = z_2) \end{aligned} \quad (2.13.62)$$

which is also known as the *product rule*.

By combining (2.12.51), the sum rule (2.13.60) and the product rule (2.13.62) we obtain

$$p(z_1) = \int p(z_1, z_2) dz_2 = \int p(z_1 | z_2) p(z_2) dz_2 = E_{\mathbf{z}_2}[p(z_1 | z_2)]$$

where the subscript \mathbf{z}_2 makes clear that the expectation is computed with respect to the distribution of \mathbf{z}_2 only (while z_1 is fixed).

2.13.2 Bayes' theorem for discrete and continuous variables

From the definitions in the previous section, it is possible to derive a set of versions of the Bayes' theorem (Section 2.6) for different combinations of discrete and continuous random variables. Given two discrete r.v.s we have

$$\text{Prob}\{\mathbf{z}_1 = z_1 | \mathbf{z}_2 = z_2\} = \frac{\text{Prob}\{\mathbf{z}_2 = z_2 | \mathbf{z}_1 = z_1\} \text{Prob}\{\mathbf{z}_1 = z_1\}}{\text{Prob}\{\mathbf{z}_2 = z_2\}} \quad (2.13.63)$$

Given two continuous r.v.s we have

$$p(\mathbf{z}_1 = z_1 | \mathbf{z}_2 = z_2) = \frac{p(\mathbf{z}_2 = z_2 | \mathbf{z}_1 = z_1) p(\mathbf{z}_1 = z_1)}{p(\mathbf{z}_2 = z_2)} \quad (2.13.64)$$

In the case of a discrete \mathbf{z}_1 and a continuous \mathbf{z}_2 we have

$$\text{Prob}\{\mathbf{z}_1 = z_1 | \mathbf{z}_2 = z_2\} = \frac{p(\mathbf{z}_2 = z_2 | \mathbf{z}_1 = z_1) \text{Prob}\{\mathbf{z}_1 = z_1\}}{p(\mathbf{z}_2 = z_2)} \quad (2.13.65)$$

while for a continuous \mathbf{z}_1 and a discrete \mathbf{z}_2 we have

$$p(\mathbf{z}_1 = z_1 | \mathbf{z}_2 = z_2) = \frac{\text{Prob}\{\mathbf{z}_2 = z_2 | \mathbf{z}_1 = z_1\} p(\mathbf{z}_1 = z_1)}{\text{Prob}\{\mathbf{z}_2 = z_2\}} \quad (2.13.66)$$

2.13.3 Independence

Having defined the joint and the conditional probability, we can now define when two random variables are independent.

Definition 13.3 (Independent discrete random variables). Two discrete random variables \mathbf{x} and \mathbf{y} are defined to be *statistically independent* (written as $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$) if the joint probability

$$\text{Prob}\{\mathbf{x} = x, \mathbf{y} = y\} = \text{Prob}\{\mathbf{x} = x\} \text{Prob}\{\mathbf{y} = y\}, \quad \forall x, y \quad (2.13.67)$$

The definition can be easily extended to the continuous case.

Definition 13.4 (Independent continuous random variables). Two continuous variables \mathbf{x} and \mathbf{y} are defined to be *statistically independent* (written as $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$) if the joint density

$$p(\mathbf{x} = \mathbf{x}, \mathbf{y} = \mathbf{y}) = p(\mathbf{x} = \mathbf{x})p(\mathbf{y} = \mathbf{y}), \quad \forall \mathbf{x}, \mathbf{y} \quad (2.13.68)$$

From the definition of independence and conditional density, it follows that

$$\mathbf{x} \perp\!\!\!\perp \mathbf{y} \Leftrightarrow p(\mathbf{x} = \mathbf{x} | \mathbf{y} = \mathbf{y}) = p(\mathbf{x} = \mathbf{x}) \quad \forall \mathbf{x}, \mathbf{y} \quad (2.13.69)$$

In layman's terms, the independence of two variables means that we do not expect that the observed outcome of one variable will affect the probability of observing the other, or equivalently that knowing something about one variable adds no information about the other. For instance, hair colour and gender are independent. Knowing someone's hair colour adds nothing to the knowledge of his gender. Height and weight are dependent, however. Knowing someone's height does not determine precisely their weight: nevertheless, you have less uncertainty about his probable weight after you have been told the height.

Though independence is symmetric

$$\mathbf{x} \perp\!\!\!\perp \mathbf{y} \Leftrightarrow \mathbf{y} \perp\!\!\!\perp \mathbf{x}$$

it is neither reflexive (i.e. a variable is not independent of itself) nor transitive. In other terms, 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.

If we consider three instead of two variables, they are said to be mutually independent if and only if each pair of r.v.s. is independent and

$$p(x, y, z) = p(x)p(y)p(z)$$

Also the relationship

$$\mathbf{x} \perp\!\!\!\perp (\mathbf{y}, \mathbf{z}) \Rightarrow \mathbf{x} \perp\!\!\!\perp \mathbf{z}, \mathbf{x} \perp\!\!\!\perp \mathbf{y}$$

holds, but not the one in the opposite direction.

Note that in mathematical terms an independence assumption implies that a bivariate density function can be written in a simple form, i.e. as the product of two univariate densities. This results in an important benefit in terms of the size of the parametrisation. For instance, consider two discrete random variables $\mathbf{z}_1 \in \mathcal{Z}_1$, $\mathbf{z}_2 \in \mathcal{Z}_2$ such that the cardinality of the two ranges is k_1 and k_2 , respectively. In the generic case, if \mathbf{z}_1 and \mathbf{z}_2 are not independent, the definition of the joint probability requires the definition of $k_1 k_2 - 1$ terms⁹ (or parameters). In the independent case because of the property (2.13.68), the definition requires $k_1 - 1$ terms for \mathbf{z}_1 and $k_2 - 1$ terms for \mathbf{z}_2 , so overall $k_1 + k_2 - 2$. This makes a big difference in case of large values of k_1 and k_2 .

Independence allows an economic parametrisation in the multivariate case as well. Consider the case of a large number n of binary discrete r.v.s., i.e. each having a range made of two values. If we need to define the joint probability, we require $2^n - 1$ terms (or parameters) in the generic case. If the n variables are independent, this number is reduced to n .

Exercise

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

•

Note that hence after, for the sake of brevity, we will limit to introduce definitions for continuous random variables only. All of them can, however, be extended to the discrete case too.

⁹minus one because of the normalisation constraint

2.13.4 Chain rule

Given a set of n random variables, from (2.4.18) it is possible to derive the *chain rule* (also called the *general product rule*) which returns the joint density as a function of conditional densities:

$$p(z_n, \dots, z_1) = p(z_n|z_{n-1}, \dots, z_1)p(z_{n-1}|z_{n-2}, \dots, z_1) \dots p(z_2|z_1)p(z_1) \quad (2.13.70)$$

This rule is convenient to simplify the representation of large variate distributions by describing them in terms of conditional probabilities.

2.13.5 Conditional independence

Independence is not a stable relation. Though $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$, the r.v. \mathbf{x} may become dependent on \mathbf{y} once we observe the value z of a third variable \mathbf{z} . In the same way, two dependent variables \mathbf{x} and \mathbf{y} may become independent once the value of \mathbf{z} is known. This leads us to introduce the notion of conditional independence.

Definition 13.5 (Conditional independence). Two r.v.s \mathbf{x} and \mathbf{y} are *conditionally independent given the value $\mathbf{z} = z$* ($\mathbf{x} \perp\!\!\!\perp \mathbf{y}|\mathbf{z} = z$) iff

$$p(\mathbf{x} = x, \mathbf{y} = y|\mathbf{z} = z) = p(\mathbf{x} = x|\mathbf{z} = z)p(\mathbf{y} = y|\mathbf{z} = z) \quad \forall x, y \quad (2.13.71)$$

Two r.v.s \mathbf{x} and \mathbf{y} are *conditionally independent given \mathbf{z}* ($\mathbf{x} \perp\!\!\!\perp \mathbf{y}|\mathbf{z}$) iff they are conditionally independent for all values of \mathbf{z} .

Since from the chain rule (2.13.70) we may write

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

it follows that $\mathbf{x} \perp\!\!\!\perp \mathbf{y}|\mathbf{z} = z$ implies the relation

$$p(\mathbf{y} = y|\mathbf{x} = x, \mathbf{z} = z) = p(\mathbf{y} = y|\mathbf{z} = z) \quad (2.13.72)$$

In plain words, the notion of conditional dependence formalises the intuition that a variable may bring (or not) information about a second one, according to the context.

Note that the statement $\mathbf{x} \perp\!\!\!\perp \mathbf{y}|\mathbf{z} = z$ means that \mathbf{x} and \mathbf{y} are independent if $\mathbf{z} = z$ occurs but does not say anything about the relation between \mathbf{x} and \mathbf{y} if $\mathbf{z} = z$ does not occur. It could follow that two variables are independent but not conditional independent (or the other way round). In general independence does not imply conditional independence and conditional independence does not imply independence [27] (as in the example below).

Example: pizzas, dependence and conditional independence

Let \mathbf{y} be a variable representing the quality of a pizza restaurant and \mathbf{x} a variable denoting the nationality of the restaurant. Intuitively, you would prefer (because of higher quality \mathbf{y}) a pizza served in the restaurant "Sole Mio" ($\mathbf{x} = \text{ITALIAN}$), rather than in the restaurant "Tot Straks" ($\mathbf{x} = \text{BELGIAN}$). In probabilistic terms, this means that \mathbf{x} and \mathbf{y} are dependent ($\mathbf{x} \not\perp\!\!\!\perp \mathbf{y}$), i.e. knowing \mathbf{x} reduces the uncertainty we have about \mathbf{y} . However, it is not the restaurant owner who makes your pizza, but the cook (*pizzaiolo*). Let \mathbf{z} represent his nationality. Now you would prefer eating a pizza in a Belgian restaurant where the *pizzaiolo* has Italian origins rather than in an Italian restaurant with a Flemish cook. In probabilistic terms \mathbf{x} and \mathbf{y} become independent once \mathbf{z} (the *pizzaiolo*'s name) is known ($\mathbf{x} \perp\!\!\!\perp \mathbf{y}|\mathbf{z}$).

A joint distribution which satisfies the above conditional independence properties is in Table 2.5. We may verify the (conditional) independence relations (Table 2.6) as follows:

$x(\text{owner})$	$z(\text{cook})$	$y(\text{pizza})$	$P(\mathbf{x} = x, \mathbf{z} = z, \mathbf{y} = y)$
ITALIAN	ITALIAN	GOOD	0.378
BELGIAN	ITALIAN	GOOD	0.168
ITALIAN	BELGIAN	GOOD	0.012
BELGIAN	BELGIAN	GOOD	0.032
ITALIAN	ITALIAN	BAD	0.162
BELGIAN	ITALIAN	BAD	0.072
ITALIAN	BELGIAN	BAD	0.048
BELGIAN	BELGIAN	BAD	0.128

Table 2.5: Joint probability distribution of the *pizzaiolo* example. Note that the sum of the probability column is one.

$x(\text{owner})$	$z(\text{cook})$	$P(\mathbf{y} = \text{GOOD} \mid \mathbf{x} = x, \mathbf{z} = z)$
ITALIAN	ITALIAN	0.7
BELGIAN	ITALIAN	0.7
ITALIAN	BELGIAN	0.2
BELGIAN	BELGIAN	0.2

Table 2.6: Conditional probability distribution of the quality of the pizza in the *pizzaiolo* example. Note that the sum of the conditional probabilities is not one anymore in agreement with (2.7.27).

- Pizza's quality depends on the restaurant ownership $\mathbf{y} \not\perp\!\!\!\perp \mathbf{x}$:

$$\begin{aligned} \text{Prob}\{\mathbf{y} = \text{GOOD}\} &= 0.59 \neq \text{Prob}\{\mathbf{y} = \text{GOOD} \mid \mathbf{x} = \text{IT}\} = 0.65 \\ &\neq \text{Prob}\{\mathbf{y} = \text{GOOD} \mid \mathbf{x} = \text{BE}\} = 0.5 \end{aligned}$$

- Pizza's quality does not depend anymore on the restaurant ownership if the cook is Italian, $\mathbf{y} \perp\!\!\!\perp \mathbf{x} \mid \mathbf{z} = \text{ITALIAN}$:

$$\text{Prob}\{\mathbf{y} = \text{GOOD} \mid \mathbf{z} = \text{IT}\} = 0.7 = \text{Prob}\{\mathbf{y} = \text{GOOD} \mid \mathbf{x} = \text{BE}, \mathbf{z} = \text{IT}\} = \text{Prob}\{\mathbf{y} = \text{GOOD} \mid \mathbf{x} = \text{IT}, \mathbf{z} = \text{IT}\}$$

- Pizza's quality does not depend anymore on the restaurant ownership if the cook is Belgian, $\mathbf{y} \perp\!\!\!\perp \mathbf{x} \mid \mathbf{z} = \text{BELGIAN}$:

$$\text{Prob}\{\mathbf{y} = \text{GOOD} \mid \mathbf{z} = \text{BE}\} = 0.2 = \text{Prob}\{\mathbf{y} = \text{GOOD} \mid \mathbf{x} = \text{BE}, \mathbf{z} = \text{BE}\} = \text{Prob}\{\mathbf{y} = \text{GOOD} \mid \mathbf{x} = \text{IT}, \mathbf{z} = \text{BE}\}$$

•

It can be shown that the following two assertions are equivalent

$$(\mathbf{x} \perp\!\!\!\perp (\mathbf{z}_1, \mathbf{z}_2) \mid \mathbf{y}) \Leftrightarrow (\mathbf{x} \perp\!\!\!\perp \mathbf{z}_1 \mid (\mathbf{y}, \mathbf{z}_2)), \quad (\mathbf{x} \perp\!\!\!\perp \mathbf{z}_2 \mid (\mathbf{y}, \mathbf{z}_1))$$

Also

$$(\mathbf{x} \perp\!\!\!\perp \mathbf{y} \mid \mathbf{z}), (\mathbf{x} \perp\!\!\!\perp \mathbf{z} \mid \mathbf{y}) \Rightarrow (\mathbf{x} \perp\!\!\!\perp (\mathbf{y}, \mathbf{z}))$$

If $(\mathbf{x} \perp\!\!\!\perp \mathbf{y} \mid \mathbf{z}), (\mathbf{z} \perp\!\!\!\perp \mathbf{y} \mid \mathbf{x}), (\mathbf{z} \perp\!\!\!\perp \mathbf{x} \mid \mathbf{y})$ then $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are mutually independent.
If \mathbf{z} is a random vector, the *order* of the conditional independence is equal to the number of variables in \mathbf{z} .

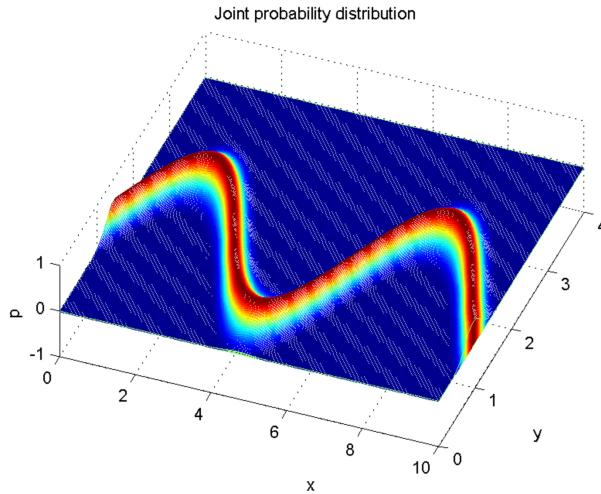


Figure 2.11: 3D visualisation of a bivariate joint density $p(x, y)$.

2.14 Bivariate continuous distribution

Let us consider two continuous r.v. \mathbf{x} and \mathbf{y} and their bivariate joint density function $p_{\mathbf{x},\mathbf{y}}(x, y)$. An example of bivariate joint density function is illustrated in Figure 2.11. From (2.13.60), 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

$$p_{\mathbf{y}|\mathbf{x}}(y|x) = \frac{p(x, y)}{p(x)} \quad (2.14.73)$$

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

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

The definition of *conditional expectation* is obtained from (2.14.73) and (2.12.51).

Definition 14.1 (Conditional expectation). The conditional expectation of \mathbf{y} given $\mathbf{x} = x$ is

$$E_{\mathbf{y}}[\mathbf{y}|\mathbf{x} = x] = \int y p_{\mathbf{y}|\mathbf{x}}(y|x) dy = \mu_{\mathbf{y}|\mathbf{x}}(x) \quad (2.14.74)$$

From (2.11.35) we may derive that

$$E_{\mathbf{y}}[\mathbf{y}|\mathbf{x} = x] = \arg \min_m E_{\mathbf{y}}[(\mathbf{y} - m)^2 | \mathbf{x} = x] \quad (2.14.75)$$

Note that $E_{\mathbf{y}}[\mathbf{y}|\mathbf{x} = x]$ is a function of x also known as the *regression function*.

The definition of *conditional variance* derives from (2.14.73) and (2.12.52).

Definition 14.2 (Conditional variance).

$$\text{Var}[\mathbf{y}|\mathbf{x} = x] = \int (y - \mu_{\mathbf{y}|\mathbf{x}}(x))^2 p_{\mathbf{y}|\mathbf{x}}(y|x) dy \quad (2.14.76)$$

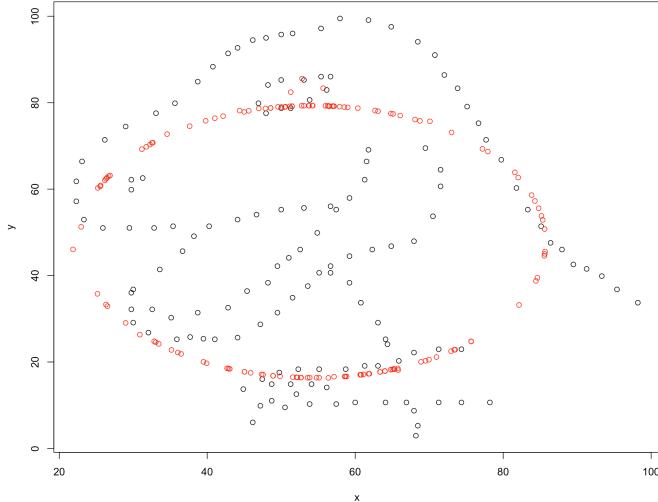


Figure 2.12: Two different bivariate distributions (in black and red) with same mean and variance.

Note that both these quantities are a function of x . If we replace the given value x by the r.v. \mathbf{x} the terms $E_{\mathbf{y}}[\mathbf{y}|\mathbf{x}]$ and $\text{Var}[\mathbf{y}|\mathbf{x}]$ are random, too.

Some important results on their expectation are contained in the following theorems [174].

Theorem 14.3 (Law of total expectation and variance). *For two r.v.s \mathbf{x} and \mathbf{y} , assuming their expectations exist, we have that*

$$E_{\mathbf{x}}[E_{\mathbf{y}}[\mathbf{y}|\mathbf{x} = x]] = E_{\mathbf{y}}[\mathbf{y}] \quad (2.14.77)$$

and

$$\text{Var}[\mathbf{y}] = E_{\mathbf{x}}[\text{Var}[\mathbf{y}|\mathbf{x} = x]] + \text{Var}[E_{\mathbf{y}}[\mathbf{y}|\mathbf{x} = x]] \quad (2.14.78)$$

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

R script

The script `Probability/datasaurus.R` shows that, though mean and variance are informative descriptors, they do not allow to uniquely identify a bivariate distribution, except in specific cases like Gaussian distributions. To illustrate this concept, Figure 2.12 shows two very different discrete bivariate distributions, yet having an identical mean and variance. For more examples of artificially created datasets with same mean and variance, but very different distributions refer to the R package [55].

•

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

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

A 2D representation of a bivariate continuous distribution is illustrated in Figure 2.13. It is worthy noting that, although the conditional distribution is bell-shaped, this is not necessarily the case for the marginal distributions.

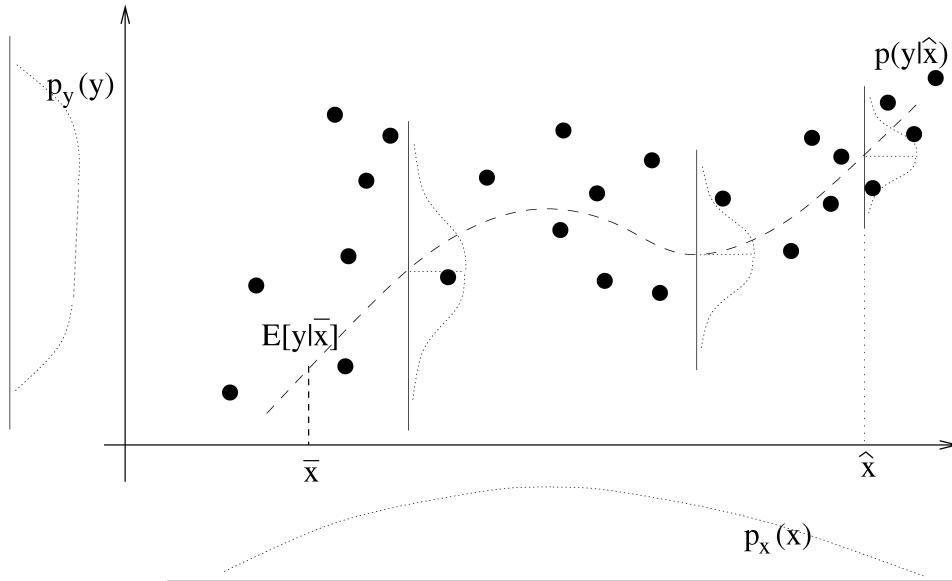


Figure 2.13: Bivariate distribution: the figure shows the two marginal densities (beside the axis), the conditional expectation function (dashed line) and some conditional densities (dotted).

2.14.1 Covariance and 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 $\sigma_{\mathbf{y}}$. The covariance between those two variables is a single-number summary of their joint distribution.

Definition 14.4 (Covariance). The *covariance* between \mathbf{x} and \mathbf{y} is defined as

$$\text{Cov}[\mathbf{x}, \mathbf{y}] = E[(\mathbf{x} - \mu_{\mathbf{x}})(\mathbf{y} - \mu_{\mathbf{y}})] = E[\mathbf{xy}] - \mu_{\mathbf{x}}\mu_{\mathbf{y}} \quad (2.14.79)$$

A positive (negative) covariance means that the two variables are positively (inversely) related, i.e. that once one is above its mean, then the other tends to be above (below) its mean as well. The covariance can take any value in real numbers. A limitation of the covariance is that it depends on variables' scales and units: for instance, if variables were measured in metres instead of centimetres, this would induce a change of their covariance. For this reason, it is common to replace covariance with correlation, a dimensionless measure of linear association.

Definition 14.5 (Correlation). The *correlation coefficient* is defined as

$$\text{Cor}[\mathbf{x}, \mathbf{y}] = \rho(\mathbf{x}, \mathbf{y}) = \frac{\text{Cov}[\mathbf{x}, \mathbf{y}]}{\sqrt{\text{Var}[\mathbf{x}] \text{Var}[\mathbf{y}]}} \quad (2.14.80)$$

It is easily shown that $-1 \leq \rho(\mathbf{x}, \mathbf{y}) \leq 1$. For this reason, the correlation is sometimes expressed as a percentage.

Definition 14.6 (Uncorrelated variables). Two r.v.s \mathbf{x} and \mathbf{y} are said to be *uncorrelated* if $\rho(\mathbf{x}, \mathbf{y}) = 0$ or equivalently if

$$E[\mathbf{xy}] = E[\mathbf{x}]E[\mathbf{y}] \quad (2.14.81)$$

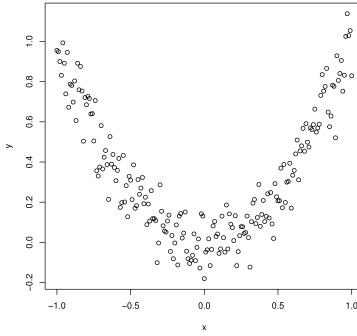


Figure 2.14: Realisations of a pair of dependent yet uncorrelated random variables

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

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

This means that independence implies uncorrelation. However, the opposite is not always true. The equivalence between independence and uncorrelation

$$\rho(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} \perp\!\!\!\perp \mathbf{y} \quad (2.14.82)$$

holds only if \mathbf{x} and \mathbf{y} are jointly Gaussian.

Example

Let $\mathbf{x} \sim \mathcal{N}(0, 1)$ and $\mathbf{y} = \mathbf{x}^2$. By construction, the two variables are dependent. However their covariance (and then their correlation) is null

$$\text{Cov}[\mathbf{x}, \mathbf{y}] = E[(\mathbf{x} - \mu_{\mathbf{x}})(\mathbf{y} - \mu_{\mathbf{y}})] = E[\mathbf{xy}] - \mu_{\mathbf{x}}\mu_{\mathbf{y}} = E[\mathbf{x}^3] = 0$$

since $\mu_{\mathbf{x}} = 0$ and the odd moments of a Gaussian distribution are null. See Figure 2.14 for a scatter plot of the realisations of those two uncorrelated yet dependent variables.

•

Exercises

1. 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 $E[\mathbf{z}] = E[\mathbf{x}] + E[\mathbf{y}]$

2. 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

- $E[\mathbf{z}] = E[\mathbf{x}] + E[\mathbf{y}]$.
- \mathbf{x} and \mathbf{y} are uncorrelated but dependent random variables.

•

2.15 Normal distribution: the multivariate case

Let $\mathbf{z} = [\mathbf{z}_1, \dots, \mathbf{z}_n]^T$ be a $[n, 1]$ random vector. The vector is said to be *normally distributed* if every linear combination of the n r.v.s has a Normal distribution. Its joint probability density function is completely determined by the parameters μ and Σ (also $\mathbf{z} \sim \mathcal{N}(\mu, \Sigma)$) and is given by

$$p_{\mathbf{z}}(z) = \frac{1}{(\sqrt{2\pi})^n \sqrt{\det(\Sigma)}} \exp \left\{ -\frac{1}{2}(z - \mu)^T \Sigma^{-1}(z - \mu) \right\} \quad (2.15.83)$$

where $\det(\Sigma)$ denotes the determinant of the matrix Σ and the quantity

$$\Delta = (z - \mu)^T \Sigma^{-1}(z - \mu) \quad (2.15.84)$$

which appears in the exponent of $p_{\mathbf{z}}$ is called the *Mahalanobis distance* from z to μ . It can be shown that the mean is the $[n, 1]$ vector $E[\mathbf{z}] = \mu$, and that the matrix

$$\Sigma = E[(\mathbf{z} - \mu)(\mathbf{z} - \mu)^T] = E[\mathbf{z}\mathbf{z}^T] - \mu\mu^T \quad (2.15.85)$$

is the $[n, n]$ covariance matrix. This matrix is symmetric and positive semidefinite and has $n(n+1)/2$ separate entries: the diagonal terms Σ_{jj} are the variances $\text{Var}[\mathbf{z}_j]$ of the vector components and the off-diagonal terms $\Sigma_{jk}, j \neq k$ are the covariance terms $\text{Cov}[\mathbf{z}_j, \mathbf{z}_k]$. The covariance matrix can be decomposed as follows

$$\Sigma = U\Lambda U^T = U \begin{bmatrix} \lambda_1, \dots, 0 \\ \vdots, \ddots, \vdots \\ 0, \dots, \lambda_n \end{bmatrix} U^T$$

where U is the orthogonal matrix containing the n eigenvectors $u_j, j = 1, \dots, n$ of Σ and $\lambda_j, j = 1, \dots, n$ are the corresponding eigenvalues. Single-number summaries of the covariance matrix are then the *total variation* $\sum_{j=1}^n \lambda_j$ and the *generalised variance* $\prod_{j=1}^n \lambda_j$. If Σ is non singular, the inverse

$$\Sigma^{-1} = U\Lambda^{-1}U^T$$

is also called the *precision* or *concentration matrix*. The relation between covariance and correlation matrix is discussed in Appendix D.6.

It can be shown that the n -dimensional surfaces of constant probability density (e.g. contour lines if $n = 2$) are hyper-ellipsoids on which Δ^2 is constant and their *principal axes* are given by the eigenvectors $u_j, j = 1, \dots, n$ of Σ which satisfy

$$\Sigma u_j = \lambda_j u_j \quad j = 1, \dots, n$$

where $\lambda_j \geq 0$ are the corresponding eigenvalues which represent the variances along the principal directions (Figure 2.15). If the covariance matrix Σ is *diagonal* then

- 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

$$p_{\mathbf{z}}(z) = \prod_{j=1}^n p_{\mathbf{z}_j}(z_j)$$

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

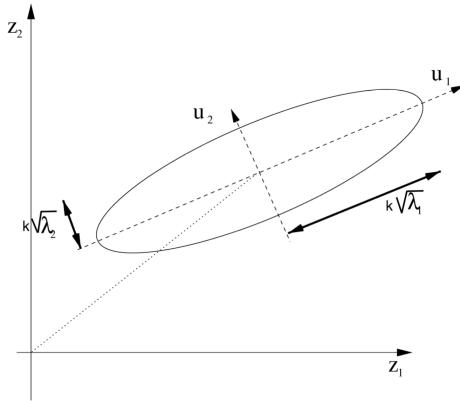


Figure 2.15: Contour curves of bivariate ($n = 2$) Normal distribution.

2.15.1 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

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$$

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 $\mu = [0, 0]^T$ and $\Sigma = [1.2919, 0.4546; 0.4546, 1.7081]$ and a corresponding contour curve are traced in Figure 2.16 by means of the script `Probability/gaussXYZ.R`. We suggest the reader to play with the Shiny dashboard `gaussian.R` in order to visualise the impact of the parameters on the Gaussian distribution.

One of the important properties of the multivariate normal density (2.15.83) 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 $\mathcal{N}(\mu_{2|1}, \sigma_{2|1}^2)$, where

$$\mu_{2|1} = E[\mathbf{z}_2 | \mathbf{z}_1 = z_1] = \mu_2 + \rho \frac{\sigma_2}{\sigma_1} (z_1 - \mu_1) \quad (2.15.86)$$

$$\sigma_{2|1}^2 = \text{Var}[\mathbf{z}_2 | \mathbf{z}_1 = z_1] = \sigma_2^2 (1 - \rho^2) \quad (2.15.87)$$

Note that

- $\mu_{2|1}$ is a linear function of z_1 : if the correlation coefficient ρ 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 μ_2 .

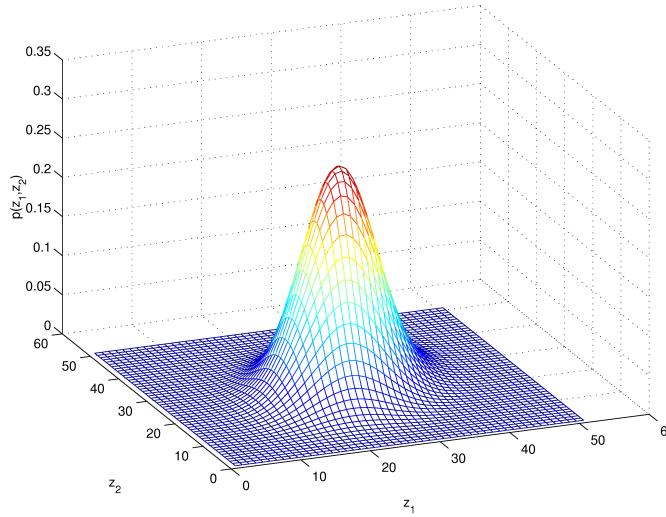


Figure 2.16: Bivariate normal density function

2.15.2 Conditional distribution in the normal multivariate distribution

Let the normal random vector $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2]^T$ be the concatenation of the two sub-vectors \mathbf{z}_1 and \mathbf{z}_2 of size n_1 and n_2 respectively and

$$\mu = E[\mathbf{z}] = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad \Sigma = \text{Var}[\mathbf{z}] = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

It is possible to derive the following multivariate version of (2.15.86) and (2.15.87)

$$\mu_{2|1} = E[\mathbf{z}_2 | \mathbf{z}_1 = z_1] = \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(z_1 - \mu_1) \quad (2.15.88)$$

$$\Sigma_{2|1} = \text{Var}[\mathbf{z}_2 | \mathbf{z}_1 = z_1] = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12} \quad (2.15.89)$$

where μ_2 is a $[n_2, 1]$ vector and Σ_{21} and Σ_{11} are a $[n_2, n_1]$ and $[n_1, n_1]$ matrix, respectively.

2.15.3 Gaussian mixture distribution

A continuous r.v. z has a Gaussian mixture distribution with m components if

$$p(\mathbf{z} = z) = \sum_{k=1}^m w_k \mathcal{N}(z; \mu_k, \Sigma_k) \quad (2.15.90)$$

where $\mathcal{N}(z; \mu_k, \Sigma_k)$ denotes the Normal density with mean μ_k and covariance Σ_k , and the mixture weights w_k satisfy

$$\sum_{k=1}^m w_k = 1, \quad 0 \leq w_k \leq 1$$

A Gaussian mixture is a linear superposition of m Gaussian components and, as such, has a higher expressive power than a unimodal Gaussian distribution: for instance, it can be used to model multimodal density distributions.

The script `Probability/gmm.R` samples a bidimensional mixture of Gaussians with 3 components with diagonal covariances. The density and the sampled points

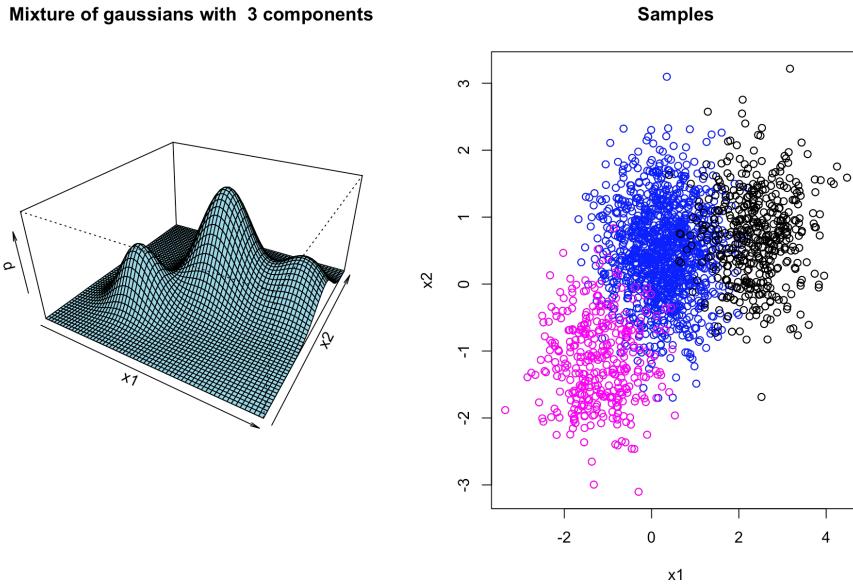


Figure 2.17: Density and observations of a bidimensional mixture of Gaussians with 3 components. Each colour corresponds to a different component.

are in Figure 2.17. An interesting property of Gaussian mixtures is that they are *universal approximator* of densities which means that any smooth density can be approximated with any specific nonzero amount of error by a Gaussian mixture model (GMM) with enough components.

2.15.4 Linear transformations of Gaussian variables

If $\mathbf{z}_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$ and $\mathbf{z}_2 \sim \mathcal{N}(\mu_2, \Sigma_2)$ are independent Gaussian r.v.s., then the sum $\mathbf{z} = \mathbf{z}_1 + \mathbf{z}_2$ is a Gaussian r.v. $\mathbf{z} \sim \mathcal{N}(\mu_1 + \mu_2, \Sigma_1 + \Sigma_2)$.

Given two real constants c_1 and c_2 , the linear combination $\mathbf{z} = c_1\mathbf{z}_1 + c_2\mathbf{z}_2$ is a Gaussian r.v. $\mathbf{z} \sim \mathcal{N}(c_1\mu_1 + c_2\mu_2, c_1^2\Sigma_1 + c_2^2\Sigma_2)$.

If $\mathbf{z} \sim \mathcal{N}(\mu, \Sigma)$ is a $[n, 1]$ Gaussian random vector and $\mathbf{y} = A\mathbf{z} + b$, with A a $[n, n]$ real matrix and b a $[n, 1]$ real vector, then $\mathbf{y} \sim \mathcal{N}(A\mu + b, A\Sigma A^T)$ is a Gaussian vector. Also $\text{Cov}[\mathbf{z}, \mathbf{y}] = \Sigma A^T$ and $\text{Cor}[\mathbf{z}, \mathbf{y}] = V_z^{-1/2} \text{Cov}[\mathbf{z}, \mathbf{y}] V_y^{-1/2}$ where $V_z = \text{diag}(\Sigma)$ and $V_y = \text{diag}(1\Sigma A^T)$.

2.15.5 Entropy in the continuous case

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

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

with the convention that $0 \log 0 = 0$. The entropy is 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} . Note however that, unlike the discrete case (Section 2.11.3) the differential entropy may be negative since the density probability $p(\mathbf{y})$ may be bigger than one.

The computation of the differential entropy for a generic r.v. is typically difficult. The analytical derivation is possible only in simple cases like the normal distribution. For a scalar normal r.v. $\mathbf{y} \sim \mathcal{N}(\mu, \sigma^2)$ we have

$$\begin{aligned} H(\mathbf{y}) &= -E \left[\log \left((2\pi\sigma^2)^{-1/2} \exp \left(-\frac{1}{2\sigma^2} (\mathbf{y} - \mu)^2 \right) \right) \right] = \\ &= \frac{1}{2} (\log 2\pi\sigma^2) + \frac{1}{2\sigma^2} E[(\mathbf{y} - \mu)^2] = \\ &= \frac{1}{2} (\log 2\pi\sigma^2 + 1) \end{aligned} \quad (2.15.91)$$

which shows that the entropy is monotonically related to the variance σ^2 in the normal case. Note that this is not necessarily the case for other distributions.

In the case of a normal random vector $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\} \sim \mathcal{N}(0, \Sigma)$ (Section 2.15)

$$H(\mathbf{Y}) = \frac{n}{2} (1 + \log(2\pi)) + \frac{1}{2} \log \det(\Sigma)$$

where the term $\det(\Sigma)$ is a scalar also known as the *generalised variance* since it generalises the notion of scalar variance to multivariate distributions.

2.15.5.1 Kullback-Leibler divergence

The continuous version of the quantity (2.11.44) is

$$D_{KL}(p_0, p_1) = \int p_0(z) \log \frac{p_0(z)}{p_1(z)} dz \quad (2.15.92)$$

where p_0 and p_1 are two density probabilities of \mathbf{z} .

2.15.5.2 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

$$\begin{aligned} H(\mathbf{x}, \mathbf{y}) &= - \int \int \log(p(x, y)) p(x, y) dx dy = \\ &= E_{\mathbf{x}, \mathbf{y}}[-\log(p(\mathbf{x}, \mathbf{y}))] = E_{\mathbf{x}, \mathbf{y}} \left[\log \frac{1}{p(\mathbf{x}, \mathbf{y})} \right] \end{aligned}$$

The *conditional entropy* is defined as

$$\begin{aligned} H(\mathbf{y}|\mathbf{x}) &= - \int \int \log(p(y|x)) p(x, y) dx dy = E_{\mathbf{x}, \mathbf{y}}[-\log(p(\mathbf{y}|\mathbf{x}))] = \\ &= E_{\mathbf{x}, \mathbf{y}} \left[\log \frac{1}{p(\mathbf{y}|\mathbf{x})} \right] = E_{\mathbf{x}}[H(\mathbf{y}|x)] \end{aligned}$$

This term represents the expected entropy of the conditional distribution of \mathbf{y} given \mathbf{x} or in other terms the remaining uncertainty about \mathbf{y} once \mathbf{x} is known. It follows that $H(\mathbf{y}|\mathbf{x}) = 0$ if \mathbf{y} is completely determined by \mathbf{x} , e.g. \mathbf{y} is a deterministic function of \mathbf{x} . Also, if $\mathbf{y} \perp\!\!\!\perp \mathbf{x}$ then $H(\mathbf{y}|\mathbf{x}) = H(\mathbf{y})$.

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

$$H(\mathbf{y}, \mathbf{x}) = H(\mathbf{y}|\mathbf{x}) + H(\mathbf{x}) \quad (2.15.93)$$

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. $\mathbf{x} \perp\!\!\!\perp \mathbf{y}$. This property formalises a fundamental principle underlying machine learning, data science and prediction in general, i.e. that by conditioning on some variables \mathbf{x} (e.g. inputs) we may reduce the uncertainty about a variable \mathbf{y} (target). Another interesting property is the *independence bound*

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

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

2.15.6 Mutual information

Mutual information is one of the most widely used measures to convey the 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 $p_{(\mathbf{x},\mathbf{y})}(x, y)$:

$$I(\mathbf{x}; \mathbf{y}) = \int \int \log \frac{p(x, y)}{p(x)p(y)} p(x, y) dx dy = H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x}) = H(\mathbf{x}) - H(\mathbf{x}|\mathbf{y}) \quad (2.15.94)$$

with the convention that $0 \log \frac{0}{0} = 0$. Note that from (2.15.92) we derive that $I(\mathbf{x}; \mathbf{y}) = D_{KL}(p(x, y), p(x)p(y))$, i.e. the mutual information is a measure of the KL divergence between a joint bivariate density and the product of its univariate terms.

From (2.15.93), we derive

$$I(\mathbf{x}; \mathbf{y}) = H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x}) = H(\mathbf{y}) + H(\mathbf{x}) - H(\mathbf{x}, \mathbf{y}) \quad (2.15.95)$$

Mutual information is null if and only if \mathbf{x} and \mathbf{y} are independent, i.e.

$$I(\mathbf{x}; \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} \perp\!\!\!\perp \mathbf{y}. \quad (2.15.96)$$

In other words, the larger the mutual information term, the stronger is the degree of dependency between two variables.

In the Gaussian case, an analytical link between correlation and mutual information exists. Let (\mathbf{x}, \mathbf{y}) a normally distributed random vector with a correlation coefficient ρ . 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 correlation coefficient (2.14.80) can be written as

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

In agreement with (2.15.96) and (2.14.82), it follows that in the Gaussian case

$$\rho(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow I(\mathbf{x}; \mathbf{y}) = 0 \quad (2.15.97)$$

2.15.6.1 Conditional mutual information

Consider three r.v.s \mathbf{x} , \mathbf{y} and \mathbf{z} . The *conditional mutual information* is defined by

$$I(\mathbf{y}; \mathbf{x}|\mathbf{z}) = H(\mathbf{y}|\mathbf{z}) - H(\mathbf{y}|\mathbf{x}, \mathbf{z}) \quad (2.15.98)$$

It can also be written as

$$I(\mathbf{y}; \mathbf{x}|\mathbf{z}) = \int \int \log \frac{p(x, y|z)}{p(x|z)p(y|z)} p(x, y, z) dx dy dz$$

While mutual information quantifies the degree of (in)dependence between two variables, conditional mutual information quantifies the degree of conditional (in)dependence (Section 2.13.5) between three variables. The conditional mutual information is null iff \mathbf{x} and \mathbf{y} are conditionally independent given \mathbf{z} , i.e.

$$I(\mathbf{x}; \mathbf{y}|\mathbf{z}) = 0 \Leftrightarrow \mathbf{x} \perp\!\!\!\perp \mathbf{y}|\mathbf{z} \quad (2.15.99)$$

Note that $I(\mathbf{x}; \mathbf{y}|\mathbf{z})$ can be null though $I(\mathbf{x}; \mathbf{y}) > 0$, like in the pizzas example in Section 2.13.5. Also a symmetric configuration is possible, e.g. $I(\mathbf{x}; \mathbf{y}) = 0$ but $I(\mathbf{x}; \mathbf{y}|\mathbf{z}) > 0$ as in the case of complementary variables which will be discussed in Section 10.8.

2.15.6.2 Joint mutual information

This section derives the information of a pair of variables $(\mathbf{x}_1, \mathbf{x}_2)$ about a third one \mathbf{y} . From (2.15.98) and (2.15.93) it follows:

$$\begin{aligned} I(\mathbf{x}; \mathbf{y}|\mathbf{z}) &= H(\mathbf{y}|\mathbf{z}) - H(\mathbf{y}|\mathbf{x}, \mathbf{z}) = H(\mathbf{y}|\mathbf{z}) + H(\mathbf{x}|\mathbf{z}) - H((\mathbf{x}, \mathbf{y})|\mathbf{z}) = \\ &= H((\mathbf{x}, \mathbf{z})) + H((\mathbf{y}, \mathbf{z})) - H(\mathbf{z}) - H((\mathbf{x}, \mathbf{y}, \mathbf{z})) \end{aligned} \quad (2.15.100)$$

From (2.15.95) it follows

$$I((\mathbf{x}_1, \mathbf{x}_2); \mathbf{y}) = H(\mathbf{x}_1, \mathbf{x}_2) + H(\mathbf{y}) - H(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y})$$

and

$$I(\mathbf{x}_1; \mathbf{y}) = H(\mathbf{x}_1) + H(\mathbf{y}) - H(\mathbf{x}_1, \mathbf{y})$$

From (2.15.100) it follows

$$\begin{aligned} I(\mathbf{x}_2; \mathbf{y}|\mathbf{x}_1) &= H(\mathbf{y}|\mathbf{x}_1) - H(\mathbf{y}|\mathbf{x}_1, \mathbf{x}_2) = \\ &= H(\mathbf{y}, \mathbf{x}_1) - H(\mathbf{x}_1) - H(\mathbf{y}, \mathbf{x}_1, \mathbf{x}_2) + H(\mathbf{x}_1, \mathbf{x}_2) \end{aligned}$$

On the basis of the results above, we derive the *chain rule* of mutual information

$$\begin{aligned} I(\mathbf{x}_1; \mathbf{y}) + I(\mathbf{x}_2; \mathbf{y}|\mathbf{x}_1) &= \\ &= H(\mathbf{x}_1) + H(\mathbf{y}) - H(\mathbf{x}_1, \mathbf{y}) + H(\mathbf{y}, \mathbf{x}_1) - H(\mathbf{x}_1) - H(\mathbf{y}, \mathbf{x}_1, \mathbf{x}_2) + H(\mathbf{x}_1, \mathbf{x}_2) = \\ &= H(\mathbf{y}) - H(\mathbf{y}, \mathbf{x}_1, \mathbf{x}_2) + H(\mathbf{x}_1, \mathbf{x}_2) = I((\mathbf{x}_1, \mathbf{x}_2); \mathbf{y}) \end{aligned} \quad (2.15.101)$$

This formula shows that the information that a pair of variables $(\mathbf{x}_1, \mathbf{x}_2)$ brings about a third variable \mathbf{y} is not simply the sum of the two mutual information terms $I(\mathbf{x}_1; \mathbf{y})$ and $I(\mathbf{x}_2; \mathbf{y})$ but is the sum of $I(\mathbf{x}_1; \mathbf{y})$ and the conditional information of \mathbf{x}_2 and \mathbf{y} given \mathbf{x}_1 . This aspect is particularly important in the feature selection context (Section 10.8) where simplistic assumptions of monotonicity and additivity do not hold.

For $n > 2$ variables $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ the chain rule formulation is

$$I(\mathbf{X}; \mathbf{y}) = I(\mathbf{X}_{-i}; \mathbf{y}|\mathbf{x}_i) + I(\mathbf{x}_i; \mathbf{y}) = I(\mathbf{x}_i; \mathbf{y}|\mathbf{X}_{-i}) + I(\mathbf{X}_{-i}; \mathbf{y}) \quad i = 1, \dots, n \quad (2.15.102)$$

where \mathbf{X}_{-i} denote the \mathbf{X} set with the i th term set aside.

2.15.7 Partial correlation coefficient

We have seen in Section 2.14.1 that correlation is a good measure of independence in the case of Gaussian distributions. The same role for conditional independence is played by partial correlation.

Definition 15.1 (First-order partial correlation). Let us consider three r.v.s \mathbf{x} , \mathbf{y} and \mathbf{z} . The first-order partial correlation is

$$\rho_{\mathbf{xy}|\mathbf{z}} = \frac{\rho_{\mathbf{xy}} - \rho_{\mathbf{xz}}\rho_{\mathbf{yz}}}{\sqrt{(1 - \rho_{\mathbf{xz}}^2)(1 - \rho_{\mathbf{yz}}^2)}} \quad (2.15.103)$$

where $\rho_{\mathbf{xy}}$ is defined in (2.14.80).

This quantity returns a measure of the correlation between \mathbf{x} and \mathbf{y} once the value of \mathbf{z} is known. It is possible to extend the partial correlation to the conditioning of two variables.

Definition 15.2 (Second-order correlation).

$$\rho_{\mathbf{x}_1\mathbf{y}|\mathbf{zx}_2} = \frac{\rho_{\mathbf{x}_1\mathbf{y}|\mathbf{z}} - \rho_{\mathbf{x}_1\mathbf{x}_2|\mathbf{z}}\rho_{\mathbf{y}\mathbf{x}_2|\mathbf{z}}}{\sqrt{(1 - \rho_{\mathbf{x}_1\mathbf{x}_2|\mathbf{z}}^2)(1 - \rho_{\mathbf{y}\mathbf{x}_2|\mathbf{z}}^2)}}$$

This can also be used to define a recurrence relationship where q th order partial correlations can be computed from $(q - 1)$ th order partial correlations.

Note that the partial correlation (2.15.103) may also be obtained by computing the correlation between \mathbf{w}_x and \mathbf{w}_y where \mathbf{w}_x and \mathbf{w}_y are the residuals of the regression (Section 7.1) of \mathbf{x} and \mathbf{y} on \mathbf{z} , respectively.

Another interesting property is the link between partial correlation and concentration matrix (Section 2.15). Let Σ and $\Omega = \Sigma^{-1}$ denote the covariance and the concentration matrix of the normal set of variables $\mathbf{Z} \cup \{\mathbf{x}, \mathbf{y}\}$. The partial correlation coefficient $\rho_{\mathbf{xy}|\mathbf{z}}$ can be obtained by matrix inversion:

$$\rho_{\mathbf{xy}|\mathbf{z}} = \frac{-\omega_{\mathbf{xy}}}{\sqrt{\omega_{\mathbf{xx}}\omega_{\mathbf{yy}}}} \quad (2.15.104)$$

where $\omega_{\mathbf{xy}}$ is the element of the concentration matrix corresponding to \mathbf{x} and \mathbf{y} .

Consider a multivariate normal vector \mathbf{X} , such that $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$, $\mathbf{X}_S \subset \mathbf{X}$ and s is the dimension of \mathbf{X}_S . Then

$$\rho_{\mathbf{x}_i\mathbf{x}_j|\mathbf{X}_S} = 0 \Leftrightarrow I(\mathbf{x}_i, \mathbf{x}_j|\mathbf{X}_S) = 0$$

Note that this is the conditional version of the relation (2.15.97).

2.16 Functions of random variables

For any deterministic function $g(\cdot) : \mathcal{Z} \rightarrow \mathbb{R}$ of \mathbf{z} , the expectation of the r.v. $g(\mathbf{z})$ is

$$E[g(\mathbf{z})] = \int g(z)p_{\mathbf{z}}(z)dz \quad (2.16.105)$$

This is also known as the *law of the unconscious statistician (LOTUS)*. Note that in general $E[g(\mathbf{z})] \neq g(E[\mathbf{z}])$, with the exception of the linear function $g(z) = az + b$ which will be discussed in the following section.

Exercise

Let \mathbf{z} be a scalar r.v. and

$$g(z) = \begin{cases} 1 & z \in [a, b] \\ 0 & \text{else} \end{cases}$$

with $a < b$. Compute $E[g(\mathbf{z})]$.

•

2.17 Monte Carlo approximation of expectation

For a generic function $g(\cdot)$, the analytical computation or numerical integration of (2.16.105) may be highly complex or impossible. A numerical alternative is represented by the adoption of a Monte Carlo method [149]. Such strategy requires a pseudo-random generator of examples according to the distribution of \mathbf{z} . In a nutshell a Monte Carlo approximation of $E_{\mathbf{z}}[g(\mathbf{z})]$ relies on the following steps:

1. generating a high number S of sample points $z_i \sim F_{\mathbf{z}}, i = 1, \dots, S$ by a using a pseudo-random generator of the distribution of \mathbf{z} ,
2. computing $g(z_i), i = 1, \dots, S$, for all the generated samples,
3. returning the approximation

$$E_{\mathbf{z}}[g(\mathbf{z})] \approx \frac{\sum_{i=1}^S g(z_i)}{S}$$

If the number S (also known as the number of *trials*) is sufficiently high, we may consider such approximation as reliable. The Monte Carlo procedure may be used in general to approximate all the parameters of the distribution (e.g. the variance) which may be written in the form of expectations. In the exercises and examples of this book, we will often use the Monte Carlo method to numerically compute expectation terms which would be otherwise analytically intractable. In particular, the Monte Carlo method will give a numerical form to important notions of machine learning (notably bias, variance and generalisation error) which would otherwise remain abstract mathematical notions.

R script

The script `Probability/mcarlo.R` implements the Monte Carlo computation of the mean and variance of $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$ as well as the computation of two more complex expectation terms: $E[\mathbf{z}^2]$ and $E[\|\mathbf{z}\|]$. Since those quantities may also be computed analytically, the script compares the analytical value and the Monte Carlo approximation. It appears that the higher the number of trials, the higher the accuracy of the approximation.

The Shiny dashboard `mcarlo.R` visualises the result of some operations on a single and two random variables by using a Monte Carlo simulation.

•

2.18 Linear combinations of r.v.

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

$$E[a\mathbf{x} + b\mathbf{y}] = aE[\mathbf{x}] + bE[\mathbf{y}], \quad a \in \mathbb{R}, b \in \mathbb{R} \quad (2.18.106)$$

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

$$\text{Var}[a\mathbf{x} + b\mathbf{y}] = a^2\text{Var}[\mathbf{x}] + b^2\text{Var}[\mathbf{y}] + 2ab(E[\mathbf{xy}] - E[\mathbf{x}]E[\mathbf{y}]) \quad (2.18.107)$$

$$= a^2\text{Var}[\mathbf{x}] + b^2\text{Var}[\mathbf{y}] + 2ab\text{Cov}[\mathbf{x}, \mathbf{y}] \quad (2.18.108)$$

where the quantity $\text{Cov}[\mathbf{x}, \mathbf{y}]$ is defined in (2.14.79).

Given n r.v. $\mathbf{z}_j, j = 1, \dots, n$

$$\text{Var}\left[\sum_{j=1}^n c_j \mathbf{z}_j\right] = \sum_{j=1}^n c_j^2 \text{Var}[\mathbf{z}_j] + 2 \sum_{i < j} c_i c_j \text{Cov}[\mathbf{z}_i, \mathbf{z}_j] \quad (2.18.109)$$

Let us consider now n random variables with the same variance σ^2 and mutual correlation ρ . Then the variance of their average is

$$\begin{aligned} \text{Var}\left[\frac{\sum_{j=1}^n \mathbf{z}_j}{n}\right] &= \frac{n\sigma^2}{n^2} + 2 \frac{1}{n^2} \frac{n(n-1)}{2} \rho \sigma^2 = \\ &= \frac{\sigma^2}{n} + \rho \sigma^2 - \frac{\rho \sigma^2}{n} = (1-\rho) \frac{\sigma^2}{n} + \rho \sigma^2 \end{aligned} \quad (2.18.110)$$

Exercise

Write a R script which implements a Monte Carlo method (Section 2.17) to prove the identity (2.18.106).

•

2.18.1 The sum of i.i.d. random variables

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

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

and the average

$$\bar{\mathbf{z}} = \frac{\mathbf{z}_1 + \mathbf{z}_2 + \dots + \mathbf{z}_N}{N} \quad (2.18.111)$$

The following relations hold

$$E[\mathbf{S}_N] = N\mu, \quad \text{Var}[\mathbf{S}_N] = N\sigma^2 \quad (2.18.112)$$

$$E[\bar{\mathbf{z}}] = \mu, \quad \text{Var}[\bar{\mathbf{z}}] = \frac{\sigma^2}{N} \quad (2.18.113)$$

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

2.19 Conclusion

The reader (especially if practitioner) might think that a chapter on probability theory is an unnecessary frill in a book on machine learning. The author has a different opinion. Probability extends the logical formalism and provides a rigorous interpretation of the human patterns of reasoning under uncertainty (e.g. abduction). Also, probability provides an effective language to formalise the task of machine learning,

i.e. using some variables (e.g. inputs) to explain, provide information (or reduce uncertainty) about other ones (e.g. targets). According to Aristotles, *philosophy begins with wonder*. From a scientific perspective, wonder originates from uncertainty, and science has the role of reducing it by explanation (e.g. by modelling). The author hopes that this chapter showed that uncertainty and information are not only philosophical concepts but quantities whose nature and relationship can be described in probabilistic terms.

What is still missing for the moment is the second major ingredient (besides uncertainty) of machine learning: data. Please be patient: the relation between uncertainty and observations will be discussed in Chapter 3, which introduces estimation as the statistical way of combining probabilistic models with real-world data.

2.20 Exercises

- Suppose you collect a dataset about spam in emails. Let the binary variables x_1 , x_2 and x_3 represent the occurrence of the words "Viagra", "Lottery" and "Won", respectively, in a email. Let the dataset of 20 emails being summarised as follows

Document	x_1 (Viagra)	x_2 (Lottery)	x_3 (Won)	y (Class)
E1	0	0	0	NOSPAM
E2	0	1	1	SPAM
E3	0	0	1	NOSPAM
E4	0	1	1	SPAM
E5	1	0	0	SPAM
E6	1	1	1	SPAM
E7	0	0	1	NOSPAM
E8	0	1	1	SPAM
E9	0	0	0	NOSPAM
E10	0	1	1	SPAM
E11	1	0	0	NOSPAM
E12	0	1	1	SPAM
E13	0	0	0	NOSPAM
E14	0	1	1	SPAM
E15	0	0	1	NOSPAM
E16	0	1	1	SPAM
E17	1	0	0	SPAM
E18	1	1	1	SPAM
E19	0	0	1	NOSPAM
E20	0	1	1	SPAM

where

- 0 stands for the case-insensitive absence of the word in the email.
- 1 stands for the case-insensitive presence of the word in the email.

Let $y = 1$ denote a spam email and $y = 0$ a no-spam email.

The student should estimate on the basis of the frequency of the data above

- $\text{Prob}\{\mathbf{x}_1 = 1, \mathbf{x}_2 = 1\}$
- $\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_2 = 1, \mathbf{x}_3 = 1\}$
- $\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{x}_2 = 1\}$
- $\text{Prob}\{\mathbf{x}_3 = 1 | \mathbf{y} = 0, \mathbf{x}_2 = 0\}$
- $\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\}$
- $\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{y} = 0\}$
- $\text{Prob}\{\mathbf{y} = 0\}$

Solution:

- $\text{Prob}\{\mathbf{x}_1 = 1, \mathbf{x}_2 = 1\} = 0.1$
 - $\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_2 = 1, \mathbf{x}_3 = 1\} = 0$
 - $\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{x}_2 = 1\} = 0.8$
 - $\text{Prob}\{\mathbf{x}_3 = 1 | \mathbf{y} = 0, \mathbf{x}_2 = 0\} = 0.5$
 - $\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\} = 1$
 - $\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{y} = 0\} = 0.875$
 - $\text{Prob}\{\mathbf{y} = 0\} = 0.4$
2. Let us consider a fraud detection problem. Suppose we collect the following transactional dataset where $\mathbf{v} = 1$ means that the transaction came from a suspicious web site and $\mathbf{f} = 1$ means that the transaction is fraudulent.

	$\mathbf{f} = 1$	$\mathbf{f} = 0$
$\mathbf{v} = 1$	500	1000
$\mathbf{v} = 0$	1	10000

Estimate the following quantities by using the frequency as estimator of probability:

- $\text{Prob}\{\mathbf{f} = 1\}$
- $\text{Prob}\{\mathbf{v} = 0\}$
- $\text{Prob}\{\mathbf{f} = 1 | \mathbf{v} = 1\}$
- $\text{Prob}\{\mathbf{v} = 1 | \mathbf{f} = 1\}$

Use the Bayes' theorem to compute $\text{Prob}\{\mathbf{v} = 1 | \mathbf{f} = 1\}$ and show that the result is identical to the one computed before.

Solution:

- $\text{Prob}\{\mathbf{f} = 1\} = 501/11501 = 0.043$
- $\text{Prob}\{\mathbf{v} = 0\} = 10001/11501 = 0.869$
- $\text{Prob}\{\mathbf{f} = 1 | \mathbf{v} = 1\} = 500/1500 = 1/3$
- $\text{Prob}\{\mathbf{v} = 1 | \mathbf{f} = 1\} = 500/501$

By Bayes theorem: $\text{Prob}\{\mathbf{v} = 1 | \mathbf{f} = 1\} = \frac{\text{Prob}\{\mathbf{f} = 1 | \mathbf{v} = 1\} \text{Prob}\{\mathbf{v} = 1\}}{\text{Prob}\{\mathbf{f} = 1\}} = \frac{1/3(500/1500)}{501/11501} = 500/501$

3. Let us consider a dataset with 4 binary variables

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	\mathbf{y}
1	1	0	1
0	0	1	0
0	1	0	0
1	1	1	1
0	0	0	0
0	1	0	0
0	1	1	0
0	0	1	0
0	0	0	0
0	1	0	0
1	1	1	1

Estimate the following quantities by using the frequency as estimator of probability

- $\text{Prob}\{\mathbf{y} = 1\}$
- $\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_1 = 0\}$

- $\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\}$

Solution:

- $\text{Prob}\{\mathbf{y} = 1\} = 3/11$
- $\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_1 = 0\} = 0$
- $\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\} = 0$

- Let us consider a task with three binary inputs and one binary target where the input distribution is

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	$P(x_1, x_2, x_3)$
0	0	0	0.2
0	0	1	0.1
0	1	0	0.1
0	1	1	0.1
1	0	0	0.1
1	0	1	0.1
1	1	0	0.1
1	1	1	0.2

and the conditional probability is

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	$P(\mathbf{y} = 1 x_1, x_2, x_3)$
0	0	0	0.8
0	0	1	0.1
0	1	0	0.5
0	1	1	0.9
1	0	0	0.05
1	0	1	0.1
1	1	0	0.05
1	1	1	0.5

Compute

- $\text{Prob}\{\mathbf{x}_1 = 1, \mathbf{x}_2 = 1\}$
- $\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_2 = 1, \mathbf{x}_3 = 0\}$
- $\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{x}_2 = 1\}$
- $\text{Prob}\{\mathbf{x}_3 = 1 | \mathbf{y} = 0, \mathbf{x}_2 = 1\}$
- $\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\}$
- $\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{y} = 0\}$

Solution:

- $\text{Prob}\{\mathbf{x}_1 = 1, \mathbf{x}_2 = 1\} = 0.1 + 0.2 = 0.3$
- By using (2.5.22) (where \mathcal{E}' stands for $\mathbf{x}_2 = 1, \mathbf{x}_3 = 0$) we obtain:

$$\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_2 = 1, \mathbf{x}_3 = 0\} = \text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 1, \mathbf{x}_3 = 0\} * \text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{x}_2 = 1, \mathbf{x}_3 = 0\} +$$

$$\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_1 = 1, \mathbf{x}_2 = 1, \mathbf{x}_3 = 0\} * \text{Prob}\{\mathbf{x}_1 = 1 | \mathbf{x}_2 = 1, \mathbf{x}_3 = 0\} = 0.5 * 0.5 +$$

$$0.95 * 0.5 = 0.725$$
- $\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{x}_2 = 1\} = (0.1 + 0.1) / (0.2 + 0.3) = 0.4$
- From the joint four variate distribution computed in the exercise below

$$\text{Prob}\{\mathbf{x}_3 = 1 | \mathbf{y} = 0, \mathbf{x}_2 = 1\} = \frac{\text{Prob}\{\mathbf{x}_3 = 1, \mathbf{y} = 0, \mathbf{x}_2 = 1\}}{\text{Prob}\{\mathbf{y} = 0, \mathbf{x}_2 = 1\}} = \frac{0.11}{0.255} = 0.4313725$$

- $\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\} = 1 - 0.8 = 0.2$

- From the joint four variate distribution computed in the exercise below

$$\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{y} = 0\} = \frac{\text{Prob}\{\mathbf{x}_1 = 0, \mathbf{y} = 0\}}{\text{Prob}\{\mathbf{y} = 0\}} = \frac{0.19}{0.57} = 0.3333$$

5. Consider the probability distribution of the previous exercise. Is \mathbf{y} conditionally independent of \mathbf{x}_1 given \mathbf{x}_2 ?

Solution:

According to Section 2.13.5, \mathbf{y} is conditionally independent of \mathbf{x}_1 given \mathbf{x}_2 if for all values x_2 :

$$\text{Prob}\{\mathbf{y} = y | \mathbf{x}_1 = x_1, \mathbf{x}_2 = x_2\} = \text{Prob}\{\mathbf{y} = y | \mathbf{x}_2 = x_2\}$$

Let us compute $\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_1 = 1, \mathbf{x}_2 = x_2\}$ and $\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = x_2\}$ for $\mathbf{x}_2 = 0$. From (2.5.22)

$$\begin{aligned} \text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 1\} &= \\ \sum_{x_3} \text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 1, \mathbf{x}_3 = x_3\} \text{Prob}\{\mathbf{x}_3 = x_3 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 1\} &= \\ = \text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 1, \mathbf{x}_3 = 0\} \text{Prob}\{\mathbf{x}_3 = 0 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 1\} + & \\ + \text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 1, \mathbf{x}_3 = 1\} \text{Prob}\{\mathbf{x}_3 = 1 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 1\} &= \\ = 0.05 * 0.1 / 0.2 + 0.1 * 0.1 / 0.2 &= 0.075 \end{aligned}$$

and

$$\begin{aligned} \text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0\} &= \\ = \sum_{x_1, x_3} \text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0, \mathbf{x}_1 = x_1, \mathbf{x}_3 = x_3\} \text{Prob}\{\mathbf{x}_1 = x_1, \mathbf{x}_3 = x_3 | \mathbf{x}_2 = 0\} &= \\ = \text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 0, \mathbf{x}_3 = 0\} \text{Prob}\{\mathbf{x}_1 = 0, \mathbf{x}_3 = 0 | \mathbf{x}_2 = 0\} + & \\ + \text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 0, \mathbf{x}_3 = 1\} \text{Prob}\{\mathbf{x}_1 = 0, \mathbf{x}_3 = 1 | \mathbf{x}_2 = 0\} + & \\ + \text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 1, \mathbf{x}_3 = 0\} \text{Prob}\{\mathbf{x}_1 = 1, \mathbf{x}_3 = 0 | \mathbf{x}_2 = 0\} + & \\ + \text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 1, \mathbf{x}_3 = 1\} \text{Prob}\{\mathbf{x}_1 = 1, \mathbf{x}_3 = 1 | \mathbf{x}_2 = 0\} &= \\ = 0.8 * 0.2 / 0.5 + 0.1 * 0.1 / 0.5 + 0.05 * 0.1 / 0.5 + 0.1 * 0.1 / 0.5 &= 0.37 \end{aligned}$$

Since those two values are different, the two variables are not conditionally independent.

An alternative would be first computing the joint distribution of the 4 variables and then deriving the conditional terms. Since

$$\text{Prob}\{y, x_1, x_2, x_3\} = \text{Prob}\{y | x_1, x_2, x_3\} \text{Prob}\{x_1, x_2, x_3\}$$

the joint distribution is :

\mathbf{y}	\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	$P(y, x_1, x_2, x_3)$
0	0	0	0	(1-0.8)*0.2=0.04
0	0	0	1	(1-0.1)*0.1=0.09
0	0	1	0	0.05
0	0	1	1	0.01
0	1	0	0	0.095
0	1	0	1	0.09
0	1	1	0	0.095
0	1	1	1	0.1
1	0	0	0	0.8*0.2=0.16
1	0	0	1	0.1*0.1=0.01
1	0	1	0	0.05
1	0	1	1	0.09
1	1	0	0	0.005
1	1	0	1	0.01
1	1	1	0	0.005
1	1	1	1	0.1

From the table above we compute the conditional terms as

$$\begin{aligned}\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0\} &= \frac{\text{Prob}\{\mathbf{y} = 1, \mathbf{x}_2 = 0\}}{\text{Prob}\{\mathbf{x}_2 = 0\}} = \\ &= \frac{0.16 + 0.01 + 0.01 + 0.005}{0.04 + 0.09 + 0.095 + 0.09 + 0.16 + 0.01 + 0.005 + 0.01} = 0.37\end{aligned}$$

and

$$\begin{aligned}\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_2 = 0, \mathbf{x}_1 = 1\} &= \frac{\text{Prob}\{\mathbf{y} = 1, \mathbf{x}_1 = 1, \mathbf{x}_2 = 0\}}{\text{Prob}\{\mathbf{x}_1 = 1, \mathbf{x}_2 = 0\}} = \\ &= \frac{0.005 + 0.01}{0.095 + 0.09 + 0.005 + 0.01} = 0.075\end{aligned}$$

Since the results are (obviously) identical to the ones obtained with the first method, the conclusion is the same, i.e. the variables are conditionally dependent.

6. Let $\mathbf{x}, \mathbf{y}, \mathbf{z}$ be three binary random variables denoting the pathological mutation of a given gene of the father, mother and child, respectively. The values 0 and 1 stand for the absence and presence of the mutation, respectively. Suppose that

- the two parents have the same probability 0.5 of having a pathological mutation in a given gene
- the variables \mathbf{x} and \mathbf{y} are independent
- the child may inherit the mutation according to this conditional probability table

$\text{Prob}\{\mathbf{z} = 1 \mathbf{x} = x, \mathbf{y} = y\}$	x	y
0	0	0
0.6	0	1
0.4	1	0
0.7	1	1

1. What is the probability that the child has no mutation if both parents are not affected?
2. What is the probability that the father had a mutated gene if the child has the mutation and the mother is not affected?
3. What is the probability that the father has a mutated gene if the child has the mutation and the mother is affected?
4. What is the probability that the child has the mutation if the father has none?
5. What is the probability that the father has a mutated gene if the child has the mutation?
6. What is the probability that the father has a mutated gene if the child has no mutation?

Solution:

Let us derive first

$$\begin{aligned}P(\mathbf{z} = 1 | \mathbf{y} = 0) &= \\ P(\mathbf{z} = 1 | \mathbf{y} = 0, \mathbf{x} = 1)P(\mathbf{x} = 1 | \mathbf{y} = 0) + P(\mathbf{z} = 1 | \mathbf{y} = 0, \mathbf{x} = 0)P(\mathbf{x} = 0 | \mathbf{y} = 0) &= \\ = P(\mathbf{z} = 1 | \mathbf{y} = 0, \mathbf{x} = 1)P(\mathbf{x} = 1) + P(\mathbf{z} = 1 | \mathbf{y} = 0, \mathbf{x} = 0)P(\mathbf{x} = 0) &= \\ = 0.4 * 0.5 + 0 * 0.5 = 0.2\end{aligned}$$

$$\begin{aligned}
P(\mathbf{z} = 1|\mathbf{y} = 1) &= \\
P(\mathbf{z} = 1|\mathbf{y} = 1, \mathbf{x} = 1)P(\mathbf{x} = 1|\mathbf{y} = 1) + P(\mathbf{z} = 1|\mathbf{y} = 1, \mathbf{x} = 0)P(\mathbf{x} = 0|\mathbf{y} = 1) &= \\
= P(\mathbf{z} = 1|\mathbf{y} = 1, \mathbf{x} = 1)P(\mathbf{x} = 1) + P(\mathbf{z} = 1|\mathbf{y} = 1, \mathbf{x} = 0)P(\mathbf{x} = 0) &= \\
= 0.7 * 0.5 + 0.6 * 0.5 &= 0.65
\end{aligned}$$

$$\begin{aligned}
P(\mathbf{z} = 1|\mathbf{x} = 1) &= \\
P(\mathbf{z} = 1|\mathbf{x} = 1, \mathbf{y} = 0)P(\mathbf{y} = 0|\mathbf{x} = 1) + P(\mathbf{z} = 1|\mathbf{y} = 1, \mathbf{x} = 1)P(\mathbf{y} = 1|\mathbf{x} = 1) &= \\
= 0.4 * 0.5 + 0.7 * 0.5 &= 0.55
\end{aligned}$$

It follows

1.

$$P(\mathbf{z} = 0|\mathbf{x} = 0, \mathbf{y} = 0) = 1$$

2.

$$P(\mathbf{x} = 1|\mathbf{z} = 1, \mathbf{y} = 0) = \frac{P(\mathbf{z} = 1|\mathbf{x} = 1, \mathbf{y} = 0)P(\mathbf{x} = 1|\mathbf{y} = 0)}{P(\mathbf{z} = 1|\mathbf{y} = 0)} = \frac{0.4 * 0.5}{0.2} = 1$$

3.

$$P(\mathbf{x} = 1|\mathbf{z} = 1, \mathbf{y} = 1) = \frac{P(\mathbf{z} = 1|\mathbf{x} = 1, \mathbf{y} = 1)P(\mathbf{x} = 1|\mathbf{y} = 1)}{P(\mathbf{z} = 1|\mathbf{y} = 1)} = \frac{0.7 * 0.5}{0.65} = 0.538$$

4.

$$\begin{aligned}
P(\mathbf{z} = 1|\mathbf{x} = 0) &= P(\mathbf{z} = 1|\mathbf{x} = 0, \mathbf{y} = 1)P(\mathbf{y} = 1|\mathbf{x} = 0) + P(\mathbf{z} = 1|\mathbf{x} = 0, \mathbf{y} = 0)P(\mathbf{y} = 0|\mathbf{x} = 0) = \\
&= 0.6 * 0.5 + 0 = 0.3
\end{aligned}$$

5.

$$P(\mathbf{x} = 1|\mathbf{z} = 1) = \frac{P(\mathbf{z} = 1|\mathbf{x} = 1)P(\mathbf{x} = 1)}{P(\mathbf{z} = 1)} = \frac{0.55 * 0.5}{0.55 * 0.5 + 0.3 * 0.5} = 0.647$$

6.

$$P(\mathbf{x} = 1|\mathbf{z} = 0) = \frac{P(\mathbf{z} = 0|\mathbf{x} = 1)P(\mathbf{x} = 1)}{P(\mathbf{z} = 0)} = \frac{0.45 * 0.5}{0.45 * 0.5 + 0.7 * 0.5} = 0.3913$$

Chapter 3

Parametric estimation: the classical approach

Given the correct probabilistic model of a phenomenon, we may derive the properties of observable data by logical deduction. The theory of statistics is designed to reverse the deductive process. It takes measured data and uses them to propose 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 we know all about the probabilistic model except the value of a finite number of 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 2.2.2).

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 human sense of uncertainty or a subjective degree of belief), and that this belief can be described in the probabilistic form.

It is well known, however, that in large-sample problems, frequentist and Bayesian approaches tend to produce similar numerical results and that in small-medium settings, though the two outcomes may not coincide, their difference is usually small. For those reasons and, mainly for reasons of space, we will limit here to consider the classical approach. It is important, however, not to underestimate the important role of the Bayesian estimation philosophy, which led recently to a large amount of research in Bayesian data analysis and important applications in machine learning [80].

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, interested in agricultural issues, focused on the estimation from data.

We will introduce estimation by considering a simple univariate setting. Let \mathbf{z} be a continuous r.v. and suppose that

1. we know the analytical form of the distribution family

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

but the parameter vector $\theta \in \Theta$ is unknown,

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

In the general case, few parameters are not enough to describe a function, like the density function: in that sense, parametric densities are an obvious simplification. An example of a parametric distribution function is the Normal distribution (Section (2.12.2)), where the parameter vector is $\theta = [\mu, \sigma]$. The goal of the estimation procedure is to find a value $\hat{\theta}$ of the parameter θ so that the parameterised distribution $F_{\mathbf{z}}(z, \hat{\theta})$ closely matches the distribution of data.

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

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

Independently distributed means that the fact that we have observed a certain value z_i does not influence the probability of observing the value 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 $D_N = \{20, 31, 14, 11, 19, \dots\}$ 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 $D_N = \{10, 11, 1, 21, 2, \dots\}$ 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 Θ . The two possible outcomes are:

1. some specific value of Θ . In this case, we have the so-called *point estimation*.
2. some particular region of Θ . 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 θ 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 $D_N = \{z_1, z_2, \dots, z_N\}$. 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

$$\hat{\theta} = h(D_N) \quad (3.1.1)$$

of the sample dataset D_N , where $h(\cdot)$ stands for the *estimation algorithm*, that is the procedure which returns the estimation starting from a dataset D_N . Note that, from a machine learning perspective, it is more appropriate to consider h , rather than a conventional mathematical function, as a generic mapping taking the sample dataset as an input and returning an estimation as output¹.

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 the mapping h 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 $F_{\mathbf{z}}(\cdot)$

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

The *empirical distribution probability* \hat{F} is defined as the *discrete distribution* that assigns probability $1/N$ to each value z_i , $i = 1, \dots, N$. 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 observations in D_N which occur in A .

It can be proved that the vector of observed frequencies in \hat{F} is a *sufficient* statistic [174] for the true distribution $F(\cdot)$, i.e. all the information about $F(\cdot)$ contained in D_N is also contained in $\hat{F}(\cdot)$.

Now consider the distribution function $F_{\mathbf{z}}(z)$ of a continuous rv \mathbf{z} and a set of N observations $D_N = \{z_1, \dots, z_N\}$. Since

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

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

$$\hat{F}_{\mathbf{z}}(z) = \frac{N(z)}{N} = \frac{\#\{z_i \leq z\}}{N} \quad (3.2.2)$$

¹For instance, an awkward, yet acceptable, estimation algorithm could take the dataset, discard all the examples except the third one and return it as the estimation.

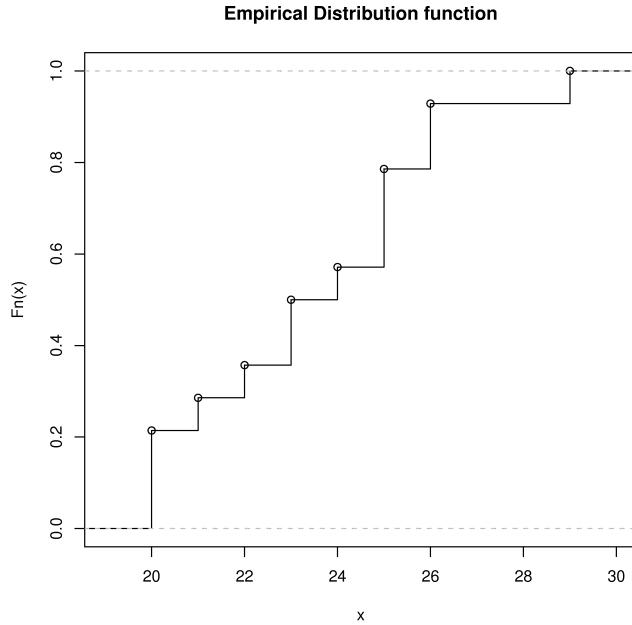


Figure 3.1: Empirical distribution.

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

Example

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

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

The empirical distribution function \hat{F}_z (which can be traced by running the script `Estimation/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 θ . A possible solution is given by the *plug-in principle*, that is a simple method of estimating parameters from observations. The *plug-in estimate* of a parameter (or target) θ is defined to be:

$$\hat{\theta} = t(\hat{F}(z)) \quad (3.3.3)$$

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

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

$$\hat{\theta} = \frac{1}{N} \sum_{i=1}^N z_i = \hat{\mu} \quad (3.3.4)$$

Note that the sample average is not a parameter (i.e. it is not a function of the probability distribution $F_{\mathbf{z}}$) but a statistic (i.e. a function of the dataset D_N).

3.3.2 Sample variance

Consider a r.v. $\mathbf{z} \sim F_{\mathbf{z}}(\cdot)$ where the mean μ and the variance σ^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 σ^2 is given by the *sample variance*

$$\hat{\sigma}^2 = \frac{1}{N-1} \sum_{i=1}^N (z_i - \hat{\mu})^2 \quad (3.3.5)$$

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

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

The expression of the plug-in estimators of other interesting probabilistic parameters are in the Appendix (E).

3.4 Sampling distribution

Given a dataset D_N of N observations sampled from \mathbf{z} , let us consider a point estimate

$$\hat{\theta} = h(D_N) \quad (3.4.6)$$

Note that since D_N is the outcome of N realisations of a r.v. \mathbf{z} , the vector D_N can be considered as the realisation of a random vector \mathbf{D}_N ².

By applying the transformation h to the random variable \mathbf{D}_N we obtain the random variable

$$\hat{\theta} = h(\mathbf{D}_N) \quad (3.4.7)$$

which is called the *point estimator* of θ . A key point is the following: *while θ is an (unknown) fixed value, the estimator $\hat{\theta}$ is a random variable*. For instance, if we aim to estimate $\theta = \mu$ (expected value of \mathbf{z}) the parameter μ is an unknown and fixed value while the average $\hat{\mu}$ is a random variable (since it is a function of a random dataset).

²This is not a mathematical detail but an essential aspect of the data-driven discovery process under uncertainty. Every model learned from data, or more in general all knowledge acquired from data, is built on random foundations and, as such, it is a random quantity and has to be assessed as such.

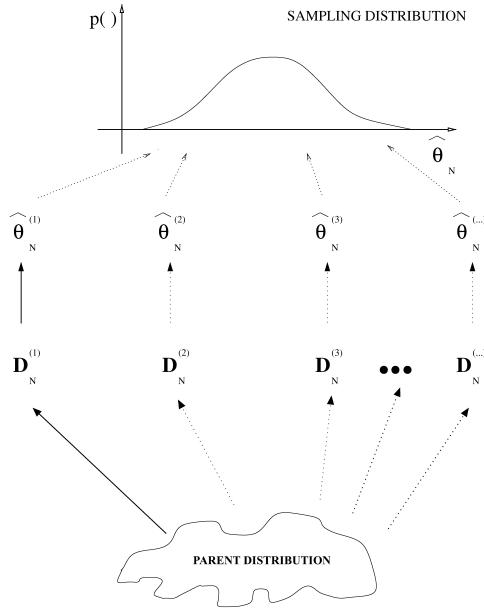


Figure 3.2: From the parametric parent distribution of $F_{\mathbf{z}}(\cdot, \theta)$ (underlying the data generation) to the sampling distribution of the estimator $\hat{\theta}_N$. Each “hypothetical” dataset has the same size N of the observed dataset.

The probability distribution of the r.v. $\hat{\theta}$ is called the *sampling distribution*, while the distribution of the r.v. \mathbf{z} (with parameter θ) is called the *parent distribution*. The sampling distribution is a key notion in the frequentist estimation where the accuracy of an estimator is defined with respect to a theoretical and infinite sequence of future samples all issued from the same parent distribution (also called the *long-run* behaviour of the estimator). An example of the process bringing from the parent to the sampling distribution is plotted in Figure 3.2 where an infinite number of “hypothetical” datasets of size N is sampled and for each of them an estimate is computed. Note that the sampling distribution, though a theoretical quantity, is of great significance in estimation since it quantifies the estimator’s accuracy in probabilistic terms, or, in simpler words, the gap between the estimation and the parameter θ .

3.4.1 Monte Carlo simulation of the sampling distribution

The Shiny dashboard `estimation.R` (Appendix H) provides an interactive visualisation of the sampling distribution of the plug-in estimators of the parameters (mean and variance) of a Normal parent distribution \mathbf{z} . As discussed in Section 2.17, Monte Carlo simulation allows a numerical approximation of mathematical expressions like the expectation. The dashboard relies on Monte Carlo simulation to approximate the distribution (and in particular the expectation and variance) of $\hat{\theta}$ with respect to the random draw of a sample D_N of size N . We invite the reader to modify the values N , μ and σ and to observe the impact on the sampling distribution. Note that the sampling distribution is obtained by a Monte Carlo simulation of the process illustrated in Figure 3.2. The simulation (Algorithm 1 and related R code in Table 3.1) consists in repeating a number (adjustable) of trials where for each trial a sample dataset of size N is generated and the plug-in estimations are computed. The dashboard shows the histograms of the estimations.

Algorithm 1 Monte Carlo simulation to generate a sampling distribution

```

1:  $S = \{\}$ 
2: for  $r = 1$  to  $R$  do
3:    $F_{\mathbf{z}} \rightarrow D_N = \{z_1, z_2, \dots, z_N\}$        $\triangleright //$  pseudo-random sample generation
4:    $\hat{\theta} = h(D_N)$                            $\triangleright //$  estimation computation
5:    $S = S \cup \{\hat{\theta}\}$ 
6: end for
7: Plot histogram of  $S$ 
8: Compute statistics of  $S$  (mean, variance)
9: Study distribution of  $S$  with respect to  $\theta$  (e.g. estimate bias)

```

```

mu<-0 # parameter
R<-10000 # number trials
N<-20 # size dataset
S<-numeric(R)
for (r in 1:R){
  D<-rnorm(N,mean=mu,sd=10)
  # pseudo-random sample generation
  S[r]<-mean(D)
  # compute estimate
}
hist(S)
# Plot histogram of S

bias=mean(S)-mu
# Estimate bias

```

Table 3.1: R version of Algorithm 1 pseudo-code to generate the sampling distribution of $\hat{\mu}$ (estimator of $E[\mathbf{z}]$).

3.5 The assessment of an estimator

Once defined an estimator $\hat{\theta}$ (e.g. in algorithmic or mathematical form), it is possible to assess its accuracy from its sampling distribution.

3.5.1 Bias and variance

The following measures rely on the sampling distribution³ to assess the estimator accuracy.

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

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

where $E_{\mathbf{D}_N}$ denotes the expectation with respect to the random draw of a sample of size N from the distribution of \mathbf{z} .

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

$$\text{Bias}[\hat{\theta}] = E_{\mathbf{D}_N}[\hat{\theta}] - \theta \quad (3.5.8)$$

³please note that we refer to the $\hat{\theta}$ distribution and not to the \mathbf{z} distribution

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

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

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

$$\sqrt{\text{Var} [\hat{\theta}]} \quad (3.5.9)$$

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

An unbiased estimator is an estimator that, on average, has the right value but averaged over what? It is important to retain that this average is over different samples of *fixed* size N , i.e. different realisations of the dataset D_N as made explicit by the notation $E_{\mathbf{D}_N} [\hat{\theta}]$, visualised in Figure 3.2 and simulated in Section (3.4.1). The unbiasedness of an estimator implies that the deviations of the estimations with respect to the parameter are not systematically up or down but that in repeated samples they average out to zero.

Note that different unbiased estimators may exist for a parameter θ . Also, a biased estimator with a known bias (i.e. not depending on θ) is equivalent to an unbiased estimator since we can easily compensate for the bias. We will see in Section 3.5.3 that for some specific estimators it is possible to derive analytically the bias. Unfortunately, in general, the bias is not measurable since this would require the knowledge of θ which is in fact the target of our estimation procedure: nevertheless, the notion of bias is an important theoretical quantity to reason about the accuracy of an estimation process.

Sometimes we are accurate (e.g. unbiased) in estimating θ though we are interested in $f(\theta)$. Given a generic transformation $f(\cdot)$, if $\hat{\theta}$ is unbiased for θ this does not imply that $f(\hat{\theta})$ is unbiased for $f(\theta)$ as well. This implies, for instance, that the standard error $\hat{\sigma}$ is not an unbiased estimator of standard deviation σ despite $\hat{\sigma}^2$ being an unbiased estimator of σ^2 .

3.5.2 Estimation and the game of darts

An intuitive manner of visualising the notion of sampling distribution of an estimator (and the related concepts of bias and variance) is to use the analogy of the darts game.

The unknown parameter θ can be seen as the darts game target and the estimator $\hat{\theta}$ 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 one is the best?

Now it's your turn to draw the shot distribution of a player with low bias and low variance and of a player with large bias and large variance.

3.5.3 Bias and variance of $\hat{\mu}$

This section shows that for a generic r.v. \mathbf{z} with a finite variance σ^2 and an i.i.d. dataset D_N , the sample average $\hat{\mu}$ is an unbiased estimator of the mean $E[\mathbf{z}]$.

Consider a random variable $\mathbf{z} \sim F_{\mathbf{z}}(\cdot)$. Let μ and σ^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 (3.3.4) we obtain

$$E_{\mathbf{D}_N} [\hat{\mu}] = E_{\mathbf{D}_N} \left[\frac{1}{N} \sum_{i=1}^N \mathbf{z}_i \right] = \frac{\sum_{i=1}^N E[\mathbf{z}_i]}{N} = \frac{N\mu}{N} = \mu \quad (3.5.10)$$

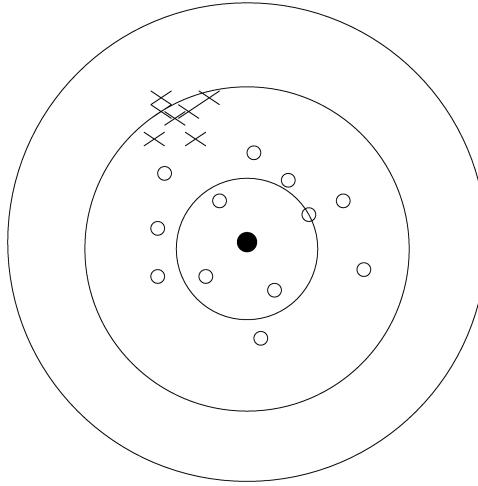


Figure 3.3: The dart analogy: the target is the unknown parameter, the round dots represent some realisations of the estimator R , while the crosses represent some realisations of the estimator C .

