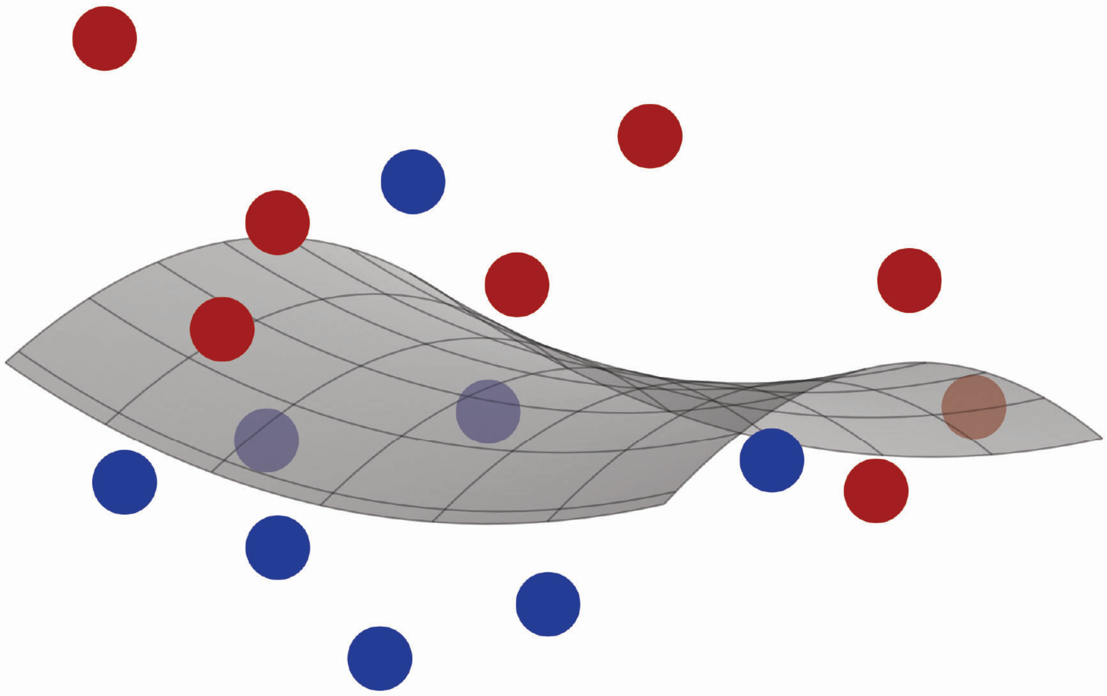
Foundations of Machine Learning

second edition



Mehryar Mohri,
Afshin Rostamizadeh,
and Ameet Talwalkar

Foundations of Machine Learning

second edition

Foundations of Machine Learning

second edition

Mehryar Mohri

Afshin Rostamizadeh

Ameet Talwalkar

The MIT Press
Cambridge, Massachusetts
London, England

This book was set in LaTeX by the authors. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Names: Mohri, Mehryar, author. | Rostamizadeh, Afshin, author. | Talwalkar, Ameet, author.
Title: Foundations of machine learning / Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar.
Description: Second edition. | Cambridge, MA : The MIT Press, [2018] | Series: Adaptive computation and machine learning series | Includes bibliographical references and index.
Identifiers: LCCN 2018022812 | ISBN 9780262039406 (hardcover : alk. paper)
Subjects: LCSH: Machine learning. | Computer algorithms.
Classification: LCC Q325.5 .M64 2018 | DDC 006.3/1--dc23 LC record available at https://lccn.loc.gov/2018022812

10 9 8 7 6 5 4 3 2

Contents

Preface xiii

1 Introduction 1

1.1 What is machine learning? 1
1.2 What kind of problems can be tackled using machine learning? 2
1.3 Some standard learning tasks 3
1.4 Learning stages 4
1.5 Learning scenarios 6
1.6 Generalization 7

2 The PAC Learning Framework 9

2.1 The PAC learning model 9
2.2 Guarantees for finite hypothesis sets — consistent case 15
2.3 Guarantees for finite hypothesis sets — inconsistent case 19
2.4 Generalities 21
2.4.1 Deterministic versus stochastic scenarios 21
2.4.2 Bayes error and noise 22
2.5 Chapter notes 23
2.6 Exercises 23

3 Rademacher Complexity and VC-Dimension 29

3.1 Rademacher complexity 30
3.2 Growth function 34
3.3 VC-dimension 36
3.4 Lower bounds 43
3.5 Chapter notes 48
3.6 Exercises 50

4 Model Selection 61

4.1 Estimation and approximation errors 61
4.2 Empirical risk minimization (ERM) 62
4.3 Structural risk minimization (SRM) 64

4.4 Cross-validation 68
4.5 n-Fold cross-validation 71
4.6 Regularization-based algorithms 72
4.7 Convex surrogate losses 73
4.8 Chapter notes 77
4.9 Exercises 78

5 Support Vector Machines 79

5.1 Linear classification 79
5.2 Separable case 80
5.2.1 Primal optimization problem 81
5.2.2 Support vectors 83
5.2.3 Dual optimization problem 83
5.2.4 Leave-one-out analysis 85
5.3 Non-separable case 87
5.3.1 Primal optimization problem 88
5.3.2 Support vectors 89
5.3.3 Dual optimization problem 90
5.4 Margin theory 91
5.5 Chapter notes 100
5.6 Exercises 100

6 Kernel Methods 105

6.1 Introduction 105
6.2 Positive definite symmetric kernels 108
6.2.1 Definitions 108
6.2.2 Reproducing kernel Hilbert space 110
6.2.3 Properties 112
6.3 Kernel-based algorithms 116
6.3.1 SVMs with PDS kernels 116
6.3.2 Representer theorem 117
6.3.3 Learning guarantees 117
6.4 Negative definite symmetric kernels 119
6.5 Sequence kernels 121
6.5.1 Weighted transducers 122
6.5.2 Rational kernels 126
6.6 Approximate kernel feature maps 130
6.7 Chapter notes 135
6.8 Exercises 137

7 Boosting 145

7.1 Introduction 145
7.2 AdaBoost 146
7.2.1 Bound on the empirical error 149
7.2.2 Relationship with coordinate descent 150
7.2.3 Practical use 154

7.3 Theoretical results 154
7.3.1 VC-dimension-based analysis 154
7.3.2 L_1-geometric margin 155
7.3.3 Margin-based analysis 157
7.3.4 Margin maximization 161
7.3.5 Game-theoretic interpretation 162
7.4 L_1-regularization 165
7.5 Discussion 167
7.6 Chapter notes 168
7.7 Exercises 170

8 On-Line Learning 177
8.1 Introduction 178
8.2 Prediction with expert advice 178
8.2.1 Mistake bounds and Halving algorithm 179
8.2.2 Weighted majority algorithm 181
8.2.3 Randomized weighted majority algorithm 183
8.2.4 Exponential weighted average algorithm 186
8.3 Linear classification 190
8.3.1 Perceptron algorithm 190
8.3.2 Winnow algorithm 198
8.4 On-line to batch conversion 201
8.5 Game-theoretic connection 204
8.6 Chapter notes 205
8.7 Exercises 206

9 Multi-Class Classification 213
9.1 Multi-class classification problem 213
9.2 Generalization bounds 215
9.3 Uncombined multi-class algorithms 221
9.3.1 Multi-class SVMs 221
9.3.2 Multi-class boosting algorithms 222
9.3.3 Decision trees 224
9.4 Aggregated multi-class algorithms 228
9.4.1 One-versus-all 229
9.4.2 One-versus-one 229
9.4.3 Error-correcting output codes 231
9.5 Structured prediction algorithms 233
9.6 Chapter notes 235
9.7 Exercises 237

10 Ranking 239
10.1 The problem of ranking 240
10.2 Generalization bound 241
10.3 Ranking with SVMs 243

10.4 RankBoost 244
10.4.1 Bound on the empirical error 246
10.4.2 Relationship with coordinate descent 248
10.4.3 Margin bound for ensemble methods in ranking 250
10.5 Bipartite ranking 251
10.5.1 Boosting in bipartite ranking 252
10.5.2 Area under the ROC curve 255
10.6 Preference-based setting 257
10.6.1 Second-stage ranking problem 257
10.6.2 Deterministic algorithm 259
10.6.3 Randomized algorithm 260
10.6.4 Extension to other loss functions 262
10.7 Other ranking criteria 262
10.8 Chapter notes 263
10.9 Exercises 264

11 Regression 267

11.1 The problem of regression 267
11.2 Generalization bounds 268
11.2.1 Finite hypothesis sets 268
11.2.2 Rademacher complexity bounds 269
11.2.3 Pseudo-dimension bounds 271
11.3 Regression algorithms 275
11.3.1 Linear regression 275
11.3.2 Kernel ridge regression 276
11.3.3 Support vector regression 281
11.3.4 Lasso 285
11.3.5 Group norm regression algorithms 289
11.3.6 On-line regression algorithms 289
11.4 Chapter notes 290
11.5 Exercises 292

12 Maximum Entropy Models 295

12.1 Density estimation problem 295
12.1.1 Maximum Likelihood (ML) solution 296
12.1.2 Maximum a Posteriori (MAP) solution 297
12.2 Density estimation problem augmented with features 297
12.3 Maxent principle 298
12.4 Maxent models 299
12.5 Dual problem 299
12.6 Generalization bound 303
12.7 Coordinate descent algorithm 304
12.8 Extensions 306
12.9 L_2-regularization 308

12.10 Chapter notes 312
12.11 Exercises 313

13 Conditional Maximum Entropy Models 315

13.1 Learning problem 315
13.2 Conditional Maxent principle 316
13.3 Conditional Maxent models 316
13.4 Dual problem 317
13.5 Properties 319
13.5.1 Optimization problem 320
13.5.2 Feature vectors 320
13.5.3 Prediction 321
13.6 Generalization bounds 321
13.7 Logistic regression 325
13.7.1 Optimization problem 325
13.7.2 Logistic model 325
13.8 L_2-regularization 326
13.9 Proof of the duality theorem 328
13.10 Chapter notes 330
13.11 Exercises 331

14 Algorithmic Stability 333

14.1 Definitions 333
14.2 Stability-based generalization guarantee 334
14.3 Stability of kernel-based regularization algorithms 336
14.3.1 Application to regression algorithms: SVR and KRR 339
14.3.2 Application to classification algorithms: SVMs 341
14.3.3 Discussion 342
14.4 Chapter notes 342
14.5 Exercises 343

15 Dimensionality Reduction 347

15.1 Principal component analysis 348
15.2 Kernel principal component analysis (KPCA) 349
15.3 KPCA and manifold learning 351
15.3.1 Isomap 351
15.3.2 Laplacian eigenmaps 352
15.3.3 Locally linear embedding (LLE) 353
15.4 Johnson-Lindenstrauss lemma 354
15.5 Chapter notes 356
15.6 Exercises 356

16 Learning Automata and Languages 359

16.1 Introduction 359

16.2 Finite automata 360
16.3 Efficient exact learning 361
16.3.1 Passive learning 362
16.3.2 Learning with queries 363
16.3.3 Learning automata with queries 364
16.4 Identification in the limit 369
16.4.1 Learning reversible automata 370
16.5 Chapter notes 375
16.6 Exercises 376

17 Reinforcement Learning 379

17.1 Learning scenario 379
17.2 Markov decision process model 380
17.3 Policy 381
17.3.1 Definition 381
17.3.2 Policy value 382
17.3.3 Optimal policies 382
17.3.4 Policy evaluation 385
17.4 Planning algorithms 387
17.4.1 Value iteration 387
17.4.2 Policy iteration 390
17.4.3 Linear programming 392
17.5 Learning algorithms 393
17.5.1 Stochastic approximation 394
17.5.2 TD(0) algorithm 397
17.5.3 Q-learning algorithm 398
17.5.4 SARSA 402
17.5.5 TD(λ) algorithm 402
17.5.6 Large state space 403
17.6 Chapter notes 405

Conclusion 407

A Linear Algebra Review 409

A.1 Vectors and norms 409
A.1.1 Norms 409
A.1.2 Dual norms 410
A.1.3 Relationship between norms 411
A.2 Matrices 411
A.2.1 Matrix norms 411
A.2.2 Singular value decomposition 412
A.2.3 Symmetric positive semidefinite (SPSD) matrices 412

B Convex Optimization 415

B.1 Differentiation and unconstrained optimization 415
B.2 Convexity 415
B.3 Constrained optimization 419
B.4 Fenchel duality 422
B.4.1 Subgradients 422
B.4.2 Core 423
B.4.3 Conjugate functions 423
B.5 Chapter notes 426
B.6 Exercises 427

C Probability Review 429

C.1 Probability 429
C.2 Random variables 429
C.3 Conditional probability and independence 431
C.4 Expectation and Markov's inequality 431
C.5 Variance and Chebyshev's inequality 432
C.6 Moment-generating functions 434
C.7 Exercises 435

D Concentration Inequalities 437

D.1 Hoeffding's inequality 437
D.2 Sanov's theorem 438
D.3 Multiplicative Chernoff bounds 439
D.4 Binomial distribution tails: Upper bounds 440
D.5 Binomial distribution tails: Lower bound 440
D.6 Azuma's inequality 441
D.7 McDiarmid's inequality 442
D.8 Normal distribution tails: Lower bound 443
D.9 Khintchine-Kahane inequality 443
D.10 Maximal inequality 444
D.11 Chapter notes 445
D.12 Exercises 445

E Notions of Information Theory 449

E.1 Entropy 449
E.2 Relative entropy 450
E.3 Mutual information 453
E.4 Bregman divergences 453
E.5 Chapter notes 456
E.6 Exercises 457

F Notation 459

Bibliography 461

Index 475

Preface

This book is a general introduction to machine learning that can serve as a reference book for researchers and a textbook for students. It covers fundamental modern topics in machine learning while providing the theoretical basis and conceptual tools needed for the discussion and justification of algorithms. It also describes several key aspects of the application of these algorithms.

We have aimed to present the most novel theoretical tools and concepts while giving concise proofs, even for relatively advanced results. In general, whenever possible, we have chosen to favor succinctness. Nevertheless, we discuss some crucial complex topics arising in machine learning and highlight several open research questions. Certain topics often merged with others or treated with insufficient attention are discussed separately here and with more emphasis: for example, a different chapter is reserved for multi-class classification, ranking, and regression.

Although we cover a very wide variety of important topics in machine learning, we have chosen to omit a few important ones, including graphical models and neural networks, both for the sake of brevity and because of the current lack of solid theoretical guarantees for some methods.

The book is intended for students and researchers in machine learning, statistics and other related areas. It can be used as a textbook for both graduate and advanced undergraduate classes in machine learning or as a reference text for a research seminar. The first three or four chapters of the book lay the theoretical foundation for the subsequent material. Other chapters are mostly self-contained, with the exception of chapter 6 which introduces some concepts that are extensively used in later ones and chapter 13, which is closely related to chapter 12. Each chapter concludes with a series of exercises, with full solutions presented separately.

The reader is assumed to be familiar with basic concepts in linear algebra, probability, and analysis of algorithms. However, to further help, we have included an extensive appendix presenting a concise review of linear algebra, an introduction to convex optimization, a brief probability review, a collection of concentration inequalities useful to the analyses and discussions in this book, and a short introduction to information theory.

Our goal has been to give a unified presentation of multiple topics and areas, as opposed to a more specialized presentation adopted by some books which favor a particular viewpoint, such as for example a Bayesian view, or a particular topic, such as for example kernel methods. The theoretical foundation of this book and its deliberate emphasis on proofs and analysis make it also very distinct from many other presentations.

In this second edition, we have updated the entire book. The changes include a different writing style in most chapters, new figures and illustrations, many simplifications, some additions to existing chapters, in particular chapter 6 and chapter 17, and several new chapters. We have added a full chapter on model selection (chapter 4), which is an important topic that was only briefly discussed in the previous edition. We have also added a new chapter on Maximum Entropy models (chapter 12) and a new chapter on Conditional Maximum Entropy models (chapter 13) which are both essential topics in machine learning. We have also significantly changed the appendix. In particular, we have added a full section on Fenchel duality to appendix B on convex optimization, made a number of changes and additions to appendix D dealing with concentration inequalities, added appendix E on information theory, and updated most of the material. Additionally, we have included a number of new exercises and their solutions for existing and new chapters.

Most of the material presented here takes its origins in a machine learning graduate course (*Foundations of Machine Learning*) taught by the first author at the Courant Institute of Mathematical Sciences in New York University over the last fourteen years. This book has considerably benefited from the comments and suggestions from students in these classes, along with those of many friends, colleagues and researchers to whom we are deeply indebted.

We are particularly grateful to Corinna Cortes and Yishay Mansour who made a number of key suggestions for the design and organization of the material presented in the first edition, with detailed comments that we have fully taken into account and that have greatly improved the presentation. We are also grateful to Yishay Mansour for using a preliminary version of the first edition of the book for teaching, and for reporting his feedback to us.

We also thank for discussions, suggested improvement, and contributions of many kinds the following colleagues and friends from academic and corporate research laboratories: Jacob Abernethy, Cyril Allauzen, Kareem Amin, Stephen Boyd, Aldo Corbisiero, Giulia DeSalvo, Claudio Gentile, Spencer Greenberg, Lisa Hellerstein, Sanjiv Kumar, Vitaly Kuznetsov, Ryan McDonald, Andrès Muñoz Medina, Tyler Neylon, Peter Norvig, Fernando Pereira, Maria Pershina, Borja de Balle Pigem,

Ashish Rastogi, Michael Riley, Dmitry Storcheus, Ananda Theertha Suresh, Umar Syed, Csaba Szepesvári, Toshiyuki Tanaka, Eugene Weinstein, Jason Weston, Scott Yang, and Ningshan Zhang.

Finally, we thank the MIT Press publication team for their help and support in the development of this text.

1 Introduction

This chapter presents a preliminary introduction to machine learning, including an overview of some key learning tasks and applications, basic definitions and terminology, and the discussion of some general scenarios.

1.1 What is machine learning?

Machine learning can be broadly defined as computational methods using experience to improve performance or to make accurate predictions. Here, *experience* refers to the past information available to the learner, which typically takes the form of electronic data collected and made available for analysis. This data could be in the form of digitized human-labeled training sets, or other types of information obtained via interaction with the environment. In all cases, its quality and size are crucial to the success of the predictions made by the learner.

An example of a learning problem is how to use a finite sample of randomly selected documents, each labeled with a topic, to accurately predict the topic of unseen documents. Clearly, the larger is the sample, the easier is the task. But the difficulty of the task also depends on the quality of the labels assigned to the documents in the sample, since the labels may not be all correct, and on the number of possible topics.

Machine learning consists of designing efficient and accurate prediction *algorithms*. As in other areas of computer science, some critical measures of the quality of these algorithms are their time and space complexity. But, in machine learning, we will need additionally a notion of *sample complexity* to evaluate the sample size required for the algorithm to learn a family of concepts. More generally, theoretical learning guarantees for an algorithm depend on the complexity of the concept classes considered and the size of the training sample.

Since the success of a learning algorithm depends on the data used, machine learning is inherently related to data analysis and statistics. More generally, learning techniques are data-driven methods combining fundamental concepts in computer science with ideas from statistics, probability and optimization.

1.2 What kind of problems can be tackled using machine learning?

Predicting the label of a document, also known as document classification, is by no means the only learning task. Machine learning admits a very broad set of practical applications, which include the following:

- Text or document classification. This includes problems such as assigning a topic to a text or a document, or determining automatically if the content of a web page is inappropriate or too explicit; it also includes spam detection.
- Natural language processing (NLP). Most tasks in this field, including part-of-speech tagging, named-entity recognition, context-free parsing, or dependency parsing, are cast as learning problems. In these problems, predictions admit some structure. For example, in part-of-speech tagging, the prediction for a sentence is a sequence of part-of-speech tags labeling each word. In context-free parsing the prediction is a tree. These are instances of richer learning problems known as *structured prediction problems*.
- Speech processing applications. This includes speech recognition, speech synthesis, speaker verification, speaker identification, as well as sub-problems such as language modeling and acoustic modeling.
- Computer vision applications. This includes object recognition, object identification, face detection, Optical character recognition (OCR), content-based image retrieval, or pose estimation.
- Computational biology applications. This includes protein function prediction, identification of key sites, or the analysis of gene and protein networks.
- Many other problems such as fraud detection for credit card, telephone or insurance companies, network intrusion, learning to play games such as chess, backgammon, or Go, unassisted control of vehicles such as robots or cars, medical diagnosis, the design of recommendation systems, search engines, or information extraction systems, are tackled using machine learning techniques.

This list is by no means comprehensive. Most prediction problems found in practice can be cast as learning problems and the practical application area of machine learning keeps expanding. The algorithms and techniques discussed in this book can be used to derive solutions for all of these problems, though we will not discuss in detail these applications.

1.3 Some standard learning tasks

The following are some standard machine learning tasks that have been extensively studied:

- *Classification*: this is the problem of assigning a category to each item. For example, document classification consists of assigning a category such as *politics*, *business*, *sports*, or *weather* to each document, while image classification consists of assigning to each image a category such as *car*, *train*, or *plane*. The number of categories in such tasks is often less than a few hundreds, but it can be much larger in some difficult tasks and even unbounded as in OCR, text classification, or speech recognition.
- *Regression*: this is the problem of predicting a real value for each item. Examples of regression include prediction of stock values or that of variations of economic variables. In regression, the penalty for an incorrect prediction depends on the magnitude of the difference between the true and predicted values, in contrast with the classification problem, where there is typically no notion of closeness between various categories.
- *Ranking*: this is the problem of learning to order items according to some criterion. Web search, e.g., returning web pages relevant to a search query, is the canonical ranking example. Many other similar ranking problems arise in the context of the design of information extraction or natural language processing systems.
- *Clustering*: this is the problem of partitioning a set of items into homogeneous subsets. Clustering is often used to analyze very large data sets. For example, in the context of social network analysis, clustering algorithms attempt to identify natural *communities* within large groups of people.
- *Dimensionality reduction* or *manifold learning*: this problem consists of transforming an initial representation of items into a lower-dimensional representation while preserving some properties of the initial representation. A common example involves preprocessing digital images in computer vision tasks.

The main practical objectives of machine learning consist of generating accurate predictions for unseen items and of designing efficient and robust algorithms to produce these predictions, even for large-scale problems. To do so, a number of algorithmic and theoretical questions arise. Some fundamental questions include: Which concept families can actually be learned, and under what conditions? How well can these concepts be learned computationally?

1.4 Learning stages

Here, we will use the canonical problem of spam detection as a running example to illustrate some basic definitions and describe the use and evaluation of machine learning algorithms in practice, including their different stages.

Spam detection is the problem of learning to automatically classify email messages as either SPAM or non-SPAM. The following is a list of definitions and terminology commonly used in machine learning:

- *Examples*: Items or instances of data used for learning or evaluation. In our spam problem, these examples correspond to the collection of email messages we will use for learning and testing.
- *Features*: The set of attributes, often represented as a vector, associated to an example. In the case of email messages, some relevant features may include the length of the message, the name of the sender, various characteristics of the header, the presence of certain keywords in the body of the message, and so on.
- *Labels*: Values or categories assigned to examples. In classification problems, examples are assigned specific categories, for instance, the SPAM and non-SPAM categories in our binary classification problem. In regression, items are assigned real-valued labels.
- *Hyperparameters*: Free parameters that are not determined by the learning algorithm, but rather specified as inputs to the learning algorithm.
- *Training sample*: Examples used to train a learning algorithm. In our spam problem, the training sample consists of a set of email examples along with their associated labels. The training sample varies for different learning scenarios, as described in section 1.5.
- *Validation sample*: Examples used to tune the parameters of a learning algorithm when working with labeled data. The validation sample is used to select appropriate values for the learning algorithm's free parameters (hyperparameters).
- *Test sample*: Examples used to evaluate the performance of a learning algorithm. The test sample is separate from the training and validation data and is not made available in the learning stage. In the spam problem, the test sample consists of a collection of email examples for which the learning algorithm must predict labels based on features. These predictions are then compared with the labels of the test sample to measure the performance of the algorithm.
- *Loss function*: A function that measures the difference, or loss, between a predicted label and a true label. Denoting the set of all labels as $\mathcal{Y}$ and the set of possible predictions as $\mathcal{Y}'$, a loss function L is a mapping $L\colon \mathcal{Y} \times \mathcal{Y}' \to \mathbb{R}_+$. In most cases, $\mathcal{Y}' = \mathcal{Y}$ and the loss function is bounded, but these conditions do not always hold. Common examples of loss functions include the zero-one (or

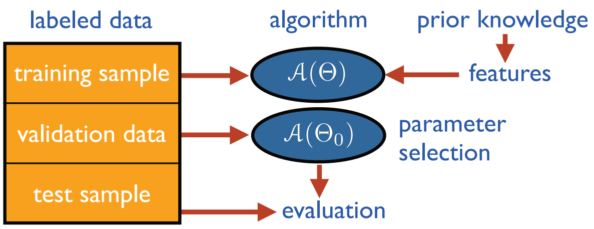


Figure 1.1
Illustration of the typical stages of a learning process.

misclassification) loss defined over $\{-1, +1\} \times \{-1, +1\}$ by $L(y, y') = 1_{y' \neq y}$ and the squared loss defined over $\mathcal{I} \times \mathcal{I}$ by $L(y, y') = (y' - y)^2$, where $\mathcal{I} \subseteq \mathbb{R}$ is typically a bounded interval.

- *Hypothesis set*: A set of functions mapping features (feature vectors) to the set of labels $\mathcal{Y}$. In our example, these may be a set of functions mapping email features to $\mathcal{Y} = \{\text{SPAM}, \text{non-SPAM}\}$. More generally, hypotheses may be functions mapping features to a different set $\mathcal{Y}'$. They could be linear functions mapping email feature vectors to real numbers interpreted as *scores* ($\mathcal{Y}' = \mathbb{R}$), with higher score values more indicative of SPAM than lower ones.

We now define the learning stages of our spam problem (see figure 1.1). We start with a given collection of labeled examples. We first randomly partition the data into a training sample, a validation sample, and a test sample. The size of each of these samples depends on a number of different considerations. For example, the amount of data reserved for validation depends on the number of hyperparameters of the algorithm, which are represented here by the vector Θ. Also, when the labeled sample is relatively small, the amount of training data is often chosen to be larger than that of the test data since the learning performance directly depends on the training sample.

Next, we associate relevant features to the examples. This is a critical step in the design of machine learning solutions. Useful features can effectively guide the learning algorithm, while poor or uninformative ones can be misleading. Although it is critical, to a large extent, the choice of the features is left to the user. This choice reflects the user's *prior knowledge* about the learning task which in practice can have a dramatic effect on the performance results.

Now, we use the features selected to train our learning algorithm $\mathcal{A}$ by tuning the values of its free parameters Θ (also called *hyperparameters*). For each value of these parameters, the algorithm selects a different hypothesis out of the hypothesis set. We choose the one resulting in the best performance on the validation sample (Θ_0). Finally, using that hypothesis, we predict the labels of the examples in the test sample. The performance of the algorithm is evaluated by using the loss function associated to the task, e.g., the zero-one loss in our spam detection task, to compare the predicted and true labels. Thus, the performance of an algorithm is of course evaluated based on its test error and not its error on the training sample.

1.5 Learning scenarios

We next briefly describe some common machine learning scenarios. These scenarios differ in the types of training data available to the learner, the order and method by which training data is received and the test data used to evaluate the learning algorithm.

- *Supervised learning*: The learner receives a set of labeled examples as training data and makes predictions for all unseen points. This is the most common scenario associated with classification, regression, and ranking problems. The spam detection problem discussed in the previous section is an instance of supervised learning.
- *Unsupervised learning*: The learner exclusively receives unlabeled training data, and makes predictions for all unseen points. Since in general no labeled example is available in that setting, it can be difficult to quantitatively evaluate the performance of a learner. Clustering and dimensionality reduction are example of unsupervised learning problems.
- *Semi-supervised learning*: The learner receives a training sample consisting of both labeled and unlabeled data, and makes predictions for all unseen points. Semi-supervised learning is common in settings where unlabeled data is easily accessible but labels are expensive to obtain. Various types of problems arising in applications, including classification, regression, or ranking tasks, can be framed as instances of semi-supervised learning. The hope is that the distribution of unlabeled data accessible to the learner can help him achieve a better performance than in the supervised setting. The analysis of the conditions under which this can indeed be realized is the topic of much modern theoretical and applied machine learning research.
- *Transductive inference*: As in the semi-supervised scenario, the learner receives a labeled training sample along with a set of unlabeled test points. However, the objective of transductive inference is to predict labels only for these particular test points. Transductive inference appears to be an easier task and matches the scenario encountered in a variety of modern applications. However, as in the semi-supervised setting, the assumptions under which a better performance can be achieved in this setting are research questions that have not been fully resolved.

- *On-line learning*: In contrast with the previous scenarios, the online scenario involves multiple rounds where training and testing phases are intermixed. At each round, the learner receives an unlabeled training point, makes a prediction, receives the true label, and incurs a loss. The objective in the on-line setting is to minimize the cumulative loss over all rounds or to minimize the *regret*, that is the difference of the cumulative loss incurred and that of the best expert in hindsight. Unlike the previous settings just discussed, no distributional assumption is made in on-line learning. In fact, instances and their labels may be chosen adversarially within this scenario.
- *Reinforcement learning*: The training and testing phases are also intermixed in reinforcement learning. To collect information, the learner actively interacts with the environment and in some cases affects the environment, and receives an immediate reward for each action. The object of the learner is to maximize his reward over a course of actions and iterations with the environment. However, no long-term reward feedback is provided by the environment, and the learner is faced with the *exploration versus exploitation* dilemma, since he must choose between exploring unknown actions to gain more information versus exploiting the information already collected.
- *Active learning*: The learner adaptively or interactively collects training examples, typically by querying an oracle to request labels for new points. The goal in active learning is to achieve a performance comparable to the standard supervised learning scenario (or *passive learning* scenario), but with fewer labeled examples. Active learning is often used in applications where labels are expensive to obtain, for example computational biology applications.

In practice, many other intermediate and somewhat more complex learning scenarios may be encountered.

1.6 Generalization

Machine learning is fundamentally about *generalization*. As an example, the standard supervised learning scenario consists of using a finite sample of labeled examples to make accurate predictions about unseen examples. The problem is typically formulated as that of selecting a function out of a *hypothesis set*, that is a subset of the family of all functions. The function selected is subsequently used to label all instances, including unseen examples.

How should a hypothesis set be chosen? With a rich or *complex* hypothesis set, the learner may choose a function or predictor that is *consistent* with the training sample, that is one that commits no error on the training sample. With a less complex family, incurring some errors on the training sample may be unavoidable. But,

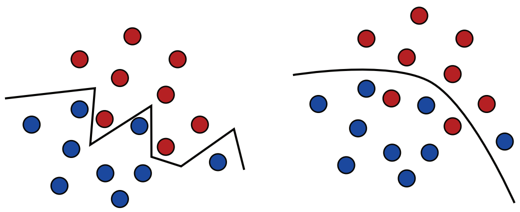


Figure 1.2
The zig-zag line on the left panel is consistent over the blue and red training sample, but it is a complex separation surface that is not likely to generalize well to unseen data. In contrast, the decision surface on the right panel is simpler and might generalize better in spite of its misclassification of a few points of the training sample.

which will lead to a better generalization? How should we define the complexity of a hypothesis set?