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 (2.18.107) we obtain that the variance of the *sample average estimator* is

$$\text{Var} [\hat{\boldsymbol{\mu}}] = \text{Var} \left[\frac{1}{N} \sum_{i=1}^N \mathbf{z}_i \right] = \frac{1}{N^2} \text{Var} \left[\sum_{i=1}^N \mathbf{z}_i \right] = \frac{1}{N^2} N \sigma^2 = \frac{\sigma^2}{N}. \quad (3.5.11)$$

Note that $\text{Var} [\hat{\boldsymbol{\mu}}]$ is also known as the *sampling variance* (i.e. the variance due to sampling), not to be confused with the sample variance quantity (3.3.5) (i.e. the variance of the sample).

Going back to the darts game analogy, the estimator $\hat{\boldsymbol{\mu}}$ behaves like the "round player" in Figure 3.3, which is centred over the target (unbiased) yet with some variability.

You can visualise the bias and variance of the sample average estimator by running the Shiny dashboard `estimation.R` presented in Section 3.4.

3.5.4 Bias of the estimator $\hat{\sigma}^2$

Let us now study the bias of the sample variance $\hat{\sigma}^2$, i.e. the estimator of the variance of \mathbf{z} .

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

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

$$= \frac{N}{N-1} E_{\mathbf{D}_N} \left[\left(\frac{1}{N} \sum_{i=1}^N \mathbf{z}_i^2 \right) - \hat{\boldsymbol{\mu}}^2 \right] \quad (3.5.14)$$

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

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

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

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

It follows that

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

This result justifies our definition (3.3.5). Once the term $N-1$ is inserted at the denominator, the sample variance estimator is not biased.

Some points are worth considering:

- The results (3.5.10), (3.5.11) and (3.5.12) are *independent* of the family of the distribution $F(\cdot)$.
- According to (3.5.11), the variance of $\hat{\mu}$ 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 observations is recommended: the larger N , the smaller is $\text{Var}[\hat{\mu}]$, so a bigger N for a given σ^2 implies a better estimate of μ .
- According to the central limit theorem (Section D.9), under quite general conditions on the distribution $F_{\mathbf{z}}$, the distribution of $\hat{\mu}$ will be approximately normal as N gets large, which we can write as

$$\hat{\mu} \sim \mathcal{N}(\mu, \sigma^2/N) \quad \text{for } N \rightarrow \infty \tag{3.5.15}$$

- The *estimated standard error*

$$\frac{\hat{\sigma}}{\sqrt{N}} \tag{3.5.16}$$

is the estimated version of the standard error⁴ (3.5.9) of the sample mean $\hat{\mu}$. It is obtained from (3.5.9) and (3.5.11) by approximating the (unknown) standard deviation σ with $\hat{\sigma}$, the squared root of the sample variance (3.3.5). This quantity is a common measure of statistical accuracy since it summarises the variability of the sample average due to random sampling. Since $\hat{\mu}$ is not biased, if the conditions (3.5.15) apply (i.e. for large N) then

$$\text{Prob} \left\{ \hat{\mu} - 2 \frac{\hat{\sigma}}{\sqrt{N}} \leq \mu \leq \hat{\mu} + 2 \frac{\hat{\sigma}}{\sqrt{N}} \right\} = 0.95$$

i.e. we expect $\hat{\mu}$ to be less than two estimated standard errors away from μ about 95% of the time (see Table 2.4). Note that this holds whatever the distribution of the r.v. \mathbf{z} .

⁴not to be confused with the standard deviation of \mathbf{z}

Script

You can visualise the bias and variance of the sample variance estimator by running the R script `Estimation/sam_dis2.R` or by running the Shiny dashboard `estimation.R` introduced in Section 3.4..

3.5.5 A tongue-twister exercise

It sounds like a tongue-twister but it is important that the reader takes some time to reason about the substantial difference between two quantities like

1. the variance of an estimator and
2. the estimator of the variance.

The first quantity is denoted by $\text{Var}[\hat{\theta}]$, is a real number and measures the accuracy of an estimator. It has been introduced in Section 3.5.

The second is denoted $\hat{\sigma}^2$, is a random quantity since it is an estimator and its properties (e.g. bias) have been discussed in Section 3.5.4.

Now, if you understand the difference between the two quantities above, you could reason about $\text{Var}[\hat{\sigma}^2]$, which is nothing more than the *variance of the estimator of the variance*. Clear, isn't it? And what about the *estimator of the variance of the estimator of the variance*?

3.5.6 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 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 criterion. This is the role of the *mean-square error* (MSE) measure.

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

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

For a generic estimator, it can be shown that

$$\text{MSE} = (E[\hat{\theta}] - \theta)^2 + \text{Var}[\hat{\theta}] = [\text{Bias}[\hat{\theta}]]^2 + \text{Var}[\hat{\theta}] \quad (3.5.17)$$

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

$$\text{MSE} = E_{\mathbf{D}_N}[(\theta - \hat{\theta})^2] = E_{\mathbf{D}_N}[(\theta - E[\hat{\theta}] + E[\hat{\theta}] - \hat{\theta})^2] = \quad (3.5.18)$$

$$= E_{\mathbf{D}_N}[(\theta - E[\hat{\theta}])^2] + E_{\mathbf{D}_N}[(E[\hat{\theta}] - \hat{\theta})^2] + E_{\mathbf{D}_N}[2(\theta - E[\hat{\theta}])(E[\hat{\theta}] - \hat{\theta})] = \quad (3.5.19)$$

$$= E_{\mathbf{D}_N}[(\theta - E[\hat{\theta}])^2] + E_{\mathbf{D}_N}[(E[\hat{\theta}] - \hat{\theta})^2] + 2(\theta - E[\hat{\theta}])(E[\hat{\theta}] - E[\hat{\theta}]) = \quad (3.5.20)$$

$$= (E[\hat{\theta}] - \theta)^2 + \text{Var}[\hat{\theta}] \quad (3.5.21)$$

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

3.5.7 Consistency

Suppose that the sample data contains N independent observations z_1, \dots, z_N of a univariate random variable. Let the estimator of θ based on N observations be denoted $\hat{\theta}_N$. As N becomes larger, we might reasonably expect that $\hat{\theta}_N$ improves as an estimator of θ (in other terms it gets closer to θ). The notion of consistency formalises this concept.

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

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

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

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

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

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

Exercise

Consider an estimator of the mean that takes into consideration only the first 10 sample points, whatever the total number $N > 10$ of observations is. Is such an estimator consistent?

•

3.5.8 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 $\hat{\theta}_1$ and $\hat{\theta}_2$. If

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

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

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

Exercise

Suppose z_1, \dots, z_N is a random sample of observations from a distribution with mean θ and variance σ^2 . Study the unbiasedness and the consistency of the three estimators of the mean μ :

$$\begin{aligned}\hat{\theta}_1 &= \hat{\mu} = \frac{\sum_{i=1}^N z_i}{N} \\ \hat{\theta}_2 &= \frac{N\hat{\theta}_1}{N+1} \\ \hat{\theta}_3 &= z_1\end{aligned}$$

•

3.6 The Hoeffding's inequality

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

Theorem 6.1. [100] Let $\mathbf{z}_1, \dots, \mathbf{z}_N$ 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

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

Corollary 6.2. If the variables $\mathbf{z}_1, \dots, \mathbf{z}_N$ 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

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

Assume that δ is a confidence parameter, that is we are $100(1-\delta)\%$ confident that the estimate $\hat{\mu}$ is within the accuracy ε 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 observations N necessary to obtain an accuracy ε and a confidence δ 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 sample points are drawn independently. Bayesian bounds are another example of statistical bounds which give tighter results under the assumption that the examples 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 $\mathbf{z}_1, \dots, \mathbf{z}_N$ be i.i.d. realisation of $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$ and let us consider the following sample statistics

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

It can be shown that the following relations hold

- $\hat{\mu} \sim \mathcal{N}(\mu, \sigma^2/N)$ and $N(\hat{\mu} - \mu)^2 \sim \sigma^2 \chi_1^2$ where the χ^2 distribution is presented in Appendix D.2.3.
- $\mathbf{z}_i - \mu \sim \mathcal{N}(0, \sigma^2)$, so $\sum_{i=1}^N (\mathbf{z}_i - \mu)^2 \sim \sigma^2 \chi_N^2$.

- $\sum_{i=1}^N (\mathbf{z}_i - \mu)^2 = \widehat{\mathbf{SS}} + N(\hat{\mu} - \mu)^2$.
- $\widehat{\mathbf{SS}} \sim \sigma^2 \chi_{N-1}^2$ or equivalently $\frac{(N-1)\hat{\sigma}^2}{\sigma^2} \sim \chi_{N-1}^2$. See R script `Estimation/sam_dis2.R`.
- $\sqrt{N}(\hat{\mu} - \mu)/\hat{\sigma} \sim \mathcal{T}_{N-1}$ where \mathcal{T} stands for the Student distribution (Section D.2.4).
- if $E[|\mathbf{z} - \mu|^4] = \mu_4$ then $\text{Var}[\hat{\sigma}^2] = \frac{1}{N} \left(\mu_4 - \frac{N-3}{N-1} \sigma^4 \right)$.

3.8 The principle of maximum likelihood

Maximum-likelihood is a well-known strategy used in statistics to design an estimator, i.e. the algorithm g in (3.4.7). Its rationale is to transform a problem of estimation into a problem of optimisation. Let us consider

1. a density distribution $p_{\mathbf{z}}(z, \theta)$ which depends on a parameter $\theta \in \Theta$,
2. a dataset $D_N = \{z_1, z_2, \dots, z_N\}$ i.i.d. drawn from this distribution.

According to (2.13.68), the joint probability density of the i.i.d. dataset is the product

$$p_{D_N}(D_N, \theta) = \prod_{i=1}^N p_{\mathbf{z}}(z_i, \theta) = L_N(\theta) \quad (3.8.22)$$

where for a fixed D_N , $L_N(\cdot)$ is a function of θ and is called the *empirical likelihood* of θ 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 θ 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 $\hat{\theta}_{\text{ml}}$ 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 $\log(\cdot)$ is a monotone function, we have

$$\hat{\theta}_{\text{ml}} = \arg \max_{\theta \in \Theta} L_N(\theta) = \arg \max_{\theta \in \Theta} \log(L_N(\theta)) = \arg \max_{\theta \in \Theta} l_N(\theta) \quad (3.8.23)$$

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 the highest probability of arising must be intuitively our best estimator of θ . 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 θ). In terms of conditional probability, $L_N(\theta)$ represents the probability of the observed dataset given θ and not the probability of θ (which is not a r.v. in the frequentist approach) given D_N .

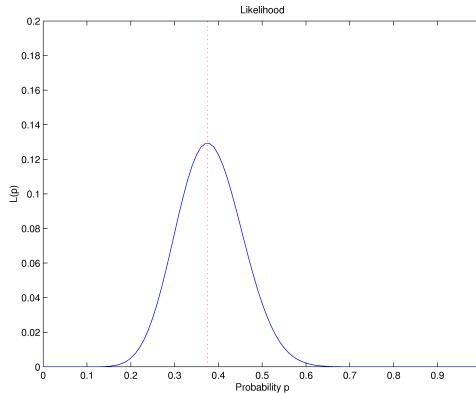


Figure 3.4: Likelihood function

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 D.1.2) 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

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

Figure 3.4 plots $L(p)$ versus $p \in [0, 1]$ (R script `Estimation/ml_bin.R`). The most likely value of p is the value where $L(\cdot)$ attains its maximum. According to Figure 3.4 this value is $\hat{p} = z/N$. The log-likelihood for this model is

$$\begin{aligned} l_N(p) &= \log L_N(p) = \log \binom{N}{z} + z \log(p) + (N - z) \log(1 - p) = \\ &= \log \binom{40}{15} + 15 \log p + 25 \log(1 - p) \end{aligned}$$

The reader can analytically find the maximum of this function by differentiating $l(p)$ with respect to 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 θ . Then $\hat{\theta}_{\text{ml}}$ is obtained simply as the solution of

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

subject to

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

to ensure that the identified stationary point is a maximum.

3.8.2 Maximum likelihood in the Gaussian case

Let D_N be a random sample from the r.v. $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$. It is possible to derive analytically the expression of the maximum likelihood estimators of the mean and variance of \mathbf{z} . According to (3.8.22), the likelihood of the N observations is

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

and the log-likelihood is

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

Note that, for a given σ , maximising the log-likelihood is equivalent to minimising the sum of squares of the difference between z_i and the mean. Taking the derivatives with respect to μ and σ^2 and setting them equal to zero, we obtain

$$\hat{\mu}_{\text{ml}} = \frac{\sum_{i=1}^N z_i}{N} = \hat{\mu} \quad (3.8.24)$$

$$\hat{\sigma}_{\text{ml}}^2 = \frac{\sum_{i=1}^N (z_i - \hat{\mu}_{\text{ml}})^2}{N} \neq \hat{\sigma}^2 \quad (3.8.25)$$

Note that the m.l. estimator (3.8.24) of the mean coincides with the sample average (3.3.4) but that the m.l. estimator (3.8.25) of the variance differs from the sample variance (3.3.5) in terms of the denominator.

In the multivariate Normal case, where \mathbf{z} is a vector with $[n, 1]$ mean μ and $[n, n]$ covariance matrix Σ , the maximum likelihood estimators are

$$\hat{\mu}_{\text{ml}} = \frac{\sum_{i=1}^N z_i}{N} \quad (3.8.26)$$

$$\hat{\Sigma}_{\text{ml}} = \frac{\sum_{i=1}^N (z_i - \hat{\mu}_{\text{ml}})(z_i - \hat{\mu}_{\text{ml}})^T}{N} \quad (3.8.27)$$

where z_i and $\hat{\mu}$ are $[n, 1]$ vectors.

Exercise

- Let $\mathbf{z} \sim \mathcal{U}(0, M)$ follow a uniform distribution and $F_{\mathbf{z}} \rightarrow D_N = \{z_1, \dots, z_N\}$. 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 $F_{\mathbf{z}}(z, \lambda) \rightarrow D_N = \{z_1, \dots, z_N\}$, find the m.l.e. of λ

•

In case of generic distributions $F_{\mathbf{z}}$ computational difficulties may arise: for example in some cases 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 θ 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 $f(x) : I \rightarrow \mathbb{R}$ but not the analytical expression of its extreme points. In this case numerical optimisation methods can be applied. The implementation of some continuous optimisation 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 instance of an optimisation problem. Let D_N be a random sample drawn from $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$. The negative log-likelihood function of the N observations 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 minimisation 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 `Estimation/emp_m1.R`.

•

3.8.3 Cramer-Rao lower bound

Assume that θ is a scalar parameter, that the first two derivatives of $L_N(\theta)$ with respect to θ exist for all θ and that certain operations of integration and differentiation may be interchanged. Let $\hat{\theta}$ be an unbiased estimator of θ and $l_N(\theta) = \log_e[L_N(\theta)]$. Suppose that the regularity condition

$$E \left[\frac{\partial l_N(\theta)}{\partial \theta} \right] = 0 \quad (3.8.28)$$

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 $\hat{\theta}$ which states that

$$\text{Var} \left[\hat{\theta} \right] \geq \frac{1}{E \left[\left(\frac{\partial l_N(\theta)}{\partial \theta} \right)^2 \right]} = -\frac{1}{NE \left[\left(\frac{\partial^2 l_N(\theta)}{\partial \theta^2} \right) \right]} = \frac{1}{I_N}$$

where the denominator term I_N is known as the Fisher information. Note that $\frac{\partial^2 l_N(\theta)}{\partial \theta^2}$ is the second derivative of $l_N(\cdot)$ and, as such, it defines the curvature of the log-likelihood function. At the maximum $\hat{\theta}$, the second derivative takes a negative value. Also, the larger its absolute value the larger is the curvature around the function peak and then the lower is the uncertainty about the m.l. estimation [137].

An estimator having a variance as low as $1/I_N$ is called a *Minimum Variance Bound (MVB) estimator*.

Example

Consider a r.v. $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$ where σ^2 is known and the unknown parameter is $\theta = \mu$. Let us consider the bound on the variance of the estimator (3.8.24). 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

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

From (3.5.11) it derives then that the m.l. estimator (3.8.24) of the mean μ is minimum variance.

•

3.8.4 Properties of m.l. estimators

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

- $\hat{\theta}_{\text{ml}}$ is *asymptotically unbiased* but usually biased in small-size samples (e.g. $\hat{\sigma}_{\text{ml}}^2$ in (3.8.25)).
- $\hat{\theta}_{\text{ml}}$ is consistent.
- If $\hat{\theta}_{\text{ml}}$ is the m.l.e. of θ and $\gamma(\cdot)$ is a *monotone* function then $\gamma(\hat{\theta}_{\text{ml}})$ is the m.l.e. of $\gamma(\theta)$.
- If $\gamma(\cdot)$ is a non/monotonic function, then even if $\hat{\theta}_{\text{ml}}$ is an unbiased estimator of θ , the m.l.e. $\gamma(\hat{\theta}_{\text{ml}})$ of $\gamma(\theta)$ is usually biased.
- the variance of $\hat{\theta}_{\text{ml}}$ is often difficult to determine. For large-size samples, we can use as approximation

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

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

$$\hat{\theta}_{\text{ml}} \sim \mathcal{N}(\theta, [I_N(\theta)]^{-1}), \quad N \rightarrow \infty$$

- if the density used in (3.8.22) is not the true $p_{\mathbf{z}}$ but an approximation $\hat{p}_{\mathbf{z}}(\theta)$ it can be shown that

$$\hat{\theta}_{\text{ml}} \rightarrow \arg \min_{\theta} KL(p_{\mathbf{z}}, \hat{p}_{\mathbf{z}}(\theta)), \quad N \rightarrow \infty$$

where KL is the Kullback-Leibler divergence (Section 2.11.3) between the true density and the approximation. This means that maximum likelihood returns the best parametric approximation of the real density $p_{\mathbf{z}}$ inside the parametric class $\hat{p}_{\mathbf{z}}(\theta)$.

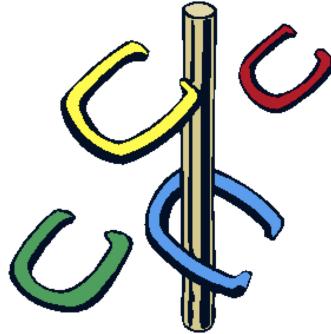


Figure 3.5: Horseshoes game as an analogy of interval estimation

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 Θ . A point estimator is a function which, given a dataset D_N generated from $F_{\mathbf{z}}(z, \theta)$, returns an estimate of θ . An *interval estimator* is a transformation which, given a dataset D_N , returns an interval estimate $[\underline{\theta}, \bar{\theta}]$ of θ . While an estimator is a random variable, an interval estimator is a random interval. Let $\underline{\theta}$ and $\bar{\theta}$ 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

$$\text{Prob} \{ \underline{\theta} \leq \theta \leq \bar{\theta} \} = 1 - \alpha \quad \alpha \in [0, 1] \quad (3.9.29)$$

then the random interval $[\underline{\theta}, \bar{\theta}]$ is called a $100(1 - \alpha)\%$ confidence interval of θ . If (3.9.29) 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 θ at least $100(1 - \alpha)\%$ of the time. Notice, however, that being θ a fixed unknown value, at each realisation D_N the interval $[\underline{\theta}, \bar{\theta}]$ either contains or not the true θ . Therefore, from a frequentist perspective, it is erroneous to think that $1 - \alpha$ is the probability of θ belonging to the interval $[\underline{\theta}, \bar{\theta}]$ computed for a given D_N . In fact, $1 - \alpha$ is not the probability of the event $\theta \in [\underline{\theta}, \bar{\theta}]$ (since θ is fixed) but the probability that the interval estimation procedure returns a (random) interval $[\underline{\theta}, \bar{\theta}]$ containing θ .

While a point estimator is characterised by bias and variance (Section 3.5), an interval estimator is characterised by its *endpoints* $\underline{\theta}$ and $\bar{\theta}$ (or its width) and by its *confidence* α . In Figure 3.3 we used an analogy between point estimation and dart game to illustrate the bias/variance notions. In the case of interval estimation, the best analogy is provided by the *horseshoe* game⁵ (Figure 3.5). A horseshoe player is like an interval estimator and her interval estimation corresponds to the tossing of a horseshoe. The horseshoe width corresponds to the interval size and the probability of encircling the stake corresponds to the confidence α .

3.9.1 Confidence interval of μ

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

⁵<https://en.wikipedia.org/wiki/Horseshoes>

$\mathcal{N}(\mu, \sigma^2/N)$ is Gaussian distributed. From (2.12.57) it follows that

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

and consequently, according to the Definition 12.8

$$\text{Prob} \left\{ -z_{\alpha/2} \leq \frac{\hat{\mu} - \mu}{\sigma/\sqrt{N}} \leq z_{\alpha/2} \right\} = 1 - \alpha \quad (3.9.30)$$

$$\text{Prob} \left\{ \hat{\mu} - z_{\alpha/2} \frac{\sigma}{\sqrt{N}} \leq \mu \leq \hat{\mu} + z_{\alpha/2} \frac{\sigma}{\sqrt{N}} \right\} = 1 - \alpha \quad (3.9.31)$$

where z_α is the upper critical point of the standard Gaussian distribution. It follows that $\underline{\theta} = \hat{\mu} - z_\alpha \sigma / \sqrt{N}$ is a lower $1 - \alpha$ confidence bound for μ while $\bar{\theta} = \hat{\mu} + z_\alpha \sigma / \sqrt{N}$ is an upper $1 - \alpha$ confidence bound for μ . By varying α 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 μ 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

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

•

R script

The R script `Estimation/confidence.R` allows the test of the formula (3.9.30) by simulation. The user sets μ , σ , N , α and a number of iterations N_{iter} .

The script generates N_{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 3.6 (R script `Estimation/confidence.R`). We can easily check that this percentage converges to $100(1 - \alpha)\%$ for $N_{\text{iter}} \rightarrow \infty$.

•

Now consider the interval of confidence of μ when the variance σ^2 is not known. Let $\hat{\mu}$ and $\hat{\sigma}^2$ be the estimators of μ and σ^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 (3.9.31) we have

$$\text{Prob} \left\{ \hat{\mu} - t_{\alpha/2} \frac{\sigma}{\sqrt{N}} \leq \mu \leq \hat{\mu} + t_{\alpha/2} \frac{\sigma}{\sqrt{N}} \right\} = 1 - \alpha \quad (3.9.32)$$

where t_α is the upper critical point of the Student distribution.

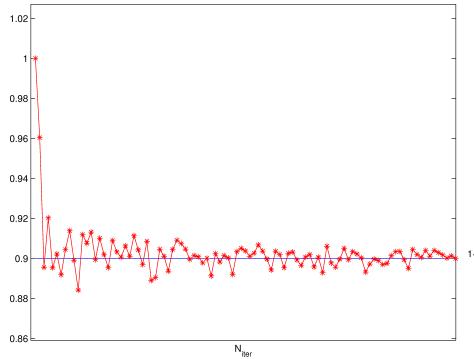


Figure 3.6: Fraction of times that the interval of confidence contains the parameter μ vs. the number of repetitions for $\alpha = 0.1$

Example

Let $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$, with σ^2 unknown and $D_N = \{10, 11, 12, 13, 14, 15\}$. We want to estimate the confidence region of μ with level $\alpha = 0.1$. We have $\hat{\mu} = 12.5$, $\hat{\sigma}^2 = 3.5$. According to (3.9.32) 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 μ is

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

•

Example

We want to estimate θ , 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 6% width with a significance of 5%. We interview N persons and estimate θ as

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

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

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

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

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

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 $\hat{\theta}_1$ and $\hat{\theta}_2$ of the same parameter θ

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

having equal and non zero variance

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

and being uncorrelated, i.e. $\text{Cov}[\hat{\theta}_1, \hat{\theta}_2] = 0$. Let $\hat{\theta}_{\text{cm}}$ be the combined estimator

$$\hat{\theta}_{\text{cm}} = \frac{\hat{\theta}_1 + \hat{\theta}_2}{2}$$

This estimator has the nice properties of being unbiased

$$E[\hat{\theta}_{\text{cm}}] = \frac{E[\hat{\theta}_1] + E[\hat{\theta}_2]}{2} = \theta \quad (3.10.33)$$

and with a smaller variance than the original estimators

$$\text{Var}[\hat{\theta}_{\text{cm}}] = \frac{1}{4}\text{Var}[\hat{\theta}_1 + \hat{\theta}_2] = \frac{\text{Var}[\hat{\theta}_1] + \text{Var}[\hat{\theta}_2]}{4} = \frac{v}{2} \quad (3.10.34)$$

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 [164, 166]. Assume we want to estimate the unknown parameter θ by combining a set of m estimators $\{\hat{\theta}_j\}$, $j = 1, \dots, m$. Let

$$E[\hat{\theta}_j] = \mu_j \quad \text{Var}[\hat{\theta}_j] = v_j \quad \text{Bias}[\hat{\theta}_j] = b_j$$

be the expected values, the variances and the bias of the m estimators, respectively.

We are interested in estimating θ by forming a linear combination

$$\hat{\theta}_{\text{cm}} = \sum_{j=1}^m w_j \hat{\theta}_j = w^T \hat{\theta} \quad (3.10.35)$$

where $\hat{\theta} = [\hat{\theta}_1, \dots, \hat{\theta}_m]^T$ is the vector of estimators and $w = [w_1, \dots, w_m]^T$ is the weighting vector.

The mean-squared error of the combined system is

$$\begin{aligned} \text{MSE} &= E[(\hat{\theta}_{\text{cm}} - \theta)^2] = E[(w^T \hat{\theta} - E[w^T \hat{\theta}])^2] + (E[w^T \hat{\theta}] - \theta)^2 \\ &= E[(w^T (\hat{\theta} - E[\hat{\theta}]))^2] + (w^T \mu - \theta)^2 = \\ &= w^T \Omega w + (w^T \mu - \theta)^2 \end{aligned}$$

where Ω is a $[m \times m]$ covariance matrix whose ij^{th} term is

$$\Omega_{ij} = E[(\hat{\theta}_i - \mu_i)(\hat{\theta}_j - \mu_j)]$$

and $\mu = (\mu_1, \dots, \mu_m)^T$ 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

$$\sum_{j=1}^m w_j = 1, \quad w_j \geq 0, \quad j = 1, \dots, m \quad (3.10.36)$$

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 $u = (1, \dots, 1)^T$ is an m -dimensional vector of ones, $g = (g_1, \dots, g_m)^T$ and $g_j > 0, \forall j = 1, \dots, m$.

The constraint can be enforced in minimising the MSE by using the Lagrangian function

$$L = w^T \Omega w + (w^T \mu - \theta)^2 + \lambda(w^T u - 1)$$

with λ 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

$$g_j^* = \frac{1}{v_j} \quad j = 1, \dots, m \quad (3.10.37)$$

This means that the optimal term g_j^* of each estimator is inversely proportional to its own variance.

3.11 Regularisation of an estimator

The previous section showed that it is possible to reduce the variance of an estimator by using a proper combination strategy. This section shows an alternative strategy to reduce the MSE of an unbiased estimator, also known as *regularisation* or *shrinkage*. The strategy consists in modifying an unbiased estimator in such a way that the bias increase is compensated by a larger variance reduction.

Let $\hat{\theta}$ be an unbiased estimator of θ ($E_{D_N}[\hat{\theta}] = \theta$) and

$$\hat{\theta}_r = \lambda \theta_0 + (1 - \lambda) \hat{\theta}, \quad \lambda \in \mathbb{R} \quad (3.11.38)$$

its regularised version where θ_0 is a constant a-priori estimator and $0 \leq \lambda \leq 1$, also called the *regularisation parameter*, denotes the weight of θ_0 in the computation of the estimator. It appears evident that, if $\lambda = 0$ the regularised estimator coincides with $\hat{\theta}$.

Let us study now the bias and variance of $\hat{\theta}_r$ for $0 < \lambda \leq 1$. Since

$$E[\hat{\theta}_r] = \lambda\theta_0 + (1 - \lambda)E[\hat{\theta}] = \lambda\theta_0 + (1 - \lambda)\theta$$

it follows that the bias of the regularised estimator

$$\text{Bias}[\hat{\theta}_r] = E_{\mathbf{D}_N}[\hat{\theta}_r] - \theta = \lambda(\theta_0 - \theta)$$

is different from zero if $\theta_0 \neq \theta$. The variance of the regularised estimator

$$\text{Var}[\hat{\theta}_r] = (1 - \lambda)^2 \text{Var}[\hat{\theta}] < \text{Var}[\hat{\theta}]$$

is instead reduced by shrinkage. It follows that the regularised estimator has a lower MSE than the original one, i.e. $\text{MSE}[\hat{\theta}_r] \leq \text{MSE}[\hat{\theta}]$ if

$$\lambda^2(\theta_0 - \theta)^2 + (1 - \lambda)^2 \text{Var}[\hat{\theta}] \leq \text{Var}[\hat{\theta}] \Leftrightarrow \lambda \leq 2 \frac{\text{Var}[\hat{\theta}]}{(\theta_0 - \theta)^2 + \text{Var}[\hat{\theta}]}$$

So, the higher the variance $\text{Var}[\hat{\theta}]$, the higher is the recommended λ value. This derivation shows that an unbiased estimator is not necessarily the best estimator we should aim to and that, especially in case of large variance setting, a proper bias increase may be beneficial in terms of the final accuracy.

3.12 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 such assertion on the basis of the observed data. But how can we be sure that what we observed is not merely a lucky draw, unlikely to be replicated in future samples? The basic idea is to formulate some statistical hypothesis (also called null hypothesis) and look to see whether observed data provides enough evidence to reject the hypothesis. Observed values that are very unlikely given the null hypothesis will be considered as evidence against the hypothesis in a sort of stochastic version of the mathematic “proof by contradiction”. 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 θ 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.

3.12.1 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 θ_0 or ω 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 μ_0 and σ_0 known values) is simple while the hypothesis $H : \mu = \mu_0$ is composite since it leaves open the value of σ in $(0, \infty)$.

3.12.2 Types of statistical test

Suppose we have sampled a dataset $D_N = \{z_1, \dots, z_N\}$ 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. This test returns 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} . This test returns a rule for rejecting H in favour of \bar{H} .

The three tests will be discussed in the following sections.

3.12.3 Pure significance test

Consider a simple null 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

$$p = \text{Prob}\{t(\mathbf{D}_N) > t_N | H\} \quad (3.12.39)$$

i.e. the probability of observing a statistic greater than t_N if the hypothesis H were true. Note that in the expression (3.12.39), 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 occurred 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 . The smaller the p-value, the stronger the evidence against H ⁶.

⁶It is common habit in life-sceince research to consider a p-value smaller than 0.05 (0.01) a (very) strong evidence against H

Note that p depends on D_N since different D_N would yield different values of t_N and consequently different values of $p \in [0, 1]$. Moreover, it can be shown that, if the null hypothesis is true, the p-value has a Uniform $U[0, 1]$ distribution. 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.

3.12.4 Tests of significance

The test of significance proposes the following decision rule: if p is less than some stated value α , we reject H . Once a *critical level* α is chosen, and the dataset D_N is observed, the rule rejects H at level α if

$$P\{t(\mathbf{D}_N) > t_\alpha | H\} = \alpha \quad (3.12.40)$$

This is equivalent to choosing some *critical value* t_α 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- α test.

The principle is that we will accept H unless what we observed has a too small probability of happening when H is true. The upper bound of this probability is α , i.e. the significance level α is the highest p-value for which we reject H . Note that the p-value changes with the observed data (i.e. it is a random variable) while α is a level fixed by the user.

Example

Let D_N consist of N independent observations of $\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2)$, with known variance σ^2 . We want to test the hypothesis $H : \mu = \mu_0$ with μ_0 known. Consider as test statistic the quantity $t(\mathbf{D}_N) = |\hat{\mu} - \mu_0|$ where $\hat{\mu}$ is the sample average estimator. If H is true we know from Section 3.4 that $\hat{\mu} \sim \mathcal{N}(\mu_0, \sigma^2/N)$. 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_α such that

$$\begin{aligned} \text{Prob}\{t(\mathbf{D}_N) > t_\alpha | H\} &= \text{Prob}\{|\hat{\mu} - \mu_0| > t_\alpha | H\} = \\ &\text{Prob}\{(\hat{\mu} - \mu_0 > t_\alpha) \cup (\hat{\mu} - \mu_0 < -t_\alpha) | H\} = 0.1 \end{aligned}$$

For a Normal variable $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2)$, we have that

$$\text{Prob}\{|z - \mu| > 1.645\sigma\} = \text{Prob}\left\{\frac{|z - \mu|}{\sigma} > 1.645\right\} = 2 * 0.05$$

It follows that being $\hat{\mu} \sim \mathcal{N}(\mu_0, \sigma^2/N)$

$$\text{Prob}\left\{|\hat{\mu} - \mu_0| > 1.645\sigma/\sqrt{N}\right\} = 0.05 + 0.05 = 0.1$$

and consequently

$$t_\alpha = 1.645\sigma/\sqrt{N} \quad (3.12.41)$$

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%. Let $N = 6$ and $D_N = \{10, 11, 12, 13, 14, 15\}$. From 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 (3.12.41) $t_\alpha = 1.645 * 0.1 / \sqrt{6} = 0.0672$, 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.

•

3.12.5 Hypothesis testing

So far we have dealt with single hypothesis tests. Let us now consider two mutually exclusive hypothesis: H and \bar{H} . Suppose we have a dataset $\{z_1, \dots, z_N\} \sim F$ drawn from a distribution F . On the basis of this dataset, one hypothesis 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 but H is true*.

For a given critical level t_α the probability of making this error is

$$\text{Prob}\{t(\mathbf{D}_N) > t_\alpha | H\} = \alpha \quad (3.12.42)$$

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

$$\text{Prob}\{t(\mathbf{D}_N) \leq t_\alpha | \bar{H}\} \quad (3.12.43)$$

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 (3.12.43) cannot be derived from (3.12.42) (see Equation (2.7.28)).

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 cheated. Let the empirical evidence t_N be represented by the number of lines of the classwork that a student shares with at least one of his classmates.

The decision rule of the professor is the following: a student passes (i.e. the null hypothesis that she is honest is accepted) if there is not enough empirical evidence against her (e.g. if $t_N \leq t_\alpha = 2$), otherwise she fails (i.e. the alternative hypothesis is chosen). Will the professor make any error? why? and does this depend on what?

•

3.12.6 The hypothesis testing procedure

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 α of the type I error (e.g. the risk I want to run when I reject the null hypothesis).
3. Define a test statistic.
4. Determine the critical value t_α of the test statistic that leads to a rejection of H according to the Type I error defined in Step 2.
5. Among the set of tests of level α , choose the test that minimises the probability of type II error.
6. Obtain the data and determine whether the observed value of the test statistic leads to an acceptance 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.

3.12.7 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 α , that is the probability of type I error (or the probability of incorrectly rejecting H)

$$\text{Prob}\{\text{reject } H|H\} = \text{Prob}\{\mathbf{D}_N \in S_0|H\} = \alpha$$

2. Among the set of tests $\{S_0, S_1\}$ of level α , choose the test that minimises the probability of type II error

$$\text{Prob}\{\text{accept } H|\bar{H}\} = \text{Prob}\{\mathbf{D}_N \in S_1|\bar{H}\}$$

that is the probability of incorrectly accepting H . This is equivalent to maximising the *power of the test*

$$\text{Prob}\{\text{reject } H|\bar{H}\} = \text{Prob}\{\mathbf{D}_N \in S_0|\bar{H}\} = 1 - \text{Prob}\{\mathbf{D}_N \in S_1|\bar{H}\} \quad (3.12.44)$$

which is the probability of *correctly* rejecting H . Note that for a given significance level, the higher the power, the better !

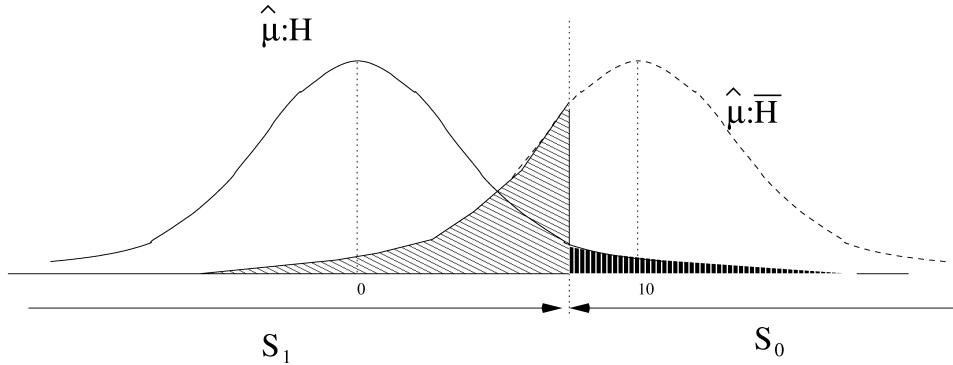


Figure 3.7: On the left: distribution of the test statistic $\hat{\mu}$ if $H : \mu_0 = 0$ is true. On the right: distribution of the test statistic $\hat{\mu}$ if $\bar{H} : \mu_1 = 10$ is true. The interval marked by S_1 denotes the set of observed $\hat{\mu}$ values for which H is accepted (non-critical 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 $\text{Prob}\{\mathbf{D}_N \in S_0 | H\}$, 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 σ 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 $\text{Prob}\{\mathbf{D}_N \in S_0 | H\} \leq \alpha$, 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?

•

3.12.8 UMP level- α test

Given a significance level α we denote by *uniformly most powerful (UMP)* test, the test

1. which satisfies

$$\text{Prob}\{\text{reject } H | H\} = \text{Prob}\{\mathbf{D}_N \in S_0 | H\} = \alpha$$

2. for which

$$\text{Prob}\{\text{reject } H | \bar{H}\} = \text{Prob}\{\mathbf{D}_N \in S_0 | \bar{H}\}$$

is maximized 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.

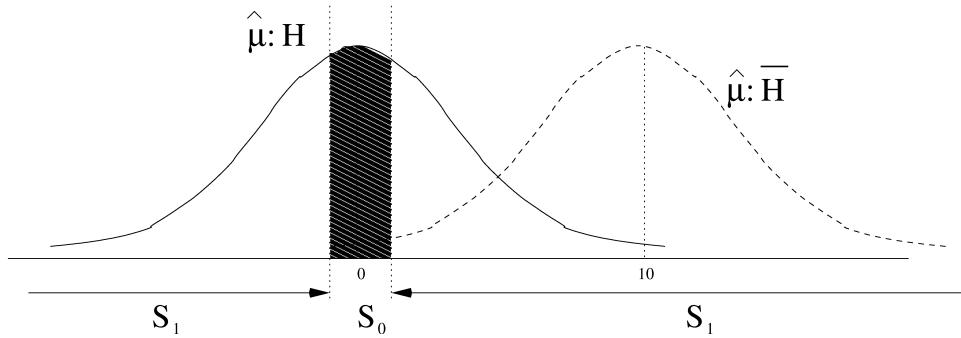


Figure 3.8: On the left: distribution of the test statistic $\hat{\mu}$ if $H : \mu_0 = 0$ is true. On the right: distribution of the test statistic $\hat{\mu}$ if $\bar{H} : \mu_1 = 10$ is true. The two intervals marked by S_1 denote the set of observed $\hat{\mu}$ values for which H is accepted (non-critical 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 $\text{Prob}\{\mathbf{D}_N \in S_0 | H\}$, 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?

3.12.9 Likelihood ratio test

Consider the simplest case $\Theta = \{\theta_0, \theta_1\}$, where $H : \theta = \theta_0$ and $\bar{H} : \theta = \theta_1$ and θ_0, θ_1 are two different values of the parameter of a r.v. \mathbf{z} . Let us denote the two likelihoods by $L(\theta_0)$ and $L(\theta_1)$, 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

$$S_0 = \{D_N : L(\theta_1) > kL(\theta_0)\} \quad S_1 = \{D_N : L(\theta_1) < kL(\theta_0)\}$$

with $\int_{S_0} p(D_N, \theta_0) dD_N = \alpha$, then $\{S_0, S_1\}$ is the most powerful level- α 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. Note, however, that the optimality relies on the knowledge of the parametric distribution of \mathbf{z} and then of its likelihood. Since this holds for all the parametric tests described in the following section, they are optimal in the UMP sense.

3.13 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 3.12.1, 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 hypothesis. 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	σ^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		$\mu_1 = \mu_2$	$\mu_1 \neq \mu_2$
χ^2 -test	single	μ	$\sigma^2 = \sigma_0^2$	$\sigma^2 \neq \sigma_0^2$
χ^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 α of having a Type I error.
3. Choose a test statistic $t(D_N)$.
4. Define the critical value t_α 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 and Appendix D.4.

3.13.1 z-test (single and one-sided)

Consider a random sample $D_N \leftarrow \mathbf{x} \sim \mathcal{N}(\mu, \sigma^2)$ with μ unknown and σ^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)

$$H : \mu = \mu_0; \quad \bar{H} : \mu > \mu_0$$

STEP 2: fix the value α of the type I error.

STEP 3: If H is true then the distribution of $\hat{\mu}$ is $\mathcal{N}(\mu_0, \sigma^2/N)$. This means that the test statistic $t(D_N)$ is

$$t_N = t(D_N) = \frac{(\hat{\mu} - \mu_0)\sqrt{N}}{\sigma} \sim \mathcal{N}(0, 1)$$

STEP 4: determine the critical value t_α .

We reject the hypothesis H if $t_N > t_\alpha = z_\alpha$ where z_α is such that $\text{Prob}\{\mathcal{N}(0, 1) > z_\alpha\} = \alpha$.

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_α for a given α 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.

•

3.13.2 t-test: single sample and two-sided

Consider a random sample from $\mathcal{N}(\mu, \sigma^2)$ with σ^2 unknown. Let

$$H : \mu = \mu_0; \quad \bar{H} : \mu \neq \mu_0$$

Let

$$t(D_N) = t_N = \frac{\sqrt{N}(\hat{\mu} - \mu_0)}{\sqrt{\frac{1}{N-1} \sum_{i=1}^N (z_i - \hat{\mu})^2}} = \frac{(\hat{\mu} - \mu_0)}{\sqrt{\frac{\hat{\sigma}^2}{N}}} \quad (3.13.45)$$

a statistic computed using the data set D_N where the denominator is the estimated standard error (3.5.16).

If the hypothesis H holds, from Sections D.2.4 and 3.7 it follows that $t(D_N) \sim \mathcal{T}_{N-1}$ is a r.v. with a Student distribution with $N - 1$ degrees of freedom. The size α t-test consists in rejecting H if

$$|t_N| > k = t_{\alpha/2, N-1}$$

where $t_{\alpha/2, N-1}$ is the upper α point of a \mathcal{T} -distribution on $N - 1$ degrees of freedom, i.e.

$$\text{Prob}\{t_{N-1} > t_{\alpha/2, N-1}\} = \alpha/2, \quad \text{Prob}\{|t_{N-1}| > t_{\alpha/2, N-1}\} = \alpha.$$

where $t_{N-1} \sim \mathcal{T}_{N-1}$. 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 α can be obtained by the R command `qt(alpha/2, N-1, lower.tail=TRUE)`.

Example [67]

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 randomly sampled from $\mathcal{N}(\mu, \sigma^2)$ where σ^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.

•

3.13.3 Statistical significance and effect size

Suppose we test the null hypothesis $H : \mu = 0$ on the basis of a dataset D_N and that we obtain a large statistic (3.13.45) and consequently a very small p-value. We will then reject the null hypothesis. Does this mean that the actual μ is necessarily a large value? Conversely, would a large p-value imply that μ is small? In both cases the answer is no. Hypothesis test and p-values do not reveal anything about the real magnitude of the parameter μ , also known as the *effect size*. A large t-statistic arises when the estimated effect (at the numerator) is large but also when the estimated standard error (at the denominator) is small. At the same time, a lack of statistical significance could be the consequence of a too high sampling variance (e.g. because of small N).

Statistical significance is the probability that an observed sample mean is different from zero because of chance. If the p-value is larger than the chosen α level (e.g, 0.05), the difference from zero is assumed to be explained by sampling variability. With a sufficiently large sample, a statistical test will almost be significant, unless there is no effect whatsoever ($\mu = 0$). Now very small μ , even if significant, could be meaningless. Also the p-value, by definition, provides no information about the alternative hypothesis.

So what? It is more and more evident in the scientific community that only reporting the p-value of hypothesis testing should not be considered sufficient or adequate for understanding the results of an experiment. The typical recommendation is then to augment the p-value with an estimation of the effect size or other quantities [90]. In particular it is important to remind that the p-value (3.12.39) of a study is a measure of the probability $P(t(D_N)|H)$ while the goal of an experimenter is typically to assess $P(H|t(D_N))$ which in Bayesian terms is related to the p-value by

$$P(H|t(D_N)) = \frac{P(t(D_N)|H)P(H)}{P(t(D_N)|H)P(H) + P(t(D_N)|\bar{H})(1 - P(H))}$$

where $P(H)$ is the a priori probability of the null hypothesis and $P(D_N|\bar{H})$ is the test power (3.12.44). It appears then that p-value provides only a limited information about the data and that it is up to the scientist to judge how the p-value together with other a priori or observed evidence support the conclusions of the study or the decision making.

3.14 A posteriori assessment of a test

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) ?

Let us consider the professor example (page 106) and the hypothesis test strategy which leads to the refusal of a student when $t_N > t_\alpha = 2$. In this case the

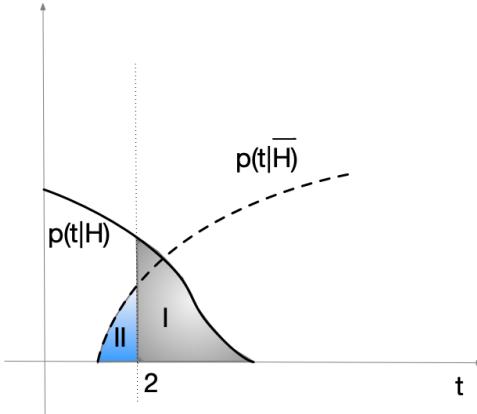


Figure 3.9: On the left: distribution of the test statistic (number of identical lines) if H is true, i.e. the student is honest. Typically honest students have very few lines in common with others though it could happen by chance that such number is more than 2. On the right: distribution of the test statistic (number of identical lines) if \bar{H} is true, i.e. the student is dishonest. Typically dishonest students have several lines in common with others though some of them are cunning enough to conceal it.

	Passed	Failed	
H : Honest 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

Table 3.2: Reality vs. decision: given N students (N_N honest and $N_{\bar{H}}$ dishonest ones) the table reports the breakdown of the N professor decisions (\hat{N}_N passes and \hat{N}_P rejections).

distributions of the t_N statistic for an honest student (or a dishonest one) has no known parametric form (Figure 3.9). Moreover, the professor has no information about such distributions and, consequently, he has no way to measure or control the Type I error rate (i.e. the grey area in Figure 3.9). Nevertheless, it is possible to estimate a posteriori the Type I and Type II error rate if we have access to the decisions of the professor and the real nature of student (honest or dishonest).

Suppose that N students took part in the exam and that N_N did not copy while N_P did. According to the professor's decision rule, \hat{N}_N were considered honest and passed the exam, while \hat{N}_P were considered dishonest and rejected. Because of the overlapping of the distributions in Figure 3.9, it happens that $F_P > 0$ honest students (the ones in the grey area) failed and $F_N > 0$ dishonest students (the ones in the blue area) 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 students who succeeded by copying but who were clever enough to avoid more than 2 identical lines.

The resulting situation can be summarised in Table 3.2 and Table 3.3 where we associated the null hypothesis H to the minus sign (non guilty or honest) and the hypothesis \bar{H} to the plus sign. In Table 3.2, F_P denotes the number of *False Positives*, i.e. the number of times that the professor considered the student as guilty (+) but in reality she 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) which is

	H accepted	H rejected
H: null hypothesis (-)	$1 - \alpha$	α
H: alternative hypothesis (+)	β	$1 - \beta$

Table 3.3: Reality vs. decision: the table reports the probability of correct and bad decisions in a hypothesis test. In particular α denotes the type I error while $1 - \beta$ the test power.

denoted by α in Table 3.3. The term F_N represents the number of *False Negatives*, i.e. the number of times that the professor considered a student to be honest (-), yet he copied (+). The ratio F_N/N is an estimation of the type II error (probability of accepting the null hypothesis when it is false) which is denoted by β in Table 3.3.

Note that the Type I and II errors are related. For instance, the professor could decide he does not want to unfairly fail even a single student by setting t_N to infinity. In this case, all honest students, like the dishonest ones, would succeed: this means we would have a null Type I error ($\alpha = 0$) at the cost of the highest Type II error ($\beta \approx N_P/N$).

3.15 Conclusion

The reader wishing to know more about machine learning could be disappointed. She has been reading more than one hundred pages and has still the sensation that she did not learn much about machine learning. All she read seems very far from intelligent agents, neural networks and fancy applications... Nevertheless, she already came across the most important notions of machine learning: conditional probability, estimation and bias/variance trade-off. Is it all about that? From an abstract perspective, yes. All the fancy algorithms that will be presented afterwards (or that the reader is used to hear about) are nothing more (often without the designer's knowledge) *estimators of conditional probability, and as such, submitted to a bias/variance tradeoff*. Such algorithms are accurate and useful only if they manage well such trade-off.

But we can go a step further and see the bias/variance tradeoff not only as a statistical concept but as a metaphor of human attitude towards models and data, beliefs and experience, ideology and observations, preconceptions and events⁷. Humans define models (not only in science but also in politics, economics, religion) to represent the regularity of nature. Now, reality often escapes or diverges from such regularity. In front of the gap between the Eden of regularity and the natural Hell of observations, humans waver between two extremes: i) negate or discredit reality and reduce all divergences to some sort of noise (measurement error) or ii) adapt, change their belief, to incorporate discordant data and measures in their model (or preconceptions).

The first attitude is exposed to bias (or dogmatism or worse conspiracy thinking): the second to variance (or instability). A biased human learner behaves as an estimator which is insensitive to data: her strength derives from the intrinsic robustness and coherence, and his weakness is due to the (in)sane attitude of disregarding data and flagrant evidence. On the other side, a highly variant human learner adapts rapidly and swiftly to data and observations, but he can be easily criticised for his excessive instability, in simple words for going where the wind blows.

When the evidence does not confirm your expectations (or what your parents,

⁷<https://tinyurl.com/y2514xyp>

teachers or media told you), what is the best attitude to take? Is there an optimal attitude? Which side are you on?

3.16 Exercises

1. Derive analytically the bias of the sample average estimator in a non i.i.d. setting: is the result different from the i.i.d. case?
2. Derive analytically the variance of the sample average estimator in an i.i.d. setting.
3. Consider a regression problem where

$$\mathbf{y} = \sin(\mathbf{x}) + \mathbf{w}$$

and \mathbf{x} is uniformly distributed on the interval $[0, 2\pi]$ and $\mathbf{w} = \mathcal{N}(1, 1)$ is a Normal variable with both mean and variance equal to 1. Let us consider a predictor $h(x)$ that is distributed like \mathbf{w} . Compute the bias and variance of the predictor in the following coordinates: $x = 0$, $x = \pi$, $x = \pi/2$.

Solution:

- $x = 0$ Bias=0, Var=1
- $x = \pi$ Bias=0, Var= 1
- $x = \pi/2$ Bias=1, Var=1

4. Let us consider a dataset $D_N = \{z_1, \dots, z_{20}\}$ of 20 observations generated according to an uniform distribution over the interval $[-1, 1]$. Suppose I want to estimate the expected value of the distribution. Compute the bias and variance of the following estimators:

- $\hat{\theta}_1 = \frac{\sum_{i=1}^{10} z_i}{10}$
- $\hat{\theta}_2 = \hat{\mu} = \frac{\sum_{i=1}^{20} z_i}{20}$
- $\hat{\theta}_3 = -1$
- $\hat{\theta}_4 = 1$
- $\hat{\theta}_5 = z_2$

Suppose I want to estimate the variance of the distribution. Compute the bias of the following estimators:

- $\hat{\sigma}_1^2 = \frac{\sum(z_i - \hat{\mu})^2}{19}$
- $\hat{\sigma}_2^2 = \frac{\sum(z_i - \hat{\mu})^2}{20}$
- $\hat{\sigma}_3^2 = 1/3$

Solution: Note that $\theta = 0$ and $\sigma_z^2 = 1/3$

$\hat{\theta}_1 : B_1 = 0, V_1 = 0.03$. Justification: $E[\hat{\theta}_1] = \theta$ and $\text{Var}[\hat{\theta}_1] = \sigma^2/10$

$\hat{\theta}_2 : B_2 = 0, V_2 = 0.015$. Justification: $E[\hat{\theta}_2] = \theta$ and $\text{Var}[\hat{\theta}_2] = \sigma^2/20$

$\hat{\theta}_3 : B_3 = -1, V_3 = 0$. Justification: $E[\hat{\theta}_3] = -1$ and $\text{Var}[\hat{\theta}_3] = 0$ since constant

$\hat{\theta}_4 : B_4 = 1, V_4 = 0$. Justification: $E[\hat{\theta}_4] = 1$ and $\text{Var}[\hat{\theta}_4] = 0$ since constant

$\hat{\theta}_5 : B_5 = 0, V_5 = 0.33$. Justification: $E[\hat{\theta}_5] = \theta$ and $\text{Var}[\hat{\theta}_5] = \sigma^2$

$\hat{\sigma}_1^2 : B = 0$. Justification: sample variance is unbiased then $E[\hat{\sigma}_1^2] = \sigma_z^2$

$\hat{\sigma}_2^2 : B - 1/60 = -0.0166$. Justification: Note first that $\hat{\sigma}^2 = \frac{19}{20} \frac{\sum(z_i - \hat{\mu})^2}{19}$. Then

$$E[\hat{\sigma}_2^2] = \frac{19}{20} E\left[\frac{\sum(z_i - \hat{\mu})^2}{19}\right] = \frac{19}{20} \sigma_z^2$$

then

$$E[\hat{\sigma}_2^2] - \sigma_z^2 = \frac{19}{20} \sigma_z^2 - \sigma_z^2 = -\sigma_z^2/20$$

$\hat{\sigma}_3^2 : B = 0$. Justification $E[1/3] = 1/3 = \sigma_z^2$

5. Let us consider the following observations of the random variable \mathbf{z}

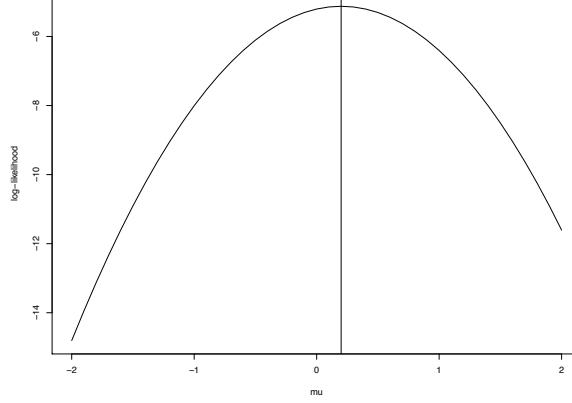
$$D_N = \{0.1, -1, 0.3, 1.4\}$$

Write the analytical form of the likelihood function of the mean μ for a Gaussian distribution with a variance $\sigma^2 = 1$. The student should:

1. Trace the log-likelihood function on the graph paper
2. Determine graphically the maximum likelihood estimator.
3. Discuss the result.

Solution: Since $N = 4$ and $\sigma = 1$

$$L(\mu) = \prod_{i=1}^N p(z_i, \mu) = \prod_{i=1}^N \frac{1}{\sqrt{2\pi}} \exp \frac{-(z_i - \mu)^2}{2}$$



Note that $\hat{\mu}_{ml}$ coincides with the sample average $\hat{\mu} = 0.2$ of D_N .

6. Suppose you want to estimate the expectation μ of the uniform r.v. $\mathbf{z} \sim \mathcal{U}[-2, 3]$ by using a dataset of size $N = 10$. By using R and its random generator to implement a Monte Carlo method, first plot the sampling distribution ($S = 1000$ trials) then estimate the bias and the variance of the following estimators:

1. $\hat{\theta} = \sum_{i=1}^N \frac{z_i}{N}$
2. $\hat{\theta} = \min_{i=1}^N z_i$
3. $\hat{\theta} = \max_{i=1}^N z_i$
4. $\hat{\theta} = z_1$
5. $\hat{\theta} = z_N$
6. $\hat{\theta} = \sum_{i=1}^N N \frac{|z_i|}{N}$
7. $\hat{\theta} = \text{median}_i z_i$
8. $\hat{\theta} = \max_{i=1}^N w_i$ where $\mathbf{w} \sim \mathcal{N}(0, 1)$.

9. $\hat{\theta} = 1$

Rank the estimators in increasing order of the absolute value of the bias and the variance.

7. The student should first create a dataset of $N = 1000$ observations according to the dependency

$$\mathbf{y} = g(\beta_0 + \beta_1 \mathbf{x}) + \mathbf{w}$$

where $\mathbf{x} \sim \mathcal{U}[-1, 1]$, $\beta_0 = 1$, $\beta_1 = -1$, $\mathbf{w} \sim \mathcal{N}(\mu = 0, \sigma^2 = 0.1)$, $g(x) = \frac{e^x}{1+e^x}$.

Then by using the same dataset she should:

- estimate by maximum likelihood the parameters β_0 and β_1 ,
- plot the contour of the likelihood function, showing in the same graph the values of the parameters and their estimations.

Hint: use a grid search to perform the maximisation.

8. The student should first create a dataset of $N = 1000$ observations according to the dependence

$$\text{Prob}\{\mathbf{y} = 1|x\} = g(\beta_0 + \beta_1 x)$$

where $\mathbf{x} \sim \mathcal{U}[0, 1]$, $\beta_0 = 1$, $\beta_1 = -1$, $g(x) = \frac{e^x}{1+e^x}$ and $y \in \{0, 1\}$.

Then by using the same dataset she should:

- estimate by maximum likelihood the parameters β_0 and β_1 ,
- plot the contour of the likelihood function, showing in the same graph the values of the parameters and their estimations.

Hint: use a grid search to perform the maximisation.

9. Let $\mathbf{z} \sim \mathcal{N}(1, 1)$, D_N a training set of N i.i.d. observations z_i and $\hat{\mu}_N$ the related sample average estimator.

1. Compute analytically

$$E_{\mathbf{z}, D_N}[(\mathbf{z} - \hat{\mu}_N)^2]$$

Hint: consider that $\mathbf{z} = \theta + \mathbf{w}$ where $\theta = E[\mathbf{z}]$ and $\mathbf{w} \sim \mathcal{N}(0, 1)$.

2. Compute analytically

$$E_{\mathbf{z}, D_N}[(\mathbf{z} - \hat{\mu}_N)]$$

3. Validate by Monte Carlo simulation the two theoretical results above.

Solution: Since $E[\hat{\mu}] = \mu$, $\text{Var}[\hat{\mu}] = \sigma_w^2/N$ and \mathbf{w} is independent of \mathbf{D}_N :

$$\begin{aligned} E_{\mathbf{z}, D_N}[(\mathbf{z} - \hat{\mu}_N)^2] &= E_{\mathbf{z}, D_N}[(\theta + \mathbf{w} - \hat{\mu}_N)^2] = \\ &= E_{\mathbf{z}, D_N}[\mathbf{w}^2 + 2\mathbf{w}(\theta - \hat{\mu}_N) + (\theta - \hat{\mu}_N)^2] = \\ &= E_{\mathbf{z}}[\mathbf{w}^2] + E_{D_N}[(\theta - \hat{\mu}_N)^2] = \sigma_w^2 + \sigma_w^2/N = 1 + 1/N \end{aligned}$$

R code to perform Monte Carlo validation :

```
rm(list=ls())
N=5
S=10000
sdw=1 ## noise variance
E=NULL
for (s in 1:S){
  DN=rnorm(N,1,sdw)
  muhat=mean(DN)
  z=rnorm(1,1,sdw)
  e=z-muhat
```

```
E=c(E,e^2)
}

cat("th=",sdw^2+sdw^2/N, "MC estimation=", mean(E),"\n")
```

10. Let us suppose that the only measurement of a Gaussian random variable $\mathbf{z} \sim \mathcal{N}(\mu, 1)$ is in the form of an interval $[-3.5, 1.5]$. Estimate μ by maximum-likelihood and show the likelihood-function $L(\mu)$. Hint: use the R function `pnorm`.
11. Let us suppose that 12 of the 31 days of August in Brussels are rainy. Estimate the probability of a rainy day by maximum likelihood by using the Binomial distribution (Section D.1.2).

Chapter 4

Nonparametric approaches to estimation and testing

4.1 Nonparametric methods

In the previous chapter, we considered estimation problems where the probability distribution is known, parameters' value (e.g. mean and/or variance) aside. Such estimation methods are called *parametric*. The meaningfulness 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 parametric 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 θ of the distribution of \mathbf{z} by using the plug-in estimate $\hat{\theta} = t(\hat{F})$ (Section 3.3). What can we say about the accuracy of the estimator $\hat{\theta}$? As shown in Section 3.5.3, 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 enjoyed considerable success in the last decades thanks to the evolution and parallelisation of computational 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.

Jackknife: 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 examples are removed. It will be presented in Section 4.3.

Bootstrap: this approach to nonparametric estimation aims to estimate the sampling distribution of an estimator by sampling (with replacement) from the original data. It will be introduced in Section 4.4.

Randomisation: 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 4.5.

Permutation: This is a resampling two-sample hypothesis-testing procedure based on repeated permutations of the dataset. It will be presented in Section 4.6.

4.2 Estimation of arbitrary statistics

Consider a set D_N of N data points sampled from a scalar r.v. \mathbf{z} . Let $E[\mathbf{z}] = \mu$ the parameter to be estimated. In Section 3.3.1 we derived the bias and the variance of the estimator $\hat{\mu}$:

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^N z_i, \quad \text{Bias}[\hat{\mu}] = 0, \quad \text{Var}[\hat{\mu}] = \frac{\sigma^2}{N}$$

Now consider 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 $\hat{\theta}$, the analytical form of the variance $\text{Var}[\hat{\theta}]$ and the bias $\text{Bias}[\hat{\theta}]$ is typically not available.*

Example

According to the plug-in principle (Section 3.3) we can design other estimators besides sampled mean and variance, like:

- Estimation of skewness (2.11.42) of \mathbf{z} : see Equation (E.0.2).
- Estimation of correlation (2.14.79) between \mathbf{x} and \mathbf{y} : : see Equation (E.0.3).

What about the accuracy (e.g. bias, variance) of such estimators?

•

Example

Let us consider an example of estimation taken from an experimental medical study [67]. 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 in 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:

$$\theta = \frac{|E(\text{new}) - E(\text{old})|}{E(\text{old}) - E(\text{placebo})} \leq 0.2 \quad (4.2.1)$$

Let us consider the following plug-in estimator (Section 3.3) of (4.2.1)

$$\hat{\theta} = \frac{|\hat{\mu}_{\text{new}} - \hat{\mu}_{\text{old}}|}{\hat{\mu}_{\text{old}} - \hat{\mu}_{\text{placebo}}}$$

Suppose we have collected the following data (details in [67])

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 Jackknife

The *jackknife* (or *leave-one-out*) resampling technique aims at providing a computational procedure to estimate the variance and the bias of a generic estimator $\hat{\theta}$. The technique was first proposed by Quenouille in 1949 and is based on removing examples from the available dataset and recalculating the estimator. It is a general-purpose tool that is easy to implement and able to solve a number of estimation problems.

4.3.1 Jackknife estimation

In order to show the theoretical foundation of the jackknife, we first apply this technique to the estimator $\hat{\mu}$ of the mean. Let $D_N = \{z_1, \dots, z_N\}$ be the available dataset. Let us remove the i th example from D_N and let us calculate the *leave-one-out (l-o-o) mean estimate* from the $N - 1$ remaining examples

$$\hat{\mu}_{(i)} = \frac{1}{N-1} \sum_{j \neq i}^N z_j = \frac{N\hat{\mu} - z_i}{N-1}$$

Observe from above that the following relation holds

$$z_i = N\hat{\mu} - (N-1)\hat{\mu}_{(i)} \quad (4.3.2)$$

that is, we can calculate the i th example z_i , $i = 1, \dots, N$ if we know both $\hat{\mu}$ and $\hat{\mu}_{(i)}$. Suppose now we wish to estimate some parameter θ by using as an estimator some complex statistic of the N data points

$$\hat{\theta} = g(D_N) = g(z_1, z_2, \dots, z_N)$$

The jackknife procedure consists in first computing

$$\hat{\theta}_{(i)} = g(z_1, z_2, \dots, z_{i-1}, z_{i+1}, \dots, z_N), \quad i = 1, \dots, N$$

which is called the i th *jackknife replication* of $\hat{\theta}$. Then by analogy with the relation (4.3.2) holding for the mean estimator, we define the i -th *pseudo value* by

$$\eta_{(i)} = N\hat{\theta} - (N-1)\hat{\theta}_{(i)}. \quad (4.3.3)$$

These pseudo values assume the same role as the z_i in calculating the sample average (3.3.4). Hence the *jackknife estimate* of θ is given by

$$\hat{\theta}_{jk} = \frac{1}{N} \sum_{i=1}^N \eta_{(i)} = \frac{1}{N} \sum_{i=1}^N \left(N\hat{\theta} - (N-1)\hat{\theta}_{(i)} \right) = N\hat{\theta} - (N-1)\hat{\theta}_{(.)} \quad (4.3.4)$$

where

$$\hat{\theta}_{(.)} = \frac{\sum_{i=1}^N \hat{\theta}_{(i)}}{N}.$$

The rationale of the jackknife technique is to use the quantity (4.3.4) in order to estimate the bias of the estimator. Since, according to (3.5.8), $\theta = E[\hat{\theta}] - \text{Bias}[\hat{\theta}]$, the jackknife approach consists in replacing θ by $\hat{\theta}_{jk}$ and $E[\hat{\theta}]$ by $\hat{\theta}$, thus obtaining

$$\hat{\theta}_{jk} = \hat{\theta} - \text{Bias}_{jk}[\hat{\theta}].$$

It follows that the *jackknife estimate of the bias of $\hat{\theta}$* is

$$\text{Bias}_{jk}[\hat{\theta}] = \hat{\theta} - \hat{\theta}_{jk} = \hat{\theta} - N\hat{\theta} + (N-1)\hat{\theta}_{(.)} = (N-1)(\hat{\theta}_{(.)} - \hat{\theta}).$$

Note that in the particular case of a mean estimator (i.e. $\hat{\theta} = \hat{\mu}$), we see that we obtain, as expected, $\text{Bias}_{jk}[\hat{\mu}] = 0$.

A jackknife estimate of the variance of $\hat{\theta}$ can be obtained from the sample variance of the pseudo-values. We define the *jackknife estimate of the variance of $\hat{\theta}$* as

$$\text{Var}_{jk}[\hat{\theta}] = \text{Var}[\hat{\theta}_{jk}] \quad (4.3.5)$$

Under the hypothesis of i.i.d. $\eta_{(i)}$

$$\text{Var}[\hat{\theta}_{jk}] = \text{Var}\left[\frac{\sum_{i=1}^N \eta_{(i)}}{N}\right] = \frac{\text{Var}[\eta_{(i)}]}{N}$$

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

$$\text{Var}_{jk}[\hat{\theta}] = \frac{\sum_{i=1}^N \left(\eta_{(i)} - \hat{\theta}_{jk} \right)^2}{N(N-1)} = \left(\frac{N-1}{N} \sum_{i=1}^N \left(\hat{\theta}_{(i)} - \hat{\theta}_{(.)} \right)^2 \right)$$

Note that in the case of the estimator of the mean (i.e. $\hat{\theta} = \hat{\mu}$), since $\eta_{(i)} = z_i$ and $\hat{\theta}_{jk} = \hat{\mu}$, we find again the result (3.5.11)

$$\text{Var}_{jk}[\hat{\theta}] = \frac{\sum_{i=1}^N (z_i - \hat{\mu})^2}{N(N-1)} = \frac{\hat{\sigma}^2}{N} = \text{Var}[\hat{\mu}] \quad (4.3.6)$$

The major motivation for jackknife estimates is that they reduce bias. Also, it can be shown that under suitable conditions on the type of estimator $\hat{\theta}$, the quantity (4.3.6) converges in probability to $\text{Var}[\hat{\theta}]$. However, the jackknife can fail if the statistic $\hat{\theta}$ is not smooth (i.e. small changes in data cause small changes in the statistic). An example of non-smooth statistic for which the jackknife works badly is the median.

4.4 Bootstrap

The method of *bootstrap* was proposed by Efron [63] as a computer-intensive technique to estimate the accuracy of a generic estimator $\hat{\theta}$. Bootstrap relies on a data-based simulation method for statistical inference. The term *bootstrap* derives from the phrase *to pull oneself up by one's bootstrap* based on the fictional Adventures of Baron Munchausen. 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. In general terms, to pull yourself up by your bootstraps means to succeed in something very difficult without any outside help¹.

The idea of statistical bootstrap is very simple, namely that in the 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 (3.4.6) is. By resampling with replacement from D_N we can build a set of B datasets $D_{(b)}, b = 1, \dots, B$. From the empirical distribution of the statistics $g(D_{(b)})$ we can construct confidence intervals and tests for significance.

4.4.1 Bootstrap sampling

Consider a data set D_N . A *bootstrap data set* $D_{(b)}$, $b = 1, \dots, B$ 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 a duplication of individual points in a bootstrap data set. Each point has an 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 D.1.2)

$$\text{Prob}\{k\} = \frac{N!}{k!(N-k)!} \left(\frac{1}{N}\right)^k \left(\frac{N-1}{N}\right)^{N-k} \quad 0 \leq k \leq N$$

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

Given the estimator (3.4.6), for each bootstrap dataset $D_{(b)}$, $b = 1, \dots, B$, we can define a *bootstrap replication*

$$\hat{\theta}_{(b)} = g(D_{(b)}) \quad b = 1, \dots, B$$

that is the value of the statistic for the specific bootstrap sample. The bootstrap approach computes the variance of the estimator $\hat{\theta}$ through the variance of the set $\hat{\theta}_{(b)}$, $b = 1, \dots, B$, given by

$$\text{Var}_{\text{BS}}[\hat{\theta}] = \frac{\sum_{b=1}^B (\hat{\theta}_{(b)} - \hat{\theta}_{(.)})^2}{(B-1)} \quad \text{where} \quad \hat{\theta}_{(.)} = \frac{\sum_{b=1}^B \hat{\theta}_{(b)}}{B} \quad (4.4.7)$$

¹This 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 an hardwired set of core instructions

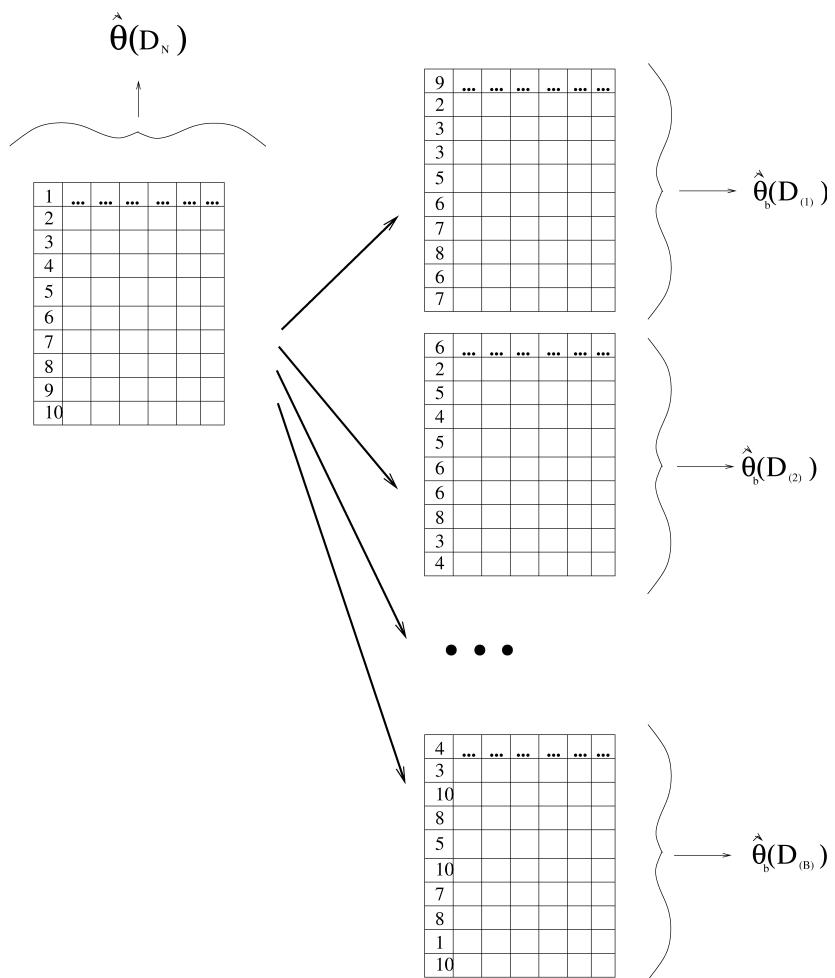


Figure 4.1: Bootstrap replications of a dataset and bootstrap statistic computation

It can be shown that if $\hat{\theta} = \hat{\mu}$, then for $B \rightarrow \infty$, the bootstrap estimate $\text{Var}_{\text{bs}}[\hat{\theta}]$ converges to the variance $\text{Var}[\hat{\mu}]$.

4.4.3 Bootstrap estimate of bias

Let $\hat{\theta}$ be a plug-in estimator (Equation (3.3.3)) based on the sample D_N and

$$\hat{\theta}_{(.)} = \frac{\sum_{b=1}^B \hat{\theta}_{(b)}}{B} \quad (4.4.8)$$

Since $\text{Bias}[\hat{\theta}] = E[\hat{\theta}] - \theta$, the *bootstrap estimate of the bias of the plug-in estimator* $\hat{\theta}$ is obtained by replacing $E[\hat{\theta}]$ with $\hat{\theta}_{(.)}$ and θ with $\hat{\theta}$:

$$\text{Bias}_{\text{bs}}[\hat{\theta}] = \hat{\theta}_{(.)} - \hat{\theta} \quad (4.4.9)$$

Then, since

$$\theta = E[\hat{\theta}] - \text{Bias}[\hat{\theta}]$$

the *bootstrap bias corrected* estimate is

$$\hat{\theta}_{\text{bs}} = \hat{\theta} - \text{Bias}_{\text{bs}}[\hat{\theta}] = \hat{\theta} - (\hat{\theta}_{(.)} - \hat{\theta}) = 2\hat{\theta} - \hat{\theta}_{(.)} \quad (4.4.10)$$

Note that if we want to estimate the bias of a generic non plug-in estimator $g(D_N)$, the $\hat{\theta}$ term in the right-hand terms of (4.4.9) should anyway refer to the plug-in estimator $t(\hat{F})$ (Equation (3.3.3)).

R script

Run the R file `NonParametric/patch.R` for the estimation of bias and variance in the case of the patch data example.

•

4.4.4 Bootstrap confidence interval

Standard bootstrap confidence limits are based on the assumption that the estimator $\hat{\theta}$ is normally distributed with mean θ and variance σ^2 . Taking the bootstrap estimate of variance, an approximate $100(1 - \alpha)\%$ confidence interval is given by

$$\hat{\theta} \pm z_{\alpha/2} \sqrt{\text{Var}_{\text{bs}}[\hat{\theta}]} = \hat{\theta} \pm z_{\alpha/2} \sqrt{\frac{\sum_{b=1}^B (\hat{\theta}_{(b)} - \hat{\theta}_{(.)})^2}{(B-1)}} \quad (4.4.11)$$

An improved interval is given by using the bootstrap correction for bias

$$2\hat{\theta} - \hat{\theta}_{(.)} \pm z_{\alpha/2} \sqrt{\frac{\sum_{b=1}^B (\hat{\theta}_{(b)} - \hat{\theta}_{(.)})^2}{(B-1)}} \quad (4.4.12)$$

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

$$[\hat{\theta}_{L,\alpha/2}, \hat{\theta}_{H,\alpha/2}] \quad (4.4.13)$$

where the two extremes are also called the *Efron's percentile confidence limits*.

4.4.5 The bootstrap principle

Given an unknown parameter θ of a distribution F_z and an estimator $\hat{\theta}$, the goal of any estimation procedure is to derive or approximate the distribution of $\hat{\theta} - \theta$. For example, the calculation of the variance of $\hat{\theta}$ requires the knowledge of F_z and the computation of $E_{D_N}[(\hat{\theta} - E[\hat{\theta}])^2]$. Now, in practical contexts, F_z is unknown, and the calculus of $E_{D_N}[(\hat{\theta} - E[\hat{\theta}])^2]$ is not possible in an analytical way. The rationale of the bootstrap approach is (i) to replace F_z by the empirical counterpart (3.2.2) and (ii) to compute $E_{D_N}[(\hat{\theta} - E[\hat{\theta}])^2]$ by a Monte Carlo simulation approach (Section 2.17) 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 $\hat{\theta}_{(b)} - \hat{\theta}$. In other terms the variability of $\hat{\theta}_{(b)}$ (based on the empirical distribution) around $\hat{\theta}$ is expected to be similar (or mimic) the variability of $\hat{\theta}$ (based on the true distribution) around θ .

The bootstrap principle relies on the two following properties (i) as N gets larger and larger, the empirical distribution $\hat{F}_z(\cdot)$ converges (almost surely) to $F_z(\cdot)$ (Glivenko-Cantelli theorem (D.11.19)) and (ii) as B gets larger, the quantity (4.4.7) converges (in probability) to the variance of the estimator $\hat{\theta}$ based on the empirical distribution (as stated in (D.10.16)). In other terms

$$\text{Var}_{\text{bs}}[\hat{\theta}] \xrightarrow{B \rightarrow \infty} E_{\widehat{D}_N}[(\hat{\theta} - E[\hat{\theta}])^2] \xrightarrow{N \rightarrow \infty} E_{D_N}[(\hat{\theta} - E[\hat{\theta}])^2] \quad (4.4.14)$$

where $E_{\widehat{D}_N}[(\hat{\theta} - E[\hat{\theta}])^2]$ stands for the plug-in estimate of the variance of $\hat{\theta}$ 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 a *simulation error*. The statistical error component is due to the difference between the underlying distribution $F_z(\cdot)$ and the empirical distribution $\hat{F}_z(\cdot)$. The magnitude of this error depends on the choice of the estimator $\hat{\theta}(D_N)$ and decreases by increasing the number N of observations.

The simulation error component is due to the use of empirical (Monte Carlo) properties of $\hat{\theta}(D_N)$ rather than exact properties. Simulation error decreases by increasing the number B of bootstrap replications.

Unlike the jackknife 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 make a good estimate of $\text{Var}[\hat{\theta}]$.
 2. Very seldom are more than $B = 200$ replications needed for estimating $\text{Var}[\hat{\theta}]$.
- Much bigger values of B are required for bootstrap confidence intervals.

Note that the use of rough statistics $\hat{\theta}$ (e.g. unsMOOTH or unstable) can make the resampling approach behave wildly. Examples 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 $N \rightarrow \infty$;
2. an estimator such that the estimate $\hat{\theta}$ is the corresponding functional of the empirical distribution.

$$\theta = t(F) \rightarrow \hat{\theta} = t(\hat{F})$$

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 discussion on the limits of bootstrap, we invite the reader to refer to [117].

4.5 Randomisation tests

Randomisation tests were introduced by R.A. Fisher in 1935. The goal of a randomisation test is to help to discover 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, randomisation tests make the null hypothesis of randomness and test this hypothesis against data. In order to test the randomness hypothesis, several random transformations of data are generated.

Suppose we are interested in some property which is related to the *order* of data. Let the original data set $D_N = \{x_1, \dots, x_N\}$ 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, $j = 1, \dots, R$ scrambled data set could be $D_N^{(j)} = \{x_{23}, x_4, x_{343}, \dots\}$
- 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 3.12.3) 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 randomised samples has a number of repetitions 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 randomisation test amounts to 0.005. You can run the randomisation test by using the R script file `NonParametric/randomiz.R`.

•

4.5.1 Randomisation and bootstrap

Both bootstrap and randomisation rely on resampling. But what are their peculiarities? A randomised 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, randomisation 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 characterise the statistical distribution of some statistics $t(D_N)$ where the order makes no difference in the statistics (e.g. mean). Randomisation 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.

4.6 Permutation test

Permutation test is used to perform a nonparametric two-sample test. Consider a random sample $\{z_1, \dots, z_M\}$ drawn from an unknown distribution $\mathbf{z} \sim F_{\mathbf{z}}(\cdot)$ and a random sample $\{y_1, \dots, y_N\}$ from an unknown distribution $\mathbf{y} \sim F_{\mathbf{y}}(\cdot)$. For example, in a bioinformatics task the two datasets could be expression measures of a gene under M normal and N pathological conditions. 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 ($i = 1, \dots, R$) 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 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 α .

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 `NonParametric/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.

•

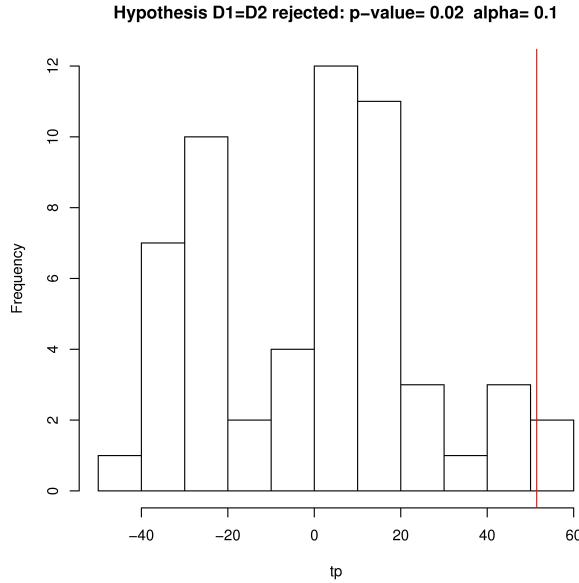


Figure 4.2: Null distribution returned by the permutation test and position (vertical red line) of the observed statistic

4.7 Considerations on nonparametric tests

Nonparametric tests are a worthy alternative to parametric approaches when no assumptions about the probability distribution may be made (e.g. in bioinformatics). 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. 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 analyse data that are inherently in ranks (e.g. taste of food), classificatory or categorical.
- Nonparametric tests are typically more intuitive and easier to implement.

Disadvantages:

- They involve high computational costs.
- The large 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.

4.8 Exercises

1. Suppose you want to estimate the skewness γ of the uniform r.v. $\mathbf{z} \sim \mathcal{U}[-2, 3]$ by using a dataset of size $N = 10$. By using R and its random generator, first plot the sampling distribution then estimate the bias and the variance of the following estimators:

$$\begin{aligned} 1. \hat{\gamma} &= \frac{\frac{1}{N} \sum_i (z_i - \bar{\mu})^3}{\hat{\sigma}^3} \\ 2. \hat{\gamma} &= \frac{\frac{1}{N} \sum_i |z_i - \bar{\mu}|^3}{\hat{\sigma}^3} \\ 3. \hat{\gamma} &= 1 \end{aligned}$$

Before each random generation set the seed to zero. Hint: the skewness of a uniform continuous variable is equal to 0.

2. Suppose you want to estimate the skewness γ of the uniform r.v. $\mathbf{z} \sim \mathcal{U}[-2, 3]$ by using a dataset of size $N = 10$. By using R and its random generator, first generate a dataset D_N with $N = 10$. By using the jackknife, plot the sampling distribution, then estimate the bias and the variance of the following estimators,

$$\begin{aligned} 1. \hat{\gamma} &= \frac{\frac{1}{N} \sum_i (z_i - \bar{\mu})^3}{\hat{\sigma}^3} \\ 2. \hat{\gamma} &= \frac{\frac{1}{N} \sum_i |z_i - \bar{\mu}|^3}{\hat{\sigma}^3} \\ 3. \hat{\gamma} &= 1 \end{aligned}$$

Compare the results with the ones of the exercise before. Before each random generation set the seed to zero.

3. Suppose you want to estimate the skewness γ of the uniform r.v. $\mathbf{z} \sim \mathcal{U}[-2, 3]$ by using a dataset of size $N = 10$. By using R and its random generator, first generate a dataset D_N with $N = 10$. By using the bootstrap method, plot the sampling distribution, then estimate the bias and the variance of the following estimators,

$$\begin{aligned} 1. \hat{\gamma} &= \frac{\frac{1}{N} \sum_i (z_i - \bar{\mu})^3}{\hat{\sigma}^3} \\ 2. \hat{\gamma} &= \frac{\frac{1}{N} \sum_i |z_i - \bar{\mu}|^3}{\hat{\sigma}^3} \\ 3. \hat{\gamma} &= 1 \end{aligned}$$

Compare the results with the ones of the two exercises before. Before each random generation set the seed to zero.

4. Let us consider a r.v. \mathbf{z} such that $E[\mathbf{z}] = \mu$ and $\text{Var}[\mathbf{z}] = \sigma^2$. Suppose we want to estimate from i.i.d. dataset D_N the parameter $\theta = \mu^2 = (E[\mathbf{z}])^2$. Let us consider three estimators:

$$\hat{\theta}_1 = \left(\frac{\sum_{i=1}^N z_i}{N} \right)^2$$

$$\hat{\theta}_2 = \frac{\sum_{i=1}^N z_i^2}{N}$$

$$\hat{\theta}_3 = \frac{(\sum_{i=1}^N z_i)^2}{N}$$

- Are they unbiased?
- Compute analytically the bias of the three estimators. Hint: use (2.11.36).
- By using R, verify the result above by Monte Carlo simulation using different values of N .
- By using R, estimate the bias of the three estimators by bootstrap.

Solution: See the file `Exercise1.pdf` in the directory `gbcodes/exercises` of the companion R package (Appendix G).

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 i.i.d. 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 conditional probability distribution $F_{\mathbf{y}}(y|\mathbf{x} = x)$. For the sake of simplicity, we will limit to consider univariate targets.
3. A collection D_N of N input/output data points $\langle x_i, y_i \rangle$, $i = 1, \dots, N$, called the *training set* and 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 encounter 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, hospitalised 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.
- Characterise the mechanical property of a steel plate on the basis of its physical and chemical composition.

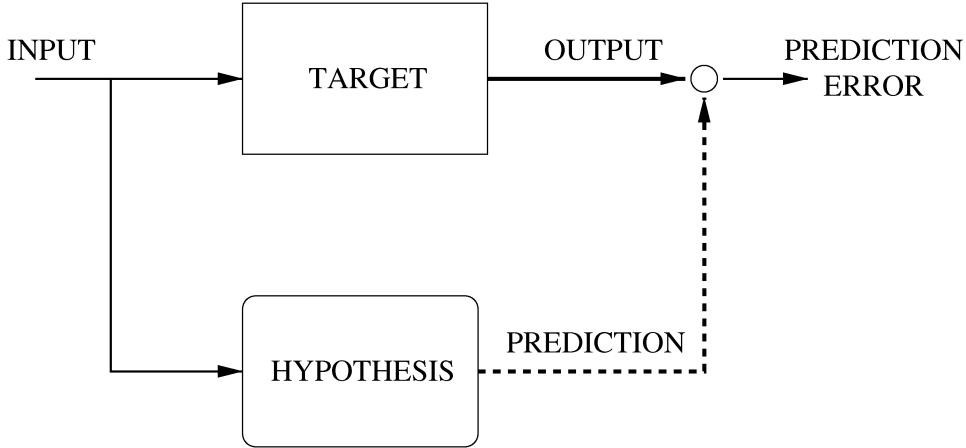


Figure 5.1: The supervised learning setting. The target operator returns an output for each input according to a fixed but unknown probabilistic law. The hypothesis predicts the value of the target output when entered with the same input.

In the case of the spam categorisation 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 i th 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 labelled 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 (3.4.7) whose goal is to estimate the parameters of the joint distribution $F_{\mathbf{x},\mathbf{y}}(y, x)$ (or sometimes of the conditional distribution $F_y(y|\mathbf{x} = x)$) on the basis of a training set D_N , i.e. a set of i.i.d. realisations of the pair \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 test value is an input to the target and the predictor (Fig. 5.1). The prediction error is also usually called *generalisation error*, since it measures the capacity of the learned hypothesis to generalise to previously unseen test samples. *A learning algorithm generalises well if it returns an accurate prediction for i.i.d. test data, i.e. input/output pairs which are independent from the training set yet are generated by the same joint distribution $F_{\mathbf{x},\mathbf{y}}(x, y)$.* We insist on the importance of the two "i" in the i.i.d. assumption: test data are supposed i) to be generated by the same distribution underlying the training set but ii) to be independent from the training set.

We will only consider hypotheses in the form $h(\cdot, \alpha)$ where $\alpha \in \Lambda^*$ is a vector of model parameters¹ or weights. Therefore, henceforth, we will denote a hypothesis $h(\cdot, \alpha)$ by the corresponding vector $\alpha \in \Lambda^*$. As we will see later, examples of hypotheses are linear models $h(x, \alpha) = x^T \alpha$ (Section 7.1) where α represents the coefficients of the model, or feed-forward neural networks (Section 8.1.1) where α is the set of values taken by the weights of the neural architecture.

¹It is important to remark that by model parameter we refer here to a tunable/trainable weight of the hypothesis function and not to the target of the estimation procedure as in Section 3.1.1

Let α_N be the hypothesis returned by the learning machine on the basis of the training set, and define G_N its generalisation error. The goal of the learning machine is then to seek the hypothesis α_N which minimises the value G_N .

In these terms, the learning problem could appear as a simple problem of optimisation which consists of searching the hypothesis α which yields the lowest generalisation 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, 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 α_N and to estimate G_N , thus inducing a strong correlation between these two quantities.

The common supervised learning practice to minimise the quantity G_N consists in

1. decomposing the set of hypothesis Λ^* into a nested sequence of hypothesis classes (or *model structures*) $\Lambda_1 \subset \Lambda_2 \subset \dots \subset \Lambda_S$ of increasing capacity (or expressiveness) s with $\Lambda^* = \cup_{s=1}^S \Lambda_s$
2. implementing a search procedure at two nested levels [119] (Fig. 5.2). The inner level, also known as *parametric identification*, considers a single class of hypotheses Λ_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 optimisation in the space of model parameters of the class Λ_s , which can be solved by (conventional) optimisation techniques. Examples of parametric identification procedures which will be presented in subsequent chapters are linear least-squares for linear models or back-propagated gradient-descent for feedforward neural networks [152]. The outer level, also called *structural identification*, ranges over nested classes of hypotheses Λ_s , ($s = 1, \dots, S$), and executes for each of them the parametric routine returning the vector α_N^s . The outcome of the parametric identification is used to assess the class Λ_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 [160] based on the jackknife strategy (Section 4.3).
3. selecting the best hypothesis in the set $\{\alpha_N^s\}$, with $s = 1, \dots, S$, according to the assessments $\{\hat{G}_N^s\}$ 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 of choosing the degree of a polynomial model or the problem of determining the best number of hidden nodes in a neural network [25].

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 introduces the statistical assessment of a learning machine. Section 5.6 discusses the notion of generalisation error and its bias/variance decomposition. Section 5.8 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 generalisation error on the basis of a finite number of data, are introduced in Section 5.9.

5.2 Estimating dependencies

This section details the main actors of the supervised learning problem:

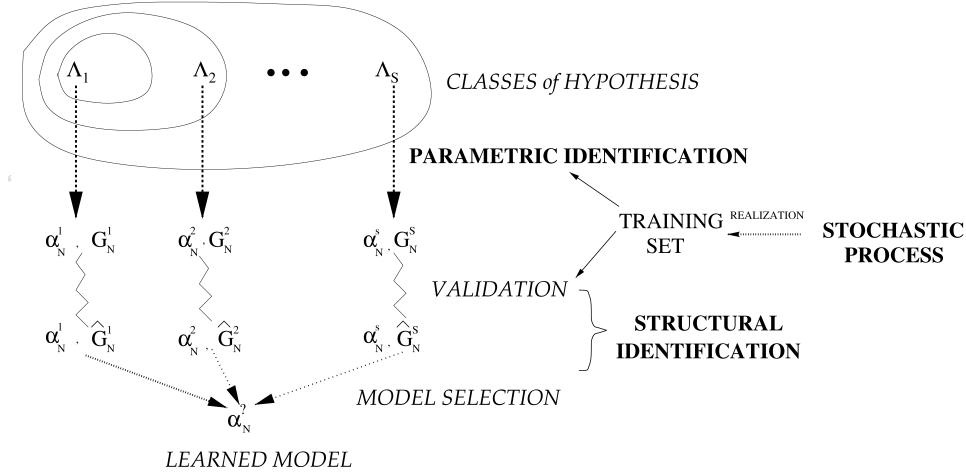


Figure 5.2: The learning problem and its decomposition in parametric and structural identification. The larger is the class of hypothesis Λ_s , the large is its expressive power in terms of functional relationships.

- 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 she does not. When the nature of inputs is completely random, we consider x as a realisation 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

$$F_{\mathbf{y}}(y|\mathbf{x} = x) \quad (5.2.1)$$

(this includes the simplest case where the target implements some deterministic function $\mathbf{y} = f(\mathbf{x})$). For the sake of simplicity, we will limit to consider here univariate targets. The conditional distribution (5.2.1) formalises the stochastic dependency between inputs and output.

- A *training set* $D_N = \{\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \dots, \langle x_N, y_N \rangle\}$ 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

$$F_{\mathbf{z}}(z) = F_{\mathbf{x}, \mathbf{y}}(\langle x, y \rangle) \quad (5.2.2)$$

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 realisation of a random variable \mathbf{D}_N .

- A *learning machine* having three components:

1. A class of *hypothesis* functions $h(\cdot, \alpha)$ with $\alpha \in \Lambda$, also called the *hypothesis space* [28]. We consider only the case where the functions $h(\cdot, \alpha) : \mathcal{X} \rightarrow \mathcal{Y}$ 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 a given hypothesis $h(\cdot, \alpha)$, the *functional risk* is the loss average over the \mathcal{XY} -domain

$$R(\alpha) = E_{\mathbf{x}, \mathbf{y}}[\mathbf{L}] = \int_{\mathcal{X}, \mathcal{Y}} L(y, h(x, \alpha)) dF_{\mathbf{x}, \mathbf{y}}(x, y) = \int_{\mathcal{X}, \mathcal{Y}} L(y, h(x, \alpha)) p(x, y) dx dy \quad (5.2.3)$$

Note that \mathbf{L} is random since \mathbf{x} and \mathbf{y} are random test points (i.i.d. drawn from the same distribution (5.2.2) of the training set) while the hypothesis $h(\cdot, \alpha)$ is given. This is the expected loss if we test the hypothesis $h(\cdot, \alpha)$ over an infinite amount of i.i.d. input/output pairs generated by (5.2.2). For the class Λ of hypothesis we define

$$\alpha_0 = \arg \min_{\alpha \in \Lambda} R(\alpha) \quad (5.2.4)$$

as the hypothesis in the class Λ 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 Λ . We define with $R(\alpha_0)$ the *functional risk of the class Λ of hypotheses*.

3. If instead of a single class of hypothesis we consider the set $\Lambda^* \subseteq \mathcal{Y}^{\mathcal{X}}$ of all measurable² single-valued mappings $h : \mathcal{X} \rightarrow \mathcal{Y}$ (contained in the set $\mathcal{Y}^{\mathcal{X}}$ of all functions from \mathcal{X} to \mathcal{Y}), we may define the quantity

$$\alpha^* = \arg \min_{\alpha \in \Lambda^*} R(\alpha) \quad (5.2.5)$$

and

$$R^* = R(\alpha^*) \quad (5.2.6)$$

as the absolute minimum rate of functional risk. Note that this quantity is ideal since it requires the complete knowledge of the distribution underlying the data. In a classification setting, the optimal model with parameters α^* is called the *Bayes classifier* and $R(\alpha^*)$ the *Bayes error* (Section 5.3.1). In a regression setting (Section 5.4) where $\mathbf{y} = f(x) + \mathbf{w}$ and the loss function is quadratic, $h(\cdot, \alpha^*) = f(\cdot)$ and $R(\alpha^*)$ amounts to the variance of \mathbf{w} .

4. 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 Minimisation) inductive principle [170] where

$$\alpha_N = \alpha(D_N) = \arg \min_{\alpha \in \Lambda} R_{\text{emp}}(\alpha) \quad (5.2.7)$$

minimise the *empirical risk* (also known as *training error* or *apparent error*)

$$R_{\text{emp}}(\alpha) = \frac{1}{N} \sum_{i=1}^N L(y_i, h(x_i, \alpha)) \quad (5.2.8)$$

constructed on the basis of the data set D_N .

²https://encyclopediaofmath.org/wiki/Measurable_function

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 Dependency and classification

Classification is one of the most common problems 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 $F_{\mathbf{x}, \mathbf{y}}(\cdot)$, such that once an input vector x is given, $\mathbf{y} \in \mathcal{Y} = \{c_1, \dots, c_K\}$ takes a value among K different classes. In the example of spam email classification, $K = 2$ and $c_1 = \text{SPAM}$, $c_2 = \text{NO.SPAM}$. We assume that the dependence is described by a conditional discrete probability distribution $\text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\}$ that satisfies

$$\sum_{k=1}^K \text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\} = 1$$

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. Pretending to have a zero-error classification in this setting is then completely unrealistic.

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 $\text{Prob}\{\mathbf{y} = \text{RAIN} | \mathbf{x} = \text{month}\}$ and $\text{Prob}\{\mathbf{y} = \text{NO.RAIN} | \mathbf{x} = \text{month}\}$ 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 $\{c_1, \dots, c_K\}$. 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 indicator function (taking only two values: zero and one)

$$L(c, \hat{c}) = \begin{cases} 0 & \text{if } c = \hat{c} \\ 1 & \text{if } c \neq \hat{c} \end{cases} \quad (5.3.9)$$

also called the *0/1 loss*. 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

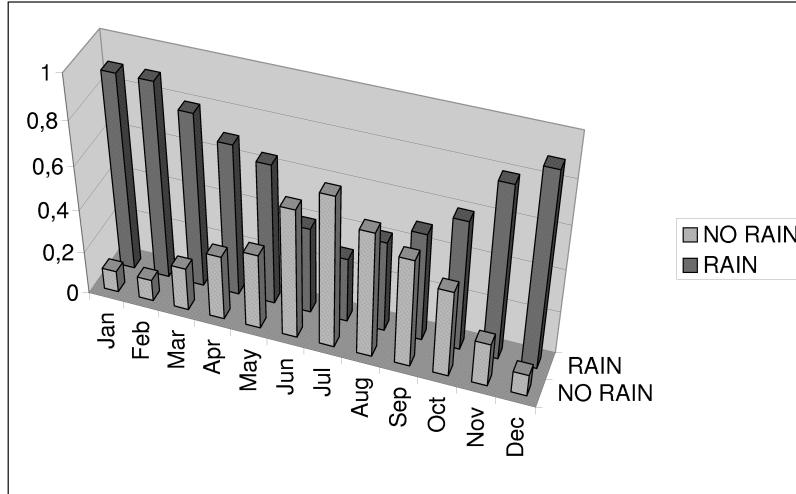


Figure 5.3: Conditional distribution $\text{Prob}\{\mathbf{y}|x\}$ where x is the current month and \mathbf{y} is the random weather state. For example, the column corresponding to $x = \text{Dec}$ and $y = \text{RAIN}$ returns the conditional probability of RAIN in December.

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 nonnegative everywhere else. In practical cases the definition of a loss matrix could be quite challenging since it should take into account and combine several criteria, some easy to quantify (e.g. financial costs) and some much less (e.g. ethical considerations)³. Note that in the case of the 0-1 loss function (Equation 5.3.9) all the elements outside the diagonal are equal to one.

The goal of the classification procedure for a given x is to find the predictor $\hat{c}(x) = h(x, \alpha)$ that minimises the quantity

$$\sum_{k=1}^K L_{(\hat{c}(x), c_k)} \text{Prob}\{\mathbf{y} = c_k | x\} \quad (5.3.10)$$

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

$$\int_{\mathcal{X}} \sum_{k=1}^K L_{(\hat{c}(x), c_k)} \text{Prob}\{\mathbf{y} = c_k | x\} dF_{\mathbf{x}} = \int_{\mathcal{X}, \mathcal{Y}} L(y, h(x, \alpha)) dF_{\mathbf{x}, \mathbf{y}} = R(\alpha) \quad (5.3.11)$$

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.

³By default, any automatic classifier (and the associated decision maker) implicitly or explicitly embeds a loss function weighting often highly heterogeneous criteria. For instance, the Tesla automatic braking systems (implicitly or explicitly) assigns a cost to false positives (e.g. a bag wrongly identified as a pedestrian) and false negatives (e.g. a pedestrian mistaken for a bag).

5.3.1 The Bayes classifier

It can be shown that the optimal classifier $h(\cdot, \alpha_0)$ where α_0 is defined as in (5.2.4) is the one that returns for all x

$$c^*(x) = h(x, \alpha_0) = \arg \min_{c_j \in \{c_1, \dots, c_K\}} \sum_{k=1}^K L_{(j,k)} \text{Prob}\{\mathbf{y} = c_k | x\} \quad (5.3.12)$$

The optimal classifier is also known as the *Bayes classifier*. In the case of a 0-1 loss function, the optimal classifier returns

$$c^*(x) = \arg \min_{c_j \in \{c_1, \dots, c_K\}} \sum_{k=1:K, k \neq j} \text{Prob}\{\mathbf{y} = c_k | x\} \quad (5.3.13)$$

$$= \arg \min_{c_j \in \{c_1, \dots, c_K\}} (1 - \text{Prob}\{\mathbf{y} = c_j | x\}) \quad (5.3.14)$$

$$= \arg \min_{c_j \in \{c_1, \dots, c_K\}} \text{Prob}\{\mathbf{y} \neq c_j | x\} = \arg \max_{c_j \in \{c_1, \dots, c_K\}} \text{Prob}\{\mathbf{y} = c_j | x\} \quad (5.3.15)$$

The Bayes decision rule selects the j , $j = 1, \dots, K$, 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		
		C1	C2	C3
PRED		0	1	5
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 (5.3.12) the Bayes classification rule for $x = 2$ returns

$$\begin{aligned} c^*(2) &= \arg \min_{k=1,2,3} \{L_{11} \text{Prob}\{\mathbf{y} = c_1 | \mathbf{x} = 2\} + L_{12} \text{Prob}\{\mathbf{y} = c_2 | \mathbf{x} = 2\} + L_{13} \text{Prob}\{\mathbf{y} = c_3 | \mathbf{x} = 2\}, \\ &\quad L_{21} \text{Prob}\{\mathbf{y} = c_1 | \mathbf{x} = 2\} + L_{22} \text{Prob}\{\mathbf{y} = c_2 | \mathbf{x} = 2\} + L_{23} \text{Prob}\{\mathbf{y} = c_3 | \mathbf{x} = 2\}, \\ &\quad L_{31} \text{Prob}\{\mathbf{y} = c_1 | \mathbf{x} = 2\} + L_{32} \text{Prob}\{\mathbf{y} = c_2 | \mathbf{x} = 2\} + L_{33} \text{Prob}\{\mathbf{y} = c_3 | \mathbf{x} = 2\}\} \\ &= \arg \min_{k=1,2,3} \{0 * 0.2 + 1 * 0.8 + 5 * 0.0, 20 * 0.2 + 0 * 0.8 + 10 * 0.0, \\ &\quad 2 * 0.2 + 1 * 0.8 + 0.0 * 0\} = \arg \min_{k=1,2,3} \{1, 4, 1.2\} = 1 \end{aligned}$$

What would have been the Bayes classification in the 0-1 case?

5.3.2 Inverse conditional distribution

An important quantity, often used in classification algorithms, is the *inverse conditional distribution*. According to the Bayes' theorem (2.6.24), if both \mathbf{x} and \mathbf{y} are discrete random variables, we have that

$$\text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\} = \frac{\text{Prob}\{\mathbf{x} = x | \mathbf{y} = c_k\} \text{Prob}\{\mathbf{y} = c_k\}}{\sum_{k=1}^K \text{Prob}\{\mathbf{x} = x | \mathbf{y} = c_k\} \text{Prob}\{\mathbf{y} = c_k\}} \quad (5.3.16)$$

and that

$$\text{Prob}\{\mathbf{x} = x | \mathbf{y} = c_k\} = \frac{\text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\} \text{Prob}\{\mathbf{x} = x\}}{\sum_x \text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\} \text{Prob}\{\mathbf{x} = x\}}. \quad (5.3.17)$$

The above relation means that by knowing the *a-posteriori* 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 *inverse conditional distribution* $\text{Prob}\{\mathbf{x} = x | \mathbf{y} = c_k\}$.

Note that the Bayes' theorem, in the case of a continuous r.v. \mathbf{x} , takes the form

$$\text{Prob}\{\mathbf{y} = c_k | \mathbf{x} = x\} = \frac{p_{\mathbf{x}}(\mathbf{x} | \mathbf{y} = c_k) \text{Prob}\{\mathbf{y} = c_k\}}{\sum_{k=1}^K p_{\mathbf{x}}(\mathbf{x} | \mathbf{y} = c_k) \text{Prob}\{\mathbf{y} = c_k\}}$$

where $p_{\mathbf{x}}(\mathbf{x} | \mathbf{y} = c_k)$ is also known as the *class conditional density*. This distribution characterises the values of the inputs x for a given class c_k . If both variables are continuous, the Bayes' theorem may be written in terms of the conditional and marginal densities.

Shiny dashboard

The Shiny dashboard `classif2.R` illustrates a binary classification task where $\mathbf{x} \in \mathbb{R}^2$ and the two classes are *green* and *red*. The green and the class conditional distributions (5.3.17) are a mixture of two Gaussians (Section 2.15.3) and a unimodal Gaussian, respectively (Figure 5.4). Figure 5.5 illustrates the associated conditional distribution (5.3.16) if the two classes have an equal a priori probability ($\text{Prob}\{\mathbf{y} = \text{red}\} = \text{Prob}\{\mathbf{y} = \text{green}\}$). Figure 5.6 shows the scattering of a set of $N = 500$ points sampled according to the class-conditional distributions in Figure 5.4.

•

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 $\mathbf{y} = \text{RAIN}$. Figure 5.7 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

$$\begin{aligned} \sum_{\text{month}} \text{Prob}\{\mathbf{x} = \text{month} | \mathbf{y} = \text{NO.RAIN}\} &= \\ &= \sum_{\text{month}} \text{Prob}\{\mathbf{x} = \text{month} | \mathbf{y} = \text{RAIN}\} = 1 \end{aligned}$$

•

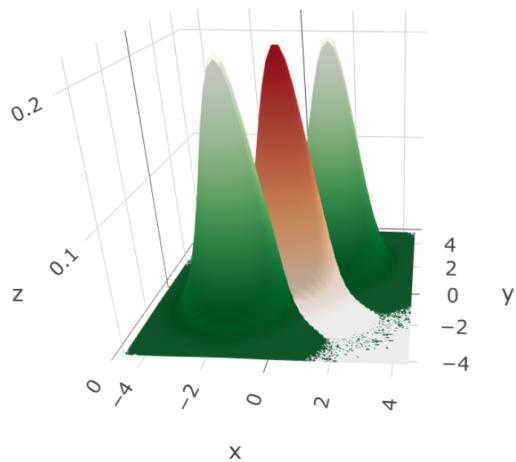


Figure 5.4: Class conditional distributions: the green class is distributed as a mixture of two Gaussians while the red class as a Gaussian.

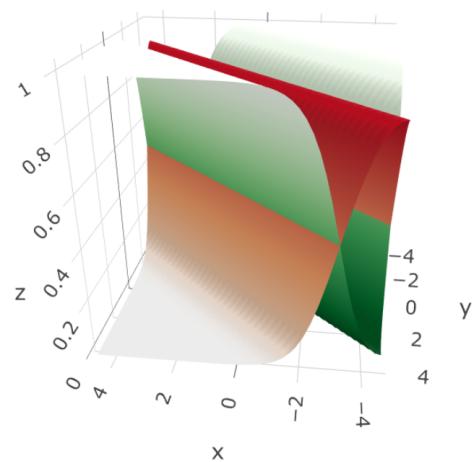


Figure 5.5: The a posteriori conditional distribution associated to the class-conditional distributions (equal a priori probability) in Figure 5.4.

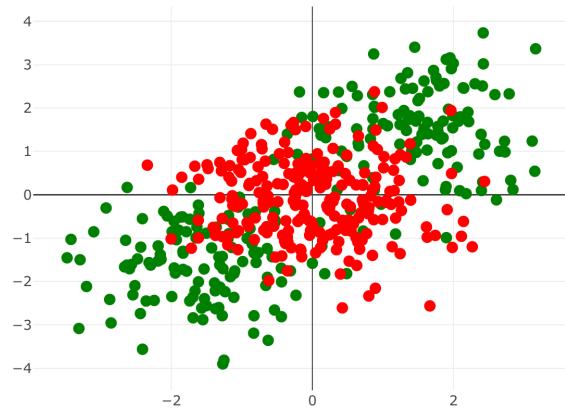


Figure 5.6: Dataset sampled according to the class-conditional distributions (equal a priori probability) in Figure 5.4.

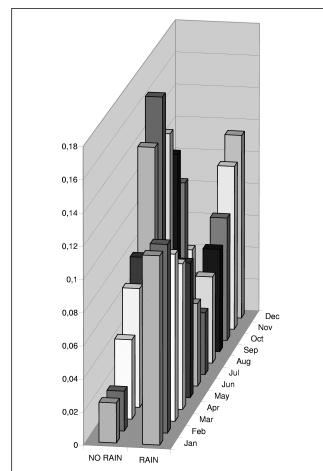


Figure 5.7: Inverse conditional distribution of the distribution in Figure 5.3

5.4 Dependency and regression

Consider the stochastic relationship between two continuous random variables $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}$ described by

$$F_{\mathbf{x},\mathbf{y}}(x, y) \quad (5.4.18)$$

This means that to each vector x sampled according to the $F_x(x)$ there corresponds a scalar y sampled from $F_y(y|\mathbf{x}=x)$. Assume that a set of N input/output observations is available. The estimation of the stochastic dependence on the basis of the empirical dataset requires the estimation of the conditional distribution $F_y(y|x)$. This is known to be a difficult problem but for prediction purposes, most of the time, it is sufficient to estimate the conditional expectation

$$f(x) = E_{\mathbf{y}}[\mathbf{y}|x] = \int_{\mathbf{y}} y dF_{\mathbf{y}}(y|x) \quad (5.4.19)$$

also known as the *regression function*.

The regression function is also related to the functional risk

$$R(\alpha) = \int L(y, h(x, \alpha)) dF_{\mathbf{x},\mathbf{y}}(x, y) = \int (y - h(x, \alpha))^2 dF_{\mathbf{x},\mathbf{y}}(x, y) \quad (5.4.20)$$

for the quadratic loss $L(y, h) = (y - h)^2$. From (2.14.75) it can be shown that the minimum (5.2.4) is attained by the regression function $h(\cdot, \alpha_0) = f(\cdot)$ if the function f belongs to the set $h(x, \alpha), \alpha \in \Lambda$.

Once defined the regression function f , the input/output stochastic dependency (5.4.18) is commonly represented in the *regression plus noise* form

$$\mathbf{y} = f(x) + \mathbf{w} = E_{\mathbf{y}}[\mathbf{y}|x] + \mathbf{w} \quad (5.4.21)$$

where \mathbf{w} denotes the noise term and satisfies $E[\mathbf{w}] = 0$ and $E[\mathbf{w}^2] = \sigma_w^2$. The role of the noise is to make explicit that some variability of the target cannot be explained by the regression function f . Notice that the assumption of an additive noise \mathbf{w} 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.

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.

5.5 Assessment of a learning machine

A learning machine works well if it exhibits good generalisation, 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 number of bad predictions, measured by the *generalisation error*. The generalisation error of a learning machine can be evaluated at two levels:

Hypothesis: Let α_N be the hypothesis returned by a learning algorithm for a training set D_N according to the ERM principle (Eq. (5.2.7)). The functional risk $R(\alpha_N)$ in (5.2.3) represents the *generalisation error of the hypothesis* α_N . This quantity is also known as *conditional error rate* [95] since it is conditional on a given training set D_N .

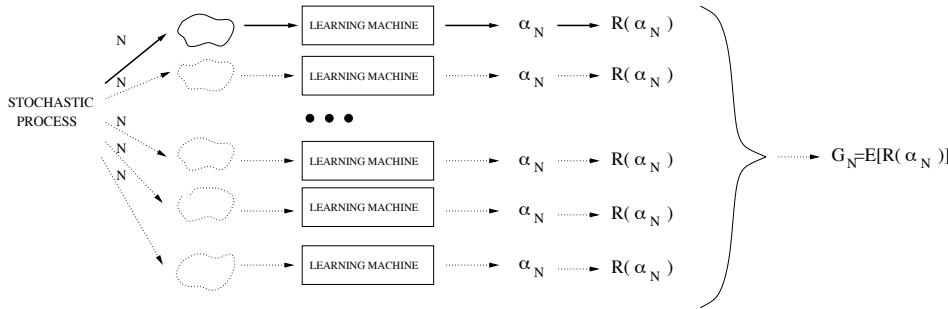


Figure 5.8: Functional risk vs. MISE

Algorithm: Let us define the average of the loss L for a given input x over the ensemble of training sets of size N as

$$g_N(x) = E_{\mathbf{D}_N, \mathbf{y}}[\mathbf{L}|\mathbf{x} = x] = \int_{\mathcal{Z}^N, \mathbf{y}} L(y, h(x, \alpha_N)) dF_{\mathbf{y}}(y|x) dF_{\mathbf{z}}^N(D_N) \quad (5.5.22)$$

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} , while the test input x is fixed. In the case of a quadratic loss function, this quantity corresponds to the *mean squared error* (MSE) defined in Section 3.5.6. By averaging the quantity (5.5.22) over the \mathcal{X} domain we have

$$G_N = \int_{\mathcal{X}} g_N(x) dF_{\mathbf{x}}(x) = E_{\mathbf{D}_N} E_{\mathbf{x}, \mathbf{y}}[L(\mathbf{y}, h(\mathbf{x}, \alpha_N))] \quad (5.5.23)$$

that is the *generalisation error of the algorithm* \mathcal{L} (also known as *expected error rate* [68] or *expected test error* [95]).

From (5.2.3) and (5.5.23) we obtain that

$$G_N = E_{\mathbf{D}_N}[R(\alpha_N)]$$

where $R(\alpha_N)$ is random because of the dependence on \mathbf{D}_N (Figure 5.8).

In the case of a quadratic loss function, the quantity

$$\text{MISE} = E_{\mathbf{D}_N} E_{\mathbf{x}, \mathbf{y}}[(\mathbf{y} - h(\mathbf{x}, \alpha_N))^2] \quad (5.5.24)$$

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 (5.2.7) chosen by ERM, the

second assesses the average performance of the algorithm over training sets with N observations. According to the hypothesis-based approach the goal of learning is to find, on the basis of observations, the hypothesis that minimises the functional risk. According to the algorithmic-based approach the goal is to find, on the basis of observations, the algorithm which minimises the generalisation error. Note that both quantities requires the knowledge of $F_{\mathbf{x}, \mathbf{y}}$ which is unfortunately unknown in real situations. A key issue in machine learning is then to take advantage of observable quantities, i.e. quantities that may be computed on the basis of the observed dataset, to estimate or approximate the measures discussed above. An important quantity in this sense is the empirical risk (5.2.8) which, however, has to be carefully manipulated in order to avoid too optimistic evaluations of the learning machine accuracy.

5.5.1 An illustrative example

The notation introduced in Section 5.2 and 5.5 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

$$p_{\mathbf{y}}(y|\mathbf{x} = x) = \mathcal{N}(x^3, 1) \quad (5.5.25)$$

where the conditional expected value $E[\mathbf{y}|x]$ is the regression function $f(x) = x^3$ and the noise \mathbf{w} has a unit variance.

- The training set $D_N = \{\langle x_i, y_i \rangle\}, i = 1, \dots, N$ consists of $N = 100$ i.i.d. pairs (Figure 5.9) generated according to the distribution 5.5.25. 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 characterised by the following three components:
 1. A class of hypothesis functions $h(x, \alpha) = \alpha x$ consisting of all the linear models passing through the origin. The class Λ 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

$$R_{\text{emp}}(\alpha) = \frac{1}{100} \sum_{i=1}^{100} (y_i - \alpha x_i)^2 \quad (5.5.26)$$

The empirical risk is a function of α and the training set. For the given training set D_N , the empirical risk as a function of α is plotted in Fig. 5.10.

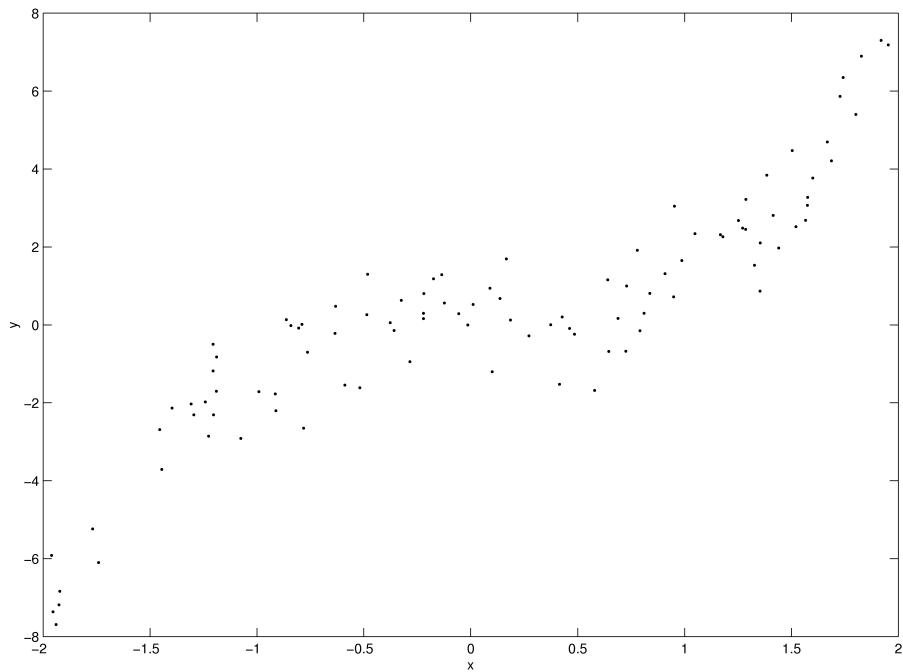


Figure 5.9: Training set (dots) obtained by sampling uniformly in the interval $[-2, 2]$ an input/output distribution with regression function $f(x) = x^3$ and unit variance.

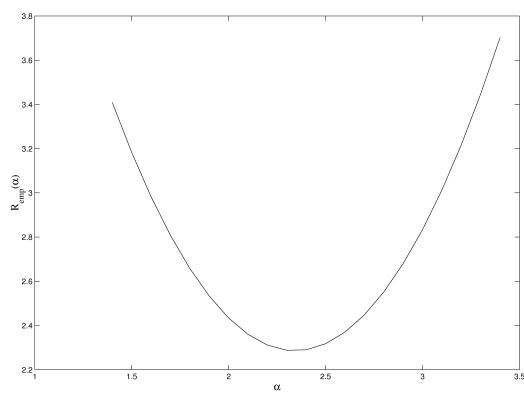


Figure 5.10: The empirical risk for the training set D_N vs. the model parameter value (x-axis). The minimum of the empirical risk is attained in $\alpha = 2.3272$.

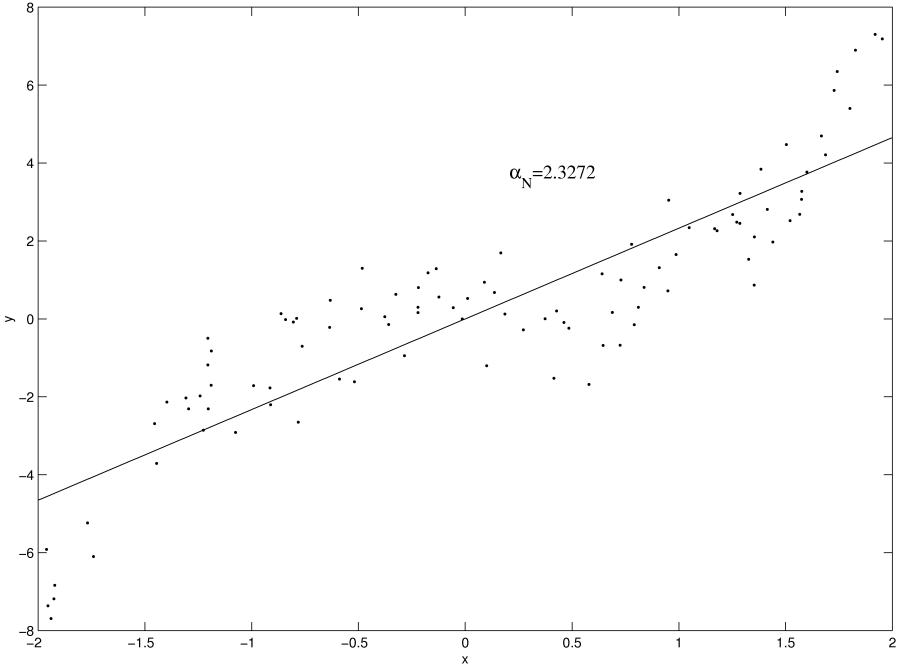


Figure 5.11: Training set (dotted points) and the linear hypothesis function $h(\cdot, \alpha_N)$ (straight line). The quantity α_N , which represents the slope of the straight line, is the value of the model parameter α which minimises the empirical risk.

For the dataset D_N in Figure 5.9, it is possible to obtain α_N by minimising the empirical risk (5.5.26)

$$\alpha_N = \arg \min_{\alpha \in \Lambda} R_{\text{emp}}(\alpha) = \arg \min_{\alpha \in \Lambda} \frac{1}{100} \sum_{i=1}^{100} (y_i - \alpha x_i)^2 = 2.3272 \quad (5.5.27)$$

The selected hypothesis is plotted in the input/output domain in Fig. 5.11.

If the joint distribution (e.g. its conditional expectation and variance) were to be known, it would also be possible to compute the risk functional (5.2.3) as

$$R(\alpha) = \frac{1}{4} \int_{-2}^2 (x^3 - \alpha x)^2 dx + 1 = 4 \frac{\alpha^2}{3} - \frac{32}{5} \alpha + 71/7 \quad (5.5.28)$$

where the derivation of the equality is sketched in Appendix D.15. For the given joint distribution, the quantity $R(\alpha)$ is plotted as a function of α in Fig. 5.12. The function takes a global minimum in $\alpha_0 = 2.4$ as can be derived from the analytical expression in (5.5.28).

The computation of the quantity (5.5.22) requires, however, an average over all the possible realisations of the random variable α_N for datasets of $N = 100$ points. Figure 5.13 shows 6 different realisations of the training set for the same conditional distribution (5.5.25) and the corresponding 6 values of α_N . Note that those six values may be considered as 6 different realisations of the sampling distribution (Section 3.4) of α_N .

It is important to remark that both the quantities (5.2.3) and (5.5.22) may be computed only if we know a priori the data joint distribution. Unfortunately, in 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.

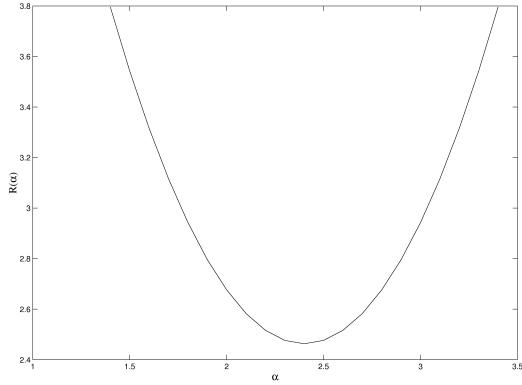


Figure 5.12: The functional risk (5.5.28) vs. the value of model parameter α (x-axis). The minimum of the functional risk is attained in $\alpha_0 = 2.4$.

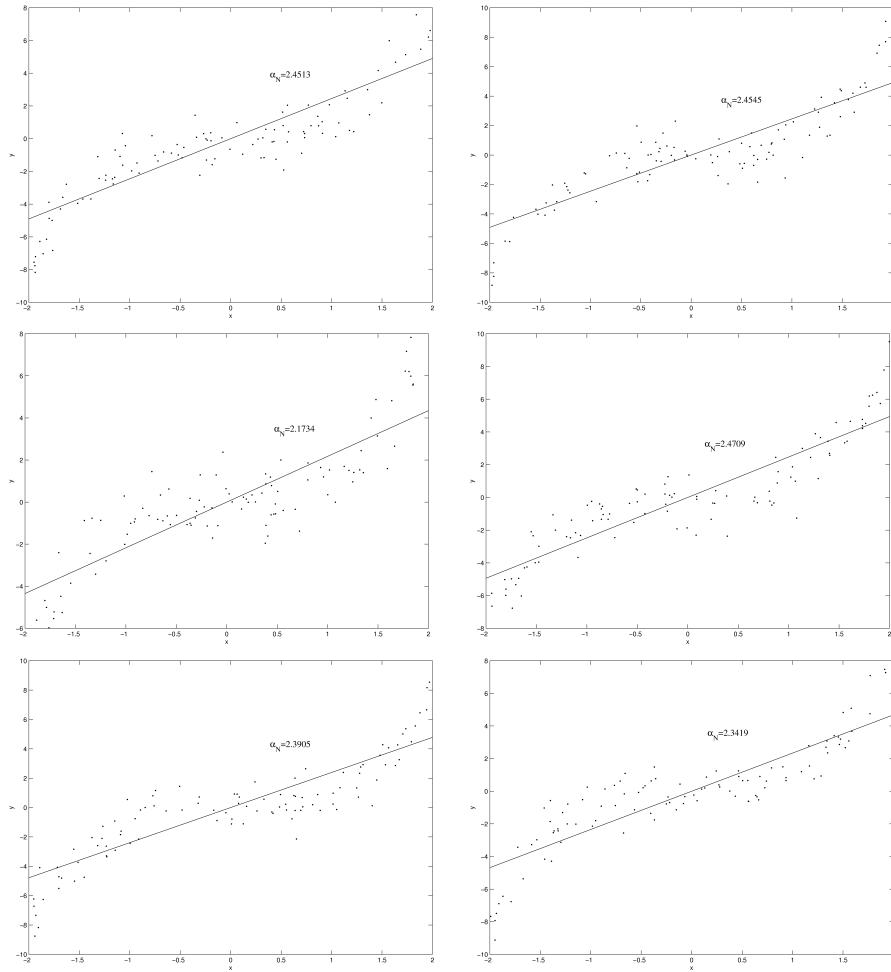


Figure 5.13: Six different realisations of a training set with $N = 100$ points (dots) and the relative hypotheses (solid straight lines) chosen according to the ERM principle (5.5.27).

Monte Carlo computation of generalisation error

The script `StatLearn/functRisk.R` computes by Monte Carlo the functional risk (5.5.28) for different values of α and returns the value of $\alpha_0 = 2.4$ which minimises it. Note that the functional risk is computed by generating a very large number of i.i.d. test examples.

The script `StatLearn/gener.R` computes by Monte Carlo the generalisation error (5.5.23). Unlike the previous script which considers only the predictive value of different hypotheses (with different α), this script assesses the average accuracy of the empirical risk minimisation strategy (5.5.27) for a finite number $N = 100$ of examples.

•

5.6 Generalisation error

In the previous section, we presented how Vapnik [169, 170, 171] formalised the learning task as the minimisation of functional risk $R(\alpha_N)$ in a situation where the joint distribution is unknown. This section focuses on the algorithm-based criterion G_N (Equation (5.5.24)) as a measure of the generalisation error of the learning machine.

In particular we will study how the generalisation error can be decomposed in the regression formulation and in the classification formulation.

5.6.1 The decomposition of the generalisation error in regression

Let us focus now on of the g_N measure (Equation (5.5.22)) of the generalisation error in the case of regression. In the case of a quadratic loss

$$L(y(x), h(x, \alpha)) = (y(x) - h(x, \alpha))^2 \quad (5.6.29)$$

the quantity g_N is often referred to as the *mean squared error* (MSE) and its marginal (5.5.24) as the *mean integrated squared error* (MISE). If the regression dependency is described in the *regression plus noise* form (5.4.21), the conditional target density can be written as

$$p_{\mathbf{y}}(y - f(x)|x) = p_{\mathbf{y}}(y - E_{\mathbf{y}}[\mathbf{y}|x]|x) = p_{\mathbf{w}}(w) \quad (5.6.30)$$

where \mathbf{w} is a noisy random variable with zero mean and variance $\sigma_{\mathbf{w}}^2$.

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 θ to be estimated is the quantity $f(x)$ and the estimator based on the training set is $\hat{\theta} = h(x, \alpha_N)$. The MSE quantity, defined in (3.5.17) coincides, apart

from an additional term, with the term (5.5.22) since

$$g_N(x) = E_{\mathbf{D}_N, \mathbf{y}}[\mathbf{L}|x] = \quad (5.6.31)$$

$$= E_{\mathbf{D}_N, \mathbf{y}}[(\mathbf{y} - h(x, \boldsymbol{\alpha}_N))^2] = \quad (5.6.32)$$

$$= E_{\mathbf{D}_N, \mathbf{y}}[(\mathbf{y} - E_{\mathbf{y}}[\mathbf{y}|x] + E_{\mathbf{y}}[\mathbf{y}|x] - h(x, \boldsymbol{\alpha}_N))^2] = \quad (5.6.33)$$

$$= E_{\mathbf{D}_N, \mathbf{y}}[(\mathbf{y} - E_{\mathbf{y}}[\mathbf{y}|x])^2 + 2\mathbf{w}(E_{\mathbf{y}}[\mathbf{y}|x] - h(x, \boldsymbol{\alpha}_N)) + \quad (5.6.34)$$

$$+ (E_{\mathbf{y}}[\mathbf{y}|x] - h(x, \boldsymbol{\alpha}_N))^2] = \quad (5.6.35)$$

$$= E_{\mathbf{y}}[(\mathbf{y} - E_{\mathbf{y}}[\mathbf{y}|x])^2] + E_{\mathbf{D}_N}[(h(x, \boldsymbol{\alpha}_N) - E_{\mathbf{y}}[\mathbf{y}|x])^2] = \quad (5.6.36)$$

$$= E_{\mathbf{y}}[\mathbf{w}^2] + E_{\mathbf{D}_N}[(h(x, \boldsymbol{\alpha}_N) - E_{\mathbf{y}}[\mathbf{y}|x])^2] \quad (5.6.37)$$

$$= \sigma_{\mathbf{w}}^2 + E_{\mathbf{D}_N}[(f(x) - h(x, \boldsymbol{\alpha}_N))^2] = \sigma_{\mathbf{w}}^2 + E_{\mathbf{D}_N}[(\theta - \hat{\theta})^2] = \quad (5.6.38)$$

$$= \sigma_{\mathbf{w}}^2 + \text{MSE} \quad (5.6.39)$$

Note that $\mathbf{y} = f(x) + \mathbf{w} = E_{\mathbf{y}}[\mathbf{y}|x] + \mathbf{w}$, f is fixed but unknown and that the noise term \mathbf{w} is independent of \mathbf{D}_N and satisfies $E[\mathbf{w}] = 0$ and $E[\mathbf{w}^2] = \sigma_{\mathbf{w}}^2$

We can then apply *bias/variance* decomposition (3.5.17) to the regression problem where $\theta = f(x)$ and $\hat{\theta} = h(x, \boldsymbol{\alpha}_N)$:

$$\begin{aligned} g_N(x) &= E_{\mathbf{D}_N, \mathbf{y}}[\mathbf{L}(x, y)] = \\ &= \sigma_{\mathbf{w}}^2 + E_{\mathbf{D}_N}[(h(x, \boldsymbol{\alpha}_N) - E_{\mathbf{y}}[\mathbf{y}|x])^2] = \\ &= \sigma_{\mathbf{w}}^2 + \quad \text{noise variance} \\ &\quad + (E_{\mathbf{D}_N}[h(x, \boldsymbol{\alpha}_N)] - E_{\mathbf{y}}[\mathbf{y}|x])^2 + \quad \text{squared bias} \\ &\quad + E_{\mathbf{D}_N}[(h(x, \boldsymbol{\alpha}_N) - E_{\mathbf{D}_N}[h(x, \boldsymbol{\alpha}_N)])^2] = \quad \text{model variance} \\ &= \sigma_{\mathbf{w}}^2 + B^2(x) + V(x) \end{aligned} \quad (5.6.40)$$

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 $f(x) = E_{\mathbf{y}}[\mathbf{y}|x]$. 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 of the target. So by Eq. (5.5.24) by averaging (5.6.40) over \mathcal{X} we obtain

$$\text{MISE} = G_N = \sigma_{\mathbf{w}}^2 + \int_{\mathcal{X}} B^2(x) dF_{\mathbf{x}} + \int_{\mathcal{X}} V(x) dF_{\mathbf{x}} \quad (5.6.41)$$

where the three terms are

1. the intrinsic noise term reflecting the target alone,
2. the integrated squared bias reflecting the target's relation with the learning algorithm and
3. the integrated variance term reflecting the learning algorithm alone.

As the aim of a learning machine is to minimise the quantity G_N and the computation of (5.6.41) requires the knowledge of the joint input/output distribution, this decomposition could appear as a useless theoretical exercise. In practical settings, the designer of a learning machine does not have access to the term G_N but can only estimate it on the basis of the training set. Nevertheless, the bias/variance decomposition is relevant in practical learning too since it provides a useful hint about how to control the error G_N . In particular, the bias term measures the lack of representational power of the class of hypotheses and is intrinsically related to the (restrictive) choice of the hypothesis space (Section 5.2). This means that to reduce

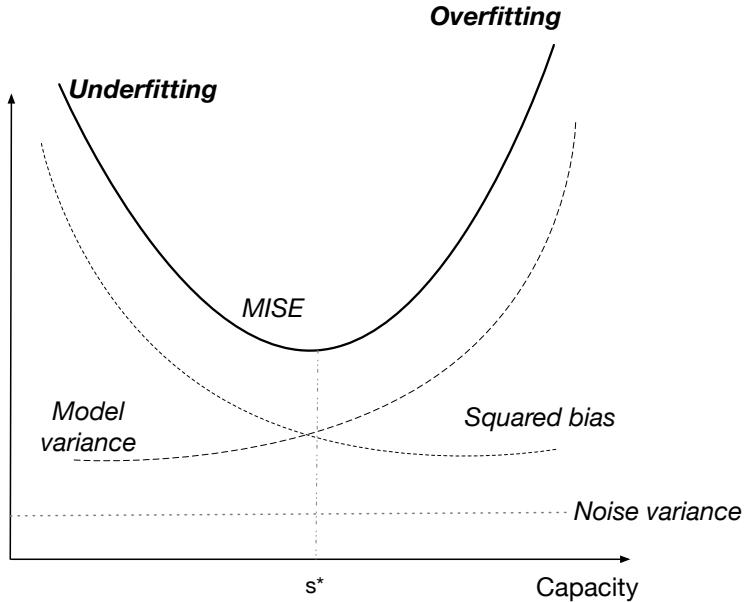


Figure 5.14: Bias/variance/noise tradeoff in regression: this is a qualitative representation of the relationship between the hypothesis' bias and variance and the capacity of the class of functions. The MISE generalisation error is the sum of the three terms (squared bias, hypothesis variance and noise variance) as shown in (5.6.41). Note that the variance of the noise is supposed to be target independent and then constant.

the bias term of the generalisation error we should consider an hypothesis space with a larger capacity s , or, in other words, hypotheses which can approximate a larger amount of input/output mappings. On the other side, the variance term warns us against an excessive capacity (or complexity) s of the hypothesis space. This means that a class of too powerful hypotheses runs the risk of being excessively sensitive to the noise (or fluctuations) affecting the training set; therefore, our class Λ_s 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 a hypothesis with large bias but low variance *underfits* the data while a 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.

A graphical illustration of the bias/variance/noise tradeoff (5.6.41) is made in Figure 5.14. The left side of the figure corresponds to an underfitting configuration where the model has too low capacity (i.e. high bias) to capture the nonlinearity of the regression function. The right side of the figure corresponds to an overfitting configuration where the model capacity is too large (i.e. high variance) leading then to high instability and poor generalisation. Note that Figure 5.14 requires a formal definition of the notion of capacity and that it is only a qualitative visualisation of the theoretical link between the hypothesis' properties and the capacity of the class of functions. Nevertheless it provides useful hints about the impact of the learning procedure on the final generalisation accuracy. The task of the model designer is to search for the optimal trade-off between the variance and the bias terms (ideally the capacity s^* in Figure 5.14), on the basis of the available training set. Section 5.8 will discuss how this search proceeds in practice in a real setting.

Two naive predictors

Consider a regression task $\mathbf{y} = f(\mathbf{x}) + \mathbf{w}$, where $\text{Var}[\mathbf{w}] = \sigma_w^2$ and two naive predictors:

$$1. h^{(1)}(x) = 0$$

$$2. h^{(2)}(x) = \frac{\sum_{i=1}^N y_i}{N}$$

What about their generalisation errors in $x = \bar{x}$? By using (5.6.40) we obtain

$$1. g_N^{(1)}(\bar{x}) = \sigma_w^2 + f(\bar{x})^2$$

$$2. g_N^{(2)}(\bar{x}) = \sigma_w^2 + (f(\bar{x}) - E[\mathbf{y}])^2 + \text{Var}[\mathbf{y}] / N$$

The script `StatLearn/naive.R` executes a Monte Carlo validation of the formulas above.

•

5.6.2 The decomposition of the generalisation 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).

$$\text{MME}(x) = E_{\mathbf{y}, \mathbf{D}_N}[L(\mathbf{y}, h(x, \boldsymbol{\alpha}_N))|x] = E_{\mathbf{y}, \mathbf{D}_N}[L(\mathbf{y}, \hat{\mathbf{y}})] = \quad (5.6.42)$$

$$= E_{\mathbf{y}, \mathbf{D}_N}\left[\sum_{k,j=1}^K L_{(j,k)} \mathbb{1}(\hat{\mathbf{y}} = c_j|x) \mathbb{1}(\mathbf{y} = c_k|x)\right] = \quad (5.6.43)$$

$$= \sum_{k,j=1}^K L_{(j,k)} E_{\mathbf{D}_N}[\mathbb{1}(\hat{\mathbf{y}} = c_j|x)] E_{\mathbf{y}}[\mathbb{1}(\mathbf{y} = c_k|x)] = \quad (5.6.44)$$

$$= \sum_{k,j=1}^K L_{(j,k)} \text{Prob}\{\hat{\mathbf{y}} = c_j|x\} \text{Prob}\{\mathbf{y} = c_k|x\} \quad (5.6.45)$$

where $\mathbb{1}(\cdot)$ is the indicator function which returns zero when the argument is false and one otherwise. Note that the distribution of $\hat{\mathbf{y}}$ depends 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

$$\begin{aligned} \text{MME}(x) &= \sum_{k,j=1}^K \mathbb{1}(c_j \neq c_k) \text{Prob}\{\hat{\mathbf{y}} = c_j|x\} \text{Prob}\{\mathbf{y} = c_k|x\} = \\ &= 1 - \sum_{k,j=1}^K \mathbb{1}(c_j = c_k) \text{Prob}\{\hat{\mathbf{y}} = c_j|x\} \text{Prob}\{\mathbf{y} = c_k|x\} = \\ &= 1 - \sum_k \text{Prob}\{\hat{\mathbf{y}} = c_k|x\} \text{Prob}\{\mathbf{y} = c_k|x\} = \text{Prob}\{\mathbf{y} \neq \hat{\mathbf{y}}\} \quad (5.6.46) \end{aligned}$$

A decomposition of a related quantity was proposed in [180]. Let us consider the squared sum:

$$\begin{aligned} & \frac{1}{2} \sum_{j=1}^K (\text{Prob}\{\mathbf{y} = c_j\} - \text{Prob}\{\hat{\mathbf{y}} = c_j\})^2 = \\ & \frac{1}{2} \left(\sum_{j=1}^K \text{Prob}\{\mathbf{y} = c_j\}^2 \right) + \frac{1}{2} \left(\sum_{j=1}^K \text{Prob}\{\hat{\mathbf{y}} = c_j\}^2 \right) - \sum_{j=1}^K \text{Prob}\{\mathbf{y} = c_j\} \text{Prob}\{\hat{\mathbf{y}} = c_j\} \end{aligned}$$

By adding one to both members and by using (5.6.42) we obtain a decomposition analogous to the one in (5.6.40)

$$\begin{aligned} g_N(x) = \text{MME}(x) = & \\ & = \frac{1}{2} \left(1 - \left(\sum_{j=1}^K \text{Prob}\{\mathbf{y} = c_j|x\}^2 \right) \right) + \quad \text{"noise"} \\ & + \frac{1}{2} \sum_{j=1}^K (\text{Prob}\{\mathbf{y} = c_j|x\} - \text{Prob}\{\hat{\mathbf{y}} = c_j|x\})^2 + \quad \text{"squared bias"} \quad (5.6.47) \\ & + \frac{1}{2} \left(1 - \left(\sum_{j=1}^K \text{Prob}\{\hat{\mathbf{y}} = c_j|x\}^2 \right) \right) \quad \text{"variance"} \end{aligned}$$

The noise term measures the degree of uncertainty of \mathbf{y} and consequently the degree of stochasticity of the dependence. It equals zero if and only if there exists a class c such that $\text{Prob}\{\mathbf{y} = c|x\} = 1$ and zero otherwise. 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 $\hat{\mathbf{y}} = h(x, \alpha_N)$ 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} .

5.6.3 Stability and generalisation

Section ?? presented some probabilistic bounds on the difference between empirical and generalisation error derived by Vapnik on the basis of the capacity of the class of hypothesis. Those bounds concern quantities like

$$\text{Prob} \left\{ \sup_{\alpha} |R(\alpha) - \mathbf{R}_{\text{emp}}(\alpha)| > \varepsilon \right\}$$

where the dependence of the algorithm on the training set \mathbf{D}_N is not taken into account and the randomness of $\mathbf{R}_{\text{emp}}(\alpha)$ is only due to the randomness of the training set \mathbf{D}_N given that α is considered as fixed.

Probabilistic bounds based on algorithm properties (notably stability) have been discussed in [36] Those algorithm-based bounds take the form

$$\text{Prob}_{\mathbf{D}_N} \{ |R(\alpha_N) - R_{\text{emp}}(\alpha_N)| > \varepsilon \}$$

where α is not fixed but has the distribution α_N . Such bounds are based on inequalities that relate moments of multi-dimensional random functions (like learning algorithms) to their sensitivity.

An important result is the McDiarmid inequality that concerns functions of datasets of size N :

Theorem 6.1. Let $F : \mathbb{R}^N \rightarrow \mathbb{R}$ be a function of a dataset $D_N = \{z_1, \dots, z_N\}$ of size N and c_i a set of constants such that

$$\sup_{D_N \in \mathbb{R}^N, z_i \in \mathbb{R}} |F(D_N) - F(D_N^{(i)})| \leq c_i$$

where $D_N^{(i)}$ is obtained by replacing the i th element of D_N with z_i . Then

$$\text{Prob}_{\mathbf{D}_N}\{|F(\mathbf{D}_N) - E_{\mathbf{D}_N}[F(\mathbf{D}_N)]| \geq \varepsilon\} \leq \exp \frac{-2\varepsilon^2}{\sum_{i=1}^N c_i^2}$$

Let us consider now a binary classification task and a learning algorithm returning the hypothesis $h(\cdot, \alpha_N)$ for a dataset D_N .

Definition 6.2. A deterministic learning algorithm is ε -uniform change-one stable if for all datasets D_N and D'_N that differ in one element and for all inputs x

$$|h(x, \alpha_N) - h(x, \alpha'_N)| \leq \varepsilon$$

It can be shown the following theorem [151].

Theorem 6.3. Let us consider a ε -uniform leave-one-out stable learning algorithm $h(\cdot, \alpha_N)$. For every data distribution the expected generalisation error is at most ε

$$|E_{\mathbf{D}_N}\{|R(\alpha_N) - R_{emp}(\alpha_N)| \leq \varepsilon$$

where

$$R_{emp}(\alpha_N) = \frac{1}{N} \sum_{i=1}^N |y_i - h(x_i, \alpha_N)|$$

It is interesting to see that this theorem formalises the role of variance (Section 5.6.1) of a learning algorithm in assessing its generalisation capability. The larger the variance of a learner, the larger the ε term, the less reliable is the empirical error in estimating the generalisation term. Given the feasibility of estimating the variance of a learner (notably by resampling techniques), such results provide a theoretical support to the adoption of resampling based strategy for model assessment and selection in machine learning.

5.7 The hypothesis-based vs the algorithm-based approach

In the previous sections we introduced two different manners of assessing the accuracy of a learning machine. The reader could logically raise the following question: which approach is the most adequate in practice?

Instead of providing a direct answer to such a question, we prefer to conduct a short comparison of the assumptions and limitations related to the two approaches.

The hypothesis-based approach formulates learning as the problem of finding the hypothesis which minimises the functional risk. Vapnik reformulates this problem into the problem of consistency of a learning process based on ERM. The main result is that it is possible to define a probabilistic distribution-free bound on the functional risk which depends on the empirical risk and the VC dimension of the class of hypothesis. Though this achievement is impressive from a theoretical and scientific perspective (it was published in a Russian book in the 60s), its adoption in practical settings is not always easy for several reasons: results derive from asymptotic considerations though learning by definition deals with finite samples,

the computation of the VC dimension is explicit only for specific classes of hypothesis functions and the bound, derived from worst-case analysis, is not always tight enough for practical purposes. Also, it is hard to associate notions of VC complexity to learning strategies characterised by increasing complexity and pipelines of multiple steps (for instance feature selection, averaging, transfer learning).

The algorithm-based approach relies on the possibility of emulating the stochastic process underlying the dataset by means of resampling procedures like cross-validation or bootstrap. Note that this approach is explicitly criticised by Vapnik and others who consider it inappropriate to reason in terms of data generation once a single dataset is available. According to [59] "averaging over the data would be unnatural, because in a given application, one has to live with the data at hand. It would be marginally useful to know the number G_N as this number would indicate the quality of an average data sequence, not *your* data sequence". Nevertheless, though it is hard to guarantee formally the accuracy of a resampling strategy, its general-purpose nature, simplicity and ease of implementation have been, along years, key ingredients of its success.

Whatever the degree of realism of the hypothesis made by the two approaches is, it is worth making a pragmatic and historical consideration. Though the Vapnik results represent a major scientific success and underlie the design of powerful learning machines (notably SVM), in a wider perspective it is fair to say that cross-validation is the most common and successful workhorse of practical learning applications. This means that, though most data scientists have been eager to formalise the consistency of their algorithms in terms of Vapnik bounds, in practice they had recourse to intensive cross-validation tricks to make it work in the real world. Now, more than 60 years after the first computational version of learning processes, we have enough evidence to say that cross-validation is a major element of the machine learning success story. Moreover, results like the ones in Section 5.6.3 support, from a theoretical perspective too, the role of leave-one-out in assessing the generalisation accuracy of a learning algorithm. This is the reason why in the following sections we will focus on an algorithm-based approach aiming to assess (and minimise) the generalisation error by means of a resampling strategy.

5.8 The supervised learning procedure

The goal of supervised learning is to return the hypothesis with the lowest generalisation error. Since we assume that data samples are generated in a random way, there is no hypothesis which gives a null generalisation error. Therefore, the generalisation error G_N of the hypothesis returned by a learning machine has to be compared to the minimal generalisation error that can be attained by the best single-valued mapping. Let us define by Λ^* the set of all measurable single-valued mappings $h : \mathcal{X} \rightarrow \mathcal{Y}$ and consider the hypothesis

$$\alpha^* = \arg \min_{\alpha \in \Lambda^*} R(\alpha) \quad (5.8.48)$$

where $R(\alpha)$ has been defined in (5.2.3).

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 5.5.1, α^* 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 generalisation error G_N of the model with parameters α_N learned from the dataset D_N 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 Λ could not be large enough to contain the hypothesis α^* .

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

$$\Lambda_1 \subseteq \dots \subseteq \Lambda_s \subseteq \dots \Lambda_S \quad (5.8.49)$$

is defined so that $\Lambda^* = \cup_{s=1}^S \Lambda_s$ where s denotes the capacity of the class. This guarantees that the set of hypotheses taken into consideration will necessarily contain the best hypothesis α^* .

A priori information 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 $h(\cdot, \alpha_N^s)$, $s = 1, \dots, S$, is selected by minimising the empirical risk (5.2.8). 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 generalisation error G_N^s of the hypothesis α_N^s . This step is called the *validation* step of the learning procedure.
4. The hypothesis $h(\cdot, \alpha_N^{\bar{s}}) \in \Lambda_{\bar{s}}$ with

$$\bar{s} = \arg \min_s \hat{G}_N^s \quad (5.8.50)$$

is returned as the final outcome. This final step is called the *model selection* step.

In order to accomplish the learning procedure, and specifically the selection in (5.8.50), we need an estimation of the generalisation error (Section 5.9). However, since the estimator of the generalisation error may be affected by an error (as any estimator), this may induce an error and variability in the model selection step (5.8.50) (Figure 5.15).

5.9 Validation techniques

This section discusses validation methods to estimate the generalisation error G_N from a finite set of N observations.

The empirical risk (also called *apparent error*) $R_{\text{emp}}(\alpha_N)$ introduced in (5.2.7) could be the most intuitive estimator of G_N . However, it is generally known that the empirical risk is a biased (and optimistic) 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 D.16. In Section 7.1.16 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, an example of the first approach is the FPE criterion (presented in Section 7.1.16.2) while examples of the second approach are the resampling procedures.

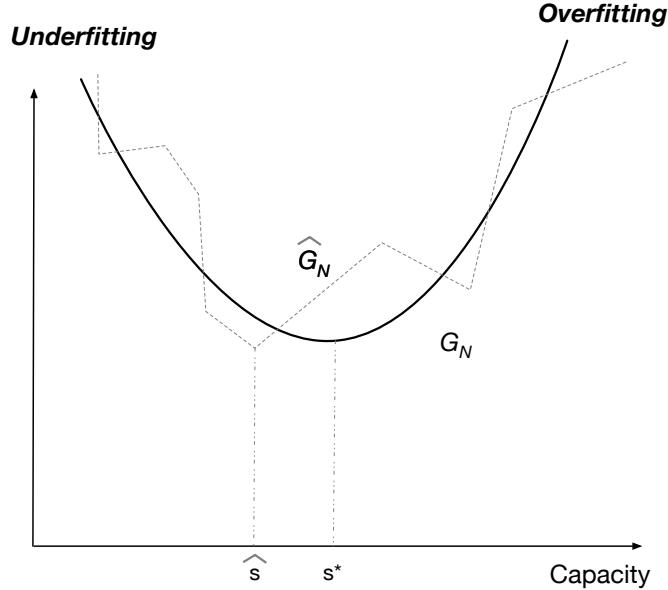


Figure 5.15: Bias/variance/noise tradeoff and model selection: since the generalisation error (e.g. MISE) is not accessible in practical settings, model selection is performed on the basis of an estimation (dotted line) which may induce an error (and variability) in the selection (5.8.50) of the best capacity.

5.9.1 The resampling methods

Cross-validation [160] 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} examples, the second containing $N_{ts} = N - N_{tr}$ examples. Each time, N_{tr} examples are used by the parametric identification algorithm \mathcal{L} to select a hypothesis $\alpha_{N_{tr}}^i$, $i = 1, \dots, l$, from Λ and the remaining N_{ts} examples are used to estimate the error of $h(\cdot, \alpha_{N_{tr}}^i)$ (Fig. 5.16)

$$\hat{R}_{ts}(\alpha_{N_{tr}}^i) = \sum_{j=1}^{N_{ts}} L(y_j, h(x_j, \alpha_{N_{tr}}^i)) \quad (5.9.51)$$

The resulting average of the l errors $\hat{R}_{ts}(\alpha_{N_{tr}}^i)$, $i = 1, \dots, l$, is the cross-validation estimate

$$\hat{G}_{cv} = \frac{1}{l} \sum_{i=1}^l \hat{R}_{ts}(\alpha_{N_{tr}}^i) \quad (5.9.52)$$

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

$$\hat{G}_{loo} = \frac{1}{N} \sum_{i=1}^N L(y_i, h(x_i, \alpha_{N(i)})) \quad (5.9.53)$$

In this case l equals the number of training points and $N_{ts} = 1$.

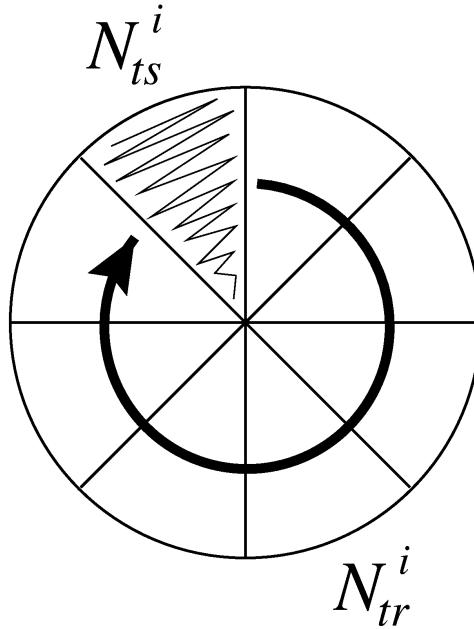


Figure 5.16: Partition of the training dataset in the i^{th} fold of cross-validation. The quantity N_{tr}^i is the amount of training points while N_{ts}^i is the number of test points.

Bootstrap (Section 4.4) is also used to return a nonparametric estimate of G_N , by repeatedly sampling the training cases *with replacement*. Since empirical risk is a biased optimistic estimation of generalisation error and bootstrap is an effective method to assess bias (Section 4.4.3), it follows that bootstrap plays a role in a validation strategy.

A bootstrap sample $D_{(b)}$ is a “fake” dataset $\{z_{1b}, z_{2b}, \dots, z_{Nb}\}$, $b = 1, \dots, B$ randomly selected from the training set $\{z_1, z_2, \dots, z_N\}$ with replacement.

Efron and Tibshirani [67] proposed to use bootstrap to correct the bias (or optimism) of empirical risk by adopting a strategy similar to Section 4.4.3. Equation (4.4.9) estimates the bias of an estimator by computing the gap between the average bootstrap (4.4.8) estimate and the sample estimation. In the case of generalisation, the sample estimation $\hat{\theta}$ is the empirical risk and the bootstrap estimate $\hat{\theta}_{(.)}$ may be computed as follows

$$\hat{G}_{(.)} = \frac{1}{B} \left[\sum_{i=1}^N (P_{ib} L(y_i, h(x_i, \alpha_{(b)}))) \right] \quad (5.9.54)$$

where P_{ib} indicates the proportion of the bootstrap sample $D_{(b)}$, $b = 1, \dots, B$ containing the i^{th} training point z_i ,

$$P_{ib} = \frac{\#_{j=1}^N (z_{jb} = z_i)}{N} \quad (5.9.55)$$

and $\alpha_{(b)}$ is the output of the parametric identification performed on the set $D_{(b)}$.

The difference between empirical risk and (5.9.54)

$$\text{Bias}_{\text{bs}} = \frac{1}{B} \sum_{b=1}^B \sum_{i=1}^N \left((P_{ib} - \frac{1}{N}) L(y_i, h(x_i, \alpha_{(b)})) \right) \quad (5.9.56)$$

is the bias correction term to be subtracted to empirical risk to obtain a *bootstrap bias corrected* estimate (4.4.10) of the generalisation error.

An alternative consists in using the holdout principle in combination with the bootstrap one [67]. Since each bootstrap set is a resampling of the original training set, it may happen that some of the original examples (called *out-of-bag*) do not belong to it: we can then use them to have an independent holdout set to be used for generalisation assessment. The bootstrap estimation of the generalisation error (also known as E0) is then

$$\hat{G}_{\text{bs}} = \frac{1}{N} \sum_{i=1}^N \frac{1}{|B^{(i)}|} \sum_{b \in B^{(i)}} L(y_i, h(x_i, \alpha_{(b)})) \quad (5.9.57)$$

where $B^{(i)}$ is the set of bootstrap samples which do not contain the i th point and $|B^{(i)}|$ is its size. The terms where $|B^{(i)}| = 0$ are discarded.

5.10 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 some major difficulties.

Models are approximations: in most situations we will not be able to guess or at least get close to the correct or true model. Still worse, even knowing the class to which the true model belongs, we should better disregard such information, if the class is too large to be properly explored with the help of a finite dataset. In the George Box's sense the most useful model could not necessarily belong to the class of the true model.

Minimisation of the empirical risk: in a general case finding the global minimum of the empirical risk as in (5.2.7) demands the resolution of a multivariate and nonlinear optimisation 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, a single random realisation of the statistical process, made of a finite number of input/output pairs, 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 realisations 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 realisation. 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.

5.11 Exercises

1. Consider an input/output regression task where $n = 1$, $E[y|x] = \sin(\pi x/2)$ and $p(y|x) = \mathcal{N}(\sin(\pi x/2), \sigma^2)$, $\sigma = 0.1$ and $\mathbf{x} \sim \mathcal{U}(-2, 2)$. Let N be the size of the training set and consider a quadratic loss function.

Let the class of hypothesis be $h_M(x) = \alpha_0 + \sum_{m=1}^M \alpha_m x^m$ with $\alpha_j \in [-2, 2], j = 0, \dots, M$.

For $N = 20$ generate $S = 50$ replicates of the training set. For each replicate, estimate the value of the parameters that minimise the empirical risk, compute the empirical risk and the functional risk.

1. Plot the evolution of the distribution of the empirical risk for $M = 0, 1, 2$.
2. Plot the evolution of the distribution of the functional risk for $M = 0, 1, 2$.

Hints: to minimise the empirical risk, perform a grid search in the space of parameter values, i.e. by sweeping all the possible values of the parameters in the set $[-1, -0.9, -0.8, \dots, 0.8, 0.9, 1]$. To compute the functional risk by Monte Carlo, generate a set of $N_{ts} = 10000$ i.i.d. input/output testing examples.

Solution: See the file `Exercise6.pdf` in the directory `gbcode/exercises` of the companion R package (Appendix G).

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. Modelling from data is often viewed as an art form, mixing the insight of the expert with the information contained in the observations. Typically, a modelling process is not a sequential process but is better represented as a sort of loop with a lot of feedback 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 modelling approaches. The following chapters will be instead devoted to the discussion of the peculiarities of specific learning approaches.

We partition the data modelling process into 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 preliminary conversion and filtering of data is performed.

Once the numeric 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 models (e.g. polynomial models, neural networks) characterised by a *capacity* (or complexity) index (or hyper-parameter) (e.g. degree of the polynomial, number of neurons, VC dimension) which controls the approximation power of the model. According to the capacity index, the set of 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 points (*validation*). Finally, a *model selection* procedure selects the final model to be used for future predictions. A common alternative to

model selection is a *model combination*, where a combination (e.g. averaging) of the most promising models is used to return a meta-model, presumably with better accuracy properties.

The problem of parametric identification is typically a problem of multivariate optimisation [2]. Section 6.6 introduces the most common optimisation algorithms for linear and nonlinear configurations. Structural identification is discussed in Section 6.8, which focuses on the existing methods for model generation, model validation and model selection. The last section concludes and resumes the whole modelling process with the support of a diagram.

6.2 Problem formulation

The problem formulation is the preliminary and somewhat the most critical step of a learning procedure. The model designer chooses a particular application domain (e.g. finance), a phenomenon to be studied (e.g. the credit risk of a customer) and hypothesises the existence of an unknown dependency (e.g. between the financial situation of the customer and the default risk) which is to be estimated from experimental data. First, the modeller specifies a set of *constructs*, i.e. abstract concepts or high-level topics which are potentially relevant for the study (e.g. the profile and the financial situation of a client). Second, a set of *variables* (e.g. the client age and her salary) is defined by grounding the constructs into a measurable form. Eventually, an *operationalisation*, i.e. the definition of how to measure those variables (e.g. by accessing a bank database), is proposed.

In this step, domain-specific knowledge and experience are the most crucial requirements to come up with a meaningful problem formulation. Note that the time spent in this phase is usually highly rewarding and can save vast amounts of modelling time. There is often no substitute for physical intuition and human analytical skills.

6.3 Experimental design

The most precious thing in data-driven modelling 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 data. In input/output modelling, it is essential that the training set be a representative sample of the phenomenon and cover the input space adequately. 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* [74]. The study of experimental design is concerned with locating training input data in the space of input variables so that the performance of the modelling process is maximised. However, in some cases, the designer cannot manipulate the process of collecting data, and the modelling process has to deal with what is available. This configuration, which is common to many real problems, is called the *observational setting* [44]. Though this setting seems the most adequate for a learning approach ("just learn from what you observe"), it is worth reminding that most of the time, behind an observation setting, there is the strong implicit assumption that the observations are i.i.d. samples of a stationary (i.e. invariant) stochastic process. Now, in most realistic cases, this assumption is not valid (at least not for a long time), and considerations of nonstationarity, drift should be

integrated in the learning process. Other problems are related to the poor causal value of inferences made in an observational setting, e.g. in situations of sampling bias or non-observable variables. Nevertheless, given the introductory nature of this book, in what follows, we will limit to consider the simplest observational and stationary setting.

6.4 Data pre-processing

Once data have been recorded, it is common practice to pre-process them. The hope is that such treatment might make learning easier and improve the final accuracy.

Pre-processing includes a large set of actions on the observed data, and some of them are worth being discussed:

Numerical encoding. Some interesting data for learning might not be in a numeric format (e.g. text, image). Since, in what follows, we will assume that all data are numeric, a preliminary conversion or encoding step is needed. Given that most encoding procedures are domain-specific, we will not further discuss them here.

Missing data treatment. In real applications, it often happens that some input values are missing (or Not Available (NA)). If the quantity of data is sufficiently large, the simplest solution is to discard the examples having missing features. When the amount of data is too restricted or there are too many partial examples, it becomes important to adopt some specific heuristics to deal with the problem [91]. Note, however, that every missing data treatment strategy makes assumptions about the process that caused observations to be missing. It is recommended to be aware of such assumptions before applying any missing data treatment strategy .

Categorical variables. It may be convenient to treat categorical variables, specifically in situations when they may take a very large number of values (e.g. names of retailers in a business intelligence application). Two common ways to deal with are: i) replace them with dummy variables encoding the different values in binary terms (e.g. K bits for K categories) ii) replace each category with numerical values informative about the conditional distribution of the target given such category: for instance, in regression (binary classification) we could replace a category $x = "black"$ with an estimation of $E[\mathbf{y}|x = "black"] = \text{Prob}\{\mathbf{y} = 1|x = "black"\}$.

Feature selection. The feature selection problem consists in selecting a relevant subset of input variables in order to maximise the performance of the learning machine. This approach is useful if there are inputs that carry only a little useful information or 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 [60], principal component analysis [131] and the general wrapper approach [112]. For more details, we refer the reader to Chapter 10.

Outliers removal. Outliers are unusual data values that are not consistent with most of the observations. Commonly, outliers are due to wrong measurement procedures, storage errors and coding malfunctioning. There are two common strategies to deal with outliers: the first is performed at the preprocessing stage [73] and consists in their detection and consequent removal, the second

is to delay their treatment at the model identification level by adopting robust methodologies [101] 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* to standardise all variables to have a zero mean and a unity variance¹, *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 pre-processing phase is a dataset in a tabular numeric form where each row represents a particular observation (also called instance, example or data point), and each column a descriptive variable (also called feature, attribute or covariate). We denote the dataset

$$D_N = \{z_1, z_2, \dots, z_N\}$$

where N is the number of examples, n is the number of features, the i^{th} example is an input/output pair $z_i = \langle x_i, y_i \rangle$, $i = 1, \dots, N$, 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 i.i.d. generated by the following stochastic dependency:

$$\mathbf{y} = f(\mathbf{x}) + \mathbf{w}, \quad (6.5.1)$$

where $E[\mathbf{w}] = 0$ and σ_w^2 is the noise variance.

The noise term \mathbf{w} is supposed to lump together all the unmeasured contributions to the variability of \mathbf{y} , like, for instance, missing or non-observable variables. There are two main assumptions underlying Equation (6.5.1). The first is that the noise is independent of the input and has a constant variance. This assumption also called *homoskedasticity* in econometrics is typically made in machine learning because of the primary focus on the dependency f and the lack of effective methodologies to assess it a priori in nonlinear and high dimensional settings. The reader should be aware, however, that heteroskedastic configurations may have a strong impact on the final model accuracy and that some a priori output variable transformation and/or a posteriori assessment is always recommended (e.g. study of the residual distribution after fitting). The second assumption is that noise enters additively to the output. 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 [9]. As far as this problem is concerned, we adopt the pragmatic approach proposed by Ljung [119], 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^{th} row is the vector x_i^T ,
2. the *output vector* Y of dimension $[N \times 1]$ whose i^{th} component is the scalar y_i .

¹This can be easily done with numeric inputs by the R command `scale`

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 $\alpha_N \in \Lambda$ minimises the loss function

$$R_{\text{emp}}(\alpha) = \frac{1}{N} \sum_{i=1}^N L(y_i, h(x_i, \alpha)) \quad (6.6.2)$$

This phase of the learning procedure requires the resolution of the optimisation task (5.2.7). 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 optimisation problem (5.2.7) as

$$\alpha_N = \arg \min_{\alpha \in \Lambda} J(\alpha). \quad (6.6.3)$$

Also, we will use the term *model* as a synonymous of *hypothesis*.

6.6.1 Error functions

The choice of an optimisation 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 minimising the mean squared error (Equation 2.11.35). This motivates the use of a quadratic function

$$L(y, h(x, \alpha)) = (y - h(x, \alpha))^2 \quad (6.6.4)$$

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 also be used for classification, there are more appropriate error functions to be considered [61]. The most used is *cross-entropy* which derives from the adoption of the maximum-likelihood principle (Section 3.8) for supervised classification². Consider a classification problem where the output variable \mathbf{y} takes values in the set $\{c_1, \dots, c_K\}$ and $\text{Prob}\{\mathbf{y} = c_j | x\}, j = 1, \dots, K$ is the conditional probability. Given a training dataset and a set of parametric models $\hat{P}_j(x, \alpha), j = 1, \dots, K$ of the conditional distribution, the classification problem boils down to the minimisation of the quantity

$$J(\alpha) = - \sum_{i=1}^N \log \hat{P}_{y_i}(x_i, \alpha) \quad (6.6.5)$$

Note that the models $\hat{P}_j(x, \alpha), j = 1, \dots, K$ must satisfy two important constraints:

²The use of the term cross-entropy (introduced in probabilistic terms in (2.11.48)) is due to the fact that the cross-entropy between a density p and its empirical version \hat{p} based on an observed dataset D_N amounts to the negative log-likelihood $-\sum_{i=1}^N \log p(z_i)$ of the dataset itself [128].

1. $\hat{P}_j(x, \alpha) > 0$
2. $\sum_{j=1}^K \hat{P}_j(x, \alpha) = 1$

In the case of 0/1 binary classification problem, the cross-entropy is written as

$$J(\alpha) = - \sum_{i=1}^N [y_i \log \hat{P}_1(x_i, \alpha) + (1 - y_i) \log(1 - \hat{P}_1(x_i, \alpha))] \quad (6.6.6)$$

where $\hat{P}_1(x_i, \alpha)$ is the estimation of the conditional probability of the class $y = 1$.

Since this chapter will focus mainly 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

$$h(x, \alpha) = \alpha^T x$$

is obtained by minimising a quadratic function $J(\alpha)$ by the well-known *linear least-squares method*. In Chapter 7 we will see in detail the linear least-squares minimisation. Here we just report that, in the case of non-singularity of the matrix $(X^T X)$, $J(\alpha)$ has a single global minimum in

$$\alpha_N = (X^T X)^{-1} X^T Y \quad (6.6.7)$$

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 α , and there may exist many minima, all of which satisfy

$$\nabla J = 0 \quad (6.6.8)$$

where ∇ denotes the gradient of J in parameter space. We will define as *stationary points* all the points which satisfy condition (6.6.8). 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. For more details on multivariate optimisation, we refer the reader to [2].

We will consider *iterative* algorithms, which involve a search through the parameter space consisting of a succession of steps of the form

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} + \Delta\alpha^{(\tau)} \quad (6.6.9)$$

where τ 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 applications and that it is not possible, or at least fair, to recommend a single universal optimisation 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 α , increasing the rate of convergence of the iterative algorithm. Some examples of gradient-based methods are reported in the following sections. Those methods require the derivatives of the cost function with respect to the model parameters. Such computation is not always easy in complex nonlinear mappings, but it has been recently facilitated by the appearance of *automatic differentiation* functionalities [18], like the ones made available by libraries like TensorFlow or PyTorch.

6.6.2.4 Gradient descent

It is the simplest of the gradient-based optimisation algorithms, also known as the *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 τ^{th} step, the estimate is updated by moving a short distance in the direction of the negative gradient evaluated in $\alpha^{(\tau)}$:

$$\Delta\alpha^{(\tau)} = -\mu \nabla J(\alpha^{(\tau)}) \quad (6.6.10)$$

where μ is called the *learning rate*. The updates are repeatedly executed until convergence, i.e. when further improvements are considered to be too small to be useful.

The gradient descent method is known to be a very inefficient procedure. One drawback is the need for a suitable value of the learning rate μ . In fact, a decrease of the cost function is guaranteed by (6.6.10) only for learning rates of infinitesimal size: 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 (6.6.8) is satisfied. Too small learning rates may considerably delay the convergence, while too large rates might result in numerical overflows.

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.

Example of gradient-based univariate optimisation

Let us consider the univariate function

$$J(\alpha) = \alpha^2 - 2\alpha + 3$$

visualised in Figure 6.1. By running the script `Learning/optim.R` you can visualise the gradient search of the minimum of the function. Note that the gradient is obtained by computing analytically the derivative

$$J'(\alpha) = 2\alpha - 2$$

We invite the reader to assess the impact of the learning rate μ on the convergence of the minimisation process.

The function

$$J(\alpha) = \alpha^4/4 - \alpha''/3 - \alpha^2 + 2$$

with two local minima is shown in Function 6.2 and minimised in the script `Learning/optim2.R`. We invite the reader to assess the impact of the initial value $\alpha^{(0)}$ of the solution on the result of the minimisation process.

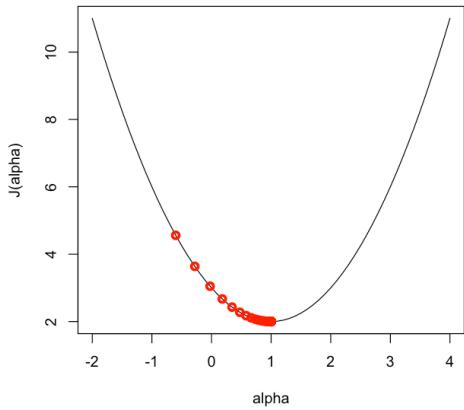


Figure 6.1: Gradient-based minimisation of the function $J(\alpha) = \alpha^2 - 2\alpha + 3$ with a single global minimum.

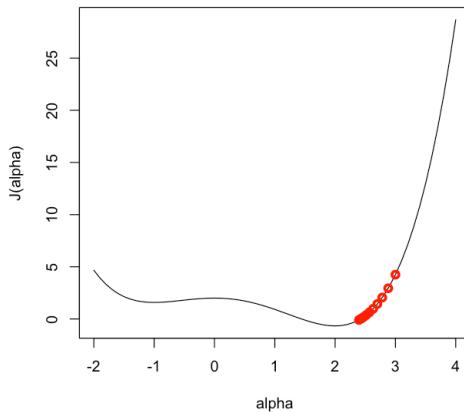


Figure 6.2: Gradient-based minimisation of the function $J(\alpha) = \alpha^4/4 - \alpha^3/3 - \alpha^2 + 2$.

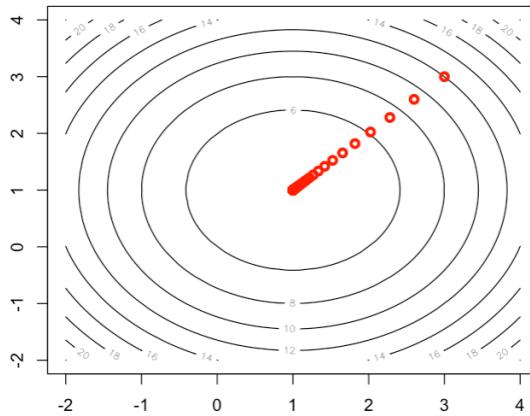


Figure 6.3: Contour plot: gradient-based minimisation of the bivariate function $J(\alpha) = \alpha_1^2 + \alpha_2^2 - 2\alpha_1 - 2\alpha_2 + 6$ with a single global minimum.

Example of gradient-based bivariate optimisation

Let us consider the bivariate function

$$J(\alpha) = \alpha_1^2 + \alpha_2^2 - 2\alpha_1 - 2\alpha_2 + 6$$

whose contour plot is visualised in Figure 6.3. By running the script `Learning/optim2D.R` you can visualise the gradient search of the minimum of the function. Note that the gradient is obtained by computing analytically the gradient vector

$$\nabla J(\alpha) = [2\alpha_1 - 2, 2\alpha_2 - 2]^T$$

We invite the reader to assess the impact of the learning rate μ on the convergence of the minimisation process.

The contour plot of the function

$$J(\alpha) = \frac{\alpha_1^4 + \alpha_2^4}{4} - \frac{\alpha_1^3 + \alpha_2^3}{3} - \alpha_1^2 - \alpha_2^2 + 4$$

with three local minima is shown in Function 6.4 and minimised in the script `Learning/optim2D2.R`. We invite the reader to assess the impact of the initial value $\alpha^{(0)}$ of the solution on the result of the minimisation process.

•

In alternative to the simplest gradient descent, there are many iterative methods in the literature, as the *momentum-based* method [142], the *enhanced gradient descent* method [173] and the *conjugate gradients* techniques [144], which make implicit use of second-order derivatives of the error function.

In the following section, we present instead a class of algorithms that make explicit use of second-order information.

6.6.2.5 The Newton method

The Newton's method is a well-known example in optimisation literature. It is an iterative algorithm which uses at the τ^{th} step a local quadratic approximation in

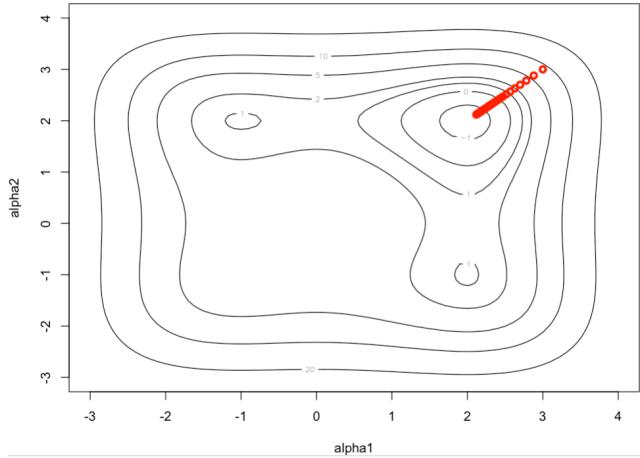


Figure 6.4: Contour plot: gradient-based minimisation of the function $J(\alpha) = \frac{\alpha_1^4 + \alpha_2^4}{4} - \frac{\alpha_1^3 + \alpha_2^3}{3} - \alpha_1^2 - \alpha_2^2 + 4$.

the neighbourhood of $\alpha^{(\tau)}$

$$\hat{J}(\alpha) = J\left(\alpha^{(\tau)}\right) + \left(\alpha - \alpha^{(\tau)}\right)^T \nabla J\left(\alpha^{(\tau)}\right) + \frac{1}{2} \left(\alpha - \alpha^{(\tau)}\right)^T H(\alpha^{(\tau)}) \left(\alpha - \alpha^{(\tau)}\right) \quad (6.6.11)$$

where $H(\alpha^{(\tau)})$ is the Hessian matrix of $J(\alpha)$ computed in $\alpha^{(\tau)}$ and

$$H(\alpha) = \begin{bmatrix} \frac{\partial^2 J}{\partial \alpha_1^2} & \frac{\partial^2 J}{\partial \alpha_1 \partial \alpha_2} & \cdots & \frac{\partial^2 J}{\partial \alpha_1 \partial \alpha_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial \alpha_p \partial \alpha_1} & \frac{\partial^2 J}{\partial \alpha_p \partial \alpha_2} & \cdots & \frac{\partial^2 J}{\partial \alpha_p^2} \end{bmatrix}$$

is a $[p, p]$ square matrix of second-order partial derivatives if $\alpha \in \mathbb{R}^p$.

The minimum of (6.6.11) satisfies

$$\alpha^{min} = \alpha^{(\tau)} - H^{-1}(\alpha^{(\tau)}) \nabla J\left(\alpha^{(\tau)}\right) \quad (6.6.12)$$

where the vector $H^{-1}(\alpha^{(\tau)}) \nabla J\left(\alpha^{(\tau)}\right)$ is denoted as the Newton direction or the Newton step and forms the basis for the iterative strategy

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - H^{-1} \nabla J\left(\alpha^{(\tau)}\right) \quad (6.6.13)$$

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 (6.6.12) 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 [121].