Figure 1.2 illustrates these two types of solution: one is a zig-zag line that perfectly separates the two populations of blue and red points and that is chosen from a complex family; the other one is a smoother line chosen from a simpler family that only imperfectly discriminates between the two sets. We will see that, in general, the best predictor on the training sample may not be the best overall. A predictor chosen from a very complex family can essentially memorize the data, but generalization is distinct from the memorization of the training labels.

We will see that the trade-off between the sample size and complexity plays a critical role in generalization. When the sample size is relatively small, choosing from a too complex a family may lead to poor generalization, which is also known as *overfitting*. On the other hand, with a too simple a family it may not be possible to achieve a sufficient accuracy, which is known as *underfitting*.

In the next chapters, we will analyze more in detail the problem of generalization and will seek to derive theoretical guarantees for learning. This will depend on different notions of complexity that we will thoroughly discuss.

2 The PAC Learning Framework

Several fundamental questions arise when designing and analyzing algorithms that learn from examples: What can be learned efficiently? What is inherently hard to learn? How many examples are needed to learn successfully? Is there a general model of learning? In this chapter, we begin to formalize and address these questions by introducing the *Probably Approximately Correct* (PAC) learning framework. The PAC framework helps define the class of learnable concepts in terms of the number of sample points needed to achieve an approximate solution, *sample complexity*, and the time and space complexity of the learning algorithm, which depends on the cost of the computational representation of the concepts.

We first describe the PAC framework and illustrate it, then present some general learning guarantees within this framework when the hypothesis set used is finite, both for the *consistent* case where the hypothesis set used contains the concept to learn and for the opposite *inconsistent* case.

2.1 The PAC learning model

We first introduce several definitions and the notation needed to present the PAC model, which will also be used throughout much of this book.

We denote by $\mathcal{X}$ the set of all possible *examples* or *instances*. $\mathcal{X}$ is also sometimes referred to as the *input space*. The set of all possible *labels* or *target values* is denoted by $\mathcal{Y}$. For the purpose of this introductory chapter, we will limit ourselves to the case where $\mathcal{Y}$ is reduced to two labels, $\mathcal{Y} = \{0, 1\}$, which corresponds to the so-called *binary classification*. Later chapters will extend these results to more general settings.

A *concept* $c\colon \mathcal{X} \to \mathcal{Y}$ is a mapping from $\mathcal{X}$ to $\mathcal{Y}$. Since $\mathcal{Y} = \{0, 1\}$, we can identify c with the subset of $\mathcal{X}$ over which it takes the value 1. Thus, in the following, we equivalently refer to a concept to learn as a mapping from $\mathcal{X}$ to $\{0, 1\}$, or as a subset of $\mathcal{X}$. As an example, a concept may be the set of points inside a triangle or the indicator function of these points. In such cases, we will say in short that the concept to learn is a triangle. A *concept class* is a set of concepts we may wish to learn and is denoted by $\mathcal{C}$. This could, for example, be the set of all triangles in the plane.

We assume that examples are independently and identically distributed (i.i.d.) according to some fixed but unknown distribution $\mathcal{D}$. The learning problem is then formulated as follows. The learner considers a fixed set of possible concepts $\mathcal{H}$, called a *hypothesis set*, which might not necessarily coincide with $\mathcal{C}$. It receives a sample $S = (x_1, \ldots, x_m)$ drawn i.i.d. according to $\mathcal{D}$ as well as the labels $(c(x_1), \ldots, c(x_m))$, which are based on a specific target concept $c \in \mathcal{C}$ to learn. The task is then to use the labeled sample S to select a hypothesis $h_S \in \mathcal{H}$ that has a small *generalization error* with respect to the concept c. The generalization error of a hypothesis $h \in \mathcal{H}$, also referred to as the *risk* or *true error* (or simply *error*) of h is denoted by $R(h)$ and defined as follows.[1]

Definition 2.1 (Generalization error) *Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and an underlying distribution $\mathcal{D}$, the* generalization error *or* risk *of h is defined by*

$$R(h) = \underset{x \sim \mathcal{D}}{\mathbb{P}}[h(x) \neq c(x)] = \underset{x \sim \mathcal{D}}{\mathbb{E}}\left[1_{h(x) \neq c(x)}\right], \tag{2.1}$$

where 1_ω is the indicator function of the event ω.[2]

The generalization error of a hypothesis is not directly accessible to the learner since both the distribution $\mathcal{D}$ and the target concept c are unknown. However, the learner can measure the *empirical error* of a hypothesis on the labeled sample S.

Definition 2.2 (Empirical error) *Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and a sample $S = (x_1, \ldots, x_m)$, the* empirical error *or* empirical risk *of h is defined by*

$$\widehat{R}_S(h) = \frac{1}{m} \sum_{i=1}^{m} 1_{h(x_i) \neq c(x_i)}. \tag{2.2}$$

Thus, the empirical error of $h \in \mathcal{H}$ is its average error over the sample S, while the generalization error is its expected error based on the distribution $\mathcal{D}$. We will see in this chapter and the following chapters a number of guarantees relating these two quantities with high probability, under some general assumptions. We can already note that for a fixed $h \in \mathcal{H}$, the expectation of the empirical error based on an i.i.d.

[1] The choice of R instead of E to denote an error avoids possible confusions with the notation for expectations and is further justified by the fact that the term *risk* is also used in machine learning and statistics to refer to an error.

[2] For this and other related definitions, the family of functions $\mathcal{H}$ and the target concept c must be measurable. The function classes we consider in this book all have this property.

sample S is equal to the generalization error:

$$\underset{S \sim \mathcal{D}^m}{\mathbb{E}}[\widehat{R}_S(h)] = R(h). \tag{2.3}$$

Indeed, by the linearity of the expectation and the fact that the sample is drawn i.i.d., we can write

$$\underset{S \sim \mathcal{D}^m}{\mathbb{E}}[\widehat{R}_S(h)] = \frac{1}{m} \sum_{i=1}^{m} \underset{S \sim \mathcal{D}^m}{\mathbb{E}}[1_{h(x_i) \neq c(x_i)}] = \frac{1}{m} \sum_{i=1}^{m} \underset{S \sim \mathcal{D}^m}{\mathbb{E}}[1_{h(x) \neq c(x)}],$$

for any x in sample S. Thus,

$$\underset{S \sim \mathcal{D}^m}{\mathbb{E}}[\widehat{R}_S(h)] = \underset{S \sim \mathcal{D}^m}{\mathbb{E}}[1_{h(x) \neq c(x)}] = \underset{x \sim \mathcal{D}}{\mathbb{E}}[1_{h(x) \neq c(x)}] = R(h).$$

The following introduces the *Probably Approximately Correct* (PAC) learning framework. Let n be a number such that the computational cost of representing any element $x \in \mathcal{X}$ is at most $O(n)$ and denote by size(c) the maximal cost of the computational representation of $c \in \mathcal{C}$. For example, x may be a vector in $\mathbb{R}^n$, for which the cost of an array-based representation would be in $O(n)$. In addition, let h_S denote the hypothesis returned by algorithm $\mathcal{A}$ after receiving a labeled sample S. To keep notation simple, the dependency of h_S on $\mathcal{A}$ is not explicitly indicated.

Definition 2.3 (PAC-learning) *A concept class $\mathcal{C}$ is said to be* PAC-learnable *if there exists an algorithm $\mathcal{A}$ and a polynomial function poly$(\cdot,\cdot,\cdot,\cdot)$ such that for any $\epsilon > 0$ and $\delta > 0$, for all distributions $\mathcal{D}$ on $\mathcal{X}$ and for any target concept $c \in \mathcal{C}$, the following holds for any sample size $m \geq$ poly$(1/\epsilon, 1/\delta, n, $ size$(c))$:*

$$\underset{S \sim \mathcal{D}^m}{\mathbb{P}}[R(h_S) \leq \epsilon] \geq 1 - \delta. \tag{2.4}$$

If $\mathcal{A}$ further runs in poly$(1/\epsilon, 1/\delta, n, $ size$(c))$, then $\mathcal{C}$ is said to be efficiently PAC-learnable. *When such an algorithm $\mathcal{A}$ exists, it is called a* PAC-learning algorithm *for $\mathcal{C}$.*

A concept class $\mathcal{C}$ is thus PAC-learnable if the hypothesis returned by the algorithm after observing a number of points polynomial in $1/\epsilon$ and $1/\delta$ is *approximately correct* (error at most ϵ) with high *probability* (at least $1 - \delta$), which justifies the PAC terminology. The parameter $\delta > 0$ is used to define the *confidence* $1 - \delta$ and $\epsilon > 0$ the *accuracy* $1 - \epsilon$. Note that if the running time of the algorithm is polynomial in $1/\epsilon$ and $1/\delta$, then the sample size m must also be polynomial if the full sample is received by the algorithm.

Several key points of the PAC definition are worth emphasizing. First, the PAC framework is a *distribution-free model*: no particular assumption is made about the distribution $\mathcal{D}$ from which examples are drawn. Second, the training sample and the test examples used to define the error are drawn according to the same distribution $\mathcal{D}$. This is a natural and necessary assumption for generalization to

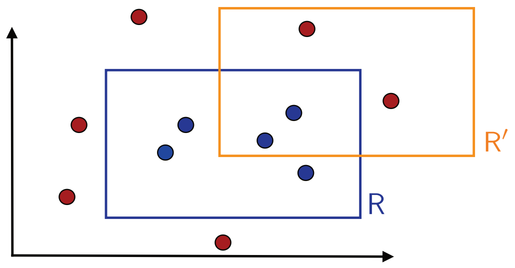


Figure 2.1
Target concept R and possible hypothesis R'. Circles represent training instances. A blue circle is a point labeled with 1, since it falls within the rectangle R. Others are red and labeled with 0.

be possible in general. It can be relaxed to include favorable *domain adaptation* problems. Finally, the PAC framework deals with the question of learnability for a concept class $\mathcal{C}$ and not a particular concept. Note that the concept class $\mathcal{C}$ is known to the algorithm, but of course the target concept $c \in \mathcal{C}$ is unknown.

In many cases, in particular when the computational representation of the concepts is not explicitly discussed or is straightforward, we may omit the polynomial dependency on n and size(c) in the PAC definition and focus only on the sample complexity.

We now illustrate PAC-learning with a specific learning problem.

Example 2.4 (Learning axis-aligned rectangles) Consider the case where the set of instances are points in the plane, $\mathcal{X} = \mathbb{R}^2$, and the concept class $\mathcal{C}$ is the set of all axis-aligned rectangles lying in $\mathbb{R}^2$. Thus, each concept c is the set of points inside a particular axis-aligned rectangle. The learning problem consists of determining with small error a target axis-aligned rectangle using the labeled training sample. We will show that the concept class of axis-aligned rectangles is PAC-learnable.

Figure 2.1 illustrates the problem. R represents a target axis-aligned rectangle and R' a hypothesis. As can be seen from the figure, the error regions of R' are formed by the area within the rectangle R but outside the rectangle R' and the area within R' but outside the rectangle R. The first area corresponds to *false negatives*, that is, points that are labeled as 0 or *negatively* by R', which are in fact *positive* or labeled with 1. The second area corresponds to *false positives*, that is, points labeled positively by R' which are in fact negatively labeled.

To show that the concept class is PAC-learnable, we describe a simple PAC-learning algorithm $\mathcal{A}$. Given a labeled sample S, the algorithm consists of returning the tightest axis-aligned rectangle $\mathsf{R}' = \mathsf{R}_S$ containing the points labeled with 1. Figure 2.2 illustrates the hypothesis returned by the algorithm. By definition, R_S does not produce any false positives, since its points must be included in the target concept R. Thus, the error region of R_S is included in R.

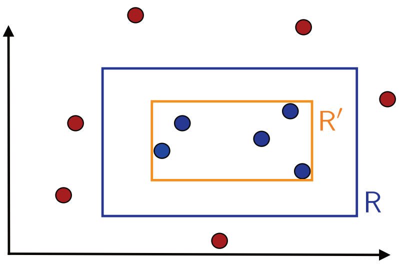


Figure 2.2
Illustration of the hypothesis $\mathsf{R}' = \mathsf{R}_\mathsf{S}$ returned by the algorithm.

Let $\mathsf{R} \in \mathcal{C}$ be a target concept. Fix $\epsilon > 0$. Let $\mathbb{P}[\mathsf{R}]$ denote the probability mass of the region defined by R, that is the probability that a point randomly drawn according to $\mathcal{D}$ falls within R. Since errors made by our algorithm can be due only to points falling inside R, we can assume that $\mathbb{P}[\mathsf{R}] > \epsilon$; otherwise, the error of R_S is less than or equal to ϵ regardless of the training sample S received.

Now, since $\mathbb{P}[\mathsf{R}] > \epsilon$, we can define four rectangular regions r_1, r_2, r_3, and r_4 along the sides of R, each with probability at least $\epsilon/4$. These regions can be constructed by starting with the full rectangle R and then decreasing the size by moving one side as much as possible while keeping a distribution mass of at least $\epsilon/4$. Figure 2.3 illustrates the definition of these regions.

Let l, r, b, and t be the four real values defining R: $\mathsf{R} = [l, r] \times [b, t]$. Then, for example, the left rectangle r_4 is defined by $r_4 = [l, s_4] \times [b, t]$, with $s_4 = \inf\{s \colon \mathbb{P}[[l, s] \times [b, t]] \geq \epsilon/4\}$. It is not hard to see that the probability of the region $\overline{r}_4 = [l, s_4[\times [b, t]$ obtained from r_4 by excluding the rightmost side is at most $\epsilon/4$. r_1, r_2, r_3 and $\overline{r}_1$, $\overline{r}_2$, $\overline{r}_3$ are defined in a similar way.

Observe that if R_S meets all of these four regions r_i, $i \in [4]$, then, because it is a rectangle, it will have one side in each of these regions (geometric argument). Its error area, which is the part of R that it does not cover, is thus included in the union of the regions $\overline{r}_i$, $i \in [4]$, and cannot have probability mass more than ϵ. By contraposition, if $R(\mathsf{R}_\mathsf{S}) > \epsilon$, then R_S must miss at least one of the regions r_i, $i \in [4]$. As a result, we can write

$$\begin{aligned}
\underset{S\sim\mathcal{D}^m}{\mathbb{P}}[R(\mathsf{R}_\mathsf{S}) > \epsilon] &\leq \underset{S\sim\mathcal{D}^m}{\mathbb{P}}[\cup_{i=1}^4 \{\mathsf{R}_\mathsf{S} \cap r_i = \emptyset\}] && (2.5)\\
&\leq \sum_{i=1}^{4} \underset{S\sim\mathcal{D}^m}{\mathbb{P}}[\{\mathsf{R}_\mathsf{S} \cap r_i = \emptyset\}] && \text{(by the union bound)}\\
&\leq 4(1-\epsilon/4)^m && \text{(since } \mathbb{P}[r_i] \geq \epsilon/4)\\
&\leq 4\exp(-m\epsilon/4),
\end{aligned}$$

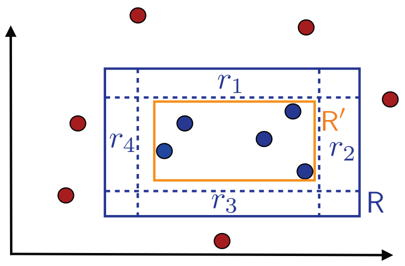


Figure 2.3
Illustration of the regions $r_1, \ldots, r_4$.

where for the last step we used the general inequality $1 - x \leq e^{-x}$ valid for all $x \in \mathbb{R}$. For any $\delta > 0$, to ensure that $\mathbb{P}_{S\sim\mathcal{D}^m}[R(\mathsf{R_S}) > \epsilon] \leq \delta$, we can impose

$$4\exp(-\epsilon m/4) \leq \delta \Leftrightarrow m \geq \frac{4}{\epsilon} \log \frac{4}{\delta}. \tag{2.6}$$

Thus, for any $\epsilon > 0$ and $\delta > 0$, if the sample size m is greater than $\frac{4}{\epsilon} \log \frac{4}{\delta}$, then $\mathbb{P}_{S\sim\mathcal{D}^m}[R(\mathsf{R_S}) > \epsilon] \leq \delta$. Furthermore, the computational cost of the representation of points in $\mathbb{R}^2$ and axis-aligned rectangles, which can be defined by their four corners, is constant. This proves that the concept class of axis-aligned rectangles is PAC-learnable and that the sample complexity of PAC-learning axis-aligned rectangles is in $O(\frac{1}{\epsilon} \log \frac{1}{\delta})$.

An equivalent way to present sample complexity results like (2.6), which we will often see throughout this book, is to give a *generalization bound*. A generalization bound states that with probability at least $1 - \delta$, $R(\mathsf{R_S})$ is upper bounded by some quantity that depends on the sample size m and δ. To obtain this, it suffices to set δ to be equal to the upper bound derived in (2.5), that is $\delta = 4\exp(-m\epsilon/4)$ and solve for ϵ. This yields that with probability at least $1 - \delta$, the error of the algorithm is bounded as follows:

$$R(\mathsf{R_S}) \leq \frac{4}{m} \log \frac{4}{\delta}. \tag{2.7}$$

Other PAC-learning algorithms could be considered for this example. One alternative is to return the largest axis-aligned rectangle not containing the negative points, for example. The proof of PAC-learning just presented for the tightest axis-aligned rectangle can be easily adapted to the analysis of other such algorithms.

Note that the hypothesis set $\mathcal{H}$ we considered in this example coincided with the concept class $\mathcal{C}$ and that its cardinality was infinite. Nevertheless, the problem admitted a simple proof of PAC-learning. We may then ask if a similar proof can readily apply to other similar concept classes. This is not as straightforward because the specific geometric argument used in the proof is key. It is non-trivial to extend the proof to other concept classes such as that of non-concentric circles (see exercise 2.4). Thus, we need a more general proof technique and more general results. The next two sections provide us with such tools in the case of a finite hypothesis set.

2.2 Guarantees for finite hypothesis sets — consistent case

In the example of axis-aligned rectangles that we examined, the hypothesis h_S returned by the algorithm was always *consistent*, that is, it admitted no error on the training sample S. In this section, we present a general sample complexity bound, or equivalently, a generalization bound, for consistent hypotheses, in the case where the cardinality $|\mathcal{H}|$ of the hypothesis set is finite. Since we consider consistent hypotheses, we will assume that the target concept c is in $\mathcal{H}$.

Theorem 2.5 (Learning bound — finite $\mathcal{H}$, consistent case) *Let $\mathcal{H}$ be a finite set of functions mapping from $\mathcal{X}$ to $\mathcal{Y}$. Let $\mathcal{A}$ be an algorithm that for any target concept $c \in \mathcal{H}$ and i.i.d. sample S returns a consistent hypothesis h_S: $\widehat{R}_S(h_S) = 0$. Then, for any $\epsilon, \delta > 0$, the inequality $\mathbb{P}_{S \sim \mathcal{D}^m}[R(h_S) \leq \epsilon] \geq 1 - \delta$ holds if*

$$m \geq \frac{1}{\epsilon}\Big(\log|\mathcal{H}| + \log\frac{1}{\delta}\Big). \tag{2.8}$$

This sample complexity result admits the following equivalent statement as a generalization bound: for any $\epsilon, \delta > 0$, with probability at least $1 - \delta$,

$$R(h_S) \leq \frac{1}{m}\Big(\log|\mathcal{H}| + \log\frac{1}{\delta}\Big). \tag{2.9}$$

Proof: Fix $\epsilon > 0$. We do not know which consistent hypothesis $h_S \in \mathcal{H}$ is selected by the algorithm $\mathcal{A}$. This hypothesis further depends on the training sample S. Therefore, we need to give a *uniform convergence bound*, that is, a bound that holds for the set of all consistent hypotheses, which a fortiori includes h_S. Thus, we will bound the probability that some $h \in \mathcal{H}$ would be consistent and have error more than ϵ. For any $\epsilon > 0$, define $\mathcal{H}_\epsilon$ by $\mathcal{H}_\epsilon = \{h \in \mathcal{H} \colon R(h) > \epsilon\}$. The probability that a hypothesis h in $\mathcal{H}_\epsilon$ is consistent on a training sample S drawn i.i.d., that is, that it would have no error on any point in S, can be bounded as follows:

$$\mathbb{P}[\widehat{R}_S(h) = 0] \leq (1-\epsilon)^m.$$

Thus, by the union bound, the following holds:

$$\begin{aligned}
\mathbb{P}\Big[\exists h \in \mathcal{H}_\epsilon \colon \widehat{R}_S(h) = 0\Big] &= \mathbb{P}\Big[\widehat{R}_S(h_1) = 0 \vee \cdots \vee \widehat{R}_S(h_{|\mathcal{H}_\epsilon|}) = 0\Big] \\
&\leq \sum_{h \in \mathcal{H}_\epsilon} \mathbb{P}\Big[\widehat{R}_S(h) = 0\Big] && \text{(union bound)} \\
&\leq \sum_{h \in \mathcal{H}_\epsilon} (1-\epsilon)^m \leq |\mathcal{H}|(1-\epsilon)^m \leq |\mathcal{H}|e^{-m\epsilon}.
\end{aligned}$$

Setting the right-hand side to be equal to δ and solving for ϵ concludes the proof. □

The theorem shows that when the hypothesis set $\mathcal{H}$ is finite, a consistent algorithm $\mathcal{A}$ is a PAC-learning algorithm, since the sample complexity given by (2.8) is dominated by a polynomial in $1/\epsilon$ and $1/\delta$. As shown by (2.9), the generalization error of consistent hypotheses is upper bounded by a term that decreases as a function of the sample size m. This is a general fact: as expected, learning algorithms benefit from larger labeled training samples. The decrease rate of $O(1/m)$ guaranteed by this theorem, however, is particularly favorable.

The price to pay for coming up with a consistent algorithm is the use of a larger hypothesis set $\mathcal{H}$ containing target concepts. Of course, the upper bound (2.9) increases with $|\mathcal{H}|$. However, that dependency is only logarithmic. Note that the term $\log|\mathcal{H}|$, or the related term $\log_2|\mathcal{H}|$ from which it differs by a constant factor, can be interpreted as the number of bits needed to represent $\mathcal{H}$. Thus, the generalization guarantee of the theorem is controlled by the ratio of this number of bits, $\log_2|\mathcal{H}|$, and the sample size m.

We now use theorem 2.5 to analyze PAC-learning with various concept classes.

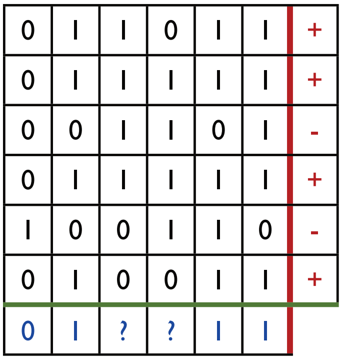
Example 2.6 (Conjunction of Boolean literals) Consider learning the concept class $\mathcal{C}_n$ of conjunctions of at most n Boolean literals $x_1, \ldots, x_n$. A Boolean literal is either a variable x_i, $i \in [n]$, or its negation $\overline{x}_i$. For $n = 4$, an example is the conjunction: $x_1 \wedge \overline{x}_2 \wedge x_4$, where $\overline{x}_2$ denotes the negation of the Boolean literal x_2. $(1, 0, 0, 1)$ is a positive example for this concept while $(1, 0, 0, 0)$ is a negative example.

Observe that for $n = 4$, a positive example $(1, 0, 1, 0)$ implies that the target concept cannot contain the literals $\overline{x}_1$ and $\overline{x}_3$ and that it cannot contain the literals x_2 and x_4. In contrast, a negative example is not as informative since it is not known which of its n bits are incorrect. A simple algorithm for finding a consistent hypothesis is thus based on positive examples and consists of the following: for each positive example $(b_1, \ldots, b_n)$ and $i \in [n]$, if $b_i = 1$ then $\overline{x}_i$ is ruled out as a possible literal in the concept class and if $b_i = 0$ then x_i is ruled out. The conjunction of all the literals not ruled out is thus a hypothesis consistent with the target. Figure 2.4 shows an example training sample as well as a consistent hypothesis for the case $n = 6$.

We have $|\mathcal{H}| = |\mathcal{C}_n| = 3^n$, since each literal can be included positively, with negation, or not included. Plugging this into the sample complexity bound for consistent hypotheses yields the following sample complexity bound for any $\epsilon > 0$ and $\delta > 0$:

$$m \geq \frac{1}{\epsilon}\Big((\log 3)n + \log\frac{1}{\delta}\Big). \tag{2.10}$$

Thus, the class of conjunctions of at most n Boolean literals is PAC-learnable. Note that the computational complexity is also polynomial, since the training cost per example is in $O(n)$. For $\delta = 0.02$, $\epsilon = 0.1$, and $n = 10$, the bound becomes



0	1	1	0	1	1	+
0	1	1	1	1	1	+
0	0	1	1	0	1	-
0	1	1	1	1	1	+
1	0	0	1	1	0	-
0	1	0	0	1	1	+
0	1	?	?	1	1	

Figure 2.4
Each of the first six rows of the table represents a training example with its label, $+$ or $-$, indicated in the last column. The last row contains 0 (respectively 1) in column $i \in [6]$ if the ith entry is 0 (respectively 1) for all the positive examples. It contains "?" if both 0 and 1 appear as an ith entry for some positive example. Thus, for this training sample, the hypothesis returned by the consistent algorithm described in the text is $\overline{x}_1 \wedge x_2 \wedge x_5 \wedge x_6$.

$m \geq 149$. Thus, for a labeled sample of at least 149 examples, the bound guarantees 90% accuracy with a confidence of at least 98%.

Example 2.7 (Universal concept class) Consider the set $\mathcal{X} = \{0,1\}^n$ of all Boolean vectors with n components, and let $\mathcal{U}_n$ be the concept class formed by all subsets of $\mathcal{X}$. Is this concept class PAC-learnable? To guarantee a consistent hypothesis the hypothesis class must include the concept class, thus $|\mathcal{H}| \geq |\mathcal{U}_n| = 2^{(2^n)}$. Theorem 2.5 gives the following sample complexity bound:

$$m \geq \frac{1}{\epsilon}\Big((\log 2)2^n + \log \frac{1}{\delta}\Big). \tag{2.11}$$

Here, the number of training samples required is exponential in n, which is the cost of the representation of a point in $\mathcal{X}$. Thus, PAC-learning is not guaranteed by the theorem. In fact, it is not hard to show that this universal concept class is not PAC-learnable.

Example 2.8 (k-term DNF formulae) A disjunctive normal form (DNF) formula is a formula written as the disjunction of several terms, each term being a conjunction of Boolean literals. A k-term DNF is a DNF formula defined by the disjunction of k terms, each term being a conjunction of at most n Boolean literals. Thus, for $k = 2$ and $n = 3$, an example of a k-term DNF is $(x_1 \wedge \overline{x}_2 \wedge x_3) \vee (\overline{x}_1 \wedge x_3)$.

Is the class $\mathcal{C}$ of k-term DNF formulae PAC-learnable? The cardinality of the class is 3^{nk}, since each term is a conjunction of at most n variables and there are 3^n such conjunctions, as seen previously. The hypothesis set $\mathcal{H}$ must contain $\mathcal{C}$ for consistency to be possible, thus $|\mathcal{H}| \geq 3^{nk}$. Theorem 2.5 gives the following sample complexity bound:

$$m \geq \frac{1}{\epsilon}\Big((\log 3)nk + \log \frac{1}{\delta}\Big), \tag{2.12}$$

which is polynomial. However, it can be shown by a reduction from the graph 3-coloring problem that the problem of learning k-term DNF, even for $k = 3$, is not efficiently PAC-learnable, unless RP, the complexity class of problems that admit a randomized polynomial-time decision solution, coincides with NP(RP = NP), which is commonly conjectured not to be the case. Thus, while the sample size needed for learning k-term DNF formulae is only polynomial, efficient PAC-learning of this class is not possible if RP $\neq$ NP.

Example 2.9 (k-CNF formulae) A conjunctive normal form (CNF) formula is a conjunction of disjunctions. A k-CNF formula is an expression of the form $T_1 \wedge \ldots \wedge T_j$ with arbitrary length $j \in \mathbb{N}$ and with each term T_i being a disjunction of at most k Boolean attributes.

The problem of learning k-CNF formulae can be reduced to that of learning conjunctions of Boolean literals, which, as seen previously, is a PAC-learnable concept class. This can be done at the cost of introducing $(2n)^k$ new variables $Y_{u_1,\ldots,u_k}$ using the following bijection:

$$(u_1, \ldots, u_k) \to Y_{u_1,\ldots,u_k}, \tag{2.13}$$

where $u_1, \ldots, u_k$ are Boolean literals over the original variables $x_1, \ldots, x_n$. The value of $Y_{u_1,\ldots,u_k}$ is determined by $Y_{u_1,\ldots,u_k} = u_1 \vee \cdots \vee u_k$. Using this mapping, the original training sample can be transformed into one defined in terms of the new variables and any k-CNF formula over the original variables can be written as a conjunction over the variables $Y_{u_1,\ldots,u_k}$. This reduction to PAC-learning of conjunctions of Boolean literals can affect the original distribution of examples, but this is not an issue since in the PAC framework no assumption is made about the distribution. Thus, using this transformation, the PAC-learnability of conjunctions of Boolean literals implies that of k-CNF formulae.

This is a surprising result, however, since any k-term DNF formula can be written as a k-CNF formula. Indeed, using associativity, a k-term DNF $T_1 \vee \cdots \vee T_k$ with $T_i = u_{i,1} \wedge \cdots \wedge u_{i,n_i}$ for $i \in [k]$ can be rewritten as a k-CNF formula via

$$\bigvee_{i=1}^{k} u_{i,1} \wedge \cdots \wedge u_{i,n_i} = \bigwedge_{j_1 \in [n_1], \ldots, j_k \in [n_k]} u_{1,j_1} \vee \cdots \vee u_{k,j_k},$$

To illustrate this rewriting in a specific case, observe, for example, that

$$(u_1 \wedge u_2 \wedge u_3) \vee (v_1 \wedge v_2 \wedge v_3) = \bigwedge_{i,j=1}^{3} (u_i \vee v_j).$$

But, as we previously saw, k-term DNF formulae are not efficiently PAC-learnable if RP $\neq$ NP! What can explain this apparent inconsistency? The issue is that converting into a k-term DNF a k-CNF formula we have learned (which is equivalent to a k-term DNF) is in general intractable if RP $\neq$ NP.

This example reveals some key aspects of PAC-learning, which include the cost of the representation of a concept and the choice of the hypothesis set. For a fixed concept class, learning can be intractable or not depending on the choice of the representation.