6.6.2.6 The Levenberg-Marquardt algorithm

This algorithm is designed specifically for minimising a sum-of-squares error function

$$J(\alpha) = \frac{1}{2} \sum_{i=1}^N L(y_i, h(x_i, \alpha)) = \frac{1}{2} \sum_{i=1}^N e_i^2 = \frac{1}{2} \|e\|^2 \quad (6.6.14)$$

where e_i is the error for the i^{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

$$\alpha^{(\tau)} \rightarrow \alpha^{(\tau+1)} \quad (6.6.15)$$

If the quantity (6.6.15) is sufficiently small, the error vector e can be expanded in a first-order Taylor series form:

$$e(\alpha^{(\tau+1)}) = e(\alpha^{(\tau)}) + E(\alpha^{(\tau+1)} - \alpha^{(\tau)}) \quad (6.6.16)$$

where the generic element of the matrix E is in the form

$$E_{ij} = \frac{\partial e_i}{\partial \alpha_j} \quad (6.6.17)$$

and α_j is the j^{th} element of the vector α . The error function can then be approximated by

$$J(\alpha^{(\tau+1)}) = \frac{1}{2} \|e(\alpha^{(\tau)}) + E(\alpha^{(\tau+1)} - \alpha^{(\tau)})\|^2 \quad (6.6.18)$$

If we minimise with respect to $\alpha^{(\tau+1)}$ we obtain:

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - (E^T E)^{-1} E^T e(\alpha^{(\tau)}) \quad (6.6.19)$$

where $(E^T E)^{-1} E^T$ is the pseudo-inverse of the matrix E . For the sum-of-squares error function (6.6.14), the elements of the Hessian take the form

$$H_{jk} = \frac{\partial^2 E}{\partial \alpha_j \partial \alpha_k} = \sum_{i=1}^N \left\{ \frac{\partial e_i}{\partial \alpha_j} \frac{\partial e_i}{\partial \alpha_k} + e_i \frac{\partial^2 e_i}{\partial \alpha_j \partial \alpha_k} \right\} \quad (6.6.20)$$

Neglecting the second term, the Hessian can be written in the form

$$H \simeq E^T E \quad (6.6.21)$$

This relation is exact in the case of linear models, while in the case of nonlinearities it represents an approximation that holds exactly only at the global minimum of the function [25]. The update formula (6.6.19) 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 (6.6.19), 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

$$J_{lm} = \frac{1}{2} \|e(\alpha^{(\tau)}) + E(\alpha^{(\tau+1)} - \alpha^{(\tau)})\|^2 + \lambda \|\alpha^{(\tau+1)} - \alpha^{(\tau)}\|^2 \quad (6.6.22)$$

where λ is a parameter that governs the step size. The minimisation of the error function (6.6.22) ensures, at the same time, the minimisation of the sum-of-square cost and a small step size. Minimising (6.6.22) with respect to $\alpha^{(\tau+1)}$ we obtain

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - (E^T E + \lambda I)^{-1} E^T e(\alpha^{(\tau)}) \quad (6.6.23)$$

where I is the unit matrix. For very small values of λ we have the Newton formula, while for large values of λ we recover the standard gradient descent.

A common approach for setting λ is to begin with some arbitrary low value (e.g. $\lambda = 0.1$) and at each step (6.6.23) check the change in J . If J decreases, the new parameter is retained, λ is decreased (e.g. by a factor of 10), and the process is repeated. Otherwise, if J increased after the step (6.6.23), the old parameter is restored, λ decreased, and a new step performed. The procedure is iterated until a decrease in J is obtained [25].

6.6.3 Online gradient-based algorithms

The algorithms above are called *batch* since they compute the gradient of the quantity (6.6.2) over the entire training set. In the case of very large datasets or sequential settings, this procedure is not recommended since it requires the storage of the entire dataset. For this reason, online modification of the batch algorithms have been proposed in the literature. The idea consists of replacing the gradient computed on the entire training set with a gradient computed on the basis of a single data point

$$\alpha^{(\tau+1)} = \alpha^{(\tau)} - \mu^{(\tau)} \nabla_{\alpha} J^{(\tau)}(\alpha^{(\tau)}) \quad (6.6.24)$$

where

$$J^{(\tau)}(\alpha^{(\tau)}) = (y_{\tau} - h(x_{\tau}, \alpha^{(\tau)}))^2$$

and $z_{\tau} = \langle y_{\tau}, x_{\tau} \rangle$ the input/output observation at the τ th instant.

The underlying assumption is that the training error obtained by replacing the average with a single term will not perturb the average behaviour of the algorithm. Note also that the dynamics of $\mu^{(\tau)}$ plays an important role in the convergence.

This algorithm can be easily used in an adaptive online setting where no training set needs to be stored, and observations are processed immediately to improve performance. A linear version of the iteration (6.6.24) is the Recursive Least Squares regression algorithm presented in Section 7.1.20. Note also that the earliest machine learning algorithms were based on sequential gradient-based minimisation. Well-known examples are Adaline and LMS [177].

6.6.4 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 converge to the nearest local minimum. As a consequence, the way a deterministic method is initialised 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 the literature. We will limit to cite the most common solutions:

Random search methods. They are iterative methods that are primarily used for continuous optimisation 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 customised 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 optimisation process could take a prohibitively long time.

Genetic algorithms. They are derivative-free stochastic optimisation methods based loosely on the concepts of natural selection and evolutionary processes [83]. Important properties are the strong parallelism and the possibility to be applied to both continuous and discrete optimisation 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 repeatedly evolved 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 optimisation problems. In Simulated Annealing (SA), the value of the cost function $J(\alpha)$ to be minimised is put in analogy to the energy in a thermodynamic system at a certain temperature T [110]. 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 a 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 a 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.

R script

The script `Learning/grad.R` compares four ways of fitting the parameters of a linear model:

1. least-squares
2. random search
3. gradient-based search
4. Levenberg-Marquardt

It is interesting to compare the number of steps required to attain a solution close to the optimal one: one for least-squares, more than one thousands for the random search, more than twenty for gradient-based and two for the Levenberg-Marquardt method.

•

6.7 Regularisation

Parameter identification relies on Empirical Risk Minimisation to return an estimator in supervised learning problems. In Section 5.6, we stressed that the accuracy of such an estimator depends on the bias/variance trade-off, which is typically controlled by capacity-related hyper-parameters. Section 3.11 showed another important strategy, called *regularisation*, to control the bias/variance trade-off of an estimator. Its adoption in supervised learning consists in restricting the set of possible solutions by transforming the unconstrained problem (6.6.3) into a constrained one. An example of constrained minimisation is

$$\alpha_N = \arg \min_{\alpha \in \Lambda} J(\alpha) + \lambda \|\alpha\| \quad (6.7.25)$$

where $\lambda > 0$ is the regularisation parameter. By adding the squared norm term, solutions with large values of components of α are penalised unless they play a major role in the $J(\alpha)$ term. An alternative version is

$$\alpha_N = \arg \min_{\alpha \in \Lambda} J(\alpha) + \lambda S(h(\cdot, \alpha)) \quad (6.7.26)$$

where the term S penalises non-smooth wiggling hypothesis functions. An example is

$$S(\alpha) = \int (h''(x, \alpha))^2 dx$$

where the integral of the second derivative is a measure of lack of smoothness.

Regularisation is a well-known strategy in numerical analysis and optimisation to avoid or limit the ill-conditioning of the solution. In estimation, regularisation is an additional way to control the variance of the estimator resulting from the optimisation procedure. This is particularly effective in learning problems with a number of observations comparable or even smaller than the input dimension.

Note, however, that variance reduction occurs at the cost of both increased bias and additional complexity of the optimisation problem. For instance, according to the nature of the regularisation and the nonlinearity of the hypothesis function we could lose some interesting properties like the closed-form of the solution. In Chapter 10, we will show some examples of regularisation to address the curse of dimensionality problem.

6.8 Structural identification

Once a class of models Λ is given, the identification procedure, described above, returns a model $h(\cdot, \cdot, \alpha_N)$ defined by the set of parameters $\alpha_N \in \Lambda$.

The choice of an appropriate class or *model structure* [119] is, however, the most crucial aspect for a successful modelling application. The procedure of selecting the best model structure is called *structural identification*. The structure of a model is made of a series of characteristics that influence the generalisation 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 characteristics 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, and so on. Those quantities are usually referred to as *hyperparameters*, i.e. free parameters that are not determined by the parametric identification algorithm, but rather given as inputs to the structural identification procedure.

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 6.8.1.

The second issue concerns the important problem of *model validation*, and will be extensively dealt with in Section 6.8.2.

Once models have been generated and validated, the last step is the *model selection*, which will be discussed in Section 6.8.3.

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.

6.8.1 Model generation

The 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 [124] 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 of enumerating all the possible configurations in terms of the number of neurons.

The exhaustive algorithm runs in a time that is generally unacceptably slow for complex architectures 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 treat 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 generalisation 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 issue 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 [19].

6.8.2 Validation

The output of the model generation procedure is a set of model structures Λ_s , $s = 1, \dots, S$. 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 Minimisation principle.

Now, the prediction quality of each one of the model structures Λ_s , $s = 1, \dots, S$, 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.

6.8.2.1 Testing

An obvious way to assess the quality of a learned model is by using a *testing sequence*

$$D_{ts} = (\langle x_{N+1}, y_{N+1} \rangle, \dots, \langle x_{N+N_{ts}}, y_{N+N_{ts}} \rangle) \quad (6.8.27)$$

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

$$\hat{R}_{ts}(\alpha_N^s) = \frac{1}{N_{ts}} \sum_{j=N+1}^{N+N_{ts}} (y_j - h(x_j, \alpha_N^s))^2 \quad (6.8.28)$$

This estimator is clearly unbiased in the sense that

$$E_{\mathbf{D}_{ts}}[\hat{\mathbf{R}}_{ts}(\alpha_N^s)] = R(\alpha_N^s) \quad (6.8.29)$$

When the number of available examples is sufficiently high, the testing technique is an effective validation technique at a 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.

6.8.2.2 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 the 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 larger the number of points used for test set, the higher the bias of the estimate $\alpha_{N_{tr}}$; at the same time, fewer test points implies a larger confidence interval of the estimate of the generalisation error.

6.8.2.3 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) [98].

Consider a learning problem with a training set of size N .

In l -fold cross-validation the available points are randomly divided into l mutually exclusive test partitions of approximately equal size. The examples not found in each test partition are independently used for selecting the hypothesis, which will be tested on the partition itself (Fig. 5.16). 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 dataset D_N , an hypothesis is generated using $N - 1$ observations and tested on the single remaining one. In leave-one-out, cross-validation is repeated $l = N$ times, each data point is used as a test case, and each time nearly all the examples 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 computationally quite expensive. 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 7.1.17).

6.8.2.4 Bootstrap in practice

Bootstrap is a resampling technique that samples the training set with replacement to return a nonparametric estimate of the 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}_b in (5.9.57), samples with replacement from the original training set B bootstrap training sets, each consisting of N cases. The cases not found in the training group form the test groups. The average of error rate on the B test groups is the E0 estimate of the generalisation error.

The rationale for the E632 technique is given by Efron [64]. He argues that while the resubstitution error R_{emp} is the error rate for patterns that 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_{emp} and E0 should involve patterns at the “right” distance from the training set in estimating the error rate:

$$\hat{G}_{\text{E632}} = 0.368 * R_{\text{emp}} + 0.632 * \hat{G}_{\text{bs}} \quad (6.8.30)$$

where the quantity \hat{G}_{bs} is defined in (5.9.57). The choice of B is not critical as long as it exceeds 100. Efron [64] suggests, however, that B need not be greater than 200.

There a lot of experimental results on comparison between cross-validation and bootstrap methods for assessing models [102], [111]. In general terms, only some guidelines can be given to the practitioner [176]:

- 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 sets ($N < 50$), in addition to the leave-one-out estimator, the \hat{G}_{E632} and the \hat{G}_{boot} estimates may be useful measures.

6.8.2.5 Complexity based criteria

In conventional statistics, various criteria have been developed, often in the context of linear models, for assessing the generalisation performance of the learned hypothesis without the use of further validation data. Such criteria aim to understand the relationship between the generalisation performance and the training error. Generally, they take the form of a prediction error, which consists of the sum of two terms

$$\hat{G}_{\text{PE}} = R_{\text{emp}} + \text{complexity term} \quad (6.8.31)$$

where the complexity (or capacity) term represents a penalty that 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 (6.8.31) 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 7.1.16.2 and [6])

$$\text{FPE} = \widehat{\text{MISE}}_{\text{emp}} \frac{1 + p/N}{1 - p/N} \quad (6.8.32)$$

with $p = n + 1$,

2. the Predicted Squared Error (PSE) (see Section 7.1.16.2)

$$\text{PSE} = \widehat{\text{MISE}}_{\text{emp}} + 2\hat{\sigma}_{\mathbf{w}}^2 \frac{p}{N} \quad (6.8.33)$$

where $\hat{\sigma}_{\mathbf{w}}^2$ is an estimate of the noise variance. This quantity is also known as the Mallows' C_p statistics [123]

3. the Generalised Cross-Validation (GCV) [50]

$$\text{GCV} = \widehat{\text{MISE}}_{\text{emp}} \frac{1}{(1 - \frac{p}{N})^2} \quad (6.8.34)$$

A comparative analysis of these different measures is reported in [15].

These estimates are computed by assuming a linear model underlying the data. In neural network literature another well-known form of complexity-based criterion is the *weight decay technique*

$$U(\lambda, \alpha, D_N) = \sum_{i=1}^N (y_i - h(x_i, \alpha))^2 + \lambda g(\alpha) \quad (6.8.35)$$

where the second term penalises either small, medium or large weights of the neurons depending on the form of $g(\cdot)$ and λ is called the *regularisation* constant. Two common examples of weight decay functions are the *ridge regression* form $g(\alpha) = \alpha^T \alpha$ which penalises large weights, and the Rumelhart form $g(\alpha) = \frac{\alpha^T \alpha}{\alpha_0 + \alpha^T \alpha}$ which penalises weights of intermediate values near α_0 .

Several roughness penalties like

$$\int [h''(x)]^2 dx$$

have been proposed too. Their aim is penalising hypothesis functions that vary too rapidly by controlling large values of the second derivative of h .

Other state-of-the-art methods for model validation and selection are the Akaike Infomation Criterion (AIC) [42], the Bayes Information Criterion (BIC) [66] and the *Minimum Description Length* (MDL) principle [148]. The AIC criterion returns the model maximising

$$\log L_N(\hat{\theta}_{\text{ml}}) - p$$

a penalised parametric log-likelihood cost function balancing good fit (high log-likelihood) and complexity (length of the parameter vector). The BIC criterion maximises a similar penalised term

$$\log L_N(\hat{\theta}_{\text{ml}}) - p \frac{\log N}{2}$$

where $\hat{\theta}_{\text{ml}}$ is defined in (3.8.23). The MDL method proposes to choose the model which induces the shortest description for the data available. Rissanen and Barron [15] have each shown a qualitative similarity between this principle and the complexity based approaches. For further details refer to the cited works.

6.8.2.6 A comparison of 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 summarise some practical arguments on behalf of one or the other method. The benefits of a CISM method are:

- All the assumptions of prior knowledge of 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 [161], under the assumptions that the real model α^* is contained in the class of hypothesis Λ 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.

6.8.2.7 Residual diagnostic

Although the main focus of machine learning validation is estimating the magnitude of the generalisation error, in statistical modeling it is common to assess the quality of a model also with criteria related to the error distribution. This is the case of residual diagnostic measures. The idea is simple: if we model the input/output dependence with the relation (6.5.1) and we use a hypothesis h to approximate f , the quality of the learned hypothesis is related to the white noise nature of the residuals. In other terms we expect that the residuals

$$w_i = y_i - \hat{y}_i = y_i - h(x_i, \alpha_N)$$

are i.i.d. realisation of a noise variable w with expected value $E[w] = 0$ and constant variance. To check this hypothesis, several residual diagnostic tests have been proposed in litterature [131]. The simplest check consists in plotting the residuals w_i vs. the predicted values \hat{y}_i . In case of reliable fit, a random pattern

around zero is expected. Alternative patterns (e.g. curvilinear trends, heterogenous variance) would instead provide evidence of model limitations. More sophisticated tests consist on regressing the residuals vs. the original inputs to check whether some residual signal is still present. In this case a prediction accuracy better than random would warn against the goodness of the learned model.

Though the use of diagnostics is warmly recommended and should be taken into account all along the modeling process, in what follows we will limit to consider model selection strategies which depend on the generalisation error.

6.8.3 Model selection criteria

Model selection concerns the final choice of the model structure in the set that has been proposed by the 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 6.8.2. We distinguish between two possible quantitative approaches: the *winner-takes-all* and the *combination of estimators* approach.

6.8.3.1 The winner-takes-all approach

This approach chooses the model structure that minimises the generalisation error according to one of the criteria described in Section 6.8.2.

Consider a set of candidate model structures Λ_s , $s = 1, \dots, S$, and an associated measure $\hat{G}(\Lambda_s)$ which quantifies the generalisation error.

The winner-takes-all method simply picks the structure

$$\bar{s} = \arg \min \hat{G}(\Lambda_s) \quad (6.8.36)$$

that minimises the generalisation error. The model which is returned as final outcome of the learning process is then $h(\cdot; \alpha_{\bar{N}}^{\bar{s}})$.

From a practitioner perspective, it may be useful to make explicit the entire *winner-takes-all* procedure in terms of pseudo-code . Here below you will find a compact pseudo-code detailing the structural, parametric, validation and selection steps in the case of a leave-one-out assessment .

1. for $s = 1, \dots, S$: (Structural loop)
 - for $j = 1, \dots, N$
 - (a) Inner parametric identification (for l-o-o):

$$\alpha_{N-1}^s = \arg \min_{\alpha \in \Lambda_s} \sum_{i=1:N, i \neq j} (y_i - h(x_i, \alpha))^2$$

- (b) $e_j = y_j - h(x_j, \alpha_{N-1}^s)$
 - $\widehat{\text{MISE}}_{\text{LOO}}(s) = \frac{1}{N} \sum_{j=1}^N e_j^2$
2. Model selection: $\bar{s} = \arg \min_{s=1, \dots, S} \widehat{\text{MISE}}_{\text{LOO}}(s)$
 3. Final parametric identification:

$$\alpha_{\bar{N}}^{\bar{s}} = \arg \min_{\alpha \in \Lambda_{\bar{s}}} \sum_{i=1}^N (y_i - h(x_i, \alpha))^2$$

4. The output prediction model is $h(\cdot, \alpha_N^{\bar{s}})$

The winner-takes-all strategy illustrated by (6.8.36) relies on the comparison of the sample estimate of the generalisation error (typically the average squared prediction error) of different model structures. Given the variability of such estimate and the need of comparing several model candidates (e.g. in wrapper feature selection), more sophisticated and robust statistical techniques have been proposed in model selection literature, notably the racing approach [124, 34] and techniques based on bandit algorithms [35].

6.8.3.2 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 [141] 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 (Figure 5.15) 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 [17] 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 (3.10.35), (3.10.36) and (3.10.37) we have that the combined model is

$$h(\cdot) = \sum_{j=1}^m w_j h(\cdot, \alpha_j) = \frac{\sum_{j=1}^m \frac{1}{\hat{v}_j} h(\cdot, \alpha_j)}{\sum_{j=1}^m \frac{1}{\hat{v}_j}} \quad (6.8.37)$$

where \hat{v}_j is an estimation of the variance of $h(\cdot, \alpha_j)$. This is an example of the *generalised ensemble method* (GEM) [141].

More advanced applications of the combination principle to supervised learning will be discussed in Chapter 9.

6.9 Partition of dataset in training, validation and test

The main challenge of machine learning consists of using a finite size dataset for i) learning several predictors, ii) assessing them, iii) selecting the most promising one and finally iv) returning it together with a reliable estimate of its generalisation error.

Section 5.9 discussed the need for avoiding correlation between training and validation examples. While the training set is used for parametric identification, a non-overlapping portion of the dataset (validation set) should be used to estimate the generalisation error of model candidates.

The use of validation (or cross-validation) does not prevent, however, a risk of overfitting inherent to the winner-take-all model selection. If we take the minimum generalisation error $\hat{G}(\Lambda_{\bar{s}})$ in (6.8.36) as the generalisation error of the winning model, we have an optimistic estimation again. This is known as *selection bias*, i.e. the bias that occurs when we make a selection in a stochastic setting and due to the fact the expectation of minimum is lower than the minimum of expectations (Appendix D.13).

A *nested cross-validation* strategy [43] is recommended to avoid such bias. If we have enough observations (i.e. large N), the strategy consists in randomly partitioning (e.g. 50%, 25%, 25%) the labelled dataset into three parts: a training set, a validation set, and a test set. The test portion is supposed to be used for the unbiased assessment of the generalisation error of the model \bar{s} in (6.8.36). It is important to use only this set to assess the generalisation accuracy of the chosen model. For this reason, the test set should be carefully made inaccessible to the learning process (locked and ideally forgotten) and considered only at the very end of the data analysis. Any other use of the test-set during the analysis (e.g. before the final assessment) would “contaminate” the procedure and make it irreversibly biased.

Selection bias

A Monte Carlo illustration of the selection bias effect in a univariate regression task is proposed in the R script `Learning/selectionbias.R`. The script estimates the generalisation errors of a constant model (h_1), a linear model (h_2) and a third model which is nothing more than the winner-takes-all of the two in terms of leave-one-out validation. It appears that the winner-takes/all model is not better than the best between h_1 and h_2 : in other terms it has a generalisation error larger than the minimum between h_1 and h_2 .

•

6.10 Evaluation of a regression model

Let us consider a test set of size N_{ts} where $Y_{ts} = \{y_1, \dots, y_{N_{ts}}\}$ is the target and $\hat{Y} = \{\hat{y}_1, \dots, \hat{y}_{N_{ts}}\}$ is the prediction returned by the learner. The canonical way to assess a regression model by using a testing set is to measure the mean-squared-error (6.8.28) (MSE):

$$\text{MSE} = \frac{\sum_{i=1}^{N_{ts}} (y_i - \hat{y}_i)^2}{N_{ts}}$$

Let us suppose that the test of a learning algorithm returns a mean-squared-error of 0.4. Is that good or bad? Is that impressive and/or convincing? How may we have a rapid and intuitive measure of the quality of a regression model?

A recommended way is to compare the learner to a baseline, e.g. the simplest (or naive) predictor we could design. This is the rationale of the Normalised Mean-Squared-Error measure which normalises the accuracy of the learner with respect to the accuracy of the average predictor, i.e. the simplest predictor we could learn from data. Then

$$\text{NMSE} = \frac{\sum_{i=1}^{N_{ts}} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N_{ts}} (y_i - \bar{y})^2} \quad (6.10.38)$$

where

$$\bar{y} = \frac{\sum_{i=1}^{N_{ts}} y_i}{N_{ts}} \quad (6.10.39)$$

is the prediction returned by an average predictor. NMSE is then the ratio between the MSE of the learner and the MSE of the baseline naive predictor (6.10.39).

As for the MSE, the lower the NMSE the better. At the same time, we should target a NMSE (significantly) lower than one, if we wish to claim that the complex learning procedure is effective. NMSE values close to (yet smaller than) one are either indicators of a bad learning design or, more probably, of a high noise to signal ratio (e.g. large σ_w^2 in (6.5.1)) which makes any learning effort ineffective.

Our recommendation is always to measure the NMSE of a regression model before making too enthusiastic claims about the success of the learning procedure. A very small NMSE could be irrelevant if not significantly smaller than what we could obtain by a simple naive predictor.

Another common way to assess the MSE of a predictor is to normalise it with respect to the MSE of the same learning algorithm, yet trained on a randomly shuffled version of the training set. For instance it is enough to shuffle the training target to cancel any dependency between inputs and outputs. Again in this case we expect that the NMSE is much lower than one. Otherwise any claim that our prediction is better than a random one would be unfounded.

6.11 Evaluation of a binary classifier

The most popular measure of performance is *error rate* or *misclassification rate*, i.e. the proportion of test examples misclassified by the rule. However, misclassification error is not necessarily the most appropriate criterion in real settings since it implicitly assumes that the costs of different types of misclassification are equal. When there are only a few or a moderate number of classes, the confusion matrix is the most complete way of summarising the classifier performance. In the following, we will focus on evaluating a binary classifier.

Suppose to use the classifier to make N test classifications and that among the values to be predicted there are N_P examples of class 1 and N_N examples of class 0. The confusion matrix is

	Negative (0)	Positive (1)	
Classified as negative	T_N	F_N	$N_{\hat{N}}$
Classified as positive	F_P	T_P	$N_{\hat{P}}$
	N_N	N_P	N

where F_P is the number of False Positives, F_N is the number of False Negatives and $N_{\hat{P}}$ is the number of examples that are classed as positive (e.g. number of alerts in a detection problem). The confusion matrix contains all the relevant information to assess the generalisation capability of a binary classifier. From its values it is possible to derive a number of commonly used error rates or measures of accuracy. For instance, the misclassification error rate is

$$\text{ER} = \frac{F_P + F_N}{N} \quad (6.11.40)$$

6.11.1 Balanced Error Rate

In a setting where the two classes are not balanced the misclassification error rate (6.11.40) can lead to a too optimistic interpretation of the rate of success.

For instance, if $N_P = 90$ and $N_N = 10$, a naive classifier returning always the positive class would have a misclassification $ER = 0.1$ since $F_N = 0$ and $F_P = 10$. This low value of misclassification gives a false sense of accuracy since humans tend to associate a 50% error to random classifiers. This is true in balanced settings while in an unbalanced setting (as the one above) this generalisation performance may be obtained with a trivial classifier making no use of the input information.

In these cases, it is preferable to adopt the balanced error rate which is the average of the errors on each class:

$$\text{BER} = \frac{1}{2} \left(\frac{F_P}{T_N + F_P} + \frac{F_N}{F_N + T_P} \right)$$

Note that in the example above $\text{BER} = 0.5$, normalising the misclassification error rate to a value correctly interpretable by humans.

6.11.2 Specificity and sensitivity

In many research works on classification, it is common usage to assess the classifier in terms of sensitivity and specificity.

Sensitivity is a synonymous of True Positive Rate (TPR)

$$SE = TPR = \frac{T_P}{T_P + F_N} = \frac{T_P}{N_P} = \frac{N_P - F_N}{N_P} = 1 - \frac{F_N}{N_P}, \quad 0 \leq SE \leq 1 \quad (6.11.41)$$

It is a quantity to be maximised, and it increases by reducing the number of false negatives. Note that it is also often called the *recall* in information retrieval.

Specificity stands for the True Negative Rate (TNR)

$$SP = TNR = \frac{T_N}{F_P + T_N} = \frac{T_N}{N_N} = \frac{N_N - F_P}{N_N} = 1 - \frac{F_P}{N_N}, \quad 0 \leq SP \leq 1$$

It is a quantity to be maximised and it increases by reducing the number of false positive.

In other terms, sensitivity is the proportion of positive examples classified as positive while specificity is the proportion of negative examples classified as negative.

There exists a trade-off between these two quantities. This is the reason why both quantities have to be calculated to have a thorough assessment of the classifier accuracy. In fact, it is trivial to maximise one of those quantities at the detriment of the other.

For instance, for a naive classifier who always returns 0 we have $N_{\hat{P}} = 0, N_{\hat{N}} = N, F_P = 0, T_N = N_N$. This means that a naive classifier may attain maximal specificity ($SP = 1$) but at the cost of minimal sensitivity ($SE = 0$).

Analogously in the case of a naive classifier who always returns 1 we have $N_{\hat{P}} = N, N_{\hat{N}} = 0, F_N = 0, T_P = N_P$, i.e. maximal sensitivity ($SE = 1$) but null specificity ($SP = 0$).

6.11.3 Additional assessment quantities

Other commonly used quantities which can be derived by the confusion matrix are

- False Positive Rate:

$$FPR = 1 - SP = 1 - \frac{T_N}{F_P + T_N} = \frac{F_P}{F_P + T_N} = \frac{F_P}{N_N}, \quad 0 \leq FPR \leq 1$$

It decreases by reducing the number of false positive.

- False Negative Rate:

$$FNR = 1 - SE = 1 - \frac{T_P}{T_P + F_N} = \frac{F_N}{T_P + F_N} = \frac{F_N}{N_P} \quad 0 \leq FNR \leq 1$$

It decreases by reducing the number of false negatives.

- Positive Predictive value: the ratio (to be maximised)

$$PPV = \frac{T_P}{T_P + F_P} = \frac{T_P}{N_{\hat{P}}}, \quad 0 \leq PPV \leq 1 \quad (6.11.42)$$

This quantity is also called *precision* in information retrieval.

- Negative Predictive value: the ratio (to be maximised)

$$PNV = \frac{T_N}{T_N + F_N} = \frac{T_N}{N_{\hat{N}}}, \quad 0 \leq PNV \leq 1$$

- False Discovery Rate: the ratio (to be minimised)

$$FDR = \frac{F_P}{T_P + F_P} = \frac{F_P}{N_{\hat{P}}} = 1 - PPV, \quad 0 \leq FDR \leq 1$$

6.11.4 Random classifier

In order to assess the quality of a generic classifier it is better to derive first the confusion matrix of a (naive) random classifier and then the related accuracy measures. Since in a random classifier $\text{Prob}\{\hat{\mathbf{y}} = 1|\mathbf{y} = 1\} = \text{Prob}\{\hat{\mathbf{y}} = 1|\mathbf{y} = 0\} = \text{Prob}\{\mathbf{y} = 1\} \approx N_P/N$ and $\text{Prob}\{\hat{\mathbf{y}} = 0|\mathbf{y} = 1\} = \text{Prob}\{\hat{\mathbf{y}} = 0|\mathbf{y} = 0\} = \text{Prob}\{\mathbf{y} = 0\} \approx N_N/N$, the confusion matrix is

	Negative (0)	Positive (1)	
Classified as negative	$TN = \frac{N_N}{N} N_{\hat{N}}$	$FN = \frac{N_P}{N} N_{\hat{N}}$	$N_{\hat{N}}$
Classified as positive	$FP = \frac{N_N}{N} N_{\hat{P}}$	$TP = \frac{N_P}{N} N_{\hat{P}}$	$N_{\hat{P}}$
	N_N	N_P	N

It follows that in a random classifier

$$\text{BER} = 0.5 \quad (6.11.43)$$

$$\text{SE} = \text{TPR} = \frac{N_{\hat{P}}}{N} \quad (6.11.44)$$

$$\text{SP} = \text{TNR} = \frac{N_{\hat{N}}}{N} \quad (6.11.45)$$

$$\text{FPR} = 1 - \text{SP} = \frac{N_{\hat{P}}}{N} \quad (6.11.46)$$

$$\text{PPV} = \frac{N_P}{N} \quad (6.11.47)$$

6.11.5 Receiver Operating Characteristic curve

All the assessment measures discussed so far make the assumption that the classifier returns a class for each test point. However, since most binary classifiers compute an estimation of conditional probability, a class may be returned as the outcome provided a threshold on the conditional probability. In other terms, the confusion matrix, as well as its derived measured depends on a specific threshold. The choice of a threshold is related to Type I error and Type II errors (Section 3.14) that we are ready to accept in a stochastic setting.

In order to avoid conditioning our assessment on a specific threshold, it is interesting to assess the overall accuracy for all possible thresholds. This is possible by plotting curves, like the Receiver Operating Characteristic (ROC) which plots the true positive rate (i.e. sensitivity or power) against the false positive rate (1-specificity) for different classification thresholds.

In other terms, ROC visualises the probability of detection vs. the probability of false alarm. Different points on the curve correspond to different thresholds used in the classifier.

The ideal ROC curves would follow the two axes. In practice, real-life classification rules produce ROC curves which lie between these two extremes. From (6.11.43) it follows that in a random classifier the true positive rate T_P/N_P is equal to the false positive rate F_P/N_N and then that its ROC curve follows the bisector line. In other terms, the more the ROC curve is over the bisector line, the more accurate is the classifier with respect to a random classifier.

A common way to summarise a ROC curve is to compute the area under the curve (AUC or AUROC). For a random classifier this area is 0.5. By measuring the AUC of different classifiers, we have a compact way to compare classifiers without setting a specific threshold.

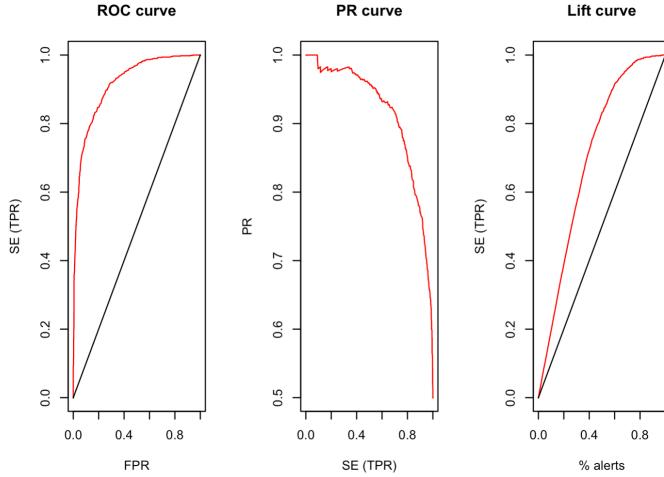


Figure 6.5: ROC, PR and lift curves of a binary classifier.

6.11.6 Precision-recall and lift curves

Other commonly used curves to visualise the accuracy of a binary classifier are the precision-recall (PR) and the lift curves.

The PR curve plots the relation between the precision (6.11.42) (probability that an example is positive given that it has been classified as positive) and the recall (6.11.41) (probability that an example is classified as positive given that it is positive). Since precision is dependent on the a priori probability of the positive class, in largely unbalanced problems (e.g. few positive classes like in fraud detection), the PR curve is more informative than the AUC.

The lift curve plots instead the true positive rate vs. the percentage $\frac{N_P}{N}$ of predicted positives. This curve is commonly used in business analytics (e.g. in fraud detection) to visualise the percentage of true positives for a given percentage of alerts. Since according to (6.11.43) in a random classifier the true positive rate is equal to the percentage of predicted positives, the lift curve of a random classifier (like in the ROC curve) coincides with the bisector line.

R script: visual assessment of a binary classifier

The R script `Learning/roc.R` illustrates the assessment of a binary classifier for a task where $x \in \mathbb{R}$, $p(x|y=+) \sim \mathcal{N}(1, 1)$ and $p(x|y=-) \sim \mathcal{N}(-1, 1)$. Suppose that the classifier categorises the examples as positive if $t > \text{Th}$ and negative if $t < \text{Th}$, where $\text{Th} \in \mathbb{R}$ is a threshold. Note that if $\text{Th} = -\infty$, all the examples are classed as positive: $T_N = F_N = 0$ which implies $SE = \frac{T_P}{N_P} = 1$ and $FPR = \frac{F_P}{F_P + T_N} = 1$. On the other way round, if $\text{Th} = \infty$, all the examples are classed as negative: $T_P = F_P = 0$ which implies $SE = 0$ and $FPR = 0$.

By sweeping over all possible values of Th we obtain the ROC and the PR curves in Figure 6.5. Each point on the ROC curve, associated to a specific threshold, has an abscissa $FPR = F_P/N_N$ and an ordinate $TPR = T_P/N_P$. Each point on the PR curve, associated to a specific threshold, has an abscissa $TPR = T_P/N_P$ and an ordinate $PR = T_P/(TP + FP)$. Each point on the lift curve, associated to a specific threshold, has an abscissa $\frac{N_P}{N}$ and an ordinate $TPR = T_P/N_P$.

•

Fraud detection example

Let us consider a fraud detection problem [53] with $N_P = 100$ frauds out of $N = 2 \cdot 10^6$ transactions. Since one of the two classes (in this case the fraud) is extremely rare, the binary classification setting is called *unbalanced* [52]. Unbalanced classification settings are very common in real-world tasks (e.g. churn detection, spam detection, predictive maintenance).

Suppose we want to compare two algorithms: the first returns 100 alerts, 90 of which are frauds. Its confusion matrix is

	Genuine (0)	Fraudulent (1)	
Classified as genuine	1,999,890	10	1,999,900
Classified as fraudulent	10	90	100
	1,999,900	100	$2 \cdot 10^6$

The second algorithm returns a much larger number of alerts (1000) 90 of which are actual frauds. Its confusion matrix is then

	Genuine (0)	Fraudulent (1)	
Classified as genuine	1,998,990	10	1,999,000
Classified as fraudulent	910	90	1000
	1,999,900	100	$2 \cdot 10^6$

Which of two algorithms is the best? In terms of TPR and FPR we have

1. $\text{TPR} = T_P/N_P = 90/100 = 0.9$, $\text{FPR} = F_P/N_N = 10/1,999,900 = 0.00000500025$
2. $\text{TPR} = 90/1000 = 0.9$, $\text{FPR} = 910/1,999,900 = 0.00045502275$

The FPR difference between the two algorithms is then negligible. Nevertheless, though the recalls of the two algorithms are almost identical, the first algorithm is definitely better in terms of false positives (much higher precision):

1. A1: $\text{PR} = T_P/(T_P + F_P) = 90/100 = 0.9$, recall=0.9
2. A2: $\text{PR} = 90/1000 = 0.09$, recall= 0.9

The example shows that, in strongly unbalanced settings, the performance of a classification algorithm may be highly sensitive to the adopted cost function.

•

6.12 Multi-class problems

So far we limited to consider binary classification tasks. However, real-world classification tasks (e.g. in bioinformatics or image recognition) are often multi-class. Some classification strategies (detailed in the following chapters) may be easily adapted to the multi-class setting, like the Naive Bayes (Section 8.2.3.1) or the KNN classifier (Section 8.2.1.1).

Suppose, however, to have a binary classification strategy that we want to use in a multi-class context. There are three main strategies to extend binary classifiers to handle multi-class tasks $\mathbf{y} \in \{c_1, \dots, c_k\}$.

1. One-versus-the rest (or one-versus-all, OVA): it builds for each class c_k a binary classifier that separates this class from the rest. To predict the class of a query point q , the outputs of the k classifiers are considered. If there is a unique class label which is consistent with the k predictions, the query point is labelled with such a class. Otherwise, one of the k classes is selected randomly.

2. Pairwise (or one-versus-one, OVO): it trains a classifier for each pair of classes, requiring in total the independent learning of $k(k - 1)/2$ binary classifiers. To predict a query point class, the output of the $k(k - 1)/2$ classifiers is calculated and a majority vote is considered. If there is a class which receives the largest number of votes, the query point is assigned to such a class. Otherwise each tie is broken randomly.
3. Coding: it first encodes each class by a binary vector of size d , then it trains a classifier for each component of the vector. The aggregation of the outputs of the d classifiers returns an *output word*, i.e. a binary of size d . Given a query q , the output word is compared against the codeword of each class, and the class having the smallest Hamming distance (the number of disagreements) to the output word is returned.

Suppose that we have a task with $k = 8$ output classes. According to the coding strategy, $\lceil \log_2 8 \rceil = 3$ binary classifiers can be used to handle this problem.

	\hat{c}_1	\hat{c}_2	\hat{c}_3
c_1	0	0	0
c_2	0	0	1
c_3	0	1	0
c_4	0	1	1
c_5	1	0	0
c_6	1	0	1
c_7	1	1	0
c_8	1	1	1

The table columns denote the classifiers while the rows contain the coding of the associated class. For instance, the \hat{c}_3 classifier will i) encode the training points labelled with the classes $\{c_2, c_4, c_6, c_8\}$ as ones ii) encode all the remaining examples as zeros and iii) learn the corresponding binary classifier.

Note that, though each strategy requires the learning of more than a single classifier, the number of trained classifiers is not the same. Given $k > 2$ classes, the number of classifiers trained for each method is mentioned here below.

- One-versus-the rest: k binary classifiers
- Pairwise: $k(k - 1)/2$ binary classifiers
- Coding: $\lceil \log_2 k \rceil$ binary classifiers where $\lceil \cdot \rceil$ denote the ceiling operator.

6.13 Concluding remarks

The chapter presented the most important steps to learn a model on the basis of a finite set of input/output data. Though the entire procedure was globally depicted as a waterfall process (Figure 6.6), it should be kept in mind that a learning process, like any modelling effort, is better represented by a spiral model characterised by feedback, iterations and adjustments. An example is the identification step, composed of two nested loops, the inner one returning the parameters of a fixed structure, and the external one searching for the best configuration.

The chapter focused on the core of the learning process which begins once the data are in a tabular format. Nevertheless, it is worth reminding that the upstream steps, sketched in Section 6.2-6.4 play a very important role as well. However, since those steps are often domain and task-dependent, we considered them beyond the scope of this book.

In the following chapters, we will quit the general perspective and we will delve into the specificities of the best known learning algorithms.

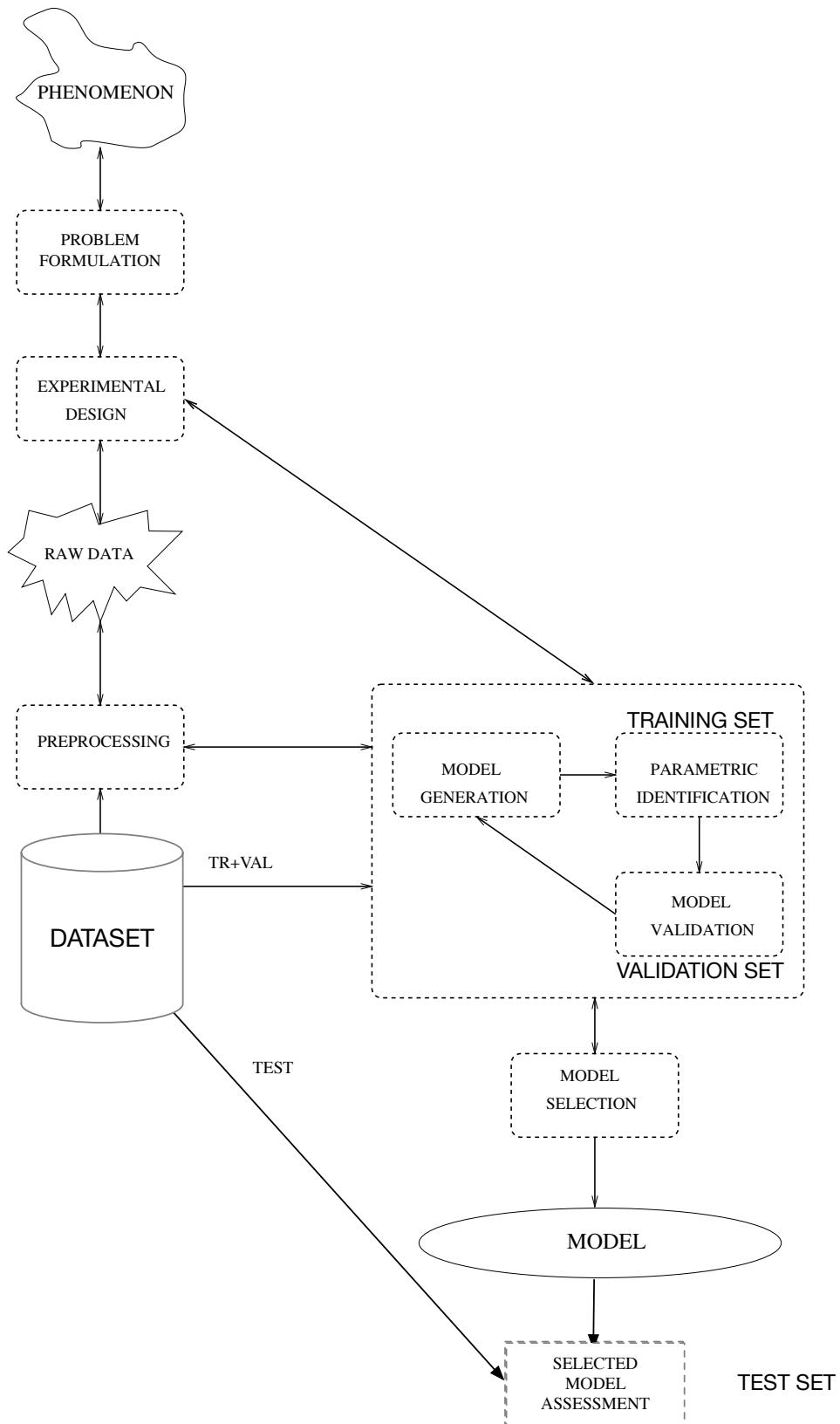


Figure 6.6: From the phenomenon to the predictive model: overview of the steps constituting the modelling process.

6.14 Exercises

1. Consider an input/output regression task where $n = 1$, $E[\mathbf{y}|x] = \sin(x)$ and $p(y|x) \sim \mathcal{N}(\sin(x), 1)$. Let $N = 100$ be the size of the training set and consider a quadratic loss function.

Let the class of hypothesis be $h_3(x) = \alpha_0 + \sum_{m=1}^3 \alpha_m x^m$.

1. Estimate the parameter by least-squares.
 2. Estimate the parameter by gradient-based search and plot the evolution of the training error as a function of the number of iterations. Show in the same figure the least-squares error.
 3. Plot the evolution of the gradient-based parameter estimations as a function of the number of iterations. Show in the same figure the least-squares parameters.
 4. Discuss the impact of the gradient-based learning rate on the training error minimisation.
 5. Estimate the parameter by Levenberg-Marquardt search and plot the evolution of the training error as a function of the number of iterations. Show in the same figure the least-squares error.
 6. Plot the evolution of the Levenberg-Marquardt parameter estimations as a function of the number of iterations. Show in the same figure the least-squares parameters.
2. Consider an input/output regression task where $n = 1$, $E[\mathbf{y}|x] = 3x + 2$ and $p(y|x) \sim \mathcal{N}(3x + 2, 1)$. Let $N = 100$ be the size of the training set and consider a quadratic loss function. Consider an iterative gradient-descent procedure to minimize the empirical error.
1. Show in a contour plot the evolution $\hat{\beta}^{(\tau)}$ of the estimated parameter vector for at least 3 different learning rates.
 2. Compute the least-squares solution and show the convergence of the iterated procedure to the least-squares solution.
3. Let us consider the dependency where the conditional distribution of \mathbf{y} is

$$\mathbf{y} = \sin(2\pi x_1 x_2 x_3) + \mathbf{w}$$

where $\mathbf{w} \sim N(0, \sigma^2)$, $\mathbf{x} \in \mathbb{R}^3$ has a 3D normal distribution with an identity covariance matrix, $N = 100$ and $\sigma = 0.25$.

Consider the following families of learners:

1. constant model returning always zero
2. constant model $h(x) = \beta_0$
3. linear model $h(x) = \mathbf{x}^T \beta$
4. K nearest neighbour for $K = 1, 3, 5, 7$ where the distance is Euclidean

Implement for each learner above a function:

```
learner<-function(Xtr,Ytr,Xts){
  #####
  ## Xtr [N,n] input training set
  ## Ytr [N,1] output training set
  ## Xts [Nts,n] input test set
  return(Yhat)
}
```

which returns a vector $[N_{ts}, 1]$ of predictions for the given input test set.

By using Monte Carlo simulation ($S = 100$ runs) and by using a fixed-input test set of size $N_{ts} = 1000$

- compute the average squared bias of all the learners,
- compute the average variance of all the learners,
- check the relation between squared bias, variance, noise variance and MSE
- define what is the best learner in terms of MSE,
- discuss the results.

Solution:

See the file `Exercise3.pdf` in the directory `gbcodes/exercises` of the companion R package (Appendix G).

4. The student should prove the following equality concerning the quantities defined in Section 6.11:

$$\text{FPR} = \frac{p}{1-p} \frac{1 - \text{PPV}}{\text{PPV}} (1 - \text{FNR})$$

where $p = \text{Prob}\{\mathbf{y} = +\}$.

Hint: use the Bayes theorem.

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 a 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 least-squares work of Gauss¹.

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

$$\mathbf{y} = \beta_0 + \beta_1 x + \mathbf{w} \quad (7.1.1)$$

where $x \in \mathbb{R}$ is the regressor (or independent) variable, \mathbf{y} is the measured response (or dependent) variable, β_0 is the intercept, β_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 σ_w^2 is independent of the x value. The assumption of constant variance is often referred to as *homoscedasticity*. From (7.1.1) we obtain

$$\text{Prob}\{\mathbf{y} = y|x\} = \text{Prob}\{\mathbf{w} = y - \beta_0 - \beta_1 x\} \quad (7.1.2)$$

$$E[\mathbf{y}|x] = f(x) = \beta_0 + \beta_1 x \quad (7.1.3)$$

The function $f(x) = E[\mathbf{y}|x]$, also known as *regression function*, is a linear function in the parameters β_0 and β_1 (Figure 7.1). In the following we will intend as

¹See <https://tinyurl.com/ytic97aa5> to know more about the famous Gauss-Legendre dispute on the least-squares discovery authorship.

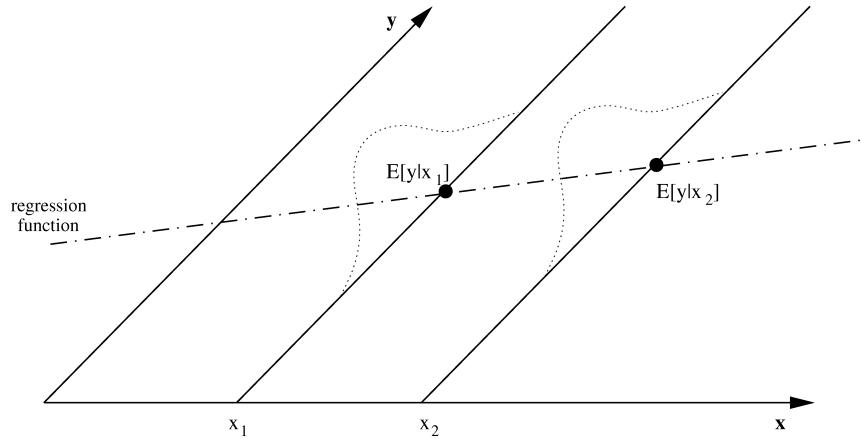


Figure 7.1: Conditional distribution and regression function for stochastic linear dependence

linear model each input/output relationship which is *linear in the parameters* but not necessarily in the dependent variables. This means that: i) any value of the response variable y is described by a linear combination of a series of parameters (regression slopes, intercept) and ii) no parameter appears as an exponent or is multiplied or divided by another parameter. According to this 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 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 linearise it.

7.1.2 Least-squares estimation

Suppose that N pairs of observations (x_i, y_i) , $i = 1, \dots, N$ are available. Let us assume that data are generated by the following stochastic dependency

$$y_i = \beta_0 + \beta_1 x_i + w_i, \quad i = 1, \dots, N \quad (7.1.4)$$

where

1. the $w_i \in \mathbb{R}$ are i.i.d realisations of the r.v. w having mean zero and constant variance σ_w^2 (homoscedasticity),
2. the $x_i \in \mathbb{R}$ are non/random and observed with negligible error.

The unknown parameters (also known as *regression coefficients*) β_0 and β_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 β_0 and β_1 , respectively
2. the fitted values of the response y

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i, \quad i = 1, \dots, N$$

so that the *residual sum of squares* (which is N times the *empirical risk*)

$$\text{SSE}_{\text{emp}} = N \cdot \widehat{\text{MISE}}_{\text{emp}} = \sum_{i=1}^N (y_i - \hat{y}_i)^2 = \sum_{i=1}^N (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$$

is minimised. 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

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}} \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

if $S_{xx} \neq 0$, where

$$\bar{x} = \frac{\sum_{i=1}^N x_i}{N}, \quad \bar{y} = \frac{\sum_{i=1}^N y_i}{N}$$

and

$$\begin{aligned} S_{xy} &= \sum_{i=1}^N (x_i - \bar{x}) y_i \\ S_{xx} &= \sum_{i=1}^N (x_i - \bar{x})^2 = \sum_{i=1}^N (x_i^2 - 2x_i \bar{x} + \bar{x}^2) = \sum_{i=1}^N (x_i^2 - x_i \bar{x} - x_i \bar{x} + \bar{x}^2) = \\ &= \sum_{i=1}^N [(x_i - \bar{x}) x_i] + \sum_{i=1}^N [\bar{x} (\bar{x} - x_i)] = \sum_{i=1}^N (x_i - \bar{x}) x_i \end{aligned}$$

It is worth noting that if $\bar{x} = 0$ and $\bar{y} = 0$ then $\hat{\beta}_0 = 0$ and

$$S_{xy} = \langle X, Y \rangle, \quad S_{xx} = \langle X, X \rangle \tag{7.1.5}$$

where X and Y are the $[N, 1]$ vectors of \mathbf{x} and \mathbf{y} observations, respectively, and the inner product $\langle \cdot, \cdot \rangle$ of two vectors is defined in Appendix B.2.

Also it is possible to write down the relation between the least squares estimation $\hat{\beta}_1$ and the sample correlation coefficient (E.0.3):

$$\hat{\rho}^2 = \frac{\hat{\beta}_1 S_{xy}}{S_{yy}} \tag{7.1.6}$$

R script

The script `Linear/lin.uni.R` computes and plots the least-squares solution for $N = 100$ observations generated according to the dependency (7.1.4) where $\beta_0 = 2$ and $\beta_1 = -2$.

If the dependency underlying the data is linear then the estimators are unbiased. We show this property for $\hat{\beta}_1$:

$$\begin{aligned} E_{\mathbf{D}_N}[\hat{\beta}_1] &= E_{\mathbf{D}_N}\left[\frac{S_{xy}}{S_{xx}}\right] = \sum_{i=1}^N \frac{(x_i - \bar{x})E[\mathbf{y}_i]}{S_{xx}} = \frac{1}{S_{xx}} \sum_{i=1}^N (x_i - \bar{x})(\beta_0 + \beta_1 x_i) = \\ &= \frac{1}{S_{xx}} \left[\sum_{i=1}^N [(x_i - \bar{x})\beta_0] + \sum_{i=1}^N [(x_i - \bar{x})\beta_1 x_i] \right] = \frac{\beta_1 S_{xx}}{S_{xx}} = \beta_1 \end{aligned}$$

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 [131] that

$$\text{Var}[\hat{\beta}_1] = \frac{\sigma_w^2}{S_{xx}} \quad (7.1.7)$$

$$E[\hat{\beta}_0] = \beta_0 \quad (7.1.8)$$

$$\text{Var}[\hat{\beta}_0] = \sigma_w^2 \left(\frac{1}{N} + \frac{\bar{x}^2}{S_{xx}} \right) \quad (7.1.9)$$

Another important result in linear regression is that the quantity

$$\hat{\sigma}_w^2 = \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{N - 2} \quad (7.1.10)$$

is an unbiased estimator of σ_w^2 under the (strong) assumption that observations 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 seen as the number N of observations reduced by the numbers of parameters estimated (slope and intercept). The estimate of the variance $\hat{\sigma}_w^2$ can be used in Equations (7.1.9) and (7.1.7) 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

$$w_i = y_i - \beta_0 - \beta_1 x_i \quad (7.1.11)$$

are i.i.d. realisations with mean zero and constant variance σ_w^2 . 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 $\mathbf{w} \sim \mathcal{N}(0, \sigma_w^2)$. By using (7.1.11), the likelihood function can be written as

$$L_N(\beta_0, \beta_1) = \prod_{i=1}^N p_{\mathbf{w}}(w_i) = \frac{1}{(2\pi)^{N/2} \sigma_w^N} \exp \left\{ \frac{-\sum_{i=1}^N (y_i - \beta_0 - \beta_1 x_i)^2}{2\sigma_w^2} \right\} \quad (7.1.12)$$

It can be shown that the estimates of β_0 and β_1 obtained by maximising $L_N(\cdot)$ 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_{Tot} (which is also N times the sample variance of \mathbf{y}) represents the total variability of the response, SS_{Mod} is the variability explained by the model and SS_{Res} is the variability left unexplained. This partition helps to determine whether the variation explained by the regression model is real or is no more than chance variation. It will be used in the following section to perform hypothesis tests 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 a hypothesis testing problem on the slope β_1 where

$$H : \beta_1 = 0, \quad \bar{H} : \beta_1 \neq 0$$

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 favour 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 D.2.5) provide evidence in favour 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 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 $\bar{\beta}$:

$$H : \beta_1 = \bar{\beta}, \quad \bar{H} : \beta_1 \neq \bar{\beta}$$

Under the assumption of normal distribution of \mathbf{w} , the following relation holds

$$\hat{\beta}_1 \sim \mathcal{N}(\beta_1, \frac{\sigma_w^2}{S_{xx}}) \quad (7.1.13)$$

It follows that

$$\frac{(\hat{\beta}_1 - \beta_1)}{\hat{\sigma}} \sqrt{S_{xx}} \sim T_{N-2}$$

where $\hat{\sigma}^2$ is the estimation of the variance σ_w^2 . This is a typical t-test applied to the regression case. Note that this statistic can also be used to test a one-sided hypothesis, e.g.

$$H : \beta_1 = \bar{\beta}, \quad \bar{H} : \beta_1 > \bar{\beta}$$

7.1.6 Interval of confidence

Under the assumption of normal distribution, according to (7.1.13)

$$\text{Prob} \left\{ -t_{\alpha/2, N-2} < \frac{(\hat{\beta}_1 - \beta_1)}{\hat{\sigma}} \sqrt{S_{xx}} < t_{\alpha/2, N-2} \right\} = 1 - \alpha$$

where $t_{\alpha/2, N-2}$ is the upper $\alpha/2$ critical point of the T -distribution with $N - 2$ degrees of freedom. Equivalently we can say that with probability $1 - \alpha$, the real parameter β_1 is covered by the interval described by

$$\hat{\beta}_1 \pm t_{\alpha/2, N-2} \sqrt{\frac{\hat{\sigma}^2}{S_{xx}}} \quad (7.1.14)$$

Note that the interval (7.1.14) may be used to test the hypothesis of input irrelevance. If the value 0 is outside the interval above, we can reject the input irrelevance hypothesis with $100(1 - \alpha)\%$ confidence.

Similarly from (7.1.9) we obtain that the $100(1 - \alpha)\%$ confidence interval of β_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

$$\hat{\mathbf{y}} = \hat{\beta}_0 + \hat{\beta}_1 x$$

be the estimator of the regression function value in x . If the linear dependence (7.1.1) holds, we have for an arbitrary $x = x_0$

$$E[\hat{\mathbf{y}}|x_0] = E[\hat{\beta}_0] + E[\hat{\beta}_1]x_0 = \beta_0 + \beta_1 x_0 = E[\mathbf{y}|x_0]$$

This means that the prediction $\hat{\mathbf{y}}$ 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

$$\text{Var}[\hat{\mathbf{y}}|x_0] = \sigma_w^2 \left[\frac{1}{N} + \frac{(x_0 - \bar{x})^2}{S_{xx}} \right]$$

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 collections from (7.1.1) and least-squares estimations were conducted.

R script

Let us consider a data set $D_N = \{x_i, y_i\}_{i=1,\dots,N}$ where

$$y_i = \beta_0 + \beta_1 x_i + w_i$$

where β_0 and β_1 are known and $\mathbf{w} \sim \mathcal{N}(0, \sigma_w^2)$ with σ_w^2 known. The R script `Linear/bv.R` may be used to:

- Study experimentally the bias and variance of the estimators $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\sigma}$ when data are generated according to the linear dependency (7.1.4) 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.

R script

Consider the medical dataset available in the R script `Linear/medical.R`. This script may be used to: i) estimate the intercept and slope of the linear model fitting the dataset, ii) plot the fitted linear model, iii) estimate the variance of the estimator of the slope, iv) test the hypothesis $\beta_1 = 0$, v) compute the confidence interval of β_1 and compare your results with the output of the R command `lm()`.

•

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_{Res} and thus increase R^2 . In other terms R^2 can be made artificially high by a practice of overfitting (Section 5.6) 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 $y \in \mathcal{Y} \subset \mathbb{R}$

$$\mathbf{y} = \beta_0 + \beta_1 x_{.1} + \beta_2 x_{.2} + \cdots + \beta_n x_{.n} + \mathbf{w} \quad (7.1.15)$$

where \mathbf{w} represents a random variable with mean zero and constant variance σ_w^2 . Note that it is possible to establish a link [168] between the partial regression coefficients β_i and partial correlation terms (Section 2.15.7) showing that β_i is related to the conditional information of \mathbf{x}_i about \mathbf{y} once fixed all the other terms (*ceteris paribus* effect) [131]:

In matrix notation² the equation 7.1.15 can be written as:

$$\mathbf{y} = x^T \boldsymbol{\beta} + \mathbf{w} \quad (7.1.16)$$

where x stands for the $[p \times 1]$ vector $x = [1, x_{.1}, x_{.2}, \dots, x_{.n}]^T$ and $p = n + 1$ is the total number of model parameters. From (7.1.16) it is easy to derive the relationship

$$E[\mathbf{y}|x] = x^T \boldsymbol{\beta} \quad (7.1.17)$$

7.1.10 The multiple linear regression model

Consider N observations $D_N = \{(x_i, y_i) : i = 1, \dots, N\}$ generated according to the stochastic dependence (7.1.16) where $x_i = [1, x_{i1}, \dots, x_{in}]^T$. We suppose that the following multiple linear relation holds

$$Y = X\boldsymbol{\beta} + W$$

²We use the notation $x_{.j}$ to denote the j th variable of the non random vector x , while $x_i = [1, x_{i1}, x_{i2}, \dots, x_{in}]^T$ denotes the i th observation of the vector x . This extension of notation is necessary when the input is not considered a random vector. In the generic case \mathbf{x}_j will be used to denote the j th variable.

where Y is the $[N \times 1]$ response vector, X is the $[N \times p]$ *data matrix*, whose j^{th} column of X contains readings on the j^{th} regressor, β is the $[p \times 1]$ vector of parameters

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{N1} & x_{N2} & \dots & x_{Nn} \end{bmatrix} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} \quad W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}$$

Here w_i are assumed uncorrelated, with mean zero and constant variance σ_w^2 (homogeneous variance). Then $\text{Var}[\mathbf{w}_1, \dots, \mathbf{w}_N] = \sigma_w^2 I_N$.

7.1.11 The least-squares solution

We seek the *least-squares estimator* $\hat{\beta}$ such that

$$\hat{\beta} = \arg \min_b \sum_{i=1}^N (y_i - x_i^T b)^2 = \arg \min_b ((Y - Xb)^T (Y - Xb)) \quad (7.1.18)$$

where

$$\text{SSE}_{\text{emp}} = N \cdot \widehat{\text{MISE}}_{\text{emp}} = ((Y - Xb)^T (Y - Xb)) = e^T e \quad (7.1.19)$$

is the *residual sum of squares* (which is N times the empirical risk (5.2.8) with quadratic loss) and

$$e = Y - Xb$$

the $[N \times 1]$ vector of residuals. The quantity SSE_{emp} is a quadratic function in the p parameters. In order to minimise

$$(Y - X\hat{\beta})^T (Y - X\hat{\beta}) = \hat{\beta}^T X^T X \hat{\beta} - \hat{\beta}^T X^T Y - Y^T X \hat{\beta} + Y^T Y$$

the vector $\hat{\beta}$ must satisfy

$$\frac{\partial}{\partial \hat{\beta}} [(Y - X\hat{\beta})^T (Y - X\hat{\beta})] = -2X^T(Y - X\hat{\beta}) = 0 \quad (7.1.20)$$

Assuming X is of full column rank, the second derivative (or Hessian)

$$\frac{\partial^2}{\partial \hat{\beta} \partial \hat{\beta}^T} [(Y - X\hat{\beta})^T (Y - X\hat{\beta})] = 2X^T X$$

is definite positive and the SSE_{emp} attains its unique global minimum in the solution of the *least-squares normal equations*

$$(X^T X)\hat{\beta} = X^T Y$$

As a result

$$\hat{\beta} = (X^T X)^{-1} X^T Y = X^\dagger Y \quad (7.1.21)$$

where the $X^T X$ matrix is a symmetric $[p \times p]$ matrix (also known as Gram matrix) and $X^\dagger = (X^T X)^{-1} X^T$ is called the *pseudo-inverse* of X since $X^\dagger X = I_N$. Note that the computation of $\hat{\beta}$ represents the parametric identification step of the supervised learning procedure (Section 5.8) when the class of hypothesis is linear.

7.1.12 Least-squares and non full-rank configurations

A full-rank X is required to ensure that the matrix $X^T X$ is invertible in (7.1.21). However, for numerical reasons it is recommended that $X^T X$ is not only invertible but also well-conditioned, or equivalently non ill-conditioned [2]. An ill-conditioned matrix is an *almost* singular matrix: its inverse may contain very large entries and sometimes numeric overflows. This means that small changes in the data may cause large and unstable changes in the solution $\hat{\beta}$. Such sensibility of the solution to the dataset should evoke in the attentive reader the notion of estimator variance (Section 3.5). In fact, in the following sections we will show that the variance of least-squares estimators is related to the inverse of $X^T X$ (e.g. Equation (7.1.22)).

But what to do in practice if X is not full-rank (or rank-deficient) or ill-conditioned? A first numerical fix consists in computing the generalised QR decomposition (Appendix B.5)

$$X = QR$$

where Q is an orthogonal $[N, p']$ matrix and R is a $[p', p]$ upper-triangular matrix of full row rank with $p' < p$. Since RR^T is invertible, the pseudo-inverse in (7.1.21) can be written as $X^\dagger = R^T(RR^T)^{-1}Q^T$ (details in Section 2.8.1 of [2]). A second solution consists in regularising the optimisation, i.e. constraining the optimisation problem (7.1.18) by adding a term which penalises solutions β with too large a norm. This leads to the ridge regression formulation which will be discussed in Section 10.5.1.1. In more general terms, since no/ invertible or ill-conditioned configurations are often due to highly correlated (multicollinear) or redundant inputs, the use of feature selection strategies (Chapter 10) before the parametric identification step may be beneficial.

7.1.13 Properties of least-squares estimators

Under the condition that the linear stochastic dependence (7.1.16) holds, it can be shown [131] that:

- If $E[\mathbf{w}] = 0$ then the random vector $\hat{\beta}$ is an unbiased estimator of β .
- The *residual mean square* estimator

$$\hat{\sigma}_{\mathbf{w}}^2 = \frac{(Y - X\hat{\beta})^T(Y - X\hat{\beta})}{N - p}$$

is an unbiased estimator of the error variance $\sigma_{\mathbf{w}}^2$.

- If the \mathbf{w}_i are uncorrelated and have common variance, the variance-covariance matrix of $\hat{\beta}$ is given by

$$\text{Var}[\hat{\beta}] = \sigma_{\mathbf{w}}^2(X^T X)^{-1} \quad (7.1.22)$$

It can also be shown (Gauss-Markov theorem) that the least-squares estimation $\hat{\beta}$ is the "best linear unbiased estimator" (BLUE) i.e. it has the lowest variance among all linear unbiased estimators.

From the results above it is possible to derive the confidence intervals of model parameters, similarly to the univariate case discussed in Section 7.1.6.

R script

A list of the most important least-squares summary statistics is returned by the `summary` of the R command `lm`. See for instance the script `Linear/ls.R`.

```

summary(lm(Y~X))

Call:
lm(formula = Y ~ X)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.40141 -0.14760 -0.02202  0.03001  0.43490 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.09781   0.11748   9.345 6.26e-09 ***
X            0.02196   0.01045   2.101   0.0479    
                                                        
(Intercept) ***
X             * 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2167 on 21 degrees of freedom
Multiple R-Squared: 0.1737,    Adjusted R-squared: 0.1343 
F-statistic: 4.414 on 1 and 21 DF,  p-value: 0.0479

```

•

7.1.14 Variance of the prediction

Since the estimator $\hat{\beta}$ is unbiased, this is also the case for the prediction $\hat{y} = x_0^T \hat{\beta}$ for a generic input value $x = x_0$. Its variance is

$$\text{Var}[\hat{y}|x_0] = \sigma_w^2 x_0^T (X^T X)^{-1} x_0 \quad (7.1.23)$$

Assuming that w is normally distributed, the $100(1-\alpha)\%$ confidence bound for the regression value in x_0 is given by

$$\hat{y}(x_0) \pm t_{\alpha/2, N-p} \hat{\sigma}_w \sqrt{x_0^T (X^T X)^{-1} x_0}$$

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 $\hat{\sigma}_w \sqrt{x_0^T (X^T X)^{-1} x_0}$, obtained from (7.1.23), is the *standard error of prediction* for multiple regression.

R script

The R script `Linear/bv_mult.R` validates by Monte Carlo simulation the properties of least-squares estimation mentioned in Section 7.1.11 and 7.1.14.

In order to assess the generality of the results, we invite the reader to run the script for different input sizes n , different number of observations N and different values of the parameter β .

•

7.1.15 The HAT matrix

The Hat matrix is defined as

$$H = X(X^T X)^{-1} X^T \quad (7.1.24)$$

It is a symmetric, idempotent $[N \times N]$ matrix that transforms the output values \mathbf{Y} of the training set in the regression predictions $\hat{\mathbf{Y}}$:

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

Using the above relation, the vector of residuals can be written as:

$$\mathbf{e} = \mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{Y} - \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = [\mathbf{I} - \mathbf{H}]\mathbf{Y}$$

and the residual sum of squares as

$$\mathbf{e}^T\mathbf{e} = \mathbf{Y}^T[\mathbf{I} - \mathbf{H}]^2\mathbf{Y} = \mathbf{Y}^T[\mathbf{I} - \mathbf{H}]\mathbf{Y} = \mathbf{Y}^T\mathbf{P}\mathbf{Y} \quad (7.1.25)$$

where \mathbf{P} is a $[N \times N]$ matrix, called the *projection* matrix.

if \mathbf{X} has full rank, by commutativity of the trace operator it follows that

$$\text{tr}(\mathbf{H}) = \text{tr}(\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T) = \text{tr}(\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}) = \text{tr}(\mathbf{I}_p) = p \quad (7.1.26)$$

If we perform a QR decomposition of \mathbf{X} (Appendix B.5) then we obtain

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T = \mathbf{Q}\mathbf{R}(\mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R})^{-1}\mathbf{R}^T\mathbf{Q}^T = \mathbf{Q}\mathbf{R}\mathbf{R}^{-1}(\mathbf{R}^T)^{-1}\mathbf{R}^T\mathbf{Q}^T = \mathbf{Q}\mathbf{Q}^T \quad (7.1.27)$$

Note that, in this case, the input matrix \mathbf{X} is replaced by the matrix \mathbf{Q} which contains an orthogonalised transformation of the original inputs.

7.1.16 Generalisation error of the linear model

Given a training dataset $D_N = \{\langle x_i, y_i \rangle : i = 1, \dots, N\}$ and a query point x , it is possible to return a linear prediction

$$\hat{y} = h(x, \alpha) = x^T\hat{\boldsymbol{\beta}}$$

where $\hat{\boldsymbol{\beta}}$ is returned by least-squares estimation (7.1.21). From an estimation perspective, $\hat{\boldsymbol{\beta}}$ is a realisation of the random estimator $\hat{\boldsymbol{\beta}}$ for the specific dataset D_N . But which precision can we expect from $\hat{y} = x^T\hat{\boldsymbol{\beta}}$ if we average the prediction error over all finite-size datasets D_N that can be generated by the linear dependency (7.1.15)?

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 (5.5.24). But how can we estimate this quantity in the linear case? Also, is the empirical risk $\widehat{\text{MISE}}_{\text{emp}}$ in (7.1.19) a reliable estimate of MISE?

7.1.16.1 The expected empirical error

This section derives analytically that the empirical risk $\widehat{\text{MISE}}_{\text{emp}}$ (defined in (7.1.19)) is a biased estimator of the MISE generalisation error.

Let us first compute the expectation of the residual sum of squares which is equal to N times the empirical risk. According to (7.1.25) and Theorem 5.2 the expectation can be written as

$$E_{D_N}[\mathbf{SSE}_{\text{emp}}] = E_{D_N}[\mathbf{e}^T\mathbf{e}] = E_{D_N}[\mathbf{Y}^T\mathbf{P}\mathbf{Y}] = \sigma_w^2 \text{tr}(\mathbf{P}) + E[\mathbf{Y}^T] P E[\mathbf{Y}]$$

Since $\text{tr}(ABC) = \text{tr}(CAB)$

$$\begin{aligned} \text{tr}(\mathbf{P}) &= \text{tr}(\mathbf{I} - \mathbf{H}) = N - \text{tr}(\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T) \\ &= N - \text{tr}(\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}) = N - \text{tr}(\mathbf{I}_p) = N - p \end{aligned}$$

and we have

$$E_{\mathbf{D}_N}[\mathbf{e}^T \mathbf{e}] = (N - p)\sigma_w^2 + (X\beta)^T P(X\beta) = \quad (7.1.28)$$

$$= (N - p)\sigma_w^2 + \beta^T X^T (I - X(X^T X)^{-1} X^T) X\beta = \quad (7.1.29)$$

$$= (N - p)\sigma_w^2 \quad (7.1.30)$$

It follows that

$$E_{\mathbf{D}_N}[\widehat{\text{MISE}}_{\text{emp}}] = E_{\mathbf{D}_N}\left[\frac{\text{SSE}_{\text{emp}}}{N}\right] = E_{\mathbf{D}_N}\left[\frac{\mathbf{e}^T \mathbf{e}}{N}\right] = (1 - p/N)\sigma_w^2 \quad (7.1.31)$$

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 (7.1.15) but independent of the training output Y .

$$\begin{aligned} & E_{\mathbf{D}_N, \mathbf{Y}_{ts}}[(\mathbf{Y}_{ts} - X\hat{\beta})^T(\mathbf{Y}_{ts} - X\hat{\beta})] = \\ &= E_{\mathbf{D}_N, \mathbf{Y}_{ts}}[(\mathbf{Y}_{ts} - X\beta + X\beta - X\hat{\beta})^T(\mathbf{Y}_{ts} - X\beta + X\beta - X\hat{\beta})] = \\ &= E_{\mathbf{D}_N, \mathbf{Y}_{ts}}[(\mathbf{W}_{ts} + X\beta - X\hat{\beta})^T(\mathbf{W}_{ts} + X\beta - X\hat{\beta})] = \\ &= N\sigma_w^2 + E_{\mathbf{D}_N}[(X\beta - X\hat{\beta})^T(X\beta - X\hat{\beta})] \end{aligned}$$

Since

$$\begin{aligned} X\beta - X\hat{\beta} &= X\beta - X(X^T X)^{-1} X^T Y = \\ &= X\beta - X(X^T X)^{-1} X^T (X\beta + W) = -X(X^T X)^{-1} X^T W \end{aligned}$$

we obtain

$$\begin{aligned} & N\sigma_w^2 + E_{\mathbf{D}_N}[(X\beta - X\hat{\beta})^T(X\beta - X\hat{\beta})] \\ &= N\sigma_w^2 + E_{\mathbf{D}_N}[(\mathbf{W}^T X(X^T X)^{-1} X^T)(X(X^T X)^{-1} X^T \mathbf{W})] \\ &= N\sigma_w^2 + E_{\mathbf{D}_N}[\mathbf{W}^T X(X^T X)^{-1} X^T \mathbf{W}] \\ &= N\sigma_w^2 + \sigma_w^2 \text{tr}(X(X^T X)^{-1} X^T) \\ &= N\sigma_w^2 + \sigma_w^2 \text{tr}(X^T X(X^T X)^{-1}) \\ &= N\sigma_w^2 + \sigma_w^2 \text{tr}(I_p) = \sigma_w^2(N + p) \end{aligned}$$

By dividing the above quantity by N

$$\text{MISE} = (1 + p/N)\sigma_w^2 \quad (7.1.32)$$

From (7.1.31) and (7.1.32) it follows that the empirical error $\widehat{\text{MISE}}_{\text{emp}}$ is a biased estimate of MISE:

$$E_{\mathbf{D}_N}[\widehat{\text{MISE}}_{\text{emp}}] = E_{\mathbf{D}_N}\left[\frac{\mathbf{e}^T \mathbf{e}}{N}\right] = \sigma_w^2(1 - p/N) \neq \text{MISE} = \sigma_w^2(1 + p/N) \quad (7.1.33)$$

As a consequence, if we replace $\widehat{\text{MISE}}_{\text{emp}}$ with

$$\frac{\mathbf{e}^T \mathbf{e}}{N} + 2\frac{\sigma_w^2 p}{N} \quad (7.1.34)$$

we correct the bias and we obtain an unbiased estimator of the MISE generalisation error. Nevertheless, this estimator requires an estimate of the noise variance.

R script

The R script `Linear/ee.R` performs a Monte Carlo validation of (7.1.33).

•

Example

Let $\{y_1, \dots, y_N\} \leftarrow F_y$ be the training set. Consider the simplest linear predictor of the output variable: the average $\hat{\mu}_y$ (i.e. $p = 1$). This means that

$$\hat{y}_i = \frac{\sum_{i=1}^N y_i}{N} = \hat{\mu}_y, \quad i = 1, \dots, N$$

We want to show that, even for this simple estimator, the empirical error is a biased estimator of the quality of this predictor. Let μ 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 $\{z_1, \dots, z_N\} \leftarrow F_y$ 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 $E[\hat{\mu}_y] = \mu$ and $\text{Var}[\hat{\mu}_y] = \sigma^2/N$

$$\begin{aligned} N \cdot \text{MISE} &= E_{\mathbf{D}_N, \mathbf{Y}_{ts}} \left[\sum_{i=1}^N (z_i - \hat{\mu}_y)^2 \right] = E_{\mathbf{D}_N, \mathbf{w}} \left[\sum_{i=1}^N (\mu + \mathbf{w}_i - \hat{\mu}_y)^2 \right] \\ &= N\sigma^2 + \sum_{i=1}^N E_{\mathbf{D}_N} [(\mu - \hat{\mu}_y)^2] \\ &= N\sigma^2 + N(\sigma^2/N) = (N+1)\sigma^2 \end{aligned}$$

Instead, since $\hat{\sigma}_y^2 = (\sum_{i=1}^N (y_i - \hat{\mu}_y)^2)/(N-1)$ and $E[\hat{\sigma}_y^2] = \sigma^2$

$$E_{\mathbf{D}_N} \left[\sum_{i=1}^N (y_i - \hat{\mu}_y)^2 \right] = E_{\mathbf{D}_N} [(N-1)\hat{\sigma}_y^2] = (N-1)\sigma^2 \neq N \cdot \text{MISE}$$

It follows that, even for a simple estimator like the estimator of the mean, the empirical error is a biased estimate of the accuracy (see R file `Linear/ee_mean.R`).

•

7.1.16.2 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 $2\sigma_{\mathbf{w}}^2 p/N$ makes it unbiased.

Suppose we have an estimate $\hat{\sigma}_{\mathbf{w}}^2$ of $\sigma_{\mathbf{w}}^2$. By replacing it into the expression (7.1.34) we obtain the so-called *Predicted Square Error (PSE)* criterion

$$\text{PSE} = \widehat{\text{MISE}}_{\text{emp}} + 2\hat{\sigma}_{\mathbf{w}}^2 p/N \tag{7.1.35}$$

In particular, if we take as estimate of $\sigma_{\mathbf{w}}^2$ the quantity

$$\hat{\sigma}_{\mathbf{w}}^2 = \frac{1}{N-p} \text{SSE}_{\text{emp}} = \frac{N}{N-p} \widehat{\text{MISE}}_{\text{emp}}$$

we obtain the so-called *Final Prediction Error (FPE)*

$$\text{FPE} = \frac{1+p/N}{1-p/N} \widehat{\text{MISE}}_{\text{emp}} \tag{7.1.36}$$

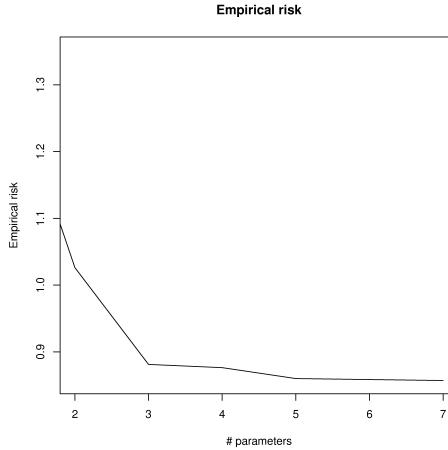


Figure 7.2: Estimation of the generalisation error of h_m , $m = 2, \dots, 7$ returned by the empirical error.

The PSE and the FPE criteria allow us to replace the empirical risk with a more accurate estimate of the generalisation error of a linear model. Although their expression is easy to compute, it is worth reminding that their derivation relies on the assumption that the stochastic input/output dependence has the linear form 7.1.16.

R script

Let us consider an input/output dependence

$$\mathbf{y} = f(x) + \mathbf{w} = 1 + x + x^2 + x^3 + \mathbf{w} \quad (7.1.37)$$

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 `Linear/fpe.R` assesses the prediction accuracy of 7 different models having the form

$$h_m(x) = \hat{\beta}_0 + \sum_{j=1}^m \hat{\beta}_j x^j \quad (7.1.38)$$

by using the empirical risk and the FPE measure. These results are compared with the generalisation error measured by

$$\text{MISE}_m = \frac{1}{N} \sum_{i=1}^N (h_m(x_i) - f(x_i))^2 \quad (7.1.39)$$

The empirical risk and the FPE values for $m = 2, \dots, 7$ are plotted in Figure 7.2 and 7.3, respectively. The values 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.

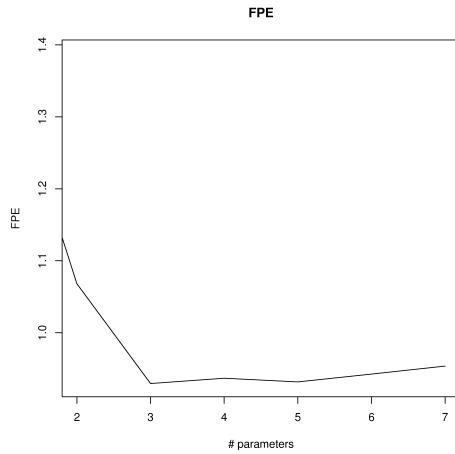


Figure 7.3: Estimation of the generalisation error of h_m , $m = 2, \dots, 7$ returned by the FPE.

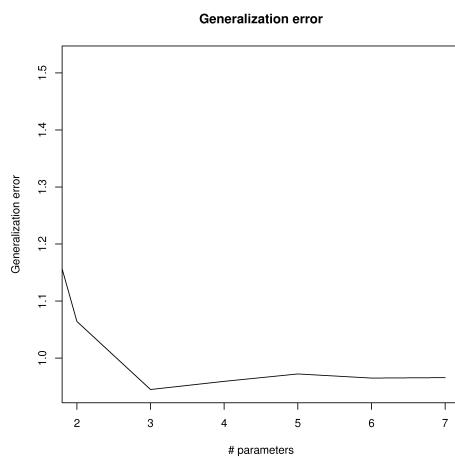


Figure 7.4: Computation of the generalisation error of h_m , $m = 2, \dots, 7$ by (7.1.39).

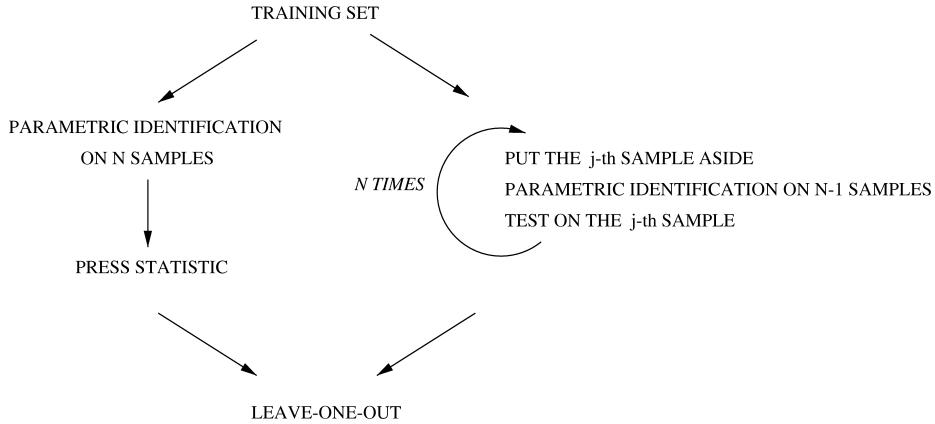


Figure 7.5: Leave-one-out for linear models. The leave-one-out error can be computed in two equivalent ways: the slowest way (on the right) which repeats N times the training and the test procedure; the fastest way (on the left) which performs only once the parametric identification and the computation of the PRESS statistic.

7.1.17 The PRESS statistic

Section 5.9.1 introduced cross-validation to provide a reliable estimate of the generalisation error G_N . The disadvantage of this approach is that it requires the training process to be repeated l times, implying a large computational effort. However, in the linear case the PRESS (Prediction Sum of Squares) statistic [7] returns the leave-one-out cross-validation error at a reduced computational cost (Fig. 7.5). PRESS relies on 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. (8.1.48). Consider a training set D_N in which for N times

1. we set aside the i^{th} observation ($i = 1, \dots, N$) $\langle x_i, y_i \rangle$ from the training set,
2. we use the remaining $N - 1$ observations to estimate the linear regression coefficients $\hat{\beta}^{-i}$,
3. we use $\hat{\beta}^{-i}$ to predict the target in x_i .

The leave-one-out residual is

$$e_i^{\text{loo}} = y_i - \hat{y}_i^{-i} = y_i - x_i^T \hat{\beta}^{-i}, \quad i = 1, \dots, N \quad (7.1.40)$$

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. use the whole training set to estimate the linear regression coefficients $\hat{\beta}$. This procedure is performed only once and returns, as a byproduct, the Hat matrix (see Section 7.1.15)

$$H = X(X^T X)^{-1} X^T \quad (7.1.41)$$

2. compute the residual vector e , whose i^{th} term is $e_i = y_i - x_i^T \hat{\beta}$,
3. use the PRESS statistic to compute e_i^{loo} as

$$e_i^{\text{loo}} = \frac{e_i}{1 - H_{ii}} \quad (7.1.42)$$

where H_{ii} is the i^{th} diagonal term of the matrix H .

Note that (7.1.42) is not an approximation of (7.1.40) but simply a faster way of computing the leave-one-out residual e_i^{loo} .

Let us now derive the formula of the PRESS statistic. Matrix manipulations show that

$$X^T X - x_i x_i^T = X_{-i}^T X_{-i} \quad (7.1.43)$$

where $X_{-i}^T X_{-i}$ is the $X^T X$ matrix obtained by putting the i^{th} row aside. Using the relation (B.9.12) we have

$$(X_{-i}^T X_{-i})^{-1} = (X^T X - x_i x_i^T)^{-1} = (X^T X)^{-1} + \frac{(X^T X)^{-1} x_i x_i^T (X^T X)^{-1}}{1 - H_{ii}} \quad (7.1.44)$$

and

$$\hat{\beta}^{-i} = (X_{-i}^T X_{-i})^{-1} X_{-i}' y_{-i} = \left[(X^T X)^{-1} + \frac{(X^T X)^{-1} x_i x_i^T (X^T X)^{-1}}{1 - H_{ii}} \right] X_{-i}' y_{-i} \quad (7.1.45)$$

where y_{-i} is the target vector with the i^{th} example set aside.

From (7.1.40) and (7.1.45) we have

$$\begin{aligned} e_i^{\text{loo}} &= y_i - x_i^T \hat{\beta}^{-i} \\ &= y_i - x_i^T \left[(X^T X)^{-1} + \frac{(X^T X)^{-1} x_i x_i^T (X^T X)^{-1}}{1 - H_{ii}} \right] X_{-i}' y_{-i} \\ &= y_i - x_i^T (X^T X)^{-1} X_{-i}' y_{-i} - \frac{H_{ii} x_i^T (X^T X)^{-1} X_{-i}' y_{-i}}{1 - H_{ii}} \\ &= \frac{(1 - H_{ii}) y_i - x_i^T (X^T X)^{-1} X_{-i}' y_{-i}}{1 - H_{ii}} \\ &= \frac{(1 - H_{ii}) y_i - x_i^T (X^T X)^{-1} (X^T y - x_i y_i)}{1 - H_{ii}} = \\ &= \frac{(1 - H_{ii}) y_i - \hat{y}_i + H_{ii} y_i}{1 - H_{ii}} = \frac{y_i - \hat{y}_i}{1 - H_{ii}} = \frac{e_i}{1 - H_{ii}} \end{aligned} \quad (7.1.46)$$

where $X_{-i}' y_{-i} + x_i y_i = X^T y$ and $x_i^T (X^T X)^{-1} X^T y = \hat{y}_i$. Thus, the leave-one-out estimate of the local mean integrated squared error is:

$$\hat{G}_{\text{loo}} = \frac{1}{N} \sum_{i=1}^N \left\{ \frac{y_i - \hat{y}_i}{1 - H_{ii}} \right\}^2 \quad (7.1.47)$$

Since from (7.1.26) the sum of the diagonal terms of the H matrix is p , the average value of H_{ii} is p/N . It follows that the PRESS may be approximated by

$$\hat{G}_{\text{loo}} \approx \frac{1}{N} \sum_{i=1}^N \left\{ \frac{y_i - \hat{y}_i}{1 - p/N} \right\}^2$$

which leads us to the GCV formula (6.8.34).