2.3 Guarantees for finite hypothesis sets — inconsistent case

In the most general case, there may be no hypothesis in $\mathcal{H}$ consistent with the labeled training sample. This, in fact, is the typical case in practice, where the learning problems may be somewhat difficult or the concept classes more complex than the hypothesis set used by the learning algorithm. However, inconsistent hypotheses with a small number of errors on the training sample can be useful and, as we shall see, can benefit from favorable guarantees under some assumptions. This section presents learning guarantees precisely for this inconsistent case and finite hypothesis sets.

To derive learning guarantees in this more general setting, we will use Hoeffding's inequality (theorem D.2) or the following corollary, which relates the generalization error and empirical error of a single hypothesis.

Corollary 2.10 *Fix $\epsilon > 0$. Then, for any hypothesis $h\colon X \to \{0,1\}$, the following inequalities hold:*

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left[\widehat{R}_S(h) - R(h) \geq \epsilon\right] \leq \exp(-2m\epsilon^2) \tag{2.14}$$

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left[\widehat{R}_S(h) - R(h) \leq -\epsilon\right] \leq \exp(-2m\epsilon^2). \tag{2.15}$$

By the union bound, this implies the following two-sided inequality:

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left[\left|\widehat{R}_S(h) - R(h)\right| \geq \epsilon\right] \leq 2\exp(-2m\epsilon^2). \tag{2.16}$$

Proof: The result follows immediately from theorem D.2. □

Setting the right-hand side of (2.16) to be equal to δ and solving for ϵ yields immediately the following bound for a single hypothesis.

Corollary 2.11 (Generalization bound — single hypothesis) *Fix a hypothesis $h\colon \mathcal{X} \to \{0,1\}$. Then, for any $\delta > 0$, the following inequality holds with probability at least $1-\delta$:*

$$R(h) \leq \widehat{R}_S(h) + \sqrt{\frac{\log\frac{2}{\delta}}{2m}}. \tag{2.17}$$

The following example illustrates this corollary in a simple case.

Example 2.12 (Tossing a coin) Imagine tossing a biased coin that lands heads with probability p, and let our hypothesis be the one that always guesses tails. Then the true error rate is $R(h) = p$ and the empirical error rate $\widehat{R}_S(h) = \widehat{p}$, where $\widehat{p}$ is the empirical probability of heads based on the training sample drawn i.i.d. Thus, corollary 2.11 guarantees with probability at least $1 - \delta$ that

$$|p - \widehat{p}| \leq \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{2.18}$$

Therefore, if we choose $\delta = 0.02$ and use a sample of size 500, with probability at least 98%, the following approximation quality is guaranteed for $\widehat{p}$:

$$|p - \widehat{p}| \leq \sqrt{\frac{\log(10)}{1000}} \approx 0.048. \tag{2.19}$$

Can we readily apply corollary 2.11 to bound the generalization error of the hypothesis h_S returned by a learning algorithm when training on a sample S? No, since h_S is not a fixed hypothesis, but a random variable depending on the training sample S drawn. Note also that unlike the case of a fixed hypothesis for which the expectation of the empirical error is the generalization error (equation (2.3)), the generalization error $R(h_S)$ is a random variable and in general distinct from the expectation $\mathbb{E}[\widehat{R}_S(h_S)]$, which is a constant.

Thus, as in the proof for the consistent case, we need to derive a uniform convergence bound, that is a bound that holds with high probability for all hypotheses $h \in \mathcal{H}$.

Theorem 2.13 (Learning bound — finite $\mathcal{H}$, inconsistent case) *Let $\mathcal{H}$ be a finite hypothesis set. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following inequality holds:*

$$\forall h \in \mathcal{H}, \quad R(h) \leq \widehat{R}_S(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{2}{\delta}}{2m}}. \tag{2.20}$$

Proof: Let $h_1, \ldots, h_{|\mathcal{H}|}$ be the elements of $\mathcal{H}$. Using the union bound and applying corollary 2.11 to each hypothesis yield:

$$\begin{aligned}
\mathbb{P}\Big[\exists h \in \mathcal{H} \big|\widehat{R}_S(h) - R(h)\big| > \epsilon\Big]& \\
&= \mathbb{P}\Big[\big(\big|\widehat{R}_S(h_1) - R(h_1)\big| > \epsilon\big) \vee \ldots \vee \big(\big|\widehat{R}_S(h_{|\mathcal{H}|}) - R(h_{|\mathcal{H}|})\big| > \epsilon\big)\Big] \\
&\leq \sum_{h \in \mathcal{H}} \mathbb{P}\Big[\big|\widehat{R}_S(h) - R(h)\big| > \epsilon\Big] \\
&\leq 2|\mathcal{H}| \exp(-2m\epsilon^2).
\end{aligned}$$

Setting the right-hand side to be equal to δ completes the proof. □

Thus, for a finite hypothesis set $\mathcal{H}$,

$$R(h) \leq \widehat{R}_S(h) + O\left(\sqrt{\frac{\log_2 |\mathcal{H}|}{m}}\right).$$

As already pointed out, $\log_2 |\mathcal{H}|$ can be interpreted as the number of bits needed to represent $\mathcal{H}$. Several other remarks similar to those made on the generalization bound in the consistent case can be made here: a larger sample size m guarantees better generalization, and the bound increases with $|\mathcal{H}|$, but only logarithmically. But, here, the bound is a less favorable function of $\frac{\log_2 |\mathcal{H}|}{m}$; it varies as the square root of this term. This is not a minor price to pay: for a fixed $|\mathcal{H}|$, to attain the same guarantee as in the consistent case, a quadratically larger labeled sample is needed.

Note that the bound suggests seeking a trade-off between reducing the empirical error versus controlling the size of the hypothesis set: a larger hypothesis set is penalized by the second term but could help reduce the empirical error, that is the first term. But, for a similar empirical error, it suggests using a smaller hypothesis set. This can be viewed as an instance of the so-called *Occam's Razor principle* named after the theologian William of Occam: *Plurality should not be posited without necessity*, also rephrased as, *the simplest explanation is best*. In this context, it could be expressed as follows: All other things being equal, a simpler (smaller) hypothesis set is better.

2.4 Generalities

In this section we will discuss some general aspects of the learning scenario, which, for simplicity, we left out of the discussion of the earlier sections.

2.4.1 Deterministic versus stochastic scenarios

In the most general scenario of supervised learning, the distribution $\mathcal{D}$ is defined over $\mathcal{X} \times \mathcal{Y}$, and the training data is a labeled sample S drawn i.i.d. according to $\mathcal{D}$:

$$S = ((x_1, y_1), \ldots, (x_m, y_m)).$$

The learning problem is to find a hypothesis $h \in \mathcal{H}$ with small generalization error

$$R(h) = \underset{(x,y)\sim\mathcal{D}}{\mathbb{P}}[h(x) \neq y] = \underset{(x,y)\sim\mathcal{D}}{\mathbb{E}}[1_{h(x)\neq y}].$$

This more general scenario is referred to as the *stochastic scenario*. Within this setting, the output label is a probabilistic function of the input. The stochastic scenario captures many real-world problems where the label of an input point is not unique. For example, if we seek to predict gender based on input pairs formed by the height and weight of a person, then the label will typically not be unique.

For most pairs, both male and female are possible genders. For each fixed pair, there would be a probability distribution of the label being male.

The natural extension of the PAC-learning framework to this setting is known as the *agnostic PAC-learning*.

Definition 2.14 (Agnostic PAC-learning) *Let $\mathcal{H}$ be a hypothesis set. $\mathcal{A}$ is an* agnostic PAC-learning *algorithm if there exists a polynomial function $poly(\cdot, \cdot, \cdot, \cdot)$ such that for any $\epsilon > 0$ and $\delta > 0$, for all distributions $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, the following holds for any sample size $m \geq poly(1/\epsilon, 1/\delta, n, size(c))$:*

$$\mathop{\mathbb{P}}_{S \sim \mathcal{D}^m}\Big[R(h_S) - \min_{h \in \mathcal{H}} R(h) \leq \epsilon\Big] \geq 1 - \delta. \tag{2.21}$$

If $\mathcal{A}$ further runs in $poly(1/\epsilon, 1/\delta, n)$, then it is said to be an efficient agnostic PAC-learning algorithm.

When the label of a point can be uniquely determined by some measurable function $f\colon \mathcal{X} \to \mathcal{Y}$ (with probability one), then the scenario is said to be *deterministic*. In that case, it suffices to consider a distribution $\mathcal{D}$ over the input space. The training sample is obtained by drawing $(x_1, \ldots, x_m)$ according to $\mathcal{D}$ and the labels are obtained via f: $y_i = f(x_i)$ for all $i \in [m]$. Many learning problems can be formulated within this deterministic scenario.

In the previous sections, as well as in most of the material presented in this book, we have restricted our presentation to the deterministic scenario in the interest of simplicity. However, for all of this material, the extension to the stochastic scenario should be straightforward for the reader.

2.4.2 Bayes error and noise

In the deterministic case, by definition, there exists a target function f with no generalization error: $R(h) = 0$. In the stochastic case, there is a minimal non-zero error for any hypothesis.

Definition 2.15 (Bayes error) *Given a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, the* Bayes error R^* *is defined as the infimum of the errors achieved by measurable functions $h\colon \mathcal{X} \to \mathcal{Y}$:*

$$R^\star = \inf_{\substack{h \\ h \text{ measurable}}} R(h). \tag{2.22}$$

A hypothesis h with $R(h) = R^*$ is called a Bayes hypothesis *or* Bayes classifier.

By definition, in the deterministic case, we have $R^* = 0$, but, in the stochastic case, $R^* \neq 0$. Clearly, the Bayes classifier h_{Bayes} can be defined in terms of the conditional probabilities as:

$$\forall x \in \mathcal{X}, \quad h_{\text{Bayes}}(x) = \operatorname*{argmax}_{y \in \{0,1\}} \mathbb{P}[y|x]. \tag{2.23}$$

The average error made by h_{Bayes} on $x \in \mathcal{X}$ is thus $\min\{\mathbb{P}[0|x], \mathbb{P}[1|x]\}$, and this is the minimum possible error. This leads to the following definition of *noise*.

Definition 2.16 (Noise) *Given a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, the* noise *at point $x \in \mathcal{X}$ is defined by*

$$noise(x) = \min\{\mathbb{P}[1|x], \mathbb{P}[0|x]\}. \tag{2.24}$$

The average noise *or the* noise *associated to $\mathcal{D}$ is $\mathbb{E}[noise(x)]$.*

Thus, the average noise is precisely the Bayes error: $\text{noise} = \mathbb{E}[\text{noise}(x)] = R^*$. The noise is a characteristic of the learning task indicative of its level of difficulty. A point $x \in \mathcal{X}$, for which $\text{noise}(x)$ is close to $1/2$, is sometimes referred to as *noisy* and is of course a challenge for accurate prediction.

2.5 Chapter notes

The PAC learning framework was introduced by Valiant [1984]. The book of Kearns and Vazirani [1994] is an excellent reference dealing with most aspects of PAC-learning and several other foundational questions in machine learning. Our example of learning axis-aligned rectangles, also discussed in that reference, is originally due to Blumer et al. [1989].

The PAC learning framework is a computational framework since it takes into account the cost of the computational representations and the time complexity of the learning algorithm. If we omit the computational aspects, it is similar to the learning framework considered earlier by Vapnik and Chervonenkis [see Vapnik, 2000]. The definition of noise presented in this chapter can be generalized to arbitrary loss functions (see exercise 2.14).

Occam's razor principle is invoked in a variety of contexts, such as in linguistics to justify the superiority of a set of rules or syntax. The Kolmogorov complexity can be viewed as the corresponding framework in information theory. In the context of the learning guarantees presented in this chapter, the principle suggests selecting the most parsimonious explanation (the hypothesis set with the smallest cardinality). We will see in the next sections other applications of this principle with different notions of simplicity or complexity.

2.6 Exercises

2.1 Two-oracle variant of the PAC model. Assume that positive and negative examples are now drawn from two separate distributions $\mathcal{D}_+$ and $\mathcal{D}_-$. For an accuracy $(1 - \epsilon)$, the learning algorithm must find a hypothesis h such that:

$$\underset{x \sim \mathcal{D}_+}{\mathbb{P}}[h(x) = 0] \leq \epsilon \text{ and } \underset{x \sim \mathcal{D}_-}{\mathbb{P}}[h(x) = 1] \leq \epsilon\,. \tag{2.25}$$

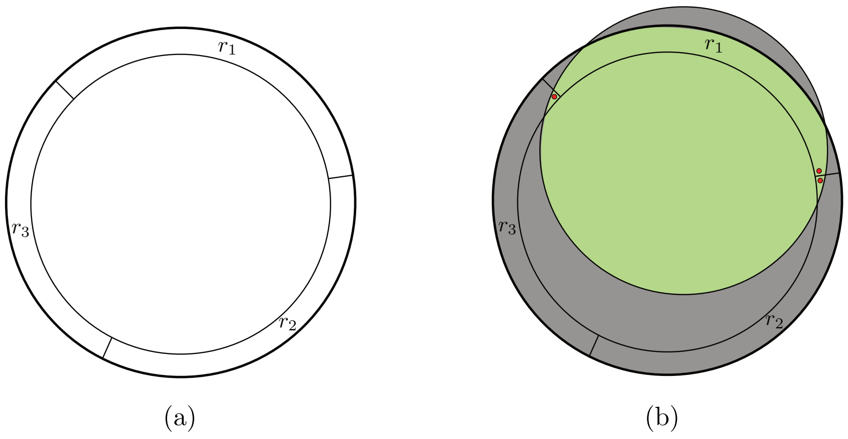


Figure 2.5
(a) Gertrude's regions r_1, r_2, r_3. (b) Hint for solution.

Thus, the hypothesis must have a small error on both distributions. Let $\mathcal{C}$ be any concept class and $\mathcal{H}$ be any hypothesis space. Let h_0 and h_1 represent the identically 0 and identically 1 functions, respectively. Prove that $\mathcal{C}$ is efficiently PAC-learnable using $\mathcal{H}$ in the standard (one-oracle) PAC model if and only if it is efficiently PAC-learnable using $\mathcal{H} \cup \{h_0, h_1\}$ in this two-oracle PAC model.

2.2 PAC learning of hyper-rectangles. An axis-aligned hyper-rectangle in $\mathbb{R}^n$ is a set of the form $[a_1, b_1] \times \ldots \times [a_n, b_n]$. Show that axis-aligned hyper-rectangles are PAC-learnable by extending the proof given in Example 2.4 for the case $n = 2$.

2.3 Concentric circles. Let $\mathcal{X} = \mathbb{R}^2$ and consider the set of concepts of the form $c = \{(x, y) \colon x^2 + y^2 \leq r^2\}$ for some real number r. Show that this class can be (ϵ, δ)-PAC-learned from training data of size $m \geq (1/\epsilon)\log(1/\delta)$.

2.4 Non-concentric circles. Let $\mathcal{X} = \mathbb{R}^2$ and consider the set of concepts of the form $c = \{x \in \mathbb{R}^2 \colon ||x - x_0|| \leq r\}$ for some point $x_0 \in \mathbb{R}^2$ and real number r. Gertrude, an aspiring machine learning researcher, attempts to show that this class of concepts may be (ϵ, δ)-PAC-learned with sample complexity $m \geq (3/\epsilon)\log(3/\delta)$, but she is having trouble with her proof. Her idea is that the learning algorithm would select the smallest circle consistent with the training data. She has drawn three regions r_1, r_2, r_3 around the edge of concept c, with each region having probability $\epsilon/3$ (see figure 2.5(a)). She wants to argue that if the generalization error is greater than or equal to ϵ, then one of these regions must have been missed by the training data, and hence this event will occur with probability at most δ. Can you tell Gertrude if her approach works? (*Hint*: You may wish to use figure 2.5(b) in your solution).

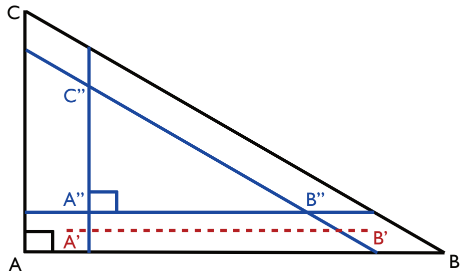


Figure 2.6
Axis-aligned right triangles.

2.5 Triangles. Let $\mathcal{X} = \mathbb{R}^2$ with orthonormal basis $(\mathbf{e}_1, \mathbf{e}_2)$, and consider the set of concepts defined by the area inside a right triangle ABC with two sides parallel to the axes, with $\overrightarrow{AB}/\|\overrightarrow{AB}\| = \mathbf{e}_1$ and $\overrightarrow{AC}/\|\overrightarrow{AC}\| = \mathbf{e}_2$, and $\|\overrightarrow{AB}\|/\|\overrightarrow{AC}\| = \alpha$ for some positive real $\alpha \in \mathbb{R}_+$. Show, using similar methods to those used in the chapter for the axis-aligned rectangles, that this class can be (ϵ, δ)-PAC-learned from training data of size $m \geq (3/\epsilon)\log(3/\delta)$. (*Hint*: You may consider using figure 2.6 in your solution).

2.6 Learning in the presence of noise — rectangles. In example 2.4, we showed that the concept class of axis-aligned rectangles is PAC-learnable. Consider now the case where the training points received by the learner are subject to the following noise: points negatively labeled are unaffected by noise but the label of a positive training point is randomly flipped to negative with probability $\eta \in (0, \frac{1}{2})$. The exact value of the noise rate η is not known to the learner but an upper bound η' is supplied to him with $\eta \leq \eta' < 1/2$. Show that the algorithm returning the tightest rectangle containing positive points can still PAC-learn axis-aligned rectangles in the presence of this noise. To do so, you can proceed using the following steps:

(a) Using the same notation as in example 2.4, assume that $\mathbb{P}[\mathsf{R}] > \epsilon$. Suppose that $R(\mathsf{R}') > \epsilon$. Give an upper bound on the probability that R' misses a region r_j, $j \in [4]$ in terms of ϵ and η'?

(b) Use that to give an upper bound on $\mathbb{P}[R(\mathsf{R}') > \epsilon]$ in terms of ϵ and η' and conclude by giving a sample complexity bound.

2.7 Learning in the presence of noise — general case. In this question, we will seek a result that is more general than in the previous question. We consider a finite hypothesis set $\mathcal{H}$, assume that the target concept is in $\mathcal{H}$, and adopt the following noise model: the label of a training point received by the learner is randomly changed with probability $\eta \in (0, \frac{1}{2})$. The exact value of the noise rate η is not known to the learner but an upper bound η' is supplied to him with $\eta \leq \eta' < 1/2$.

(a) For any $h \in \mathcal{H}$, let $d(h)$ denote the probability that the label of a training point received by the learner disagrees with the one given by h. Let h^* be the target hypothesis, show that $d(h^*) = \eta$.

(b) More generally, show that for any $h \in \mathcal{H}$, $d(h) = \eta + (1 - 2\eta)\, R(h)$, where $R(h)$ denotes the generalization error of h.

(c) Fix $\epsilon > 0$ for this and all the following questions. Use the previous questions to show that if $R(h) > \epsilon$, then $d(h) - d(h^*) \geq \epsilon'$, where $\epsilon' = \epsilon(1 - 2\eta')$.

(d) For any hypothesis $h \in \mathcal{H}$ and sample S of size m, let $\widehat{d}(h)$ denote the fraction of the points in S whose labels disagree with those given by h. We will consider the algorithm L which, after receiving S, returns the hypothesis h_S with the smallest number of disagreements (thus $\widehat{d}(h_S)$ is minimal). To show PAC-learning for L, we will show that for any h, if $R(h) > \epsilon$, then with high probability $\widehat{d}(h) \geq \widehat{d}(h^*)$. First, show that for any $\delta > 0$, with probability at least $1 - \delta/2$, for $m \geq \frac{2}{\epsilon'^2} \log \frac{2}{\delta}$, the following holds:

$$\widehat{d}(h^*) - d(h^*) \leq \epsilon'/2$$

(e) Second, show that for any $\delta > 0$, with probability at least $1 - \delta/2$, for $m \geq \frac{2}{\epsilon'^2}(\log |\mathcal{H}| + \log \frac{2}{\delta})$, the following holds for all $h \in \mathcal{H}$:

$$d(h) - \widehat{d}(h) \leq \epsilon'/2$$

(f) Finally, show that for any $\delta > 0$, with probability at least $1 - \delta$, for $m \geq \frac{2}{\epsilon^2(1-2\eta')^2}(\log |\mathcal{H}| + \log \frac{2}{\delta})$, the following holds for all $h \in \mathcal{H}$ with $R(h) > \epsilon$:

$$\widehat{d}(h) - \widehat{d}(h^*) \geq 0.$$

(*Hint*: use $\widehat{d}(h) - \widehat{d}(h^*) = [\widehat{d}(h) - d(h)] + [d(h) - d(h^*)] + [d(h^*) - \widehat{d}(h^*)]$ and use previous questions to lower bound each of these three terms).

2.8 Learning intervals. Give a PAC-learning algorithm for the concept class $\mathcal{C}$ formed by closed intervals $[a, b]$ with $a, b \in \mathbb{R}$.

2.9 Learning union of intervals. Give a PAC-learning algorithm for the concept class $\mathcal{C}_2$ formed by unions of two closed intervals, that is $[a, b] \cup [c, d]$, with $a, b, c, d \in \mathbb{R}$. Extend your result to derive a PAC-learning algorithm for the concept class $\mathcal{C}_p$ formed by unions of $p \geq 1$ closed intervals, thus $[a_1, b_1] \cup \cdots \cup [a_p, b_p]$, with $a_k, b_k \in \mathbb{R}$ for $k \in [p]$. What are the time and sample complexities of your algorithm as a function of p?

2.10 Consistent hypotheses. In this chapter, we showed that for a finite hypothesis set $\mathcal{H}$, a consistent learning algorithm $\mathcal{A}$ is a PAC-learning algorithm. Here, we consider a converse question. Let $\mathcal{Z}$ be a finite set of m labeled points. Suppose that you are given a PAC-learning algorithm $\mathcal{A}$. Show that you can use $\mathcal{A}$ and a finite training sample S to find in polynomial time a hypothesis $h \in \mathcal{H}$ that is consistent with $\mathcal{Z}$, with high probability. (*Hint*: you can select an appropriate distribution $\mathcal{D}$ over $\mathcal{Z}$ and give a condition on $R(h)$ for h to be consistent.)

2.11 Senate laws. For important questions, President Mouth relies on expert advice. He selects an appropriate advisor from a collection of $\mathcal{H} = 2{,}800$ experts.

(a) Assume that laws are proposed in a random fashion independently and identically according to some distribution $\mathcal{D}$ determined by an unknown group of senators. Assume that President Mouth can find and select an expert senator out of $\mathcal{H}$ who has consistently voted with the majority for the last $m = 200$ laws. Give a bound on the probability that such a senator incorrectly predicts the global vote for a future law. What is the value of the bound with 95% confidence?

(b) Assume now that President Mouth can find and select an expert senator out of $\mathcal{H}$ who has consistently voted with the majority for all but $m' = 20$ of the last $m = 200$ laws. What is the value of the new bound?

2.12 Bayesian bound. Let $\mathcal{H}$ be a countable hypothesis set of functions mapping $\mathcal{X}$ to $\{0, 1\}$ and let p be a probability measure over $\mathcal{H}$. This probability measure represents the *prior probability* over the hypothesis class, i.e. the probability that a particular hypothesis is selected by the learning algorithm. Use Hoeffding's inequality to show that for any $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds:

$$\forall h \in \mathcal{H}, R(h) \leq \widehat{R}_S(h) + \sqrt{\frac{\log \frac{1}{p(h)} + \log \frac{1}{\delta}}{2m}}. \tag{2.26}$$

Compare this result with the bound given in the inconsistent case for finite hypothesis sets (*Hint*: you could use $\delta' = p(h)\delta$ as confidence parameter in Hoeffding's inequality).

2.13 Learning with an unknown parameter. In example 2.9, we showed that the concept class of k-CNF is PAC-learnable. Note, however, that the learning algorithm is given k as input. Is PAC-learning possible even when k is not provided? More generally, consider a family of concept classes $\{\mathcal{C}_s\}_s$ where $\mathcal{C}_s$ is the set of concepts in $\mathcal{C}$ with size at most s. Suppose we have a PAC-learning algorithm $\mathcal{A}$ that can be used for learning any concept class $\mathcal{C}_s$ when s is given.

Can we convert $\mathcal{A}$ into a PAC-learning algorithm $\mathcal{B}$ that does not require the knowledge of s? This is the main objective of this problem.

To do this, we first introduce a method for testing a hypothesis h, with high probability. Fix $\epsilon > 0$, $\delta > 0$, and $i \geq 1$ and define the sample size n by $n = \frac{32}{\epsilon}[i \log 2 + \log \frac{2}{\delta}]$. Suppose we draw an i.i.d. sample S of size n according to some unknown distribution $\mathcal{D}$. We will say that a hypothesis h is *accepted* if it makes at most $3/4\epsilon$ errors on S and that it is *rejected* otherwise. Thus, h is accepted iff $\widehat{R}(h) \leq 3/4\epsilon$.

(a) Assume that $R(h) \geq \epsilon$. Use the (multiplicative) Chernoff bound to show that in that case $\mathbb{P}_{S \sim \mathcal{D}^n}[h \text{ is accepted}] \leq \frac{\delta}{2^{i+1}}$.

(b) Assume that $R(h) \leq \epsilon/2$. Use the (multiplicative) Chernoff bounds to show that in that case $\mathbb{P}_{S \sim \mathcal{D}^n}[h \text{ is rejected}] \leq \frac{\delta}{2^{i+1}}$.

(c) Algorithm $\mathcal{B}$ is defined as follows: we start with $i = 1$ and, at each round $i \geq 1$, we guess the parameter size s to be $\widetilde{s} = \lfloor 2^{(i-1)/\log \frac{2}{\delta}} \rfloor$. We draw a sample S of size n (which depends on i) to test the hypothesis h_i returned by $\mathcal{A}$ when it is trained with a sample of size $S_{\mathcal{A}}(\epsilon/2, 1/2, \widetilde{s})$, that is the sample complexity of $\mathcal{A}$ for a required precision $\epsilon/2$, confidence $1/2$, and size $\widetilde{s}$ (we ignore the size of the representation of each example here). If h_i is accepted, the algorithm stops and returns h_i, otherwise it proceeds to the next iteration. Show that if at iteration i, the estimate $\widetilde{s}$ is larger than or equal to s, then $\mathbb{P}[h_i \text{ is accepted}] \geq 3/8$.

(d) Show that the probability that $\mathcal{B}$ does not halt after $j = \lceil \log \frac{2}{\delta} / \log \frac{8}{5} \rceil$ iterations with $\widetilde{s} \geq s$ is at most $\delta/2$.

(e) Show that for $i \geq \lceil 1 + (\log_2 s) \log \frac{2}{\delta} \rceil$, the inequality $\widetilde{s} \geq s$ holds.

(f) Show that with probability at least $1 - \delta$, algorithm $\mathcal{B}$ halts after at most $j' = \lceil 1 + (\log_2 s) \log \frac{2}{\delta} \rceil + j$ iterations and returns a hypothesis with error at most ϵ.

2.14 In this exercise, we generalize the notion of noise to the case of an arbitrary loss function $L \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$.

(a) Justify the following definition of the noise at point $x \in \mathcal{X}$:

$$\text{noise}(x) = \min_{y' \in \mathcal{Y}} \mathop{\mathbb{E}}_{y}[L(y, y')|x].$$

What is the value of noise(x) in a deterministic scenario? Does the definition match the one given in this chapter for binary classification?

(b) Show that the average noise coincides with the Bayes error (minimum loss achieved by a measurable function).

3 Rademacher Complexity and VC-Dimension

The hypothesis sets typically used in machine learning are infinite. But the sample complexity bounds of the previous chapter are uninformative when dealing with infinite hypothesis sets. One could ask whether efficient learning from a finite sample is even possible when the hypothesis set $\mathcal{H}$ is infinite. Our analysis of the family of axis-aligned rectangles (Example 2.4) indicates that this is indeed possible at least in some cases, since we proved that that infinite concept class was PAC-learnable. Our goal in this chapter will be to generalize that result and derive general learning guarantees for infinite hypothesis sets.

A general idea for doing so consists of reducing the infinite case to the analysis of finite sets of hypotheses and then proceed as in the previous chapter. There are different techniques for that reduction, each relying on a different notion of complexity for the family of hypotheses. The first complexity notion we will use is that of *Rademacher complexity*. This will help us derive learning guarantees using relatively simple proofs based on McDiarmid's inequality, while obtaining high-quality bounds, including data-dependent ones, which we will frequently make use of in future chapters. However, the computation of the empirical Rademacher complexity is NP-hard for some hypothesis sets. Thus, we subsequently introduce two other purely combinatorial notions, the *growth function* and the *VC-dimension*. We first relate the Rademacher complexity to the growth function and then bound the growth function in terms of the VC-dimension. The VC-dimension is often easier to bound or estimate. We will review a series of examples showing how to compute or bound it, then relate the growth function and the VC-dimensions. This leads to generalization bounds based on the VC-dimension. Finally, we present lower bounds based on the VC-dimension for two different settings: The *realizable* setting, where there is at least one hypothesis in the hypothesis set under consideration that achieves zero expected error, as well as the *non-realizable* setting, where no hypothesis in the set achieves zero expected error.

3.1 Rademacher complexity

We will continue to use $\mathcal{H}$ to denote a hypothesis set as in the previous chapters. Many of the results of this section are general and hold for an arbitrary loss function $L\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. In what follows, $\mathcal{G}$ will generally be interpreted as *the family of loss functions associated to* $\mathcal{H}$ mapping from $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ to $\mathbb{R}$:

$$\mathcal{G} = \{g\colon (x, y) \mapsto L(h(x), y)\colon h \in \mathcal{H}\}.$$

However, the definitions are given in the general case of a family of functions $\mathcal{G}$ mapping from an arbitrary input space $\mathcal{Z}$ to $\mathbb{R}$.

The Rademacher complexity captures the richness of a family of functions by measuring the degree to which a hypothesis set can fit random noise. The following states the formal definitions of the empirical and average Rademacher complexity.

Definition 3.1 (Empirical Rademacher complexity) *Let $\mathcal{G}$ be a family of functions mapping from $\mathcal{Z}$ to $[a, b]$ and $S = (z_1, \ldots, z_m)$ a fixed sample of size m with elements in $\mathcal{Z}$. Then, the* empirical Rademacher complexity *of $\mathcal{G}$ with respect to the sample S is defined as:*

$$\widehat{\mathfrak{R}}_S(\mathcal{G}) = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sup_{g \in \mathcal{G}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i g(z_i)\right], \tag{3.1}$$

where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)^\top$, with σ_is independent uniform random variables taking values in $\{-1, +1\}$.[3] *The random variables σ_i are called* Rademacher variables.

Let $\mathbf{g}_S$ denote the vector of values taken by function g over the sample S: $\mathbf{g}_S = (g(z_1), \ldots, g(z_m))^\top$. Then, the empirical Rademacher complexity can be rewritten as

$$\widehat{\mathfrak{R}}_S(\mathcal{G}) = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sup_{g \in \mathcal{G}} \frac{\boldsymbol{\sigma} \cdot \mathbf{g}_S}{m}\right].$$

The inner product $\boldsymbol{\sigma} \cdot \mathbf{g}_S$ measures the correlation of $\mathbf{g}_S$ with the vector of random noise $\boldsymbol{\sigma}$. The supremum $\sup_{g \in \mathcal{G}} \frac{\boldsymbol{\sigma} \cdot \mathbf{g}_S}{m}$ is a measure of how well the function class $\mathcal{G}$ correlates with $\boldsymbol{\sigma}$ over the sample S. Thus, the empirical Rademacher complexity measures on average how well the function class $\mathcal{G}$ correlates with random noise on S. This describes the richness of the family $\mathcal{G}$: richer or more complex families $\mathcal{G}$ can generate more vectors $\mathbf{g}_S$ and thus better correlate with random noise, on average.

[3] We assume implicitly that the supremum over the family $\mathcal{G}$ in this definition is measurable and in general will adopt the same assumption throughout this book for other suprema over a class of functions. This assumption does not hold for arbitrary function classes but it is valid for the hypotheses sets typically considered in practice in machine learning, and the instances discussed in this book.

Definition 3.2 (Rademacher complexity) *Let $\mathcal{D}$ denote the distribution according to which samples are drawn. For any integer $m \geq 1$, the* Rademacher complexity *of $\mathcal{G}$ is the expectation of the empirical Rademacher complexity over all samples of size m drawn according to $\mathcal{D}$:*

$$\mathfrak{R}_m(\mathcal{G}) = \mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[\widehat{\mathfrak{R}}_S(\mathcal{G})]. \tag{3.2}$$

We are now ready to present our first generalization bounds based on Rademacher complexity.

Theorem 3.3 *Let $\mathcal{G}$ be a family of functions mapping from $\mathcal{Z}$ to $[0,1]$. Then, for any $\delta > 0$, with probability at least $1-\delta$ over the draw of an i.i.d. sample S of size m, each of the following holds for all $g \in \mathcal{G}$:*

$$\mathbb{E}[g(z)] \leq \frac{1}{m}\sum_{i=1}^{m} g(z_i) + 2\mathfrak{R}_m(\mathcal{G}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \tag{3.3}$$

$$\textit{and} \quad \mathbb{E}[g(z)] \leq \frac{1}{m}\sum_{i=1}^{m} g(z_i) + 2\widehat{\mathfrak{R}}_S(\mathcal{G}) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}\,. \tag{3.4}$$

Proof: For any sample $S = (z_1, \ldots, z_m)$ and any $g \in \mathcal{G}$, we denote by $\widehat{\mathbb{E}}_S[g]$ the empirical average of g over S: $\widehat{\mathbb{E}}_S[g] = \frac{1}{m}\sum_{i=1}^{m} g(z_i)$. The proof consists of applying McDiarmid's inequality to function Φ defined for any sample S by

$$\Phi(S) = \sup_{g \in \mathcal{G}} \Big(\mathbb{E}[g] - \widehat{\mathbb{E}}_S[g] \Big). \tag{3.5}$$

Let S and S' be two samples differing by exactly one point, say z_m in S and z'_m in S'. Then, since the difference of suprema does not exceed the supremum of the difference, we have

$$\Phi(S') - \Phi(S) \leq \sup_{g \in \mathcal{G}} \Big(\widehat{\mathbb{E}}_S[g] - \widehat{\mathbb{E}}_{S'}[g] \Big) = \sup_{g \in \mathcal{G}} \frac{g(z_m) - g(z'_m)}{m} \leq \frac{1}{m}. \tag{3.6}$$

Similarly, we can obtain $\Phi(S) - \Phi(S') \leq 1/m$, thus $|\Phi(S) - \Phi(S')| \leq 1/m$. Then, by McDiarmid's inequality, for any $\delta > 0$, with probability at least $1 - \delta/2$, the following holds:

$$\Phi(S) \leq \mathop{\mathbb{E}}_{S}[\Phi(S)] + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{3.7}$$

We next bound the expectation of the right-hand side as follows:

$$\begin{aligned}
\mathop{\mathbb{E}}_{S}[\Phi(S)] &= \mathop{\mathbb{E}}_{S}\Big[\sup_{g\in\mathcal{G}}\big(\mathbb{E}[g]-\widehat{\mathbb{E}}_S(g)\big)\Big] \\
&= \mathop{\mathbb{E}}_{S}\Big[\sup_{g\in\mathcal{G}}\mathop{\mathbb{E}}_{S'}\big[\widehat{\mathbb{E}}_{S'}(g)-\widehat{\mathbb{E}}_S(g)\big]\Big] && (3.8)\\
&\le \mathop{\mathbb{E}}_{S,S'}\Big[\sup_{g\in\mathcal{G}}\big(\widehat{\mathbb{E}}_{S'}(g)-\widehat{\mathbb{E}}_S(g)\big)\Big] && (3.9)\\
&= \mathop{\mathbb{E}}_{S,S'}\Big[\sup_{g\in\mathcal{G}}\frac{1}{m}\sum_{i=1}^{m}(g(z'_i)-g(z_i))\Big] && (3.10)\\
&= \mathop{\mathbb{E}}_{\boldsymbol{\sigma},S,S'}\Big[\sup_{g\in\mathcal{G}}\frac{1}{m}\sum_{i=1}^{m}\sigma_i(g(z'_i)-g(z_i))\Big] && (3.11)\\
&\le \mathop{\mathbb{E}}_{\boldsymbol{\sigma},S'}\Big[\sup_{g\in\mathcal{G}}\frac{1}{m}\sum_{i=1}^{m}\sigma_i g(z'_i)\Big]+\mathop{\mathbb{E}}_{\boldsymbol{\sigma},S}\Big[\sup_{g\in\mathcal{G}}\frac{1}{m}\sum_{i=1}^{m}-\sigma_i g(z_i)\Big] && (3.12)\\
&= 2\mathop{\mathbb{E}}_{\boldsymbol{\sigma},S}\Big[\sup_{g\in\mathcal{G}}\frac{1}{m}\sum_{i=1}^{m}\sigma_i g(z_i)\Big] = 2\mathfrak{R}_m(\mathcal{G}). && (3.13)
\end{aligned}$$

Equation (3.8) uses the fact that points in S' are sampled in an i.i.d. fashion and thus $\mathbb{E}[g] = \mathbb{E}_{S'}[\widehat{\mathbb{E}}_{S'}(g)]$, as in (2.3). Inequality 3.9 holds due to the sub-additivity of the supremum function.

In equation (3.11), we introduce Rademacher variables σ_i, which are uniformly distributed independent random variables taking values in $\{-1,+1\}$ as in definition 3.2. This does not change the expectation appearing in (3.10): when $\sigma_i = 1$, the associated summand remains unchanged; when $\sigma_i = -1$, the associated summand flips signs, which is equivalent to swapping z_i and z'_i between S and S'. Since we are taking the expectation over all possible S and S', this swap does not affect the overall expectation; we are simply changing the order of the summands within the expectation.

Equation (3.12) holds by the sub-additivity of the supremum function, that is the inequality $\sup(U+V) \le \sup(U)+\sup(V)$. Finally, (3.13) stems from the definition of Rademacher complexity and the fact that the variables σ_i and $-\sigma_i$ are distributed in the same way.

The reduction to $\mathfrak{R}_m(\mathcal{G})$ in equation (3.13) yields the bound in equation (3.3), using δ instead of $\delta/2$. To derive a bound in terms of $\widehat{\mathfrak{R}}_S(\mathcal{G})$, we observe that, by definition 3.1, changing one point in S changes $\widehat{\mathfrak{R}}_S(\mathcal{G})$ by at most $1/m$. Then, using again McDiarmid's inequality, with probability $1-\delta/2$ the following holds:

$$\mathfrak{R}_m(\mathcal{G}) \le \widehat{\mathfrak{R}}_S(\mathcal{G}) + \sqrt{\frac{\log\frac{2}{\delta}}{2m}}. \tag{3.14}$$

Finally, we use the union bound to combine inequalities 3.7 and 3.14, which yields with probability at least $1 - \delta$:

$$\Phi(S) \leq 2\widehat{\mathfrak{R}}_S(\mathcal{G}) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}, \tag{3.15}$$

which matches (3.4). □

The following result relates the empirical Rademacher complexities of a hypothesis set $\mathcal{H}$ and to the family of loss functions $\mathcal{G}$ associated to $\mathcal{H}$ in the case of binary loss (zero-one loss).

Lemma 3.4 *Let $\mathcal{H}$ be a family of functions taking values in $\{-1, +1\}$ and let $\mathcal{G}$ be the family of loss functions associated to $\mathcal{H}$ for the zero-one loss: $\mathcal{G} = \{(x, y) \mapsto 1_{h(x) \neq y} \colon h \in \mathcal{H}\}$. For any sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$ of elements in $\mathcal{X} \times \{-1, +1\}$, let $S_{\mathcal{X}}$ denote its projection over $\mathcal{X}$: $S_{\mathcal{X}} = (x_1, \ldots, x_m)$. Then, the following relation holds between the empirical Rademacher complexities of $\mathcal{G}$ and $\mathcal{H}$:*

$$\widehat{\mathfrak{R}}_S(\mathcal{G}) = \frac{1}{2}\widehat{\mathfrak{R}}_{S_{\mathcal{X}}}(\mathcal{H}). \tag{3.16}$$

Proof: For any sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$ of elements in $\mathcal{X} \times \{-1, +1\}$, by definition, the empirical Rademacher complexity of $\mathcal{G}$ can be written as:

$$\begin{aligned}
\widehat{\mathfrak{R}}_S(\mathcal{G}) &= \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i 1_{h(x_i) \neq y_i}\Big] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i \tfrac{1 - y_i h(x_i)}{2}\Big] \\
&= \frac{1}{2} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} -\sigma_i y_i h(x_i)\Big] \\
&= \frac{1}{2} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i h(x_i)\Big] = \frac{1}{2}\widehat{\mathfrak{R}}_{S_{\mathcal{X}}}(\mathcal{H}),
\end{aligned}$$

where we used the fact that $1_{h(x_i) \neq y_i} = (1 - y_i h(x_i))/2$ and the fact that for a fixed $y_i \in \{-1, +1\}$, σ_i and $-y_i \sigma_i$ are distributed in the same way. □

Note that the lemma implies, by taking expectations, that for any $m \geq 1$, $\mathfrak{R}_m(\mathcal{G}) = \frac{1}{2}\mathfrak{R}_m(\mathcal{H})$. These connections between the empirical and average Rademacher complexities can be used to derive generalization bounds for binary classification in terms of the Rademacher complexity of the hypothesis set $\mathcal{H}$.

Theorem 3.5 (Rademacher complexity bounds – binary classification) *Let $\mathcal{H}$ be a family of functions taking values in $\{-1, +1\}$ and let $\mathcal{D}$ be the distribution over the input space $\mathcal{X}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over a sample S of* *size m drawn according to $\mathcal{D}$, each of the following holds for any $h \in \mathcal{H}$:*

$$R(h) \leq \widehat{R}_S(h) + \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \tag{3.17}$$

$$\textit{and} \quad R(h) \leq \widehat{R}_S(h) + \widehat{\mathfrak{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{3.18}$$

Proof: The result follows immediately by theorem 3.3 and lemma 3.4. □

The theorem provides two generalization bounds for binary classification based on the Rademacher complexity. Note that the second bound, (3.18), is data-dependent: the empirical Rademacher complexity $\widehat{\mathfrak{R}}_S(\mathcal{H})$ is a function of the specific sample S drawn. Thus, this bound could be particularly informative if we could compute $\widehat{\mathfrak{R}}_S(\mathcal{H})$. But, how can we compute the empirical Rademacher complexity? Using again the fact that σ_i and $-\sigma_i$ are distributed in the same way, we can write

$$\widehat{\mathfrak{R}}_S(\mathcal{H}) = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} -\sigma_i h(x_i)\Big] = -\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\inf_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \sigma_i h(x_i)\Big].$$

Now, for a fixed value of $\boldsymbol{\sigma}$, computing $\inf_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \sigma_i h(x_i)$ is equivalent to an *empirical risk minimization* problem, which is known to be computationally hard for some hypothesis sets. Thus, in some cases, computing $\widehat{\mathfrak{R}}_S(\mathcal{H})$ could be computationally hard. In the next sections, we will relate the Rademacher complexity to combinatorial measures that are easier to compute and also of independent interest for their usefulness in the analysis of learning in many contexts.

3.2 Growth function

Here we will show how the Rademacher complexity can be bounded in terms of the *growth function*.

Definition 3.6 (Growth function) *The* growth function $\Pi_\mathcal{H} \colon \mathbb{N} \to \mathbb{N}$ *for a hypothesis set $\mathcal{H}$ is defined by:*

$$\forall m \in \mathbb{N},\ \Pi_\mathcal{H}(m) = \max_{\{x_1,\ldots,x_m\}\subseteq X} \Big|\big\{\big(h(x_1),\ldots,h(x_m)\big) \colon h \in \mathcal{H}\big\}\Big|. \tag{3.19}$$

In other words, $\Pi_\mathcal{H}(m)$ is the maximum number of distinct ways in which m points can be classified using hypotheses in $\mathcal{H}$. Each one of these distinct classifications is called a *dichotomy* and, thus, the growth function counts the number of dichotomies that are realized by the hypothesis. This provides another measure of the richness of the hypothesis set $\mathcal{H}$. However, unlike the Rademacher complexity, this measure does not depend on the distribution, it is purely combinatorial.

To relate the Rademacher complexity to the growth function, we will use Massart's lemma.

Theorem 3.7 (Massart's lemma) *Let $\mathcal{A} \subseteq \mathbb{R}^m$ be a finite set, with $r = \max_{\mathbf{x} \in \mathcal{A}} \|\mathbf{x}\|_2$, then the following holds:*

$$\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\frac{1}{m} \sup_{\mathbf{x} \in \mathcal{A}} \sum_{i=1}^{m} \sigma_i x_i\right] \leq \frac{r\sqrt{2 \log |\mathcal{A}|}}{m}, \tag{3.20}$$

where σ_is are independent uniform random variables taking values in $\{-1, +1\}$ and $x_1, \ldots, x_m$ are the components of vector $\mathbf{x}$.

Proof: The result follows immediately from the bound on the expectation of a maximum given by Corollary D.11 since the random variables $\sigma_i x_i$ are independent and each $\sigma_i x_i$ takes values in $[-|x_i|, |x_i|]$ with $\sqrt{\sum_{i=1}^m x_i^2} \leq r^2$. □

Using this result, we can now bound the Rademacher complexity in terms of the growth function.

Corollary 3.8 *Let $\mathcal{G}$ be a family of functions taking values in $\{-1, +1\}$. Then the following holds:*

$$\mathfrak{R}_m(\mathcal{G}) \leq \sqrt{\frac{2 \log \Pi_{\mathcal{G}}(m)}{m}}. \tag{3.21}$$

Proof: For a fixed sample $S = (x_1, \ldots, x_m)$, we denote by $\mathcal{G}_{|S}$ the set of vectors of function values $(g(x_1), \ldots, g(x_m))^\top$ where g is in $\mathcal{G}$. Since $g \in \mathcal{G}$ takes values in $\{-1, +1\}$, the norm of these vectors is bounded by $\sqrt{m}$. We can then apply Massart's lemma as follows:

$$\mathfrak{R}_m(\mathcal{G}) = \mathop{\mathbb{E}}_{S}\left[\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sup_{u \in \mathcal{G}_{|S}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i u_i\right]\right] \leq \mathop{\mathbb{E}}_{S}\left[\frac{\sqrt{m}\sqrt{2 \log |\mathcal{G}_{|S}|}}{m}\right].$$

By definition, $|\mathcal{G}_{|S}|$ is bounded by the growth function, thus,

$$\mathfrak{R}_m(\mathcal{G}) \leq \mathop{\mathbb{E}}_{S}\left[\frac{\sqrt{m}\sqrt{2 \log \Pi_{\mathcal{G}}(m)}}{m}\right] = \sqrt{\frac{2 \log \Pi_{\mathcal{G}}(m)}{m}},$$

which concludes the proof. □

Combining the generalization bound (3.17) of theorem 3.5 with corollary 3.8 yields immediately the following generalization bound in terms of the growth function.

Corollary 3.9 (Growth function generalization bound) *Let $\mathcal{H}$ be a family of functions taking values in $\{-1, +1\}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, for any $h \in \mathcal{H}$,*

$$R(h) \leq \widehat{R}_S(h) + \sqrt{\frac{2 \log \Pi_{\mathcal{H}}(m)}{m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}. \tag{3.22}$$

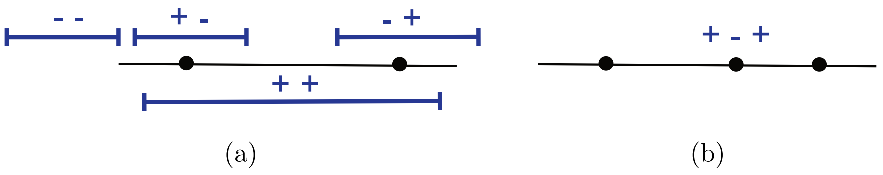


Figure 3.1
VC-dimension of intervals on the real line. (a) Any two points can be shattered. (b) No sample of three points can be shattered as the $(+, -, +)$ labeling cannot be realized.

Growth function bounds can be also derived directly (without using Rademacher complexity bounds first). The resulting bound is then the following:

$$\mathbb{P}\left[\left|R(h) - \widehat{R}_S(h)\right| > \epsilon\right] \leq 4\Pi_{\mathcal{H}}(2m) \exp\left(-\frac{m\epsilon^2}{8}\right), \tag{3.23}$$

which only differs from (3.22) by constants.

The computation of the growth function may not be always convenient since, by definition, it requires computing $\Pi_{\mathcal{H}}(m)$ for all $m \geq 1$. The next section introduces an alternative measure of the complexity of a hypothesis set $\mathcal{H}$ that is based instead on a single scalar, which will turn out to be in fact deeply related to the behavior of the growth function.

3.3 VC-dimension

Here, we introduce the notion of *VC-dimension* (Vapnik-Chervonenkis dimension). The VC-dimension is also a purely combinatorial notion but it is often easier to compute than the growth function (or the Rademacher Complexity). As we shall see, the VC-dimension is a key quantity in learning and is directly related to the growth function.

To define the VC-dimension of a hypothesis set $\mathcal{H}$, we first introduce the concept of *shattering*. Recall from the previous section, that given a hypothesis set $\mathcal{H}$, a dichotomy of a set S is one of the possible ways of labeling the points of S using a hypothesis in $\mathcal{H}$. A set S of $m \geq 1$ points is said to be shattered by a hypothesis set $\mathcal{H}$ when $\mathcal{H}$ realizes all possible dichotomies of S, that is when $\Pi_{\mathcal{H}}(m) = 2^m$.

Definition 3.10 (VC-dimension) *The VC-dimension of a hypothesis set $\mathcal{H}$ is the size of the largest set that can be shattered by $\mathcal{H}$:*

$$\text{VCdim}(\mathcal{H}) = \max\{m \colon \Pi_{\mathcal{H}}(m) = 2^m\}. \tag{3.24}$$

Note that, by definition, if $\text{VCdim}(\mathcal{H}) = d$, there exists a set of size d that can be shattered. However, this does not imply that all sets of size d or less are shattered and, in fact, this is typically not the case.

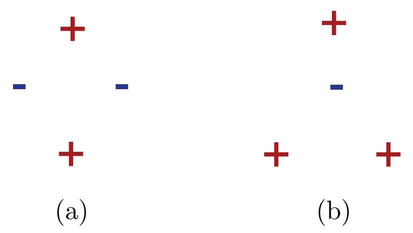


Figure 3.2
Unrealizable dichotomies for four points using hyperplanes in $\mathbb{R}^2$. (a) All four points lie on the convex hull. (b) Three points lie on the convex hull while the remaining point is interior.

To further illustrate this notion, we will examine a series of examples of hypothesis sets and will determine the VC-dimension in each case. To compute the VC-dimension we will typically show a lower bound for its value and then a matching upper bound. To give a lower bound d for $\text{VCdim}(\mathcal{H})$, it suffices to show that a set S of cardinality d can be shattered by $\mathcal{H}$. To give an upper bound, we need to prove that no set S of cardinality $d+1$ can be shattered by $\mathcal{H}$, which is typically more difficult.

Example 3.11 (Intervals on the real line) Our first example involves the hypothesis class of intervals on the real line. It is clear that the VC-dimension is at least two, since all four dichotomies $(+,+), (-,-), (+,-), (-,+)$ can be realized, as illustrated in figure 3.1(a). In contrast, by the definition of intervals, no set of three points can be shattered since the $(+,-,+)$ labeling cannot be realized. Hence, $\text{VCdim}(\text{intervals in } \mathbb{R}) = 2$.

Example 3.12 (Hyperplanes) Consider the set of hyperplanes in $\mathbb{R}^2$. We first observe that any three non-collinear points in $\mathbb{R}^2$ can be shattered. To obtain the first three dichotomies, we choose a hyperplane that has two points on one side and the third point on the opposite side. To obtain the fourth dichotomy we have all three points on the same side of the hyperplane. The remaining four dichotomies are realized by simply switching signs. Next, we show that four points cannot be shattered by considering two cases: (i) the four points lie on the convex hull defined by the four points, and (ii) three of the four points lie on the convex hull and the remaining point is internal. In the first case, a positive labeling for one diagonal pair and a negative labeling for the other diagonal pair cannot be realized, as illustrated in figure 3.2(a). In the second case, a labeling which is positive for the points on the convex hull and negative for the interior point cannot be realized, as illustrated in figure 3.2(b). Hence, $\text{VCdim}(\text{hyperplanes in } \mathbb{R}^2) = 3$.

More generally in $\mathbb{R}^d$, we derive a lower bound by starting with a set of $d+1$ points in $\mathbb{R}^d$, setting $\mathbf{x}_0$ to be the origin and defining $\mathbf{x}_i$, for $i \in \{1, \ldots, d\}$, as the point whose ith coordinate is 1 and all others are 0. Let $y_0, y_1, \ldots, y_d \in \{-1, +1\}$ be an arbitrary set of labels for $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_d$. Let $\mathbf{w}$ be the vector whose ith coordinate is y_i. Then the classifier defined by the hyperplane of equation $\mathbf{w} \cdot \mathbf{x} + \frac{y_0}{2} = 0$ shatters $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_d$ since for any $i \in \{0, \ldots, d\}$,

$$\operatorname{sgn}\Big(\mathbf{w} \cdot \mathbf{x}_i + \frac{y_0}{2}\Big) = \operatorname{sgn}\Big(y_i + \frac{y_0}{2}\Big) = y_i. \tag{3.25}$$

To obtain an upper bound, it suffices to show that no set of $d+2$ points can be shattered by halfspaces. To prove this, we will use the following general theorem.

Theorem 3.13 (Radon's theorem) Any set $\mathcal{X}$ of $d+2$ points in $\mathbb{R}^d$ can be partitioned into two subsets $\mathcal{X}_1$ and $\mathcal{X}_2$ such that the convex hulls of $\mathcal{X}_1$ and $\mathcal{X}_2$ intersect.

Proof: Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{d+2}\} \subset \mathbb{R}^d$. The following is a system of $d+1$ linear equations in $\alpha_1, \ldots, \alpha_{d+2}$:

$$\sum_{i=1}^{d+2} \alpha_i \mathbf{x}_i = 0 \qquad \text{and} \qquad \sum_{i=1}^{d+2} \alpha_i = 0, \tag{3.26}$$

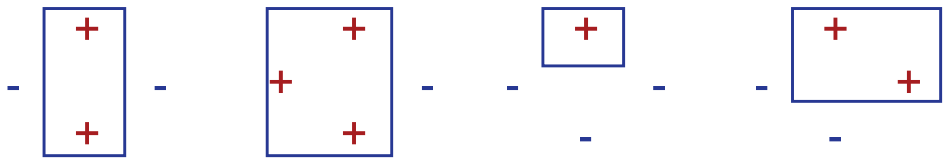
since the first equality leads to d equations, one for each component. The number of unknowns, $d+2$, is larger than the number of equations, $d+1$, therefore the system admits a non-zero solution $\beta_1, \ldots, \beta_{d+2}$. Since $\sum_{i=1}^{d+2} \beta_i = 0$, both $\mathcal{I}_1 = \{i \in [d+2] \colon \beta_i > 0\}$ and $\mathcal{I}_2 = \{i \in [d+2] \colon \beta_i \leq 0\}$ are non-empty sets and $\mathcal{X}_1 = \{\mathbf{x}_i \colon i \in \mathcal{I}_1\}$ and $\mathcal{X}_2 = \{\mathbf{x}_i \colon i \in \mathcal{I}_2\}$ form a partition of $\mathcal{X}$. By the last equation of (3.26), $\sum_{i \in \mathcal{I}_1} \beta_i = -\sum_{i \in \mathcal{I}_2} \beta_i$. Let $\beta = \sum_{i \in \mathcal{I}_1} \beta_i$. Then, the first part of (3.26) implies

$$\sum_{i \in \mathcal{I}_1} \frac{\beta_i}{\beta} \mathbf{x}_i = \sum_{i \in \mathcal{I}_2} \frac{-\beta_i}{\beta} \mathbf{x}_i,$$

with $\sum_{i \in \mathcal{I}_1} \frac{\beta_i}{\beta} = \sum_{i \in \mathcal{I}_2} \frac{-\beta_i}{\beta} = 1$, and $\frac{\beta_i}{\beta} \geq 0$ for $i \in \mathcal{I}_1$ and $\frac{-\beta_i}{\beta} \geq 0$ for $i \in \mathcal{I}_2$. By definition of the convex hulls (B.6), this implies that $\sum_{i \in \mathcal{I}_1} \frac{\beta_i}{\beta} \mathbf{x}_i$ belongs both to the convex hull of $\mathcal{X}_1$ and to that of $\mathcal{X}_2$. □

Now, let $\mathcal{X}$ be a set of $d+2$ points. By Radon's theorem, it can be partitioned into two sets $\mathcal{X}_1$ and $\mathcal{X}_2$ such that their convex hulls intersect. Observe that when two sets of points $\mathcal{X}_1$ and $\mathcal{X}_2$ are separated by a hyperplane, their convex hulls are also separated by that hyperplane. Thus, $\mathcal{X}_1$ and $\mathcal{X}_2$ cannot be separated by a hyperplane and $\mathcal{X}$ is not shattered. Combining our lower and upper bounds, we have proven that $\text{VCdim}(\text{hyperplanes in } \mathbb{R}^d) = d+1$.

Example 3.14 (Axis-aligned Rectangles) We first show that the VC-dimension is at least four, by considering four points in a diamond pattern. Then, it is clear that all 16 dichotomies can be realized, some of which are illustrated in figure 3.3(a). In contrast, for any set of five distinct points, if we construct the minimal axis-aligned rectangle containing these points, one of the five points is in the interior of


(a)

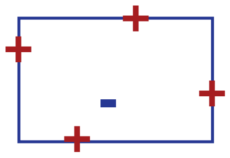

(b)

Figure 3.3
VC-dimension of axis-aligned rectangles. (a) Examples of realizable dichotomies for four points in a diamond pattern. (b) No sample of five points can be realized if the interior point and the remaining points have opposite labels.

this rectangle. Imagine that we assign a negative label to this interior point and a positive label to each of the remaining four points, as illustrated in figure 3.3(b). There is no axis-aligned rectangle that can realize this labeling. Hence, no set of five distinct points can be shattered and VCdim(axis-aligned rectangles) $= 4$.

Example 3.15 (Convex Polygons) We focus on the class of convex d-gons in the plane. To get a lower bound, we show that any set of $2d+1$ points can be shattered. To do this, we select $2d+1$ points that lie on a circle, and for a particular labeling, if there are more negative than positive labels, then the points with the positive labels are used as the polygon's vertices, as in figure 3.4(a). Otherwise, the tangents of the negative points serve as the edges of the polygon, as shown in (3.4)(b). To derive an upper bound, it can be shown that choosing points on the circle maximizes the number of possible dichotomies, and thus VCdim(convex d-gons) $= 2d + 1$. Note also that VCdim(convex polygons) $= +\infty$.

Example 3.16 (Sine Functions) The previous examples could suggest that the VC-dimension of $\mathcal{H}$ coincides with the number of free parameters defining $\mathcal{H}$. For example, the number of parameters defining hyperplanes matches their VC-dimension. However, this does not hold in general. Several of the exercises in this chapter illustrate this fact. The following provides a striking example from this point of view. Consider the following family of sine functions: $\{t \mapsto \sin(\omega t) \colon \omega \in \mathbb{R}\}$. One instance of this function class is shown in figure 3.5. These sine functions can be

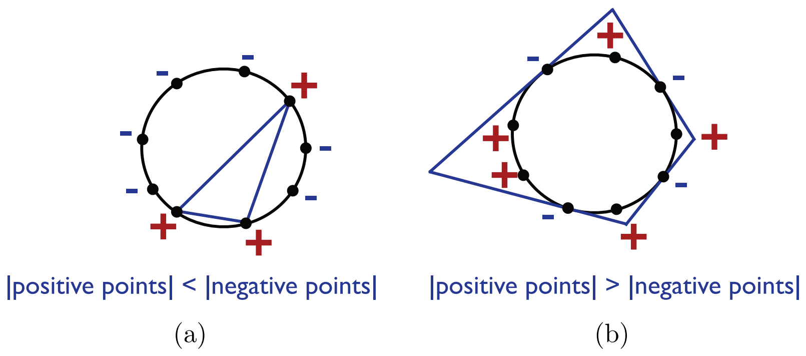

|positive points| < |negative points|
|positive points| > |negative points|
(a)
(b)

Figure 3.4
Convex d-gons in the plane can shatter $2d+1$ points. (a) d-gon construction when there are more negative labels. (b) d-gon construction when there are more positive labels.

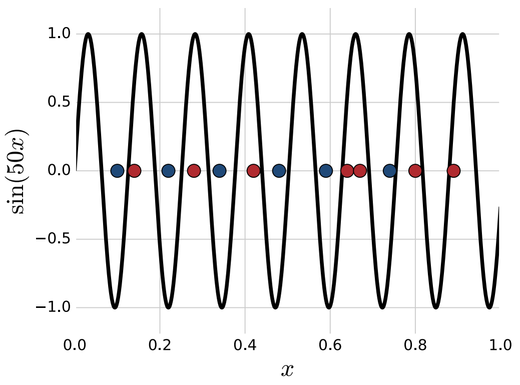


Figure 3.5
An example of a sine function (with $\omega = 50$) used for classification.

used to classify the points on the real line: a point is labeled positively if it is above the curve, negatively otherwise. Although this family of sine functions is defined via a single parameter, ω, it can be shown that VCdim(sine functions) $= +\infty$ (exercise 3.20).