7.1.18 Dual linear formulation

Consider a linear regression problem with $[N, n]$ input matrix X and $[N, 1]$ output vector y . The conventional least-squares solution is the $[p, 1]$ parameter vector (7.1.21) where $p = n+1$. This formulation is common in conventional statistical settings where the number of observations is supposed to be much larger than the number of variables.

However, machine learning may be confronted with high-dimensional settings where the ratio of observations to features is low: this would imply a very large value of p and risks of ill-conditioning of the numerical solution. In this case it is interesting to consider a dual formulation of the least-squares problem based on (B.9.13). In this formulation

$$\hat{\beta} = (X'X)^{-1}X'y = X' \underbrace{X(X'X)^{-2}X'y}_{\alpha} = X'\alpha = \sum_{i=1}^N \alpha_i x_i$$

where α is a $[N, 1]$ vector and x_i is the $[n, 1]$ vector which represents the i th observation. It follows that if $N \ll p$, the dual formulation has fewer parameters than the conventional one with advantages in terms of storage requirements and numerical conditioning.

7.1.19 The weighted least-squares

The assumption of homogeneous variance of the noise \mathbf{w} made in Eq. (7.1.16) is often violated in practical situations. Suppose we relax the assumption that $\text{Var}(\mathbf{w}) = \sigma_w^2 I_N$ with I_N the 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

$$V = \text{diag}[\sigma_1^2, \sigma_2^2, \dots, \sigma_N^2] \quad (7.1.48)$$

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 β 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 minimised is no more (7.1.19) but depends on V and is given by

$$(y - X\hat{\beta})^T V^{-1} (y - X\hat{\beta}) \quad (7.1.49)$$

The estimate of β is then

$$\hat{\beta} = (X^T V^{-1} X)^{-1} X^T V^{-1} y \quad (7.1.50)$$

The corresponding estimator is called the *generalised least-squares estimator* and has the following properties: i) it is unbiased, that is $E[\hat{\beta}] = \beta$, ii) under the assumption $\mathbf{w} \sim \mathcal{N}(0, V)$ it is the minimum variance estimator among all the unbiased estimators.

7.1.20 Recursive least-squares

In many analytics tasks, data records are not statically available but have to be processed and analysed continuously rather than in batches. Examples are the data streams generated from sensors (notably IoT), financial, business intelligence or adaptive control applications. In those 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 (7.1.21) for a training set of N observations 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 observations used for the estimation. Suppose that a new data point $\langle x_{N+1}, y_{N+1} \rangle$ 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* [26].

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

$$\begin{aligned} S_{(N+1)} &= (X_{(N+1)}^T X_{(N+1)}) = \left(\begin{bmatrix} X_{(N)}^T x_{N+1}^T \\ x_{N+1} \end{bmatrix} \begin{bmatrix} X_{(N)} \\ x_{N+1} \end{bmatrix} \right) \\ &= \left(X_{(N)}^T X_{(N)} + x_{N+1}^T x_{N+1} \right) = S_{(N)} + x_{N+1}^T x_{N+1} \end{aligned} \quad (7.1.51)$$

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

$$\begin{aligned} S_{(N+1)} \hat{\beta}_{(N+1)} &= \begin{bmatrix} X_{(N)} \\ x_{N+1} \end{bmatrix}^T \begin{bmatrix} Y_{(N)} \\ y_{N+1} \end{bmatrix} = S_{(N)} \hat{\beta}_{(N)} + x_{N+1}^T y_{N+1} \\ &= (S_{(N+1)} - x_{N+1}^T x_{N+1}) \hat{\beta}_{(N)} + x_{N+1}^T y_{N+1} \\ &= S_{(N+1)} \hat{\beta}_{(N)} - x_{N+1}^T x_{N+1} \hat{\beta}_{(N)} + x_{N+1}^T y_{N+1} \end{aligned}$$

or equivalently

$$\hat{\beta}_{(N+1)} = \hat{\beta}_{(N)} + S_{(N+1)}^{-1} x_{N+1}^T (y_{N+1} - x_{N+1} \hat{\beta}_{(N)}) \quad (7.1.52)$$

7.1.20.1 1st Recursive formulation

From (7.1.51) and (7.1.52) 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 observation $\langle x_{N+1}, y_{N+1} \rangle$. 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.

7.1.20.2 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

$$V_{(N+1)} = V_{(N)} - V_{(N)} x_{(N+1)}^T (I + x_{N+1} V_{(N)} x_{N+1}^T)^{-1} x_{N+1} V_{(N)} \quad (7.1.53)$$

$$= V_{(N)} - \frac{V_{(N)} x_{N+1}^T x_{N+1} V_{(N)}}{1 + x_{N+1} V_{(N)} x_{N+1}^T} \quad (7.1.54)$$

From (7.1.53) and (7.1.52) we obtain a second recursive formulation:

$$\begin{cases} V_{(N+1)} &= V_{(N)} - \frac{V_{(N)} x_{N+1}^T x_{N+1} V_{(N)}}{1 + x_{N+1} V_{(N)} x_{N+1}^T} \\ \gamma_{(N+1)} &= V_{(N+1)} x_{N+1}^T \\ e &= y_{N+1} - x_{N+1} \hat{\beta}_{(N)} \\ \hat{\beta}_{(N+1)} &= \hat{\beta}_{(N)} + \gamma_{(N+1)} e \end{cases} \quad (7.1.55)$$

7.1.20.3 RLS initialisation

Both recursive formulations presented above require the initialisation 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

$$\begin{aligned} V_{(N)} &= (X_{(N)}^T X_{(N)})^{-1} \\ \hat{\beta}_{(N)} &= V_{(N)} X_{(N)}^T Y_{(N)} \end{aligned}$$

and to start iterating from the $N + 1^{\text{th}}$ point. Otherwise, in case of a generic initialisation $\hat{\beta}_{(0)}$ and $V_{(0)}$ we have the following relations

$$\begin{aligned} V_{(N)} &= (V_{(0)} + X_{(N)}^T X_{(N)})^{-1} \\ \hat{\beta}_{(N)} &= V_{(N)} (X_{(N)}^T Y_{(N)} + V_{(0)}^{-1} \hat{\beta}_{(0)}) \end{aligned}$$

A common choice is to put

$$V_{(0)} = aI, \quad a > 0$$

Since $V_{(0)}$ represents the variance of the estimator to choose a very large a is equivalent to consider the initial estimation of β as very uncertain. By setting a equal to a large number, the RLS algorithm will diverge very rapidly from the initialisation $\hat{\beta}_{(0)}$. 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 the absence of further information, the initial value $\hat{\beta}_{(0)}$ is usually put equal to a zero vector.

7.1.20.4 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 of these situations are common in adaptive control problems.

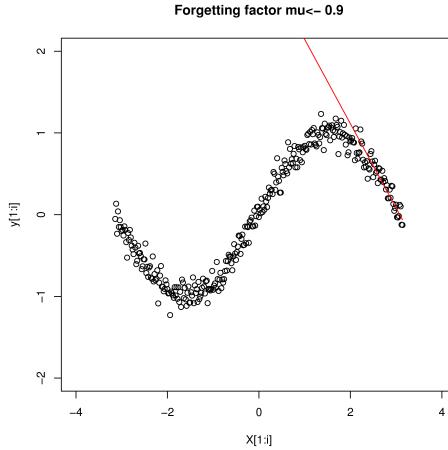


Figure 7.6: RLS fitting of a nonlinear function where the arrival order of data is from left to right

RLS techniques can deal with these situations by a modification of the formulation (7.1.55) 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 μ , the higher the forgetting, (ii) for $\mu = 1$ we have the conventional RLS formulation.

R script

The R script `Linear/lin_rls.R` implements the RLS fitting of a nonlinear univariate function. The simulation shows that the fitting evolves as long as data $x_i, y_i, i = 1, \dots, N$ are collected. Note that the values $x_i, i = 1, \dots, N$ are increasingly ordered. 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 points since the values on the left, which are also the oldest ones, are forgotten.

•

7.2 Linear approaches to classification

The methods presented so far deal with linear regression tasks. Those methods may be easily extended to classification once we consider that in a binary 0/1 classification case the conditional expectation coincides with conditional probability:

$$E[\mathbf{y}|x] = 1 \cdot \text{Prob}\{\mathbf{y} = 1|x\} + 0 \cdot \text{Prob}\{\mathbf{y} = 0|x\} = \text{Prob}\{\mathbf{y} = 1|x\} \quad (7.2.56)$$

In other words, by encoding the two classes in 0/1 values and estimating the conditional expectation with regression techniques, we estimate as well the conditional probability. Such value may be used to return the most probable class associated to a query point x .

This section will present some additional strategies to learn linear boundaries between classes. The first strategy relies on modelling the class conditional densities and derive from them the equation of the boundary region. The other strategies aim to learn directly the equations of separating hyperplanes.

7.2.1 Linear discriminant analysis

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 $\{c_1, \dots, c_K\}$ such that

$$\sum_{k=1}^K \text{Prob}\{\mathbf{y} = c_k | \mathbf{x}\} = 1$$

A classifier can be represented in terms of a set of K discriminant functions $g_k(x)$, $k = 1, \dots, K$ such that the classifier applies the following decision rule [62]: *assigns a feature vector x to a class $\hat{y}(x) = c_k$ if*

$$k = \arg \max_j g_j(x) \quad (7.2.57)$$

Section 5.3 showed that in the case of a zero-one loss function (Equation (5.3.13)), the optimal classifier corresponds to a *maximum a posteriori* discriminant function $g_k(x) = \text{Prob}\{\mathbf{y} = c_k | \mathbf{x}\}$. This means that if we are able to define the K functions $g_k(\cdot)$, $k = 1, \dots, K$ and we apply the classification rule (7.2.57) 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 [62]. 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:

$$g_k(x) = \text{Prob}\{\mathbf{y} = c_k | \mathbf{x}\} = \frac{p(x|\mathbf{y} = c_k)P(\mathbf{y} = c_k)}{\sum_{k=1}^K p(x|\mathbf{y} = c_k)P(\mathbf{y} = c_k)} \quad (7.2.58)$$

$$g_k(x) = p(x|\mathbf{y} = c_k)P(\mathbf{y} = c_k) \quad (7.2.59)$$

$$g_k(x) = \ln p(x|\mathbf{y} = c_k) + \ln P(\mathbf{y} = c_k) \quad (7.2.60)$$

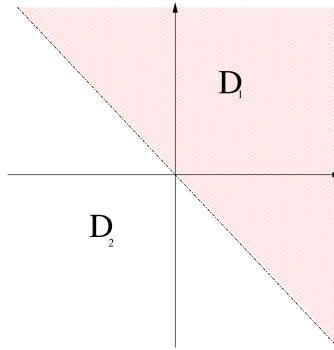


Figure 7.7: Decision boundary and decision regions for the binary discrimination functions $g_1(x) = 3x_1 + x_2 + 2$ and $g_2(x) = 2x_1 + 2$

and returns a Bayes classifier.

7.2.1.1 Discriminant functions in the Gaussian case

Let us consider a binary classification task where the inverse conditional densities are multivariate normal (Section 2.15), i.e. $p(\mathbf{x} = x | \mathbf{y} = c_k) \sim \mathcal{N}(\mu_k, \Sigma_k)$ where $x \in \mathbb{R}^n$, μ_k is a $[n, 1]$ vector and Σ_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 (7.2.60) we obtain

$$g_k(x) = \ln p(x | \mathbf{y} = c_k) + \ln P(\mathbf{y} = c_k) \quad (7.2.61)$$

$$= -\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) - \frac{n}{2} \ln 2\pi - \frac{1}{2} \ln \det(\Sigma_k) + \ln P(\mathbf{y} = c_k) \quad (7.2.62)$$

If we make no assumptions about Σ_k the discriminant function is quadratic. Now let us consider a simpler case where all the distributions have the same diagonal covariance matrix $\Sigma_k = \sigma^2 I$ where I is the $[n, n]$ identity matrix. It follows that

$$\det(\Sigma_k) = \sigma^{2n}, \quad \Sigma_k^{-1} = (1/\sigma^2)I$$

are independent of k and can be ignored by the decision rule (7.2.57). From (7.2.61), 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 this is equivalent to a linear discriminant function

$$g_k(x) = w_k^T x + w_{k0} \quad (7.2.63)$$

where w_k is a $[n, 1]$ vector

$$w_k = \frac{1}{\sigma^2} \mu_k \quad (7.2.64)$$

and

$$w_{k0} = -\frac{1}{2\sigma^2} \mu_k^T \mu_k + \ln P(\mathbf{y} = c_k) \quad (7.2.65)$$

In the two-class 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 a hyperplane having equation

$$w^T (x - x_0) = 0 \quad (7.2.66)$$

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 (7.2.66) defines a hyperplane through the point x_0 and orthogonal to the vector w .

7.2.1.2 Uniform prior case

If the prior probabilities $P(\mathbf{y} = c_k)$ are identical for the K classes, then the term $\ln P(\mathbf{y} = c_k)$ is a constant that can be ignored. In this case, it can be shown that the optimum decision rule is a *minimum distance classifier* [62]. 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 minimising the *Mahalanobis distance*

$$\hat{c}(x) = \arg \min_k (x - \mu_k)^T \Sigma^{-1} (x - \mu_k) \quad (7.2.67)$$

R script

The R script `Linear/discrim.R` considers a binary classification task ($c_1 = \text{red}$, $c_2 = \text{green}$) 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 (7.2.66) are plotted in Figure 7.8.

The R script `Linear/discrim2.R` shows instead the limitations of the LDA approach when the assumption of Gaussian unimodal class-conditional distributions is not respected. Suppose that the two a priori probabilities are identical, but that the class conditional distribution of the green class is a mixture of two Gaussians. The positions of 1000 points randomly drawn from the two class-conditional distributions together with the LDA decision boundary computed by (7.2.66) are plotted in Figure 7.9.

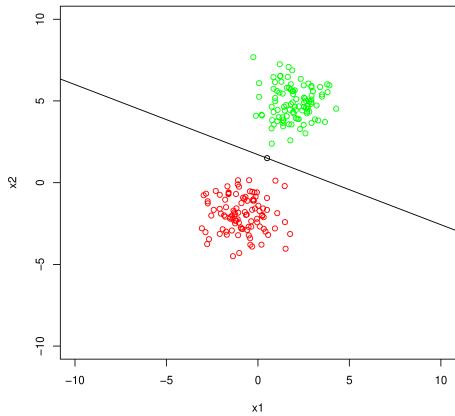


Figure 7.8: Binary classification problem: distribution of inputs and linear decision boundary

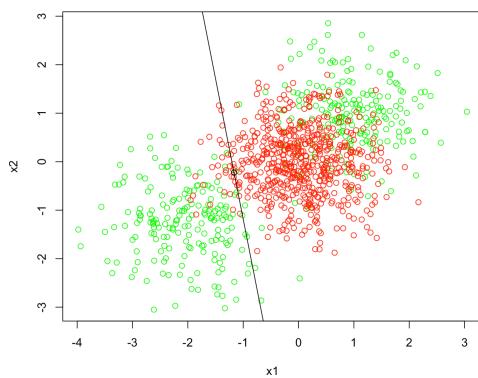


Figure 7.9: Binary classification problem where one class distribution is bimodal: distribution of inputs and linear decision boundary. Since the classification task is not linearly separable, the LDA classifier performs poorly.

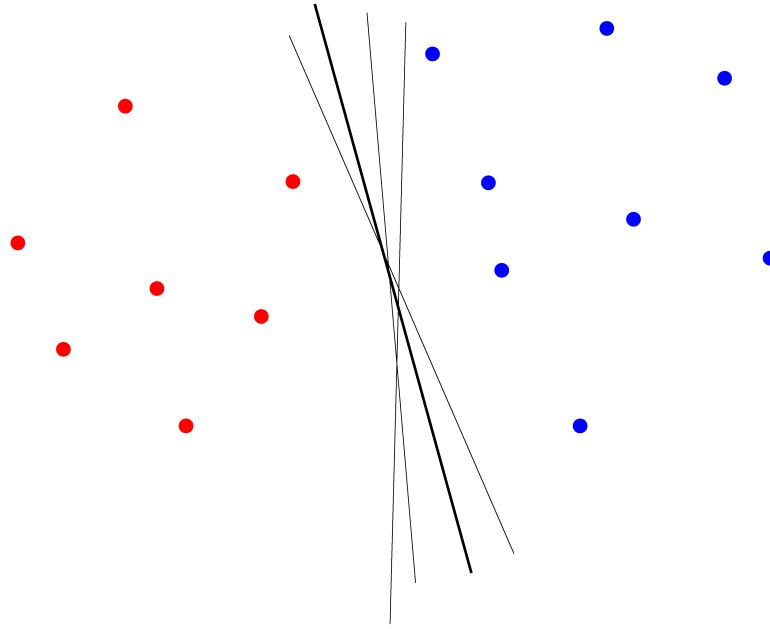


Figure 7.10: Several hyperplanes separating the two classes (blue and red).

7.2.1.3 LDA parameter identification

In a real setting, we do not have access to the quantities μ_k , Σ and $\text{Prob}\{\mathbf{y} = c_k\}$ to compute the boundary (7.2.66). Before applying the discrimination rule above, we need to estimate those quantities from the dataset D_N :

$$\widehat{\text{Prob}}\{\mathbf{y} = c_k\} = \frac{N_k}{N} \quad (7.2.68)$$

$$\hat{\mu}_k = \frac{\sum_{i:y_i=c_k} x_i}{N_k} \quad (7.2.69)$$

$$\hat{\Sigma} = \frac{\sum_{k=1}^K \sum_{i:y_i=c_k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T}{N - K} \quad (7.2.70)$$

where N_k is the number of observations labelled with the class c_k and (7.2.70) is also known as *pooled covariance* [96].

7.2.2 Perceptrons

Consider a binary classification task (Figure 7.10) where the two classes are denoted by $+1$ and -1 . The previous section presented a technique to separate input data by a linear boundary by making assumptions on the class conditional densities and their covariances. In a generic setting, however, the problem is ill-posed and there are infinitely many possible *separating hyperplanes* (Figure 7.10) characterised by the equation

$$\beta_0 + x^T \beta = 0 \quad (7.2.71)$$

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

$$(x^{(1)} - x^{(2)})^T \beta = 0$$

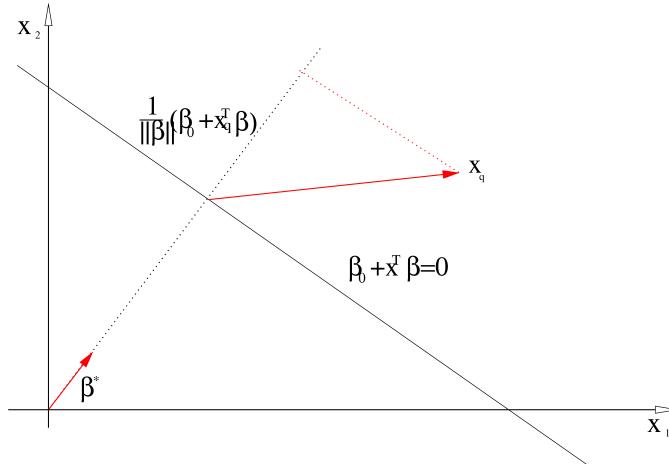


Figure 7.11: Bi-dimensional space ($n = 2$): vector β^* normal to the hyperplane and distance of a point from an hyperplane.

the vector normal to the hyperplane (Figure 7.11) is given by

$$\beta^* = \frac{\beta}{\|\beta\|}$$

- The signed distance of a point x to the hyperplane (Figure 7.11) is called the *geometric margin* and is given by

$$\beta^{*T}(x - x_0) = \frac{x^T \beta - \beta x_0^T}{\|\beta\|} = \frac{1}{\|\beta\|}(x^T \beta + \beta_0)$$

A *perceptron* is a classifier that uses the sign of the linear combination $h(x, \hat{\beta}) = \hat{\beta}_0 + \hat{\beta}^T x$ to perform classification [95]. The class returned by a perceptron for a given input x_q is

$$\begin{cases} 1 & \text{if } \hat{\beta}_0 + x_q^T \hat{\beta} = \hat{\beta}_0 + \sum_{j=1}^n x_{qj} \hat{\beta}_j > 0 \\ -1 & \text{if } \hat{\beta}_0 + x_q^T \hat{\beta} = \hat{\beta}_0 + \sum_{j=1}^n x_{qj} \hat{\beta}_j < 0 \end{cases}$$

In other terms the decision rule is given by

$$h(x) = \operatorname{sgn}(\hat{\beta}_0 + x^T \hat{\beta}) \quad (7.2.72)$$

For all well classified points in the training set the following relation holds

$$\gamma_i = y_i(x_i^T \hat{\beta} + \hat{\beta}_0) > 0 \quad (7.2.73)$$

where the quantity γ_i is called the *functional margin* of the pair $\langle x_i, y_i \rangle$ with respect to the hyperplane (7.2.71). Misclassifications in the training set occur when

$$\begin{cases} y_i = 1 & \text{but } \hat{\beta}_0 + \hat{\beta}^T x_i < 0 \\ y_i = -1 & \text{but } \hat{\beta}_0 + \hat{\beta}^T x_i > 0 \end{cases} \Leftrightarrow y_i(\hat{\beta}_0 + \hat{\beta}^T x_i) < 0$$

The parametric identification step of a perceptron learning procedure aims at finding the values $\{\hat{\beta}, \hat{\beta}_0\}$ that minimise the quantity

$$\text{SSE}_{\text{emp}}(\hat{\beta}, \hat{\beta}_0) = - \sum_{i \in \mathcal{M}} y_i(x_i^T \hat{\beta} + \hat{\beta}_0)$$

where \mathcal{M} is the subset of misclassified points in the training set. Note that this quantity is nonnegative and proportional to the distance of the misclassified points to the hyperplane. Since the gradients are

$$\frac{\partial \text{SSE}_{\text{emp}}(\hat{\beta}, \hat{\beta}_0)}{\partial \hat{\beta}} = - \sum_{i \in \mathcal{M}} y_i x_i, \quad \frac{\partial \text{SSE}_{\text{emp}}(\hat{\beta}, \hat{\beta}_0)}{\partial \hat{\beta}_0} = - \sum_{i \in \mathcal{M}} y_i$$

a batch gradient descent minimisation procedure (Section 6.6.2.3) or the online version (Section 6.6.3) 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 [95]:

- When the data are separable, there are many possible solutions, and which one is found depends on the initialisation 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 minimisation can be very slow.

R script

The script `Linear/hyperplane.R` visualises the evolution of the separating hyperplane during the perceptron learning procedure. We invite the reader to run the script for different numbers of points and different data distributions (e.g. by changing the mean and the variance of the 2D gaussians).

A possible solution to the separating hyperplane problem has been proposed by the SVM technique.

7.2.3 Support vector machines

This technique relies on an optimisation approach to compute the separating hyperplane.

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 attaining such maximum is known as a *maximal margin hyperplane*.

The SVM approach [170] 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 maximising the distance to the closest point from both classes. 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 modelled as the optimisation problem

$$\max_{\beta, \beta_0} C \tag{7.2.74}$$

$$\text{subject to } \frac{1}{\|\beta\|} y_i (x_i^T \beta + \beta_0) \geq C \quad \text{for } i = 1, \dots, N \tag{7.2.75}$$

where the constraint ensures that all the points are at least a distance C from the decision boundary defined by β and β_0 . The SVM parametric identification step seeks the largest C that satisfies the constraints and the associated parameters.

Since the hyperplane (7.2.71) is equivalent to the original hyperplane where the parameters β_0 and β have been multiplied by a constant, we can set $\|\beta\| = 1/C$. The maximisation problem can be reformulated in a minimisation form

$$\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2 \quad (7.2.76)$$

$$\text{subject to } y_i(x_i^T \beta + \beta_0) \geq 1 \quad \text{for } i = 1, \dots, N \quad (7.2.77)$$

where the constraints impose a margin around the linear decision of thickness $1/\|\beta\|$. This is a convex optimisation problem [37] where the primal Lagrangian is

$$L_P(\beta, \beta_0) = \frac{1}{2} \|\beta\|^2 - \sum_{i=1}^N \alpha_i [y_i(x_i^T \beta + \beta_0) - 1] \quad (7.2.78)$$

and $\alpha_i \geq 0$ are the Lagrangian multipliers.

Setting the derivatives of $L_P(\beta, \beta_0)$ wrt β and β_0 to zero we obtain:

$$\beta = \sum_{i=1}^N \alpha_i y_i x_i, \quad 0 = \sum_{i=1}^N \alpha_i y_i \quad (7.2.79)$$

Substituting these in the primal form (7.2.78) we obtain

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k x_i^T x_k \quad (7.2.80)$$

subject to $\alpha_i \geq 0$.

The dual optimisation problem, where the number of parameters is equal to N , is

$$\begin{aligned} \max_{\alpha} & \left\{ \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k x_i^T x_k \right\} = \\ & \max_{\alpha} \left\{ \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k \langle x_i, x_k \rangle \right\} \end{aligned} \quad (7.2.81)$$

subject to

$$0 = \sum_{i=1}^N \alpha_i y_i, \quad \alpha_i \geq 0, \quad i = 1, \dots, N$$

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 $\langle x_i, x_k \rangle, i = 1, \dots, N, k = 1, \dots, N$. This boils down to the computation of the Gram matrix

$$G = X X^T \quad (7.2.82)$$

If such matrix is positive definite, the optimisation (7.2.81) is an instance of quadratic programming (Appendix C.3) and the solution is global and unique. For an interesting least-squares version of SVM we refer to the work of [162].

Since the optimal solution must satisfy the complementarity Karush-Kuhn-Tucker (KKT) condition [37] stating that the active constraints must have a zero multiplier, it follows

$$\alpha_i [y_i(x_i^T \beta + \beta_0) - 1] = 0, \quad i = 1, \dots, N$$

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 x_i 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 x_i 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 α and obtained β from (7.2.79), the term β_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, from (7.2.79), in the sparse form

$$h(x, \beta, \beta_0) = \text{sign}\left[\sum_{\text{support vectors}} y_i \alpha_i \langle x_i, x \rangle + \beta_0\right] \quad (7.2.83)$$

Sparseness 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 lie 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 (7.2.66)). It can be shown, also, that in the separable case

$$C = \frac{1}{\|\beta\|} = \frac{1}{\sqrt{\sum_{i=1}^N \alpha_i}} \quad (7.2.84)$$

R script

The R script `Linear/svm.R` considers a binary classification problem. It generates sets of separable data and builds a separating hyperplane by solving the problem (7.2.78). 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 7.12).

•

A modification of the formulation (7.2.74) occurs when we suppose that the classes are nonlinearly separable. In this case the dual problem (7.2.81) is unbounded. The idea in [171] is to maximise the margin but tolerating that some points will be misclassified. For each example $\langle x_i, y_i \rangle$ we define the *slack variable* ξ_i and we relax the constraints (7.2.75) into

$$\frac{1}{\|\beta\|} y_i (x_i^T \beta + \beta_0) \geq C(1 - \xi_i) \quad \text{for } i = 1, \dots, N \quad (7.2.85)$$

$$\xi_i \geq 0 \quad (7.2.86)$$

$$\sum_{i=1}^N \xi_i \leq \gamma \quad (7.2.87)$$

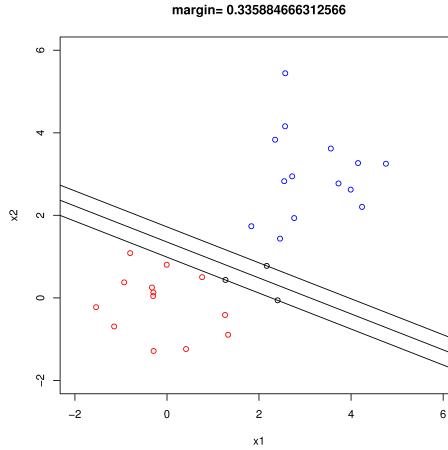


Figure 7.12: Maximal margin hyperplane for a binary classification task with the support vectors in black.

The value ξ_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 γ of $\sum_{i=1}^N \xi_i$ represents the maximum number of allowed misclassifications in the training set. It can be shown [95] that the maximisation (7.2.74) with the above constraints can be put in the equivalent quadratic form

$$\max_{\alpha} \left\{ \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k x_i^T x_k \right\} \quad (7.2.88)$$

subject to

$$0 = \sum_{i=1}^N \alpha_i y_i, \quad 0 \leq \alpha_i \leq \gamma, \quad i = 1, \dots, N$$

The decision function takes again the form (7.2.83) where β_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

$$C = \left(\sum_{k=1}^N \alpha_i \alpha_k y_i y_k x_i^T x_k \right)^{-1/2} \quad (7.2.89)$$

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 `Linear/svm.R` solves a non-separable problem by setting the boolean variable `separable` to FALSE. Figure 7.13 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).

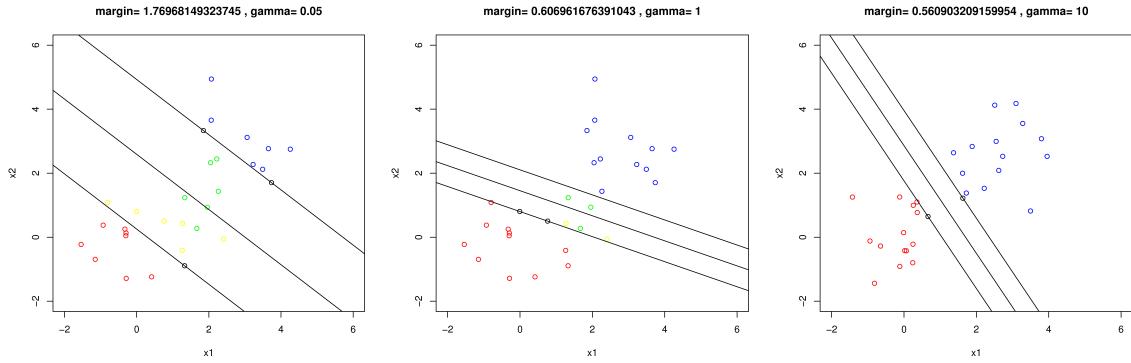


Figure 7.13: Maximal margin hyperplane for a non/separable binary classification task for different values of C : support vectors are in black, the slack points of the red class are in yellow and the slack points of the blue class are in green.

Once the value γ is fixed, the parametric identification in the SVM approach boils down to a quadratic optimisation problem for which a large number of methods and numerical software exists. The value γ plays the role of capacity hyperparameter 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 γ is varied through a wide range of values and assessed through a validation strategy.

7.3 Generalised linear models

Generalised linear models (GLM) extends the linear modelling approach in order to model other conditional distributions than Normal ones.

In this case the relation (7.1.17) is generalised to

$$g(E[\mathbf{y}|x]) = g(\mu) = x^T \beta \quad (7.3.90)$$

where g is a smooth monotonic function, known as the *link function* and the conditional distribution of \mathbf{y} belongs to an exponential family (D.3.5). In this formulation, the conventional linear case is a specific instance of GLM where the g is an identity function and the exponential density is Normal. The use of a non Normal exponential distribution extends the linear approach to target variables which represent counts (e.g. Poisson) or whose range is constrained (e.g. Beta).

Given an input/output dataset D_N the log-likelihood function (Section 3.8) is

$$\begin{aligned} l_N(\beta) &= \sum_{i=1}^N \log p(y_i|x_i) = \\ &= \sum_{i=1}^N [y_i \theta_i(\beta) - b_i(\theta_i(\beta))] / a_i(\phi) + c_i(y_i, \phi) \end{aligned} \quad (7.3.91)$$

If we partially differentiate $l_N(\beta)$ w.r.t. β and set the expressions to zero we obtain

$$\sum_{i=1}^N \frac{y_i - \mu_i}{V(\mu_i)} \frac{\partial \mu_i}{\partial \beta_j} = 0, \quad \forall j$$

where $\text{Var}[y] = V(\mu)\phi$. If the link function is an identity and the terms $V(\mu_i)$ are fixed, those equations boil down to conventional least-squares. Otherwise an iterative procedure, known as Iteratively Re-weighted Least Squares (IRLS) is adopted. For more details we refer the reader to [181]

7.4 Conclusion

In this chapter we considered input/output regression problems where the relationship 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,
- the statistical properties of the estimator can be easily defined,
- a 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 relationship is often unknown, and only a limited number of observations is available. For this reason, machine learning proposed a number of nonlinear approaches to address nonlinear tasks.

7.5 Exercises

1. Consider an input/output regression task where $n = 1$, $E[y|x] = \sin(x)$ and $p(y|x) \sim \mathcal{N}(\sin(x), 1)$. Let $N = 100$ be the size of the training set and consider a quadratic loss function.

Let the class of hypothesis be $h_M(x) = \alpha_0 + \sum_{m=1}^M \alpha_m x^m$.

1. Estimate the parameter by least-squares.
2. Compute the error by leave-one-out and by using the PRESS statistic.
3. Plot the empirical error as a function of the degree M for $M = 0, 1, \dots, 7$.
4. Plot the leave-one-out error as a function of the degree M for $M = 0, 1, \dots, 7$.
2. Consider a univariate linear regression problem. Write a R script which, using Monte Carlo simulation, validates the formula (7.1.9) for at least three regression tasks differing in terms of
 - parameters β_0, β_1 ,
 - variance σ^2 ,
 - number N of observations.
3. Consider a univariate linear regression problem. Write a R script which, using Monte Carlo simulation, validates the formula (7.1.10) for at least three regression tasks differing in terms of
 - parameters β_0, β_1 ,
 - variance σ^2 ,
 - number N of observations.
4. Consider a univariate linear regression problem. Write a R script which, using Monte Carlo simulation, shows that the least-squares estimates of β_0 and β_1 minimise the quantity (7.1.12) for at least three regression tasks differing in terms of

- parameters β_0, β_1 ,
- variance σ^2 ,
- number N of observations.

5. Consider a regression task with input \mathbf{x} and output \mathbf{y} . Suppose we observe the following training set

X	Y
0.1	1
0	0.5
-0.3	1.2
0.2	1
0.4	0.5
0.1	0
-1	1.1

1. Fit a linear model to the dataset.
2. Trace the data and the linear regression function on graph paper.
3. Are the two variables positively or negatively correlated?

Hint:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix} \Rightarrow A^{-1} = \frac{1}{a_{11}a_{22} - a_{12}^2} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{12} & a_{11} \end{bmatrix}$$

Solution:

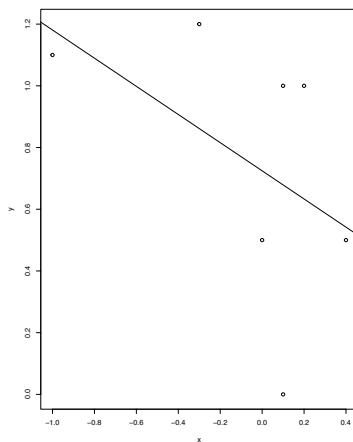
1. Once we set $X = \begin{bmatrix} 1 & 0.1 \\ 1 & 0 \\ 1 & -0.3 \\ 1 & 0.2 \\ 1 & 0.4 \\ 1 & 0.1 \\ 1 & -1 \end{bmatrix}$ we have

$$X'X = \begin{bmatrix} 7.0 & -0.50 \\ -0.5 & 1.31 \end{bmatrix}$$

and

$$\beta = (X'X)^{-1}X'Y = \begin{bmatrix} 0.725 \\ -0.456 \end{bmatrix}$$

2.



3. Since $\hat{\beta}_1 < 0$ the two variables are negatively correlated.
6. Let us consider the dependency where the conditional distribution of \mathbf{y} is

$$\mathbf{y} = 1 - x + x^2 - x^3 + \mathbf{w}$$

and $\mathbf{w} \sim N(0, \sigma^2)$ and $\sigma = 0.5$. Suppose that $x \in \mathbb{R}$ takes the values `seq(-1, 1, length.out = N)` (with $N = 50$).

Consider the family of regression models

$$h^{(m)}(x) = \beta_0 + \sum_{j=1}^m \beta_j x^j$$

where p denote the number of weights of the polynomial model $h^{(m)}$ of degree m .

Let $\widehat{\text{MISE}}_{\text{emp}}^{(m)}$ denote the least-squares empirical risk and MISE the mean integrated empirical risk. By using Monte Carlo simulation and for $m = 0, \dots, 6$

- plot $E[\widehat{\text{MISE}}_{\text{emp}}^{(m)}]$ as a function of p ,
- plot $\text{MISE}^{(m)}$ as a function of p ,
- plot the difference $E[\widehat{\text{MISE}}_{\text{emp}}^{(m)}] - \text{MISE}^{(m)}$ as a function of p and compare it with the theoretical result seen during the class.

For a single observed dataset:

- plot $\widehat{\text{MISE}}_{\text{emp}}^{(m)}$ as a function of the number of model parameters p ,
- plot PSE as a function of p ,
- discuss the relation between

$$\arg \min_m \widehat{\text{MISE}}_{\text{emp}}^{(m)}$$

and

$$\arg \min_m \text{PSE}^{(m)}$$

Solution: See the file `Exercise2.pdf` in the directory `gbcode/exercises` of the companion R package `gbcode` (Appendix G).

Chapter 8

Nonlinear approaches

This chapter will present several algorithms proposed in machine learning literature to deal with nonlinear regression and nonlinear classification tasks. 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 generalisation with effective learning procedures. The presentation of these techniques could be organised 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 the literature are the linear models discussed in the previous chapter, generalised linear models and neural networks which will be presented in Section 8.1.1.

A nice property of global modelling is that, even for huge datasets, the storage of a parametric model requires a small amount of 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 of a nonlinear optimisation problem (see Equation 5.2.7) which is not analytically tractable due to the numerous local minima and for which only a sub-optimal 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, alternatives to global modelling techniques, as the *divide-and-conquer* approach, gained popularity in the modelling community. 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.

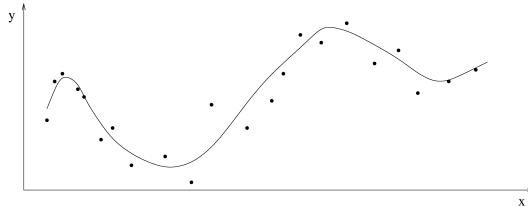


Figure 8.1: A global model (solid line) which fits the training set (dotted points) for a learning problem with one input variable (x-axis) and one output variable (y-axis).

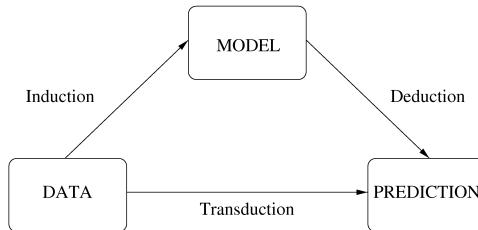


Figure 8.2: Function estimation (model induction + model evaluation) vs. value estimation (direct prediction from data).

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 modelling approach, which was first proposed in the nonparametric statistical 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 modelling problems (Section 8.1.3).

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 modelling* techniques (Section 8.1.11), 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 modelling 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 modelling techniques take an alternative approach, defined as *transduction* by Vapnik [170] (Fig. 8.2). They focus on approximating the function only

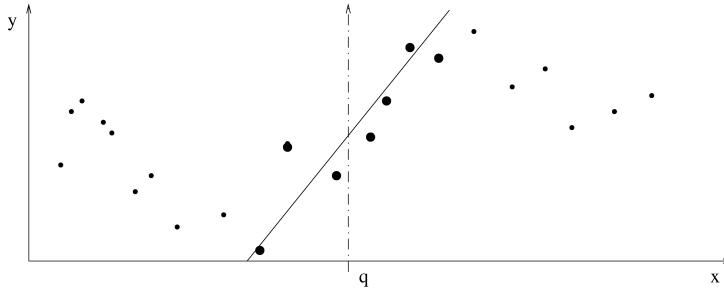


Figure 8.3: Local modelling of the input/output relationship between the input variable x and the output variable y , on the basis of a finite set of observations (dots). The value of the variable y for $x = q$ is returned by a linear model (solid line) which fits the training points in a neighbourhood of the query point (bigger dots).

in the neighbourhood 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 modelling case. At the same time, local modelling requires only simple approximators, e.g. constant and/or linear, to model the dataset in a neighbourhood of the query point. An example of local linear modelling 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 modelling approach: memory-based reasoning [158], case-based reasoning [115], local weighted regression [46], nearest neighbour [49], just-in-time [51], lazy learning [5], exemplar-based, instance based [4],... These approaches are also called *nonparametric* in the literature [93, 156], since they relax the assumptions on the form of a regression function, and let the data search for a suitable function that describes the available data well.

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* (5.4.21) 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 $E[\mathbf{w}] = 0$. Suppose that we collect a *training set* $\{\langle x_i, y_i \rangle : i = 1, \dots, N\}$ with $x_i = [x_{i1}, \dots, x_{in}]^T$, generated according to the model (5.4.21). 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 *Doppler* function

$$f(x) = 20\sqrt{x(1-x)} \sin\left(2\pi \frac{1.05}{x+0.05}\right) \quad (8.1.1)$$

distorted by additive Gaussian noise \mathbf{w} with unit variance.

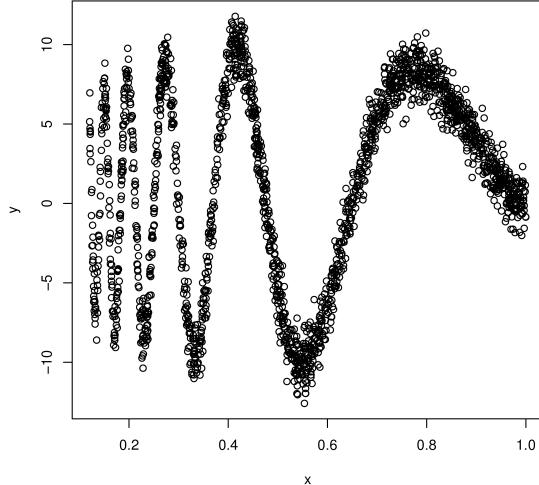


Figure 8.4: Training set obtained by sampling uniformly in the input domain of a Doppler function distorted with Gaussian noise.

The training set is made of $N = 2000$ points obtained by sampling the input domain $\mathcal{X} = [0.12, 1]$ through a uniform random distribution (Fig. 8.4). This stochastic dependency and the related training dataset (see R script `Nonlinear/doppler.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 ANNs (FNNs) have been applied to a wide range of prediction tasks in such diverse fields as speech recognition, financial prediction, image compression, adaptive industrial control.

8.1.1.1 Feed-forward architecture

Feed-forward NNs have a *layered* architecture, with each layer comprising one or simpler 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. FNNs are generally implemented with an additional node, called the *bias¹ unit*, in all layers except the output layer. This

¹Note that this has nothing to do with the estimator bias concept. In neural network literature, bias is used to denote the intercept term

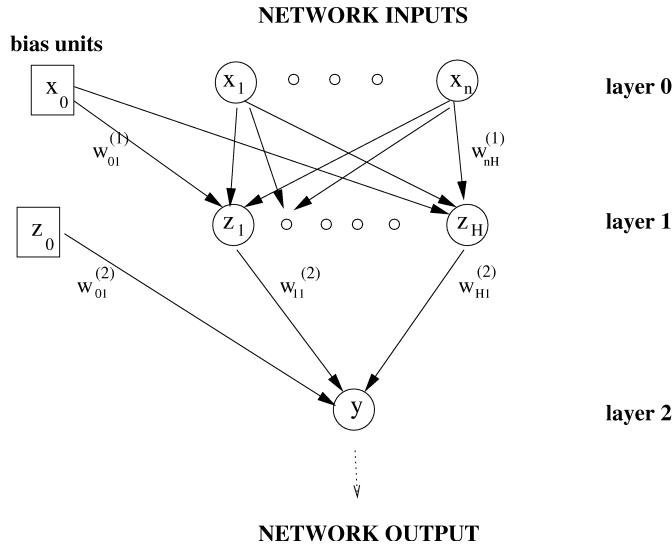


Figure 8.5: Two-layer feed-forward NN

node plays the role of the intercept term β_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 ($l = 1, \dots, L$) 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,
- $z_v^{(l)}$, $v = 1, \dots, H^{(l)}$ the output of the v th hidden node of the l th layer,
- $z_0^{(l)}$ denote the bias for the l , $l = 1, \dots, L$ layer.
- Let $H^{(0)} = n$ and $z_v^{(0)} = x_v$, $v = 0, \dots, n$.

For $l \geq 1$ the output of the v th, $v = 1, \dots, H^{(l)}$, 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

$$a_v^{(l)} = \sum_{k=1}^{H^{(l-1)}} w_{kv}^{(l)} z_k^{(l-1)} + w_{0v}^{(l)} z_0^{(l-1)}, \quad v = 1, \dots, H^{(l)}$$

and then by transforming the sum using an *activation* function to give

$$z_v^{(l)} = g^{(l)}(a_v^{(l)}), \quad v = 1, \dots, H^{(l)}$$

The activation function $g^{(l)}(\cdot)$ is typically a nonlinear transformation like the *logistic* or *sigmoid* function

$$g^{(l)}(z) = \frac{1}{1 + e^{-z}} \tag{8.1.2}$$

whose derivative is

$$g'(z) = \frac{e^{-z}}{(1 + e^{-z})^2} \quad (8.1.3)$$

Common alternatives are the *hyperbolic tangent function*

$$\tanh(z) = \frac{\exp^{2z} - 1}{\exp^{2z} + 1}$$

and the *rectified linear unit (ReLU)*

$$\text{ReLU}(z) = \max(z, 0) \quad (8.1.4)$$

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

$$z_k = g^{(1)} \left(\sum_{j=1}^n w_{jk}^{(1)} x_j + w_{0k}^{(1)} x_0 \right), \quad k = 1, \dots, H$$

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 an FNN represents the set of parameters α_N introduced in Section 5.1 when the hypothesis function $h(\cdot)$ is modelled 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 back-propagation algorithm which will be discussed in the following section.

The number H of hidden nodes represents the complexity s in Equation (5.8.49). 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 8.1.1.4).

8.1.1.2 Back-propagation

Back-propagation is an algorithm which, once the number of hidden nodes H is given, estimates the weights $\alpha_N = \{w^{(l)}, l = 1, \dots, L\}$ on the basis of the training set D_N . It is a gradient-based algorithm which aims to minimise the non-convex cost function

$$\text{SSE}_{\text{emp}}(\alpha_N) = \sum_{i=1}^N (y_i - \hat{y}_i)^2 = \sum_{i=1}^N (y_i - h(x_i, \alpha_N))^2$$

where $\alpha_N = \{w^{(l)}, l = 1, \dots, L\}$ is the set of weights.

The back-propagation algorithm exploits the network structure and the differentiable nature of the activation functions in order to compute the gradient recursively.

The simplest (and least effective) back-prop algorithm is an iterative gradient descent which is based on the iterative formula

$$\alpha_N(k+1) = \alpha_N(k) - \eta \frac{\partial \text{SSE}_{\text{emp}}(\alpha_N(k))}{\partial \alpha_N(k)} \quad (8.1.5)$$

where $\alpha_N(k)$ is the weight vector at the k th iteration, η is the learning rate which indicates the relative size of the change in weights and for each $w \in \alpha_N$

$$\frac{\partial \text{SSE}_{\text{emp}}}{\partial w} = -2 \sum_{i=1}^N (y_i - \hat{y}(x_i)) \frac{\partial \hat{y}(x_i)}{\partial w} \quad (8.1.6)$$

The backprop algorithm needs the derivatives (or more generally the Jacobian (Appendix B.7)) of SSE_{emp} w.r.t. to each weight $w \in \alpha_N$: since the terms $(y_i - \hat{y}(x_i))$ in (8.1.6) are easy to be computed, in the following sections we will mainly focus on how to derive $\frac{\partial \hat{y}}{\partial w}$.

The weights are initialised 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 SSE_{emp} is guaranteed. More effective versions of the algorithm are based on the Levenberg-Marquardt algorithm (Section 6.6.2.6). Note that this algorithm presents all the typical drawbacks of the gradient-based procedures discussed in Section 6.6.4, like slow convergence, local minima convergence, sensitivity to the weights initialisation.

Example: single-input single-output FNN

In order to better illustrate how the derivatives are computed in (8.1.6), let us consider a simple single-input (i.e. $n = 1$) single-output neural network with one hidden layer ($L = 2$), two hidden nodes ($H^{(1)} = 2$) 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 $\alpha_N = [w_{11}^{(1)}, w_{12}^{(1)}, w_{11}^{(2)}, w_{21}^{(2)}]$. Let us see how we can derive the terms $\frac{\partial \hat{y}}{\partial w}$ in (8.1.6).

As far as the weights $\{w_{11}^{(2)}, w_{21}^{(2)}\}$ of the hidden/output layer are concerned, from the chain rule of differential calculus (Appendix B.7) we derive

$$\frac{\partial \hat{y}(x)}{\partial w_{v1}^{(2)}} = \frac{\partial g}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial w_{v1}^{(2)}} = g'(a_1^{(2)}(x)) z_v(x), \quad v = 1, \dots, 2 \quad (8.1.7)$$

where g' is in (8.1.3).

As far as the weights $\{w_{11}^{(1)}, w_{12}^{(1)}\}$ of the input/hidden layer

$$\frac{\partial \hat{y}(x)}{\partial w_{1v}^{(1)}} = \frac{\partial g}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_v} \frac{\partial z_v}{\partial a_v^{(1)}} \frac{\partial a_v^{(1)}}{\partial w_{1v}^{(1)}} = g'(a_1^{(2)}(x)) w_{v1}^{(2)} g'(a_v^{(1)}(x)) x \quad (8.1.8)$$

where the term $g'(a_1^{(2)}(x))$ has been already obtained during the computation of (8.1.7). The computation of the derivatives with respect to the weights of the lower layers relies on 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 back-propagation of numerical terms from the upper layer to the lower layers that justifies the name of the procedure.

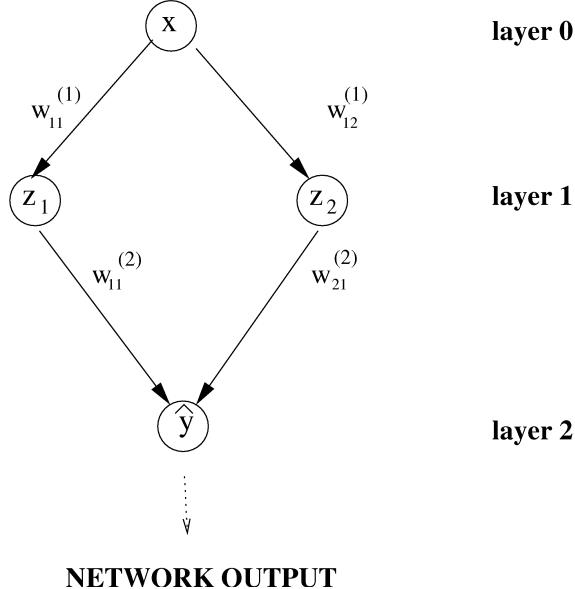
NETWORK INPUT

Figure 8.6: Single-input single-output neural network with one hidden layer, two hidden nodes and no bias units.

8.1.1.3 Back-propagation in matrix notation

For a generic FNN architecture with a single output and L layers the backpropagation algorithm may be decomposed in the following steps:

1. For the single output unit compute

$$\delta_1^{(L)} = \frac{\partial \hat{y}}{\partial a_1^{(L)}} = g'(a_1^{(L)})$$

2. For the layers $l = 1, \dots, L - 1$ and for each node $v = 1, \dots, H^{(l)}$ compute

$$\delta_v^{(l)} = \left(\sum_{j=1}^{H^{(l+1)}} w_{vj}^{(l+1)} \delta_j^{(l+1)} \right) g'(a_v^{(l)}) \quad (8.1.9)$$

3. The partial derivatives terms in (8.1.6) for $k = 1, \dots, H^{(l-1)}$ and $v = 1, \dots, H^{(l)}$ are

$$\frac{\partial \hat{y}}{\partial w_{kv}^{(l)}} = z_k^{(l-1)} \delta_v^{(l)} \quad (8.1.10)$$

It is possible to rewrite the above steps in matrix notation by denoting with $W^{(l)} = [w_{kv}^{(l)}]$ the matrix of size $[H^{(l-1)}, H^{(l)}]$ containing the weights connecting the $(l-1)$ th layer to the the l th.

Then (8.1.9) becomes

$$D^{(l)} = (W^{(l+1)} D^{(l+1)}) \odot G^{(l)} \quad (8.1.11)$$

where \odot denotes the entrywise Hadamard product of vectors, $D^{(l)}$ is the $[H^{(l)}, 1]$ vector containing the $\delta_v^{(l)}$, $v = 1, \dots, H^{(l)}$ terms and $G^{(l)}$ is the $[H^{(l)}, 1]$ vector containing the $g'(a_v^{(l)})$, $v = 1, \dots, H^{(l)}$ terms. The equation (8.1.10) may be written

$$\frac{\partial \hat{y}}{\partial W^{(l)}} = Z^{(l-1)}(D^{(l)})^T \quad (8.1.12)$$

where $Z^{(l-1)}$ is the $[H^{(l-1)}, 1]$ vector contains the output of the $(l-1)$ layer and $Z^{(0)}$ corresponds to the input vector.

Let us validate the formulas (8.1.11) and (8.1.12) in the case of the SISO example in Figure 8.6. Since

$$\begin{aligned} W^{(2)} &= \begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \end{bmatrix}, \quad W^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \end{bmatrix} \\ D^{(2)} &= g'(a_1^{(2)}), \quad G^1 = \begin{bmatrix} g'(a_1^{(1)}) \\ g'(a_2^{(1)}) \end{bmatrix} \\ D^{(1)} &= (W^2 D^2) \odot G^1 = \begin{bmatrix} w_{11}^{(2)} g'(a_1^{(2)}) g'(a_1^{(1)}) \\ w_{11}^{(2)} g'(a_1^{(2)}) g'(a_2^{(1)}) \end{bmatrix} \end{aligned}$$

we obtain

$$\frac{\partial \hat{y}}{\partial W^{(2)}} = \begin{bmatrix} \frac{\partial \hat{y}(x)}{\partial w_{11}^{(2)}} \\ \frac{\partial \hat{y}(x)}{\partial w_{21}^{(2)}} \end{bmatrix} = Z^{(1)}(D^{(2)})^T = \begin{bmatrix} g'(a_1^{(2)}) z_1 \\ g'(a_1^{(2)}) z_2 \end{bmatrix}$$

which coincides with (8.1.7) and

$$\begin{aligned} \frac{\partial \hat{y}}{\partial W^{(1)}} &= \begin{bmatrix} \frac{\partial \hat{y}(x)}{\partial w_{11}^{(1)}} & \frac{\partial \hat{y}(x)}{\partial w_{12}^{(1)}} \end{bmatrix} = \\ &= Z^{(0)}(D^{(1)})^T = \begin{bmatrix} w_{11}^{(2)} g'(a_1^{(2)}) g'(a_1^{(1)}) x & w_{11}^{(2)} g'(a_1^{(2)}) g'(a_2^{(1)}) x \end{bmatrix} \end{aligned}$$

which coincides with (8.1.8).

R example

Tensorflow [1] is an open-source library developed by Google² which had a great success in recent years as flexible environment for building and training of neural network architectures. In particular this library provides automatic differentiation functionalities to speed up the backpropagation implementation.

The script `Nonlinear/tf_nn.R` uses the R `tensorflow` package (wrapper over the python library) to compute the derivatives (8.1.8) and (8.1.7) for the network in Figure 8.6 and checks that the TensorFlow result coincides with the one derived analytically.

•

R example

The FNN learning algorithm for a single-hidden layer architecture is implemented by the R library `nnet`. The script `Nonlinear/nnet.R` shows the prediction accuracy for different number of hidden nodes (Figure 8.7 and Figure 8.8).

•

²<https://www.tensorflow.org>

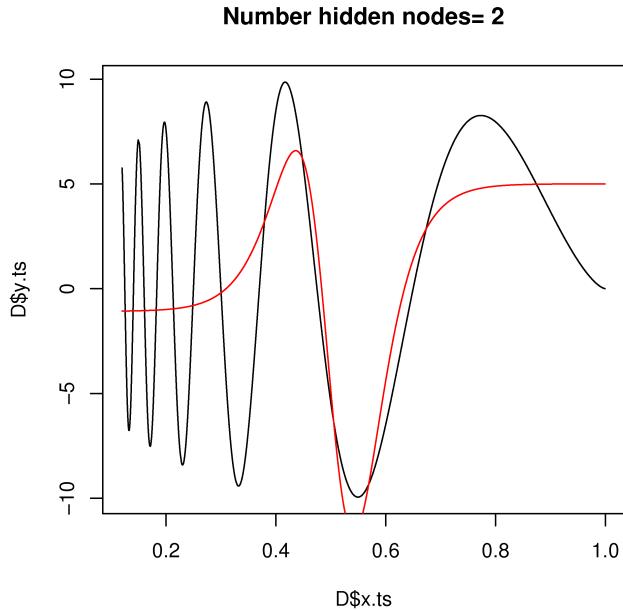


Figure 8.7: Neural network fitting with $s = 2$ hidden nodes. The red continuous line represents the neural network estimation of the Doppler function (8.1.1).

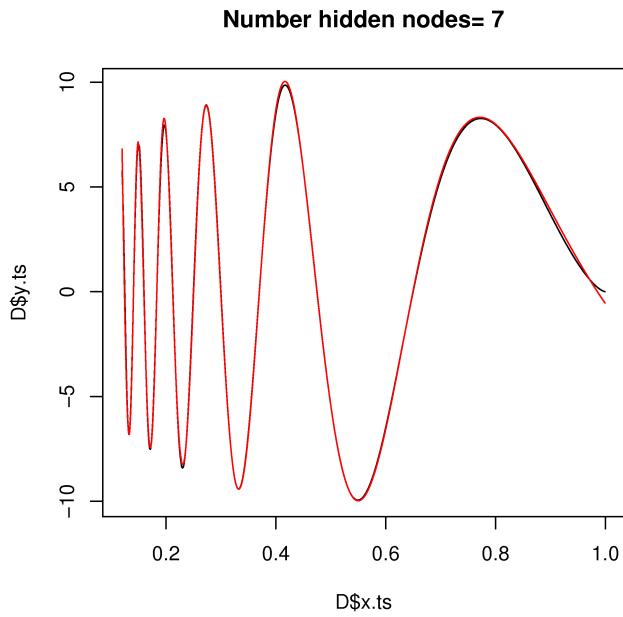


Figure 8.8: Neural network fitting with $s = 7$ hidden nodes. The continuous red line represents the neural network estimation of the Doppler function(8.1.1).

8.1.1.4 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 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 observations and a generic nonlinear mapping.

In practice, the choice of the number of hidden nodes requires a structural identification procedure (Section 6.8) which assesses and compares several different FNN architectures before choosing the ones expected to be the closest to the optimum. Cross-validation techniques or regularisation strategies based on complexity-based criteria (e.g. the weight-decay in Section 6.8.2.5) 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 generalisation error.

Consider a dataset $D_N = \{x_i, y_i\}$, $i = 1, \dots, N$ 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 relationship

$$y = x_1^2 + 4 \log(|x_2|) + 5x_3$$

where x_i is the i th component of the vector x . Consider as nonlinear 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 $N_{ts} = 50$ examples. Let us train the neural network on the whole training set by using the R script `Nonlinear/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 α_N is obtained by the parametric identification step. However, if we test $h(\cdot, \alpha_N)$ on the test set we obtain

$$\widehat{\text{MISE}}_{\text{ts}} = \frac{1}{N_{ts}} \sum_{i=1}^{N_{ts}} (y_i - h(x_i, \alpha_N))^2 = 22.41$$

This neural network is *seriously overfitting* the dataset. The empirical error is a *very bad* estimate of the MISE.

Now we perform 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.

•

8.1.2 From shallow to deep learning architectures

Until 2006, FNNs with more than two layers have been rarely used in literature because of poor training and large generalisation errors. The common belief was that the solutions returned by deep neural networks were worse solutions than the ones obtained with shallower networks. This was mainly attributed to two aspects: i) gradient-based training of deep supervised FNN gets stuck in local minima or plateaus, and ii) the higher the number of layers in a neural network, the smaller the impact of the back-propagation on the first layers.

However, an incredible resurgence of the domain occurred from 2006 on when some teams (notably the Bengio team in Montreal, the Hinton team in Toronto and the Le Cun team in Facebook)³ were able to show that some adaptation of the FNN algorithm could bring a remedy to the above-mentioned problems and lead to major accuracy improvements with respect to other learning machines. In particular deep architectures (containing up to hundreds of layers) showed a number of advantages:

- some highly nonlinear functions can be represented much more compactly with deep architectures than with shallow ones,
- the XOR parity function for n -bit inputs⁴ can be coded by a feed-forward neural network with $O(\log n)$ hidden layers and $O(n)$, neurons, while a feed-forward neural network with only one hidden layer needs an exponential number of the same neurons to perform the same task,
- DL allows automatic generation and extraction of new features in large dimensional tasks with spatial dependency and location invariance,
- DL allows easy management of datasets where inputs and outputs are stored in tensors (multidimensional matrices),
- DL relies on the learning of successive layers of increasingly meaningful representations of input data (*layered representation learning*) and is a powerful automatic alternative to time-consuming human crafted feature engineering,
- portions of DL pre-trained networks may be reused for similar, yet different, tasks (transfer learning) or calibrated in online learning pipelines (continuous learning),
- iterative gradient optimisation (Section 6.6.3) is a very effective manner of ingesting a huge amount of data in large networks.
- new activation functions and weight-initialisation schemes (e.g. layer-wise pretraining) improve the training process.

³Yoshua Bengio, Geoffrey Hinton, and Yann LeCun were awarded the 2018 Turing Award, known as the Nobel Prize of computing

⁴This is a canonical challenge in classification where the target function is a boolean function whose value is one if and only if the n -dimensional input vector has an odd number of ones.

Also, new network architectures were proposed, like *auto-encoders* or *convolutional networks*. An auto-encoder is a multi-input multi-output neural network that maps its input to itself. It has a hidden layer that describes a code used to represent the input and is composed of two parts: an encoder function and a decoder that produces a reconstruction. They can be used for dimensionality reduction or compression (if the number of hidden nodes is smaller than the number of inputs).

Convolutional networks are biologically inspired architectures imitating the processing of cortical cells. They are ideal for taking into consideration local and spatial correlation and consist of a combination of convolution, pooling and normalisation steps applied to inputs taking the generic form of tensors. The convolution phase applies a number of filters with shared weights to the same image. It ensures translation invariance since the weights depend on spatial separation and not on absolute positions. Pooling is a way to take large images and shrink them down while preserving the most important information in them. This step allows the creation of new features as the combination of previous level features. The normalisation ensures that every negative value is set to zero.

Those works had such a major impact on theoretical and applied research that nowadays, deep learning is a de facto synonymous of the entire machine learning domain and more generally of AI. This comeback has been supported by a number of headlines in the news like the success of a deep learning solution in the ImageNet Large-Scale Visual Recognition Competition (2012): (bringing down the state-of-the-art error rate from 26.1% to 15.3%) or the DL program AlphaGo, developed by the company DeepMind, beating the no.1 human GO player. Other impressive applications of deep learning are near-human-level speech recognition, near-human-level handwriting transcription, autonomous cars (e.g. traffic sign recognition, pedestrian detection), image segmentation (e.g. face detection), analysis of particle accelerator data in physics, prediction of mutation effects in bioinformatics and machine translation (LSTM model sequence-to-sequence relationships). The breakthroughs of DL in the AI community have been acknowledged by the attribution of the 2019 ACM Turing prize to Bengio, Hinton and Le Cun.

The domain is so large and rich that the most honest recommendation of the author is to refer the reader, for more details, to seminal books [86] and articles [116] authored by the pioneers in this domain. Nevertheless, we would like to make a number of pedagogical considerations about the role of deep learning with respect to other learning machines:

- DL models are not faithful models of the brain.
- the astonishingly success of DL makes of it a privileged approach in recent years, but definitely, it should not be considered a machine learning panacea.
- DL, like all machine learning techniques, relies on a number of hyper-parameters which affect its capacity, its bias/variance trade-off and the expected generalisation power. The setting of those parameters has a major impact on the generalisation power. An important factor of the recent success of deep networks learning is the effective integration of computational strategies already adopted in other learning approaches, like regularisation, averaging, resampling.
- the success of DL, though fulgurant, if often restricted to some specific perceptual tasks, e.g. convolutional networks have been explicitly designed to process data that come in the form of multiple arrays (1D for signals and sequences, including language; 2D for images or audio spectrograms; and 3D for video or volumetric images.)

- there is no evidence that representation learning is by default a better strategy than feature engineering: it is surely less biased but very probably more variant.
- DL is particularly successful in tasks where it is possible to collect (and label) huge amounts of examples: nevertheless, there are still a number of challenging tasks where the number of examples is typically low or scarce (e.g. bioinformatics or time series forecasting),
- the success of DL has been amplified by the advent of fast parallel graphics processing units (GPUs), tensor processing units (TPU) and related libraries (e.g. TensorFlow, Keras, PyTorch) that are convenient to program and allow researchers to train networks 10 or 20 times faster.
- any assumption of a priori superiority of DL over other techniques for a given learning task is more often due to hype consideration than to a scientific attitude that should instead rely on the validation of a number of alternative strategies and the pondering of different criteria (accuracy, computational cost, energy consumption, interpretability).

Exercise

The script `Nonlinear/keras.regr.R` compares a Keras [45] implementation of a DNN and a Random Forest (Section 9.4) in a very simple nonlinear regression task where a single input out of n is informative about the target. The default setting of the DNN is very disappointing in terms of NMSE accuracy (6.10.38) with respect to the Random Forest. We invite the reader to spend some time performing DNN model selection (e.g. by changing the architecture, tuning the number of layers and/or the number of nodes per layers) or increasing the amount of training points to bring the DNN accuracy closer to the RF one. Is that easy? Is that fast? What is your opinion?

•

8.1.3 From global modelling to divide-and-conquer

Neural networks are a typical example of *global* modelling. 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 best approximate the whole data distribution (Chapter 5).

The *divide-and-conquer* paradigm originates from the idea of relaxing the global modelling 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 modelling* 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 modelling and control problems [106]. The following sections will introduce some examples of modular techniques.

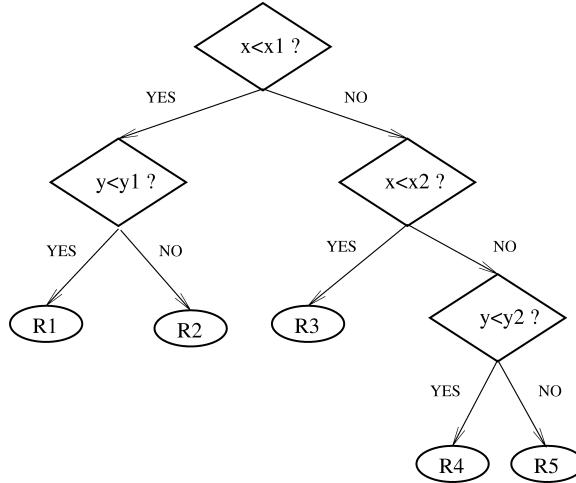


Figure 8.9: A binary decision tree.

8.1.4 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 [145] and the C4 [146] algorithms. Similar techniques were introduced in statistics by Breiman *et al.* [39], 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 characterise 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 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^{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; according to 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 an 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 $x_q < x_1$ and $y_q > y_1$. The predicted output will be $y_q = h_2(q, \alpha_2)$ where α_2 is the vector of parameters of the model localised 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

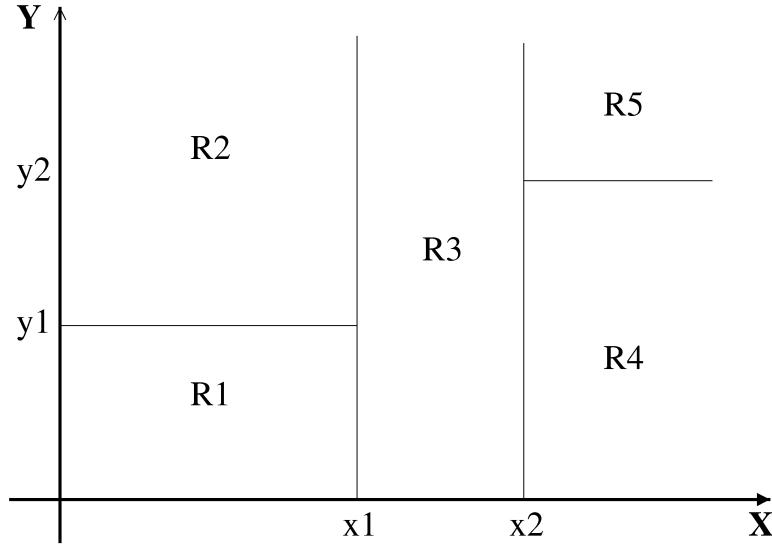


Figure 8.10: Input space partitioning induced on the input space by the binary tree in Fig. 8.9

crisp boundaries. In the case of linear terms, the resulting approximator is instead a piecewise linear model.

8.1.4.1 Learning in Regression Trees: 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 `Nonlinear/tree.R` shows the prediction accuracy for different minimum number of observations per leaf. (Figure 8.11 and Figure 8.12).

8.1.4.2 Learning in Regression Trees: structural identification

This section presents a summary of the CART procedure [39] 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 6.8.1.

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

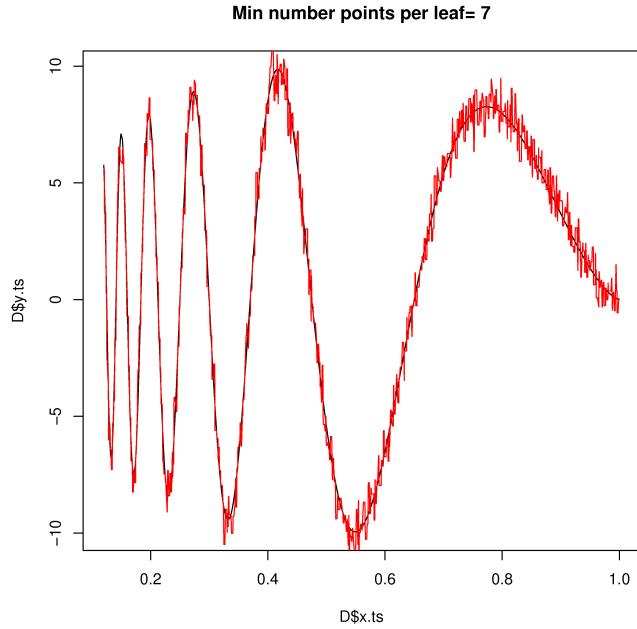


Figure 8.11: Regression tree fitting with a minimum number of points per leaf equal to $s = 7$.

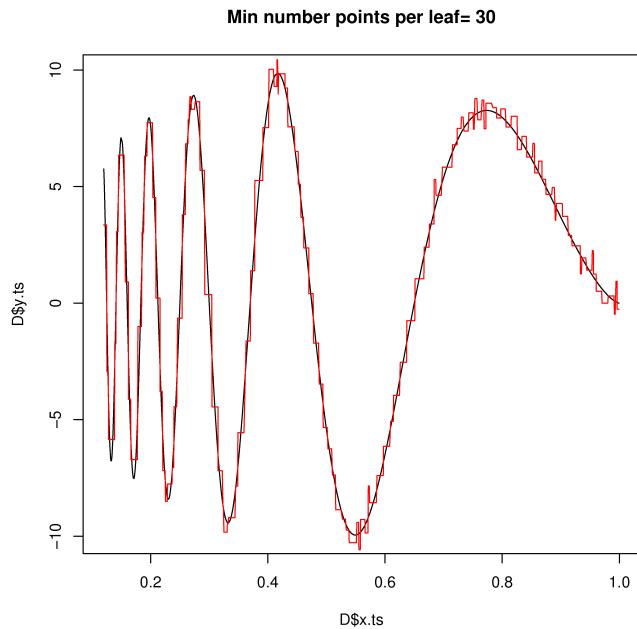


Figure 8.12: Regression tree fitting with a minimum number of points per leaf equal to $s = 30$.

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 :

$$R_{emp}(t) = \min_{\alpha_t} \sum_{i=1}^{N(t)} L(y_i, h_t(x_i, \alpha_t)) \quad (8.1.13)$$

For any possible split s of node t into the two children t_r and t_l , we define the quantity

$$\begin{aligned} \Delta E(s, t) &= R_{emp}(t) - (R_{emp}(t_l) + R_{emp}(t_r)) \\ &\text{with } N(t_r) + N(t_l) = N(t) \end{aligned} \quad (8.1.14)$$

that represents the change in the empirical error due to a further partition of the dataset. The best split is the one that maximises the decrease ΔE

$$s^* = \arg \max_s \Delta E(s, t) \quad (8.1.15)$$

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 ΔE resulting from further splitting does not exceed a threshold value.

The tree returned by the growing procedure is typically too large and presents a serious risk of overfitting the dataset (Section 5.6). For that reason, a pruning procedure is often adopted.

- **Tree pruning.** Consider a fully expanded tree T_{max} characterised by L terminal nodes.

Let us introduce a complexity-based measure of the tree performance

$$R_\lambda(T) = R_{emp}(T) + \lambda|T| \quad (8.1.16)$$

where λ 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 λ we define with $T(\lambda)$ the tree structure which minimises the quantity (8.1.16).

The parameter λ is gradually increased in order to generate a sequence of tree configurations with decreasing complexity

$$T_L = T_{max} \supset T_{L-1} \supset \cdots \supset T_2 \supset T_1 \quad (8.1.17)$$

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 λ 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 λ_t which makes of it the minimiser of (8.1.16). For a generic subtree T_t this value must satisfy

$$R_{\lambda_t}(T_t) \leq R_{\lambda_t}(T) \quad (8.1.18)$$

that is

$$R_{emp}(T_t) + \lambda_t|T_t| \leq R_{emp}(T) + \lambda_t|T|$$

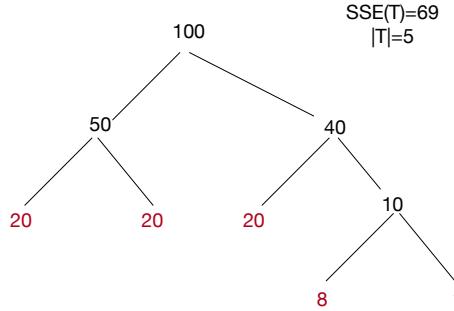


Figure 8.13: Regression tree before the pruning step

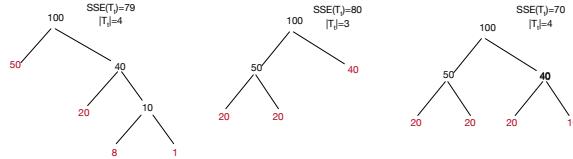


Figure 8.14: Regression tree after three different pruning steps

which means

$$\lambda_t \geq \frac{R_{emp}(T) - R_{emp}(T_t)}{|T| - |T_t|} \quad (8.1.19)$$

Hence, $\lambda_t = \frac{R_{emp}(T) - R_{emp}(T_t)}{|T| - |T_t|}$ makes of T_t the minimising tree. Therefore we choose among all the admissible subtrees T_t the one with the smallest right-hand term in Eq. (8.1.19). This implies a minimal increase in λ toward the next minimising tree.

At the end of the shrinking process, we have a sequence of candidate trees that have to be properly assessed 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 6.8.3.

An example of pruning step

Let us consider an example of pruning step. Suppose that the fully-grown tree in represented in Figure 8.13 where the values in the leaves denote the R_{emp} in terms of Sum of Squared Errors. Figure 8.14 represents three different pruning steps and the related changes in number of leaves and SSE. The corresponding values of $\frac{R_{emp}(T_t) - R_{emp}(T)}{|T| - |T_t|}$ for the three pruning steps (left to right) are

$$\frac{79 - 69}{5 - 4} = 10, \quad \frac{80 - 69}{5 - 3} = 5.5, \quad \frac{70 - 69}{5 - 4} = 1$$

The weakest link pruning (i.e. the first pruning to occur) is the one on the right.

•

Regression trees are a very easy-to-interpret representation of a nonlinear input/output mapping. However, these methods are characterised by a rough discontinuity at the

decision boundaries, which might bring undesired effects to the overall generalisation. Dividing the data by partitioning the input space shows typically small estimator bias but at the cost of 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 minimise the variance at the cost of an increased bias. A second solution is to make use of soft splits, allowing data to lie simultaneously in multiple regions, like in Basis Function Networks (BFN) which will be discussed in the following section. An averaging solution is proposed by Random Forests which will be discussed in Chapter 9.

8.1.5 Basis Function Networks

Basis Function Networks (BFN) are a family of modular architectures which are described by a *linear basis expansion*, i.e. the weighted linear combination

$$y = \sum_{j=1}^m \rho_j(x) h_j \quad (8.1.20)$$

where the weights are returned by the activation of m local nonlinear *basis functions* ρ_j and where the term h_j is the output of a generic module of the architecture.

The *basis* or *activation* function ρ_j is a function

$$\rho_j : \mathcal{X} \rightarrow [0, 1] \quad (8.1.21)$$

usually designed so that its value monotonically decreases towards zero as the input point moves away from its centre 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.

8.1.6 Radial Basis Functions

A well-known example of basis functions are the *Radial Basis Functions* (RBF) [143]. Each basis function in RBF takes the form of a kernel

$$\rho_j = K(x, c_j, B_j) \quad (8.1.22)$$

where c_j is the centre of the kernel and B_j is the bandwidth. An example of kernel functions is illustrated in Fig. 8.15. Other examples of kernel functions are available in Appendix F.1. Once we define with η_j the set $\{c_j, B_j\}$ of parameters of the basis function, we have

$$\rho_j = \rho_j(\cdot, \eta_j) \quad (8.1.23)$$

If the basis ρ_j have localised receptive fields and a limited degree of overlap with their neighbours, the weights h_j in Eq. (8.1.20) 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.

8.1.7 Local Model Networks

Local Model Networks (LMN) were first introduced by Johansen and Foss [106]. They are a generalised form of Basis Function Network in the sense that the constant

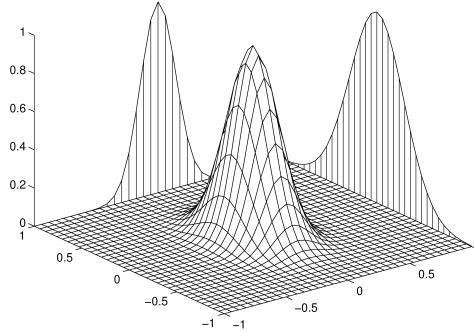


Figure 8.15: A Gaussian kernel function in a two-dimensional input space.

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

$$y = \sum_{j=1}^m \rho_j(x, \eta_j) h_j(x, \alpha_j) \quad (8.1.24)$$

where the ρ_j are constrained to satisfy

$$\sum_{j=1}^m \rho_j(x, \eta_j) = 1 \quad \forall x \in \mathcal{X} \quad (8.1.25)$$

This means that the basis functions form a *partition of unity* [129]. 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. 8.16 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. (8.1.24) can be of any form: linear, nonlinear, physical models or black-box parametric models.

Note that, in the case of local linear models

$$h_j(x, \alpha_j) = \sum_{i=1}^n a_{ji} x_i + b_j \quad (8.1.26)$$

where the vector of parameters of the local model is $\alpha_j = [a_{j1}, \dots, a_{jn}, b_j]$ and x_i is the i^{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

$$h_{lin}(x) = \sum_{j=1}^m \rho_j(x, \eta_j) \left(\sum_{i=1}^n a_{ji} x_i + b_j \right) \quad (8.1.27)$$

8.1.8 Neuro-Fuzzy Inference Systems

Fuzzy modelling consists of describing relationships between variables by means of *if-then* rules, such as

If x is high then y is low

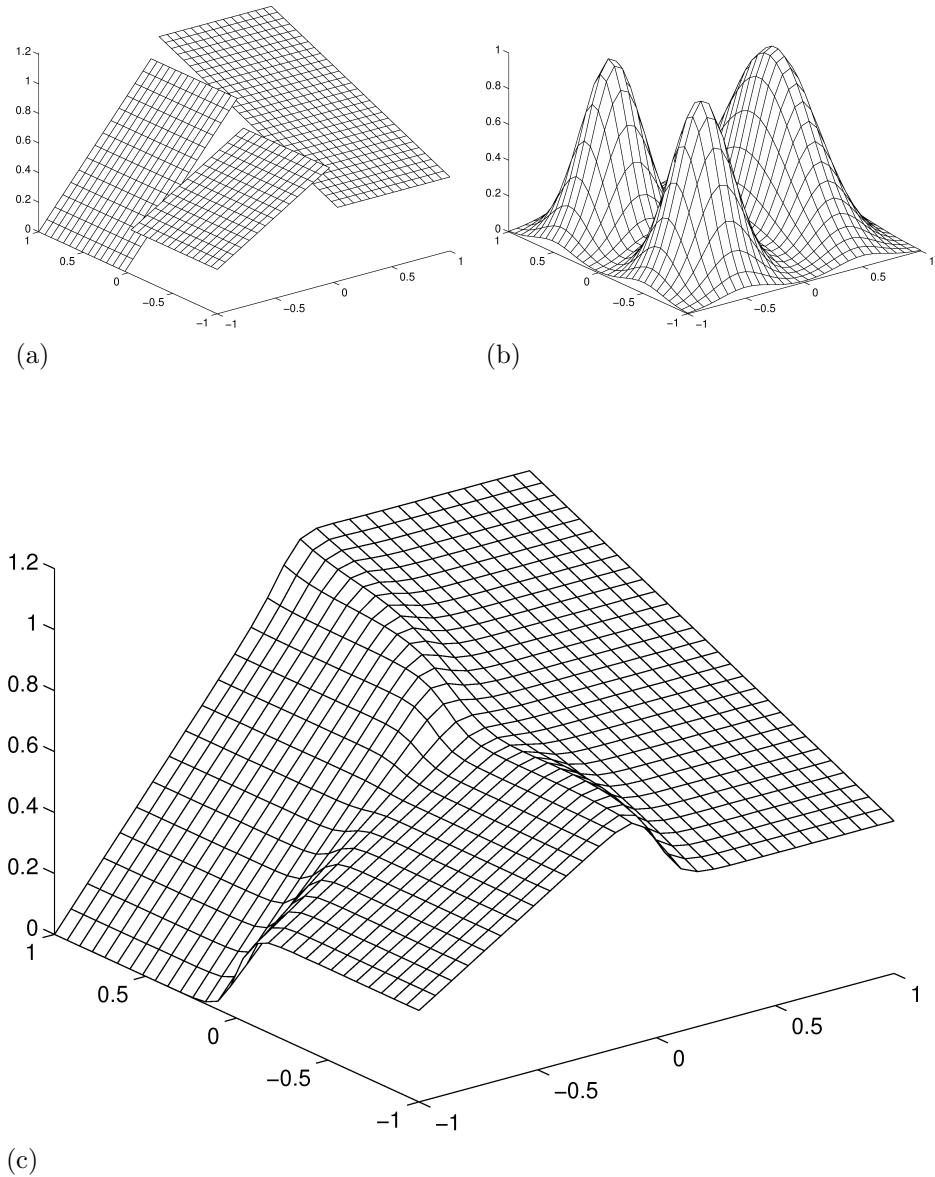


Figure 8.16: A Local Model Network with $m = 3$ local models: the nonlinear input/output approximator in (c) is obtained by combining the three local linear models in (a) according to the three basis functions in (b).

where the linguistic terms, as *high* and *low*, are described by fuzzy sets [183].

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 [11].

Here we will focus on the fuzzy architecture for nonlinear modelling introduced by Takagi and Sugeno [163]. A Takagi-Sugeno (TS) *fuzzy inference system* is a set of m fuzzy if-then rules having the form:

$$\begin{cases} \text{If } x_1 \text{ is } A_{11} \text{ and } x_2 \text{ is } A_{21} \dots \text{ and } x_n \text{ is } A_{n1} \text{ then } y = h_1(x_1, x_2, \dots, x_n, \alpha_1) \\ \dots \\ \text{If } x_1 \text{ is } A_{1m} \text{ and } x_2 \text{ is } A_{2m} \dots \text{ and } x_n \text{ is } A_{nm} \text{ then } y = h_m(x_1, x_2, \dots, x_n, \alpha_m) \end{cases} \quad (8.1.28)$$

The *antecedent* is defined as a fuzzy AND proposition where A_{kj} is a fuzzy set on the k^{th} premise variable defined by the membership function $\mu_{kj}: \mathfrak{R} \rightarrow [0, 1]$. 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

$$y = \frac{\sum_{j=1}^m \mu_j(x) h_j(x, \alpha_j)}{\sum_{j=1}^m \mu_j(x)} \quad (8.1.29)$$

where μ_j is the degree of fulfilment of the j^{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

$$\rho_j(x) = \frac{\mu_j(x)}{\sum_{j=1}^m \mu_j(x)} \quad (8.1.30)$$

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 [103]. For a thorough introduction to Neuro-Fuzzy architecture see [104] and the references therein. Further work on this subject was presented by the author in [21, 22, 31, 20].

8.1.9 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 6.8) 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 η_j of the basis functions (e.g. centre and width in the Gaussian case)

and the optimal set of parameters α_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.

8.1.9.1 Parametric identification: basis functions

The relationship between the model output and the parameters η_j of the basis function is typically nonlinear, hence, methods for nonlinear optimisation are currently employed. A typical approach consists in decomposing the identification procedure into two steps: first, an *initialisation* step, which computes the initial location and width of the basis functions, then a *nonlinear optimisation* procedure which uses the outcome $\eta_j^{(0)}$ of the previous step as initial value.

Since the methods for nonlinear optimisation have already been discussed in Section 6.6.2.2, here we will focus on the different initialisation techniques for Basis Function Networks.

One method for placing the centres of the basis functions is to locate them at the interstices of some coarse lattice defined over the input space [41]. 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 [126] 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 parameterised by a centre c_j and a width σ_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 a heuristic technique and finding a dataset to which this technique cannot be applied satisfactorily is not uncommon.

An alternative to K-means clustering for initialisation has been proposed in the Neuro-Fuzzy literature [11, 12]. The initialisation of the architecture is provided by a *hyperellipsoidal fuzzy clustering* procedure. This procedure clusters the data in the input/output domain, obtaining a set of hyperellipsoids which are a rough preliminary representation of the mapping. The parameters of the ellipsoids (eigenvalues) are used to initialise the parameters α_j of the consequent models, while the projection of their barycentres on the input domain determines the initial positions of the membership functions (see Fig. 8.17).

8.1.9.2 Parametric identification: local models

A common approach to the optimisation of the parameters α_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 optimisation technique is required (Section 6.6.2.2). Such a procedure is typically computationally expensive and does not guarantee the convergence to the global minimum.

However, in the case of local linear models (Eq. 8.1.26), the parametric identification can take advantage of linear techniques. Assume that the local models are linear, i.e.

$$h_j(x, \alpha_j) = h_j(x, \beta_j) = x^T \beta_j \quad (8.1.31)$$

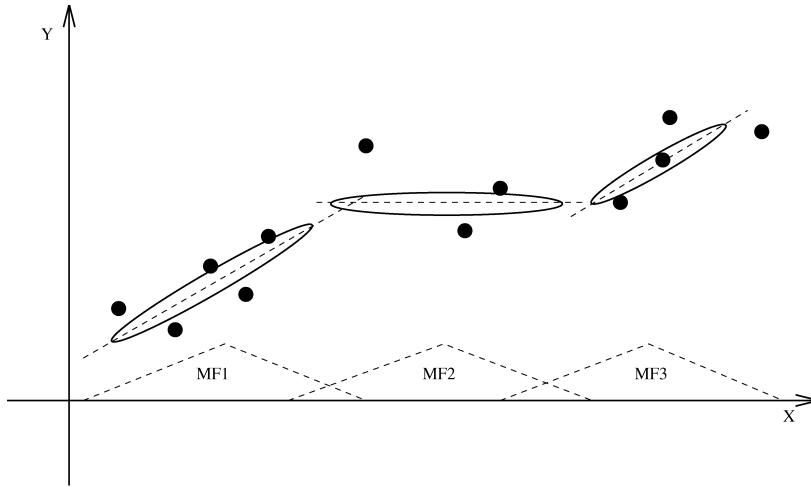


Figure 8.17: The hyperellipsoidal clustering initialisation procedure for a single-input single-output mapping. The training points (dots) are grouped in three ellipsoidal clusters after a procedure of fuzzy clustering in the input/output domain. The projection of the resulting clusters in the input domain (x -axis) determines the centre and the width of the triangular membership functions.

There are two possible variants for the parameter identification [129, 130]:

Local optimisation. The parameters of each local model are estimated independently.