The VC-dimension of many other hypothesis sets can be determined or upper-bounded in a similar way (see this chapter's exercises). In particular, the VC-dimension of any vector space of dimension $r < \infty$ can be shown to be at most r (exercise 3.19). The next result, known as *Sauer's lemma*, clarifies the connection between the notions of growth function and VC-dimension.

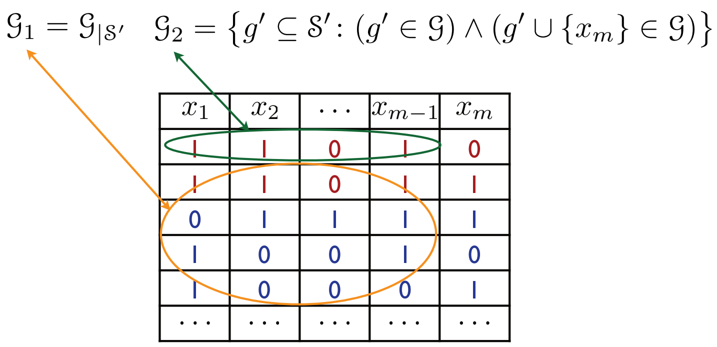

$\mathcal{G}_1 = \mathcal{G}_{|\mathcal{S}'}$
$\mathcal{G}_2 = \{g' \subseteq \mathcal{S}' \colon (g' \in \mathcal{G}) \wedge (g' \cup \{x_m\} \in \mathcal{G})\}$
x_1 x_2 $\cdots$ x_{m-1} x_m
1 1 0 1 0
1 1 0 1 1
0 1 1 1 1
1 0 0 1 0
1 0 0 0 1
$\cdots$ $\cdots$ $\cdots$ $\cdots$ $\cdots$

Figure 3.6
Illustration of how $\mathcal{G}_1$ and $\mathcal{G}_2$ are constructed in the proof of Sauer's lemma.

Theorem 3.17 (Sauer's lemma) *Let $\mathcal{H}$ be a hypothesis set with* $\mathrm{VCdim}(\mathcal{H}) = d$. *Then, for all $m \in \mathbb{N}$, the following inequality holds:*

$$\Pi_{\mathcal{H}}(m) \leq \sum_{i=0}^{d} \binom{m}{i}. \tag{3.27}$$

Proof: The proof is by induction on $m + d$. The statement clearly holds for $m = 1$ and $d = 0$ or $d = 1$. Now, assume that it holds for $(m-1, d-1)$ and $(m-1, d)$. Fix a set $\mathcal{S} = \{x_1, \ldots, x_m\}$ with $\Pi_{\mathcal{H}}(m)$ dichotomies and let $\mathcal{G} = \mathcal{H}_{|\mathcal{S}}$ be the set of concepts $\mathcal{H}$ induced by restriction to $\mathcal{S}$.

Now consider the following families over $\mathcal{S}' = \{x_1, \ldots, x_{m-1}\}$. We define $\mathcal{G}_1 = \mathcal{G}_{|\mathcal{S}'}$ as the set of concepts $\mathcal{H}$ induced by restriction to S'. Next, by identifying each concept as the set of points (in $\mathcal{S}'$ or $\mathcal{S}$) for which it is non-zero, we can define $_2$ as

$$\mathcal{G}_2 = \{g' \subseteq \mathcal{S}' \colon (g' \in \mathcal{G}) \wedge (g' \cup \{x_m\} \in \mathcal{G})\}.$$

Since $g' \subseteq \mathcal{S}'$, $g' \in \mathcal{G}$ means that without adding x_m it is a concept of $\mathcal{G}$. Further, the constraint $g' \cup \{x_m\} \in \mathcal{G}$ means that adding x_m to g' also makes it a concept of $\mathcal{G}$. The construction of $\mathcal{G}_1$ and $\mathcal{G}_2$ is illustrated pictorially in figure 3.6. Given our definitions of $\mathcal{G}_1$ and $\mathcal{G}_2$, observe that $|\mathcal{G}_1| + |\mathcal{G}_2| = |\mathcal{G}|$.

Since $\mathrm{VCdim}(\mathcal{G}_1) \leq \mathrm{VCdim}(\mathcal{G}) \leq d$, then by definition of the growth function and using the induction hypothesis,

$$|\mathcal{G}_1| \leq \Pi_{\mathcal{G}_1}(m-1) \leq \sum_{i=0}^{d} \binom{m-1}{i}.$$

Further, by definition of $\mathcal{G}_2$, if a set $\mathcal{Z} \subseteq \mathcal{S}'$ is shattered by $\mathcal{G}_2$, then the set $\mathcal{Z} \cup \{x_m\}$ is shattered by $\mathcal{G}$. Hence,

$$\mathrm{VCdim}(\mathcal{G}_2) \leq \mathrm{VCdim}(\mathcal{G}) - 1 = d - 1,$$

and by definition of the growth function and using the induction hypothesis,

$$|\mathcal{G}_2| \leq \Pi_{\mathcal{G}_2}(m-1) \leq \sum_{i=0}^{d-1} \binom{m-1}{i}.$$

Thus,

$$|\mathcal{G}| = |\mathcal{G}_1| + |\mathcal{G}_2| \leq \sum_{i=0}^{d} \tbinom{m-1}{i} + \sum_{i=0}^{d-1} \tbinom{m-1}{i} = \sum_{i=0}^{d} \tbinom{m-1}{i} + \tbinom{m-1}{i-1} = \sum_{i=0}^{d} \tbinom{m}{i},$$

which completes the inductive proof. □

The significance of Sauer's lemma can be seen by corollary 3.18, which remarkably shows that growth function only exhibits two types of behavior: either $\text{VCdim}(\mathcal{H}) = d < +\infty$, in which case $\Pi_{\mathcal{H}}(m) = O(m^d)$, or $\text{VCdim}(\mathcal{H}) = +\infty$, in which case $\Pi_{\mathcal{H}}(m) = 2^m$.

Corollary 3.18 *Let $\mathcal{H}$ be a hypothesis set with* $\text{VCdim}(\mathcal{H}) = d$. *Then for all $m \geq d$,*

$$\Pi_{\mathcal{H}}(m) \leq \left(\frac{em}{d}\right)^d = O(m^d). \tag{3.28}$$

Proof: The proof begins by using Sauer's lemma. The first inequality multiplies each summand by a factor that is greater than or equal to one since $m \geq d$, while the second inequality adds non-negative summands to the summation.

$$\begin{aligned}
\Pi_{\mathcal{H}}(m) &\leq \sum_{i=0}^{d} \binom{m}{i} \\
&\leq \sum_{i=0}^{d} \binom{m}{i} \left(\frac{m}{d}\right)^{d-i} \\
&\leq \sum_{i=0}^{m} \binom{m}{i} \left(\frac{m}{d}\right)^{d-i} \\
&= \left(\frac{m}{d}\right)^{d} \sum_{i=0}^{m} \binom{m}{i} \left(\frac{d}{m}\right)^{i} \\
&= \left(\frac{m}{d}\right)^{d} \left(1 + \frac{d}{m}\right)^{m} \leq \left(\frac{m}{d}\right)^{d} e^d.
\end{aligned}$$

After simplifying the expression using the binomial theorem, the final inequality follows using the general inequality $(1 - x) \leq e^{-x}$. □

The explicit relationship just formulated between VC-dimension and the growth function combined with corollary 3.9 leads immediately to the following generalization bounds based on the VC-dimension.

Corollary 3.19 (VC-dimension generalization bounds) *Let $\mathcal{H}$ be a family of functions taking values in $\{-1, +1\}$ with VC-dimension d. Then, for any $\delta > 0$, with proba-*

bility at least $1-\delta$, *the following holds for all* $h \in \mathcal{H}$:

$$R(h) \leq \widehat{R}_S(h) + \sqrt{\frac{2d \log \frac{em}{d}}{m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}. \tag{3.29}$$

Thus, the form of this generalization bound is

$$R(h) \leq \widehat{R}_S(h) + O\left(\sqrt{\frac{\log(m/d)}{(m/d)}}\right), \tag{3.30}$$

which emphasizes the importance of the ratio m/d for generalization. The theorem provides another instance of Occam's razor principle where simplicity is measured in terms of smaller VC-dimension.

VC-dimension bounds can be derived directly without using an intermediate Rademacher complexity bound, as for (3.23): combining Sauer's lemma with (3.23) leads to the following high-probability bound

$$R(h) \leq \widehat{R}_S(h) + \sqrt{\frac{8d \log \frac{2em}{d} + 8 \log \frac{4}{\delta}}{m}},$$

which has the general form of (3.30). The log factor plays only a minor role in these bounds. A finer analysis can be used in fact to eliminate that factor.

3.4 Lower bounds

In the previous section, we presented several upper bounds on the generalization error. In contrast, this section provides lower bounds on the generalization error of any learning algorithm in terms of the VC-dimension of the hypothesis set used.

These lower bounds are shown by finding for any algorithm a 'bad' distribution. Since the learning algorithm is arbitrary, it will be difficult to specify that particular distribution. Instead, it suffices to prove its existence non-constructively. At a high level, the proof technique used to achieve this is the *probabilistic method* of Paul Erdös. In the context of the following proofs, first a lower bound is given on the expected error over the parameters defining the distributions. From that, the lower bound is shown to hold for at least one set of parameters, that is one distribution.

Theorem 3.20 (Lower bound, realizable case) *Let* $\mathcal{H}$ *be a hypothesis set with VC-dimension* $d > 1$. *Then, for any* $m \geq 1$ *and any learning algorithm* $\mathcal{A}$, *there exist a distribution* $\mathcal{D}$ *over* $\mathcal{X}$ *and a target function* $f \in \mathcal{H}$ *such that*

$$\underset{S \sim \mathcal{D}^m}{\mathbb{P}}\left[R_{\mathcal{D}}(h_S, f) > \frac{d-1}{32m}\right] \geq 1/100. \tag{3.31}$$

Proof: Let $\overline{\mathcal{X}} = \{x_0, x_1, \ldots, x_{d-1}\} \subseteq \mathcal{X}$ be a set that is shattered by $\mathcal{H}$. For any $\epsilon > 0$, we choose $\mathcal{D}$ such that its support is reduced to $\overline{\mathcal{X}}$ and so that one point (x_0)

has very high probability $(1 - 8\epsilon)$, with the rest of the probability mass distributed uniformly among the other points:

$$\underset{\mathcal{D}}{\mathbb{P}}[x_0] = 1 - 8\epsilon \quad \text{and} \quad \forall i \in [d-1], \underset{\mathcal{D}}{\mathbb{P}}[x_i] = \frac{8\epsilon}{d-1}. \tag{3.32}$$

With this definition, most samples would contain x_0 and, since $\mathcal{X}$ is shattered, $\mathcal{A}$ can essentially do no better than tossing a coin when determining the label of a point x_i not falling in the training set.

We assume without loss of generality that $\mathcal{A}$ makes no error on x_0. For a sample S, we let $\overline{S}$ denote the set of its elements falling in $\{x_1, \ldots, x_{d-1}\}$, and let $\mathcal{S}$ be the set of samples S of size m such that $|\overline{S}| \leq (d-1)/2$. Now, fix a sample $S \in \mathcal{S}$, and consider the uniform distribution $\mathcal{U}$ over all labelings $f \colon \overline{\mathcal{X}} \to \{0, 1\}$, which are all in $\mathcal{H}$ since the set is shattered. Then, the following lower bound holds:

$$\begin{aligned}
\underset{f \sim \mathcal{U}}{\mathbb{E}}[R_{\mathcal{D}}(h_S, f)] &= \sum_f \sum_{x \in \overline{\mathcal{X}}} 1_{h_S(x) \neq f(x)} \, \mathbb{P}[x] \, \mathbb{P}[f] \\
&\geq \sum_f \sum_{x \notin \overline{S}} 1_{h_S(x) \neq f(x)} \, \mathbb{P}[x] \, \mathbb{P}[f] \\
&= \sum_{x \notin \overline{S}} \Big(\sum_f 1_{h_S(x) \neq f(x)} \, \mathbb{P}[f] \Big) \mathbb{P}[x] \\
&= \frac{1}{2} \sum_{x \notin \overline{S}} \mathbb{P}[x] \geq \frac{1}{2} \frac{d-1}{2} \frac{8\epsilon}{d-1} = 2\epsilon.
\end{aligned} \tag{3.33}$$

The first lower bound holds because we remove non-negative terms from the summation when we only consider $x \notin \overline{S}$ instead of all x in $\overline{\mathcal{X}}$. After rearranging terms, the subsequent equality holds since we are taking an expectation over $f \in \mathcal{H}$ with uniform weight on each f and $\mathcal{H}$ shatters $\overline{\mathcal{X}}$. The final lower bound holds due to the definitions of $\mathcal{D}$ and $\overline{S}$, the latter which implies that $|\overline{\mathcal{X}} - \overline{S}| \geq (d-1)/2$.

Since (3.33) holds for all $S \in \mathcal{S}$, it also holds in expectation over all $S \in \mathcal{S}$: $\mathbb{E}_{S \in \mathcal{S}}\big[\mathbb{E}_{f \sim \mathcal{U}}[R_{\mathcal{D}}(h_S, f)]\big] \geq 2\epsilon$. By Fubini's theorem, the expectations can be permuted, thus,

$$\underset{f \sim \mathcal{U}}{\mathbb{E}} \Big[\underset{S \in \mathcal{S}}{\mathbb{E}}[R_{\mathcal{D}}(h_S, f)] \Big] \geq 2\epsilon. \tag{3.34}$$

This implies that $\mathbb{E}_{S \in \mathcal{S}}[R_{\mathcal{D}}(h_S, f_0)] \geq 2\epsilon$ for at least one labeling $f_0 \in \mathcal{H}$. Decomposing this expectation into two parts and using $R_{\mathcal{D}}(h_S, f_0) \leq \mathbb{P}_{\mathcal{D}}[\overline{\mathcal{X}} - \{x_0\}]$, we obtain:

$$\begin{aligned}
\underset{S \in \mathcal{S}}{\mathbb{E}}[R_{\mathcal{D}}(h_S, f_0)] &= \sum_{S \colon R_{\mathcal{D}}(h_S, f_0) \geq \epsilon} R_{\mathcal{D}}(h_S, f_0) \, \mathbb{P}[R_{\mathcal{D}}(h_S, f_0)] + \sum_{S \colon R_{\mathcal{D}}(h_S, f_0) < \epsilon} R_{\mathcal{D}}(h_S, f_0) \, \mathbb{P}[R_{\mathcal{D}}(h_S, f_0)] \\
&\leq \underset{\mathcal{D}}{\mathbb{P}}[\overline{\mathcal{X}} - \{x_0\}] \underset{S \in \mathcal{S}}{\mathbb{P}}[R_{\mathcal{D}}(h_S, f_0) \geq \epsilon] + \epsilon \underset{S \in \mathcal{S}}{\mathbb{P}}[R_{\mathcal{D}}(h_S, f_0) < \epsilon] \\
&\leq 8\epsilon \underset{S \in \mathcal{S}}{\mathbb{P}}[R_{\mathcal{D}}(h_S, f_0) \geq \epsilon] + \epsilon \big(1 - \underset{S \in \mathcal{S}}{\mathbb{P}}[R_{\mathcal{D}}(h_S, f_0) \geq \epsilon]\big).
\end{aligned}$$

Collecting terms in $\mathbb{P}_{S\in\mathcal{S}}[R_{\mathcal{D}}(h_S, f_0) \geq \epsilon]$ yields

$$\underset{S\in\mathcal{S}}{\mathbb{P}}[R_{\mathcal{D}}(h_S, f_0) \geq \epsilon] \geq \frac{1}{7\epsilon}(2\epsilon - \epsilon) = \frac{1}{7}. \tag{3.35}$$

Thus, the probability over all samples S (not necessarily in $\mathcal{S}$) can be lower bounded as

$$\underset{S}{\mathbb{P}}[R_{\mathcal{D}}(h_S, f_0) \geq \epsilon] \geq \underset{S\in\mathcal{S}}{\mathbb{P}}[R_{\mathcal{D}}(h_S, f_0) \geq \epsilon]\,\mathbb{P}[\mathcal{S}] \geq \frac{1}{7}\,\mathbb{P}[\mathcal{S}]. \tag{3.36}$$

This leads us to find a lower bound for $\mathbb{P}[\mathcal{S}]$. By the multiplicative Chernoff bound (Theorem D.4), for any $\gamma > 0$, the probability that more than $(d-1)/2$ points are drawn in a sample of size m verifies:

$$1 - \mathbb{P}[\mathcal{S}] = \mathbb{P}[S_m \geq 8\epsilon m(1+\gamma)] \leq e^{-8\epsilon m \frac{\gamma^2}{3}}. \tag{3.37}$$

Therefore, for $\epsilon = (d-1)/(32m)$ and $\gamma = 1$,

$$\mathbb{P}[S_m \geq \tfrac{d-1}{2}] \leq e^{-(d-1)/12} \leq e^{-1/12} \leq 1 - 7\delta, \tag{3.38}$$

for $\delta \leq .01$. Thus $\mathbb{P}[\mathcal{S}] \geq 7\delta$ and $\mathbb{P}_S[R_{\mathcal{D}}(h_S, f_0) \geq \epsilon] \geq \delta$. □

The theorem shows that for any algorithm $\mathcal{A}$, there exists a 'bad' distribution over $\mathcal{X}$ and a target function f for which the error of the hypothesis returned by $\mathcal{A}$ is a constant times $\frac{d}{m}$ with some constant probability. This further demonstrates the key role played by the VC-dimension in learning. The result implies in particular that PAC-learning in the realizable case is not possible when the VC-dimension is infinite.

Note that the proof shows a stronger result than the statement of the theorem: the distribution $\mathcal{D}$ is selected independently of the algorithm $\mathcal{A}$. We now present a theorem giving a lower bound in the non-realizable case. The following two lemmas will be needed for the proof.

Lemma 3.21 *Let α be a uniformly distributed random variable taking values in $\{\alpha_-, \alpha_+\}$, where $\alpha_- = \frac{1}{2} - \frac{\epsilon}{2}$ and $\alpha_+ = \frac{1}{2} + \frac{\epsilon}{2}$, and let S be a sample of $m \geq 1$ random variables $X_1, \ldots, X_m$ taking values in $\{0, 1\}$ and drawn i.i.d. according to the distribution $\mathcal{D}_\alpha$ defined by $\mathbb{P}_{\mathcal{D}_\alpha}[X = 1] = \alpha$. Let h be a function from $\mathcal{X}^m$ to $\{\alpha_-, \alpha_+\}$, then the following holds:*

$$\underset{\alpha}{\mathbb{E}}\Big[\underset{S\sim\mathcal{D}_\alpha^m}{\mathbb{P}}[h(S) \neq \alpha]\Big] \geq \Phi(2\lceil m/2\rceil, \epsilon), \tag{3.39}$$

where $\Phi(m, \epsilon) = \frac{1}{4}\Big(1 - \sqrt{1 - \exp\big(-\frac{m\epsilon^2}{1-\epsilon^2}\big)}\Big)$ for all m and ϵ.

Proof: The lemma can be interpreted in terms of an experiment with two coins with biases α_- and α_+. It implies that for a discriminant rule $h(S)$ based on a sample S drawn from $\mathcal{D}_{\alpha_-}$ or $\mathcal{D}_{\alpha_+}$, to determine which coin was tossed, the sample size m must be at least $\Omega(1/\epsilon^2)$. The proof is left as an exercise (exercise D.3). □

We will make use of the fact that for any fixed ϵ the function $m \mapsto \Phi(m, x)$ is convex, which is not hard to establish.

Lemma 3.22 *Let Z be a random variable taking values in $[0, 1]$. Then, for any $\gamma \in [0, 1)$,*

$$\mathbb{P}[z > \gamma] \geq \frac{\mathbb{E}[Z] - \gamma}{1 - \gamma} > \mathbb{E}[Z] - \gamma. \tag{3.40}$$

Proof: Since the values taken by Z are in $[0, 1]$,

$$\begin{aligned}\mathbb{E}[Z] &= \sum_{z \leq \gamma} \mathbb{P}[Z = z] z + \sum_{z > \gamma} \mathbb{P}[Z = z] z \\ &\leq \sum_{z \leq \gamma} \mathbb{P}[Z = z] \gamma + \sum_{z > \gamma} \mathbb{P}[Z = z] \\ &= \gamma \, \mathbb{P}[Z \leq \gamma] + \mathbb{P}[Z > \gamma] \\ &= \gamma (1 - \mathbb{P}[Z > \gamma]) + \mathbb{P}[Z > \gamma] \\ &= (1 - \gamma) \, \mathbb{P}[Z > \gamma] + \gamma,\end{aligned}$$

which concludes the proof. □

Theorem 3.23 (Lower bound, non-realizable case) *Let $\mathcal{H}$ be a hypothesis set with VC-dimension $d > 1$. Then, for any $m \geq 1$ and any learning algorithm $\mathcal{A}$, there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$ such that:*

$$\mathbb{P}_{S \sim \mathcal{D}^m}\left[R_{\mathcal{D}}(h_S) - \inf_{h \in \mathcal{H}} R_{\mathcal{D}}(h) > \sqrt{\frac{d}{320m}}\right] \geq 1/64. \tag{3.41}$$

Equivalently, for any learning algorithm, the sample complexity verifies

$$m \geq \frac{d}{320\epsilon^2}. \tag{3.42}$$

Proof: Let $\overline{\mathcal{X}} = \{x_1, \ldots, x_d\} \subseteq \mathcal{X}$ be a set shattered by $\mathcal{H}$. For any $\alpha \in [0, 1]$ and any vector $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_d)^\top \in \{-1, +1\}^d$, we define a distribution $\mathcal{D}_{\boldsymbol{\sigma}}$ with support $\overline{\mathcal{X}} \times \{0, 1\}$ as follows:

$$\forall i \in [d], \quad \mathbb{P}_{\mathcal{D}_{\boldsymbol{\sigma}}}[(x_i, 1)] = \frac{1}{d}\Big(\frac{1}{2} + \frac{\sigma_i \alpha}{2}\Big). \tag{3.43}$$

Thus, the label of each point x_i, $i \in [d]$, follows the distribution $\mathbb{P}_{\mathcal{D}_{\boldsymbol{\sigma}}}[\cdot | x_i]$, that of a biased coin where the bias is determined by the sign of σ_i and the magnitude of α. To determine the most likely label of each point x_i, the learning algorithm will therefore need to estimate $\mathbb{P}_{\mathcal{D}_{\boldsymbol{\sigma}}}[1 | x_i]$ with an accuracy better than α. To make this further difficult, α and $\boldsymbol{\sigma}$ will be selected based on the algorithm, requiring, as in lemma 3.21, $\Omega(1/\alpha^2)$ instances of each point x_i in the training sample.

Clearly, the Bayes classifier $h^*_{\mathcal{D}_{\boldsymbol{\sigma}}}$ is defined by $h^*_{\mathcal{D}_{\boldsymbol{\sigma}}}(x_i) = \operatorname{argmax}_{y \in \{0,1\}} \mathbb{P}[y|x_i] = 1_{\sigma_i > 0}$ for all $i \in [d]$. $h^*_{\mathcal{D}_{\boldsymbol{\sigma}}}$ is in $\mathcal{H}$ since $\overline{\mathcal{X}}$ is shattered. For all $h \in \mathcal{H}$,

$$R_{\mathcal{D}_{\boldsymbol{\sigma}}}(h) - R_{\mathcal{D}_{\boldsymbol{\sigma}}}(h^*_{\mathcal{D}_{\boldsymbol{\sigma}}}) = \frac{1}{d} \sum_{x \in \overline{\mathcal{X}}} \Big(\frac{\alpha}{2} + \frac{\alpha}{2}\Big) 1_{h(x) \neq h^*_{\mathcal{D}_{\boldsymbol{\sigma}}}(x)} = \frac{\alpha}{d} \sum_{x \in \overline{\mathcal{X}}} 1_{h(x) \neq h^*_{\mathcal{D}_{\boldsymbol{\sigma}}}(x)}. \tag{3.44}$$

Let h_S denote the hypothesis returned by the learning algorithm $\mathcal{A}$ after receiving a labeled sample S drawn according to $\mathcal{D}_{\boldsymbol{\sigma}}$. We will denote by $|S|_x$ the number of occurrences of a point x in S. Let $\mathcal{U}$ denote the uniform distribution over $\{-1, +1\}^d$. Then, in view of (3.44), the following holds:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{\boldsymbol{\sigma} \sim \mathcal{U} \\ S \sim \mathcal{D}^m_{\boldsymbol{\sigma}}}} \Big[\frac{1}{\alpha}\big[R_{\mathcal{D}_{\boldsymbol{\sigma}}}(h_S) - R_{\mathcal{D}_{\boldsymbol{\sigma}}}(h^*_{\mathcal{D}_{\boldsymbol{\sigma}}})\big]\Big] \\
&= \frac{1}{d} \sum_{x \in \overline{\mathcal{X}}} \mathop{\mathbb{E}}_{\substack{\boldsymbol{\sigma} \sim \mathcal{U} \\ S \sim \mathcal{D}^m_{\boldsymbol{\sigma}}}} \Big[1_{h_S(x) \neq h^*_{\mathcal{D}_{\boldsymbol{\sigma}}}(x)}\Big] \\
&= \frac{1}{d} \sum_{x \in \overline{\mathcal{X}}} \mathop{\mathbb{E}}_{\boldsymbol{\sigma} \sim \mathcal{U}} \Big[\mathop{\mathbb{P}}_{S \sim \mathcal{D}^m_{\boldsymbol{\sigma}}} \big[h_S(x) \neq h^*_{\mathcal{D}_{\boldsymbol{\sigma}}}(x)\big]\Big] \\
&= \frac{1}{d} \sum_{x \in \overline{\mathcal{X}}} \sum_{n=0}^{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma} \sim \mathcal{U}} \Big[\mathop{\mathbb{P}}_{S \sim \mathcal{D}^m_{\boldsymbol{\sigma}}} \big[h_S(x) \neq h^*_{\mathcal{D}_{\boldsymbol{\sigma}}}(x) \,\big|\, |S|_x = n\big] \, \mathbb{P}[|S|_x = n]\Big] \\
&\geq \frac{1}{d} \sum_{x \in \overline{\mathcal{X}}} \sum_{n=0}^{m} \Phi(n+1, \alpha) \, \mathbb{P}[|S|_x = n] && \text{(lemma 3.21)} \\
&\geq \frac{1}{d} \sum_{x \in \overline{\mathcal{X}}} \Phi(m/d + 1, \alpha) && \text{(convexity of } \Phi(\cdot, \alpha) \text{ and Jensen's ineq.)} \\
&= \Phi(m/d + 1, \alpha).
\end{aligned}$$

Since the expectation over $\boldsymbol{\sigma}$ is lower-bounded by $\Phi(m/d + 1, \alpha)$, there must exist some $\boldsymbol{\sigma} \in \{-1, +1\}^d$ for which

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m_{\boldsymbol{\sigma}}} \Big[\frac{1}{\alpha}\big[R_{\mathcal{D}_{\boldsymbol{\sigma}}}(h_S) - R_{\mathcal{D}_{\boldsymbol{\sigma}}}(h^*_{\mathcal{D}_{\boldsymbol{\sigma}}})\big]\Big] > \Phi(m/d + 1, \alpha). \tag{3.45}$$

Then, by lemma 3.22, for that $\boldsymbol{\sigma}$, for any $\gamma \in [0, 1]$,

$$\mathop{\mathbb{P}}_{S \sim \mathcal{D}^m_{\boldsymbol{\sigma}}} \Big[\frac{1}{\alpha}\big[R_{\mathcal{D}_{\boldsymbol{\sigma}}}(h_S) - R_{\mathcal{D}_{\boldsymbol{\sigma}}}(h^*_{\mathcal{D}_{\boldsymbol{\sigma}}})\big] > \gamma u\Big] > (1 - \gamma) u, \tag{3.46}$$

where $u = \Phi(m/d + 1, \alpha)$. Selecting δ and ϵ such that $\delta \leq (1 - \gamma) u$ and $\epsilon \leq \gamma \alpha u$ gives

$$\mathop{\mathbb{P}}_{S \sim \mathcal{D}^m_{\boldsymbol{\sigma}}} \big[R_{\mathcal{D}_{\boldsymbol{\sigma}}}(h_S) - R_{\mathcal{D}_{\boldsymbol{\sigma}}}(h^*_{\mathcal{D}_{\boldsymbol{\sigma}}}) > \epsilon\big] > \delta. \tag{3.47}$$

To satisfy the inequalities defining ϵ and δ, let $\gamma = 1 - 8\delta$. Then,

$$\begin{aligned}
\delta \leq (1-\gamma)u \iff & u \geq \frac{1}{8} && (3.48)\\
\iff & \frac{1}{4}\left(1 - \sqrt{1 - \exp\left(-\frac{(m/d+1)\alpha^2}{1-\alpha^2}\right)}\right) \geq \frac{1}{8} && (3.49)\\
\iff & \frac{(m/d+1)\alpha^2}{1-\alpha^2} \leq \log\frac{4}{3} && (3.50)\\
\iff & \frac{m}{d} \leq \left(\frac{1}{\alpha^2} - 1\right)\log\frac{4}{3} - 1. && (3.51)
\end{aligned}$$

Selecting $\alpha = 8\epsilon/(1-8\delta)$ gives $\epsilon = \gamma\alpha/8$ and the condition

$$\frac{m}{d} \leq \left(\frac{(1-8\delta)^2}{64\epsilon^2} - 1\right)\log\frac{4}{3} - 1. \tag{3.52}$$

Let $f(1/\epsilon^2)$ denote the right-hand side. We are seeking a sufficient condition of the form $m/d \leq \omega/\epsilon^2$. Since $\epsilon \leq 1/64$, to ensure that $\omega/\epsilon^2 \leq f(1/\epsilon^2)$, it suffices to impose $\frac{\omega}{(1/64)^2} = f\big(\frac{1}{(1/64)^2}\big)$. This condition gives

$$\omega = (7/64)^2\log(4/3) - (1/64)^2(\log(4/3)+1) \approx .003127 \geq 1/320 = .003125.$$

Thus, $\epsilon^2 \leq \frac{1}{320(m/d)}$ is sufficient to ensure the inequalities. □

The theorem shows that for any algorithm $\mathcal{A}$, in the non-realizable case, there exists a 'bad' distribution over $\mathcal{X} \times \{0,1\}$ such that the error of the hypothesis returned by $\mathcal{A}$ is a constant times $\sqrt{\frac{d}{m}}$ with some constant probability. The VC-dimension appears as a critical quantity in learning in this general setting as well. In particular, with an infinite VC-dimension, agnostic PAC-learning is not possible.