A weighted least squares optimisation 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 $h_j(\cdot, \beta_j)$, $j = 1, \dots, m$, are then estimated using a set of locally weighted estimation criteria

$$J_j(\beta_j) = \frac{1}{N} (y - X\beta_j)^T Q_j (y - X\beta_j) \quad (8.1.32)$$

where Q_j is a $[N \times N]$ diagonal weighting matrix, having as diagonal elements the weights $\rho_j(x_1, \eta_j), \dots, \rho_j(x_N, \eta_j)$. The weight $\rho_j(x_i, \eta_j)$ represents the relevance of the i^{th} example of the training set in the definition of the j^{th} local model.

The locally weighted least squares estimate $\hat{\beta}_j$ of the local model parameter vector β_j is

$$\hat{\beta}_j = (X^T Q_j X)^{-1} X^T Q_j y \quad (8.1.33)$$

Global optimisation. 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 optimisation is a simple least-squares problem. We get the following regression model:

$$y = \sum_{j=1}^m \rho_j(x, \eta_j) x^T \beta_j = \Phi \Theta \quad (8.1.34)$$

where Φ is a matrix $[N \times (n + 1)m]$

$$\Phi = \begin{bmatrix} \Phi_1 \\ \vdots \\ \Phi_N \end{bmatrix} \quad (8.1.35)$$

with

$$\Phi_i = [\rho_1(x_i, \eta_j)x_i^T, \dots, \rho_m(x_i, \eta_j)x_i^T] \quad (8.1.36)$$

and Θ is a matrix $[(n+1)m \times 1]$

$$\Theta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_m \end{bmatrix} \quad (8.1.37)$$

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 behaviour of the target in the region described by ρ_j [130], the second approach disregards any qualitative interpretation by pursuing only a global prediction accuracy of the modular architecture.

8.1.9.3 Structural identification

The structure of a BFN is characterised 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 6.8, different BFN architectures with different numbers 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 generalisation 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 [21].

Note that BFN structural identification, unlike the parametric procedure described in Section 8.1.9.2, is a non/convex problem and cannot take advantage of 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 characterised by an iterative loop over different model structures as illustrated in the flow chart of Fig. 8.18.

8.1.10 From modular techniques to local modelling

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 optimisation. 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.

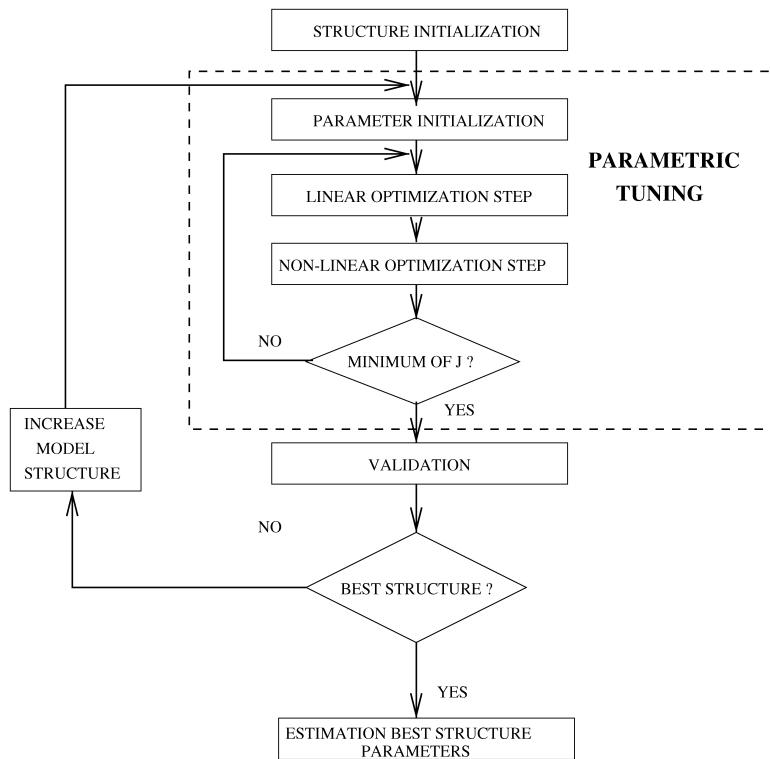


Figure 8.18: Flow chart of the BFN learning procedure. The learning procedure is made of two nested loops: the inner one (made of a linear and nonlinear step) is the parametric identification loop which minimises the empirical error J , the outer one searches for the model structure which minimises the validation criterion.

The learning problem for modular architectures is still a problem of function estimation formulated in terms of minimisation of the empirical risk over the whole training set. The modular configuration makes the minimisation 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 global optimisation is relaxed, the divide-and-conquer idea leads to the local modelling approach.

8.1.11 Local modelling

Local modelling 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 modelling 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 19th century. The early literature on smoothing by local fitting focused on one independent variable with equally spaced values. For a historical review of early work on local regression see [48].

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 [132] and Watson [175].

8.1.11.1 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 neighbourhood. The function K satisfies two conditions:

$$0 \leq K(x, q, B) \leq 1 \quad (8.1.38)$$

$$K(q, q, B) = 1 \quad (8.1.39)$$

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. 8.19)

$$K(x, q, B) = \begin{cases} 1 & \text{if } \|x - q\| < \frac{B}{2} \\ 0 & \text{otherwise} \end{cases} \quad (8.1.40)$$

and the soft threshold vicinity function (Fig. 8.20)

$$K(x, q, B) = \exp \left\{ -\frac{(x - q)^2}{B^2} \right\} \quad (8.1.41)$$

satisfy these requirements. Other examples of kernel functions are reported in Appendix F.1.

The Nadaraya-Watson kernel regression estimator is given by

$$h(q) = \frac{\sum_{i=1}^N K(x_i, q, B) y_i}{\sum_{i=1}^N K(x_i, q, B)} \quad (8.1.42)$$

where N is the size of 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 (8.1.42) is a simple moving average with equal weights: the estimate

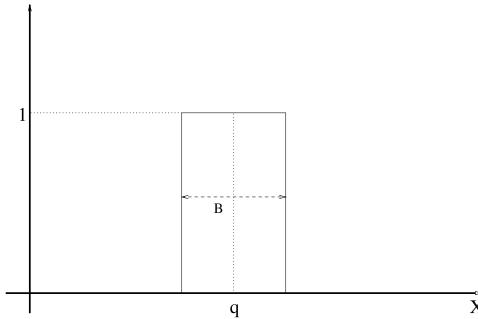


Figure 8.19: Hard-threshold kernel function.

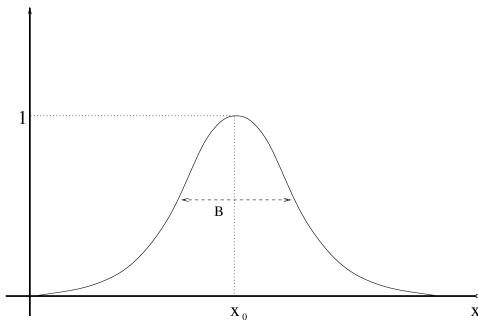


Figure 8.20: Soft-threshold kernel function.

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 $B \rightarrow \infty$ 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 becomes 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 extreme 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 5.6).

From a function approximation point of view, the Nadaraya-Watson estimator is a least-squares constant approximator. Suppose we want to approximate locally the unknown function $f(\cdot)$ by a constant θ . The local weighted least-squares estimate is

$$\hat{\theta} = \arg \min_{\theta} \sum_{i=1}^N w_i (y_i - \theta)^2 = \frac{\sum_{i=1}^N w_i y_i}{\sum_{i=1}^N w_i} \quad (8.1.43)$$

It follows that the kernel estimator is an example of locally weighted constant 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. 8.21 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

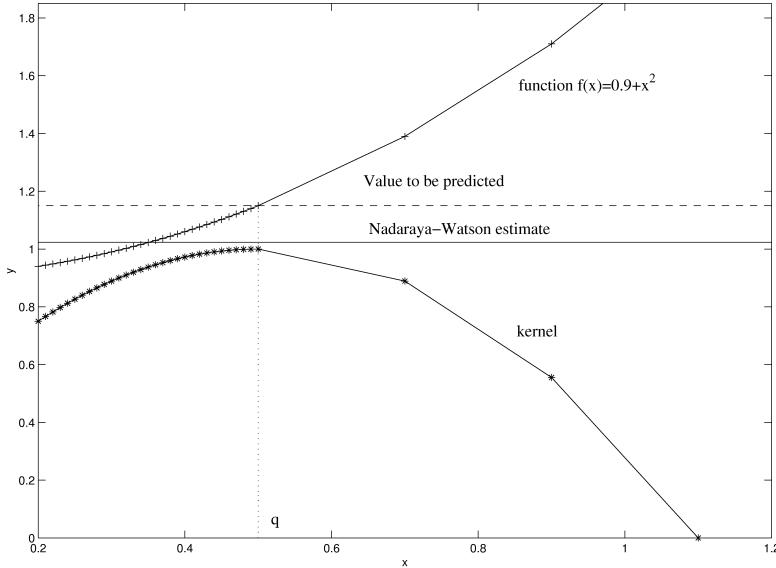


Figure 8.21: Effect of asymmetric data distribution on the Nadaraya-Watson estimator: the plot reports in the input/output domain the function $f = 0.9 + x^2$ to be estimated, the available points (crosses), the values of the kernel function (stars), the value to be predicted in $q = 0.5$ (dotted horizontal line) and the value predicted by the NW estimator (solid horizontal line).

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. 8.22 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.

8.1.11.2 Higher order local regression

Once the weakness of the local constant approximation was recognised, a more general local regression appeared in the late 1970's [46, 159, 109]. Work on local regression continued throughout the 1980's and 1990's, focusing on the application of smoothing to multidimensional problems [47].

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 π (e.g. random, fixed, highly clustered or nearly uniform). See in Fig. 8.23 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. 8.22 and Fig. 8.24).

8.1.11.3 Parametric identification in local regression

Given two variables $x \in \mathcal{X} \subset \mathbb{R}^n$ and $y \in \mathcal{Y} \subset \mathbb{R}$, let us consider the mapping $f: \mathbb{R}^n \rightarrow \mathbb{R}$, known only through a set of N examples $\{(x_i, y_i)\}_{i=1}^N$ obtained as follows:

$$y_i = f(x_i) + w_i, \quad (8.1.44)$$

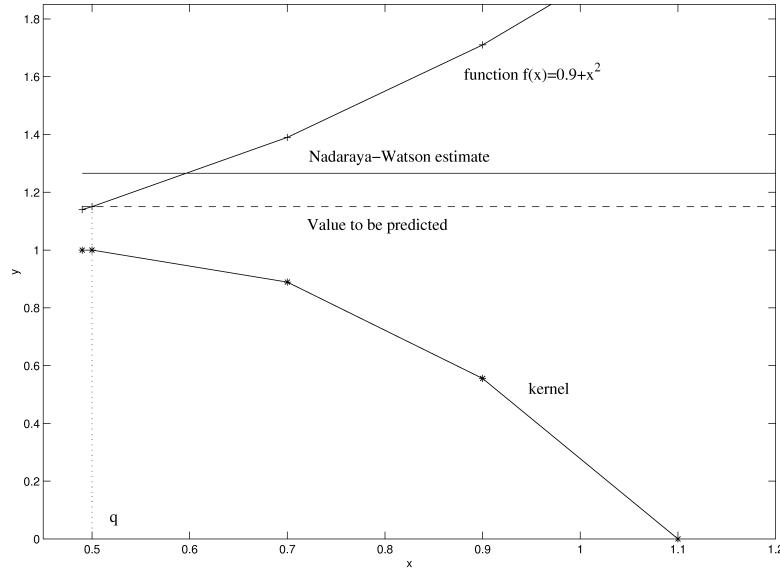


Figure 8.22: Effect of a boundary on the Nadaraya-Watson estimator: the plot reports in the input/output domain the function $f = 0.9 + x^2$ to be estimated, the available points (crosses), the values of the kernel function (stars), the value to be predicted in $q = 0.5$ (dotted horizontal line) and the value predicted by the NW estimator (solid horizontal line).

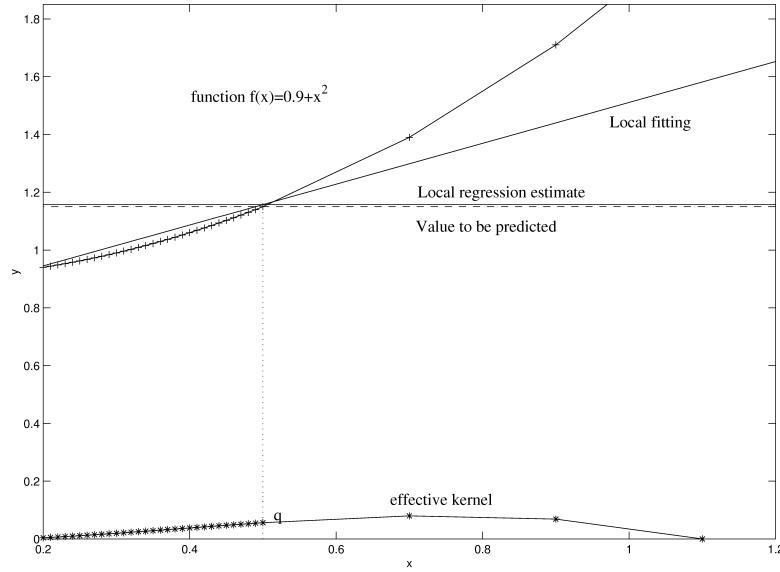


Figure 8.23: Local linear regression in asymmetric data configuration: the plot reports in the input/output domain the function $f = 0.9 + x^2$ to be estimated, the available points (crosses), the values of the effective kernel (stars), the local linear fitting, the value to be predicted in $q = 0.5$ (dotted horizontal line) and the value predicted by the local regression (solid horizontal line).

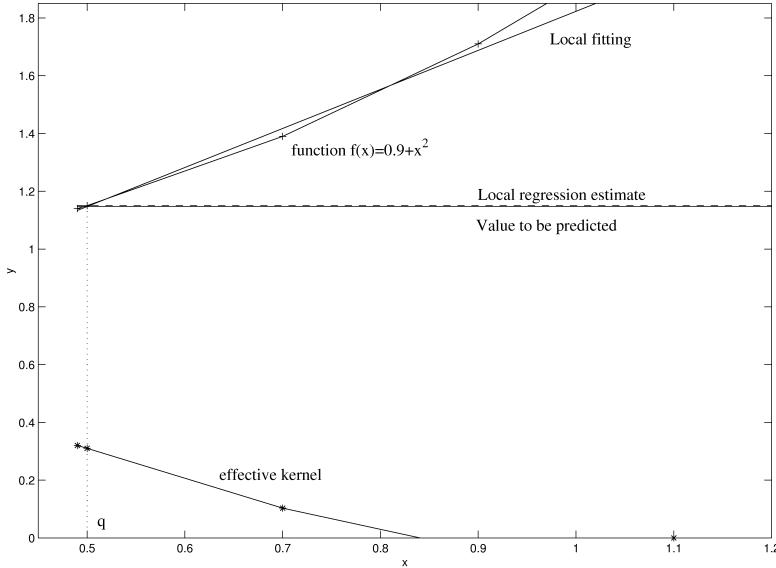


Figure 8.24: Local linear regression in boundary configurations: the plot reports in the input/output domain the function $f = 0.9 + x^2$ to be estimated, the available points (crosses), the values of the effective kernel (stars), the local linear fitting, the value to be predicted in $q = 0.5$ (dotted horizontal line) and the value predicted by the local regression (solid horizontal line).

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^{th} moment (Eq. (2.11.41)) 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 constant global 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 neighbourhood of q .

By using the Taylor's expansion truncated to the order p , a generic smooth regression function $f(\cdot)$ can be approximated by

$$f(x) \approx \sum_{j=0}^p \frac{f^{(j)}(q)}{j!} (x - q)^j \quad (8.1.45)$$

for x in a neighbourhood 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 neighbourhood of q can be obtained by solving the *locally weighted regression* (LWR)

$$\hat{\beta} = \arg \min_{\beta} \sum_{i=1}^N \left\{ (y_i - x_i^T \beta)^2 K(x_i, q, B) \right\}, \quad (8.1.46)$$

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 (8.1.46) can be written as

$$\hat{\beta} = \arg \min_{\beta} (y - X\beta)^T W (y - X\beta) \quad (8.1.47)$$

where X denotes the $[N \times (n+1)]$ input matrix whose i^{th} row is x_i^T , y is a $[N \times 1]$ vector whose i^{th} element is y_i and W is a $[N \times N]$ diagonal matrix whose i^{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:

$$\hat{\beta} = (X^T W^T W X)^{-1} X^T W^T W y = (Z^T Z)^{-1} Z^T v = P Z^T v, \quad (8.1.48)$$

where $Z = W X$, $v = W y$, 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:

$$\hat{y}_q = q^T \hat{\beta}. \quad (8.1.49)$$

8.1.11.4 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 localised 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.

8.1.11.5 The kernel function

Under the assumption that the data to be analysed are generated by a continuous mapping $f(\cdot)$, we want to consider positive kernel functions $K(\cdot, \cdot)$ that are peaked at $x = q$ and that decay smoothly to 0 as the distance between x and q increases. Examples of distance-based kernel functions are reported in Appendix F.1.

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 (8.1.40), 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 $|x - q| \rightarrow \infty$. This allows faster implementations, since points further from the query than the bandwidth can be ignored with no error.

8.1.11.6 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 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 [72]. In practice, this means that if the goal

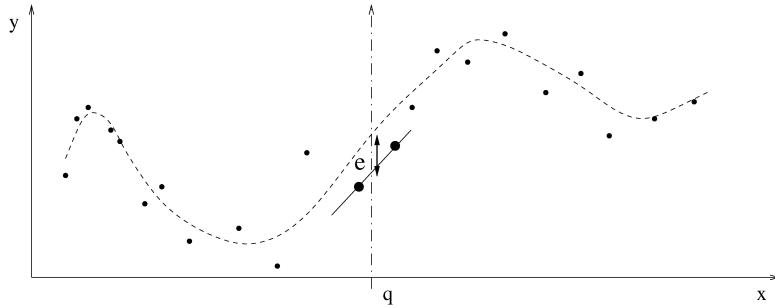


Figure 8.25: Too narrow bandwidth \Rightarrow overfitting \Rightarrow large prediction error e ..

of local polynomial regression is to estimate the value of the function in the query point (degree zero in the Taylor expansion (8.1.45)), 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 [48].

In the previous sections, we have 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 neighbourhood 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 [122] and in local regression by Cleveland and Loader [48]. 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

$$f_p(\cdot) = (1 - c)f_m(\cdot) + cf_{m+1}(\cdot) \quad (8.1.50)$$

We can choose a single mixing degree for all x or we can use an adaptive method by letting p vary with x .

8.1.11.7 The bandwidth

A natural question is how wide the local neighbourhood should be so that the local approximation (8.1.45) holds. This is equivalent to asking how large the bandwidth parameter should be in (8.1.40). 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 modelling bias small. However, since the number of data points falling in this local neighbourhood is also small, we cannot average out the noise and the variance of the prediction will be consequently large (Fig. 8.25).

On the other hand, if the bandwidth is too large, we could smooth excessively the data, then introducing a large modelling bias (Fig. 8.26). In the limit case of an infinite bandwidth, for example, a local linear model turns out to be a global linear fitting which, by definition, cannot take into account any type of nonlinearity.

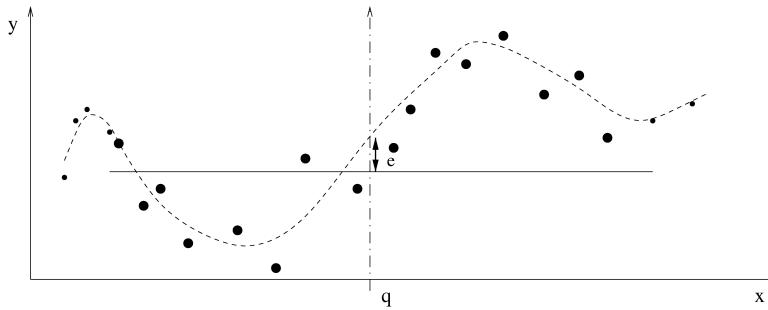


Figure 8.26: Too large bandwidth \Rightarrow underfitting \Rightarrow large prediction error e .

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 [107] 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 .

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 neighbour bandwidth selection where the bandwidth is set to be the distance between the query point and the k^{th} nearest point [159].
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 the *point-based* case where the bandwidth $B(x_i)$ is a function of the training point x_i , and the *uniform* case where B is constant.

A constant bandwidth is easy to interpret and can be sufficient if the unknown curve is not too wiggly, i.e. has a high smoothness. Such a bandwidth, however, fails to do a good job when the unknown curve has a rather complicated 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 levels of noise and to changes in the smoothness of the function. Fan and Gijbels [70] argue for point-based in favour of query-based local bandwidth selection mainly for computational efficiency reasons.

8.1.11.8 The distance function

The performance of any local method depends critically on the choice of the distance function $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$. In the following, we define some distance functions for ordered inputs:

Unweighted Euclidean distance

$$d(x, q) = \sqrt{\sum_{j=1}^n (x_j - q_j)^2} = \sqrt{(x - q)^T (x - q)} \quad (8.1.51)$$

Weighted Euclidean distance

$$d(x, q) = \sqrt{(x - q)^T M^T M (x - q)} \quad (8.1.52)$$

The unweighted distance is a particular case of the weighted case for M diagonal with $m_{jj} = 1$.

Unweighted L_p norm (Minkowski metric)

$$d(x, q) = \left(\sum_{j=1}^N |x_j - q_j|^p \right)^{\frac{1}{p}} \quad (8.1.53)$$

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.* [10] 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 [158, 94, 78].

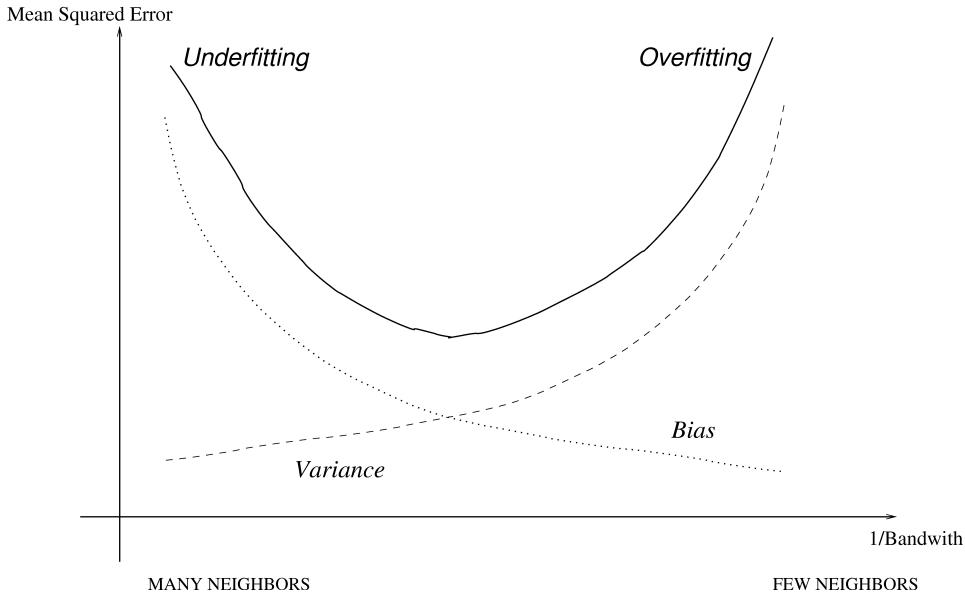
Point-based local distance functions. Each training point has an associated distance metric [158]. This is typical of classification problems where each class has an associated distance metric [3, 4].

8.1.11.9 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 optimised 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 a 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 [71] and in [92].



Data-driven estimation. It is a selection procedure which estimates the generalisation 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 modelling.

There are several ways in which data-driven methods can be used for structural identification. Atkeson *et al.* [10] distinguish between

Global tuning. The structural parameters are tuned by optimising a data driven assessment criterion on the whole data set. An example is the General Memory Based Learning (GMBL) described in [127].

Query-based local tuning. The structural parameters are tuned by optimising a data driven assessment criterion query-by-query. An example is the *lazy learning* algorithm proposed by the author and colleagues in [24, 33, 32].

Point-based local tuning. A different set of structural parameters is associated with each point of the training set.

R implementation

A local linear algorithm for regression is implemented by the R library `lazy` [23]. The script `Nonlinear/lazy.R` shows the prediction accuracy in the Doppler dataset for different number of neighbours. (Figure 8.27 and Figure 8.28).

•

8.1.11.10 Bias/variance decomposition of the local constant model

An interesting aspect of local models is that it is easy to derive an analytical expression of the bias/variance decomposition.

In the case of a constant local model, the prediction in q is

$$h(q, \alpha_N) = \frac{1}{k} \sum_{i=1}^k y_{[i]}$$

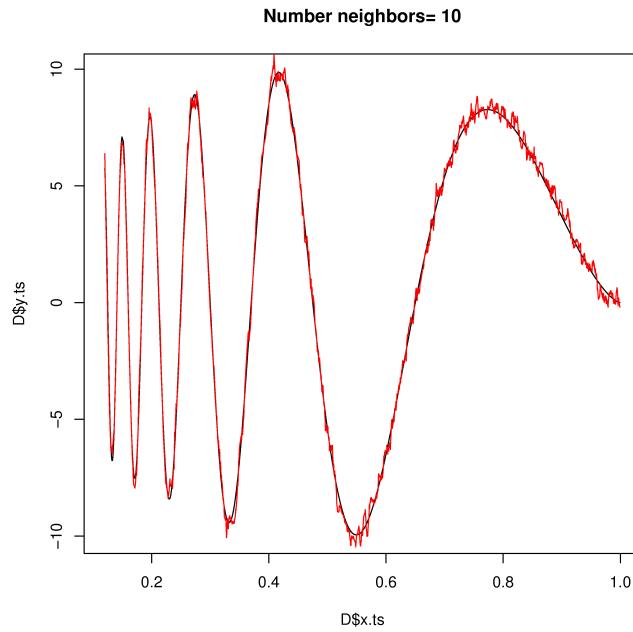


Figure 8.27: Locally linear fitting with a rectangular kernel and a bandwidth made of 10 neighbours.

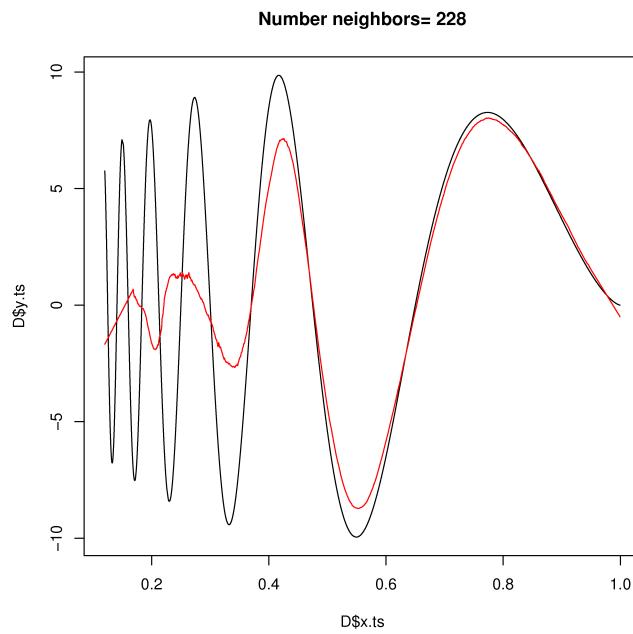


Figure 8.28: Locally linear fitting with a rectangular kernel and a bandwidth made of 228 neighbours.

computed by averaging the value of y for the k closest neighbours $x_{[i]}$, $i = 1, \dots, k$ of q .

The bias/variance decomposition takes the form discussed in Equation (3.5.18) that is

$$\text{MSE}(q) = \sigma_w^2 + \left(\frac{1}{k} \sum_{i=1}^k f(x_{[i]}) - f(q) \right)^2 + \sigma_w^2/k \quad (8.1.54)$$

where σ_w^2 is the variance of the noise and σ_w^2/k is the variance of a sample average estimator based on k points (Equation (3.5.11)). Note the behaviour of the MSE term as a function of k . By increasing k (i.e. larger neighbourhood) the first term is invariant, the bias is likely to increase (since farther points are potentially uncorrelated with q) and the variance decreases.

8.2 Nonlinear classification

In Section 7.2.1, we have shown that optimal classification is possible only if the quantities $\text{Prob}\{\mathbf{y} = c_k | x\}$, $k = 1, \dots, K$ are known. What happens if this is not the case? Three strategies are generally used.

8.2.1 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 a binary classification problem can be put in the form of a regression problem where the output takes value in $\{0, 1\}$. This means that, in principle, all the regression techniques presented so far could be used to solve a classification task. In practice, most of those techniques do not make any assumption about the fact that the outcome in a classification task should satisfy the probabilistic constraints, e.g. $0 \leq \text{Prob}\{\mathbf{y} = 1|x\} \leq 1$. This means that only some regression algorithms (e.g. local constant models) are commonly used for binary classification as well.

8.2.1.1 The nearest-neighbour classifier

The nearest-neighbour algorithm is an example of local modelling (Section 8.1.11) algorithm for classification.

Let us consider a binary $\{0, 1\}$ classification task where a training set is available and the classification is required for an input vector q (*query point*). The classification procedure of a k-NN classifier can be summarised in the following steps:

1. Compute the distance between the query q and the training examples according to a predefined metric.
2. Rank the observed inputs on the basis of their distance to the query.
3. Select a subset $\{x_{[1]}, \dots, x_{[k]}\}$ of the $k \geq 1$ nearest neighbours. Each of these neighbours $x_{[k]}$ has an associated class $y_{[k]}$.
4. Compute the estimation of the conditional probability of the class 1 by constant fitting

$$\hat{p}_1(q) = \frac{\sum_{i=1}^k y_{[i]}}{k} \quad (8.2.55)$$

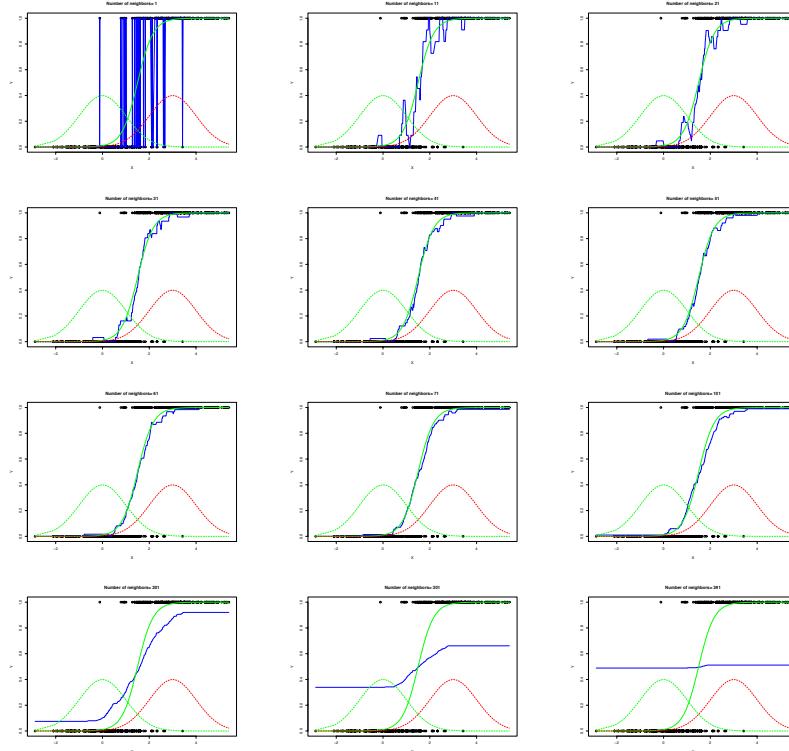


Figure 8.29: kNN prediction (blue line) of the conditional probability (green line) for different values of k . Dotted-lines represent the class-conditional densities.

or linear fitting

$$\hat{p}_1(q) = \hat{a}q + \hat{b}$$

where the parameters \hat{a} and \hat{b} are locally fitted by least-squares regression.

5. Return the prediction, either by a majority vote or according to the conditional probability.

It is evident that the hyperparameter k plays a key role in the trade-off between bias and variance. Figure 8.29 illustrates the trade-off in an $n = 1$ and $N = 400$ binary classification task where the two class-conditional distributions are Normal with means in -1 and 3 , respectively. Note that by increasing k the prediction profile becomes smoother and smoother.

Figure 8.30 shows the trade-off in an $n = 2$ classification task. Note that, though the separating region becomes closer to the optimal for large k , an extrapolation bias occurs in regions far from the observed examples.

It is interesting to see that the kNN classifier can be justified in terms of the Bayes theorem. Suppose that the dataset has the form $D_N = \{(x_1, y_1), \dots, (x_N, y_N)\}$ where $y \in \{c_1, \dots, c_K\}$ and $q \in \mathbb{R}^n$ is the query point where we want to compute the a posteriori probability. Suppose that the dataset contains N_j points labelled with the class c_j , i.e.

$$\sum_{j=1}^K N_j = N$$

Let us consider a region R around the input x having volume V . If the volume

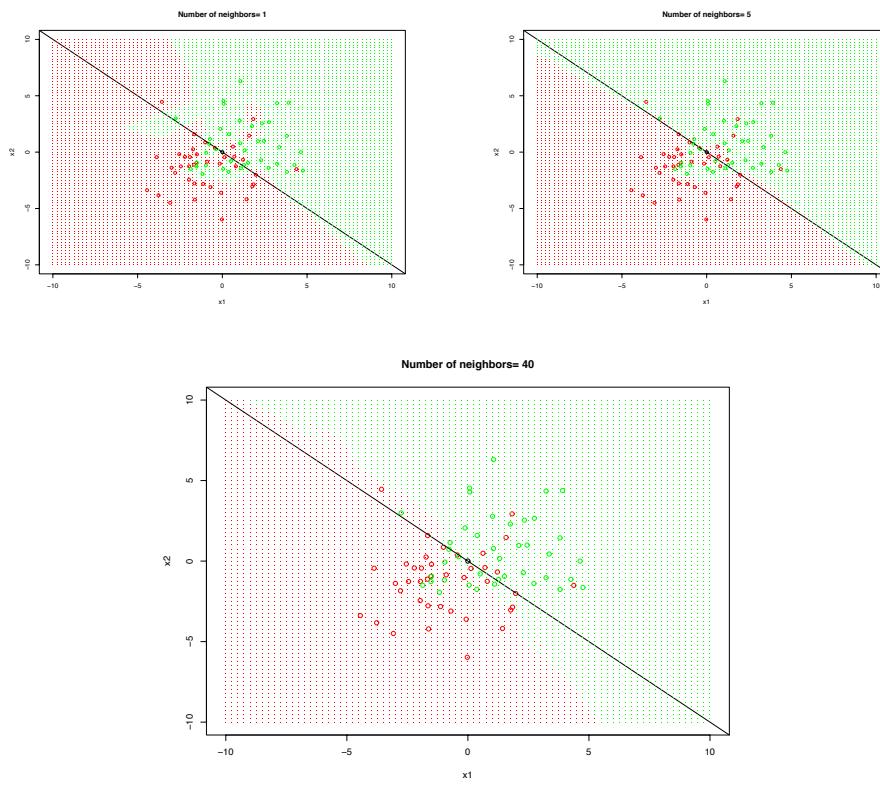


Figure 8.30: kNN class predictions in the $n = 2$ input space for different values of k . Dots represent the training points. Continuous black line is the optimal separating hyperplane.

is small enough, we may consider the density constant over the entire region⁵. It follows that the probability of observing a point within this volume is

$$P = \int_R p(z) dz \approx p(x)V \Rightarrow p(x) \approx \frac{P}{V}$$

Given a training dataset of size N , if we observe N_R example in a region R we can approximate P with $\hat{P} = \frac{N_R}{N}$ and consequently obtain

$$\hat{p}(x) = \frac{N_R}{NV} \quad (8.2.56)$$

Now consider the query point q and a neighbouring volume V containing k points of which $k_j \leq k$ are labelled with the class c_j . From (8.2.56) we obtain the k-NN density estimate (A.1.11) of the class-conditional probability

$$\hat{p}(q|c_j) = \frac{k_j}{N_j V}$$

and of the unconditional density

$$\hat{p}(q) = \frac{k}{NV}$$

Since the class priors can be estimated by $\widehat{\text{Prob}}\{\mathbf{y} = c_j\} = \frac{N_j}{N}$ from (5.3.16) it follows

$$\widehat{\text{Prob}}\{\mathbf{y} = c_j|q\} = \frac{k_j}{k} \quad j = 1, \dots, K$$

This implies that in a binary $\{0, 1\}$ case the computation (8.2.55) estimates the conditional probability of the class 1.

8.2.2 Direct estimation via cross-entropy

The approach consists in modelling the conditional distribution $\text{Prob}\{\mathbf{y} = c_j|x\}, j = 1, \dots, K$ with a set of models $\hat{P}_j(x, \alpha), j = 1, \dots, K$ satisfying the constraints $\hat{P}_j(x, \alpha) > 0$ and

$$\sum_{j=1}^K \hat{P}_j(x, \alpha) = 1.$$

In order to satisfy the normalisation constraint, it is common to have recourse to a *softmax* (or *multinomial logit*) function

$$\left[\frac{\exp^{a_1}}{\exp^{a_1} + \dots + \exp^{a_K}}, \dots, \frac{\exp^{a_K}}{\exp^{a_1} + \dots + \exp^{a_K}} \right] \quad (8.2.57)$$

i.e. a function taking as input a vector $[a_j], j = 1, \dots, K$ of K real numbers and returning the vector $[\hat{P}_j], j = 1, \dots, K$ of probabilities of a categorical distribution.

Parametric estimation boils down to the minimisation of the cross-entropy cost function (6.6.5). Typical approaches are logistic regression and neural networks.

In logistic regression for a two-class task we have

$$\hat{P}_1(x, \alpha) = \frac{\exp^{x^T \alpha}}{1 + \exp^{x^T \alpha}} = \frac{1}{1 + \exp^{-x^T \alpha}}, \quad \hat{P}_2(x, \alpha) = \frac{1}{1 + \exp^{x^T \alpha}} \quad (8.2.58)$$

⁵For a discussion about the validity of this assumption in large dimensional settings, refer to Section 10.1

where x and α are $[p, 1]$ vectors and

$$\left[\frac{\exp^{x^T \alpha}}{1 + \exp^{x^T \alpha}}, \frac{1}{1 + \exp^{x^T \alpha}} \right]$$

is the binary version of the softmax function (8.2.57). This implies

$$\log \frac{\hat{P}_1(x, \alpha)}{\hat{P}_2(x, \alpha)} = x^T \alpha$$

where the transformation $\log \frac{p}{1-p}$ is called the *logit* transformation and the function (8.1.2) is the logistic function. Note that in the case of Normal class-conditional distributions (Section 7.2.1.1) it is possible to derive the value of α from the a priori probabilities, the means and the covariances of the two distributions [162].

In a nonlinear classifier (e.g. neural net)

$$\log \frac{\hat{P}_1(x, \alpha)}{\hat{P}_2(x, \alpha)} = h(x, \alpha)$$

where $h(x, \alpha)$ is the output of the learner.

In a binary case ($c_1 = 1, c_2 = 0$) the cross-entropy function (to minimise) becomes

$$\begin{aligned} J(\alpha) &= -\sum_{i=1}^N \log \hat{P}_{y_i}(x_i, \alpha) = \\ &= -\sum_{i=1}^N \left(y_i \log \hat{P}_1(x_i, \alpha) + (1 - y_i) \log(1 - \hat{P}_1(x_i, \alpha)) \right) = \\ &= \sum_{i=1}^N \left(-y_i h(x_i, \alpha) + \log(1 + \exp^{h(x_i, \alpha)}) \right) \quad (8.2.59) \end{aligned}$$

In the logistic regression case (linear h) the cost function is minimised by iteratively reweighted least squares. For a generic h , a gradient-based iterative approach is required.

Another formulation of the binary case is ($c_1 = 1, c_2 = -1$) with

$$\hat{P}(y|x) = \frac{1}{1 + \exp^{-yh(x, \alpha)}}$$

that satisfies $\hat{P}(y = 1|x) = 1 - \hat{P}(y = -1|x)$. In this case the classification rule is the sign function $\text{sign}[h(x)]$ and the cost function to minimise is

$$J(\alpha) = \sum_{i=1}^N \log(1 + \exp^{-y_i h(x_i, \alpha)}) \quad (8.2.60)$$

also known as the *log-loss* function which is a monotone decreasing function of the terms $y_i h(x_i, \alpha) = y_i h_i$ called the margins (see (7.2.73)). Minimising (8.2.60) is then equivalent to minimise the set of training points for which y_i and the prediction $h(x_i, \alpha)$ have a different sign (or negative margins). The decreasing nature of the function $\exp^{-y h(x, \alpha)}$ guarantees that negative margins are more penalised than positive ones and that the loss of positive margins tends asymptotically to zero (see the log-loss curve in Figure 8.31). Note that this is not guaranteed by a least-squares criterion (used in regression) $(y - h(x, \alpha))^2$ where in some cases a positive margin may have higher cost than a negative one (see the quadratic loss curve in Figure 8.31). This is a reason why the use of regression techniques (Section 8.2.1) is not typically recommended for classification tasks.

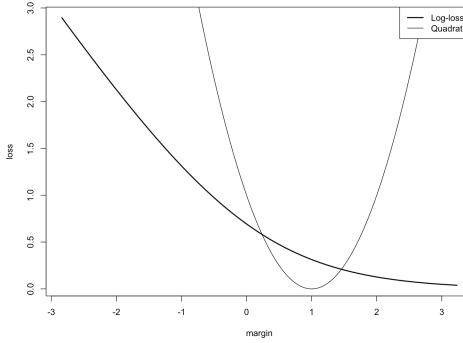


Figure 8.31: Log-loss and quadratic loss as a function of the margin for a binary classification task where $c_1 = 1, c_2 = -1$. Note that the quadratic loss attains its minimum when $y = h = 1$ or $y = h = -1$ (i.e. when $yh = 1$).

8.2.3 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.

8.2.3.1 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 $\{c_1, \dots, c_K\}$. The Bayes optimal classifier should return

$$c^*(x) = \arg \max_{j=1, \dots, K} \text{Prob}\{\mathbf{y} = c_j | x\}$$

We can use the Bayes theorem to rewrite this expression as

$$\begin{aligned} c^*(x) &= \arg \max_{j=1, \dots, K} \frac{\text{Prob}\{x|\mathbf{y} = c_j\} \text{Prob}\{\mathbf{y} = c_j\}}{\text{Prob}\{x\}} \\ &= \arg \max_{j=1, \dots, K} \text{Prob}\{x|\mathbf{y} = c_j\} \text{Prob}\{\mathbf{y} = c_j\} \end{aligned}$$

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 (see Section 2.13.5):

$$\text{Prob}\{x|\mathbf{y} = c_j\} = \text{Prob}\{x_1, \dots, x_n | \mathbf{y} = c_j\} = \prod_{h=1}^n \text{Prob}\{x_h | \mathbf{y} = c_j\}$$

The NB classification is then

$$c_{NB}(x) = \arg \max_{j=1, \dots, K} \text{Prob}\{\mathbf{y} = c_j\} \prod_{h=1}^n \text{Prob}\{x_h | \mathbf{y} = c_j\}$$

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

$$\begin{aligned} \text{Prob}\{\mathbf{y} = P.HIGH\} &= 2/10, \quad \text{Prob}\{\mathbf{y} = P.LOW\} = 5/10 \\ \text{Prob}\{\mathbf{y} = N.HIGH\} &= 1/10, \quad \text{Prob}\{\mathbf{y} = N.LOW\} = 2/10 \\ \text{Prob}\{G1 = N.LOW | \mathbf{y} = P.HIGH\} &= 0/2, \quad \text{Prob}\{G1 = N.LOW | \mathbf{y} = P.LOW\} = 1/5 \\ \text{Prob}\{G1 = N.LOW | \mathbf{y} = N.HIGH\} &= 1/1, \quad \text{Prob}\{G1 = N.LOW | \mathbf{y} = N.LOW\} = 0/2 \\ \text{Prob}\{G2 = N.HIGH | \mathbf{y} = P.HIGH\} &= 0/2, \quad \text{Prob}\{G2 = N.HIGH | \mathbf{y} = P.LOW\} = 1/5 \\ \text{Prob}\{G2 = N.HIGH | \mathbf{y} = N.HIGH\} &= 0/1, \quad \text{Prob}\{G2 = N.HIGH | \mathbf{y} = N.LOW\} = 0/2 \\ \text{Prob}\{G3 = N.LOW | \mathbf{y} = P.HIGH\} &= 0/2, \quad \text{Prob}\{G3 = N.LOW | \mathbf{y} = P.LOW\} = 3/5 \\ \text{Prob}\{G3 = N.LOW | \mathbf{y} = N.HIGH\} &= 0/1, \quad \text{Prob}\{G3 = N.LOW | \mathbf{y} = N.LOW\} = 0/2 \end{aligned}$$

it follows that

$$\begin{aligned} c_{NB}(x) &= \\ \arg \max_{P.H, P.L, N.H, N.L} &\{2/10 * 0 * 0 * 0, 5/10 * 1/5 * 1/5 * 3/5, 1/10 * 1 * 0 * 1, 2/10 * 0 * 0 * 0\} = \\ &= P.LOW \end{aligned}$$

•

The NB classifier relies on the naive (i.e. simplistic) assumption that the inputs are independent given the target class. But why is this assumption made and when may it be considered as realistic? There are essentially two reasons underlying the NB approach, one of statistical nature and the second of causal nature. From a statistical perspective, the conditional independence assumption largely reduces the capacity of the classifier by reducing the number of parameters. This is a variance reduction argument which makes the algorithm effective in large dimensional classification tasks. However, there are classification tasks which by their own nature

are more compliant with the NB assumptions than others. Those are tasks where the features used to predict the class are descriptors of the phenomenon represented by the class. Think for instance to the classification task where a doctor predicts if a patient got flu by means of symptomatic information (does she cough? has he fever?). All those measures are correlated but they become independent once we know the latent state.

8.2.3.2 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 space.

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 $z_j = z_j(x)$, $j = 1, \dots, m$, where $z_j(\cdot)$ is a predefined 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 (Chapter 10).

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 but is particularly efficient in the SVM framework. Before discussing it, we introduce the notion of the dot-product kernel.

Definition 2.1 (Dot-product kernel). A dot-product kernel is a symmetric continuous function $K(\cdot, \cdot)$ such that for all $x_i, x_j \in \mathcal{X}$

$$K(x_i, x_j) = \langle z(x_i), z(x_j) \rangle \quad (8.2.61)$$

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 an Hilbert⁶ feature space \mathcal{Z} .

It can be shown that any symmetric continuous function satisfying the Mercer's condition (Appendix F) is a dot-product kernel.

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 (Section 7.2.3). From (7.2.81) we derive that the parametric identification step requires the solution of the dual quadratic programming problem

$$\begin{aligned} \max_{\alpha} & \left\{ \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k z_i^T z_k \right\} = \\ & = \max_{\alpha} \left\{ \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k \langle z_i, z_k \rangle \right\} = \\ & = \left\{ \max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k K(x_i, x_k) \right\} \end{aligned} \quad (8.2.62)$$

subject to $0 = \sum_{i=1}^N \alpha_i y_i$ and $\alpha_i \geq 0$, $i = 1, \dots, N$. If the kernel matrix K whose ik term is $K(x_i, x_k)$ is positive definite the solution is global and unique.

⁶Note that an Hilbert space is a vector space equipped with an inner product operation which generalises the notion of Euclidean space to spaces that may be infinite dimensional.

What is interesting is that the resolution of this problem differs from the linear one (Equation (7.2.74)) by the replacement of the quantities $\langle x_i, x_k \rangle$ with $\langle z_i, z_k \rangle = K(x_i, x_k)$. The resulting nonlinear SVM classifier takes then the form

$$h(x, \beta, \beta_0) = \text{sign} \left[\sum_{\text{support vectors}} y_i \alpha_i K(x_i, x) + \beta_0 \right] \quad (8.2.63)$$

where the kernel function takes the place of the dot products in (7.2.83).

Whatever the feature transformation $z(x)$ and the dimensionality m of \mathcal{Z} , the SVM computation requires only the availability of the symmetric positive definite Gram matrix, also referred to as the *kernel matrix* K . What is interesting is that once we know how to derive the kernel matrix we are not required to make explicit the underlying transformation function $z(x)$.

The use of a kernel function is an attractive computational short cut (also known as the *kernel trick*) which enables the resolution in a high dimensional space without doing explicit computations in that space.

8.3 Is there a best learner?

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 tons of experimental simulations have been submitted to the scientific community. Every machine learning researcher dreams of inventing the most accurate algorithm, without realising that the attainment of such an objective would necessarily mean the end of machine learning... But is there an algorithm to be universally preferred over others in terms of prediction accuracy?

If there was a universally best learning machine, research on machine learning would be unnecessary: we would use it all the time. (Un)fortunately, the theoretical results on this subject are not encouraging [59]. For any number N of observations, there exist an input/output distribution for which the estimate of generalisation error is arbitrarily poor. At the same time, for any learning machine \mathcal{L}_1 it exists a data distribution and another learning machine \mathcal{L}_2 such that for all N , \mathcal{L}_2 is better than \mathcal{L}_1 .

It can be shown that there is no learning algorithm which is inherently superior to any other, or even to random guessing. The accuracy depends on the match between the (unknown) target distribution and the (implicit or explicit) inductive bias of the learner.

This (surprising ?) result has been formalised by the No Free Lunch (NFL) theorems by D. Wolpert [179]. In his seminal work, Wolpert characterises in probabilistic terms the relation between target function, dataset and hypothesis. The main difference with respect to other research on generalisation is that he does not consider the generating process as constant (e.g. f fixed as in the bias/variance decomposition 5.6.31), but he supposes the existence of a probability distribution $p(f)$ over the target functions \mathbf{f} and that a learning algorithm implements a probability distribution $p(h)$ over the hypothesis \mathbf{h} . For instance, $p(\mathbf{h} = h | D_N)$ denotes the probability that a learning algorithm will return the hypothesis h given the training set D_N ⁷. Based on this formalism, he encodes the following assumptions in a probabilistic language:

- the target distribution is completely outside the researcher's control [179],
- the learning algorithm designer has no knowledge about f when guessing a hypothesis function.

⁷Note that throughout this book we have only considered deterministic learning algorithms, for which $p(h|D_N)$ is a Dirac function

This means that, over the input space region where we observed no training examples (off-training region), the hypothesis \mathbf{h} is conditionally independent (Section 2.13.5) of \mathbf{f} given the training set:

$$p(h|f, D_N) = p(h|D_N)$$

which in turn is equivalent to

$$p(f|h, D_N) = p(f|D_N)$$

In other terms the only information about the target process that a hypothesis may take advantage of is the one contained in the training set.

He then derives the generalisation error of a learning algorithm \mathcal{L} conditioned on a training set D_N and computed on input values which do not belong to the training set (i.e. off-training set region) as

$$\begin{aligned} \sum_{x \notin D_N} E_{\mathbf{f}, \mathbf{h}}[L(\mathbf{f}, \mathbf{h})|D_N](x) &= \\ \sum_{x \notin D_N} \int_{f,h} L(h(x), f(x))p(f, h|D_N)dfdh &= \sum_{x \notin D_N} \int_{f,h} L(h(x), f(x))p(f|D_N)p(h|D_N)dfdh \end{aligned} \quad (8.3.64)$$

It follows that the generalisation error⁸ depends on the alignment (or match) between the hypothesis h returned by \mathcal{L} and the target f which is represented by the inner product $p(f|D_N)p(h|D_N)$. Since the target is unknown, this match may only be assessed a posteriori: a priori there is no reason to consider a learning algorithm better than another. For any learning algorithm which is well aligned with the distribution $p(f|D_N)$ in the off-training set, it is possible to find another distribution for which the match is much worse. Equation (8.3.64) is one of the several NFL results stating that there is no problem-independent reason to favour one learning algorithm \mathcal{L} over another (not even random guessing) if

1. we are interested only in generalisation accuracy,
2. we make no a priori assumption on the target distribution,
3. we restrain to the accuracy over the off-training set region.

The presumed overall superiority of a learning algorithm (no matter the number of publications or the H-factor of the author) is apparent and depends on the specific task and underlying data generating process. The NFL theorems are then a modern probabilistic version of the Hume sceptical argument : there is no logical evidence that the future will behave like the past. Any prediction or modelling effort demands (explicitly or implicitly) an assumption about the data generating process and the resulting accuracy is strictly related to the validity of such an assumption. Note that such assumptions underlie also learning procedures that seem to be general-purpose and data-driven like holdout or cross-validation: for instance, a holdout strategy makes the assumption that the relation between the training portion of the dataset and the validation one is informative about the relation between the observed dataset and future query points (in off-training regions).

⁸Note the differences between the definitions (5.6.31) and (8.3.64) of generalisation error. In (5.6.31) f is fixed and D_N is random: in (8.3.64) f is random and D_N is fixed

x_1	x_2	x_3	y	\hat{y}_1	\hat{y}_2	\hat{y}_3	\hat{y}_4
0	0	0	1	1	1	1	1
0	0	1	0	0	0	0	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	?	0	0	1	1
1	1	1	?	0	1	0	1

Table 8.1: Off-training set prediction binary example

x_1	x_2	x_3	y	\hat{y}_1	\hat{y}_2	\hat{y}_3	\hat{y}_4
0	0	0	1	1	1	1	1
0	0	1	0	0	0	0	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	?	0	0	1	1
1	1	1	?	0	1	0	1

Table 8.2: Off-training set prediction binary example: nearest-neighbour interpretation of the four classifiers. The colours show which training points have been used according to a nearest-neighbour strategy to return the off-training set predictions.

A NFL example

Let us consider a classification example from [62] where we have three binary inputs and one binary target which is a deterministic function of the inputs. Let us suppose that the value of the target is known for 6 input configurations and that we want to predict the value for the 2 remaining ones (Table 8.1). Let us consider 4 classifiers which have identical behaviour for the training set yet they differ in terms of their predictions for the off-training region. Which one is the best one in the off-training set region? May we discriminate between them on the basis of the training behaviour, if we make no additional assumption about the input/output relationship in the off-training set region? The No Free Lunch answer is no. If we assume no a priori information about the conditional distribution, we have 4 equiprobable off-training behaviours. On average the four predictors have the same accuracy. Note that each predictor could have a justification in terms of the nearest neighbour (Table 8.2). For instance, the first classifier (in black) relies on the inductive bias that the first and the third features are the most informative about the target, i.e. if we consider in the dataset the nearest neighbours of the off-training inputs we obtain two zeros as predictions. This is not the case of the fourth (red) which makes implicitly a different hypothesis (i.e. the target depends on x_2 only) and returns two ones accordingly.

8.4 Conclusions

A large part of machine learning research in the last forty years has been devoted to the quest for the Holy Grail of generalisation. This chapter presented a number of learning algorithms and their rationale. Most of those algorithms made the history of the machine learning and were undeniably responsible for the success of the discipline. When they were introduced, and every time they were used afterwards, they were shown to be competitive and often outperform other algorithms. So, how may all this be compatible with the No Free Lunch result? First of all the NFL does not deny that some algorithms may generalise well under some circumstances. It simply states that there is not a single algorithm outperforming consistently all the others. Also, NFL results assume that the off-training set is the most pertinent measure for assessing algorithms. Last but not least, the success of statistics (and ML) is probably indicative that the prediction tasks we are commonly confronted with belong to not such a wide and uniform distribution but that some of them are more probable than others.

Nevertheless, the NFL results may appear as frustrating to a young researcher aiming to pursue a career in machine learning (and incidentally finding the Holy Grail): this is not necessarily so if we define in a less utopian, yet more scientific way, the mission of a data scientist. The mission of a data scientist should not be the promotion of a specific algorithm (or family of algorithms) but acting as a scientist through the analysis of data. This means (s)he should use his know-how NOT to return information about the merits of his preferred algorithm BUT about the nature of the data generating distribution (s)he is dealing with. The outcome of a ML research activity (e.g. publication) should be additional insight about the observed reality (or Nature) and not a contingent statement about the temporary superiority of an algorithm. Newtons aim was to use differential calculus to model and explain dynamics in Nature and not to promote a fancy differential equation tool⁹. Consider also that every ML algorithm (also the least fashionable and the least performing) might return some information (e.g. about the degree of noise, nonlinearity) about the phenomenon we are observing. For instance, a low accurate linear model tells us a lot about the lack of validity of the (embedded) linear assumption in the observed phenomenon. In that sense wrong models might play a relevant role as well since they might return important information about the phenomenon under observation, notably (non)linearity, (non)stationarity, degree of stochasticity, relevance of features and nature of noise.

8.5 Exercises

1. Suppose you want to learn a classifier for detecting spam in emails. Let the binary variables \mathbf{x}_1 , \mathbf{x}_2 and \mathbf{x}_3 represent the occurrence of the words "Viagra", "Lottery" and "Won", respectively, in a email.

Let the dataset of 20 emails being summarized as follows

⁹...or patent it !

Document	x_1 (Viagra)	x_2 (Lottery)	x_3 (Won)	y (Class)
E1	0	0	0	NOSPAM
E2	0	1	1	SPAM
E3	0	0	1	NOSPAM
E4	0	1	1	SPAM
E5	1	0	0	SPAM
E6	1	1	1	SPAM
E7	0	0	1	NOSPAM
E8	0	1	1	SPAM
E9	0	0	0	NOSPAM
E10	0	1	1	SPAM
E11	1	0	0	NOSPAM
E12	0	1	1	SPAM
E13	0	0	0	NOSPAM
E14	0	1	1	SPAM
E15	0	0	1	NOSPAM
E16	0	1	1	SPAM
E17	1	0	0	SPAM
E18	1	1	1	SPAM
E19	0	0	1	NOSPAM
E20	0	1	1	SPAM

where

- 0 stands for the case-insensitive absence of the word in the email.
- 1 stands for the case-insensitive presence of the word in the email.

Let $y = 1$ denote a spam email and $y = 0$ a no-spam email.

The student should

2. Estimate on the basis of the data of exercise 1:

- $\text{Prob}\{\mathbf{x}_1 = 1, \mathbf{x}_2 = 1\}$
- $\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_2 = 1, \mathbf{x}_3 = 1\}$
- $\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{x}_2 = 1\}$
- $\text{Prob}\{\mathbf{x}_3 = 1 | \mathbf{y} = 0, \mathbf{x}_2 = 0\}$
- $\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\}$
- $\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{y} = 0\}$
- $\text{Prob}\{\mathbf{y} = 0\}$

Solution:

- $\text{Prob}\{\mathbf{x}_1 = 1, \mathbf{x}_2 = 1\} = 0.1$
- $\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_2 = 1, \mathbf{x}_3 = 1\} = 0$
- $\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{x}_2 = 1\} = 0.8$
- $\text{Prob}\{\mathbf{x}_3 = 1 | \mathbf{y} = 0, \mathbf{x}_2 = 0\} = 0.5$
- $\text{Prob}\{\mathbf{y} = 0 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\} = 1$
- $\text{Prob}\{\mathbf{x}_1 = 0 | \mathbf{y} = 0\} = 0.875$
- $\text{Prob}\{\mathbf{y} = 0\} = 0.4$

3. Answer to the following questions (Yes or No) on the basis of the data of exercise 1:

- Are \mathbf{x}_1 and \mathbf{x}_2 independent?
- Are \mathbf{x}_1 and \mathbf{y} independent?
- Are the events $\mathbf{x}_1 = 1$ and $\mathbf{x}_2 = 1$ mutually exclusive?

Solution:

- Are x_1 and x_2 independent? NO
- Are x_1 and y independent? NO
- Are the events $x_1 = 1$ and $x_2 = 1$ mutually exclusive? NO

4. Consider the following three emails

- M1: "Lowest Viagra, Cialis, Levitra price".
- M2: "From Google Promo (GOOGLEPROMOASIA) Congratulations! Your mobile won 1 MILLION USD in the GOOGLE PROMO"
- M3: "This is to inform you on the release of the EL-GORDO SWEEPSTAKE LOTTERY PROGRAM. Your name is attached to ticket number 025-11-464-992-750 with serial number 2113-05 drew the lucky numbers 13-15 which consequently won the lottery in the 3rd category."

Use a Naive Bayes Classifier to compute for email M1 on the basis of the data of exercise 1::

- the input x
- $\text{Prob}\{y = \text{SPAM}|x\} \text{Prob}\{x\}$
- $\text{Prob}\{y = \text{NOSPAM}|x\} \text{Prob}\{x\}$
- the email class

Solution:

- the input $x = [1, 0, 0]$
- $\text{Prob}\{y = \text{SPAM}|x\} \text{Prob}\{x\} = 1/180 = 0.0055$
- $\text{Prob}\{y = \text{NOSPAM}|x\} \text{Prob}\{x\} = 1/40 = 0.025$
- the email class: NOSP

5. Use a Naive Bayes Classifier to compute for email M2 on the basis of the data of exercise 1::

- the input x
- $\text{Prob}\{y = \text{SPAM}|x\} \text{Prob}\{x\}$
- $\text{Prob}\{y = \text{NOSPAM}|x\} \text{Prob}\{x\}$
- the email class.

Solution:

- the input $x = [0, 0, 1]$
- $\text{Prob}\{y = \text{SPAM}|x\} \text{Prob}\{x\} = 1/18 = 0.055$
- $\text{Prob}\{y = \text{NOSPAM}|x\} \text{Prob}\{x\} = 7/40 = 0.175$
- the email class is NOSPAM.

6. Use a Naive Bayes Classifier to compute for email M3 on the basis of the data of exercise 1:

- the input x
- $\text{Prob}\{y = \text{SPAM}|x\} \text{Prob}\{x\}$
- $\text{Prob}\{y = \text{NOSPAM}|x\} \text{Prob}\{x\}$
- the email class

Solution:

- the input $x = [0, 1, 1]$

- $\text{Prob}\{\mathbf{y} = \text{SPAM}|x\} \text{Prob}\{x\} = 5/18 = 0.27$
- $\text{Prob}\{\mathbf{y} = \text{NOSPAM}|x\} \text{Prob}\{x\} = 0$
- the email class is SPAM

7. Consider a classification task with two binary inputs and one binary target $\mathbf{y} \in \{-1, +1\}$ where the conditional distribution is

\mathbf{x}_1	\mathbf{x}_2	$P(\mathbf{y} = 1 x_1, x_2)$
0	0	0.8
0	1	0.1
1	0	0.5
1	1	1

Suppose that all the input configurations have the same probability.

Let the classifier be the rule:

$$\text{IF } x_2 = 0 \text{ THEN } \hat{y} = -1 \text{ ELSE } \hat{y} = 1.$$

Consider a test set of size $N = 10000$.

For this classifier compute:

- the confusion matrix,
- the precision,
- the specificity (true negative rate)
- the sensitivity (true positive rate)

Solution:

- the confusion matrix,

	$\hat{y} = -1$	$\hat{y} = 1$
$y = -1$	TN=1750	FP=2250
$y = 1$	FN=3250	TP=2750

- the precision $TP/(TP+FP)=2750/5000=0.55$
- the specificity (true negative rate) $TN/(TN+FP)=1750/4000=0.4375$
- the sensitivity (true positive rate) $TP/(TP+FN)=2750/6000=0.458$

8. Consider a regression task with input \mathbf{x} and output \mathbf{y} and the following training set

X	Y
0	0.5
-0.3	1.2
0.2	1
0.4	0.5
0.1	0
-1	1.1

Consider the three following models:

- constant
- 1NN, Nearest Neighbour with K=1
- 3NN, Nearest Neighbour with K=3

Compute for the constant model

- the vector of training errors $e_i = y_i - \hat{y}_i$
- the vector of leave-one-out errors $e_i^{-i} = y_i - \hat{y}_i^{-i}$

- the mean-squared training error ,
- the mean-square leave-one-out error.

Compute for the 1NN model

- the vector of training errors $e_i = y_i - \hat{y}_i$
- the vector of leave-one-out errors $e^{-i} = y_i - \hat{y}_i^{-i}$
- the mean-squared training error ,
- the mean-squared leave-one-out error.

Compute for the 3NN model

- the vector of training errors $e_i = y_i - \hat{y}_i$
- the vector of leave-one-out errors $e^{-i} = y_i - \hat{y}_i^{-i}$
- the mean-squared training error ,
- the mean-squared leave-one-out error.

Solution: Constant model

- the vector of training errors $e_i = y_i - \hat{y}_i = [-0.2167, 0.4833, 0.2833, -0.2167, -0.7167, 0.3833]$
- the vector of leave-one-out errors $e_i^{-i} = y_i - \hat{y}_i^{-i} = [-0.26, 0.58, 0.34, -0.26, -0.86, 0.46]$
- the mean-squared training error = 0.178
- the mean-square leave-one-out error = 0.2564

1NN model:

- the vector of training errors $e_i = y_i - \hat{y}_i = [000000]$
- the vector of leave-one-out errors $e^{-i} = y_i - \hat{y}_i^{-i} = [0.5, 0.7, 1, -0.5, -0.5, -0.1] [0.5, 0.7, 1, -0.5, -1, -0.1]$
- the mean-squared training error = 0
- the mean-squared leave-one-out error = 0.375 or 0.5

3NN model

- the vector of training errors $e_i = y_i - \hat{y}_i = [0, 0.6333, 0.5, 0, -0.5, 0.1667]$
- the vector of leave-one-out errors $e^{-i} = y_i - \hat{y}_i^{-i} = [-0.2333, 0.7, 0.6667, 0, -0.6667, 0.5333]$
- the mean-squared training error = 0.1548
- the mean-squared leave-one-out error=0.2862

9. Consider a classification task with three binary inputs and one binary target where the conditional distribution is

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	$P(\mathbf{y} = 1 x_1, x_2, x_3)$
0	0	0	0.8
0	0	1	0.9
0	1	0	0.5
0	1	1	1
1	0	0	0.8
1	0	1	0.1
1	1	0	0.1
1	1	1	0

Suppose that all the input configurations have the same probability.

Let the classifier be the rule:

IF $x_1 = 0$ OR $x_2 = 0$ THEN $\hat{y} = 1$ ELSE $\hat{y} = 0$.

Suppose we have a test set of size $N = 10000$.

Considering the class 1 as the positive class, for this classifier compute:

- the confusion matrix,
- the precision,
- the specificity (true negative rate) and
- the sensitivity (true positive rate)

Solution:

- the confusion matrix,

x_1	x_2	x_3	$y = 1$	$\hat{y} = 1$	TP	FP	TN	FN
0	0	0	1000	1250	1000	250	0	0
0	0	1	1125	1250	1125	125	0	0
0	1	0	625	1250	625	625	0	0
0	1	1	1250	1250	1250	0	0	0
1	0	0	1000	1250	1000	250	0	0
1	0	1	125	1250	125	1125	0	0
1	1	0	125	0	0	0	1125	125
1	1	1	0	0	0	0	1250	0

	$\hat{y} = 1$	$\hat{y} = 0$
$y = 1$	TP=5125	FN=125
$y = 0$	FP=2375	TN=2375

- the precision = $5125/(5125+2375)=0.68$
- the specificity (true negative rate) = $2375/(2375+2375)=0.5$
- the sensitivity (true positive rate) = $5125/(5125+125)=0.976$

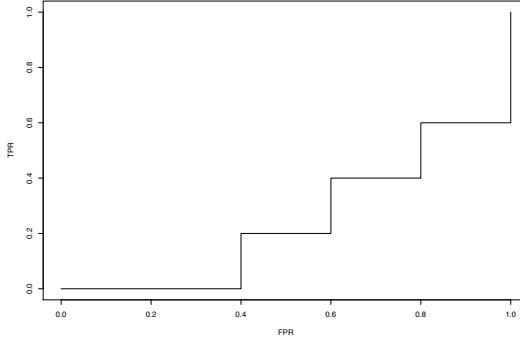
10. Let us consider the following classification dataset where y is the binary target.

x_1	x_2	y
-4	7.0	1
-3	-2.0	1
-2	5.0	0
-1	2.5	1
1	1.0	0
2	4.0	1
3	6.0	0
4	3.0	1
5	-1.0	0
6	8.0	0

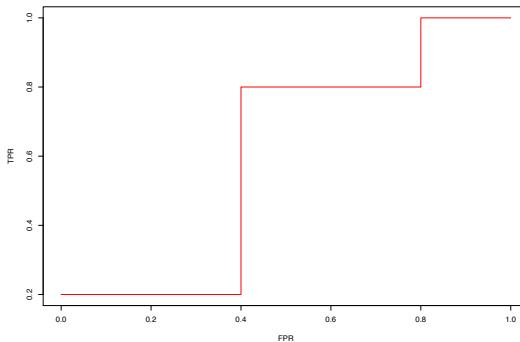
- Consider the 1st classifier: IF $x_1 > h$ THEN $\hat{y} = 1$ ELSE $\hat{y} = 0$
Trace its ROC curve (considering 1 as the positive class)
- Consider the 2nd classifier: IF $x_2 > k$ THEN $\hat{y} = 0$ ELSE $\hat{y} = 1$
Trace its ROC curve (considering 1 as the positive class)
- Which classifier is the best one (1st/2nd)?

Solution:

- 1st classifier ROC curve (considering 1 as the positive class):



- 2nd classifier ROC curve (considering 1 as the positive class):



- Which classifier is the best one (1st/2nd)? The 2nd.

11. Consider a binary classification task and the training set

x_1	x_2	y
1	1	-1
2	0.5	-1
1.5	2.5	-1
3	1.5	1
2.5	3	1
4	2.5	1

Consider a linear perceptron initialised with the boundary line $x_2 = 2$ which classifies as positive the points over the line. The student should:

1. Perform one step of gradient descent with stepsize 0.1 and compute the updated coefficients of the perceptron line with equation

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$$

2. Trace the initial boundary line (in black), the updated boundary line (in red) and the training points.

Solution:

In the initial perceptron $\beta_0 = -2$, $\beta_1 = 0$ and $\beta_2 = 1$. The misclassified points are the third and the fourth (opposite label). Since

$$\frac{\partial R}{\partial \beta} = - \sum_{miscl} y_i x_i = \begin{bmatrix} 1.5 \\ 2.5 \end{bmatrix} - \begin{bmatrix} 3 \\ 1.5 \end{bmatrix} = \begin{bmatrix} -1.5 \\ 1 \end{bmatrix}$$

and

$$\frac{\partial R}{\partial \beta_0} = - \sum_{misl} y_i = 0$$

after one iteration β_0 remains the same while

$$\begin{bmatrix} \beta_1^{t+1} \\ \beta_2^{t+1} \end{bmatrix} = \begin{bmatrix} \beta_1^t \\ \beta_2^t \end{bmatrix} - 0.1 * \begin{bmatrix} -1.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.15 \\ -0.1 \end{bmatrix} = \begin{bmatrix} 0.15 \\ 0.9 \end{bmatrix}$$

Updated coefficients of the perceptron line are then

- $\beta_0 = -2$
- $\beta_1 = 0.15$
- $\beta_2 = 0.9$

12. Consider the data set in exercise 8 and fit to it a Radial Basis Function with 2 basis functions having as parameters $\mu^{(1)} = -0.5$ and $\mu^{(2)} = 0.5$. The equation of the basis function is

$$\rho(x, \mu) = \exp^{-(x-\mu)^2}$$

The student should

1. write in matrix notation the linear system to be solved for obtaining the weights of the radial basis function
2. compute the weights of the radial basis function

Hint:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix} \Rightarrow A^{-1} = \frac{1}{a_{11}a_{22} - a_{12}^2} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{12} & a_{11} \end{bmatrix}$$

Solution:

1. matrix notation $w = (X'X)^{-1}X'Y$ where

$$X = \begin{bmatrix} 0.779 & 0.105 \\ 0.961 & 0.527 \\ 0.779 & 0.779 \\ 0.698 & 0.852 \\ 0.613 & 0.914 \\ 0.445 & 0.990 \end{bmatrix}$$

2. weights of the radial basis function : $w = [1.25, -0.27]$

13. Let us consider a classification task with 3 binary inputs and one binary output. Suppose we collected the following training set

x₁	x₂	x₃	y
0	1	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
0	1	1	0
1	0	1	0
1	0	0	0
1	1	0	0
0	1	1	0

1. Estimate the following quantities by using the frequency as estimator of probability

- $\text{Prob}\{\mathbf{y} = 1\}$
 - $\text{Prob}\{\mathbf{y} = 1|\mathbf{x}_1 = 0\}$
 - $\text{Prob}\{\mathbf{y} = 1|\mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\}$
2. Compute the classification returned by using the Naive Bayes Classifier for the input $\mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0$.
 3. Suppose we test a classifier for this task and that we obtain a misclassification error equal to 20%. Is it more accurate than a zero classifier, i.e. a classifier returning always zero?

Solution: Let us note that $N = 12$

1.
 - $\widehat{\text{Prob}}\{\mathbf{y} = 1\} = 2/12 = 1/6$
 - $\widehat{\text{Prob}}\{\mathbf{y} = 1|\mathbf{x}_1 = 0\} = \frac{1}{6}$
 - $\widehat{\text{Prob}}\{\mathbf{y} = 1|\mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\}$ cannot be estimated using the frequency since there is no observation where $\mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0$
2. Since

$$\begin{aligned} \widehat{\text{Prob}}\{\mathbf{y} = 1|\mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\} &\propto \\ \widehat{\text{Prob}}\{\mathbf{x}_1 = 0|\mathbf{y} = 1\} \widehat{\text{Prob}}\{\mathbf{x}_2 = 0|\mathbf{y} = 1\} \widehat{\text{Prob}}\{\mathbf{x}_3 = 0|\mathbf{y} = 1\} \widehat{\text{Prob}}\{\mathbf{y} = 1\} &= \\ &(0.5 * 0.5 * 0.5 * 1/6) = 0.02 \end{aligned}$$

and

$$\begin{aligned} \widehat{\text{Prob}}\{\mathbf{y} = 0|\mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\} &\propto \\ \widehat{\text{Prob}}\{\mathbf{x}_1 = 0|\mathbf{y} = 0\} \widehat{\text{Prob}}\{\mathbf{x}_2 = 0|\mathbf{y} = 0\} \widehat{\text{Prob}}\{\mathbf{x}_3 = 0|\mathbf{y} = 0\} \widehat{\text{Prob}}\{\mathbf{y} = 0\} &= \\ &(5/10 * 4/10 * 5/10 * 5/6) = 0.08 \end{aligned}$$

the NB classification is 0

3. A zero classifier would return always the class with the highest a priori probability, that is the class 0. Its misclassification error would be then 1/6. Since $1/5 > 1/6$ the classifier is less accurate than the zero classifier.
14. Let us consider a classification task with 3 binary inputs and one binary output. Suppose we collected the following training set

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	\mathbf{y}
0	1	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
0	1	1	0
1	0	1	0
1	0	0	0
1	1	0	0
0	1	1	0

1. Estimate the following quantities by using the frequency as estimator of probability
 - $\text{Prob}\{\mathbf{y} = 1\}$
 - $\text{Prob}\{\mathbf{y} = 1|\mathbf{x}_1 = 0\}$
 - $\text{Prob}\{\mathbf{y} = 1|\mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\}$

2. Compute the classification returned by using the Naive Bayes Classifier for the input $\mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0$.
3. Suppose we test a classifier for this task and that we obtain a misclassification error equal to 20%. Is it working better than a zero classifier, i.e. a classifier ignoring the value of the inputs?

Solution: Let us note that $N = 12$

1.
 - $\widehat{\text{Prob}}\{\mathbf{y} = 1\} = 2/12 = 1/6$
 - $\widehat{\text{Prob}}\{\mathbf{y} = 1 | \mathbf{x}_1 = 0\} = \frac{1}{6}$
 - $\widehat{\text{Prob}}\{\mathbf{y} = 1 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\}$ cannot be estimated using the frequency since there is no observation where $\mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0$
2. Since

$$\begin{aligned}\widehat{\text{Prob}}\{\mathbf{y} = 1 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\} &\propto \\ \widehat{\text{Prob}}\{\mathbf{x}_1 = 0 | \mathbf{y} = 1\} \widehat{\text{Prob}}\{\mathbf{x}_2 = 0 | \mathbf{y} = 1\} \widehat{\text{Prob}}\{\mathbf{x}_3 = 0 | \mathbf{y} = 1\} \widehat{\text{Prob}}\{\mathbf{y} = 1\} &= \\ (0.5 * 0.5 * 0.5 * 1/6) &= 0.02\end{aligned}$$

and

$$\begin{aligned}\widehat{\text{Prob}}\{\mathbf{y} = 0 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\} &\propto \\ \widehat{\text{Prob}}\{\mathbf{x}_1 = 0 | \mathbf{y} = 0\} \widehat{\text{Prob}}\{\mathbf{x}_2 = 0 | \mathbf{y} = 0\} \widehat{\text{Prob}}\{\mathbf{x}_3 = 0 | \mathbf{y} = 0\} \widehat{\text{Prob}}\{\mathbf{y} = 0\} &= \\ (5/10 * 4/10 * 5/10 * 5/6) &= 0.08\end{aligned}$$

the NB classification is 0

3. A zero classifier would return always the class with the highest a priori probability, that is the class 0. Its misclassification error would be then $1/6$. Since $1/5 > 1/6$ the classifier is less accurate than the zero classifier.
15. Consider a regression task with input \mathbf{x} and output \mathbf{y} . Suppose we observe the following training set

X	Y
0.1	1
0	0.5
-0.3	1.2
0.2	1
0.4	0.5
0.1	0
-1	1.1

and that the prediction model is constant. Compute an estimation of its mean integrated squared error by leave-one-out.

Solution: Since the leave-one-out error is

$$e_i^{-i} = y_i - \frac{\sum_{j=1, j \neq i}^N y_j}{N-1}$$

we can compute the vector of errors in leave-one-out

e_1^{-1}	$1 - 0.716 = 0.283$
e_2^{-2}	$0.5 - 0.8 = -0.3$
e_3^{-3}	$1.2 - 0.683 = 0.516$
e_4^{-4}	$1 - 0.716 = 0.283$
e_5^{-5}	$0.5 - 0.8 = -0.3$
e_6^{-6}	$0 - 0.883 = -0.883$
e_7^{-7}	$1.1 - 0.7 = 0.4$

and then derive the MISE estimation

$$\widehat{\text{MISE}}_{\text{loo}} = \frac{\sum_{i=1}^N (e_i^{-i})^2}{N} = 0.22$$

16. Consider a regression task with input \mathbf{x} and output \mathbf{y} . Suppose we observe the following training set

X	Y
0.1	1
0	0.5
-0.3	1.2
0.3	1
0.4	0.5
0.1	0
-1	1.1

and that the prediction model is a KNN (nearest neighbour) where $K = 1$ and the distance metric is euclidean. Compute an estimation of its mean squared error by leave-one-out.

Solution:

The leave-one-out error is

$$e_i^{-i} = y_i - y_i^*$$

where y_i^* is the value of the target associated to x_i^* and x_i^* is the nearest neighbour of x_i . Once we rank the training set according to the input value

X	Y
-1	1.1
-0.3	1.2
0	0.5
0.1	1
0.1	0
0.3	1
0.4	0.5

we can compute the vector of errors in leave-one-out

e_1^{-1}	$1.1 - 1.2 = -0.1$
e_2^{-2}	$1.2 - 0.5 = 0.7$
e_3^{-3}	$0.5 - 1 = -0.5$
e_4^{-4}	$1 - 0 = 1$
e_5^{-5}	$0 - 1 = -1$
e_6^{-6}	$1 - 0.5 = 0.5$
e_7^{-7}	$0.5 - 1 = -0.5$

and then derive the MISE estimation

$$\widehat{\text{MISE}}_{\text{loo}} = \frac{\sum_{i=1}^N (e_i^{-i})^2}{N} = 0.464$$

17. Consider a regression task with input \mathbf{x} and output \mathbf{y} . Suppose we observe the following training set

X	Y
0.5	1
1	1
-1	1
-0.25	1
0	0.5
0.1	0
0.25	0.5

Trace the estimation of the regression function returned by a KNN (nearest neighbor) where $K = 3$ on the interval $[-2, 1]$.

Solution: The resulting graph is piecewise constant and each piece has an ordinate equal to the mean of three points. Once ordered the points according to the abscissa

	X	Y
x_1	-1	1
x_2	-0.25	1
x_3	0	0.5
x_4	0.1	0
x_5	0.25	0.5
x_6	0.5	1
x_7	1	1

these are the five sets of 3 points

$$x_1, x_2, x_3 \Rightarrow \hat{y} = 2.5/3 \quad (8.5.65)$$

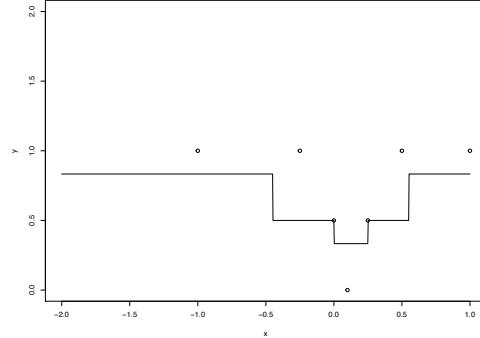
$$x_2, x_3, x_4 \Rightarrow \hat{y} = 0.5 \quad (8.5.66)$$

$$x_3, x_4, x_5 \Rightarrow \hat{y} = 1/3 \quad (8.5.67)$$

$$x_4, x_5, x_6 \Rightarrow \hat{y} = 0.5 \quad (8.5.68)$$

$$x_5, x_6, x_7 \Rightarrow \hat{y} = 2.5/3 \quad (8.5.69)$$

The transitions from x_i, x_{i+1}, x_{i+2} to $x_{i+1}, x_{i+2}, x_{i+3}, i = 1, \dots, 4$ occur at the $x = q$ points where $q - x_i = x_{i+3} - q \Rightarrow q = \frac{x_{i+3} + x_i}{2}$



18. Consider a supervised learning problem, a training set of size $N = 50$ and a neural network predictor with a single hidden layer. Suppose that we are able to compute the generalisation error for different number H of hidden nodes and we discover that the lowest generalisation error occurs for $H = 3$. Suppose now that the size of the training set increases ($N = 500$). For which value of H would you expect the lowest generalisation error? Equal, larger or smaller than 3? Justify your answer by reasoning on the bias/variance trade-off in graphical terms (Figure).

Solution:

According to 5.6.41 the MISE generalisation error may be decomposed as the sum of the squared bias, the model variance and the noise variance.

In Figure 8.32 we depict the first setting in black and the second one (i.e. increased training set size) in red.

The relationship between the squared bias and the capacity of the model (number H) is represented by the dashed line and the relationship between the variance and the capacity is represented by the continuous thin line. The MISE (taking its minimum in $H = 3$) is represented by the black thick line. Note that in the figure we do not consider the noise variance since we are comparing two models for the same regression task and then the noise variance is in this case an irrelevant additive term.

If the training set size increases we can expect a variance reduction. This means that the minimum of the MISE term will move to right. We should then expect that the optimal number of hidden layers is $H > 3$.

Note that additional observations have no impact on the squared bias while they contribute to reduce the variance (red thin line). From the red thick line denoting the MISE of the second setting, it appears that $\arg \min_H MSE(H)$ moved to the right.

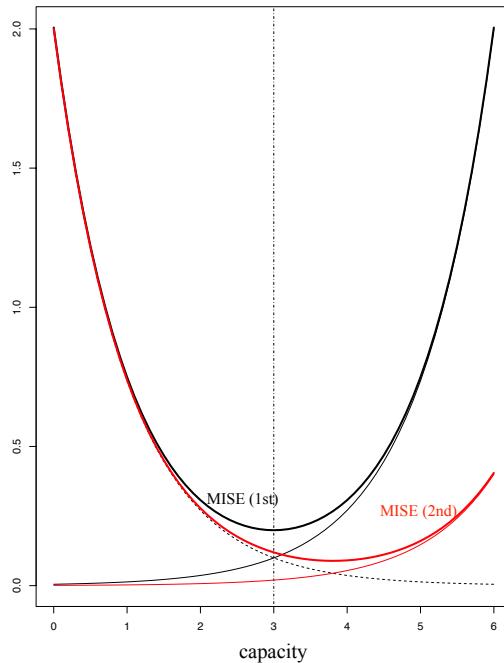
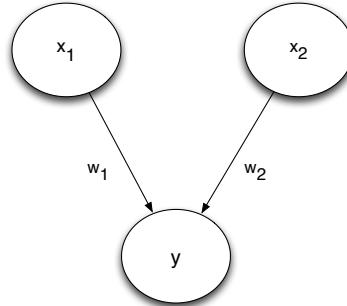


Figure 8.32:

19. Consider a feedforward neural network with two inputs, no hidden layer and a logistic activation function. Suppose we want to use backpropagation to compute the weights w_1 and w_2 and that a training dataset is collected. The student should
 1. Write the equation of the mapping between x_1 , x_2 and y .
 2. Write the two iterative backpropagation equations to compute w_1 and w_2 .



Solution:

1. $\hat{y} = g(z) = g(w_1x_1 + w_2x_2)$ where $g(z) = \frac{1}{1+e^{-z}}$ and $g'(z) = \frac{e^{-z}}{(1+e^{-z})^2}$
2. The training error is

$$E = \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{N}$$

For $j = 1, 2$

$$\frac{\partial E}{\partial w_j} = -\frac{2}{N} \sum_{i=1}^N (y_i - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial w_j}$$

where

$$\frac{\partial \hat{y}_i}{\partial w_j} = g'(z_i)x_{ij}$$

where $z_i = w_1x_{1i} + w_2x_{2i}$

The two backpropagation equations are then

$$w_j(k+1) = w_j(k) + \eta \frac{2}{N} \sum_{i=1}^N (y_i - \hat{y}_i) g'(z_i) x_{ij}, \quad j = 1, 2$$

20. Consider a binary classification problem and the following estimations of the conditional probability $\widehat{\text{Prob}} \{y = 1|x\}$ vs. the real value of the target.

Trace a precision recall and the AUC curve

$\widehat{\text{Prob}} \{y = 1 x\}$	CLASS
0.6	1
0.5	-1
0.99	1
0.49	-1
0.1	-1
0.26	-1
0.33	1
0.15	-1
0.05	-1

Solution: Let us first order the dataset in terms of ascending score

$\widehat{\text{Prob}} \{y = 1 x\}$	CLASS
0.05	-1
0.10	-1
0.15	-1
0.26	-1
0.33	1
0.49	-1
0.50	-1
0.60	1
0.99	1

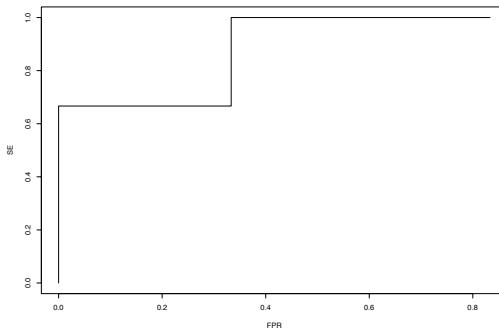
We let the threshold range over all the values of the score. For each value of the threshold we define as positively classified the terms having a score bigger than the threshold and negatively classified the terms having a score lower equal than the threshold.