3.5 Chapter notes

The use of Rademacher complexity for deriving generalization bounds in learning was first advocated by Koltchinskii [2001], Koltchinskii and Panchenko [2000], and Bartlett, Boucheron, and Lugosi [2002a], see also [Koltchinskii and Panchenko, 2002, Bartlett and Mendelson, 2002]. Bartlett, Bousquet, and Mendelson [2002b] introduced the notion of *local Rademacher complexity*, that is the Rademacher complexity restricted to a subset of the hypothesis set limited by a bound on the variance. This can be used to derive better guarantees under some regularity assumptions about the noise.

Theorem 3.7 is due to Massart [2000]. The notion of VC-dimension was introduced by Vapnik and Chervonenkis [1971] and has been since extensively studied [Vapnik, 2006, Vapnik and Chervonenkis, 1974, Blumer et al., 1989, Assouad, 1983, Dudley, 1999]. In addition to the key role it plays in machine learning, the VC-dimension is also widely used in a variety of other areas of computer science and mathematics (e.g., see Shelah [1972], Chazelle [2000]). Theorem 3.17 is known as *Sauer's lemma* in the learning community, however the result was first given by Vapnik and Chervonenkis [1971] (in a somewhat different version) and later independently by Sauer [1972] and Shelah [1972].

In the realizable case, lower bounds for the expected error in terms of the VC-dimension were given by Vapnik and Chervonenkis [1974] and Haussler et al. [1988]. Later, a lower bound for the probability of error such as that of theorem 3.20 was given by Blumer et al. [1989]. Theorem 3.20 and its proof, which improves upon this previous result, are due to Ehrenfeucht, Haussler, Kearns, and Valiant [1988]. Devroye and Lugosi [1995] gave slightly tighter bounds for the same problem with a more complex expression. Theorem 3.23 giving a lower bound in the non-realizable case and the proof presented are due to Anthony and Bartlett [1999]. For other examples of application of the probabilistic method demonstrating its full power, consult the reference book of Alon and Spencer [1992].

There are several other measures of the complexity of a family of functions used in machine learning, including *covering numbers*, *packing numbers*, and some other complexity measures discussed in chapter 11. A covering number $\mathcal{N}_p(\mathcal{G}, \epsilon)$ is the minimal number of L_p balls of radius $\epsilon > 0$ needed to cover a family of loss functions $\mathcal{G}$. A packing number $\mathcal{M}_p(\mathcal{G}, \epsilon)$ is the maximum number of non-overlapping L_p balls of radius ϵ centered in $\mathcal{G}$. The two notions are closely related, in particular it can be shown straightforwardly that $\mathcal{M}_p(\mathcal{G}, 2\epsilon) \leq \mathcal{N}_p(\mathcal{G}, \epsilon) \leq \mathcal{M}_p(\mathcal{G}, \epsilon)$ for $\mathcal{G}$ and $\epsilon > 0$. Each complexity measure naturally induces a different reduction of infinite hypothesis sets to finite ones, thereby resulting in generalization bounds for infinite hypothesis sets. Exercise 3.31 illustrates the use of covering numbers for deriving generalization bounds using a very simple proof. There are also close relationships between these complexity measures: for example, by Dudley's theorem, the empirical Rademacher complexity can be bounded in terms of $\mathcal{N}_2(\mathcal{G}, \epsilon)$ [Dudley, 1967, 1987] and the covering and packing numbers can be bounded in terms of the VC-dimension [Haussler, 1995]. See also [Ledoux and Talagrand, 1991, Alon et al., 1997, Anthony and Bartlett, 1999, Cucker and Smale, 2001, Vidyasagar, 1997] for a number of upper bounds on the covering number in terms of other complexity measures.

3.6 Exercises

3.1 Growth function of intervals in $\mathbb{R}$. Let $\mathcal{H}$ be the set of intervals in $\mathbb{R}$. The VC-dimension of $\mathcal{H}$ is 2. Compute its shattering coefficient $\Pi_{\mathcal{H}}(m)$, $m \geq 0$. Compare your result with the general bound for growth functions.

3.2 Growth function and Rademacher complexity of thresholds in $\mathbb{R}$. Let $\mathcal{H}$ be the family of threshold functions over the real line: $\mathcal{H} = \{x \mapsto 1_{x \leq \theta} \colon \theta \in \mathbb{R}\} \cup \{x \mapsto 1_{x \geq \theta} \colon \theta \in \mathbb{R}\}$. Give an upper bound on the growth function $\Pi_m(\mathcal{H})$. Use that to derive an upper bound on $\mathfrak{R}_m(\mathcal{H})$.

3.3 Growth function of linear combinations. A *linearly separable labeling* of a set $\mathcal{X}$ of vectors in $\mathbb{R}^d$ is a classification of $\mathcal{X}$ into two sets $\mathcal{X}^+$ and $\mathcal{X}^-$ with $\mathcal{X}^+ = \{\mathbf{x} \in \mathcal{X} \colon \mathbf{w} \cdot \mathbf{x} > 0\}$ and $\mathcal{X}^- = \{\mathbf{x} \in \mathcal{X} \colon \mathbf{w} \cdot \mathbf{x} < 0\}$ for some $\mathbf{w} \in \mathbb{R}^d$.

Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be a subset of $\mathbb{R}^d$.

(a) Let $\{\mathcal{X}^+, \mathcal{X}^-\}$ be a dichotomy of $\mathcal{X}$ and let $\mathbf{x}_{m+1} \in \mathbb{R}^d$. Show that $\{\mathcal{X}^+ \cup \{\mathbf{x}_{m+1}\}, \mathcal{X}^-\}$ and $\{\mathcal{X}^+, \mathcal{X}^- \cup \{\mathbf{x}_{m+1}\}\}$ are linearly separable by a hyperplane going through the origin if and only if $\{\mathcal{X}^+, \mathcal{X}^-\}$ is linearly separable by a hyperplane going through the origin and $\mathbf{x}_{m+1}$.

(b) Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be a subset of $\mathbb{R}^d$ such that any k-element subset of $\mathcal{X}$ with $k \leq d$ is linearly independent. Then, show that the number of linearly separable labelings of $\mathcal{X}$ is $C(m, d) = 2\sum_{k=0}^{d-1} \binom{m-1}{k}$. (*Hint*: prove by induction that $C(m+1, d) = C(m, d) + C(m, d-1)$.

(c) Let $f_1, \ldots, f_p$ be p functions mapping $\mathbb{R}^d$ to $\mathbb{R}$. Define $\mathcal{F}$ as the family of classifiers based on linear combinations of these functions:

$$\mathcal{F} = \left\{ x \mapsto \operatorname{sgn}\left(\sum_{k=1}^{p} a_k f_k(x) \right) \colon a_1, \ldots, a_p \in \mathbb{R} \right\}.$$

Define Ψ by $\Psi(x) = (f_1(x), \ldots, f_p(x))$. Assume that there exists $x_1, \ldots, x_m \in \mathbb{R}^d$ such that every p-subset of $\{\Psi(x_1), \ldots, \Psi(x_m)\}$ is linearly independent. Then, show that

$$\Pi_{\mathcal{F}}(m) = 2\sum_{i=0}^{p-1} \binom{m-1}{i}.$$

3.4 Lower bound on growth function. Prove that Sauer's lemma (theorem 3.17) is tight, i.e., for any set $\mathcal{X}$ of $m > d$ elements, show that there exists a hypothesis class $\mathcal{H}$ of VC-dimension d such that $\Pi_{\mathcal{H}}(m) = \sum_{i=0}^{d} \binom{m}{i}$.

3.5 Finer Rademacher upper bound. Show that a finer upper bound on the Rademacher complexity of the family $\mathcal{G}$ can be given in terms of $\mathbb{E}_S[\Pi(\mathcal{G}, S)]$, where $\Pi(\mathcal{G}, S)$ is the number of ways to label the points in sample S.

3.6 Singleton hypothesis class. Consider the trivial hypothesis set $\mathcal{H} = \{h_0\}$.

(a) Show that $\mathfrak{R}_m(\mathcal{H}) = 0$ for any $m > 0$.

(b) Use a similar construction to show that Massart's lemma (theorem 3.7) is tight.

3.7 Two function hypothesis class. Let $\mathcal{H}$ be a hypothesis set reduced to two functions: $\mathcal{H} = \{h_{-1}, h_{+1}\}$ and let $S = (x_1, \ldots, x_m) \subseteq \mathcal{X}$ be a sample of size m.

(a) Assume that h_{-1} is the constant function taking value -1 and h_{+1} the constant function taking the value $+1$. What is the VC-dimension d of $\mathcal{H}$? Upper bound the empirical Rademacher complexity $\widehat{\mathfrak{R}}_S(\mathcal{H})$ (*Hint*: express $\widehat{\mathfrak{R}}_S(\mathcal{H})$ in terms of the absolute value of a sum of Rademacher variables and apply Jensen's inequality) and compare your bound with $\sqrt{d/m}$.

(b) Assume that h_{-1} is the constant function taking value -1 and h_{+1} the function taking value -1 everywhere except at x_1 where it takes the value $+1$. What is the VC-dimension d of $\mathcal{H}$? Compute the empirical Rademacher complexity $\widehat{\mathfrak{R}}_S(\mathcal{H})$.

3.8 Rademacher identities. Fix $m \geq 1$. Prove the following identities for any $\alpha \in \mathbb{R}$ and any two hypothesis sets $\mathcal{H}$ and $\mathcal{H}'$ of functions mapping from $\mathcal{X}$ to $\mathbb{R}$:

(a) $\mathfrak{R}_m(\alpha\mathcal{H}) = |\alpha|\mathfrak{R}_m(\mathcal{H})$.

(b) $\mathfrak{R}_m(\mathcal{H} + \mathcal{H}') = \mathfrak{R}_m(\mathcal{H}) + \mathfrak{R}_m(\mathcal{H}')$.

(c) $\mathfrak{R}_m(\{\max(h, h') \colon h \in \mathcal{H}, h' \in \mathcal{H}'\}) \leq \mathfrak{R}_m(\mathcal{H}) + \mathfrak{R}_m(\mathcal{H}')$,
where $\max(h, h')$ denotes the function $x \mapsto \max_{x \in \mathcal{X}}(h(x), h'(x))$ (*Hint*: you could use the identity $\max(a, b) = \frac{1}{2}[a + b + |a - b|]$ valid for all $a, b \in \mathbb{R}$ and Talagrand's contraction lemma (see lemma 5.7)).

3.9 Rademacher complexity of intersection of concepts. Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two families of functions mapping $\mathcal{X}$ to $\{0, 1\}$ and let $\mathcal{H} = \{h_1 h_2 \colon h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2\}$. Show that the empirical Rademacher complexity of $\mathcal{H}$ for any sample S of size m can be bounded as follows:

$$\widehat{\mathfrak{R}}_S(H) \leq \widehat{\mathfrak{R}}_S(\mathcal{H}_1) + \widehat{\mathfrak{R}}_S(\mathcal{H}_2).$$

Hint: use the Lipschitz function $x \mapsto \max(0, x-1)$ and Talagrand's contraction lemma.

Use that to bound the Rademacher complexity $\mathfrak{R}_m(\mathcal{U})$ of the family $\mathcal{U}$ of intersections of two concepts c_1 and c_2 with $c_1 \in \mathcal{C}_1$ and $c_2 \in \mathcal{C}_2$ in terms of the Rademacher complexities of $\mathcal{C}_1$ and $\mathcal{C}_2$.

3.10 Rademacher complexity of prediction vector. Let $S = (x_1, \ldots, x_m)$ be a sample of size m and fix $h\colon \mathcal{X} \to \mathbb{R}$.

(a) Denote by $\mathbf{u}$ the vector of predictions of h for S: $\mathbf{u} = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_m) \end{bmatrix}$. Give an upper bound on the empirical Rademacher complexity $\widehat{\mathfrak{R}}_S(\mathcal{H})$ of $\mathcal{H} = \{h, -h\}$ in terms of $\|\mathbf{u}\|_2$ (*Hint*: express $\widehat{\mathfrak{R}}_S(\mathcal{H})$ in terms of the expectation of an absolute value and apply Jensen's inequality). Suppose that $h(x_i) \in \{0, -1, +1\}$ for all $i \in [m]$. Express the bound on the Rademacher complexity in terms of the sparsity measure $n = |\{i \mid h(x_i) \neq 0\}|$. What is that upper bound for the extreme values of the sparsity measure?

(b) Let $\mathcal{F}$ be a family of functions mapping $\mathcal{X}$ to $\mathbb{R}$. Give an upper bound on the empirical Rademacher complexity of $\mathcal{F} + h = \{f + h\colon f \in \mathcal{F}\}$ and that of $\mathcal{F} \pm h = (\mathcal{F} + h) \cup (\mathcal{F} - h)$ in terms of $\widehat{\mathfrak{R}}_S(\mathcal{F})$ and $\|\mathbf{u}\|_2$.

3.11 Rademacher complexity of regularized neural networks. Let the input space be $\mathcal{X} = \mathbb{R}^{n_1}$. In this problem, we consider the family of regularized neural networks defined by the following set of functions mapping $\mathcal{X}$ to $\mathbb{R}$:

$$\mathcal{H} = \left\{ \mathbf{x} \mapsto \sum_{j=1}^{n_2} w_j \sigma(\mathbf{u}_j \cdot \mathbf{x})\colon \|\mathbf{w}\|_1 \leq \Lambda', \|\mathbf{u}_j\|_2 \leq \Lambda, \forall j \in [n_2] \right\},$$

where σ is an L-Lipschitz function. As an example, σ could be the sigmoid function which is 1-Lipschitz.

(a) Show that $\widehat{\mathfrak{R}}_S(\mathcal{H}) = \frac{\Lambda'}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\|\mathbf{u}\|_2 \leq \Lambda} \left| \sum_{i=1}^m \sigma_i \sigma(\mathbf{u} \cdot \mathbf{x}_i) \right| \right]$.

(b) Use the following form of Talagrand's lemma valid for all hypothesis sets $\mathcal{H}$ and L-Lipschitz function Φ:

$$\frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{h \in \mathcal{H}} \left| \sum_{i=1}^m \sigma_i (\Phi \circ h)(x_i) \right| \right] \leq \frac{L}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{h \in \mathcal{H}} \left| \sum_{i=1}^m \sigma_i h(x_i) \right| \right],$$

to upper bound $\widehat{\mathfrak{R}}_S(\mathcal{H})$ in terms of the empirical Rademacher complexity of $\mathcal{H}'$, where $\mathcal{H}'$ is defined by

$$\mathcal{H}' = \{\mathbf{x} \mapsto s(\mathbf{u} \cdot \mathbf{x})\colon \|\mathbf{u}\|_2 \leq \Lambda, s \in \{-1, +1\}\}\,.$$

(c) Use the Cauchy-Schwarz inequality to show that

$$\widehat{\mathfrak{R}}_S(\mathcal{H}') = \frac{\Lambda}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\left\| \sum_{i=1}^{m} \sigma_i \mathbf{x}_i \right\|_2 \right].$$

(d) Use the inequality $\mathbb{E}_{\mathbf{v}}[\|\mathbf{v}\|_2] \leq \sqrt{\mathbb{E}_{\mathbf{v}}[\|\mathbf{v}\|_2^2]}$, which holds by Jensen's inequality to upper bound $\widehat{\mathfrak{R}}_S(\mathcal{H}')$.

(e) Assume that for all $\mathbf{x} \in S$, $\|\mathbf{x}\|_2 \leq r$ for some $r > 0$. Use the previous questions to derive an upper bound on the Rademacher complexity of $\mathcal{H}$ in terms of r.

3.12 Rademacher complexity. Professor Jesetoo claims to have found a better bound on the Rademacher complexity of any hypothesis set $\mathcal{H}$ of functions taking values in $\{-1, +1\}$, in terms of its VC-dimension VCdim($\mathcal{H}$). His bound is of the form $\mathfrak{R}_m(\mathcal{H}) \leq O\big(\frac{\text{VCdim}(\mathcal{H})}{m}\big)$. Can you show that Professor Jesetoo's claim cannot be correct? (*Hint*: consider a hypothesis set $\mathcal{H}$ reduced to just two simple functions.)

3.13 VC-dimension of union of k intervals. What is the VC-dimension of subsets of the real line formed by the union of k intervals?

3.14 VC-dimension of finite hypothesis sets. Show that the VC-dimension of a finite hypothesis set $\mathcal{H}$ is at most $\log_2 |\mathcal{H}|$.

3.15 VC-dimension of subsets. What is the VC-dimension of the set of subsets I_α of the real line parameterized by a single parameter α: $I_\alpha = [\alpha, \alpha+1] \cup [\alpha+2, +\infty)$?

3.16 VC-dimension of axis-aligned squares and triangles.

(a) What is the VC-dimension of axis-aligned squares in the plane?

(b) Consider right triangles in the plane with the sides adjacent to the right angle both parallel to the axes and with the right angle in the lower left corner. What is the VC-dimension of this family?

3.17 VC-dimension of closed balls in $\mathbb{R}^n$. Show that the VC-dimension of the set of all closed balls in $\mathbb{R}^n$, i.e., sets of the form $\{x \in \mathbb{R}^n : \|x - x_0\|^2 \leq r\}$ for some $x_0 \in \mathbb{R}^n$ and $r \geq 0$, is less than or equal to $n + 2$.

3.18 VC-dimension of ellipsoids. What is the VC-dimension of the set of all ellipsoids in $\mathbb{R}^n$?

3.19 VC-dimension of a vector space of real functions. Let F be a finite-dimensional vector space of real functions on $\mathbb{R}^n$, $\dim(F) = r < \infty$. Let $\mathcal{H}$ be the set of hypotheses:

$$\mathcal{H} = \{\{x : f(x) \geq 0\} : f \in F\}.$$

Show that d, the VC-dimension of $\mathcal{H}$, is finite and that $d \leq r$. (*Hint*: select an arbitrary set of $m = r + 1$ points and consider linear mapping $u : F \to \mathbb{R}^m$ defined by: $u(f) = (f(x_1), \ldots, f(x_m))$.)

3.20 VC-dimension of sine functions. Consider the hypothesis family of sine functions (Example 3.16): $\{x \to \sin(\omega x) : \omega \in \mathbb{R}\}$.

(a) Show that for any $x \in \mathbb{R}$ the points $x, 2x, 3x$ and $4x$ cannot be shattered by this family of sine functions.

(b) Show that the VC-dimension of the family of sine functions is infinite. (*Hint*: show that $\{2^{-i} : i \leq m\}$ can be shattered for any $m > 0$.)

3.21 VC-dimension of union of halfspaces. Provide an upper bound on the VC-dimension of the class of hypotheses described by the unions of k halfspaces.

3.22 VC-dimension of intersection of halfspaces. Consider the class $\mathcal{C}_k$ of convex intersections of k halfspaces. Give lower and upper bound estimates for $\text{VCdim}(\mathcal{C}_k)$.

3.23 VC-dimension of intersection concepts.

(a) Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two concept classes. Show that for any concept class $\mathcal{C} = \{c_1 \cap c_2 : c_1 \in \mathcal{C}_1, c_2 \in \mathcal{C}_2\}$,

$$\Pi_{\mathcal{C}}(m) \leq \Pi_{\mathcal{C}_1}(m)\, \Pi_{\mathcal{C}_2}(m). \tag{3.53}$$

(b) Let $\mathcal{C}$ be a concept class with VC-dimension d and let $\mathcal{C}_s$ be the concept class formed by all intersections of s concepts from $\mathcal{C}$, $s \geq 1$. Show that the VC-dimension of $\mathcal{C}_s$ is bounded by $2ds \log_2(3s)$. (*Hint*: show that $\log_2(3x) < 9x/(2e)$ for any $x \geq 2$.)

3.24 VC-dimension of union of concepts. Let $\mathcal{A}$ and $\mathcal{B}$ be two sets of functions mapping from $\mathcal{X}$ into $\{0,1\}$, and assume that both $\mathcal{A}$ and $\mathcal{B}$ have finite VC-dimension, with $\text{VCdim}(\mathcal{A}) = d_{\mathcal{A}}$ and $\text{VCdim}(\mathcal{B}) = d_{\mathcal{B}}$. Let $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$ be the union of $\mathcal{A}$ and $\mathcal{B}$.

(a) Prove that for all m, $\Pi_{\mathcal{C}}(m) \leq \Pi_{\mathcal{A}}(m) + \Pi_{\mathcal{B}}(m)$.

(b) Use Sauer's lemma to show that for $m \geq d_{\mathcal{A}} + d_{\mathcal{B}} + 2$, $\Pi_{\mathcal{C}}(m) < 2^m$, and give a bound on the VC-dimension of $\mathcal{C}$.

3.25 VC-dimension of symmetric difference of concepts. For two sets $\mathcal{A}$ and $\mathcal{B}$, let $\mathcal{A}\Delta\mathcal{B}$ denote the symmetric difference of $\mathcal{A}$ and $\mathcal{B}$, i.e., $\mathcal{A}\Delta\mathcal{B} = (\mathcal{A}\cup\mathcal{B}) - (\mathcal{A}\cap\mathcal{B})$. Let $\mathcal{H}$ be a non-empty family of subsets of $\mathcal{X}$ with finite VC-dimension. Let $\mathcal{A}$ be an element of $\mathcal{H}$ and define $\mathcal{H}\Delta\mathcal{A} = \{X\Delta\mathcal{A} \colon X \in \mathcal{H}\}$. Show that

$$\text{VCdim}(\mathcal{H}\Delta\mathcal{A}) = \text{VCdim}(\mathcal{H}).$$

3.26 Symmetric functions. A function $h \colon \{0,1\}^n \to \{0,1\}$ is *symmetric* if its value is uniquely determined by the number of 1's in the input. Let $\mathcal{C}$ denote the set of all symmetric functions.

(a) Determine the VC-dimension of $\mathcal{C}$.

(b) Give lower and upper bounds on the sample complexity of any consistent PAC learning algorithm for $\mathcal{C}$.

(c) Note that any hypothesis $h \in \mathcal{C}$ can be represented by a vector $(y_0, y_1, \ldots, y_n) \in \{0,1\}^{n+1}$, where y_i is the value of h on examples having precisely i 1's. Devise a consistent learning algorithm for $\mathcal{C}$ based on this representation.

3.27 VC-dimension of neural networks.

Let $\mathcal{C}$ be a concept class over $\mathbb{R}^r$ with VC-dimension d. A $\mathcal{C}$-neural network with one intermediate layer is a concept defined over $\mathbb{R}^n$ that can be represented by a directed acyclic graph such as that of Figure 3.7, in which the input nodes are those at the bottom and in which each other node is labeled with a concept $c \in \mathcal{C}$.

The output of the neural network for a given input vector $(x_1, \ldots, x_n)$ is obtained as follows. First, each of the n input nodes is labeled with the corresponding value $x_i \in \mathbb{R}$. Next, the value at a node u in the higher layer and labeled with c is obtained by applying c to the values of the input nodes admitting an

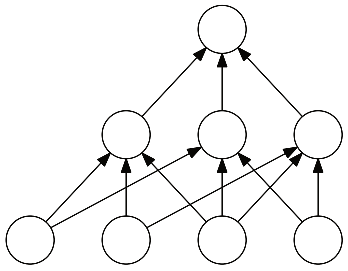


Figure 3.7
A neural network with one intermediate layer.

edge ending in u. Note that since c takes values in $\{0, 1\}$, the value at u is in $\{0, 1\}$. The value at the top or output node is obtained similarly by applying the corresponding concept to the values of the nodes admitting an edge to the output node.

(a) Let $\mathcal{H}$ denote the set of all neural networks defined as above with $k \geq 2$ internal nodes. Show that the growth function $\Pi_{\mathcal{H}}(m)$ can be upper bounded in terms of the product of the growth functions of the hypothesis sets defined at each intermediate layer.

(b) Use that to upper bound the VC-dimension of the $\mathcal{C}$-neural networks (*Hint*: you can use the implication $m = 2x \log_2(xy) \Rightarrow m > x \log_2(ym)$ valid for $m \geq 1$, and $x, y > 0$ with $xy > 4$).

(c) Let $\mathcal{C}$ be the family of concept classes defined by threshold functions $\mathcal{C} = \{\text{sgn}(\sum_{j=1}^{r} w_j x_j) \colon \mathbf{w} \in \mathbb{R}^r\}$. Give an upper bound on the VC-dimension of $\mathcal{H}$ in terms of k and r.

3.28 VC-dimension of convex combinations. Let $\mathcal{H}$ be a family of functions mapping from an input space $\mathcal{X}$ to $\{-1, +1\}$ and let T be a positive integer. Give an upper bound on the VC-dimension of the family of functions $\mathcal{F}_T$ defined by

$$\mathcal{F} = \left\{ \text{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t \right) : h_t \in \mathcal{H}, \alpha_t \geq 0, \sum_{t=1}^{T} \alpha_t \leq 1 \right\}.$$

(*Hint*: you can use exercise 3.27 and its solution).

3.29 Infinite VC-dimension.

(a) Show that if a concept class $\mathcal{C}$ has infinite VC-dimension, then it is not PAC-learnable.

(b) In the standard PAC-learning scenario, the learning algorithm receives all examples first and then computes its hypothesis. Within that setting, PAC-learning of concept classes with infinite VC-dimension is not possible as seen in the previous question.

Imagine now a different scenario where the learning algorithm can alternate between drawing more examples and computation. The objective of this problem is to prove that PAC-learning can then be possible for some concept classes with infinite VC-dimension.

Consider for example the special case of the concept class $\mathcal{C}$ of all subsets of natural numbers. Professor Vitres has an idea for the first stage of a learning algorithm L PAC-learning $\mathcal{C}$. In the first stage, L draws a sufficient number of points m such that the probability of drawing a point beyond the maximum value M observed be small with high confidence. Can you complete Professor Vitres' idea by describing the second stage of the algorithm so that it PAC-learns $\mathcal{C}$? The description should be augmented with the proof that L can PAC-learn $\mathcal{C}$.

3.30 VC-dimension generalization bound – realizable case. In this exercise we show that the bound given in corollary 3.19 can be improved to $O(\frac{d\log(m/d)}{m})$ in the realizable setting. Assume we are in the realizable scenario, i.e. the target concept is included in our hypothesis class $\mathcal{H}$. We will show that if a hypothesis h is consistent with a sample $S \sim \mathcal{D}^m$ then for any $\epsilon > 0$ such that $m\epsilon \geq 8$

$$\mathbb{P}[R(h) > \epsilon] \leq 2\Big[\frac{2em}{d}\Big]^d 2^{-m\epsilon/2}\,. \tag{3.54}$$

(a) Let $\mathcal{H}_S \subseteq \mathcal{H}$ be the subset of hypotheses consistent with the sample S, let $\widehat{R}_S(h)$ denote the empirical error with respect to the sample S and define S' as another independent sample drawn from $\mathcal{D}^m$. Show that the following inequality holds for any $h_0 \in \mathcal{H}_S$:

$$\mathbb{P}\Big[\sup_{h\in\mathcal{H}_S} |\widehat{R}_S(h) - \widehat{R}_{S'}(h)| > \frac{\epsilon}{2}\Big] \geq \mathbb{P}\Big[B(m,\epsilon) > \frac{m\epsilon}{2}\Big]\,\mathbb{P}[R(h_0) > \epsilon]\,,$$

where $B(m,\epsilon)$ is a binomial random variable with parameters (m,ϵ). (*Hint*: prove and use the fact that $\mathbb{P}[\widehat{R}_S(h) \geq \frac{\epsilon}{2}] \geq \mathbb{P}[\widehat{R}_S(h) > \frac{\epsilon}{2} \wedge R(h) > \epsilon]$.)

(b) Prove that $\mathbb{P}\Big[B(m,\epsilon) > \frac{m\epsilon}{2}\Big] \geq \frac{1}{2}$. Use this inequality along with the result from (a) to show that for any $h_0 \in \mathcal{H}_S$

$$\mathbb{P}\Big[R(h_0) > \epsilon\Big] \leq 2\,\mathbb{P}\Big[\sup_{h\in\mathcal{H}_S} |\widehat{R}_S(h) - \widehat{R}_{S'}(h)| > \frac{\epsilon}{2}\Big].$$

(c) Instead of drawing two samples, we can draw one sample T of size $2m$ then uniformly at random split it into S and S'. The right hand side of part (b) can then be rewritten as:

$$\mathbb{P}\Big[\sup_{h\in\mathcal{H}_S} |\widehat{R}_S(h) - \widehat{R}_{S'}(h)| > \frac{\epsilon}{2}\Big] = \mathop{\mathbb{P}}_{\substack{T\sim\mathcal{D}^{2m}:\\ T\to[S,S']}}\Big[\exists h\in\mathcal{H}\colon \widehat{R}_S(h) = 0 \wedge \widehat{R}_{S'}(h) > \frac{\epsilon}{2}\Big].$$

Let h_0 be a hypothesis such that $\widehat{R}_T(h_0) > \frac{\epsilon}{2}$ and let $l > \frac{m\epsilon}{2}$ be the total number of errors h_0 makes on T. Show that the probability of all l errors falling into S' is upper bounded by 2^{-l}.

(d) Part (b) implies that for any $h \in \mathcal{H}$

$$\mathop{\mathbb{P}}_{\substack{T\sim\mathcal{D}^{2m}:\\ T\to(S,S')}}\Big[\widehat{R}_S(h) = 0 \;\wedge\; \widehat{R}_{S'}(h) > \frac{\epsilon}{2} \;\Big|\; \widehat{R}_T(h_0) > \frac{\epsilon}{2}\Big] \leq 2^{-l}.$$

Use this bound to show that for any $h \in \mathcal{H}$

$$\mathop{\mathbb{P}}_{\substack{T\sim\mathcal{D}^{2m}:\\ T\to(S,S')}}\Big[\widehat{R}_S(h) = 0 \;\wedge\; \widehat{R}_{S'}(h) > \frac{\epsilon}{2}\Big] \leq 2^{-\frac{\epsilon m}{2}}.$$

(e) Complete the proof of inequality (3.54) by using the union bound to upper bound $\mathbb{P}_{\substack{T\sim\mathcal{D}^{2m}:\\ T\to(S,S')}}\Big[\exists h \in \mathcal{H}\colon \widehat{R}_S(h) = 0 \;\wedge\; \widehat{R}_{S'}(h) > \frac{\epsilon}{2}\Big]$. Show that we can achieve a high probability generalization bound that is of the order $O(\frac{d\log(m/d)}{m})$.

3.31 Generalization bound based on covering numbers. Let $\mathcal{H}$ be a family of functions mapping $\mathcal{X}$ to a subset of real numbers $\mathcal{Y} \subseteq \mathbb{R}$. For any $\epsilon > 0$, the *covering number* $\mathcal{N}(\mathcal{H}, \epsilon)$ of $\mathcal{H}$ for the L_∞ norm is the minimal $k \in \mathbb{N}$ such that $\mathcal{H}$ can be covered with k balls of radius ϵ, that is, there exists $\{h_1, \ldots, h_k\} \subseteq \mathcal{H}$ such that, for all $h \in \mathcal{H}$, there exists $i \leq k$ with $\|h - h_i\|_\infty = \max_{x\in\mathcal{X}} |h(x) - h_i(x)| \leq \epsilon$. In particular, when $\mathcal{H}$ is a compact set, a finite covering can be extracted from a covering of $\mathcal{H}$ with balls of radius ϵ and thus $\mathcal{N}(\mathcal{H}, \epsilon)$ is finite.