For instance for $\text{Thr}=0.26$ this is the returned classification

$\widehat{\text{Prob}} \{y = 1 x\}$	\hat{y}	CLASS
0.05	-1	-1
0.10	-1	-1
0.15	-1	-1
0.26	-1	-1
0.33	1	1
0.49	1	-1
0.50	1	-1
0.60	1	1
0.99	1	1

Then we measure the quantity of TP, FP, TN and FN and $FPR = FP/(TN + FP)$, $TPR = TP/(TP + FN)$

Threshold	TP	FP	TN	FN	FPR	TPR
0.05	3	5	1	0	5/6	1
0.10	3	4	2	0	2/3	1
0.15	3	3	3	0	1/2	1
0.26	3	2	4	0	1/3	1
0.33	2	2	4	1	1/3	2/3
0.49	2	1	5	1	1/6	2/3
0.50	2	0	6	1	0	2/3
0.60	1	0	6	2	0	1/3
0.99	0	0	6	3	0	0



21. Let us consider a classification task with 3 binary inputs and one binary output. Suppose we collected the following training set

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	\mathbf{y}
1	1	0	1
0	0	1	0
0	1	0	0
1	1	1	1
0	0	0	0
0	1	0	0
0	1	1	0
0	0	1	0
0	0	0	0
0	1	0	0
1	1	1	1

1. Estimate the following quantities by using the frequency as estimator of probability
 - $\text{Prob}\{\mathbf{y} = 1\}$
 - $\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_1 = 0\}$
 - $\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\}$
2. Consider a Naive Bayes classifier and compute its classifications if the same dataset is used also for testing
3. Trace the ROC curve associated to the Naive Bayes classifier if the same dataset is used also for testing. (Hint: make the assumption that the denominator of the Bayes formula is 1 for all test points)

Solution:

1.
 - $\text{Prob}\{\mathbf{y} = 1\} = 3/11$
 - $\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_1 = 0\} = 0$
 - $\text{Prob}\{\mathbf{y} = 1 | \mathbf{x}_1 = 0, \mathbf{x}_2 = 0, \mathbf{x}_3 = 0\} = 0$
2. Note that the values of \mathbf{x}_1 are identical to the ones of \mathbf{y} . Then $\text{Prob}\{\mathbf{x}_1 = A | \mathbf{y} = \neg A\} = 0$. It follows that if use a Naive Bayes and the test dataset is equal to the training set all the predictions will coincide with the values of \mathbf{x}_1 . The training error is then zero
3. Since all the predictions are correct the ROC curve is equal to 1 for all FPR values
22. Let us consider a binary classification task where the input $\mathbf{x} \in \mathbb{R}^2$ is bivariate and the categorical output variable \mathbf{y} may take two values: 0 (associated to red) and 1 (associated to green). Suppose that the a-priori probability is $p(\mathbf{y} = 1) = 0.2$ and that the inverse (or class-conditional) distributions are the bivariate Gaussian distributions $p(x|\mathbf{y} = 0) = \mathcal{N}(\mu_0, \Sigma_0)$ and $p(x|\mathbf{y} = 1) = \mathcal{N}(\mu_1, \Sigma_1)$ where
 - $\mu_0 = [0, 0]^T$
 - $\mu_1 = [1, 1]^T$

and both Σ_0 and Σ_1 are diagonal identity matrices. The student should

1. by using the R function `rmvnorm`, sample a dataset of $N = 1000$ input/output observations according to the conditional distribution described above,
2. visualise in a 2D graph the dataset by using the appropriate colors,
3. fit a logistic classifier to the dataset (see details below),
4. plot the evolution of the cost function $J(\alpha)$ during the gradient-based minimisation,
5. plot in the 2D graph the decision boundary.

Logistic regression estimates

$$\hat{P}(\mathbf{y} = 1|x) = \frac{\exp^{x^T \alpha_N}}{1 + \exp^{x^T \alpha_N}} = \frac{1}{1 + \exp^{-x^T \alpha_N}}, \quad \hat{P}(\mathbf{y} = 0|x) = \frac{1}{1 + \exp^{x^T \alpha_N}}$$

where

$$\alpha_N = \arg \min_{\alpha} J(\alpha)$$

and

$$J(\alpha) = \sum_{i=1}^N \left(-y_i x_i^T \alpha + \log(1 + \exp^{x_i^T \alpha}) \right)$$

Note that α is the vector $[\alpha_0, \alpha_1, \alpha_2]^T$ and that $x_i = [1, x_{i1}, x_{i2}]^T, i = 1, \dots, N$.

The value of α_N has to be computed by gradient-based minimisation of the cost function $J(\alpha)$ by performing $I = 200$ iterations of the update rule

$$\alpha^{(\tau)} = \alpha^{(\tau-1)} - \eta \frac{dJ(\alpha^{(\tau-1)})}{d\alpha}, \quad \tau = 1, \dots, I$$

where $\alpha^{(0)} = [0, 0, 0]^T$ and $\eta = 0.001$.

Solution:

See the file `Exercise5.pdf` in the directory `gbcodes/exercises` of the companion R package (Appendix G).

23. Consider a binary classification task where the input $\mathbf{x} \in \mathbb{R}^2$ is bivariate and the categorical output variable \mathbf{y} may take two values: 0 (associated to red) and 1 (associated to green). Suppose that the a-priori probability is $p(\mathbf{y} = 1) = 0.4$ and that the inverse (or class-conditional) distributions are

- green/cross class : mixture of three Gaussians

$$p(x|\mathbf{y} = 1) = \sum_{i=1}^3 w_i \mathcal{N}(\mu_{1i}, \Sigma)$$

where in $\mu_{11} = [1, 1]^T, \mu_{12} = [-1, -1]^T, \mu_{13} = [3, -3]^T$, and $w_1 = 0.2, w_2 = 0.3$.

- red/circle class: bivariate Gaussian $p(x|\mathbf{y} = 0) = \mathcal{N}(\mu_0, \Sigma)$ where $\mu_0 = [0, 0]^T$

The matrix Σ is a diagonal identity matrix.

The student should

- by using the R function `rmvnorm`, sample a dataset of $N = 1000$ input/output observations according to the conditional distributions described above,
- visualise in a 2D graph the dataset by using the appropriate colours/marks,
- plot the ROC curves of the following classifiers
 1. linear regression coding the two classes by 0 and 1,
 2. Linear Discriminant Analysis where $\sigma^2 = 1$,
 3. Naive Bayes where the univariate conditional distributions are Gaussian,
 4. k Nearest Neighbour with $k = 3, 5, 10$.

The classifiers should be trained and tested on the same training set.

- Choose the best classifier on the basis of the ROC curves above.

No R package should be used to implement the classifiers.

Solution:

See the file `Exercise6.pdf` in the directory `gbcodes/exercises` of the companion R package (Appendix G).

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 that should work the best. However, recent results in machine learning show that the final accuracy can be improved not by choosing the model structure which is expected to predict the best but by creating a model combining the output of models with different structures. The reason is that 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 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 $h_j(\cdot, \alpha_N)$, $j = 1, \dots, m$ 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 $D_N = \{h_i, y_i\}$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad H = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_N \end{bmatrix} = \begin{bmatrix} h_1(x_1, \alpha_N) & h_2(x_1, \alpha_N) & \dots & h_m(x_1, \alpha_N) \\ h_1(x_2, \alpha_N) & h_2(x_2, \alpha_N) & \dots & h_m(x_2, \alpha_N) \\ \vdots & \vdots & \vdots & \vdots \\ h_1(x_N, \alpha_N) & h_2(x_N, \alpha_N) & \dots & h_m(x_N, \alpha_N) \end{bmatrix}$$

where h_i , $i = 1, \dots, N$ is a vector of m terms.

Once computed the least-squares solution $\hat{\beta}$ the combined estimator is

$$h_{cm}(x) = \sum_{j=1}^m \hat{\beta}_j h_j(x, \alpha_N)$$

Despite 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 [178] presented an interesting idea, called *stacked generalisation* for combining estimators without suffering of the correlation problem. This proposal was translated in statistical language by Breiman who introduced the *stacked regression* principle [38].

The idea consists in estimating the m parameters $\hat{\beta}_j$ by solving the following optimisation 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 (6.8.2.3) 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 $D_N = \{h_i^-, y_i\}$, $i = 1, \dots, N$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad H = \begin{bmatrix} h_1^- \\ h_2^- \\ \vdots \\ h_N^- \end{bmatrix} = \begin{bmatrix} h_1(x_1, \alpha_N^{(-1)}) & h_2(x_1, \alpha_N^{(-1)}) & \dots & h_m(x_1, \alpha_N^{(-1)}) \\ h_1(x_2, \alpha_N^{(-2)}) & h_2(x_2, \alpha_N^{(-2)}) & \dots & h_m(x_2, \alpha_N^{(-2)}) \\ \vdots & \vdots & \vdots & \vdots \\ h_1(x_N, \alpha_N^{(-N)}) & h_2(x_N, \alpha_N^{(-N)}) & \dots & h_m(x_N, \alpha_N^{(-N)}) \end{bmatrix}$$

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 observation (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 minimisation 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 of 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.

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

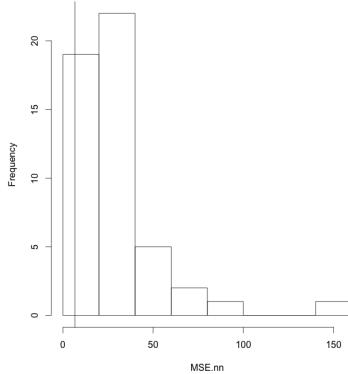


Figure 9.1: Histogram of misclassification rates of resampled trees: the vertical line represents the misclassification rate of the bagging predictor.

is to improve the accuracy by reducing the variance: since the generalisation error of a predictor $h(\cdot, \alpha_N)$ depends on its bias and variance, we obtain an error reduction if we remove the variance term by replacing $h(\cdot, \alpha_N)$ with $E_{D_N}[h(\cdot, \alpha_N)]$. In practice, since the knowledge of the sampling distribution of the predictor is not available, a non-parametric estimation is required.

Consider a dataset D_N and a learning procedure to build a hypothesis α_N from D_N . The idea of *bagging* or *bootstrap aggregating* is to imitate the stochastic process underlying the realisation of D_N . A set of B repeated bootstrap samples $D_N^{(b)}$, $b = 1, \dots, B$ 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 `ModAver/bagging.R` shows the efficacy of bagging as a remedy against overfitting.

Consider a dataset $D_N = \{x_i, y_i\}$, $i = 1, \dots, N$ 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. Figure (9.1) shows that the bagging predictor is much better than average.

Tests on real and simulated datasets showed that bagging can give a substantial gain of 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 crossover 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 $h(\cdot, \alpha_N^{(b)})$, $b = 1, \dots, B$ can proceed independently of the others. At the same time, bagging is a relatively easy way to improve an existing method. It simply needs adding

1. a loop that selects the bootstrap sample and sends it to the learning machine and
2. a back-end to perform 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 [77, 154] 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 takes 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 $h_j(\cdot)$, $j = 1, \dots, m$. The predictions of the m *weak* classifiers are then combined through a *weighted majority vote* to produce the final prediction

$$h_{\text{boo}} = \text{sign} \left(\sum_{j=1}^m \alpha_j h_j(x, \alpha_N) \right)$$

The weights α_j of the different classifiers are computed by the algorithm. The idea is to give stronger 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 example (x_i, y_i) , $i = 1, \dots, N$ 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 $j = 1, \dots, m$ the probability weights are individually modified, and the classification algorithm is reapplied 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 hard 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 α_j , $j = 1, \dots, m$ that measure the importance of the classifiers and the weights w_i , $i = 1, \dots, N$ that measure the importance of the observations.

Weak learners are added until some desired low training error has been achieved.

This is the algorithm in detail:

1. Initialise the observation weights $w_i = 1/N$, $i = 1, \dots, N$.
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 α_j gets larger as $\widehat{\text{MME}}_{\text{emp}}^{(j)}$ gets smaller.

3. (d) For $i = 1, \dots, N$ 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 normalised 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 `ModAver/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 clinical measures. We consider a training set of $N = 40$ and a test set of 160 points. 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 weak classifier. The misclassification rate of the boosted classifier is $\widehat{\text{MME}}_{\text{ts}} = 0.3$.

•

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_{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 also showed how to bound the generalisation 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 examples 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. Initialise the observation weights $w_i = 1/N$, $i = 1, \dots, N$.
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 example by the j classifiers h_1, \dots, 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 `ModAver/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) and other properties (clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion,...). The classification task is to predict the class of breast cancer on the basis of clinical measures. We consider a training set of size $N = 400$ and a test set of size 299. 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 $\widehat{\text{MME}}_{\text{ts}} = 0.010$.

•

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, she 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.

9.3.3 Bagging and boosting

This section makes a short comparison of bagging and boosting techniques. First of all, in terms of bias/variance trade-off, it is important to stress that the rationale of bagging is to reduce variance of low bias (and then high variance) learners trained on i.d. (identically distributed) data while boosting aims to reduce sequentially the bias of weak learners trained on non i.d. data.

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. Also, a number of recent theoretical results show that boosting is fundamentally different from bagging [95].

Some caveats are notwithstanding worth mentioning: the actual performance of boosting on a particular problem is dependent on the data and the nature of the weak learner. Also boosting can fail to perform well given insufficient data, overly complex weak hypothesis, and definitely too weak hypothesis.

9.4 Random Forests

Ensemble learning is efficient when it combines low bias and independent estimators, like non-pruned decision trees.

Random Forests (RF) is an ensemble learning technique proposed by Breiman [40] which combines bagging and random feature selection by using a large number of non-pruned decision trees. The rationale of RF is to reduce the variance by decorrelating as much as possible the single trees. This is achieved in the tree-growing process through a random selection of the input variables. In a nutshell, the algorithm consists in:

1. generating by bootstrap a set of B training sets,
2. fitting to each of them a decision tree $h_b(\cdot, \alpha_b)$, $b = 1, \dots, B$ where the set of variables considered for each split (Section 8.1.4.2) is a *random subset* of size n' of the original one (feature bagging),
3. storing at each split for the corresponding split variable, the improvement of the cost function,
4. returning as the final prediction the average of the B predictions

$$h_{\text{rf}}(x) = \frac{1}{B} \sum_{b=1}^B h_b(x, \alpha_b)$$

in a regression task and the majority vote in a classification task,

5. returning for each variable an *importance* measure.

Suppose that the B trees in the forest are almost unbiased, have a comparable variance $\text{Var}[\mathbf{h}_b] = \sigma^2$ and a mutual correlation ρ . The RF regression predictor h_{rf} is then almost unbiased and from (2.18.110) its variance is

$$\text{Var}[\mathbf{h}_{\text{rf}}] = \frac{(1 - \rho)\sigma^2}{B} + \rho\sigma^2$$

It appears then by increasing the forest size B and making the trees as uncorrelated as possible, a Random Forest strategy reduces the resulting variance.

A rule of thumb consists of setting the size of the random subset to $n' = \sqrt{n}$. The main hyperparameters of RF are the hyperparameters of single trees (e.g. depth, max number of leaves), the number B of trees and the size n' of the random feature set. Note that by reducing n' we make the trees less correlated, yet we increase the bias of each single tree (and then of the RF) by constraining its number of features. In particular, a too small number n' may be detrimental to accuracy in configurations with very large n and small number of informative features.

9.4.1 Why are Random Forests successful?

Random Forests are often considered among the best "off-the-shelf" learning algorithms since they do not require complex tuning to perform reasonably well on challenging tasks. There are many reasons for their success [75]: (i) they use an out-of-bag (Section 5.9.1) strategy to effectively manage the bias/variance trade-off and to assess the importance of input variables, (ii) since based on trees, they easily cope with mixtures of numeric and categorical predictor variables, (iii) they are resilient to input outliers and invariant under monotone input transformation, (iv) they embed a feature ranking mechanism based on an *importance* measure related to the average cost function decrease during splitting, (v) they are fast to construct and can be made massively parallel and (vi) there exist a number of very effective implementations (e.g. in the R package `randomForest`) and enhanced version (notably gradient boosting trees).

9.5 Gradient boosting trees

Gradient boosting (GB) trees are an enhanced version of averaging algorithms which rely on combining m trees according to a *forward stage-wise additive strategy* [95]. The strategy consists of adding one component (e.g. a tree) at the time: after m iterations, the resulting model is the sum of the M individual trees

$$h_m(x) = \sum_{j=1}^m T(x, \alpha_j)$$

Given $j-1$ trees, the j th tree is learned such to compensate for the error between the target and the current ensemble prediction $h_{j-1}(x)$. This means that

$$\alpha_j = \arg \min_{\alpha} \sum_{i=1}^N L(y_i, h_{j-1}(x_i) + T(x_i, \alpha)) \quad (9.5.1)$$

where α_j contains the j th tree parameters, e.g. the set of disjoint regions and the local model holding in each region. Note that in the forward stage-wise philosophy, no adjustment of the previously added trees is considered.

It can be shown that, for a regression task with a squared error loss function L , the solution α_j corresponds to the regression tree that best predicts the residuals

$$r_i = y_i - h_{j-1}(x_i), \quad i = 1, \dots, N$$

It follows that gradient boosting uses averaging more for reducing the overall bias of the predictor rather than for reducing variance (e.g. random forests).

Gradient based versions exist for other differentiable loss criteria and for classification tasks. Also weighted versions of (9.5.1) exist

$$(\alpha_j, w_j) = \arg \min_{\alpha, w} \sum_{i=1}^N L(y_i, h_{j-1}(x_i) + wT(x_i, \alpha))$$

where the contribution α_j of each new tree is properly tuned.

A stochastic version of gradient boosting has been proposed in [79] where at each iteration only a subsample of the training set is used to train the new tree.

Though gradient-boosting algorithms are considered ones of the most promising in complex learning tasks, it is recommended to remember that their accuracy depends, like all learning algorithms, on a number of hyperparameters, notably the size of the constituent trees, the number m of iterations, contribution w_j of each tree, loss function degree and subsample size.

9.6 Conclusion

The averaging of ensembles of estimators relies on the counter-intuitive principle that combining predictors is (most of the time) more convenient than selecting (what seems to be) the best. This principle is (probably together with the idea of regularisation) one of the most genial and effective ideas proposed by researchers in Machine Learning¹. Most state-of-the-art learning strategies do owe a considerable part of their success to the integration of the combination principle. Such principle is so powerful that some authors suggest nowadays not to include combination in the assessment of learning strategies (e.g. in new publications) given the risk that the only visible beneficial effect is the one due to the combination.

The fact that this idea might appear counter-intuitive sheds light on the stochastic nature of the learning problem and the importance of taking a stochastic perspective to really grasp the problem of learning and generalising from a finite set of observations.

9.7 Exercises

1. Verify by Monte Carlo simulation the relations (3.10.34) and 3.10.33 concerning the combination of two unbiased estimators.

Hint: define an estimation task (e.g. estimate the expected value of a random variable) and choose two unbiased estimators.

¹...and a note of distinction should be here definitely attributed to the seminal work of researchers like Jerome H. Friedman and Leo Breiman.

Chapter 10

Feature selection

In many challenging learning tasks, the number of inputs (or features) may be extremely high: this is the case of bioinformatics [153] where the number of variables (typically markers of biological activity at different functional levels) may go up to hundreds of thousands. The race to high-throughput measurement techniques in many domains allows us to easily foresee that this number could grow by several orders of magnitude.

Using such a large number of features in learning may negatively affect generalisation performance, especially in the presence of irrelevant or redundant features. Nevertheless, traditional supervised learning algorithms techniques have been designed for supervised tasks where the ratio between the input dimension and the training size is small, and most inputs (or features) are informative. As a consequence, their accuracy may rapidly degrade when used in tasks with few observations and a huge number of inputs.

At the same time, it is common to make the assumption that data are *sparse* or possess an intrinsic low dimensional structure. This means that most input dimensions are correlated, only a few of them contain information or equivalently that most dimensions are irrelevant for the learning task.

For this reason, learning pipelines include more and more a feature selection phase aiming to select a small subset of informative (or relevant) features to capture most of the signal and avoid variance and instability issues during learning. In that sense, feature selection can be seen as an instance of model selection problem where the alternative models do not differ in terms of functional representation but in terms of the used subset of inputs.

Example

This example illustrates the impact of the number of features on the model variance in a learning task with a comparable number of features and observations. Let us consider a linear regression dependency

$$\mathbf{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \mathbf{w}$$

where $\text{Var}[\mathbf{w}] = 0.5$, $\beta_0 = 0.5$, $\beta_1 = -0.5$, $\beta_2 = 0.5$, $\beta_3 = -0.5$, $\beta_4 = 0.5$. Suppose we collect a dataset of $N = 20$ input/output observations where the input set ($n = 8$) contains, together with the four variables $\mathbf{x}_1, \dots, \mathbf{x}_4$, a set of 4 irrelevant variables $\mathbf{x}_5, \dots, \mathbf{x}_8$.

Let us consider a set of linear regression models with an increasing number of features, ranging from zero (constant model) to 8.

The script `FeatureSel/bv_linfs.R` illustrates the impact of the number of features on the average bias (estimated by Monte Carlo) and the average variance

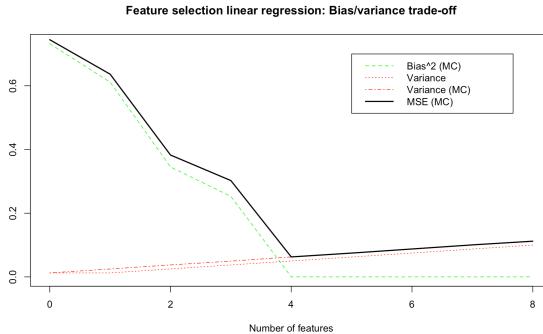


Figure 10.1: Trade-off bias/variance for different number of features. Bias and variance are averaged over the set of N inputs.

(both analytical and Monte Carlo estimated) of the predictors. Figure 10.1 shows that the larger is the number of features, the higher is the prediction variance. Note that the analytical form of the variance of a linear model prediction is presented in Section 7.1.14. The bias has the opposite trend, reaching zero once the 4 inputs $\mathbf{x}_1, \dots, \mathbf{x}_4$ are included in the regression model. Overall, the more variables are considered, the more bias is reduced at the cost of an increased variance. If a variable has no predictive value (e.g. it belongs to the set $\mathbf{x}_5, \dots, \mathbf{x}_8$), considering it merely increases the variance with no benefit in terms of bias reduction. In general, if the addition of a variable has a small impact on bias then the increase in prediction variance may exceed the benefit from bias reduction [125]. The role of a feature selection technique should be to detect those variables and remove them from the input set.

•

The benefits of feature selection have been thoroughly discussed in literature [87, 88]:

- facilitating data visualisation and data understanding,
- reducing the measurement and storage requirements,
- reducing training and utilisation times of the final model,
- defying the curse of dimensionality to improve prediction performance.

At the same time, feature selection implies additional time for learning since it introduces an additional layer to the search in the model hypothesis space.

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 modelling tasks.

The main issue in supervised learning is that the sparsity of data increases exponentially with the dimension n . This can be illustrated by several arguments.

Let us consider a n dimensional space and a unit volume around a query point $x_q \in \mathbb{R}^n$ (Figure 10.2) [95]. Let $V < 1$ be the volume of a neighbourhood hypercube of edge d . It follows that $d^n = V$ and $d = V^{1/n}$. Figure 10.3 illustrates the link

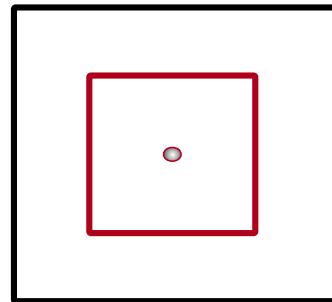
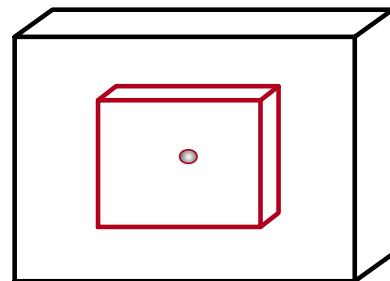
$n=1 \ d=1/2 \ V=1/2$  $n=2 \ d=1/2 \ V=1/4$  $n=3 \ d=1/2 \ V=1/8$ 

Figure 10.2: Locality and dimensionality of the input space for different values of n : unit volume (in black) around a query point (circle) containing a neighbourhood (in red) of volume V and edge d .

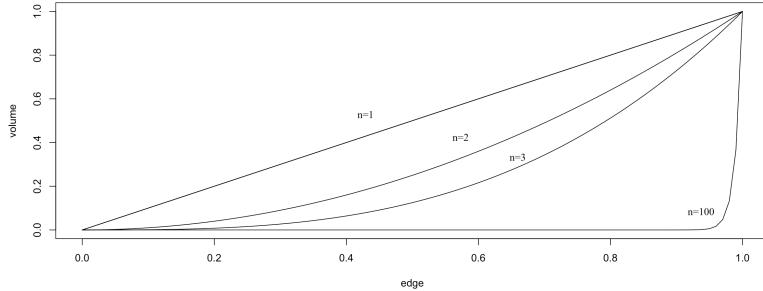


Figure 10.3: Neighbourhood volume vs. edge size for different values of n .

between neighbourhood volume V and edge size d for different values of n . It appears that for a given neighbourhood volume V , the edge length increases by increasing n while for a given edge length d , the neighbourhood volume decreases by increasing n . For instance if $V = 0.5$ we have $d = 0.7, 0.87, 0.98$ for $n = 2, 5, 50$; if $V = 0.1$ we have $d = 0.3, 0.63, 0.95$ for $n = 2, 5, 50$. This means that for $n = 50$ we need to have an edge length which is 95% of the unit length if we want to barely cover 10% of the total volume.

Let us now assess the impact of dimensionality on the accuracy of a local learning algorithm (e.g. k nearest neighbour) by considering the relation between the training set size N , the input dimension n and the number of neighbours k . If the N points are uniformly distributed in the unit volume around the query point, the number of neighbours k in the neighbourhood V amounts to roughly $k = NV$. Given the value of N and k (and consequently the local volume V) the edge d of the neighbourhood increases with the dimension n and converges rapidly to one (Figure 10.4). This implies that if we use a k NN (nearest neighbour) learner for two supervised learning tasks with same N but different n , the degree of locality of the learner (represented by the length of d) is the smaller the larger is n . Analogously if N and $0 < d < 1$ are fixed, the number $k = Nd^n$ of neighbours in V decreases by increasing n . In other terms, as n increases the amount of local data goes to zero (Figure 10.5) or equivalently all data sets are sparse for large n .

Let us now consider the case where $k > 0$ and $0 < d < 1$ (degree of locality) are fixed and N may be adjusted (e.g. by observing more points). Since

$$N = k/d^n$$

we need to exponentially grow the size of the training set N to guarantee a constant k for increasing n . Suppose that $k = 10$, $d = 0.1$ and $N = 100$ for $n = 1$. If we want to preserve the same number k of neighbours for increasing n then N has to grow according to the following law

$$N = k/d^n = \frac{10}{(1/10)^n} = 10^{n+1}$$

For instance we need to observe $N = 10^6$ observations for $n = 5$ if we want the same degree of locality we had for $n = 1$. This implies that given two supervised learning tasks (one with $n = 1$ and the other with $n \gg 1$), the second should be trained with a number N of a much higher order of magnitude (Figure 10.6) to guarantee the same degree of locality of the $n = 1$ configuration.

Another interesting result about the impact of dimensionality on data distribution is the following: given N observations uniformly distributed in a n -dimensional

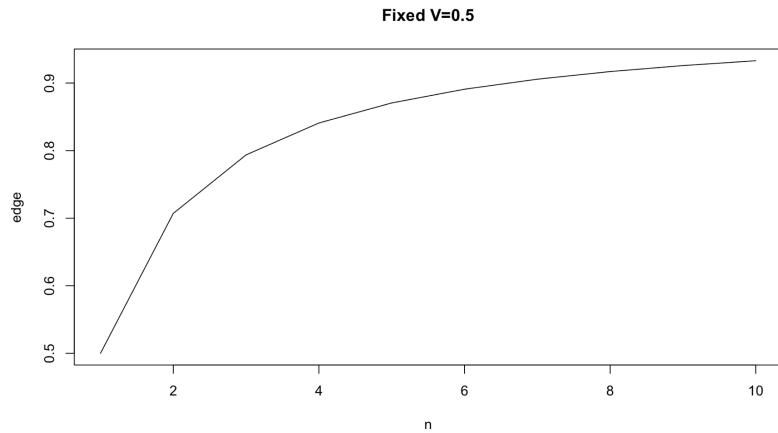


Figure 10.4: Neighbourhood edge size vs. dimension n (for fixed N and k)

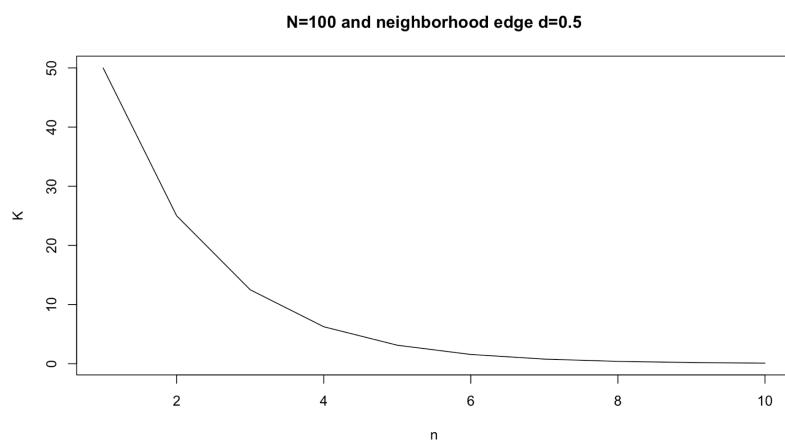


Figure 10.5: Number of neighbours K vs. dimension n for fixed N and d

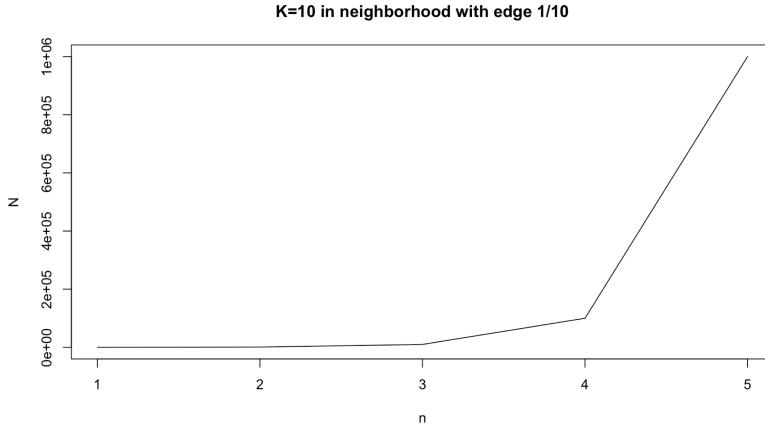


Figure 10.6: Number of training examples required to preserve the same kind of locality obtained for $n = 1$ with $k = 10$ and $d = 0.1$

unit ball centred at the origin, the median of the distance from the origin to the closest data point is $(1 - 1/2^{1/N})^{1/n}$ (Figure 10.7).

All those considerations should sound like a warning for those willing to extend local learning approaches to large dimensional settings where familiar notions of distance and closeness lose their meaning and relevance. Large dimensionality induces high sparseness with negative impact on predictive accuracy as shown by the bias/variance decomposition in (8.1.54). For a fixed N and by increasing n the algorithm is more and more exposed to one of those two low generalisation configurations: i) too small k , i.e. too few points are close to the query points (with negative impact in terms of variance) or ii) too large d implying that the nearest neighbours are not sufficiently close the query point (with negative impact on bias).

Though, from a bias/variance perspective, the curse of dimensionality is particularly harmful for local learning strategies, the other learning strategies should not be considered immune either. A too large n/N ratio implies an overparametriza-

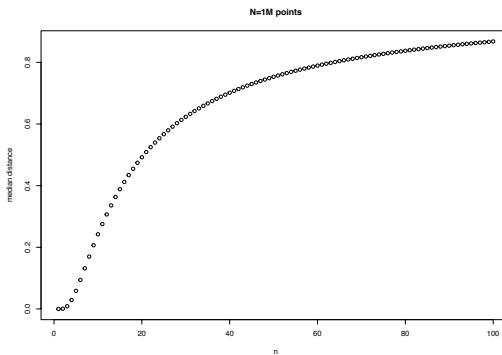


Figure 10.7: Median nearest neighbour as a function of n for very large N .

tion of the learned hypothesis and a consequent increase of the variance term in the generalisation error which is hardly compensated by the related bias reduction. For this reason, the adoption of a feature selection step is more and more common in modern machine learning pipelines.

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 learning algorithm's performance. Examples are methods that select variables by ranking them through compression techniques (like PCA or clustering) or computing correlation with the output.
- **Wrapper methods:** these methods assess subsets of variables according to their usefulness to a given predictor. The method searches 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. Examples are the stepwise methods proposed in linear regression analysis (notably the `leaps` subset selection algorithm available in R [16]).
- **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 regularisation techniques (e.g. lasso)

Note that, in practice, hybrid strategies combining the three approaches above are often considered as well. For instance in the case of a huge dimensional task (e.g. $n > 1000K$ as in epigenetics) it would make sense to first reduce the size of features to a more reasonable size (e.g. some thousands or hundreds of features) by filtering and then use some search approaches within this smaller space.

10.3 Filter methods

Filter methods are commonly used in very large dimensional tasks (e.g. $n > 2000$) for the following reasons: they easily scale to very high-dimensional datasets, they are quick because computationally simple, and they are independent of the classification algorithm. Also, since feature selection needs to be performed only once, they can be integrated into validation pipelines comparing several classifiers.

However, they are not perfect. Filter methods, by definition, ignore any interaction with the classifier and are often univariate or low-variate. The relevance of each feature is assessed separately, thereby ignoring feature dependencies. This may be detrimental in case of complex multivariate dependencies.

10.3.1 Principal component analysis

Principal component analysis (PCA) is one of the oldest and most popular preprocessing methods to perform dimensionality reduction. It returns a set of linear combinations of the original features so as to retain most of their variance and their information. Those combinations may be used as compressed (or latent) versions of the original features and used to perform learning in a lower dimensional space.

The method consists of projecting the data from the original orthogonal space \mathcal{X} into a lower-dimensional space \mathcal{Z} , in an unsupervised manner, maximising the

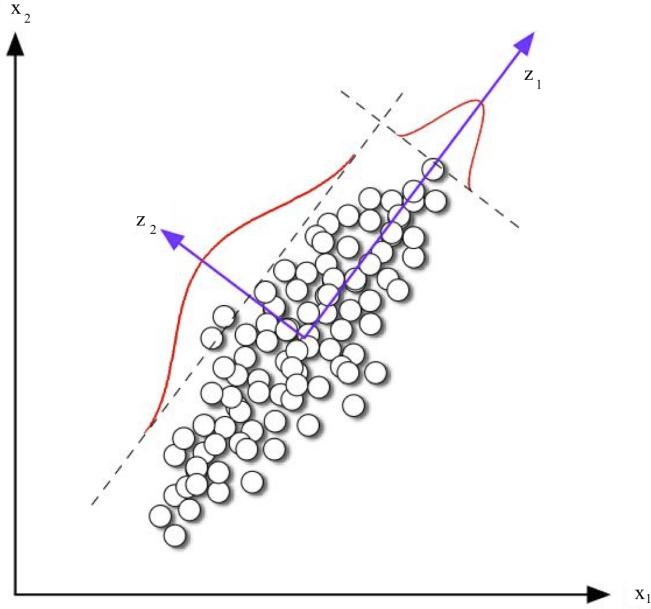


Figure 10.8: Two first principal components for a $n = 2$ dimensional Gaussian distribution.

variance and minimising the loss due to the projection. The new space is orthogonal (as the original) and its axes, called *principal components*, are specific linear combinations of the original ones.

The first principal component (i.e. the axis z_1 in Figure 10.8) is the axis along which the projected data have the greatest variation. Its direction $a^* = [a_1^*, \dots, a_n^*] \in \mathbb{R}^n$ is obtained by maximising the variance of

$$\mathbf{z} = a_1 \mathbf{x}_{\cdot 1} + \dots + a_n \mathbf{x}_{\cdot n} = a^T \mathbf{x}$$

a linear combination of the original features. It can be shown that a^* is also the eigenvector of the covariance matrix $\text{Var}[\mathbf{x}]$ associated to the largest eigenvalue [57].

The procedure for finding the other principal components is based on the same principle of variance maximisation. The second principal component (i.e. the axis z_2 in Figure 10.8) is the axis, orthogonal to the first, along which the projected data have the largest variation, and so forth.

10.3.1.1 PCA: the algorithm

Consider the training input matrix X having size $[N, n]$. The PCA consists of the following steps:

1. the matrix X is normalised and transformed to a matrix \tilde{X} such that each column $\tilde{X}[:, j]$, $j = 1, \dots, n$, has null mean and unit variance¹,
2. the Singular Value Decomposition (SVD) [84] (Appendix B.6.10) of \tilde{X} is computed

$$\tilde{X} = UDV^T$$

where U is a $[N, N]$ matrix with orthonormal columns, D is a $[N, n]$ rectangular diagonal matrix with diagonal singular values $d_1 \geq d_2 \geq \dots \geq d_n \geq 0$,

¹A R dataframe may be easily normalised by using the R command `scale`.

$d_j = \sqrt{\lambda_j}$ with λ_j eigenvalue of $X^T X$ and V is a $[n, n]$ matrix whose orthonormal columns are the eigenvectors of $X^T X$,

3. the matrix \tilde{X} is replaced by the linear transformation

$$Z = \tilde{X}V = UD \quad (10.3.1)$$

whose columns (also called eigen-features) are a linear combination of the original features and the related variances are sorted in a decreasing order,

4. a truncated version of Z made of the first $h < n$ columns (associated to the h largest singular values) is returned.

But how do we select the convenient number h of eigen-features? In the literature, three main strategies are considered:

1. fix a threshold α on the proportion of variance to be explained by the principal components, e.g. choose h such that

$$\frac{\lambda_1 + \dots + \lambda_h}{\sum_{j=1}^n \lambda_j} \geq \alpha$$

where λ_j is the j th largest eigenvalue and $\sum_{j=1}^h \lambda_j$ is the amount of variance retained by the first h components,

2. plot the decreasing values of λ_j as a function of j (scree plot) and choose the value of h corresponding to a knee in the curve,
3. select the value of h as if it was a hyperparameter, e.g. by cross-validation.

The outcome of PCA is a rotated, compressed and lower dimension version of the original input set $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ made of $h < n$ orthogonal features $\{\mathbf{z}_1, \dots, \mathbf{z}_h\}$, sorted by decreasing variance. In that sense, PCA can be considered as a linear auto-encoder where the encoding step is performed by (10.3.1) and the reconstruction of the coded data to the original space is obtained by $\hat{X} = ZV^T$. It can also be shown [57] that the PCA implements an optimal linear auto-encoder since it minimises the average reconstruction error

$$\sum_{i=1}^N \|x_i - V^T V x_i\|^2 \quad (10.3.2)$$

which amounts, for h components, to $\sum_{j=h+1}^n \lambda_j / N$.

PCA works in a completely unsupervised manner since the entire algorithm is independent of the target \mathbf{y} . Though such unsupervised nature reduces the risk of overfitting, in some cases, it may cause a deterioration of the generalisation accuracy since there is no reason that principal components be associated with \mathbf{y} . For instance, in the classification example of Figure 10.9, the choice of the first PCA component would reduce the accuracy of the classifier instead of increasing it. In order to account both for input variation and correlation with the target, supervised versions of PCA exist, like principal component regression or partial least squares.

Another limitation of PCA is that it does not return a subset but a weighted average of the original features (eigen-feature). In some cases, e.g. in bioinformatics gene selection, PCA is then not recommended since it may hinder the interpretability of the resulting model.

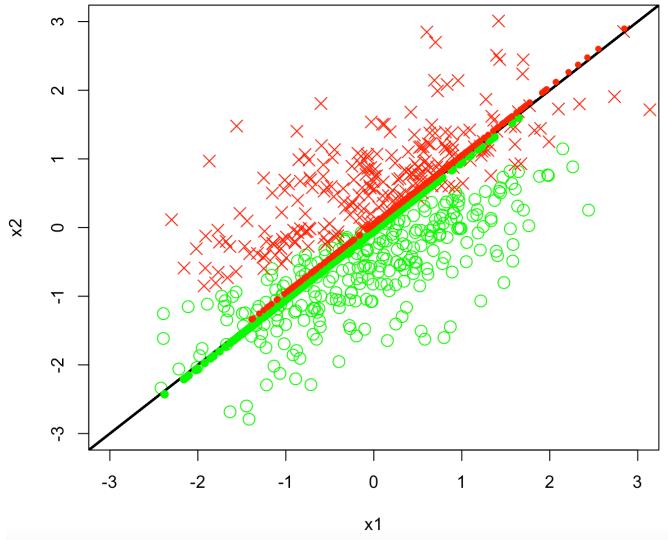


Figure 10.9: A separable $n = 2$ dimensional binary classification task reduced to a non separable one because of PCA dimensionality reduction.

R script

The scripts `FeatureSel/pca.R` and `FeatureSel/pca3D.R` illustrate the PCA decomposition in the $n = 2$ and $n = 3$ case for Gaussian distributed data and compute the reconstruction error (10.3.2).

The script `FeatureSel/pca_uns.R` illustrates the limits of PCA due to its unsupervised nature. Consider a binary classification task with $n = 2$ and a separating boundary between the two classes which is directed as the first component. In this case a dimensional reduction is rather detrimental to the final accuracy since it transforms the separable $n = 2$ problem into a non separable $n = 1$ problem (Figure 10.9).

•

PCA is an example of linear dimensionality reduction. In the machine learning literature, however, there are several examples of nonlinear versions of PCA: among the most important we mention the kernel-based version of PCA (KPCA) and (deep) neural auto-encoders (Section 8.1.2).

10.3.2 Clustering

Clustering, also known as unsupervised learning, is presented in Appendix A. Here we will discuss how it plays a role 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-neighbour clustering:** the number of clusters is set by the user, then each variable is assigned to a cluster at the end of an iterative procedure. Examples are Self Organizing Maps (SOM) and K-means.

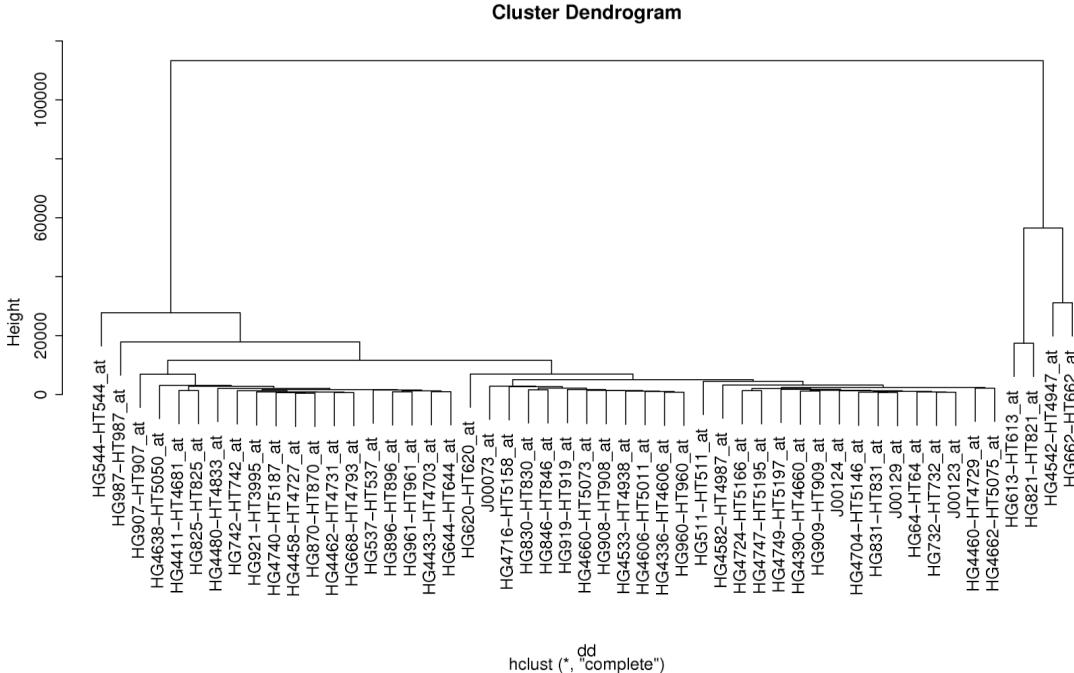


Figure 10.10: Dendrogram.

- **Agglomerative clustering:** it is a bottom-up method where clusters are initially empty and sequentially filled with variables. An example is hierarchical clustering (R command `hclust`) which starts by considering all the variables as belonging to separate clusters. Next, it joins pairs of similar features in the same cluster and then it proceeds hierarchically by merging the closest pairs of clusters. The algorithm requires a measure of dissimilarity between sets of features and a linkage criterion that quantifies the set dissimilarity as a function of the set elements pairwise distances. The visual output of hierarchical clustering is a *dendrogram*, a tree diagram used to illustrate the arrangement of the clusters. Figure 10.10 illustrates the dendrogram returned by a clustering of features in a bioinformatics task. Note that the dendrogram returns different clusters of features (and a different number of clusters) at different heights. The choice of the optimal height cut is typically done by means of a cross-validation strategy [120].

Clustering and PCA are both unsupervised dimensionality reduction techniques, which are commonly used in several domains (notably bioinformatics). However, the main advantage of clustering resides in the higher interpretability of the outcome. Unlike the PCA linear weighting, the grouping of the original features is much more informative and may return useful insights to the domain expert (e.g. about the interaction of a group of genes in a pathology [89]).

10.3.3 Ranking methods

Unlike PCA and clustering, ranking methods are supervised filters since they take into account the relation between inputs and target y to proceed with the selection. Ranking methods consist of three steps: i) they first assess the importance (or

relevance) of each variable for the output by using a univariate measure, ii) they rank them in decreasing order of relevance and iii) select the top k variables.

Relevance measures commonly used in assessing a feature are:

- the Pearson linear correlation (the larger, the more relevant);
- in case of binary classification tasks, the p-value of hypothesis tests like t-test or Wilcoxon (the lower, the more relevant).
- mutual information (Section 2.15.6) (the larger the more relevant).

Ranking 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. Two typical situations where ranking does not perform well are complementary and highly redundant configurations. In the complementary case, two input features are very low informative about the target, yet they are very informative if taken together (see the XOR configuration later). Because of their low univariate relevance, ranking methods will rank them low and consequently discard them. Otherwise, two variables could be both highly relevant about the target but very similar (or identical). In this redundant case, both will be ranked very high and selected, despite their evident redundancy.

Feature selection in a gene expression dataset

A well-known high-dimensional classification task is gene expression classification in bioinformatics, where the variables correspond to genomic features (e.g. gene probes), the observations to patients and the targets are biological phenotypes (e.g. cancer grade). Because of the growing capabilities of sequencing technology, the number of genomic features is typically much larger than patient cohorts' size. In the script `FeatureSel/featsel.R` we analyze the microarray dataset from [85]. This dataset contains the genome expressions of $n = 7129$ genes for $N = 72$ patients, and $V = 11$ related phenotype variables. The expression matrix X and the phenotype vector Y are contained in the dataset `data(golub)`. The script studies the dependency between the gene expressions and the binary phenotype ALL.AML indicating the leukaemia type: lymphoblastic leukaemia (ALL) or acute myeloid leukaemia (AML). Relevant features are selected by correlation ranking and the misclassification errors are computed for different sizes of the feature 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 the interaction between features, and this in a supervised manner. Unfortunately, this implies a much higher computational cost, especially in the case of expensive training phases. Also, the dependence of the final result on the learner choice could be considered as a *nuisance factor* confounding the impact of the feature set on the final accuracy². In other terms, the issue is: was the feature set returned by the wrapper because it was good in general or only for that specific learner (e.g. a neural network)?

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

$$w[j] = \begin{cases} 0 & \text{if the input } j \text{ does NOT belong to the set of features} \\ 1 & \text{if the input } j \text{ belongs to the set of features} \end{cases}$$

²This is the reason why a blocking factor approach to control the variability due to the learner algorithm and improve the robustness of the solution has been proposed in [29].

Wrappers look for the optimal vector $w^* \in \{0, 1\}^n$ such that

$$w^* = \arg \min_{w \in W} \widehat{\text{MISE}}_w \quad (10.4.3)$$

where $\widehat{\text{MISE}}_w$ is the estimation of the generalisation error of the model based on the set of variables encoded by w . Since in real-settings the actual generalisation error is not directly observable, the computation of $\widehat{\text{MISE}}_w$ requires the definition of a learner and of a validation strategy.

Note that the number of vectors in W is equal to 2^n , that it doubles for each new feature and that for moderately large n (e.g. $n > 20$), the exhaustive search is no more affordable. For this reason, wrappers typically rely on heuristic search strategies.

10.4.1 Wrapping search strategies

Three greedy strategies are commonly used to avoid the exponential complexity $O(2^n)$ of the exhaustive approach:

- **Forward selection:** the procedure starts with no variables and progressively incorporates features. The first selected input is the one that returns the lowest generalisation error. The second input selected is the one that, together with the first, has the lowest error, and so on, until no further improvement is made or the required number of features is attained. An example of forward selection is implemented in the R script `FeatureSel/fs_wrap.R`.
- **Backward selection:** it works in the opposite direction of the forward approach by progressively removing features from the original feature set. The procedure starts by learning a model using all the n variables and, therefore, requires at least $N > n$. Then the impact of dropping one feature at a time from the current subset is assessed. The feature which is actually removed is the one that yields the lowest generalisation error after deletion. In other terms, it is the one whose absence causes the lowest increase (or highest decrease) of the generalisation error. The procedure iterates until the desired number of features is attained.
- **Stepwise selection:** it combines the previous two techniques by testing for each set of variables, first the removal of features belonging to the set, then the addition of variables not in the set.

It can be shown that the forward and the backward strategies have a $O(n^2)$ time complexity in the case of n steps: since the i th step ($i = 0, \dots, n - 1$) requires $n - i$ assessments to select (or remove) the $(i + 1)$ th feature, the computational complexity for n steps amounts to $\sum_{i=0}^{n-1} (n - i) = \frac{n(n+1)}{2}$.

Nevertheless, since such complexity cannot be affordable either in case of very large n , it is common usage to reduce first the number of features by using a fast filter method (e.g. ranking) and then apply a wrapper strategy on the remaining number of features. Another trick consists of limiting the maximum size of the feature set, then reducing the computational cost.

10.4.2 The Cover and van Campenhout theorem

The rationale of forward and backward greedy heuristics is that an optimal set of size k should contain the optimal set of size $k - 1$. Though this seems intuitive, in the general case there is no reason why this relation should hold. A formal result in that sense is provided by the Cover and van Campenhout theorem [59], which

contains a negative result about the aim of wrapper search techniques to find the optimal subset by local procedures.

Let us consider a learning problem and denote by $R^*(w)$ the lowest functional risk (5.2.6) for the subset of variables w . Cover and van Campenhout proved that the only generally valid (i.e. which holds for all data distributions) monotonic relation linking feature size and generalisation is :

$$w_2 \subset w_1 \Rightarrow R^*(w_1) \leq R^*(w_2) \quad (10.4.4)$$

i.e. by adding variables we reduce the minimal risk³.

Given n features, *any* ordering of the 2^n subsets which is consistent with the above constraint is indeed possible. This means for any possible ordering, there exists a distribution of the data that is compatible with that. If the three variables optimal set is

$$\{\mathbf{x}_{.1}, \mathbf{x}_{.3}, \dots, \mathbf{x}_{.13}\}$$

there is no guarantee that the best set of four variables is a superset of w_1 (as it is assumed in forward selection). According to this theorem, there exists a distribution for which the best set of 4 features could well be

$$\{\mathbf{x}_{.2}, \mathbf{x}_{.6}, \mathbf{x}_{.16}, \mathbf{x}_{.23}\}$$

since this is not in contradiction with the constraint (10.4.4). In other words, the Cover and van Campenhout theorem states that there are data distributions for which forward/backward strategies could be arbitrarily bad.

10.5 Embedded methods

They are typically less computationally intensive than wrapper methods but are specific to a learning machine. Well-known examples are classification trees, Random Forests (Section 9.4), Naive Bayes (Section 8.2.3.1), shrinkage methods and kernels.

10.5.1 Shrinkage methods

Shrinkage is a technique to improve a least-squares estimator by regularisation and consists of reducing the model variance by adding constraints on the value of coefficients. In what follows, we present two shrinkage approaches that penalise the least-squares solutions having a large number of coefficients with values different from zero. The rationale is that only those variables, whose impact on the empirical risk is considerable, deserve a coefficient different from zero and should appear in the fitted model. Shrinkage is an implicit (and more continuous) embedded manner of doing feature selection since only a subset of variables contributes to the final predictor.

³Note that this relation refers to the optimal model that could be learned with the input subset w and that the notion of lowest functional risk does not take into consideration the model family nor the finite-size setting. In other terms, this inequality refers only to the bias and not the variance component of the generalisation error. So in practice though in theory $R^*(w_1) \leq R^*(w_2)$ it could happen that $G_N(w_1) \geq G_N(w_2)$ where G_N is the generalisation error of the model learned with N observations

10.5.1.1 Ridge regression

Ridge regression is an example of shrinkage method applied to least squares regression

$$\begin{aligned}\hat{\beta}_r &= \arg \min_b \left\{ \sum_{i=1}^N (y_i - x_i^T b)^2 + \lambda \sum_{j=1}^p b_j^2 \right\} = \\ &= \arg \min_b ((Y - Xb)^T (Y - Xb) + \lambda b^T b)\end{aligned}$$

where $\lambda > 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ , the greater the amount of shrinkage. Note that if $\lambda = 0$ the approach boils down to a conventional unconstrained least-squares.

An equivalent formulation of the ridge problem is

$$\begin{aligned}\hat{\beta}_r &= \arg \min_b \sum_{i=1}^N (y_i - x_i^T b)^2, \\ \text{subject to } &\sum_{j=1}^p b_j^2 \leq L\end{aligned}$$

where there is a one-to-one correspondence between the parameter λ and L [95].

It can be shown that the ridge regression solution is

$$\hat{\beta}_r = (X^T X + \lambda I_p)^{-1} X^T Y \quad (10.5.5)$$

where I_p is the $[p, p]$ identity matrix ($p = n + 1$) and it is typically recommended that the X columns are normalised (zero mean and unit variance) [125]. In algebraic terms, a positive λ ensures that the matrix to be inverted be symmetric and strictly positive definite.

If $n \gg N$ it is recommended to take advantage of the SVD decomposition (B.6.11) to avoid the inversion of a too large matrix [97]. If we set $X = UDV^T$ then we obtain from (10.5.5) and (B.9.14)

$$\begin{aligned}\hat{\beta}_r &= (VDU^T UDV^T + \lambda I_p)^{-1} VDU^T Y = V(R^T R + \lambda I_N)^{-1} R^T Y = \\ &= V(D^2 + \lambda I_N)^{-1} R^T Y \quad (10.5.6)\end{aligned}$$

where $R = UD$ is a $[N, N]$ matrix and I_N is the $[N, N]$ identity matrix. It follows that

$$\begin{aligned}\hat{Y} &= X \hat{\beta}_r = UDV^T V(D^2 + \lambda I_N)^{-1} D U^T Y = U \tilde{D} U^T = \\ &= \sum_{i=1}^n u_i \tilde{D}_i u_i^T Y \quad (10.5.7)\end{aligned}$$

where $\tilde{D} = D(D^2 + \lambda I_N)^{-1} D$ is a diagonal matrix. The i th diagonal term of \tilde{D} is

$$\tilde{D}_i = \frac{d_i^2}{d_i^2 + \lambda}$$

where d_i^2 is the squared singular value of X , then eigenvalue of $X^T X$ and inverse of the eigenvalue of $(X^T X)^{-1}$ which is a term appearing in (7.1.22). This shows that the smaller the eigenvalue d_i^2 , the larger is the impact on the linear model variance and the stronger is the shrinkage effect of λ on it [128].

In general, ridge regression is beneficial in numerical, statistical and interpretability terms. From a numerical perspective, it is able to deal with rank deficient matrices X and reduces the ill-conditioning of the matrix $X^T X$. From a statistical perspective, it reduces the variance of the least-squares solution $\hat{\beta}_r$ (Section (7.1.14)) at the cost of a slight bias increase. Given the predominance of the variance term in high-dimensional tasks, ridge regression enables a reduction of the generalisation error. Last but not least, pushing the absolute value of many coefficients to zero, it allows the identification of a small (then interpretable) number of input features.

10.5.1.2 Lasso

Another well-known shrinkage method is *lasso* which estimates the linear parameters by

$$\hat{\beta}_r = \arg \min_b \sum_{i=1}^N (y_i - x_i^T b)^2, \quad (10.5.8)$$

$$\text{subject to } \sum_{j=1}^p |b_j| \leq L \quad (10.5.9)$$

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 of a *quadratic programming algorithm* (details in Appendix C.3).

To formulate the problem (10.5.8) in the form (C.3.4) with linear constraints, we may write the b_j terms as the sum of two non-negative numbers

$$b_j = b_j^+ - b_j^- = \frac{|b_j| + b_j}{2} - \frac{|b_j| - b_j}{2}$$

The function to optimize becomes

$$\begin{aligned} J(b) &= b^T X^T X b - 2Y^T X b = (b^+ - b^-)^T X^T X (b^+ - b^-) - 2Y^T X (b^+ - b^-) = \\ &[b^+ \ b^-] \begin{bmatrix} X^T X & -X^T X \\ -X^T X & X^T X \end{bmatrix} \begin{bmatrix} b^+ \\ b^- \end{bmatrix} + [-2Y^T X \ 2Y^T X] \begin{bmatrix} b^+ \\ b^- \end{bmatrix} \end{aligned} \quad (10.5.10)$$

with the constraints

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ 0 & 0 & \dots & -1 \end{bmatrix} \begin{bmatrix} b^+ \\ b^- \end{bmatrix} \leq \begin{bmatrix} L \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

where the left-hand matrix is $[2p+1, 2p]$. The first line of the inequality is (10.5.9) since $\sum_{j=1}^p (b_j^+ + b_j^-) = \sum_{j=1}^p |b_j| \leq L$.

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.

Though the difference between ridge regression and lasso might seem negligible, the use of a 1-norm penalty instead of a 2-norm has a sensible impact on the number of final coefficients which are set to zero. Figure 10.11 visualises this in a bivariate case: $\hat{\beta}$ denotes the least-squares solution which would be returned by both methods if $\lambda = 0$. Since $\lambda > 0$, the minimisation combines the empirical risk function (whose contour lines are the ellipsoids around $\hat{\beta}$) and the regularisation term (whose contour lines are around the origin). Note that the only difference between the two figures is the shape of the regularisation contour lines (related to the used norm). The minimisation solution is a bivariate vector which lies somewhat (depending on the

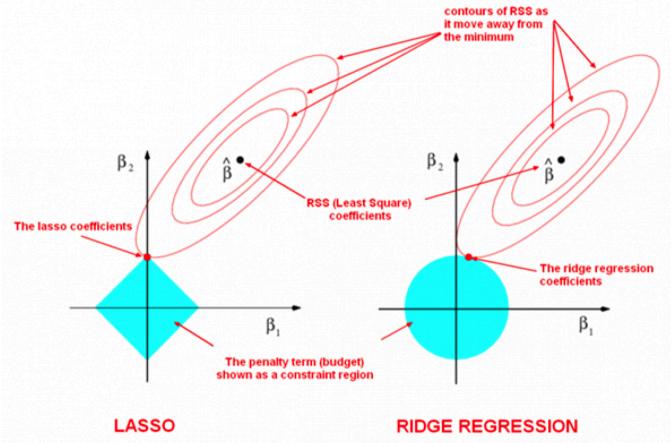


Figure 10.11: Ridge regression vs lasso [95].

λ value) at the intersection of an empirical risk contour line and a regularisation one. The figure shows that this intersection in the lasso case tends to be closer to the axis $\beta_1 = 0$, this meaning that the first estimated coefficient is set to zero. Because of the circular shape of regularisation contours, this is much less probable in the ridge regression case. As stated in [66] *the lasso does both feature selection and shrinkage while ridge regression only shrinks.*

R script

The R script `FeatureSel/lasso.R` implements the quadratic programming minimisation in (10.5.10) by using the R library `quadprog`. The script applies the lasso strategy to a regression task where the number of features n is comparable to the number of observations N and only a small number of features is relevant. The results show the impact of the constraint L on the empirical risk and the evolution of the lasso solution moving towards one of the axis. In particular the smaller L , the less importance is given to minimise J , the larger the empirical risk and the smaller is the number of estimated parameters different from zero.

•

The shrinkage approach has been very successful in recent years and several variants of the methods mentioned above exist in literature: some adopt different penalty norms, some combine different norms (e.g. Elastic-net) and some combine shrinkage with greedy search (e.g. Least Angle Regression).

10.5.2 Kernel methods

Many learning algorithms, such the perceptron, support vector machine (SVM) and PCA, process data in a linear manner through inner products (Section B.2). Those techniques are exposed to two main limitations: the linear nature of the model and the curse of dimensionality for large n .

Kernel methods [157] adapt those techniques by relying on the combination of two smart ideas: i) address large dimension n problems by solving a *dual* problem in a space of dimension N ii) generalise the notion of inner product by adopting a user-specified kernel function, i.e., a similarity function over pairs of data points.

Kernel functions operate in a high-dimensional, implicit feature space without computing the coordinates of the data in that space. This allows to take advantage of high nonlinear dimensional representations without actually having to work in the high dimensional space.

10.5.3 Dual ridge regression

We introduced the dual formulation of the linear least-squares problem in Section 7.1.18. Consider now a ridge regression problem (Section 10.5.1.1) with parameter $\lambda \in \mathbb{R}^+$. The conventional least-squares solution is the $[p, 1]$ parameter vector

$$\hat{\beta} = (X'X + \lambda I_p)^{-1} X'y$$

where $p = n + 1$ and I_p is the identity matrix of size p . Since from (B.9.14)

$$(X'X + \lambda I_p)^{-1} X' = X'(XX' + \lambda I_N)^{-1}$$

where I_N is the identity matrix of size N , the dual formulation is

$$\hat{\beta} = X'(XX' + \lambda I_N)^{-1}y = X'\alpha$$

where

$$\alpha = (K + \lambda I_N)^{-1}y$$

is the $[N, 1]$ vector of *dual variables* and $K = XX'$ is the Kernel or Gram $[N, N]$ matrix. Note that all the information required to compute α is this matrix of inner products.

The prediction for a test $[N_{ts}, p]$ dataset X_{ts} is

$$\hat{y}_{ts} = X_{ts}\hat{\beta} = X_{ts}X'\alpha = K_{ts}(K + \lambda I_N)^{-1}y$$

where K_{ts} is a $[N_{ts}, N]$ matrix with $k_{j,i} = \langle x_j, x_i \rangle, j = 1, \dots, N_{ts}, i = 1, \dots, N$.

This derivation allows transforming a n dimensional linear task into a N dimensional one. This is of course very relevant if $n \gg N$. However, the model remains linear. What about non-linear models?

10.5.4 Kernel function

A nonlinear input transformation can transform a non linearly separable classification task into an higher dimension yet easier linear classification task (Figure 10.12).

R script

The script `FeatureSel/featrans.R` shows an example of nonlinear classification task (bidimensional input space $n = 2$) (Figure 10.13 left) and the corresponding linear separable task (Figure 10.13 right) obtained with the creation of a new variable $\mathbf{x}^3 = \mathbf{x}_1^2 + \mathbf{x}_2^2$.

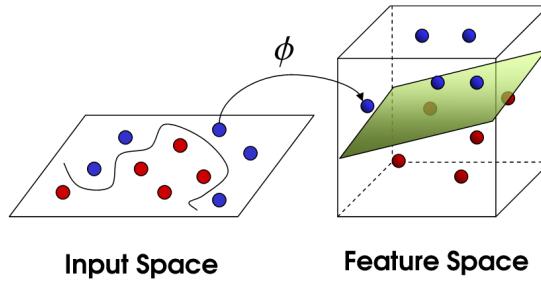


Figure 10.12: Implicit transformation of the problem to a high-dimensional space.

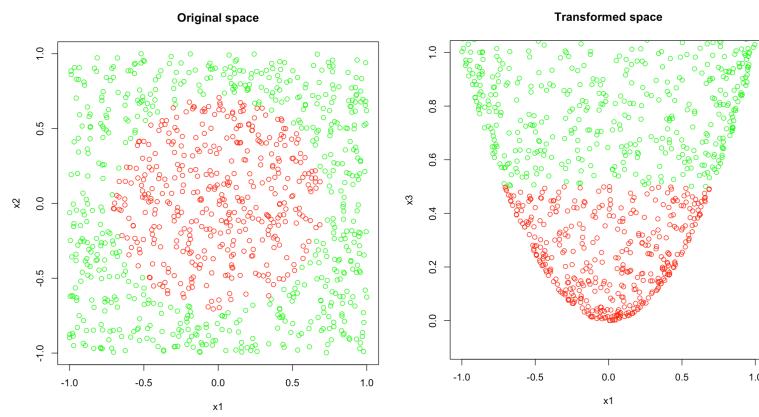


Figure 10.13: Transformation of the nonlinear classification task (left) to a linearly separable task (right).

Suppose to apply the nonlinear transformation $\Phi : x \in \mathbb{R}^n \rightarrow \Phi(x) \in \mathbb{R}^M$ to the inputs of the ridge regression problem discussed above. The prediction for an input x would now be

$$\hat{y} = y'(K + \lambda I_N)^{-1}k$$

where

$$K_{i,j} = \langle \Phi(x_i), \Phi(x_j) \rangle, \quad k_i = \langle \Phi(x_i), \Phi(x) \rangle$$

The rational of kernel methods is that those inner products can be computed efficiently without explicitly computing the mapping Φ thanks to a *kernel* function [157]. A kernel function is a function κ that for all $x, z \in X$ satisfies

$$\kappa(x, z) = \langle \Phi(x), \Phi(z) \rangle$$

where Φ is a mapping from X to a feature space F . For instance

$$\kappa(x, z) = \langle x, z \rangle^2 = \langle \Phi(x), \Phi(z) \rangle$$

where

$$\Phi : x = (x_1, x_2) \rightarrow \Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in F$$

Kernels de-couple the specification of the algorithm from the specification of the feature space since they provide a way to compute dot products in some feature space without even knowing what this space the function Φ are.

For instance

$$\kappa(x, z) = (1 + x^T z)^2$$

corresponds to a transformation to $M = 6$ dimensional space

$$\Phi(x_1, x_2) = (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$$

A Gaussian kernel $\kappa(x, z) = \exp^{-\gamma \|x - z\|^2}$ corresponds to a transformation to an infinite-dimensional space.

Theoretically, a Gram matrix must be positive semi-definite (PSD). Empirically, for machine learning heuristics, choices of a function κ that do not satisfy PSD condition may still perform reasonably if κ at least approximates the intuitive idea of similarity.

The general idea of transposing a low-dimensional method to a nonlinear high-dimensional setting by using a dual formulation is generally referred to as *kernel trick*: given any algorithm that can be expressed solely in terms of dot products, the kernel trick allows us to construct different nonlinear versions of it.

Kernel methods are together with deep learning and random forests among the most successful methods in the history of machine learning. We decided to present them in this section for their powerful strategy in dealing with settings with high dimension and low number of observations. Their strength can however turn into a weakness if we aim to scale the approach to very large N . At the same time, as for all the other methods presented in this book, their generalisation accuracy is strictly dependent on the adequate choice of the related hyperparameters. In the case of kernel methods the most important hyperparameters are the regularisation term λ , the analytical form of the kernel function and the related parameters.

10.6 Similarity matrix and non numeric data

In the previous sections, we have considered feature selection techniques for conventional supervised tasks where data are numeric and represented in a conventional tabular form D_N . What about non-conventional tasks where the training set is not

a data table but a set of items? Examples of items could be music tracks, texts, images, web sites or graphs. Often, in those cases, we are not able (or confident) in encoding each item as a numeric vector of size n . Nevertheless, we could be confident in defining a *similarity* score between pairs of items. For instance, we may use musical genre to measure the similarity between tracks or user access statistics to obtain the similarity between web sites.

As a result, we may encode the item set as a similarity matrix S of size $[N, N]$ which becomes an alternative way of representing the dataset.

A symmetric factorisation of a symmetric $[N, N]$ matrix

$$S \approx FF^T \quad (10.6.11)$$

is an approximation of the similarity matrix where F is a $[N, K]$ matrix. The matrix F may be used as an approximate K dimensional numeric representation of the non-numeric item-set.

Note that the positive definitiveness of S is a necessary and sufficient condition for having an exact factorisation, i.e. an identity in (10.6.11). This is guaranteed in the numeric case where S is the covariance matrix and the pairwise similarity is computed by dot product. In the generic non-numeric case, techniques to repair the positive definitiveness of S may be adopted. An alternative is the use of optimisation techniques to obtain F as the solution of the minimisation task

$$F = \arg \min_U \|S - UU^T\|_F^2$$

Another limitation of the factorisation approach is that it is hardly scalable for very large N . For such cases sampling based solution have been proposed in [2].

10.7 Averaging and feature selection

The role of averaging methods in supervised learning has been discussed in the previous chapter. 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 fits the data equally well, model combination approaches have been adapted to improve the robustness and stability of final, discriminative methods.

Ensemble techniques typically rely on averaging the outcome of multiple models learned with different feature subsets. A well-known technique is the *random subspace method* [99], also known as feature bagging, which combines a set of learners trained on random subsets of features.

10.8 Feature selection from an information-theoretic perspective

So far, we focused on algorithmic methods to return a subset of relevant features, without making any formal definition of relevance. In this section, we formalise the notion of feature relevance by using concepts of information theory, like entropy, mutual information and conditional information from Sections 2.15.6 and 2.15.6.1. The most important message is that the relevance of a feature is not an absolute but a context dependent concept.

10.8.1 Relevance, redundancy and interaction

This section defines 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 [112].

Definition 8.1 (Strong 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 can carry. Strong relevance indicates that the feature is always necessary for an optimal subset.

Definition 8.2 (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. This definition makes clear that for some variables (typically the majority) the relevance is not absolute but more a context-based notion. In a large variate setting, those features are the hardest to deal with since their importance depends on the other selected ones.

Definition 8.3 (Irrelevance). A variable is *irrelevant* if it is neither strongly or weakly relevant.

Irrelevance indicates that the feature is not necessary at all. This is definitely the easiest case in feature selection. Irrelevant variables should be simply discarded.

Example

Consider a learning problem where $n = 4$, $\mathbf{x}_2 = -\mathbf{x}_3 + \mathbf{w}_2$

$$\mathbf{y} = \begin{cases} 1 + \mathbf{w}, & \mathbf{x}_1 + \mathbf{x}_2 > 0 \\ 0, & \text{else} \end{cases}$$

where \mathbf{w} and \mathbf{w}_2 are noise terms. Which variables are strongly, weakly relevant and irrelevant?

•

Definition 8.4 (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}_y is said to be a *Markov blanket* of \mathbf{y} , $\mathbf{y} \notin \mathbf{M}_y$ iff

$$I(\mathbf{y}; \mathbf{X}_{-(M_y)} | \mathbf{M}_y) = 0$$

It can be shown [167, 139] that, under specific assumptions about the distribution, 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 the information of a subset of variables is the notion of interaction.

Definition 8.5 (Interaction). Given three r.v.s. \mathbf{x}_1 , \mathbf{x}_2 and \mathbf{y} we define the interaction between these three variables as

$$I(\mathbf{x}_1; \mathbf{y}) - I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2)$$

The interaction term satisfies the following relation:

$$I(\mathbf{x}_1; \mathbf{y}) - I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2) = I(\mathbf{x}_1; \mathbf{x}_2) - I(\mathbf{x}_1; \mathbf{x}_2|\mathbf{y}) = I(\mathbf{x}_2; \mathbf{y}) - I(\mathbf{x}_2; \mathbf{y}|\mathbf{x}_1)$$

In what follows, we show that it is possible to decompose the joint information of two variables in the sum of the two univariate terms and the interaction. From the chain rule (2.15.101)

$$I(\mathbf{x}_2; \mathbf{y}|\mathbf{x}_1) + I(\mathbf{x}_1; \mathbf{y}) = I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2) + I(\mathbf{x}_2; \mathbf{y})$$

we have

$$I(\mathbf{x}_2; \mathbf{y}|\mathbf{x}_1) = I(\mathbf{x}_2; \mathbf{y}) - I(\mathbf{x}_1; \mathbf{y}) + I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2)$$

By summing $I(\mathbf{x}_1; \mathbf{y})$ to both sides, from (2.15.101) it follows that the joint information of two variables about a target \mathbf{y} can be decomposed as follows:

$$\begin{aligned} I(\{\mathbf{x}_1, \mathbf{x}_2\}; \mathbf{y}) &= I(\mathbf{x}_1; \mathbf{y}) + I(\mathbf{x}_2; \mathbf{y}) - \underbrace{[I(\mathbf{x}_1; \mathbf{y}) - I(\mathbf{x}_1; \mathbf{y}|\mathbf{x}_2)]}_{\text{interaction}} = \\ &= I(\mathbf{x}_1; \mathbf{y}) + I(\mathbf{x}_2; \mathbf{y}) - \underbrace{[I(\mathbf{x}_1; \mathbf{x}_2) - I(\mathbf{x}_1; \mathbf{x}_2|\mathbf{y})]}_{\text{interaction}} \end{aligned} \quad (10.8.12)$$

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 jointly bring a higher information than the sum of the univariate terms. This is typically the case of the XOR example illustrated in Figure 10.14 [87]. 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 (10.8.12) holds also when \mathbf{x}_1 and/or \mathbf{x}_2 are sets of variables, this result sheds an interesting light about the non-monotonic nature of feature selection [182].

10.8.2 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 $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, the optimal subset of d variables is the solution of the optimisation problem

$$X^* = \arg \max_{\mathbf{X}_S \subset \mathbf{X}, |\mathbf{X}_S|=d} I(\mathbf{X}_S; \mathbf{y}) \quad (10.8.13)$$

Thanks to the chain rule (2.15.101), this maximisation task can be tackled by adopting an incremental approach (e.g. forward approach).

Let $\mathbf{X} = \{\mathbf{x}_i\}, i = 1, \dots, n$ the whole set of variables and \mathbf{X}_S the set of s variables selected after s steps. The choice of the $(s+1)$ th variable $\mathbf{x}^{(s+1)} \in \mathbf{X} - \mathbf{X}_S$ can be done by solving

$$x^{(s+1)} = \arg \max_{\mathbf{x}_k \in \mathbf{X} - \mathbf{X}_S} I(\{\mathbf{X}_S, \mathbf{x}_k\}; \mathbf{y}) \quad (10.8.14)$$

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

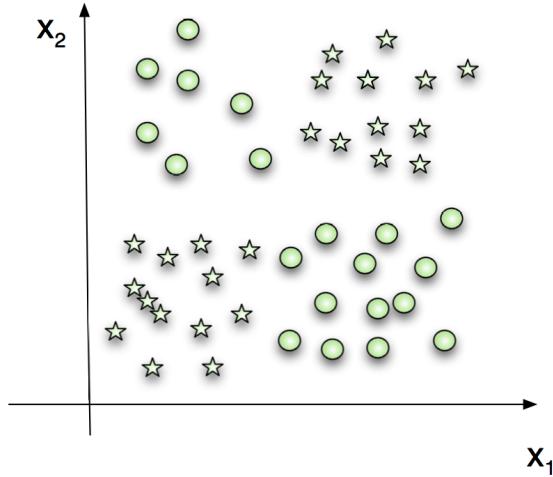


Figure 10.14: XOR classification separable task with two inputs and one binary class taking two values (stars and rounds). Two variables \mathbf{x}_1 and \mathbf{x}_2 are complementary: they bring alone no information but they bring the maximal information about \mathbf{y} when considered together.

is often inaccurate in large variate settings (i.e. large n and large s) because of ill-conditioning and high variance issues.

In literature several filter approaches have been proposed to solve the optimisation (10.8.14) 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 [76]: since according to the first (chain-rule) formulation

$$\arg \max_{\mathbf{x}_k \in \mathbf{X} - \mathbf{X}_S} I(\{\mathbf{X}_S, \mathbf{x}_k\}; \mathbf{y}) = \arg \max_{\mathbf{x}_k \in \mathbf{X} - \mathbf{X}_S} I(\mathbf{x}_k; \mathbf{y} | \mathbf{X}_S)$$

this filter adopts the low-variate 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) [140]: the mRMR method approximates at the $(s + 1)$ th step $I(\{\mathbf{X}_S, \mathbf{x}_k\}; \mathbf{y})$ with

$$I(\mathbf{x}_k; \mathbf{y}) - \frac{1}{s} \sum_{\mathbf{x}_i \in \mathbf{X}_S} I(\mathbf{x}_i; \mathbf{x}_k)$$

where s is the number of features in \mathbf{X}_S . The method implements a forward selection which selects at the $(s + 1)$ th step

$$\mathbf{x}^{(s+1)} = \arg \max_{\mathbf{x}_k \in \mathbf{X} - \mathbf{X}_S} \left[I(\mathbf{x}_k; \mathbf{y}) - \frac{1}{s} \sum_{\mathbf{x}_i \in \mathbf{X}_S} I(\mathbf{x}_i; \mathbf{x}_k) \right]$$

that is a variable which has both high relevance $I(\mathbf{x}_k; \mathbf{y})$ and low average redundancy with the set \mathbf{X}_S .

10.8.3 Information-theoretic notions and generalisation

Most of this book has dealt with generalisation error to assess, compare and select prediction models and in this chapter, we presented feature selection as an instance of model selection. Nonetheless, this last part of the chapter has been mainly referring to information-theoretic notions for performing feature selection. It is then important to provide some elucidation on how information-theoretic notions relate to generalisation error.

Given a set \mathbf{X} of input features and a target feature \mathbf{y} , the quantity $I(\mathbf{X}; \mathbf{y})$ is not directly observable and has to be estimated before use. Since

$$I(\mathbf{X}; \mathbf{y}) = H(\mathbf{y}) - H(\mathbf{y}|\mathbf{X}),$$

maximising $I(\mathbf{X}; \mathbf{y})$ in (10.8.13) is equivalent to minimise $H(\mathbf{y}|\mathbf{X})$. The term $H(\mathbf{y}|\mathbf{X})$ is the entropy (or uncertainty) of \mathbf{y} once the value of the input set \mathbf{X} is given. In the Normal case, this term is proportional to the conditional variance (Equation (2.15.91)). It follows that finding the set of inputs \mathbf{X} which minimises $H(\mathbf{y}|\mathbf{X})$ boils down to find the set of features that attains the lowest generalisation error (5.2.6).

In real-world settings, since the conditional entropy $H(\mathbf{y}|\mathbf{X})$ is not observable, it may be approximated by the generalisation error, e.g. by MISE in the regression case. Hopefully, the link between feature selection, generalisation error and information-theory becomes clear: finding the set that maximises the mutual information in (10.8.13) boils down to find the set that minimises the estimated generalisation error (10.4.3).

10.9 Assessment of feature selection

Most of the discussed techniques aim to find the best subset of features by performing a large number of comparisons and selections. This additional search layer increases inevitably the space of possible models and the variance of the resulting one. Despite the use of validation procedures, low misclassification or low prediction errors may be found only due to chance. As stated in [125], *given a sufficiently exhaustive search, some apparent pattern can always be found, even if all predictors have come from a random number generator*. This is due to the fact that, as a consequence of the search process, the set of features is dependent on the data used to train the model, introducing then what is called *selection bias* [125].

A bad (and dangerous) practice is using the same set of observations to select the feature set and assess the accuracy of the classifier. Even if cross-validation is used to assess the accuracy of the classifier, this will return an overoptimistic assessment of the generalisation error (Figure 10.15). Cross-validation has to be used to assess the *entire learning process* which is composed of both a feature selection and a classification step. This means that for each fold, both feature selection and classification has to be performed before testing on the observations set aside. Keeping feature selection out of cross-validation will return an assessment which will be as much biased as the number of observations is small.

If cross-validation cannot be carried out (e.g. because of too small training size), then the use of external validation sets is strongly recommended.

If no additional data are available, an alternative consists of comparing the generalisation accuracy returned by cross-validation on the original data with the one obtained by re-running the learning procedure with randomised datasets. This is inspired by the method of permutation testing described in Section 4.6. The procedure consists of repeating the feature selection and the cross-validation assessment

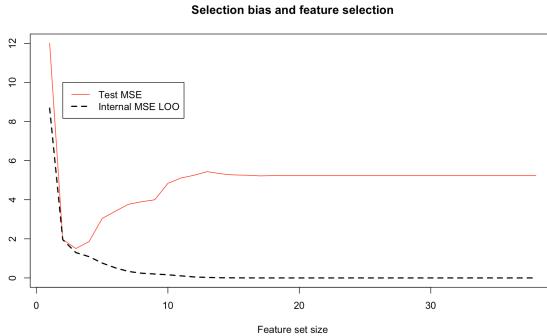


Figure 10.15: Selection bias associated to feature selection: the internal leave-one-out is an overoptimistic estimator of the test generalisation error.

several times by using a randomised dataset instead of the original one. For instance, a randomised dataset may be obtained by reshuffling the output vector, a permuting operation that artificially removes the dependency between the inputs and the output. After a number of repetitions with randomised datasets, we obtain the null distribution of the accuracy in case of no dependency between inputs and output. If the accuracy associated with the original data is not significantly better than the one obtained with randomised data, we are overfitting the data. For instance, let us consider a large-variate classification task where the cross-validated misclassification error after feature selection is 5%. If we repeat the same learning procedure 100 times with randomised datasets and we obtain a significant (e.g. 10) number of times a misclassification error smaller or equal than 5%, this is a sign of potential overfitting.

Making a robust assessment of a feature selection outcome has a striking importance today because we are more and more confronted with tasks characterised by a very large feature to sample ratio (e.g. in bioinformatics [8]), where a bad assessment procedure can give too optimistic (overfitted) results.

Example

The script `FeatureSel/fselbias.R` illustrates the problem of selection bias in the case of intensive search during feature selection. Consider a linear input/output regression dependency with $n = 54$ inputs (of which only 4 relevant and others irrelevant) and a dataset of size $N = 20$. Let us perform a forward search based on internal leave-one-out. Figure 10.15 shows the evolution of the internal leave-one-out MSE and a more reliable estimation of the generalisation MSE based on an independent test set (5000 i.i.d. examples from the same input/output process). It appears that, as the feature set size increases, the internal leave-one-out error returns a very optimistic estimation of the generalisation error. Therefore, the internal leave-one-out error is unable to detect that the optimal size of the input set (i.e. the number of strongly relevant variables) is equal to four.

10.10 Conclusion

Nowadays, feature selection is an essential component of a real-world learning pipeline. This chapter discussed how the problem is typically addressed as a stochastic optimisation task in a combinatorial state-space where the assessment of each

solution and the search strategy are key elements. Most heuristic approaches rely on a *monotonic* assumption, stating that the best subset of size k is always contained in the best subset of size $> k$. The theorem in Section 10.4.2 and the notions of interaction discussed in Section 10.8.1 show that this assumption is simplistic. Variables that are almost non-informative alone may become extremely informative together since the relevance of a feature is context-based. . Our opinion is that the best way of conceiving feature selection is not by black-box optimisation but by reasoning on the conditional structure of the distribution underlying the data. The final aim should be, as much as possible, to shed light on the context-based role of each feature⁴. In recent years there have been many discussions about the interpretability of data-driven models though it is not always made clear what is the most valuable information for the human user. We deem that in a large variate task the most useful outcome should be an interpretable description of features, returning for each of them a context-based degree of relevance. Accuracy is only a proxy of information: the real information is in the structure.

10.11 Exercises

1. Consider the dataset

\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3	\mathbf{y}
1	1	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	0
1	0	1	0
1	0	0	0
1	1	0	0
0	1	1	0

Rank the input features in a decreasing order of relevance by using the correlation

$$\rho_{\mathbf{xy}} = \frac{\hat{\sigma}_{\mathbf{xy}}}{\hat{\sigma}_{\mathbf{x}} \hat{\sigma}_{\mathbf{y}}}$$

as measure of relevance.

Solution: Since $\rho_{\mathbf{x}_1\mathbf{y}} = 0.488$, $\rho_{\mathbf{x}_2\mathbf{y}} = -0.293$, $\rho_{\mathbf{x}_3\mathbf{y}} = -0.192$, the ranking is $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$.

2. Consider a regression task with two inputs \mathbf{x}_1 , \mathbf{x}_2 and output \mathbf{y} . Suppose we observe the following training set

X_1	X_2	Y
-0.2	0.1	1
0.1	0	0.5
1	-0.3	1.2
0.1	0.2	1
-0.4	0.4	0.5
0.1	0.1	0
1	-1	1.1

1. Fit a multivariate linear model with $\beta_0 = 0$ to the dataset.

⁴The notion of context is essential in human cognition. For instance, according to the philosopher Wittgenstein (1968), the meaning of a word is defined by the circumstances of its use.

2. Compute the mean squared training error.
3. Suppose you use a correlation-based ranking strategy for ranking the features. What would be the top ranked variable?

Hint:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix} \Rightarrow A^{-1} = \frac{1}{a_{11}a_{22} - a_{12}^2} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{12} & a_{11} \end{bmatrix}$$

Solution:

1. $X^T X = \begin{bmatrix} 2.23 & -1.45 \\ -1.45 & 1.31 \end{bmatrix}$
 $(X^T X)^{-1} = \begin{bmatrix} 1.599 & 1.77 \\ 1.77 & 2.72 \end{bmatrix}$
 $X^T Y = \begin{bmatrix} 2.05 \\ -0.96 \end{bmatrix}$
 $\beta = (X^T X)^{-1} X^T Y = \begin{bmatrix} 1.58 \\ 1.016 \end{bmatrix}$

2. $e = Y - X\beta = \begin{bmatrix} 1.21 \\ 0.34 \\ -0.08 \\ 0.64 \\ 0.73 \\ -0.26 \\ 0.54 \end{bmatrix}$ It follows that the Mean Squared training error amounts to 0.41.

3. Since

$$\rho_{X_1 Y} = \frac{\sum_{i=1}^N (X_{i1} - \mu_1)(Y_i - \mu_Y)}{\sqrt{\sum_{i=1}^N (X_{i1} - \mu_1)^2(Y_i - \mu_Y)^2}} = 0.53$$

and

$$\rho_{X_2 Y} = \frac{\sum_{i=1}^N (X_{i2} - \mu_2)(Y_i - \mu_Y)}{\sqrt{\sum_{i=1}^N (X_{i2} - \mu_2)^2(Y_i - \mu_Y)^2}} = -0.48$$

where $\mu_1 = 0.24$, $\mu_2 = -0.07$, $\mu_Y = 0.75$, X_1 is the top ranked variable.

3. The .Rdata file `bonus4.Rdata` in the directory `gbcodes/exercises` of the companion R package contains a regression dataset with $N = 200$ observations, $n = 50$ input features (in the matrix \mathbf{X}) and one target variable (vector \mathbf{Y}).

Knowing that there are 3 strongly relevant variables and 2 weakly relevant variables, the student has to define and implement a strategy to find them.

No existing feature selection code has to be used. However, the student may use libraries to implement supervised learning algorithms.

The student code should

- return the position of the 3 strongly relevant variables and 2 weakly relevant variables,
- discuss what strategy could have been used if the number of strongly and weakly variables was not known in advance.

Solution:

See the file `Exercise4.pdf` in the directory `gbcodes/exercises` of the companion R package (Appendix G).

Chapter 11

Conclusions

We have come to the end, almost. We will take a few words to remind you that machine learning is not perfect and to cover a bit of ethical considerations. Then we will conclude with some take-home messages and final recommendations.

11.1 About ML limitations

From the dawn of the AI discipline, machine learning has been considered a key component of autonomous intelligent agents. In recent years, though a full-fledged artificial intelligence does not seem within reach yet, machine learning found great success thanks to its data-driven and assumption-free nature.

This book insisted on the fact that no modelling effort may be completely assumption-free. Assumptions (though often implicit and hard-coded in the algorithms) are everywhere in the learning process, from problem formulation to data collection, model generation and assessment! When such assumptions happen to match with reality, the resulting method is successful: if it is not the case, the result may be disappointing (see NFL theorem).

Another misjudgment about machine learning is to consider it as a reliable proxy of human learning. Machine learning owes its success to the generic and effective way of transforming a learning problem into a (stochastic) optimisation one. Machines do not think or learn like us: if this is the key to their success, this makes them fragile, too. A large part of human rational decision making and understanding cannot be reduced to the optimisation of a cost function.

This has been put into evidence by a recent trend taking a critical attitude about the machine learning approach (e.g. the i.i.d. assumption) and its limits. For instance, research on *adversarial learning* shows that the limited training and validation set may induce very optimistic expectations about the generalisation of learners. Recent research showed that automatic learners, which appear to be accurate emulators of human knowledge (e.g. in terms of classification accuracy), may be easily fooled once required to work in specific situations. A well-known example (Figure 11.1) shows that deep learning classifiers, able to reach an almost 100% accuracy rate in recognising animal images, may return pitiful predictions, when confronted with properly tweaked inputs [69]. Though this seems anecdotal, such vulnerabilities in learning machines could be very dangerous in safety-critical settings (e.g. self-driving cars).

Another interesting research direction is the study of the robustness of automatically learned model in settings which are not identical to the one used for training, e.g. because of nonstationarity and concept drift. This is particularly critical in health problems where models returning high-quality predictions for a specific co-

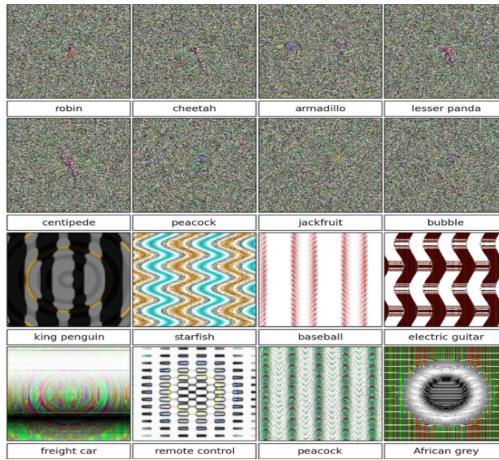


Figure 11.1: Bad generalisation in front of adversarial examples

hort (e.g. in a given hospital) miserably fail when tested on different patients (e.g. from another hospital). How to transfer learned models to close settings is then another hot topic in recent learning research. In this context, causal interpretation of data generation could play an important role in reducing the risk of drift and increasing the model stability.