Covering numbers provide a measure of the complexity of a class of functions: the larger the covering number, the richer is the family of functions. The objective of this problem is to illustrate this by proving a learning bound in the case of the squared loss. Let $\mathcal{D}$ denote a distribution over $\mathcal{X} \times \mathcal{Y}$ according to which labeled examples are drawn. Then, the generalization error of $h \in \mathcal{H}$ for the squared loss is defined by $R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[(h(x)-y)^2]$ and its empirical error for a labeled sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$ by $\widehat{R}_S(h) = \frac{1}{m}\sum_{i=1}^m (h(x_i) - y_i)^2$. We will assume that $\mathcal{H}$ is bounded, that is there exists $M > 0$ such that $|h(x) - y| \leq M$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$. The following is the generalization bound proven in this problem:

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left[\sup_{h\in\mathcal{H}} |R(h) - \widehat{R}_S(h)| \geq \epsilon\right] \leq \mathcal{N}\Big(\mathcal{H}, \frac{\epsilon}{8M}\Big) 2\exp\Big(\frac{-m\epsilon^2}{2M^4}\Big). \tag{3.55}$$

The proof is based on the following steps.

(a) Let $L_S = R(h) - \widehat{R}_S(h)$, then show that for all $h_1, h_2 \in \mathcal{H}$ and any labeled sample S, the following inequality holds:

$$|L_S(h_1) - L_S(h_2)| \leq 4M\|h_1 - h_2\|_\infty .$$

(b) Assume that $\mathcal{H}$ can be covered by k subsets $\mathcal{B}_1, \ldots, \mathcal{B}_k$, that is $\mathcal{H} = \mathcal{B}_1 \cup \ldots \cup \mathcal{B}_k$. Then, show that, for any $\epsilon > 0$, the following upper bound holds:

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left[\sup_{h\in\mathcal{H}} |L_S(h)| \geq \epsilon\right] \leq \sum_{i=1}^k \mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left[\sup_{h\in\mathcal{B}_i} |L_S(h)| \geq \epsilon\right].$$

(c) Finally, let $k = \mathcal{N}(\mathcal{H}, \frac{\epsilon}{8M})$ and let $\mathcal{B}_1, \ldots, \mathcal{B}_k$ be balls of radius $\epsilon/(8M)$ centered at $h_1, \ldots, h_k$ covering $\mathcal{H}$. Use part (a) to show that for all $i \in [k]$,

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left[\sup_{h\in\mathcal{B}_i} |L_S(h)| \geq \epsilon\right] \leq \mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left[|L_S(h_i)| \geq \frac{\epsilon}{2}\right],$$

and apply Hoeffding's inequality (theorem D.2) to prove (3.55).

4 Model Selection

A key problem in the design of learning algorithms is the choice of the hypothesis set $\mathcal{H}$. This is known as the *model selection* problem. How should the hypothesis set $\mathcal{H}$ be chosen? A rich or complex enough hypothesis set could contain the ideal Bayes classifier. On the other hand, learning with such a complex family becomes a very difficult task. More generally, the choice of $\mathcal{H}$ is subject to a trade-off that can be analyzed in terms of the *estimation* and *approximation errors*.

Our discussion will focus on the particular case of binary classification but much of what is discussed can be straightforwardly extended to different tasks and loss functions.

4.1 Estimation and approximation errors

Let $\mathcal{H}$ be a family of functions mapping $\mathcal{X}$ to $\{-1, +1\}$. The *excess error* of a hypothesis h chosen from $\mathcal{H}$, that is the difference between its error $R(h)$ and the Bayes error R^*, can be decomposed as follows:

$$R(h) - R^* = \underbrace{\Big(R(h) - \inf_{h \in \mathcal{H}} R(h)\Big)}_{\text{estimation}} + \underbrace{\Big(\inf_{h \in \mathcal{H}} R(h) - R^*\Big)}_{\text{approximation}}. \tag{4.1}$$

The first term is called the *estimation error*, the second term the *approximation error*. The estimation error depends on the hypothesis h selected. It measures the error of h with respect to the infimum of the errors achieved by hypotheses in $\mathcal{H}$, or that of the best-in-class hypothesis h^* when that infimum is reached. Note that the definition of agnostic PAC-learning is precisely based on the estimation error.

The approximation error measures how well the Bayes error can be approximated using $\mathcal{H}$. It is a property of the hypothesis set $\mathcal{H}$, a measure of its richness. For a more complex or richer hypothesis $\mathcal{H}$, the approximation error tends to be smaller at the price of a larger estimation error. This is illustrated by Figure 4.1.

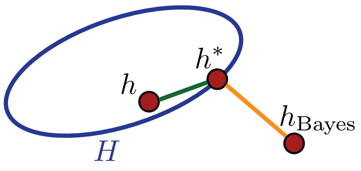


Figure 4.1
Illustration of the estimation error (in green) and approximation error (in orange). Here, it is assumed that there exists a best-in-class hypothesis, that is h^* such that $R(h^*) = \inf_{h \in \mathcal{H}} R(h)$.

Model selection consists of choosing $\mathcal{H}$ with a favorable trade-off between the approximation and estimation errors. Note, however, that the approximation error is not accessible, since in general the underlying distribution $\mathcal{D}$ needed to determine R^* is not known. Even with various noise assumptions, estimating the approximation error is difficult. In contrast, the *estimation error of an algorithm* $\mathcal{A}$, that is, the estimation error of the hypothesis h_S returned after training on a sample S, can sometimes be bounded using generalization bounds as shown in the next section.

4.2 Empirical risk minimization (ERM)

A standard algorithm for which the estimation error can be bounded is *Empirical Risk Minimization* (ERM). ERM seeks to minimize the error on the training sample:[4]

$$h_S^{\text{ERM}} = \operatorname*{argmin}_{h \in \mathcal{H}} \widehat{R}_S(h). \tag{4.2}$$

Proposition 4.1 *For any sample S, the following inequality holds for the hypothesis returned by ERM:*

$$\mathbb{P}\left[R(h_S^{\text{ERM}}) - \inf_{h \in \mathcal{H}} R(h) > \epsilon\right] \leq \mathbb{P}\left[\sup_{h \in \mathcal{H}} |R(h) - \widehat{R}_S(h)| > \frac{\epsilon}{2}\right]. \tag{4.3}$$

Proof: By definition of $\inf_{h \in \mathcal{H}} R(h)$, for any $\epsilon > 0$, there exists h_ϵ such that $R(h_\epsilon) \leq \inf_{h \in \mathcal{H}} R(h) + \epsilon$. Thus, using $\widehat{R}_S(h_S^{\text{ERM}}) \leq \widehat{R}_S(h_\epsilon)$, which holds by the

[4] Note that, if there exists multiple hypotheses with minimal error on the training sample, then ERM returns an arbitrary one.

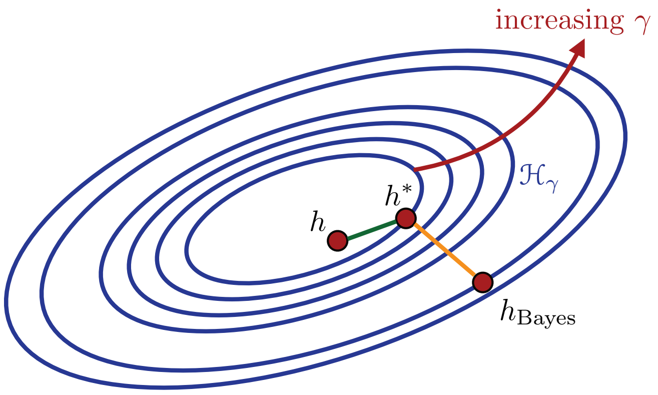


Figure 4.2
Illustration of the decomposition of a rich family $\mathcal{H} = \bigcup_{\gamma \in \Gamma} \mathcal{H}_\gamma$.

definition of the algorithm, we can write

$$\begin{aligned}
R(h_S^{\text{ERM}}) - \inf_{h \in \mathcal{H}} R(h) &= R(h_S^{\text{ERM}}) - R(h_\epsilon) + R(h_\epsilon) - \inf_{h \in \mathcal{H}} R(h)\\
&\leq R(h_S^{\text{ERM}}) - R(h_\epsilon) + \epsilon\\
&= R(h_S^{\text{ERM}}) - \widehat{R}_S(h_S^{\text{ERM}}) + \widehat{R}_S(h_S^{\text{ERM}}) - R(h_\epsilon) + \epsilon\\
&\leq R(h_S^{\text{ERM}}) - \widehat{R}_S(h_S^{\text{ERM}}) + \widehat{R}_S(h_\epsilon) - R(h_\epsilon) + \epsilon\\
&\leq 2 \sup_{h \in \mathcal{H}} |R(h) - \widehat{R}_S(h)| + \epsilon.
\end{aligned}$$

Since the inequality holds for all $\epsilon > 0$, it implies the following:

$$R(h_S^{\text{ERM}}) - \inf_{h \in \mathcal{H}} R(h) \leq 2 \sup_{h \in \mathcal{H}} |R(h) - \widehat{R}_S(h)|,$$

which concludes the proof. □

The right-hand side of (4.3) can be upper-bounded using the generalization bounds presented in the previous chapter in terms of the Rademacher complexity, the growth function, or the VC-dimension of $\mathcal{H}$. In particular, it can be bounded by $2e^{-2m[\epsilon - \mathfrak{R}_m(\mathcal{H})]^2}$. Thus, when $\mathcal{H}$ admits a favorable Rademacher complexity, for example a finite VC-dimension, for a sufficiently large sample, with high probability, the estimation error is guaranteed to be small. Nevertheless, the performance of ERM is typically very poor. This is because the algorithm disregards the complexity of the hypothesis set $\mathcal{H}$: in practice, either $\mathcal{H}$ is not complex enough, in which case the approximation error can be very large, or $\mathcal{H}$ is very rich, in which case the bound on the estimation error becomes very loose. Additionally, in many cases, determining the ERM solution is computationally intractable. For example, finding

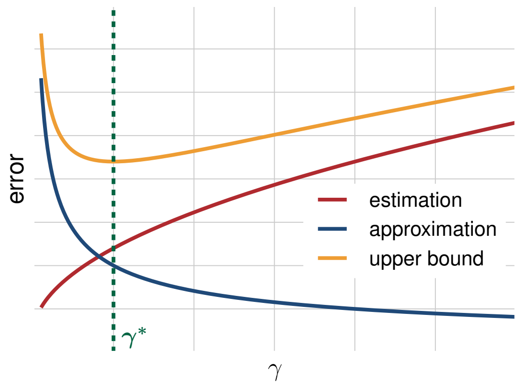


Figure 4.3
Choice of γ^* with the most favorable trade-off between estimation and approximation errors.

a linear hypothesis with the smallest error on the training sample is NP-hard, as a function of the dimension of the space.

4.3 Structural risk minimization (SRM)

In the previous section, we showed that the estimation error can be sometimes bounded or estimated. But, since the approximation error cannot be estimated, how should we choose $\mathcal{H}$? One way to proceed is to choose a very complex family $\mathcal{H}$ with no approximation error or a very small one. $\mathcal{H}$ may be too rich for generalization bounds to hold for $\mathcal{H}$, but suppose we can decompose $\mathcal{H}$ as a union of increasingly complex hypothesis sets $\mathcal{H}_\gamma$, that is $\mathcal{H} = \bigcup_{\gamma \in \Gamma} \mathcal{H}_\gamma$, with the complexity of $\mathcal{H}_\gamma$ increasing with γ, for some set Γ. Figure 4.2 illustrates this decomposition. The problem then consists of selecting the parameter $\gamma^* \in \Gamma$ and thus the hypothesis set $\mathcal{H}_{\gamma^*}$ with the most favorable trade-off between estimation and approximation errors. Since these quantities are not known, instead, as illustrated by Figure 4.3, a uniform upper bound on their sum, the excess error (also called excess risk), can be used.

This is precisely the idea behind the *Structural Risk Minimization* (SRM) method. For SRM, $\mathcal{H}$ is assumed to be decomposable into a countable set, thus, we will write its decomposition as $\mathcal{H} = \bigcup_{k \geq 1} \mathcal{H}_k$. Additionally, the hypothesis sets $\mathcal{H}_k$ are assumed to be nested: $\mathcal{H}_k \subset \mathcal{H}_{k+1}$ for all $k \geq 1$. However, many of the results presented in this section also hold for non-nested hypothesis sets. Thus, we will not make use of that assumption, unless explicitly specified. SRM consists of choosing the index $k^* \geq 1$ and the ERM hypothesis h in $\mathcal{H}_{k^*}$ that minimize an upper bound on the excess error.

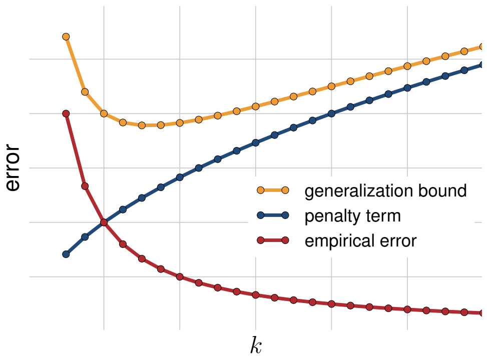


Figure 4.4
Illustration of structural risk minimization. The plots of three errors are shown as a function of the index k. Clearly, as k, or equivalently the complexity the hypothesis set $\mathcal{H}_k$, increases, the training error decreases, while the penalty term increases. SRM selects the hypothesis minimizing a bound on the generalization error, which is a sum of the empirical error and the penalty term.

As we shall see, the following learning bound holds for all $h \in \mathcal{H}$: for any $\delta > 0$, with probability at least $1 - \delta$ over the draw of a sample S of size m from $\mathcal{D}^m$, for all $h \in \mathcal{H}_k$ and $k \geq 1$,

$$R(h) \leq \widehat{R}_S(h) + \mathfrak{R}_m(\mathcal{H}_{k(h)}) + \sqrt{\frac{\log k}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}.$$

Thus, to minimize the resulting bound on the excess error $(R(h) - R^*)$, the index k and the hypothesis $h \in \mathcal{H}_k$ should be chosen to minimize the following objective function:

$$F_k(h) = \widehat{R}_S(h) + \mathfrak{R}_m(\mathcal{H}_k) + \sqrt{\frac{\log k}{m}}.$$

This is precisely the definition of the SRM solution h_S^{SRM}:

$$h_S^{\mathrm{SRM}} = \operatorname*{argmin}_{k \geq 1, h \in \mathcal{H}_k} F_k(h) = \operatorname*{argmin}_{k \geq 1, h \in \mathcal{H}_k} \widehat{R}_S(h) + R_m(\mathcal{H}_k) + \sqrt{\frac{\log k}{m}}. \tag{4.4}$$

Thus, SRM identifies an optimal index k^* and therefore hypothesis set $\mathcal{H}_{k^*}$, and returns the ERM solution based on that hypothesis set. Figure 4.4 further illustrates the selection of the index k^* and hypothesis set $\mathcal{H}_{k^*}$ by SRM by minimizing an upper bound on the sum of the training error and the penalty term $R_m(\mathcal{H}_k) + \sqrt{\log k/m}$. The following theorem shows that the SRM solution benefits from a strong learning guarantee. For any $h \in \mathcal{H}$, we will denote by $\mathcal{H}_{k(h)}$ the least complex hypothesis set among the $\mathcal{H}_k$s that contain h.

Theorem 4.2 (SRM Learning guarantee) *For any $\delta > 0$, with probability at least $1-\delta$ over the draw of an i.i.d. sample S of size m from $\mathcal{D}^m$, the generalization error of the hypothesis h_S^{SRM} returned by the SRM method is bounded as follows:*

$$R(h_S^{\mathrm{SRM}}) \leq \inf_{h\in\mathcal{H}} \left(R(h) + 2\mathfrak{R}_m(\mathcal{H}_{k(h)}) + \sqrt{\frac{\log k(h)}{m}} \right) + \sqrt{\frac{2\log\frac{3}{\delta}}{m}}.$$

Proof: Observe first that, by the union bound, the following general inequality holds:

$$\begin{aligned}
&\mathbb{P}\left[\sup_{h\in\mathcal{H}} R(h) - F_{k(h)}(h) > \epsilon\right] \\
&\qquad = \mathbb{P}\left[\sup_{k\geq 1}\sup_{h\in\mathcal{H}_k} R(h) - F_k(h) > \epsilon\right] \\
&\qquad \leq \sum_{k=1}^{\infty} \mathbb{P}\left[\sup_{h\in\mathcal{H}_k} R(h) - F_k(h) > \epsilon\right] \\
&\qquad = \sum_{k=1}^{\infty} \mathbb{P}\left[\sup_{h\in\mathcal{H}_k} R(h) - \widehat{R}_S(h) - \mathfrak{R}_m(\mathcal{H}_k) > \epsilon + \sqrt{\frac{\log k}{m}}\right] \\
&\qquad \leq \sum_{k=1}^{\infty} \exp\Big(-2m\Big[\epsilon + \sqrt{\tfrac{\log k}{m}}\Big]^2\Big) \\
&\qquad \leq \sum_{k=1}^{\infty} e^{-2m\epsilon^2} e^{-2\log k} \\
&\qquad = e^{-2m\epsilon^2} \sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6} e^{-2m\epsilon^2} \leq 2e^{-2m\epsilon^2}.
\end{aligned} \tag{4.5}$$

Next, for any two random variables X_1 and X_2, if $X_1 + X_2 > \epsilon$, then either X_1 or X_2 must be larger than $\epsilon/2$. In view of that, by the union bound, $\mathbb{P}[X_1 + X_2 > \epsilon] \leq \mathbb{P}[X_1 > \frac{\epsilon}{2}] + \mathbb{P}[X_2 > \frac{\epsilon}{2}]$. Using this inequality, inequality (4.5), and the inequality $F_{k(h_S^{\mathrm{SRM}})}(h_S^{\mathrm{SRM}}) \leq F_{k(h)}(h)$, which holds for all $h \in \mathcal{H}$, by definition of h_S^{SRM}, we can write, for any $h \in \mathcal{H}$,

$$\begin{aligned}
&\mathbb{P}\left[R(h_S^{\text{SRM}}) - R(h) - 2\mathfrak{R}_m(\mathcal{H}_{k(h)}) - \sqrt{\tfrac{\log k(h)}{m}} > \epsilon\right] \\
&\quad \leq \mathbb{P}\left[R(h_S^{\text{SRM}}) - F_{k(h_S^{\text{SRM}})}(h_S^{\text{SRM}}) > \frac{\epsilon}{2}\right] \\
&\qquad + \mathbb{P}\left[F_{k(h_S^{\text{SRM}})}(h_S^{\text{SRM}}) - R(h) - 2\mathfrak{R}_m(\mathcal{H}_{k(h)}) - \sqrt{\tfrac{\log k(h)}{m}} > \frac{\epsilon}{2}\right] \\
&\quad \leq 2e^{-\frac{m\epsilon^2}{2}} + \mathbb{P}\left[F_{k(h)}(h) - R(h) - 2\mathfrak{R}_m(\mathcal{H}_{k(h)}) - \sqrt{\tfrac{\log k(h)}{m}} > \frac{\epsilon}{2}\right] \\
&\quad = 2e^{-\frac{m\epsilon^2}{2}} + \mathbb{P}\left[\widehat{R}_S(h) - R(h) - \mathfrak{R}_m(\mathcal{H}_{k(h)}) > \frac{\epsilon}{2}\right] \\
&\quad = 2e^{-\frac{m\epsilon^2}{2}} + e^{-\frac{m\epsilon^2}{2}} = 3e^{-\frac{m\epsilon^2}{2}}.
\end{aligned}$$

Setting the right-hand side to be equal to δ completes the proof. □

The learning guarantee just proven for SRM is remarkable. To simplify its discussion, let us assume that there exists h^* such that $R(h^*) = \inf_{h \in \mathcal{H}} R(h)$, that is, that there exists a best-in-class classifier $h^* \in \mathcal{H}$. Then, the theorem implies in particular that, with probability at least $1-\delta$, the following inequality holds for all $h \in \mathcal{H}$:

$$R(h_S^{\text{SRM}}) \leq R(h^*) + 2\mathfrak{R}_m(\mathcal{H}_{k(h^*)}) + \sqrt{\frac{\log k(h^*)}{m}} + \sqrt{\frac{2\log\frac{3}{\delta}}{m}}. \tag{4.6}$$

Observe that, remarkably, this bound is similar to the estimation error bound for $\mathcal{H}_{k(h^*)}$: it differs from it only by the term $\sqrt{\log k(h^*)/m}$. Thus, modulo that term, the guarantee for SRM is as favorable as the one we would have obtained, had an oracle informed us of the index $k(h^*)$ of the best-in-class classifier's hypothesis set.

Furthermore, observe that when $\mathcal{H}$ is rich enough that $R(h^*)$ is close to the Bayes error, the learning bound (4.6) is approximately a bound on the excess error of the SRM solution. Note that, if for some k_0, the empirical error of the ERM solution for $\mathcal{H}_{k_0}$ is zero, which holds in particular if $\mathcal{H}_{k_0}$ contains the Bayes error, then, we have $\min_{h \in \mathcal{H}_k} F_{k_0}(h) \leq \min_{h \in \mathcal{H}_k} F_k(h)$ for all $k > k_0$ and only finitely many indices need to be considered in SRM.

Assume more generally that if $\min_{h \in \mathcal{H}_k} F_k(h) \leq \min_{h \in \mathcal{H}_{k+1}} F_k(h)$ for some k, then indices beyond $k+1$ need not be inspected. This property may hold for example if the empirical error cannot be further improved after some index k. In that case, the minimizing index k^* can be determined via a binary search in the interval $[1, k_{\max}]$, given some maximum value $k_{\max}$. $k_{\max}$ itself can be found by inspecting $\min_{h \in \mathcal{H}_{2^n}} F_k(h)$ for exponentially growing indices 2^n, $n \geq 1$, and setting $k_{\max} = 2^n$ for n such that $\min_{h \in \mathcal{H}_{2^n}} F_k(h) \leq \min_{h \in \mathcal{H}_{2^{n+1}}} F_k(h)$. The number of ERM computations needed to find $k_{\max}$ is in $O(n) = O(\log k_{\max})$ and similarly the number of ERM computations due to the binary search is in $O(\log k_{\max})$. Thus, if n is the smallest integer such that $k^* < 2^n$, the overall number of ERM computations is in $O(\log k^*)$.

While it benefits from a very favorable guarantee, SRM admits several drawbacks. First, the decomposability of $\mathcal{H}$ into countably many hypothesis sets, each with a converging Rademacher complexity, remains a strong assumption. As an example, the family of all measurable functions cannot be written as a union of countably many hypothesis sets with finite VC-dimension. Thus, the choice of $\mathcal{H}$ or that of the hypothesis sets $\mathcal{H}_k$ is a key component of SRM. Second, and this is the main disadvantage of SRM, the method is typically computationally intractable: for most hypothesis sets, finding the solution of ERM is NP-hard and in general SRM requires determining that solution for a large number of indices k.

4.4 Cross-validation

An alternative method for model selection, *cross-validation*, consists of using some fraction of the training sample as a *validation set* to select a hypothesis set $\mathcal{H}_k$. This is in contrast with the SRM model which relies on a theoretical learning bound assigning a penalty to each hypothesis set. In this section, we analyze the cross-validation method and compare its performance to that of SRM.

As in the previous section, let $(\mathcal{H}_k)_{k\geq 1}$ be a countable sequence of hypothesis sets with increasing complexities. The cross-validation (CV) solution is obtained as follows. Let S be an i.i.d. labeled sample of size m. S is divided into a sample S_1 of size $(1-\alpha)m$ and a sample S_2 of size αm, with $\alpha \in (0,1)$ typically chosen to be relatively small. S_1 is reserved for training, S_2 for validation. For any $k \in \mathbb{N}$, let $h^{\mathrm{ERM}}_{S_1,k}$ denote the solution of ERM run on S_1 using the hypothesis set $\mathcal{H}_k$. The hypothesis h^{CV}_S returned by cross-validation is the ERM solution $h^{\mathrm{ERM}}_{S_1,k}$ with the best performance on S_2:

$$h^{\mathrm{CV}}_S = \operatorname*{argmin}_{h \in \left\{h^{\mathrm{ERM}}_{S_1,k} \colon k \geq 1\right\}} \widehat{R}_{S_2}(h). \tag{4.7}$$

The following general result will help us derive learning guarantees for cross-validation.

Proposition 4.3 *For any $\alpha > 0$ and any sample size $m \geq 1$, the following general inequality holds:*

$$\mathbb{P}\left[\sup_{k\geq 1}\left|R(h^{\mathrm{ERM}}_{S_1,k}) - \widehat{R}_{S_2}(h^{\mathrm{ERM}}_{S_1,k})\right| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\right] \leq 4e^{-2\alpha m\epsilon^2}.$$

Proof: By the union bound, we can write

$$\begin{aligned}
&\mathbb{P}\left[\sup_{k\geq 1}\left|R(h^{\mathrm{ERM}}_{S_1,k}) - \widehat{R}_{S_2}(h^{\mathrm{ERM}}_{S_1,k})\right| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\right] \\
&\qquad\leq \sum_{k=1}^{\infty}\mathbb{P}\left[\left|R(h^{\mathrm{ERM}}_{S_1,k}) - \widehat{R}_{S_2}(h^{\mathrm{ERM}}_{S_1,k})\right| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\right] \\
&\qquad= \sum_{k=1}^{\infty}\mathbb{E}\left[\mathbb{P}\left[\left|R(h^{\mathrm{ERM}}_{S_1,k}) - \widehat{R}_{S_2}(h^{\mathrm{ERM}}_{S_1,k})\right| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\,\middle|\, S_1\right]\right].
\end{aligned} \tag{4.8}$$

The hypothesis $h^{\mathrm{ERM}}_{S_1,k}$ is fixed conditioned on S_1. Furthermore, the sample S_2 is independent from S_1. Therefore, by Hoeffding's inequality, we can bound the conditional probability as follows:

$$\begin{aligned}
\mathbb{P}\left[\left|R(h^{\mathrm{ERM}}_{S_1,k}) - \widehat{R}_{S_2}(h^{\mathrm{ERM}}_{S_1,k})\right| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\,\middle|\, S_1\right] &\leq 2e^{-2\alpha m\left(\epsilon+\sqrt{\frac{\log k}{\alpha m}}\right)^2}. \\
&\leq 2e^{-2\alpha m\epsilon^2-2\log k} \\
&= \frac{2}{k^2}e^{-2\alpha m\epsilon^2}.
\end{aligned}$$

Plugging in the right-hand side of this bound in (4.8) and summing over k yields

$$\mathbb{P}\left[\sup_{k\geq 1}\left|R(h^{\mathrm{ERM}}_{S_1,k}) - \widehat{R}_{S_2}(h^{\mathrm{ERM}}_{S_1,k})\right| > \epsilon + \sqrt{\frac{\log k}{\alpha m}}\right] \leq \frac{\pi^2}{3}e^{-2\alpha m\epsilon^2} < 4e^{-2\alpha m\epsilon^2},$$

which completes the proof. □

Let $R(h^{\mathrm{SRM}}_{S_1})$ be the generalization error of the SRM solution using a sample S_1 of size $(1-\alpha m)$ and $R(h^{\mathrm{CV}}_S, S)$ the generalization error of the cross-validation solution using a sample S of size m. Then, using Proposition 4.3, the following learning guarantee can be derived which compares the error of the CV method to that of SRM.

Theorem 4.4 (Cross-validation versus SRM) *For any $\delta > 0$, with probability at least $1-\delta$, the following holds:*

$$R(h^{\mathrm{CV}}_S) - R(h^{\mathrm{SRM}}_{S_1}) \leq 2\sqrt{\frac{\log\max(k(h^{\mathrm{CV}}_S), k(h^{\mathrm{SRM}}_{S_1}))}{\alpha m}} + 2\sqrt{\frac{\log\frac{4}{\delta}}{2\alpha m}},$$

where, for any h, $k(h)$ denotes the smallest index of a hypothesis set containing h.

Proof: By Proposition 4.3 and Theorem 4.2, using the property of h^{CV}_S as a minimizer, for any $\delta > 0$, with probability at least $1-\delta$, the following inequalities hold:

$$\begin{aligned}
R(h_S^{\mathrm{CV}}) &\leq \widehat{R}_{S_2}(h_S^{\mathrm{CV}}) + \sqrt{\frac{\log(k(h_S^{\mathrm{CV}}))}{\alpha m}} + \sqrt{\frac{\log \frac{4}{\delta}}{2\alpha m}} \\
&\leq \widehat{R}_{S_2}(h_{S_1}^{\mathrm{SRM}}) + \sqrt{\frac{\log(k(h_S^{\mathrm{CV}}))}{\alpha m}} + \sqrt{\frac{\log \frac{4}{\delta}}{2\alpha m}} \\
&\leq R(h_{S_1}^{\mathrm{SRM}}) + \sqrt{\frac{\log(k(h_S^{\mathrm{CV}}))}{\alpha m}} + \sqrt{\frac{\log(k(h_{S_1}^{\mathrm{SRM}}))}{\alpha m}} + 2\sqrt{\frac{\log \frac{4}{\delta}}{2\alpha m}} \\
&\leq R(h_{S_1}^{\mathrm{SRM}}) + 2\sqrt{\frac{\log(\max(k(h_S^{\mathrm{CV}}), k(h_{S_1}^{\mathrm{SRM}}))}{\alpha m}} + 2\sqrt{\frac{\log \frac{4}{\delta}}{2\alpha m}},
\end{aligned}$$

which completes the proof. □

The learning guarantee just proven shows that, with high probability, the generalization error of the CV solution for a sample of size m is close to that of the SRM solution for a sample of size $(1-\alpha)m$. For α relatively small, this suggests a guarantee similar to that of SRM, which, as previously discussed, is very favorable. However, in some unfavorable regimes, an algorithm (here SRM) trained on $(1-\alpha)m$ points may have a significantly worse performance than when trained on m points (avoiding this phase transition issue is one of the main motivations behind the use of the n-fold cross-validation method in practice, see section 4.5). Thus, the bound suggests in fact a trade-off: α should be chosen sufficiently small to avoid the unfavorable regimes just mentioned and yet sufficiently large for the right-hand side of the bound to be small and thus informative.