11.2 A bit of ethics

Last but not least, a word of ethics should not hurt in a book for computer scientists. “Data-driven does not necessarily mean “objective. Machine learning models predict what they have been trained to predict and their forecasts are only as good as the data used for their training. In that sense, machine learning can reinforce human prejudices if trained on biased data sets derived from human decisions. Feeding learners with biased data can have dangerous consequences [133]. In 2016 Twitter chatbot Tay began uttering racist statements after a single interaction day. The predictive justice software COMPAS, deciding whether a suspect should be incarcerated before trial or not, has been accused of being racially biased by an NGO. In 2015, Google Photos identified two African American people as “gorillas”.

Every ML practitioner should be aware that even models developed with the best of intentions may exhibit discriminatory biases, perpetuate inequality, or perform less well for historically disadvantaged groups¹. Recent efforts in modelling and introducing fairness in machine learning (most of the time based on causal considerations) are then more than welcome.

At the same time, the problem goes beyond the scientific and technical realm and involves human responsibility. Automating a task is a responsible decision-making act which implies encoding (implicitly or explicitly) ethical priorities in an autonomous agent. This encoding step is not necessarily easy or comfortable for humans since it implies making ethical alternatives commensurable², e.g. by mapping them to numerical or quantitative cost functions. For instance, a self-driving car that decides to brake (or not to brake) is somewhat trading the cost of human life vs. the cost of an over-conservative action. The choice of entrusting to machines tasks that could have an impact on human security or human sensibility should never exempt humans (from the programmer to the decision maker) from

¹see also the presentation <https://tinyurl.com/y4ld3ohz>

²[https://en.wikipedia.org/wiki/Commensurability_\(ethics\)](https://en.wikipedia.org/wiki/Commensurability_(ethics))

legal and moral responsibilities of probable errors.

To conclude, the ethical dilemma of ML may be summarized by the contraposition of the two citations at the beginning of this book: on the one hand, any machine learning endeavour harbours the ambition (or the illusion) of catching the essence of reality with numbers or quantitative tools. On the other hand, “not everything that counts” (notably ethics) can be counted or easily translated into numerical terms.

11.3 Take-home notions

Quoting Einstein “the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience”. This sentence catches probably the primary take-home concept in machine learning: trade-off, notably the trade-off between bias and variance, underfitting and overfitting, parametric and non-parametric, false positive and false negative, type I and type II error,...(please add).

Other bulk notions the author would like you to remember (or revise) are:

- information theory: it is a powerful language to talk about stochastic dependence,
- estimators: do not forget they are random variables with their own sampling distribution, and, even if very (very) good, they may be wrong sometimes³,
- conditional probability: supervised learning is all about estimating it,
- conditional (in)dependence and its non-monotonicity: mastering the complexity (and beauty) of high dimensionality goes that way.

11.4 Recommendations

We would like then to end this manuscript not by selling you 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 modelling and data analysis:

- However complex is your learning algorithm (either adaptive or deep or preferred by GAFA), do not forget it is an estimator, and as such, it makes assumptions (often implicitly). Each approach has its own assumptions! Be aware of them before using one.
- Simpler things first! According to Wasserman [174], *using fancy tools like neural nets,...without understanding basic statistics is like doing brain surgery before knowing how to use a band-aid.*
- Reality is probably almost always nonlinear but a massive amount of (theoretical and algorithmic) results exists only for linear methods.
- Expert knowledge MATTERS... But data too :-)
- It is better to be confident with a number of alternative techniques (preferably linear and nonlinear) and use them in parallel on the same task.
- Resampling and combining are at the forefront of the data analysis techniques. Do not forget to test them when you have a data analysis problem.

³...even the divine Roberto Baggio missed a penalty in the 1994 FIFA World Cup Final against Brazil :-(

- Do not be religious about learning/modelling techniques. The best learning algorithm does NOT exist.
- Statistical dependency does not imply causality, though it may shed some light on it.

and the best motto for a machine learner:

Once you stop learning, you start dying (Albert Einstein).

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 $\{z_1, z_2, \dots, z_N\}$ 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 [150, 136].

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

$$P_R = \text{prob}\{z \in R\} = \int_R p_{\mathbf{z}}(z) dz \quad (\text{A.1.1})$$

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 (D.1.1) we have that its probability distribution is

$$p_{\mathbf{k}}(k) = \frac{N!}{k!(N-k)!} P_R^k (1 - P_R)^{(N-k)} \quad (\text{A.1.2})$$

Moreover, the random variable \mathbf{k}/N satisfies

$$E[\mathbf{k}/N] = P_R \quad (\text{A.1.3})$$

and

$$\text{Var}[\mathbf{k}/N] = E[(\mathbf{k}/N - P_R)^2] = \frac{P_R(1 - P_R)}{N} \quad (\text{A.1.4})$$

Since according to (A.1.4), the variance of \mathbf{k}/N converges to zero as $N \rightarrow \infty$, it is reasonable to expect that the fraction k/N return a good estimate of the probability P_R

$$P_R \cong \frac{k}{N} \quad (\text{A.1.5})$$

At the same time, if we assume that $p_{\mathbf{z}}(z)$ is continuous and does not vary appreciably over R , we can approximate P_R with:

$$P_R = \int_R p(z) dz \cong p(z)V \quad (\text{A.1.6})$$

with V volume of R . From (A.1.5) and (A.1.6) it follows that for values of z inside R

$$p(z) \cong \frac{k}{NV} \quad (\text{A.1.7})$$

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

$$K(u) = \begin{cases} 1 & \text{if } |u_j| < 1/2 \\ 0 & \text{else} \end{cases} \quad j = 1, \dots, n \quad (\text{A.1.8})$$

where u_j is the j^{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

$$k = \sum_{i=1}^N K\left(\frac{z - z_i}{B}\right) \quad (\text{A.1.9})$$

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

$$\hat{p}(z) = \frac{\sum_{i=1}^N K\left(\frac{z - z_i}{B}\right)}{NB^n} \quad (\text{A.1.10})$$

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 [25],
2. it requires the memorisation of the whole set of observations. 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

$$\hat{p}_z(z) = \frac{k}{NV} \quad (\text{A.1.11})$$

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

$$p_z(z) = \sum_{j=1}^m p(z|j)\pi(j) \quad (\text{A.1.12})$$

where m is a parameter of the model and typically $m \ll N$. The coefficients $\pi(j)$ are called *mixing coefficients* and satisfy the following constraints

$$\sum_{j=1}^m \pi(j) = 1 \quad 0 \leq \pi(j) \leq 1 \quad (\text{A.1.13})$$

The quantity $\pi(j)$ is typically interpreted as the *prior* probability that a data point be generated by the j^{th} component of the mixture. According to Bayes' theorem, the corresponding *posterior* probabilities is

$$p(j|z) = \frac{p(z|j)\pi(j)}{p(z)} = \frac{p(z|j)\pi(j)}{\sum_{j=1}^m p(z|j)\pi(j)} \quad (\text{A.1.14})$$

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 μ_j and σ_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 optimisation 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–maximisation* or EM algorithm [58] is a simple and practical method for estimating the mixture parameters avoiding complex nonlinear optimisation 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 maximisation (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 θ . Assume also that further data, denoted by Ξ , exist but are not observable. The quantity Δ_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 $\mathbf{l}_{\text{comp}}(\theta)$ the log likelihood of the parameter θ given the complete data. This is a random variable because the values of Ξ are not known. Hence, it is possible for a given value $\theta^{(\tau)}$ of the parameter vector to compute the expected value of $\mathbf{l}_{\text{comp}}(\theta^{(\tau)})$. 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 maximisation step is expected to find the parameter value $\theta^{(\tau+1)}$ which maximizes Q . The EM procedure in detail is the following:

1. Make an initial estimate $\theta^{(0)}$ of the parameter vector.
2. The log likelihood $\mathbf{l}_{\text{comp}}(\theta^{(\tau)})$ of the parameters $\theta^{(\tau)}$ with respect to the complete data Δ_N is calculated. This is a random function of the unknown dataset Ξ .
3. The E-step: the expectation $Q(\theta^{(\tau)})$ of $\mathbf{l}_{\text{comp}}(\theta^{(\tau)})$ is calculated.
4. The M-step: a new estimate of the parameters is found by the maximisation

$$\theta^{(\tau+1)} = \arg \max_{\theta} Q(\theta) \quad (\text{A.1.15})$$

The theoretical justification comes from the following result proved in [58]: for a sequence $\theta^{(\tau)}$ generated by the EM algorithm it is always true that for the incomplete likelihood

$$l(\theta^{(\tau+1)}) \geq l(\theta^{(\tau)}) \quad (\text{A.1.16})$$

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 ζ_{ij} , $j = 1, \dots, m$, such that

$$\zeta_{ij} = \begin{cases} 1 & \text{if } z_i \text{ is generated by the } j^{\text{th}} \text{ basis} \\ 0 & \text{otherwise} \end{cases} \quad (\text{A.1.17})$$

Let Δ_N be the extension of the dataset D_N , i.e. it represents the complete dataset, including the unobservable ζ_{ij} . The probability distribution for each (z_i, ζ_{ij}) is either zero or $p(z_i|j)$. If we let ζ_i represent the set $\{\zeta_{i1}, \zeta_{i2}, \dots, \zeta_{im}\}$ then

$$p_{\zeta_i}(\zeta_i) = \pi(j') \text{ where } j' \text{ is such that } \zeta_{ij'} = 1 \quad (\text{A.1.18})$$

so

$$p(z_i, \zeta_i) = p(\zeta_i)p(z_i|j') = \pi(j')p(z_i|j') = \prod_{j=1}^m [\pi(j)p(z_i|j)]^{\zeta_{ij}} \quad (\text{A.1.19})$$

Thus the complete log likelihood is given by

$$\mathbf{l}_{comp}(\theta) = \ln \mathbf{L}_{comp}(\theta) = \ln \prod_{i=1}^N \prod_{j=1}^m [\pi(j)p(z_i|j)]^{\zeta_{ij}} \quad (\text{A.1.20})$$

$$= \sum_{i=1}^N \ln \prod_{j=1}^m [\pi(j)p(z_i|j)]^{\zeta_{ij}} \quad (\text{A.1.21})$$

$$= \sum_{i=1}^N \sum_{j=1}^m \zeta_{ij} \{\ln \pi(j) + \ln p(z_i|j)\} \quad (\text{A.1.22})$$

where the vector θ includes the mixing coefficients and the parameters of the density $p(z|j)$ in Eq. (A.1.12).

Introducing the terms ζ_{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 ζ_{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:

$$Q(\theta^{(\tau)}) = E[\mathbf{l}_{comp}(\theta^{(\tau)})] = E\left[\sum_{i=1}^N \sum_{j=1}^m \zeta_{ij} \{\ln \pi(j) + \ln p(z_i|j)\}\right] \quad (\text{A.1.23})$$

$$= \sum_{i=1}^N \sum_{j=1}^m E[\zeta_{ij}] \{\ln \pi(j) + \ln p(z_i|j)\} \quad (\text{A.1.24})$$

Since

$$E[\zeta_{ij}] = P(\zeta_{ij} = 1|z_i) = \frac{p(z_i|\zeta_{ij})P(\zeta_{ij})}{p(z_i)} = \frac{p(z_i|j)\pi(j)}{p(z_i)} = p(j|z_i) \quad (\text{A.1.25})$$

from Eq. (A.1.14) and (A.1.18) we have

$$Q(\theta^{(\tau)}) = \sum_{i=1}^N \sum_{j=1}^m p(j|z_i) \{\ln \pi(j) + \ln p(z_i|j)\} \quad (\text{A.1.26})$$

The M-step maximizes Q with respect to the whole set of parameters θ but it is known that this can be done individually for each parameter, if we consider a Gaussian mixture model

$$p(z|j) = \frac{1}{(2\pi\sigma_j^2)^{n/2}} \exp\left\{-\frac{(z - \mu_j)^2}{2\sigma_j^2}\right\} \quad (\text{A.1.27})$$

In this case we have:

$$Q(\theta^{(\tau)}) = \sum_{i=1}^N \sum_{j=1}^m p(j|z_i) \{\ln \pi(j) + \ln p(z_i|j)\} \quad (\text{A.1.28})$$

$$= \sum_{i=1}^N \sum_{j=1}^m p(j|z_i) \left\{ \ln \pi(j) - n \ln \sigma_j^{(\tau)} - \frac{(z_i - \mu_j^{(\tau)})^2}{2(\sigma_j^{(\tau)})^2} \right\} + \text{constant} \quad (\text{A.1.29})$$

We can now perform the maximisation (A.1.15). For the parameters μ_j and σ_j the maximisation is straightforward:

$$\mu_j^{(\tau+1)} = \frac{\sum_{i=1}^N p(j|z_i)z_i}{\sum_{i=1}^N p(j|z_i)} \quad (\text{A.1.30})$$

$$\left(\sigma_j^{(\tau+1)}\right)^2 = \frac{1}{n} \frac{\sum_{i=1}^N p(j|z_i)(z_i - \mu_j^{(\tau+1)})^2}{\sum_{i=1}^N p(j|z_i)} \quad (\text{A.1.31})$$

For the mixing parameters the procedure is more complex [25] and returns:

$$\pi(j)^{(\tau+1)} = \frac{1}{N} \sum_{i=1}^N p(j|z_i) \quad (\text{A.1.32})$$

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 $x_i, i = 1, \dots, N$, into K groups $G_k, k = 1, \dots, K$, 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^{th} group and the corresponding cluster center c_k , the cost function can be defined by

$$J = \sum_{k=1}^K J_k = \sum_{k=1}^K \sum_{x \in G_k} d(x, c_k) \quad (\text{A.2.33})$$

where J_k is the cost function within group k and d is a generic distance function

$$d(x, c_k) = (x - c_k)^T M (x - c_k) \quad (\text{A.2.34})$$

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^{th} data point x_i belongs to group k , and 0 otherwise. The matrix U satisfies the following conditions:

$$\begin{aligned} \sum_{k=1}^K u_{ki} &= 1 \quad \forall i = 1, \dots, N \\ \sum_{k=1}^K \sum_{i=1}^N u_{ki} &= N \end{aligned} \quad (\text{A.2.35})$$

Once the cluster centers c_k are fixed, the terms u_{ki} which minimize Eq. (A.2.33) are:

$$u_{ki} = \begin{cases} 1 & \text{if } d(x_i, c_k) \leq d(x_i, c_j), \text{ for each } j \neq k \\ 0 & \text{otherwise} \end{cases} \quad (\text{A.2.36})$$

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:

$$c_k = \frac{1}{|G_k|} \sum_{x \in G_k} x \quad (\text{A.2.37})$$

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.

The script `Appendix/kmeans.R` implements and visualises the outcome of the algorithm in a simple bidimensional task.

Appendix B

Linear algebra notions

Linear algebra, the science of vector spaces, plays a major role in machine learning, where data are represented in a vectorial form. Though the readers are supposed to have attended numerical analysis classes, here we will remind some basic notions of linear algebra. For a more extensive presentation of linear algebra and its links with machine learning, we refer the reader to recent references like [2, 57].

B.1 Rank of a matrix

Let us consider a $[N, n]$ matrix. Many definitions exist for the rank of a matrix: here we limit to consider the rank of X as the maximal number of linearly independent columns of X . Since the rank of a $[N, n]$ matrix is at most $\min\{N, n\}$, a matrix is *full-rank* if its rank is $\min\{N, n\}$. A matrix which is not full-rank is also called *rank-deficient*.

B.2 Inner product

In linear algebra, the *dot product*, also known as the *scalar or inner product*, is an operation which takes two vectors over the real numbers \mathbb{R} and returns a real-valued scalar quantity. It is the standard inner product of the orthonormal Euclidean space. The dot product of two $[n, 1]$ vectors $x = [x_1, x_2, \dots, x_n]^T$ and $y = [y_1, y_2, \dots, y_n]^T$ is defined as:

$$\langle x, y \rangle = \sum_{j=1}^n x_j y_j = x^T y \quad (\text{B.2.1})$$

The dot product underlies the definition of the following quantities:

- the *Euclidean norm* of a vector x :

$$\|x\| = \sqrt{\langle x, x \rangle}, \quad (\text{B.2.2})$$

also known as the L_2 norm,

- the Euclidean distance of two $[n, 1]$ vectors x_1 and x_2

$$\|x_1 - x_2\| = \sqrt{\langle x_1 - x_2, x_1 - x_2 \rangle}, \quad (\text{B.2.3})$$

- the angle ω between two vectors x_1 and x_2 which satisfies the relation

$$-1 \leq \frac{\langle x_1 x_2 \rangle}{\|x_1\| \|x_2\|} = \cos(\omega) \leq 1, \quad (\text{B.2.4})$$

- the projection of a vector x_1 onto a direction x_2

$$\pi_{x_2}(x_1) = \frac{\langle x_1, x_2 \rangle}{\|x_2\|^2} x_2 = \frac{x_2 x_2^T}{\|x_2\|^2} x_1 \quad (\text{B.2.5})$$

where the $[n, n]$ matrix $P = x_2 x_2^T$ is called the projection matrix.

In more qualitative terms, the notion of inner product allows the introduction of a *similarity score* between vectors. In this sense, the least similar vectors are two orthogonal vectors, i.e. two vectors x and y such that $\langle x, y \rangle = 0$ and $\omega = \pi/2$.

Note also that the following relation holds:

$$xx^T y = \langle x, y \rangle x$$

B.3 Gramian matrix

A real matrix A such that $A = X^T X$ where X is $[N, n]$ is called a *Gramian matrix* (also called *sum of squares* or *cross products* matrix) [81]. Its (i, j) th element is the inner product of $X[, i]$ and $X[, j]$. Also

$$X^T X = \sum_{i=1}^N X[, i]^T X[, i]$$

where $X[, i]$ is the i th row of X . A Gramian matrix is symmetric, non negative definite and of full rank if X is of full rank. The non-zero eigenvalues of XX^T are the same as the non zero eigenvalues of $X^T X$ which are also the squared singular values of X .

If A is a non negative definite matrix then also

$$X^T X + A = \begin{bmatrix} X \\ A^{1/2} \end{bmatrix}^T \begin{bmatrix} X \\ A^{1/2} \end{bmatrix}$$

is a Gramian matrix.

B.4 Diagonalisation

A $[N, N]$ matrix X is diagonalisable if it exists an invertible matrix P such that

$$X = PDP^{-1}. \quad (\text{B.4.6})$$

A symmetric matrix can always be orthogonally diagonalised ($PP^T = P^T P = I$) and can be written (spectral decomposition) as

$$X = \sum_{i=1}^N d_i p_i p_i^T$$

where the diagonal entries d_i are the eigenvalues of X and p_i are the corresponding eigenvectors. Also, any matrix which is orthogonally diagonalisable is symmetric [81].

B.5 QR decomposition

Let us consider a $[N, n]$ matrix X with $N \geq n$ and n linearly independent columns. By *Gram-Schmidt orthogonalisation* [2] it is possible to write

$$X = QR \quad (\text{B.5.7})$$

where Q is a $[N, n]$ matrix with n orthonormal columns q_j (i.e. $q_j^T q_j = 1$ and $q_j^T q_k = 0$ if $j \neq k$) and R is a $[n, n]$ upper-triangular matrix. Since $Q^T Q = I_n$ the pseudo-inverse of X can be written as

$$X^\dagger = (X^T X)^{-1} X^T = (R^T Q^T Q R)^{-1} R^T Q^T = R^{-1} Q^T \quad (\text{B.5.8})$$

If X is rank-deficient (i.e. only $n' < n < N$ columns of X are linearly independent) it is possible to perform the *generalised QR decomposition*

$$X = QR$$

where Q is $[N, n']$ and R is a $[n', n]$ rectangular upper-triangular matrix with $n' < n$. Since R is of full row rank, the matrix RR^T is invertible and the pseudo-inverse of X can be written as

$$X^\dagger = R^T (RR^T)^{-1} Q^T \quad (\text{B.5.9})$$

also known as the *Moore-Penrose* pseudo-inverse.

B.6 Singular Value Decomposition

Let us consider a $[N, n]$ matrix X with $N \geq n$: such matrix can always be factorised into the product of three matrices

$$X = UDV^T \quad (\text{B.6.10})$$

where U is a $[N, N]$ matrix with orthonormal columns (i.e. $U^T U = I_N$), D is a $[N, n]$ diagonal matrix whose diagonal entries $d_{ii} \geq 0$ are called the *singular values* and V is a $[n, n]$ matrix with orthonormal columns.

It can be shown that the N columns of U (also called the left singular vectors) are the N eigenvectors of the $[N, N]$ symmetric matrix XX^T and the n columns of V (also called the right singular vectors) are the n eigenvectors of the $[n, n]$ symmetric matrix $X^T X$. The non-zero singular values are the square-roots of the non-zero eigenvalues of $X^T X$ and of the non-zero eigenvalues of XX^T . This is made evident by the link between SVD and diagonalisation of $X^T X$:

$$X^T X = (UDV^T)^T (UDV^T) = VD^T U^T UDV^T = VD^T DV^T$$

The SVD of a matrix X of rank r can be written also as:

$$X = \sum_{j=1}^r d_{ii} u_i v_i^T$$

where u_j is the j th column of U and v_j is the j th column of V .

If in the decomposition above we stop at the order $r' < r$, we obtain a low-rank approximation of X :

$$X' = \sum_{j=1}^{r'} d_{ii} u_i v_i^T.$$

Another common SVD decomposition is the *economy* (or *reduced*) SVD:

$$X = UDV^T \quad (\text{B.6.11})$$

where $k = \min\{N, n\}$, U is $[N, k]$ with orthonormal columns, D is a square $[k, k]$ matrix and V is a $[n, k]$ matrix with orthonormal columns.

SVD plays an important role in determining the ill-conditioning of a square matrix, i.e. how much the matrix is close to be singular. The condition number of a matrix is the ratio of its largest singular value to its smallest singular value. The larger this number (which is ≥ 1) the larger is the ill-conditioning of the matrix.

Note also that if X is a symmetric matrix, the SVD decomposition returns the diagonalisation (B.4.6).

B.7 Chain rules of differential calculus

Let J be the scalar function of $\alpha \in \mathbb{R}$:

$$J(\alpha) = f(g(h(\alpha)))$$

where $f, g, h : \mathbb{R} \rightarrow \mathbb{R}$ are scalar functions. Then the univariate chain rule is

$$\frac{dJ}{d\alpha} = \frac{dJ}{df} \frac{df}{dg} \frac{dg}{dh} \frac{dh}{d\alpha}$$

Let us consider the function $J : \mathbb{R} \rightarrow \mathbb{R}$

$$J = f(g_1(\alpha), g_2(\alpha), \dots, g_n(\alpha))$$

between $\alpha \in \mathbb{R}$ and the scalar J , where $g_j : \mathbb{R} \rightarrow \mathbb{R}, j = 1, \dots, n$. Then the multivariate chain rule returns the scalar gradient

$$\frac{dJ}{d\alpha} = \sum_{j=1}^n \frac{\partial J}{\partial g_j} \left(\frac{dg_j}{d\alpha} \right)$$

Let $J : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be the mapping between an input vector $\alpha \in \mathbb{R}^n$ and an output vector of size m . The associated Jacobian matrix is the $[m, n]$ matrix

$$\nabla_\alpha J = \begin{bmatrix} \frac{\partial J(\alpha)}{\partial \alpha_1} & \frac{\partial J(\alpha)}{\partial \alpha_2} & \dots & \frac{\partial J(\alpha)}{\partial \alpha_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial J_m(\alpha)}{\partial \alpha_1} & \frac{\partial J_m(\alpha)}{\partial \alpha_2} & \dots & \frac{\partial J_m(\alpha)}{\partial \alpha_n} \end{bmatrix}$$

In the most generic case, suppose that $J = \mathbb{R}^n \rightarrow \mathbb{R}^m$, $\alpha \in \mathbb{R}^n$, and

$$J = F_k(F_{k-1}(\dots F_1(\alpha)))$$

where $F_i : \mathbb{R}^{n_i} \rightarrow \mathbb{R}^{n_{i+1}}$, $n_1 = n$ and $n_{k+1} = m$. Then the *vectored chain rule* [2] is

$$\underbrace{\frac{\partial J}{\partial \alpha}}_{[m,n]} = \underbrace{\frac{\partial F_k}{\partial F_{k-1}}}_{[m,n_k]} \underbrace{\frac{\partial F_{k-1}}{\partial F_{k-2}}}_{[n_k, n_{k-1}]} \cdots \underbrace{\frac{\partial F_1}{\partial \alpha}}_{[n_2, n]}$$

B.8 Quadratic norm

Consider the quadratic norm

$$J(x) = \|Ax + b\|^2$$

where $J : \mathbb{R}^n \rightarrow \mathbb{R}$, A is a $[N, n]$ matrix, x is a $[n, 1]$ vector and b is a $[N, 1]$ vector. It can be written in the matrix form

$$J(x) = x^T A^T Ax + 2b^T Ax + b^T b$$

The first derivative of J with respect to x is the $[n, 1]$ vector

$$\frac{\partial J(x)}{\partial x} = 2A^T(Ax + b)$$

and the second derivative is the $[n, n]$ matrix

$$\frac{\partial^2 J(x)}{\partial x \partial x^T} = 2A^T A$$

B.9 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

$$(F + GHK)^{-1} = F^{-1} - F^{-1}G(H^{-1} + KF^{-1}G)^{-1}KF^{-1} \quad (\text{B.9.12})$$

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.

If X and Z are two $[N, p]$ matrices, from (B.9.12) it can be shown the *push-through identity* [2]

$$X^T(I_N + ZX^T)^{-1} = (I_p + X^TZ)^{-1}X^T \quad (\text{B.9.13})$$

Then for any $[N, p]$ matrix X and scalar $\lambda > 0$

$$X^T(\lambda I_N + XX^T)^{-1} = (\lambda I_p + X^TX)^{-1}X^T \quad (\text{B.9.14})$$

Appendix C

Optimisation notions

Optimisation is the workhorse of machine learning algorithms: all machine learning techniques rely on the optimisation of some cost function, as made explicit by the notions of maximum likelihood, least-squares or more in general parametric identification. In this appendix we will review some basic notions of optimisation but the reader should be aware that optimisation is such a wide research domain that, for the sake of completeness, she should refer to texts like [82, 37, 2].

An optimisation problem is characterised by an *objective function* (also called *cost* or *loss function*) $J(\mathbf{x})$ of a set $\mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^T$ of n *optimisation variables*. The goal is to compute the values of the optimisation variables

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} J(\mathbf{x}) \quad (\text{C.0.1})$$

at which the cost function $J : \mathbb{R}^n \rightarrow \mathbb{R}$ is minimised (or its opposite maximised) subject to m constraints

$$J_i(\mathbf{x}) \leq c_i, \quad i = 1, \dots, m \quad (\text{C.0.2})$$

where $J_i(\cdot)$ are the *constraint functions*.

If $m = 0$, i.e. no constraints are imposed on the set of feasible values of the solution \mathbf{x}^* , the optimisation problem is called *unconstrained*.

If both the objective and the constraint functions are linear where a linear function satisfies the equality

$$J(w_1 \mathbf{x}_1 + w_2 \mathbf{x}_2) = w_1 J(\mathbf{x}_1) + w_2 J(\mathbf{x}_2), \quad w_1, w_2 \in \mathbb{R}$$

the optimisation problem is linear otherwise it is non-linear.

If both the objective and the constraint functions are convex where a convex function satisfies the inequality

$$J(w_1 \mathbf{x}_1 + w_2 \mathbf{x}_2) \leq w_1 J(\mathbf{x}_1) + w_2 J(\mathbf{x}_2), \quad w_1 + w_2 = 1, w_1 \geq 0, w_2 \geq 0 \quad (\text{C.0.3})$$

the problem is called *convex*. Form the definition above it appears that a linear optimisation problem is an instance of convex optimisation.

Special cases of convex optimisation are least-squares problems and linear programming

C.1 Least-squares

A *least-squares optimisation problem* is an unconstrained optimisation problem where the cost function is

$$J(\mathbf{x}) = \|A\mathbf{x} - b\|_2^2$$

b is $[N, 1]$ vector and A is a $[N, n]$ matrix. The solution (see Section 7.1.11) is

$$\mathbf{x}^* = (A^T A)^{-1} A^T b$$

C.2 Linear programming

A linear programming optimisation problem is a problem where both the objective and the constraint functions are linear

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} c^T \mathbf{x}$$

subject to

$$a_i^T \mathbf{x} \leq b_i, \quad i = 1, \dots, m$$

where a_i are $[n, 1]$ vectors and $b_i \in \mathbb{R}$.

Unlike the least-squares problem, there is no simple analytical formula to compute the solution but there are very effective methods, like the simplex or the interior-point method.

C.3 Quadratic programming

Quadratic programming is the resolution procedure of continuous optimisation problems with a squared objective function, for instance

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} J(\mathbf{x}) = \arg \min_{\mathbf{x}} \mathbf{x}^T D \mathbf{x}$$

where \mathbf{x} is a $[n, 1]$ vector and D is a $[n, n]$ matrix. For instance if $n = 2$ and A is a diagonal matrix, $J(\mathbf{x}) = x_1^2 + x_2^2$ and has a single global minimum in $[0, 0]$. If the solution is submitted to no constraints, the problem is called *unconstrained*. If D is a positive (negative) semidefinite matrix the function J is convex (concave).

In machine learning the most common quadratic programming task is strictly convex (definition (C.0.3)) since it derives from the least-squares formulation (Section (C.1)) where D is a definite positive matrix. The general form of an unconstrained strictly convex quadratic objective function is

$$\mathbf{x}^* = J(\mathbf{x}) = \arg \min_{\mathbf{x}} \mathbf{x}^T D \mathbf{x} - d^T \mathbf{x} + k = \arg \min_{\mathbf{x}} \mathbf{x}^T D \mathbf{x} - d^T \mathbf{x} \quad (\text{C.3.4})$$

where D is definite positive, d is a $[n, 1]$ vector and k is a scalar (which has no impact on the minimisation problem).

The constrained version has a set of linear inequality constraints in the form

$$A^T b \geq \mathbf{x}_0$$

where A is a $[n, c]$ matrix defining the c constraints under which we want to minimise the J function.

The R package `quadprog` provides the implementation `solve.QP` of a method to solve a strictly convex constrained quadratic programming task.

Appendix D

Probability and statistical notions

D.1 Common univariate discrete probability functions

D.1.1 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

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

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

D.1.2 The Binomial probability function

A *binomial random variable* represents the number of successes z in a *fixed number* N of *independent Bernoulli trials* with the *same probability* p 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

$$\text{Prob}\{\mathbf{z} = z\} = P_{\mathbf{z}}(z) = \binom{N}{z} p^z (1 - p)^{N-z}, \quad z = 0, 1, \dots, N \quad (\text{D.1.1})$$

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 $\mathbf{z}_2 \sim \text{Bin}(N_2, p)$ are independent then $\mathbf{z}_1 + \mathbf{z}_2 \sim \text{Bin}(N_1 + N_2, p)$

The Binomial distribution returns then the probability of z successes out of N draws *with replacement*. The probability of z successes out of N draws *without replacement* from a population of size P that contains k terms associated to success is returned by the *hypergeometric distribution*:

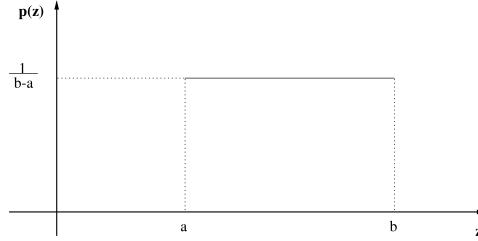
$$\text{Prob}\{\mathbf{z} = z\} = \frac{\binom{k}{z} \binom{P-k}{N-z}}{\binom{P}{N}}. \quad (\text{D.1.2})$$

D.2 Common univariate continuous distributions

D.2.1 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

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



It can be shown that the skewness of a continuous random variable which is uniformly distributed is equal to 0.

Exercise

Show that the variance of $\mathcal{U}(a, b)$ is equal to $\frac{1}{12}(b - a)^2$.

•

D.2.2 The Beta distribution

It is a continuous distribution on the interval $[0, 1]$ defined by two positive parameters α and β that control the shape of the distribution. Note that

- the expectation is $E[\mathbf{z}] = \frac{\alpha}{\alpha+\beta}$ and its mode is $\frac{\alpha-1}{\alpha+\beta-2}$
- the variance is $\text{Var}[\mathbf{z}] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$
- if $\alpha = \beta = 1$ the distribution corresponds to $\mathcal{U}(0, 1)$

D.2.3 The chi-squared distribution

It describes the distribution of squared normal r.v.s. An r.v. \mathbf{z} has a χ_N^2 distribution if

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

where $N \in \mathbb{N}$ and $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ 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 χ_N^2 density and distribution function for $N = 10$ are plotted in Figure D.1 (R script [Appendix/chisq.R](#) in the package `gbcodex`).

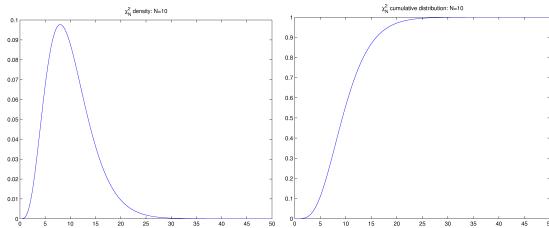
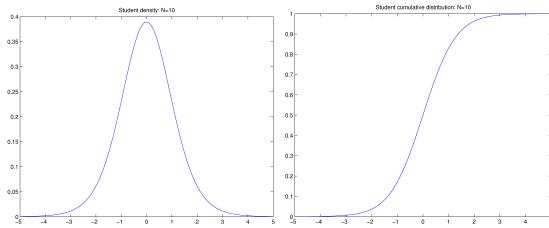
D.2.4 Student's t -distribution

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

$$\mathbf{z} = \frac{\mathbf{x}}{\sqrt{\mathbf{y}/N}} \tag{D.2.3}$$

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

The Student density and distribution function for $N = 10$ are plotted in Figure D.2 by means of the script [Appendix/stu.R](#) in the package `gbcodex`.

Figure D.1: χ_N^2 probability distribution ($N = 10$)Figure D.2: Student probability distribution ($N = 10$)

D.2.5 F-distribution

It describes the distribution of the ratio of χ squared r.v.s. Let $\mathbf{x} \sim \chi_M^2$ and $\mathbf{y} \sim \chi_N^2$ 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

$$\mathbf{z} = \frac{\mathbf{x}/M}{\mathbf{y}/N} \quad (\text{D.2.4})$$

Note that if $\mathbf{z} \sim F_{M,N}$ then $1/\mathbf{z} \sim F_{N,M}$, while if $\mathbf{z} \sim T_N$ then $\mathbf{z}^2 \sim F_{1,N}$. The F-density and distribution function are plotted in Figure D.3 by means of the script `Appendix/f.R` in the package `gbcode`.

D.3 Exponential family of distributions

A r.v. \mathbf{z} belongs to the exponential family of distributions if its density (or probability mass) can be written as

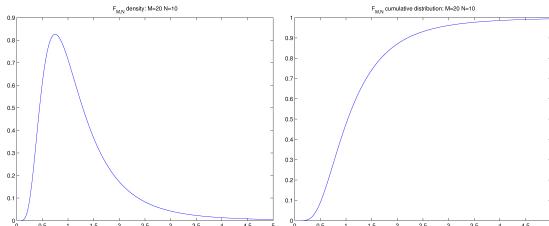
$$p_{\mathbf{z}}(z, \theta) = \exp \left[\frac{z\theta - b(\theta)}{a(\phi)} + c(z, \phi) \right] \quad (\text{D.3.5})$$

where θ is the *canonical parameter* and ϕ is the *scale parameter*. It can be shown that

$$E[\mathbf{z}] = b'(\theta)$$

and

$$\text{Var } [\mathbf{z}] = b''(\theta)a(\phi)$$

Figure D.3: F probability distribution ($N = 10$)

Note that if $\theta = \mu$, $\phi = \sigma^2$, $a(\phi) = \phi$, $b(\theta) = \theta^2/2$, $c(z, \phi) = -\frac{1}{2} \left[\frac{z^2}{\phi} + \log(2\pi\phi) \right]$, $p_{\mathbf{z}}(z)$ coincides with the Normal density (2.12.56). Other distributions that belongs to the exponential family are the Poisson, binomial, Beta and Gamma distributions.

D.4 Common statistical hypothesis tests

D.4.1 χ^2 -test: single sample and two-sided

Consider a random sample from $\mathcal{N}(\mu, \sigma^2)$ with μ known. Let

$$H : \sigma^2 = \sigma_0^2; \quad \bar{H} : \sigma^2 \neq \sigma_0^2$$

Let $\widehat{SS} = \sum_i (z_i - \mu)^2$. From Section 3.7 it follows that if H is true then $\widehat{SS}/\sigma_0^2 \sim \chi_N^2$ (Section D.2.3)

The level α χ^2 -test rejects H if $\widehat{SS}/\sigma_0^2 < a_1$ or $\widehat{SS}/\sigma_0^2 > a_2$ where

$$\text{Prob} \left\{ \frac{\widehat{SS}}{\sigma_0^2} < a_1 \right\} + \text{Prob} \left\{ \frac{\widehat{SS}}{\sigma_0^2} > a_2 \right\} = \alpha$$

A slight modification is necessary if μ is unknown. In this case you must replace μ with $\hat{\mu}$ in the quantity \widehat{SS} and use a χ_{N-1}^2 distribution.

D.4.2 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

$$t(\mathbf{D}_N) = \frac{\hat{\mu}_x - \hat{\mu}_y}{\sqrt{\left(\frac{1}{M} + \frac{1}{N}\right) \left(\frac{\widehat{SS}_x + \widehat{SS}_y}{M+N-2}\right)}} \sim T_{M+N-2}$$

It follows that the test of size α rejects H if

$$|t(D_N)| > t_{\alpha/2, M+N-2}$$

D.4.3 F-test: two samples, two sided

Consider a random sample $\{x_1, \dots, x_M\} \leftarrow \mathbf{x} \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and a random sample $\{y_1, \dots, y_N\} \leftarrow \mathbf{y} \sim \mathcal{N}(\mu_2, \sigma_2^2)$ with μ_1 and μ_2 unknown. Suppose we want to test

$$H : \sigma_1^2 = \sigma_2^2; \quad \bar{H} : \sigma_1^2 \neq \sigma_2^2$$

Let us consider the statistic

$$\mathbf{f} = \frac{\hat{\sigma}_1^2}{\hat{\sigma}_2^2} = \frac{\widehat{SS}_1/(M-1)}{\widehat{SS}_2/(N-1)} \sim \frac{\sigma_1^2 \chi_{M-1}^2 / (M-1)}{\sigma_2^2 \chi_{N-1}^2 / (N-1)} = \frac{\sigma_1^2}{\sigma_2^2} F_{M-1, N-1}$$

It can be shown that if H is true, the ratio \mathbf{f} has a F-distribution $F_{M-1, N-1}$ (Section D.2.5) The F-test rejects H if the ratio f is large, i.e. $f > F_{\alpha, M-1, N-1}$ where

$$\text{Prob} \{ \mathbf{f} > F_{\alpha, M-1, N-1} \} = \alpha$$

if $\mathbf{f} \sim F_{M-1, N-1}$.

D.5 Transformation of random variables and vectors

Theorem 5.1 (Jensen's inequality). *Let \mathbf{x} be a continuous r.v. and f a convex function. Then, $E[f(\mathbf{x})] \geq f(E[\mathbf{x}])$ while if f is concave then $E[f(\mathbf{x})] \leq f(E[\mathbf{x}])$*

Given a $[n \times 1]$ constant vector a and a random vector \mathbf{z} of dimension $[n \times 1]$ with expected value $E[\mathbf{z}] = \mu$ and covariance matrix $\text{Var}[\mathbf{z}] = \Sigma$, then

$$E[a^T \mathbf{z}] = a^T \mu, \quad \text{Var}[a^T \mathbf{z}] = a^T \Sigma a$$

Also if $\mathbf{z} \sim \mathcal{N}(\mu, \Sigma)$ then $a^T \mathbf{z} \sim \mathcal{N}(a^T \mu, a^T \Sigma a)$

Given a $[n \times n]$ constant matrix A and a random vector \mathbf{z} of dimension $[n \times 1]$ with expected value $E[\mathbf{z}] = \mu$ and covariance matrix $\text{Var}[\mathbf{z}] = \Sigma$, then

$$E[A\mathbf{z}] = A\mu, \quad \text{Var}[A\mathbf{z}] = A\Sigma A^T \quad (\text{D.5.6})$$

R script

The relation above may be used to sample a $[n, 1]$ random vector \mathbf{x} with covariance $\text{Var}[\mathbf{x}] = \Sigma$, starting from the sampling of a $[n, 1]$ random vector \mathbf{z} with $\text{Var}[\mathbf{z}] = I_n$. If we factorize $\Sigma = AA^T$, then $\text{Var}[\mathbf{x}] = \text{Var}[A\mathbf{z}] = AI_n A^T = \Sigma$. In the script `Appendix/cho12cor.R`, we first define the symmetric matrix Σ , then we sample N times the \mathbf{z} vector in the dataset D_N and we multiply D_N by A : it is possible to verify numerically that this is equivalent to sample N times a vector \mathbf{x} with covariance Σ .

•

Theorem 5.2. *Given a random vector \mathbf{z} of dimension $[n \times 1]$ with expected value $E[\mathbf{z}] = \mu$ and covariance matrix $\text{Var}[\mathbf{z}] = \sigma^2 I$, for a generic matrix A of dimension $[n \times n]$ the following relation holds*

$$E[\mathbf{z}^T A \mathbf{z}] = \sigma^2 \text{tr}(A) + \mu^T A \mu \quad (\text{D.5.7})$$

where $\text{tr}(A)$ is the trace of matrix A .

D.6 Correlation and covariance matrices

Given n r.v.s $\mathbf{z}_1, \dots, \mathbf{z}_n$ the correlation matrix C is a symmetric positive-semidefinite $[n, n]$ matrix whose (i, j) entry is the correlation coefficient $\rho(\mathbf{z}_i, \mathbf{z}_j)$ (Equation (2.14.80)).

The following relation exists between the covariance Σ of the n variables and the correlation matrix:

$$C = (\text{diag}(\Sigma))^{-1/2} \Sigma (\text{diag}(\Sigma))^{-1/2}, \quad \Sigma = \begin{bmatrix} \sigma_1 & \dots & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & \dots & \sigma_n \end{bmatrix} C \begin{bmatrix} \sigma_1 & \dots & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & \dots & \sigma_n \end{bmatrix}$$

By using the formula above, the script `Appendix/corcov.R` shows how it is possible to generate a set of examples with predefined pairwise correlation $\bar{\rho}$.

D.7 Conditioning reduce entropy

Let us consider two discrete r.v.s $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$. Since

$$H(\mathbf{y}) = - \sum_{y \in \mathcal{Y}} P_{\mathbf{y}}(y) \log P_{\mathbf{y}}(y)$$

and

$$H(\mathbf{y}|\mathbf{x}) = - \sum_{x \in \mathcal{X}} P_{\mathbf{x}}(x) \sum_{y \in \mathcal{Y}} P_{\mathbf{y}}(y|x) \log P_{\mathbf{y}}(y|x)$$

we have

$$\begin{aligned}
I(\mathbf{x}; \mathbf{y}) &= H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x}) = \\
&= - \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} P_{\mathbf{xy}}(x, y) \log P_{\mathbf{y}}(y) + \sum_{x \in \mathcal{X}} P_{\mathbf{x}}(x) \sum_{y \in \mathcal{Y}} P_{\mathbf{y}}(y|x) \log P_{\mathbf{y}}(y|x) = \\
&= - \sum_{x, y} P(x, y) \log P(y) + \sum_{x, y} P(x, y) \log P(y|x) = \\
&\quad = \sum_{x, y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)}
\end{aligned}$$

Let us introduce the convex function $\varphi(x) = x \log(x)$. Since

$$\varphi\left(\frac{P(x, y)}{P(x)P(y)}\right) = \frac{P(x, y)}{P(x)P(y)} \log \frac{P(x, y)}{P(x)P(y)}$$

we can write

$$I(\mathbf{x}; \mathbf{y}) = \sum_{x, y} P(x)P(y)\varphi\left(\frac{P(x, y)}{P(x)P(y)}\right)$$

From the Jensen's theorem 5.1 we obtain

$$I(\mathbf{x}; \mathbf{y}) = \sum_{x, y} P(x)P(y)\varphi\left(\frac{P(x, y)}{P(x)P(y)}\right) \geq \varphi\left(\sum_{x, y} P(x)P(y)\frac{P(x, y)}{P(x)P(y)}\right) = \varphi(1) = 0$$

that is, $I(\mathbf{x}; \mathbf{y}) \geq 0$.

D.8 Convergence of random variables

Let $\{\mathbf{z}_N\}$, $N = 1, 2, \dots$, 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}_N and $F_{\mathbf{z}}$ the distribution of \mathbf{z} . We introduce the following definitions:

Definition 8.1 (Convergence in probability). We say that

$$\lim_{N \rightarrow \infty} \mathbf{z}_N = \mathbf{z} \text{ in probability} \tag{D.8.8}$$

and we note $\mathbf{z}_N \xrightarrow{P} \mathbf{z}$ if for each $\varepsilon > 0$

$$\lim_{N \rightarrow \infty} P\{|\mathbf{z}_N - \mathbf{z}| \geq \varepsilon\} = 0 \tag{D.8.9}$$

Definition 8.2 (Convergence with probability one). We say that

$$\lim_{N \rightarrow \infty} \mathbf{z}_N = \mathbf{z} \text{ with probability one (or almost surely)} \tag{D.8.10}$$

and we note $\mathbf{z}_N \xrightarrow{a.s.} \mathbf{z}$ if

$$P\{\omega : \lim_{N \rightarrow \infty} \mathbf{z}_N(\omega) = \mathbf{z}(\omega)\} = 1 \tag{D.8.11}$$

Definition 8.3 (Convergence in L_p). For a fixed number $p \geq 1$ we say that

$$\lim_{N \rightarrow \infty} \mathbf{z}_N = \mathbf{z} \text{ in } L_p \tag{D.8.12}$$

if

$$\lim_{N \rightarrow \infty} E[|\mathbf{z}_N - \mathbf{z}|^p] = 0 \tag{D.8.13}$$

The following theorems hold:

Theorem 8.4. *Convergence in L_p implies convergence in probability.*

Theorem 8.5. *Convergence with probability one implies convergence in probability.*

Note however that convergence in probability does not imply convergence in L_2 .

Definition 8.6 (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

$$\lim_{N \rightarrow \infty} F_N(z) = F(z) \quad (\text{D.8.14})$$

for all z for which F is continuous.

It can be shown that

Theorem 8.7. *Convergence with probability implies convergence in distribution.*

Note however that convergence in distribution does not imply convergence in probability.

As a summary

$$\mathbf{z}_N \xrightarrow{a.s.} \mathbf{z} \text{ implies } \mathbf{z}_N \xrightarrow{P} \mathbf{z} \text{ implies } \mathbf{z}_N \xrightarrow{D} \mathbf{z}$$

D.8.1 Example

Let $\mathbf{z} \sim \mathcal{U}(1, 2)$ and $\theta = 0$. Consider the two estimators (stochastic processes) $\hat{\theta}_N^{(1)}$ and $\hat{\theta}_N^{(2)}$ for $N \rightarrow \infty$ where

$$\hat{\theta}_N^1 = \exp^{-\mathbf{z}^N}, \quad \hat{\theta}_N^2 = \begin{cases} \exp^{-N} & \text{with probability } 1 - 1/N \\ 1 & \text{with probability } 1/N \end{cases}$$

For the first estimator, all the trajectories converge to θ (strongly consistent). For the second process, the trajectory which does not converge has a probability decreasing to zero for $N \rightarrow \infty$ (weakly consistent).

D.9 The central limit theorem

Theorem 9.1. *Assume that $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N$ are i.i.d. random variables, discrete or continuous, each having the same probability distribution with finite mean μ and finite variance σ^2 . As $N \rightarrow \infty$, the standardised 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 8.6) 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 2.10 which can be considered as the combined effect of several causes.

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

D.10 The Chebyshev's inequality

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

$$\text{Prob}\{|\mathbf{z} - \mu| \geq d\} \leq \frac{\sigma^2}{d^2} \quad (\text{D.10.15})$$

A validation of the Chebyshev's inequality by Monte Carlo simulation can be found in the R script `Appendix/cheby.R`.

Note that if we put \mathbf{z} equal to the quantity in (2.18.111), then from (2.18.112) and (D.10.15) we find

$$\text{Prob}\{\|\bar{\mathbf{z}} - \mu\| \geq d\} \leq \frac{\sigma^2}{Nd^2} \quad (\text{D.10.16})$$

i.e. the *weak law of large numbers* (Section 2.2.2.1). This law states that the average of a large sample converges in probability to the mean of the distribution.

D.11 Empirical distribution properties

Let (3.2.2) be the empirical distribution of \mathbf{z} obtained from a dataset D_N . 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) are valid:

Theorem 11.1. *For any fixed z*

$$E_{\mathbf{D}_N}[\hat{\mathbf{F}}_{\mathbf{z}}(z)] = F_{\mathbf{z}}(z) \quad (\text{D.11.17})$$

$$\text{Var}[\hat{\mathbf{F}}_{\mathbf{z}}(z)] = \frac{F_{\mathbf{z}}(z)(1 - F_{\mathbf{z}}(z))}{N} \quad (\text{D.11.18})$$

Theorem 11.2 (Glivenko-Cantelli theorem).

$$\sup_{-\infty < z < \infty} |\hat{\mathbf{F}}_{\mathbf{z}}(z) - F_{\mathbf{z}}(z)| \xrightarrow{N \rightarrow \infty} 0 \quad \text{almost surely} \quad (\text{D.11.19})$$

where the definition of almost sure convergence is in Appendix (Def. 8.2).

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

D.12 Useful relations

Some relations

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

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 , $E[(\mathbf{z}_1 + \mathbf{z}_2 + \dots + \mathbf{z}_N)^2] = N^2\mu^2 + N\sigma^2$.

D.13 Minimum of expectation vs. expectation of minimum

Theorem 13.1. *Let us consider M random variables $\mathbf{z}_m, m = 1, \dots, M$. Then*

$$E[\min_m \mathbf{z}_m] \leq \min_m E[\mathbf{z}_m] \quad (\text{D.13.20})$$

Proof. For each m define the r.v. $\delta_m = \mathbf{z}_m - \min_m \mathbf{z}_m$. Now $E[\delta_m] \geq 0$ since $\mathbf{z}_m \geq \min_m \mathbf{z}_m$. Then $E[\mathbf{z}_m] - E[\min_m \mathbf{z}_m] \geq 0$. It follows that

$$\forall m, \quad E[\min_m \mathbf{z}_m] \leq E[\mathbf{z}_m]$$

then

$$E[\min_m \mathbf{z}_m] \leq \min_m E[\mathbf{z}_m]$$

□

The difference $\min_m E[\mathbf{z}_m] - E[\min_m \mathbf{z}_m]$ quantifies then the *selection bias* that occurs in a selection (e.g. by minimisation) process that relies on observed data in a random setting.

D.14 Taylor expansion of function

Let $J(\cdot)$ be a function with p -dimensional argument of the form $\alpha = [\alpha_1, \dots, \alpha_p]$. The Taylor expansion of the function $J(\cdot)$ about $\bar{\alpha}$ can be written as follows

$$J(\alpha) = J(\bar{\alpha}) + \sum_{j=1}^p (\alpha_j - \bar{\alpha}_j) \frac{\partial J(\alpha)}{\partial \alpha_j} \Big|_{\alpha=\bar{\alpha}} + \sum_{i=1}^p \sum_{j=1}^p \frac{(\alpha_i - \bar{\alpha}_i)(\alpha_j - \bar{\alpha}_j)}{2} \frac{\partial^2 J(\alpha)}{\partial \alpha_i \partial \alpha_j} \Big|_{\alpha=\bar{\alpha}} + \dots$$

which can be written in vector-form as follows:

$$J(\alpha) \approx J(\bar{\alpha}) + (\alpha - \bar{\alpha})^T \nabla J(\bar{\alpha}) + (\alpha - \bar{\alpha})^T H(\bar{\alpha})(\alpha - \bar{\alpha})$$

where $\nabla J(\alpha)$ is the gradient vector and $H(\alpha) = [H_{ij}]$ is the Hessian square matrix $[p, p]$ of all second-order derivatives

$$H_{ij} = \frac{\partial^2 J(\alpha)}{\partial \alpha_i \partial \alpha_j}$$

D.15 Proof of Eq. (5.5.28)

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

$$R(\alpha) = \int_{\mathcal{X}, \mathcal{Y}} C(y, \alpha x) p_f(y|x) \pi(x) dy dx \quad (\text{D.15.21})$$

$$= \int_{x=-2}^2 \int_{\mathcal{Y}} (y - \alpha x)^2 p_f(y|x) \pi(x) dx dy \quad (\text{D.15.22})$$

$$= \int_{x=-2}^2 \int_{\mathcal{W}} (x^3 + w - \alpha x)^2 p_w(w) \frac{1}{4} dx dw \quad (\text{D.15.23})$$

$$= \frac{1}{4} \left[\int_{\mathcal{W}} p_w(w) dw \int_{x=-2}^2 (x^3 - \alpha x)^2 dx + \right. \quad (\text{D.15.24})$$

$$\left. + \int_{x=-2}^2 dx \int_{\mathcal{W}} w^2 p_w(w) dw + \int_{\mathcal{W}} \int_{x=-2}^2 2w(x^3 - \alpha x) p_w(w) dw dx \right] \quad (\text{D.15.25})$$

$$= \frac{1}{4} \left[\int_{-2}^2 (x^3 - \alpha x)^2 dx + 4\sigma_w^2 \right] \quad (\text{D.15.26})$$

$$= \frac{1}{4} \int_{-2}^2 (x^3 - \alpha x)^2 dx + \sigma_w^2 \quad (\text{D.15.27})$$

D.16 Optimism of the empirical risk

The optimism of the minimal empirical risk $R_{\text{emp}}(\alpha_N)$ as estimator of the generalisation error $R(\alpha_N)$ (Equation (5.2.3)) can be easily derived from the selection bias formula (D.13.20).

From (5.2.7) we obtain that

$$R_{\text{emp}}(\alpha_N) = \min_{\alpha} R_{\text{emp}}(\alpha, \mathbf{D}_N)$$

is a random quantity since it is a function of the random dataset \mathbf{D}_N .

Since \mathbf{D}_N is i.i.d., from (D.13.20) and (5.2.3) we derive

$$\begin{aligned} E_{\mathbf{D}_N}[R_{\text{emp}}(\alpha_N)] &= E_{\mathbf{D}_N}[\min_{\alpha} L(\mathbf{y}, h(\mathbf{x}, \alpha))] \leq \min_{\alpha} E_{\mathbf{D}_N}[L(\mathbf{y}, h(\mathbf{x}, \alpha))] = \\ &= \min_{\alpha} E_{\mathbf{x}, \mathbf{y}}[L(\mathbf{y}, h(\mathbf{x}, \alpha))] = \min_{\alpha} R(\alpha) \leq R(\alpha_N) \end{aligned}$$

Though this derivation shows the biasedness of the empirical risk, it does not provide information about the size of the bias. The following section derives an analytical expression of such value.

D.16.1 Bias of the quadratic empirical risk

Consider a regression framework where $\mathbf{y} = f(x) + \mathbf{w}$, with $E[\mathbf{w}] = 0$ and $\text{Var}[\mathbf{w}] = \sigma_{\mathbf{w}}^2$, and h_N an estimation of f obtained by minimising the empirical risk in a dataset $D_N \sim \mathbf{y}$. According to the derivation in [65], let us consider the quantity

$$g_N(x) = E_{\mathbf{D}_N, \mathbf{y}}[(\mathbf{y} - h(x, \alpha(\mathbf{D}_N)))^2] = E_{\mathbf{D}_N}[E_{\mathbf{y}}[(\mathbf{y} - h_N)^2]]$$

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

$$(y - f)^2 + (f - h_N)^2 = (y - h_N)^2 + 2(y - f)(h_N - f) \quad (\text{D.16.28})$$

Note that since $E_{\mathbf{y}}[\mathbf{y}] = f$, for a given h_N

$$\begin{aligned} E_{\mathbf{y}}[(\mathbf{y} - h_N)^2] &= E_{\mathbf{y}}[(\mathbf{y} - f)^2 + (f - h_N)^2 + 2(\mathbf{y} - f)(f - h_N)] = \\ &= E_{\mathbf{y}}[(\mathbf{y} - f)^2 + (f - h_N)^2] = E_{\mathbf{y}}[(\mathbf{y} - f)^2] + (f - h_N)^2 \quad (\text{D.16.29}) \end{aligned}$$

From $E_{\mathbf{y}}[(\mathbf{y} - f)^2] = E_{\mathbf{D}_N}[(\mathbf{y} - f)^2]$, (D.16.29) and (D.16.28) it follows

$$\begin{aligned} g_N(x) &= E_{\mathbf{D}_N}[E_{\mathbf{y}}[(\mathbf{y} - h_N)^2]] = \\ &= E_{\mathbf{D}_N}[E_{\mathbf{y}}[(\mathbf{y} - f)^2] + (f - h_N)^2] = E_{\mathbf{D}_N}[(\mathbf{y} - f)^2 + (f - h_N)^2] = \\ &= E_{\mathbf{D}_N}[(\mathbf{y} - h_N)^2 + 2(\mathbf{y} - f)(h_N - f)] \end{aligned}$$

By averaging the quantity $g_N(x)$ over the \mathcal{X} domain we obtain

$$\begin{aligned} \text{MISE} &= E_{\mathbf{D}_N, \mathbf{y}, \mathbf{x}}[(\mathbf{y} - h(x, \alpha(\mathbf{D}_N)))^2] = E_{\mathbf{x}}[g_N(x)] = \\ &= E_{\mathbf{D}_N, \mathbf{x}}[(\mathbf{y} - h_N)^2] + 2E_{\mathbf{D}_N, \mathbf{x}}[(\mathbf{y} - f)(h_N - f)] = \\ &= E_{\mathbf{D}_N}[\widehat{\text{MISE}}_{\text{emp}}] + 2\text{Cov}[\mathbf{h}_N, \mathbf{y}] \quad (\text{D.16.30}) \end{aligned}$$

where $\text{Cov}[\mathbf{h}_N, \mathbf{y}] = E_{\mathbf{D}_N, \mathbf{x}}[(\mathbf{y} - f)(h_N - f)]$ and $\widehat{\text{MISE}}_{\text{emp}}$ is the quantity (5.2.8) 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. A Monte Carlo verification of this formula is made available in the script `Appendix/genercov.R`.

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 $H^T = H$

$$\begin{aligned} N \text{Cov}[\mathbf{h}_N, \mathbf{y}] &= E_{\mathbf{D}_N}[(\mathbf{Y} - F)^T (H\mathbf{Y} - F)] = \\ &= E_{\mathbf{D}_N}[\mathbf{Y}^T H\mathbf{Y} - \mathbf{Y}^T F - F^T H\mathbf{Y} - \mathbf{Y}^T H^T F] = E_{\mathbf{D}_N}[\mathbf{Y}^T H(\mathbf{Y} - F)] = \\ &= \sigma^2 \text{tr}(H) + F^T HF - F^T HF = \sigma^2 \text{tr}(H) = \sigma^2 \text{tr}((X^T X)^{-1} X^T X) = \sigma^2 p \end{aligned}$$

where $\text{tr}(H)$ is the trace of the matrix H , \mathbf{Y} is a random vector of size $[N, 1]$ and F is the vector of the N regression function values $f(x_i)$. It follows that $\text{Cov}[\mathbf{h}_N, \mathbf{y}] = \sigma^2 p/N$ and then the C_p formula (6.8.33). Note that the trace of H is also known as the effective number of parameters.

Appendix E

Plug-in estimators

This appendix contains the expression of the plug-in estimators of some interesting parameters:

- Skewness of a scalar random variable \mathbf{z} : given a dataset $D_N = \{z_1, \dots, z_N\}$ the plug-in estimate of the skewness (2.11.42) is

$$\hat{\gamma} = \frac{\sum_{i=1}^N (z_i - \hat{\mu})^3}{N \hat{\sigma}^3} \quad (\text{E.0.1})$$

where $\hat{\mu}$ and $\hat{\sigma}$ are defined in (3.3.4) and (3.3.5), respectively.

- Kurtosis of a scalar random variable \mathbf{z} : given a dataset $D_N = \{z_1, \dots, z_N\}$ the plug-in estimate of the kurtosis (2.11.43) is

$$\hat{\kappa} = \frac{\sum_{i=1}^N (z_i - \hat{\mu})^4}{N \hat{\sigma}^4} \quad (\text{E.0.2})$$

where $\hat{\mu}$ and $\hat{\sigma}$ are defined in (3.3.4) and (3.3.5), respectively.

- Correlation of two scalar random variables \mathbf{x} and \mathbf{y} : given a dataset $D_N = \{\langle x_1, y_1 \rangle, \dots, \langle x_N, y_N \rangle\}$ where $x_i \in \mathbb{R}, y_i \in \mathbb{R}$ the plug-in estimate of the correlation (2.14.80) is

$$\hat{\rho} = \frac{\sum_{i=1}^N (x_i - \hat{\mu}_x)(y_i - \hat{\mu}_y)}{\hat{\sigma}_x \hat{\sigma}_y} \quad (\text{E.0.3})$$

where $\hat{\mu}_x$ ($\hat{\mu}_y$) and $\hat{\sigma}_x^2$ ($\hat{\sigma}_y^2$) denote the sample mean and sample variance of \mathbf{x} (\mathbf{y}). The sampling distribution of $\hat{\rho}$ in the case of linearly related variables ($\mathbf{y} = K\mathbf{x}$) is computed by Monte Carlo and shown in the R script `Appendix/sam_cordis.R`.

- Covariance matrix of a n -dimensional random vector \mathbf{z} : given a dataset $D_N = \{z_1, \dots, z_N\}$ where $z_i = [z_{i1}, \dots, z_{in}]^T$ is a $[n, 1]$ vector, the plug-in estimator of the covariance matrix (2.15.85) is the $[n, n]$ matrix

$$\hat{\Sigma} = \frac{\sum_{i=1}^N (z_i - \hat{\mu})(z_i - \hat{\mu})^T}{N - 1} \quad (\text{E.0.4})$$

whose jk th entry is

$$\hat{\Sigma}_{jk} = \frac{\sum_{i=1}^N (z_{ij} - \hat{\mu}_j)(z_{ik} - \hat{\mu}_k)^T}{N - 1}$$

and $\hat{\mu}$ is the $[n, 1]$ vector

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

and

$$\hat{\mu}_j = \frac{\sum_{i=1}^N z_{ij}}{N}.$$

Note that (E.0.4) can be also written in matrix form

$$\hat{\Sigma} = \frac{(Z - 1_N \hat{\mu}^T)^T (Z - 1_N \hat{\mu}^T)}{N - 1}$$

where Z is a $[N, n]$ matrix whose i th row is z_i^T and 1_N is a $[N, 1]$ vector of ones.

- Correlation matrix of a n -dimensional random vector \mathbf{z} : the correlation matrix is a symmetric $[n, n]$ matrix whose jk th entry is the correlation between the scalar random variables \mathbf{z}_j and \mathbf{z}_k . Given a dataset $D_N = \{z_1, \dots, z_N\}$ where $z_i = [z_{i1}, \dots, z_{in}]^T$ is a $[n, 1]$ vector, the plug-in estimator can be written as the covariance¹

$$\hat{\mathbf{P}} = \frac{\tilde{\mathbf{Z}}^T \mathbf{Z}}{N}$$

of the scaled matrix

$$\tilde{\mathbf{Z}} = C \mathbf{Z} \mathbf{D}^{-1}$$

where

$$C = I_N - \frac{1_N 1_N^T}{N}$$

is the centering $[N, N]$ matrix, I_N is the diagonal unit matrix, 1_N is a $[N, 1]$ vector of ones and

$$\mathbf{D} = \text{diag}(\hat{\sigma}_1, \dots, \hat{\sigma}_n)$$

is a diagonal $[n, n]$ scaling matrix where $\hat{\sigma}_j^2$ is the sample variance of \mathbf{z}_j .

The diagonal entries of $\hat{\mathbf{P}}$ are all 1. The jk th entry ($j \neq k$) of the matrix $\hat{\mathbf{P}}$ can be also obtained by applying (E.0.3) to the j th and k th column of Z .

¹see also <http://users.stat.umn.edu/~helwig/notes/datamat-Notes.pdf>

Appendix F

Kernel functions

A real-valued kernel function $K(\cdot, \cdot)$ is a nonnegative symmetric continuous function

$$K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^+$$

where symmetric means that $K(x_1, x_2) = K(x_2, x_1)$ for all $x_1 \in \mathbb{R}^n, x_2 \in \mathbb{R}^n$.

Examples of kernel functions are:

- Linear: $K(x_1, x_2) = \langle x_1, x_2 \rangle = x_1^T x_2$
- Polynomial of degree $d \in \mathbb{Z}^+$: $K(x_1, x_2) = (b + x_1^T x_2)^d, \quad b \in \mathbb{R}$
- Perceptron: $K(x_1, x_2) = \tanh(ax_1^T x_2 + b), \quad a, b \in \mathbb{R}$

A kernel function is said to satisfy the Mercer's condition if for all square-integrable functions g (i.e. satisfying $\int |g(x)|^2 dx < \infty$) one has

$$\int \int g(x)K(x, y)g(y)dxdy \geq 0 \quad (\text{F.0.1})$$

F.1 Distance-based kernel functions

Let us define a distance function

$$d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^+ \quad (\text{F.1.2})$$

which depends on a parameter B called *width* or *bandwidth*. The kernel function can be expressed as a function

$$K : \mathbb{R}^+ \rightarrow \mathbb{R}^+ \quad (\text{F.1.3})$$

of the distance d between the two inputs. 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 distance-based kernel functions where B is the bandwidth parameter:

Inverse distance:

$$K(d) = \frac{1}{(d/B)^p} \quad (\text{F.1.4})$$

This function goes to infinity as the distance approaches zero.

Corrected inverse distance:

$$K(d) = \frac{1}{1 + (d/B)^p} \quad (\text{F.1.5})$$

Gaussian kernel:

$$K(d) = \exp\left(-\frac{d^2}{B^2}\right) \quad (\text{F.1.6})$$

Exponential kernel:

$$K(d) = \exp\left(-\left|\frac{d}{B}\right|\right) \quad (\text{F.1.7})$$

Quadratic or Epanechnikov kernel:

$$K(d) = \begin{cases} \left(1 - \frac{d^2}{B}\right) & \text{if } |d| < B \\ 0 & \text{otherwise} \end{cases} \quad (\text{F.1.8})$$

Tricube kernel:

$$K(d) = \begin{cases} \left(1 - \left|\frac{d}{B}\right|^3\right)^3 & \text{if } |d| < B \\ 0 & \text{otherwise} \end{cases} \quad (\text{F.1.9})$$

Uniform kernel:

$$K(d) = \begin{cases} 1 & \text{if } |d| < B \\ 0 & \text{otherwise} \end{cases} \quad (\text{F.1.10})$$

Triangular kernel:

$$K(d) = \begin{cases} 1 - \left|\frac{d}{B}\right| & \text{if } |d| < B \\ 0 & \text{otherwise} \end{cases} \quad (\text{F.1.11})$$

Appendix G

Companion R package

Several scripts are used in the main text to illustrate statistical and machine learning notions. All the scripts have been implemented in R and are contained in the R package `gbcodem`.

To install the R package `gbcodem` containing all the scripts mentioned in the text you should run the following R instructions in the R console.

```
> library(devtools)  
  
> install_github("gbonte/gbcodem")  
  
> require(gbcodem)
```

Once installed, all the scripts will be available in the root directory of the package. In order to retrieve the directory containing the `gbcodem` package you should type

```
> system.file(package = "gbcodem")
```

To change the directory to the one containing scripts

```
> setwd(find.package("gbcodem"))
```

If you wish to run a script mentioned in the main text (e.g. the script `Probability/freq.R`) without changing the local directory you should run

```
> source(system.file("scripts","Probability/freq.R",package = "gbcodem"))
```

If you wish to edit a script mentioned in the main text (e.g. the script `Probability/freq.R`) without changing the local directory you should run

```
> edit(file=system.file("scripts","Probability/freq.R",package = "gbcodem"))
```

If you wish to execute a Shiny dashboard (e.g. `leastsquares.R`) you should run

```
> library(shiny)  
> source(system.file("shiny","leastsquares.R",package = "gbcodem"))
```


Appendix H

Companion R Shiny dashboards

Several Shiny dashboards are used in the main text to illustrate statistical and machine learning notions. All the Shiny dashboards are contained in the directory `shiny` of the R package `gbcod`e and require the installation of the library `shiny`. To run a Shiny dashboard (e.g. `condpro.R`) you should first move to their directory by

```
> setwd(paste(find.package("gbcod"), "shiny", sep="/"))
```

and then run

```
> runApp("condpro.R")
```

The Shiny dashboards are also active under Shinyapps¹. To run a Shiny dashboard named `NAME.R` go to <https://gbonte.shinyapps.io/NAME>. For instance, to run the Shiny dashboard `condpro.R` go to:

<https://gbonte.shinyapps.io/condpro>

H.1 List of Shiny dashboards

- `mcarlo.R`: visualisation by means of Monte Carlo simulation of
 1. transformation of a r.v.
 2. result of operation on two r.v.s
 3. central limit theorem
 4. result of linear combination of two independent r.v.s
- `condpro.R`: visualisation of conditional probability vs. marginal probability in the bivariate gaussian case and in the regression function case.
- `estimation.R`: visualisation of different problems of estimation:
 1. estimation of mean and variance of a univariate normal r.v.: bias/variance visualisation
 2. estimation of mean and variance of a univariate uniform r.v.: bias/variance visualisation
 3. estimation of confidence interval of the mean of a univariate normal r.v.
 4. maximum-likelihood estimation of the mean of a univariate normal r.v.: visualisation of the log-likelihood function together with the value of the maximum-likelihood estimator

¹<https://www.shinyapps.io>

- 5. maximum-likelihood estimation of the mean and the variance of a univariate normal r.v.: visualisation of the bivariate log-likelihood function together with the value of the maximum-likelihood estimator
- 6. estimation of mean and covariance of a bivariate normal r.v.: bias/variance visualisation
- 7. least-squares estimation of the parameters of a linear target function: visualisation of bias/variance of the predicted conditional expectation and of the parameter estimators
- 8. least-squares estimation of the parameters of a nonlinear target function: visualisation of bias/variance of the predicted conditional expectation
- **bootstrap.R:** study of the accuracy of the bootstrap estimation of the sampling distribution, estimator variance and estimator bias. The dashboard considers the case of sample average for which it is known that bias is null and variance is inversely proportional to N (Section 3.5.3). The dashboard shows that the bootstrap returns an accurate estimation of bias and variance of sample average.
- **leastsquares.R:** visualisation of the minimisation of the empirical risk with gradient-based iteration. 3 dashboards:
 1. linear least-squares: visualisation of the convex empirical risk function and position of the estimation as gradient-based iteration proceeds
 2. NNet least-squares: visualisation of the estimated regression function (single layer, 3 hidden nodes NNET) and the associated empirical risk as gradient-based iteration proceeds
 3. KNN cross-validation: illustration of the points used for test as cross-validation proceeds in the case of a KNN regressor (variable number of neighbours).
- **regression.R:** visualisation of the model selection trade-off in regression by showing the impact of different kinds of hyper-parameters (degree of polynomial model, number of neighbours in locally constant and locally linear fitting, number of trees in Random Forest) on the bias, variance and generalisation error.
- **classif.R:** visualisation of different classification notions in 4 dashboards:
 1. Univariate: visualise the relation between posterior probability and class conditional densities in a univariate binary classification task
 2. Linear discriminant: visualise the relation between bivariate class conditional densities and linear discriminant
 3. Perceptron: visualise the evolution of the perceptron hyperplane during the gradient based minimisation of the hyperplane misclassification and the SVM hyperplane
 4. Assessment: visualise the relation between ROC curve, PR curve, confusion matrix and classifier threshold in a univariate binary classification task.
- **classif2.R:** visualisation of direct and inverse conditional distributions in the unimodal and bimodal case.

Bibliography

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [2] Charu C. Aggarwal. *Linear Algebra and Optimization for Machine Learning - A Textbook*. Springer, 2020.
- [3] 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.
- [4] 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.
- [5] D. W. Aha. Editorial of special issue on lazy learning. *Artificial Intelligence Review*, 11(1–5):1–6, 1997.
- [6] H. Akaike. Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mechanics*, 21:243–247, 1969.
- [7] D. M. Allen. The relationship between variable and data augmentation and a method of prediction. *Technometrics*, 16:125–127, 1974.
- [8] Christophe Ambroise and Geoffrey J. McLachlan. Selection bias in gene extraction on the basis of microarray gene-expression data. *PNAS*, 99(10):6562–6566, 2002.
- [9] B. D. O. Anderson and M. Deistler. Identifiability in dynamic errors-in-variables models. *Journal of Time Series Analysis*, 5:1–13, 1984.
- [10] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1–5):11–73, 1997.
- [11] R. Babuska. *Fuzzy Modeling and Identification*. PhD thesis, Technische Universiteit Delft, 1996.
- [12] 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.
- [13] D. Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.
- [14] Vic Barnett. *Comparative statistical inference*. Wiley series in probability and mathematical statistics. Wiley, Chichester, 2. ed edition, 1982.
- [15] 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.

- [16] Thomas Lumley based on Fortran code by Alan Miller. *leaps: Regression Subset Selection*, 2020. R package version 3.1.
- [17] W. G. Baxt. Improving the accuracy of an artificial neural network using multiple differently trained networks. *Neural Computation*, 4:772–780, 1992.
- [18] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey, 2015. cite arxiv:1502.05767Comment: 43 pages, 5 figures.
- [19] 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.
- [20] 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.
- [21] H. Bersini and G. Bontempi. Now comes the time to defuzzify the neuro-fuzzy models. *Fuzzy Sets and Systems*, 90(2):161–170, 1997.
- [22] 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.
- [23] M. Birattari and G. Bontempi. The lazy package for r. lazy learning for local regression. Technical Report 38, IRIDIA ULB, 2003.
- [24] 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.
- [25] C. M. Bishop. *Neural Networks for Statistical Pattern Recognition*. Oxford University Press, Oxford, UK, 1994.
- [26] S. Bittanti. *Model Identification and Data Analysis*. Wiley, 2019.
- [27] Joseph K. Blitzstein and Jessica Hwang. *Introduction to Probability Second Edition*. 2019.
- [28] Hendrik Blockeel, Claude Sammut, and Geoffrey Webb. Hypothesis space, 2011-01-01.
- [29] G Bontempi. A blocking strategy to improve gene selection for classification of gene expression data. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 4(2):293–300, 2007.
- [30] G. Bontempi. *Statistical foundations of machine learning: the book*. LeanPub, 2022. <https://leanpub.com/statisticalfoundationsofmachinelearning>.
- [31] 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), Stockholm, Sweden)*, pages 325–328, 1997.
- [32] G. Bontempi, M. Birattari, and H. Bersini. Lazy learning for modeling and control design. *International Journal of Control*, 72(7/8):643–658, 1999.
- [33] G. Bontempi, M. Birattari, and H. Bersini. A model selection approach for local learning. *Artificial Intelligence Communications*, 121(1), 2000.
- [34] G. Bontempi, M. Birattari, and P.E. Meyer. Combining lazy learning, racing and subsampling for effective feature selection. In *Adaptive and Natural Computing Algorithms: Proceedings of the International Conference ICANNGA05*, pages 393–396. Springer Computer Science, 2005.
- [35] G. Bontempi and O. Caelen. A selecting-the-best method for budgeted model selection. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 249–262, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

- [36] Olivier Bousquet and Andr Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.
- [37] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [38] L. Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996.
- [39] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.
- [40] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [41] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.
- [42] K. P. Burnham and D. R. Anderson. *Model Selection and Multimodel Inference*. Springer, 1998.
- [43] Gavin C. Cawley. Over-fitting in model selection and its avoidance. In Jaakko Hollmn, Frank Klawonn, and Allan Tucker, editors, *IDA*, volume 7619 of *Lecture Notes in Computer Science*, page 1. Springer, 2012.
- [44] V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory, and Methods*. Wiley, New York, 1998.
- [45] F. Chollet and J.J. Allaire. *Deep Learning with R*. Manning, 2018.
- [46] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:829–836, 1979.
- [47] 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.
- [48] W. S. Cleveland and C. Loader. Smoothing by local regression: Principles and methods. *Computational Statistics*, 11, 1995.
- [49] T. Cover and P. Hart. Nearest neighbor pattern classification. *Proc. IEEE Trans. Inform. Theory*, pages 21–27, 1967.
- [50] 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.
- [51] 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.
- [52] Andrea Dal Pozzolo, Olivier Caelen, and Gianluca Bontempi. When is undersampling effective in unbalanced classification tasks? In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 200–215. Springer, Cham, 2015.
- [53] Andrea Dal Pozzolo, Olivier Caelen, Yann-Ael Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications*, 41(10):4915–4928, 2014.
- [54] Peter Dalgaard. *Introductory statistics with R*. Springer, 2002.
- [55] Rhian Davies, Steph Locke, and Lucy D'Agostino McGowan. *datasauRus: Datasets from the Datasaurus Dozen*, 2022. R package version 0.1.6.
- [56] A. Dean and D. Voss. *Design and Analysis of Experiments*. Springer Verlag, New York, NY, USA, 1999.
- [57] Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020.
- [58] 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.

- [59] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer Verlag, 1996.
- [60] N. R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley and Sons, New York, 1981.
- [61] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1976.
- [62] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley and sons, 2001.
- [63] B. Efron. The estimation of prediction error: Covariance penalties and cross-validation. *Annals of Statistics*, pages 1–26, 1979.
- [64] B. Efron. *The Jackknife, the Bootstrap and Other Resampling Plans*. SIAM, 1982. Monograph 38.
- [65] B. Efron. Bootstrap methods: Another look at the jackknife. *JASA*, pages 619–642, 2004.
- [66] B. Efron and T. Hastie. *Computer Age Statistical Inference*. CambridgePress, 2016.
- [67] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, New York, NY, 1993.
- [68] B. Efron and R. J. Tibshirani. Cross-validation and the bootstrap: estimating the error rate of a prediction rule. Technical report, Stanford University, 1995.
- [69] Gamaleldin F. Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alexey Kurakin, Ian J. Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both computer vision and time-limited humans. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicol Cesa-Bianchi, and Roman Garnett, editors, *NeurIPS*, pages 3914–3924, 2018.
- [70] J. Fan and I. Gijbels. Variable bandwidth and local linear regression smoothers. *The Annals of Statistics*, 20(4):2008–2036, 1992.
- [71] J. Fan and I. Gijbels. Adaptive order polynomial fitting: bandwidth robustification and bias reduction. *J. Comp. Graph. Statist.*, 4:213–227, 1995.
- [72] J. Fan and I. Gijbels. *Local Polynomial Modelling and Its Applications*. Chapman and Hall, 1996.
- [73] 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.
- [74] V. Fedorov. *Theory of Optimal Experiments*. Academic Press, 1972.
- [75] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(1):3133–3181, January 2014.
- [76] F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, 2004.
- [77] 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.
- [78] J. H. Friedman. Flexible metric nearest neighbor classification. Technical report, Stanford University, 1994.
- [79] Jerome H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, February 2002.
- [80] A. Gelman. *Bayesian Data Analysis*. Chapman and Hall, 2004.
- [81] J.E. Gentle. *Matrix Algebra (2nd ed.)*. Springer, 2017.
- [82] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic Press, London, 1981.
- [83] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

- [84] G.H. Golub and C.F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 1996.
- [85] T. R. Golub, D. K. Slonin, P. Tamayo, C. Huard, and M. Gaasenbeek. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.
- [86] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [87] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
- [88] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lotfi A. Zadeh. *Feature Extraction: Foundations and Applications*. Springer-Verlag New York, Inc., 2006.
- [89] B. Haibe-Kains, C. Desmedt, S. Loi, M. Delorenzi, C. Sotiriou, and G. Bontempi. *Computational Intelligence in Clinical Oncology: Lessons Learned from an Analysis of a Clinical Study*, pages 237–268. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [90] Lewis G Halsey. The reign of the p-value is over: what alternative analyses could we employ to fill the power vacuum? *Biology Letters*, 15, 2019.
- [91] D. J. Hand. *Discrimination and classification*. John Wiley, New York, 1981.
- [92] W. Hardle and J. S. Marron. Fast and simple scatterplot smoothing. *Comp. Statist. Data Anal.*, 20:1–17, 1995.
- [93] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, London, UK, 1990.
- [94] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–615, 1996.
- [95] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, 2001.
- [96] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning (2nd edition)*. Springer, 2009.
- [97] Trevor Hastie and Robert Tibshirani. Efficient quadratic regularization for expression arrays. *Biostatistics (Oxford, England)*, 5(3):329–40, Jul 2004.
- [98] J. S. U. Hjorth. *Computer Intensive Statistical Methods*. Chapman and Hall, 1994.
- [99] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):832–844, 1998.
- [100] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of American Statistical Association*, 58:13–30, 1963.
- [101] P. J. Huber. *Robust Statistics*. Wiley, New York, 1981.
- [102] 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.
- [103] J.-S. R. Jang. Anfis: Adaptive-network-based fuzzy inference systems. *IEEE Transactions on Fuzzy Systems*, 23(3):665–685, 1993.
- [104] J. S. R. Jang, C. T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing*. Matlab Curriculum Series. Prentice Hall, 1997.
- [105] E.T. Jaynes. *Probability theory : the logic of science*. Cambridge University Press, 2003.
- [106] T. A. Johansen and B. A. Foss. Constructing narmax models using armax models. *International Journal of Control*, 58:1125–1153, 1993.
- [107] 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.
- [108] Daniel Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York, 2011.

- [109] V. Y. Katkovnik. Linear and nonlinear methods of nonparametric regression analysis. *Soviet Automatic Control*, 5:25–34, 1979.
- [110] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
- [111] 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>.
- [112] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.
- [113] D. Koller and N. Friedman. *Probabilistic graphical models*. The MIT Press, 2009.
- [114] A. N. Kolmogorov. *Foundations of Probability*. Berlin, 1933.
- [115] J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.
- [116] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [117] Raoul LePage and Lynne Billard. *Exploring the limits of bootstrap*. John Wiley & Sons, 1992.
- [118] R. J.A. Little and D. B. Rubin. *Statistical analysis with missing data*. Wiley, 2002.
- [119] L. Ljung. *System identification: Theory for the User*. Prentice-Hall, Englewood Cliffs, NJ, 1987.
- [120] Sherene Loi, Benjamin Haibe-Kains, Christine Desmedt, Pratyaksha Wirapati, Françoise Lallemand, Andrew M Tutt, Cheryl Gillet, Paul Ellis, Kenneth Ryder, James F Reid, Gianluca Bontempi, et al. Predicting prognosis using molecular profiling in estrogen receptor-positive breast cancer treated with tamoxifen. *BMC genomics*, 9(1):1–12, 2008.
- [121] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison Wesley, Reading, MA, 1984.
- [122] C. Mallows. Discussion of a paper of beaton and tukey. *Technometrics*, 16:187–188, 1974.
- [123] C. L. Mallows. Some comments on c_p . *Technometrics*, 15:661, 1973.
- [124] O. Maron and A. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11(1–5):193–225, 1997.
- [125] A. Miller. *Subset Selection in Regression (2nd ed.)*. Chapman and Hall, 2002.
- [126] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.
- [127] 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.
- [128] K.P. Murphy, editor. *Probabilistic Machine Learning: an introduction*. The MIT Press, 2022.
- [129] R. Murray-Smith. *A local model network approach to nonlinear modelling*. PhD thesis, Department of Computer Science, University of Strathclyde, Strathclyde, UK, 1994.
- [130] 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.
- [131] R. H. Myers. *Classical and Modern Regression with Applications*. PWS-KENT Publishing Company, Boston, MA, second edition, 1994.
- [132] E. Nadaraya. On estimating regression. *Theory of Prob. and Appl.*, 9:141–142, 1964.
- [133] Cathy O’Neil. *Weapons of Math Destruction*. Crown, New York, 2016.

- [134] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, 1991.
- [135] Simon Parsons and Anthony Hunter. *A Review of Uncertainty Handling Formalisms*, pages 8–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.
- [136] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.
- [137] Y. Pawitan. *In all likelihood: statistical modelling and inference using likelihood*. Oxford Science, 2001.
- [138] Judea Pearl. *Probabilistic reasoning in intelligent systems : networks of plausible inference*. Morgan Kaufmann, San Francisco, Calif., 2009. Example for Explaning away.
- [139] Jean-Philippe Pellet and Andr Elisseeff. Using markov blankets for causal structure learning. *J. Mach. Learn. Res.*, 9:1295–1342, 2008.
- [140] 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.
- [141] 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.
- [142] 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.
- [143] M. J. D. Powell. *Algorithms for Approximation*, chapter Radial Basis Functions for multivariable interpolation: a review, pages 143–167. Clarendon Press, Oxford, 1987.
- [144] 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.
- [145] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
- [146] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.
- [147] 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.
- [148] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
- [149] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, August 1999.
- [150] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics*, 27:832–837, 1956.
- [151] A. Roth. *The Algorithmic Foundations of Adaptive Data Analysis*. 2017. Lectures at the University of Pennsylvania.
- [152] D. E. Rumelhart, G. E. Hinton, and R. K. Williams. Learning representations by backpropagating errors. *Nature*, 323(9):533–536, 1986.
- [153] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23:2507–2517, 2007.
- [154] R. E. Schapire. *Nonlinear Estimation and Classification*, chapter The boosting approach to machine learning: An overview. Springer,.
- [155] L. Schneps and C. Colmez. *Math on Trial: How Numbers Get Used and Abused in the Courtroom*. EBL ebooks online. Basic Books, 2013.
- [156] D. W. Scott. *Multivariate density estimation*. Wiley, New York, 1992.
- [157] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, illustrated edition edition, 2004.

- [158] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1987.
- [159] C. Stone. Consistent nonparametric regression. *The Annals of Statistics*, 5:595–645, 1977.
- [160] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36(1):111–147, 1974.
- [161] 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.
- [162] Johan AK Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor, and Joos Vandewalle. *Least squares support vector machines*, volume 4. World Scientific, 2002.
- [163] 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.
- [164] M. Taniguchi and V. Tresp. Averaging regularized estimators. *Neural Computation*, (9), 1997.
- [165] H. Tijms. *Understanding probability*. Cambridge, 2004.
- [166] V. Tresp. *Handbook for neural network signal processing*, chapter Committee machines. CRC Press, 2001.
- [167] I Tsamardinos and CF Aliferis. Towards Principled Feature Selection: Relevancy, Filters and Wrappers. In *Ninth International Workshop on Artificial Intelligence and Statistics, AISTAT*, 2003.
- [168] ttphns (<https://stats.stackexchange.com/users/3277/ttphns>). Multiple regression or partial correlation coefficient? and relations between the two. Cross Validated. URL:<https://stats.stackexchange.com/q/76819> (version: 2021-08-14).
- [169] V. N. Vapnik. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems*, volume 4, Denver, CO, 1992.
- [170] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, NY, 1995.
- [171] V. N. Vapnik. *Statistical Learning Theory*. Springer, 1998.
- [172] W. N. Venables and D. M. Dmth. *An Introduction to R*. Network Theory, 2002.
- [173] 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.
- [174] L. Wasserman. *All of statistics*. Springer, 2004.
- [175] G. Watson. Smooth regression analysis. *Sankhya, Series A*(26):359–372, 1969.
- [176] S. M. Weiss and C. A. Kulikowski. *Computer Systems that learn*. Morgan Kaufmann, San Mateo, California, 1991.
- [177] B. Widrow and M.E. Hoff. Adaptive switching circuits. In *WESCON Convention Record Part IV*, 1960.
- [178] D. H. Wolpert. Stacked generalization. Technical Report LA-UR-90-3460, Los Alamos, NM, 1990. He proposes a more sophisticated voting scheme where the a second level function output performs the final estimation of the true class. I don’t think this is going to referenced in the final thesis.
- [179] D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8:1341–1390, 1996.
- [180] D. H. Wolpert and R. Kohavi. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the 13th International Conference on Machine Learning*, pages 275–283, 1996.
- [181] Simon Wood. *Generalized additive models: an introduction with R*. CRC Press, 2006.

- [182] Zenglin Xu, Rong Jin, Jieping Ye, Michael R. Lyu, and Irwin King. Non-monotonic feature selection. In Andrea Pohoreckyj Danyluk, Lon Bottou, and Michael L. Littman, editors, *ICML*, volume 382 of *ACM International Conference Proceeding Series*, pages 1145–1152. ACM, 2009.
- [183] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.