The learning bound for CV can be made more explicit in some cases in practice. Assume for example that the hypothesis sets $\mathcal{H}_k$ are nested and that the empirical errors of the ERM solutions $h_{S_1,k}^{\mathrm{ERM}}$ are decreasing before reaching zero: for any k, $\widehat{R}_{S_1}(h_{S_1,k+1}^{\mathrm{ERM}}) < \widehat{R}_{S_1}(h_{S_1,k}^{\mathrm{ERM}})$ for all k such that $\widehat{R}_{S_1}(h_{S_1,k}^{\mathrm{ERM}}) > 0$ and $\widehat{R}_{S_1}(h_{S_1,k+1}^{\mathrm{ERM}}) \leq \widehat{R}_{S_1}(h_{S_1,k}^{\mathrm{ERM}})$ otherwise. Observe that $\widehat{R}_{S_1}(h_{S_1,k}^{\mathrm{ERM}}) > 0$ implies at least one error for $h_{S_1,k}^{\mathrm{ERM}}$, therefore $\widehat{R}_{S_1}(h_{S_1,k}^{\mathrm{ERM}}) > \frac{1}{m}$. In view of that, we must then have $\widehat{R}_{S_1}(h_{S_1,n}^{\mathrm{ERM}}) = 0$ for all $n \geq m+1$. Thus, we have $h_{S_1,n}^{\mathrm{ERM}} = h_{S_1,m+1}^{\mathrm{ERM}}$ for all $n \geq m+1$ and we can assume that $k(f_{CV}) \leq m+1$. Since the complexity of $\mathcal{H}_k$ increases with k we also have $k(f_{SRM}) \leq m+1$. In view of that, we obtain the following more explicit learning bound for cross-validation:

$$R(f_{CV}, S) - R(f_{SRM}, S_1) \leq 2\sqrt{\frac{\log(\frac{4}{\delta})}{2\alpha m}} + 2\sqrt{\frac{\log(m+1)}{\alpha m}}.$$

4.5 n-Fold cross-validation

In practice, the amount of labeled data available is often too small to set aside a validation sample since that would leave an insufficient amount of training data. Instead, a widely adopted method known as *n-fold cross-validation* is used to exploit the labeled data both for *model selection* and for training.

Let $\boldsymbol{\theta}$ denote the vector of free parameters of the algorithm. For a fixed value of $\boldsymbol{\theta}$, the method consists of first randomly partitioning a given sample S of m labeled examples into n subsamples, or folds. The ith fold is thus a labeled sample $((x_{i1}, y_{i1}), \ldots, (x_{im_i}, y_{im_i}))$ of size m_i. Then, for any $i \in [n]$, the learning algorithm is trained on all but the ith fold to generate a hypothesis h_i, and the performance of h_i is tested on the ith fold, as illustrated in figure 4.5a. The parameter value $\boldsymbol{\theta}$ is evaluated based on the average error of the hypotheses h_i, which is called the *cross-validation error*. This quantity is denoted by $\widehat{R}_{\text{CV}}(\boldsymbol{\theta})$ and defined by

$$\widehat{R}_{\text{CV}}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \underbrace{\frac{1}{m_i} \sum_{j=1}^{m_i} L(h_i(x_{ij}), y_{ij})}_{\text{error of } h_i \text{ on the } i\text{th fold}} .$$

The folds are generally chosen to have equal size, that is $m_i = m/n$ for all $i \in [n]$. How should n be chosen? The appropriate choice is subject to a trade-off. For a large n, each training sample used in n-fold cross-validation has size $m - m/n = m(1 - 1/n)$ (illustrated by the right vertical red line in figure 4.5b), which is close to m, the size of the full sample, and also implies all training samples are quite similar. At the same time, the ith fold used to measure the error is relatively small and thus the cross-validation error tends to have a small bias but a large variance. In contrast, smaller values of n lead to more diverse training samples but their size (shown by the left vertical red line in figure 4.5b) is significantly less than m. In this regime, the ith fold is relatively large and thus the cross-validation error tends to have a smaller variance but a larger bias.

In applications, n is typically chosen to be 5 or 10. n-fold cross-validation is used as follows in model selection. The full labeled data is first split into a training and a test sample. The training sample of size m is then used to compute the n-fold cross-validation error $\widehat{R}_{\text{CV}}(\boldsymbol{\theta})$ for a small number of possible values of $\boldsymbol{\theta}$. The free parameter $\boldsymbol{\theta}$ is next set to the value $\boldsymbol{\theta}_0$ for which $\widehat{R}_{\text{CV}}(\boldsymbol{\theta})$ is smallest and the algorithm is trained with the parameter setting $\boldsymbol{\theta}_0$ over the full training sample of size m. Its performance is evaluated on the test sample as already described in the previous section.

The special case of n-fold cross-validation where $n = m$ is called *leave-one-out cross-validation*, since at each iteration exactly one instance is left out of the train-

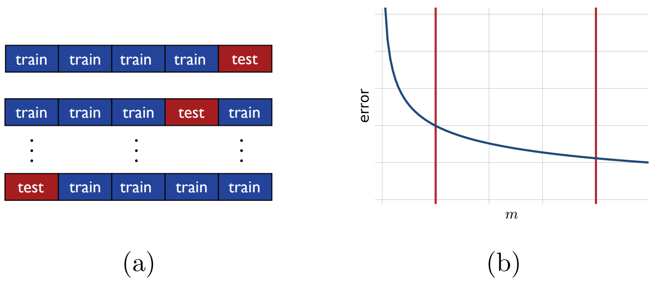


Figure 4.5
n-fold cross-validation. (a) Illustration of the partitioning of the training data into 5 folds. (b) Typical plot of a classifier's prediction error as a function of the size of the training sample m: the error decreases as a function of the number of training points. The red line on the left side marks the region for small values of n, while the red line on the right side marks the region for large values of n.

ing sample. As shown in chapter 5, the average leave-one-out error is an approximately unbiased estimate of the average error of an algorithm and can be used to derive simple guarantees for some algorithms. In general, the leave-one-out error is very costly to compute, since it requires training m times on samples of size $m-1$, but for some algorithms it admits a very efficient computation (see exercise 11.9).

In addition to model selection, n-fold cross-validation is also commonly used for performance evaluation. In that case, for a fixed parameter setting $\boldsymbol{\theta}$, the full labeled sample is divided into n random folds with no distinction between training and test samples. The performance reported is the n-fold cross-validation error on the full sample as well as the standard deviation of the errors measured on each fold.

4.6 Regularization-based algorithms

A broad family of algorithms inspired by the SRM method is that of *regularization-based algorithm*. This consists of selecting a very complex family $\mathcal{H}$ that is an uncountable union of nested hypothesis sets $\mathcal{H}_\gamma$: $\mathcal{H} = \bigcup_{\gamma>0} \mathcal{H}_\gamma$. $\mathcal{H}$ is often chosen to be dense in the space of continuous functions over $\mathcal{X}$. For example, $\mathcal{H}$ may be chosen to be the set of all linear functions in some high-dimensional space and $\mathcal{H}_\gamma$ the subset of those functions whose norm is bounded by γ: $\mathcal{H}_\gamma = \{x \mapsto \mathbf{w} \cdot \mathbf{\Phi}(x) \colon \|\mathbf{w}\| \leq \gamma\}$. For some choices of $\mathbf{\Phi}$ and the high-dimensional space, it can be shown that $\mathcal{H}$ is indeed dense in the space of continuous functions over $\mathcal{X}$.

Given a labeled sample S, the extension of the SRM method to an uncountable union would then suggest selecting h based on the following optimization problem:

$$\operatorname*{argmin}_{\gamma>0, h\in H_\gamma} \widehat{R}_S(h) + \mathfrak{R}_m(\mathcal{H}_\gamma) + \sqrt{\frac{\log\gamma}{m}},$$

where other penalty terms $\text{pen}(\gamma, m)$ can be chosen in lieu of the specific choice $\text{pen}(\gamma, m) = \mathfrak{R}_m(\mathcal{H}_\gamma) + \sqrt{\frac{\log\gamma}{m}}$. Often, there exists a function $\mathcal{R}\colon \mathcal{H} \to \mathbb{R}$ such that, for any $\gamma > 0$, the constrained optimization problem $\operatorname{argmin}_{\gamma>0, h\in H_\gamma} \widehat{R}_S(h) + \text{pen}(\gamma, m)$ can be equivalently written as the unconstrained optimization problem

$$\operatorname*{argmin}_{h\in\mathcal{H}} \widehat{R}_S(h) + \lambda\mathcal{R}(h),$$

for some $\lambda > 0$. $\mathcal{R}(h)$ is called a *regularization term* and $\lambda > 0$ is treated as a hyperparameter since its optimal value is often not known. For most algorithms, the regularization term $\mathcal{R}(h)$ is chosen to be an increasing function of $\|h\|$ for some choice of the norm $\|\cdot\|$, when $\mathcal{H}$ is the subset of a Hilbert space. The variable λ is often called a *regularization parameter*. Larger values of λ further penalize more complex hypotheses, while, for λ close or equal to zero, the regularization term has no effect and the algorithm coincides with ERM. In practice, λ is typically selected via cross-validation or using n-fold cross-validation.

When the regularization term is chosen to be $\|h\|_p$ for some choice of the norm and $p \geq 1$, then it is a convex function of h, since any norm is convex. However, for the zero-one loss, the first term of the objective function is non-convex, thereby making the optimization problem computationally hard. In practice, most regularization-based algorithms instead use a convex upper bound on the zero-one loss and replace the empirical zero-one term with the empirical value of that convex surrogate. The resulting optimization problem is then convex and therefore admits more efficient solutions than SRM. The next section studies the properties of such convex surrogate losses.

4.7 Convex surrogate losses

The guarantees for the estimation error that we presented in previous sections hold either for ERM or for SRM, which itself is defined in terms of ERM. However, as already mentioned, for many choices of the hypothesis set $\mathcal{H}$, including that of linear functions, solving the ERM optimization problem is NP-hard mainly because the zero-one loss function is not convex. One common method for addressing this problem consists of using a convex surrogate loss function that upper bounds the zero-one loss. This section analyzes learning guarantees for such surrogate losses in terms of the original loss.

The hypotheses we consider are real-valued functions $h\colon \mathcal{X} \to \mathbb{R}$. The sign of h defines a binary classifier $f_h\colon \mathcal{X} \to \{-1, +1\}$ defined for all $x \in \mathcal{X}$ by

$$f_h(x) = \begin{cases} +1 & \text{if } h(x) \geq 0 \\ -1 & \text{if } h(x) < 0. \end{cases}$$

The loss or error of h at point $(x, y) \in \mathcal{X} \times \{-1, +1\}$ is defined as the binary classification error of f_h:

$$1_{f_h(x) \neq y} = 1_{yh(x)<0} + 1_{h(x)=0 \wedge y=-1} \leq 1_{yh(x) \leq 0}.$$

We will denote by $R(h)$ the expected error of h: $R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[1_{f_h(x)\neq y}\right]$. For any $x \in \mathcal{X}$, let $\eta(x)$ denote $\eta(x) = \mathbb{P}[y = +1|x]$ and let $\mathcal{D}_{\mathcal{X}}$ denote the marginal distribution over $\mathcal{X}$. Then, for any h, we can write

$$\begin{aligned} R(h) &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\left[1_{f_h(x)\neq y}\right] \\ &= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}}}\left[\eta(x)1_{h(x)<0} + (1-\eta(x))1_{h(x)>0} + (1-\eta(x))1_{h(x)=0}\right] \\ &= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}}}\left[\eta(x)1_{h(x)<0} + (1-\eta(x))1_{h(x)\geq 0}\right]. \end{aligned}$$

In view of that, the Bayes classifier can be defined as assigning label $+1$ to x when $\eta(x) \geq \frac{1}{2}$, -1 otherwise. It can therefore be induced by the function h^* defined by

$$h^*(x) = \eta(x) - \tfrac{1}{2}. \tag{4.9}$$

We will refer to $h^*\colon \mathcal{X} \to \mathbb{R}$ as the *Bayes scoring function* and will denote by R^* the error of the Bayes classifier or Bayes scoring function: $R^* = R(h^*)$.

Lemma 4.5 *The excess error of any hypothesis $h\colon \mathcal{X} \to \mathbb{R}$ can be expressed as follows in terms of η and the Bayes scoring function h^*:*

$$R(h) - R^* = 2 \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}}}\Big[|h^*(x)|\, 1_{h(x)h^*(x)\leq 0}\Big].$$

Proof: For any h, we can write

$$\begin{aligned} R(h) &= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}}}\Big[\eta(x)1_{h(x)<0} + (1-\eta(x))1_{h(x)\geq 0}\Big] \\ &= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}}}\Big[\eta(x)1_{h(x)<0} + (1-\eta(x))(1 - 1_{h(x)<0})\Big] \\ &= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}}}\Big[[2\eta(x) - 1]1_{h(x)<0} + (1-\eta(x))\Big] \\ &= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}}}\Big[2h^*(x)1_{h(x)<0} + (1-\eta(x))\Big], \end{aligned}$$

where we used for the last step equation (4.9). In view of that, for any h, the following holds:

$$\begin{aligned} R(h) - R(h^*) &= \mathop{\mathbb{E}}_{x \sim \mathcal{D}_\mathcal{X}} \Big[2[h^*(x)](1_{h(x) \leq 0} - 1_{h^*(x) \leq 0}) \Big] \\ &= \mathop{\mathbb{E}}_{x \sim \mathcal{D}_\mathcal{X}} \Big[2[h^*(x)] \operatorname{sgn}(h^*(x)) 1_{(h(x)h^*(x) \leq 0) \wedge ((h(x), h^*(x)) \neq (0,0))} \Big] \\ &= 2 \mathop{\mathbb{E}}_{x \sim \mathcal{D}_\mathcal{X}} \Big[|h^*(x)| \, 1_{h(x)h^*(x) \leq 0} \Big], \end{aligned}$$

which completes the proof, since $R(h^*) = R^*$. □

Let $\Phi \colon \mathbb{R} \to \mathbb{R}$ be a convex and non-decreasing function so that for any $u \in \mathbb{R}$, $1_{u \leq 0} \leq \Phi(-u)$. The Φ-loss of a function $h \colon \mathcal{X} \to \mathbb{R}$ at point $(x, y) \in \mathcal{X} \times \{-1, +1\}$ is defined as $\Phi(-yh(x))$ and its expected loss given by

$$\begin{aligned} \mathcal{L}_\Phi(h) &= \mathop{\mathbb{E}}_{(x,y) \sim \mathcal{D}} \big[\Phi(-yh(x)) \big] \\ &= \mathop{\mathbb{E}}_{x \sim \mathcal{D}_\mathcal{X}} \Big[\eta(x)\Phi(-h(x)) + (1 - \eta(x))\Phi(h(x)) \Big]. \end{aligned} \tag{4.10}$$

Notice that since $1_{yh(x) \leq 0} \leq \Phi(-yh(x))$, we have $R(h) \leq \mathcal{L}_\Phi(h)$. For any $x \in \mathcal{X}$, let $u \mapsto L_\Phi(x, u)$ be the function defined for all $u \in \mathbb{R}$ by

$$L_\Phi(x, u) = \eta(x)\Phi(-u) + (1 - \eta(x))\Phi(u).$$

Then, $\mathcal{L}_\Phi(h) = \mathbb{E}_{x \sim \mathcal{D}_\mathcal{X}}[L_\Phi(x, h(x))]$. Since Φ is convex, $u \mapsto L_\Phi(x, u)$ is convex as a sum of two convex functions. Define $h^*_\Phi \colon \mathcal{X} \to [-\infty, +\infty]$ as the *Bayes solution for the loss function* L_Φ. That is, for any x, $h^*_\Phi(x)$ is a solution of the following convex optimization problem:

$$\begin{aligned} h^*_\Phi(x) &= \operatorname*{argmin}_{u \in [-\infty, +\infty]} L_\Phi(x, u) \\ &= \operatorname*{argmin}_{u \in [-\infty, +\infty]} \eta(x)\Phi(-u) + (1 - \eta(x))\Phi(u). \end{aligned}$$

The solution of this optimization is in general not unique. When $\eta(x) = 0$, $h^*_\Phi(x)$ is a minimizer of $u \mapsto \Phi(u)$ and since Φ is non-decreasing, we can choose $h^*_\Phi(x) = -\infty$ in that case. Similarly, when $\eta(x) = 1$, we can choose $h^*_\Phi(x) = +\infty$. When $\eta(x) = \frac{1}{2}$, $L_\Phi(x, u) = \frac{1}{2}[\Phi(-u) + \Phi(u)]$, thus, by convexity, $L_\Phi(x, u) \geq \Phi(-\frac{u}{2} + \frac{u}{2}) = \Phi(0)$. Thus, we can choose $h^*_\Phi(x) = 0$ in that case. For all other values of $\eta(x)$, in case of non-uniqueness, an arbitrary minimizer is chosen in this definition. We will denote by $\mathcal{L}^*_\Phi$ the Φ-loss of h^*_Φ: $\mathcal{L}^*_\Phi = \mathbb{E}_{(x,y) \sim \mathcal{D}} \big[\Phi(-yh^*_\Phi(x)) \big]$.

Proposition 4.6 *Let Φ be a convex and non-decreasing function that is differentiable at 0 with $\Phi'(0) > 0$. Then, the minimizer of Φ defines the Bayes classifier: for any $x \in \mathcal{X}$, $h^*_\Phi(x) > 0$ iff $h^*(x) > 0$ and $h^*(x) = 0$ iff $h^*_\Phi(x) = 0$, which implies $\mathcal{L}^*_\Phi = R^*$.*

Proof: Fix $x \in \mathcal{X}$. If $\eta(x) = 0$, then $h^*(x) = -\frac{1}{2}$ and $h^*_\Phi(x) = -\infty$, thus $h^*(x)$ and $h^*_\Phi(x)$ admit the same sign. Similarly, if $\eta(x) = 1$, then $h^*(x) = +\frac{1}{2}$ and $h^*_\Phi(x) = +\infty$, and $h^*(x)$ and $h^*_\Phi(x)$ admit the same sign.

Let u^* denote the minimizer defining $h^*_\Phi(x)$. u^* is a minimizer of $u \mapsto L_\Phi(x, u)$ iff the subdifferential of that function at u^* contains 0, that is, since $\partial L_\Phi(x, u^*) = -\eta(x)\partial\Phi(-u^*) + (1 - \eta(x))\partial\Phi(u^*)$, iff there exist $v_1 \in \partial\Phi(-u^*)$ and $v_2 \in \partial\Phi(u^*)$ such that

$$\eta(x)v_1 = (1 - \eta(x))v_2. \tag{4.11}$$

If $u^* = 0$, by the differentiability of Φ at 0 we have $v_1 = v_2 = \Phi'(0) > 0$ and thus $\eta(x) = \frac{1}{2}$, that is $h^*(x) = 0$. Conversely, If $h^*(x) = 0$, that is $\eta(x) = \frac{1}{2}$, then, by definition, we have $h^*_\Phi(x) = 0$. Thus, $h^*(x) = 0$ iff $h^*_\Phi(x) = 0$ iff $\eta(x) = \frac{1}{2}$.

We can assume now that $\eta(x)$ is not in $\{0, 1, \frac{1}{2}\}$. We first show that for any $u_1, u_2 \in \mathbb{R}$ with $u_1 < u_2$, and any two choices of the subgradients at u_1 and u_2, $v_1 \in \partial\Phi(u_1)$ and $v_2 \in \partial\Phi(u_2)$, we have $v_1 \leq v_2$. By definition of the subgradients at u_1 and u_2, the following inequalities hold:

$$\Phi(u_2) - \Phi(u_1) \geq v_1(u_2 - u_1) \qquad \Phi(u_1) - \Phi(u_2) \geq v_2(u_1 - u_2).$$

Summing up these inequalities yields $v_2(u_2 - u_1) \geq v_1(u_2 - u_1)$ and thus $v_2 \geq v_1$, since $u_1 < u_2$.

Now, if $u^* > 0$, then we have $-u^* < u^*$. By the property shown above, this implies $v_1 \leq v_2$. We cannot have $v_1 = v_2 \neq 0$ since (4.11) would then imply $\eta(x) = \frac{1}{2}$. We also cannot have $v_1 = v_2 = 0$ since by the property shown above, we must have $\Phi'(0) \leq v_2$ and thus $v_2 > 0$. Thus, we must have $v_1 < v_2$ with $v_2 > 0$, which, by (4.11), implies $\eta(x) > 1 - \eta(x)$, that is $h^*(x) > 0$.

Conversely, if $h^*(x) > 0$ then $\eta(x) > 1 - \eta(x)$. We cannot have $v_1 = v_2 = 0$ or $v_1 = v_2 \neq 0$ as already shown. Thus, since $\eta(x) \neq 1$, by (4.11), this implies $v_1 < v_2$. We cannot have $u^* < -u^*$ since, by the property shown above, this would imply $v_2 \leq v_1$. Thus, we must have $-u^* \leq u^*$, that is $u^* \geq 0$, and more specifically $u^* > 0$ since, as already shown above, $u^* = 0$ implies $h^*(x) = 0$. □

Theorem 4.7 *Let Φ be a convex and non-decreasing function. Assume that there exists $s \geq 1$ and $c > 0$ such that the following holds for all $x \in \mathcal{X}$:*

$$|h^*(x)|^s = \left|\eta(x) - \tfrac{1}{2}\right|^s \leq c^s \big[L_\Phi(x, 0) - L_\Phi(x, h^*_\Phi(x))\big].$$

Then, for any hypothesis h, the excess error of h is bounded as follows:

$$R(h) - R^* \leq 2c \left[\mathcal{L}_\Phi(h) - \mathcal{L}^*_\Phi\right]^{\frac{1}{s}}$$

Proof: We will use the following inequality which holds by the convexity of Φ:

$$\begin{aligned}\Phi\big(-2h^*(x)h(x)\big) &= \Phi\big((1-2\eta(x))h(x)\big)\\ &= \Phi\big(\eta(x)(-h(x)) + (1-\eta(x))h(x)\big)\\ &\leq \eta(x)\Phi((-h(x))) + (1-\eta(x))\Phi(h(x)) = L_\Phi(x, h(x)). \quad (4.12)\end{aligned}$$

By Lemma 4.5, Jensen's inequality, and $h^*(x) = \eta(x) - \frac{1}{2}$, we can write

$$\begin{aligned}&R(h) - R(h^*)\\ &= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_\mathcal{X}}\Big[|2\eta(x)-1|\, 1_{h(x)h^*(x)\leq 0}\Big]\\ &\leq \mathop{\mathbb{E}}_{x\sim\mathcal{D}_\mathcal{X}}\Big[|2\eta(x)-1|^s\, 1_{h(x)h^*(x)\leq 0}\Big]^{\frac{1}{s}} && \text{(Jensen's ineq.)}\\ &\leq 2c \mathop{\mathbb{E}}_{x\sim\mathcal{D}_\mathcal{X}}\Big[\big[\Phi(0) - L_\Phi(x, h^*_\Phi(x))\big]\, 1_{h(x)h^*(x)\leq 0}\Big]^{\frac{1}{s}} && \text{(assumption)}\\ &\leq 2c \mathop{\mathbb{E}}_{x\sim\mathcal{D}_\mathcal{X}}\Big[\big[\Phi\big(-2h^*(x)h(x)\big) - L_\Phi(x, h^*_\Phi(x))\big]\, 1_{h(x)h^*(x)\leq 0}\Big]^{\frac{1}{s}} && \text{(Φ non-decreasing)}\\ &\leq 2c \mathop{\mathbb{E}}_{x\sim\mathcal{D}_\mathcal{X}}\Big[\big[L_\Phi(x, h(x)) - L_\Phi(x, h^*_\Phi(x))\big]\, 1_{h(x)h^*(x)\leq 0}\Big]^{\frac{1}{s}} && \text{(convexity ineq. (4.12))}\\ &\leq 2c \mathop{\mathbb{E}}_{x\sim\mathcal{D}_\mathcal{X}}\Big[L_\Phi(x, h(x)) - L_\Phi(x, h^*_\Phi(x))\Big]^{\frac{1}{s}},\end{aligned}$$

which completes the proof, since $\mathbb{E}_{x\sim\mathcal{D}_\mathcal{X}}[L_\Phi(x, h^*_\Phi(x))] = L^*_\Phi$. □

The theorem shows that, when the assumption holds, the excess error of h can be upper bounded in terms of the excess Φ-loss. The assumption of the theorem holds in particular for the following convex loss functions:

- Hinge loss, where $\Phi(u) = \max(0, 1+u)$, with $s = 1$ and $c = \frac{1}{2}$.
- Exponential loss, where $\Phi(u) = \exp(u)$, with $s = 2$ and $c = \frac{1}{\sqrt{2}}$.
- Logistic loss, where $\Phi(u) = \log_2(1+e^u)$, with $s = 2$ and $c = \frac{1}{\sqrt{2}}$.

They also hold for the square loss and the squared Hinge loss (see Exercises 4.2 and 4.3).

4.8 Chapter notes

The structural risk minimization (SRM) technique is due to Vapnik [1998]. The original penalty term used by Vapnik [1998] is based on the VC-dimension of the hypothesis set. The version of SRM with Rademacher complexity-based penalties that we present here leads to finer data-dependent learning guarantees. Penalties based on alternative complexity measures can be used similarly leading to learning bounds in terms of the corresponding complexity measure [Bartlett et al., 2002a].

An alternative model selection theory of *Voted Risk Minimization* (VRM) has been recently developed by Cortes, Mohri, and Syed [2014] and other related publications [Kuznetsov et al., 2014, DeSalvo et al., 2015, Cortes et al., 2015].

Theorem 4.7 is due to Zhang [2003a]. The proof given here is somewhat different and simpler.

4.9 Exercises

4.1 For any hypothesis set $\mathcal{H}$, show that the following inequalities hold:

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}\left[\widehat{R}_S\left(h_S^{\mathrm{ERM}}\right)\right] \leq \inf_{h\in\mathcal{H}} R(h) \leq \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}\left[R\left(h_S^{\mathrm{ERM}}\right)\right]. \tag{4.13}$$

4.2 Show that for the squared loss, $\Phi(u) = (1+u)^2$, the statement of Theorem 4.7 holds with $s = 2$ and $c = \frac{1}{2}$ and therefore that the excess error can be upper bounded as follows:

$$R(h) - R^* \leq \left[\mathcal{L}_\Phi(h) - \mathcal{L}_\Phi^*\right]^{\frac{1}{2}}.$$

4.3 Show that for the squared Hinge loss, $\Phi(u) = \max(0, 1+u)^2$, the statement of Theorem 4.7 holds with $s = 2$ and $c = \frac{1}{2}$ and therefore that the excess error can be upper bounded as follows:

$$R(h) - R^* \leq \left[\mathcal{L}_\Phi(h) - \mathcal{L}_\Phi^*\right]^{\frac{1}{2}}.$$

4.4 In this problem, the loss of $h\colon \mathcal{X} \to \mathbb{R}$ at point $(x, y) \in \mathcal{X} \times \{-1, +1\}$ is defined to be $1_{yh(x)\leq 0}$.

(a) Define the Bayes classifier and a Bayes scoring function h^* for this loss.

(b) Express the excess error of h in terms of h^* (counterpart of Lemma 4.5, for loss considered here).

(c) Give a counterpart of the result of Theorem 4.7 for this loss.

4.5 Same questions as in Exercise 4.5 with the loss of $h\colon \mathcal{X} \to \mathbb{R}$ at point $(x, y) \in \mathcal{X} \times \{-1, +1\}$ defined instead to be $1_{yh(x)<0}$.

5 Support Vector Machines

This chapter presents one of the most theoretically well motivated and practically most effective classification algorithms in modern machine learning: Support Vector Machines (SVMs). We first introduce the algorithm for separable datasets, then present its general version designed for non-separable datasets, and finally provide a theoretical foundation for SVMs based on the notion of margin. We start with the description of the problem of linear classification.

5.1 Linear classification

Consider an input space $\mathcal{X}$ that is a subset of $\mathbb{R}^N$ with $N \geq 1$, and the output or target space $\mathcal{Y} = \{-1, +1\}$, and let $f \colon \mathcal{X} \to \mathcal{Y}$ be the target function. Given a hypothesis set $\mathcal{H}$ of functions mapping $\mathcal{X}$ to $\mathcal{Y}$, the binary classification task is formulated as follows. The learner receives a training sample S of size m drawn i.i.d. from $\mathcal{X}$ according to some unknown distribution $\mathcal{D}$, $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$, with $y_i = f(x_i)$ for all $i \in [m]$. The problem consists of determining a hypothesis $h \in \mathcal{H}$, a *binary classifier*, with small generalization error:

$$R_{\mathcal{D}}(h) = \mathop{\mathbb{P}}_{x \sim \mathcal{D}}[h(x) \neq f(x)]. \tag{5.1}$$

Different hypothesis sets $\mathcal{H}$ can be selected for this task. In view of the results presented in chapter 3, which formalized Occam's razor principle, hypothesis sets with smaller complexity — e.g., smaller VC-dimension or Rademacher complexity — provide better learning guarantees, everything else being equal. A natural hypothesis set with relatively small complexity is that of *linear classifiers*, or hyperplanes, which can be defined as follows:

$$\mathcal{H} = \{\mathbf{x} \mapsto \operatorname{sign}(\mathbf{w} \cdot \mathbf{x} + b) \colon \mathbf{w} \in \mathbb{R}^N, b \in \mathbb{R}\}. \tag{5.2}$$

The learning problem is then referred to as a *linear classification problem*. The general equation of a hyperplane in $\mathbb{R}^N$ is $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w} \in \mathbb{R}^N$ is a

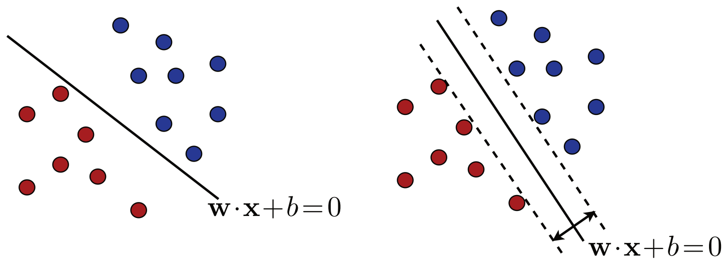

$\mathbf{w}\cdot\mathbf{x}+b=0$
$\mathbf{w}\cdot\mathbf{x}+b=0$

Figure 5.1
Two possible separating hyperplanes. The right-hand side figure shows a hyperplane that maximizes the margin.

non-zero vector normal to the hyperplane and $b \in \mathbb{R}$ a scalar. A hypothesis of the form $\mathbf{x} \mapsto \operatorname{sign}(\mathbf{w} \cdot \mathbf{x} + b)$ thus labels positively all points falling on one side of the hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ and negatively all others.

5.2 Separable case

In this section, we assume that the training sample S can be linearly separated, that is, we assume the existence of a hyperplane that perfectly separates the training sample into two populations of positively and negatively labeled points, as illustrated by the left panel of figure 5.1. This is equivalent to the existence of $(\mathbf{w}, b) \in (\mathbb{R}^N - \{\mathbf{0}\}) \times \mathbb{R}$ such that

$$\forall i \in [m], \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 0. \tag{5.3}$$

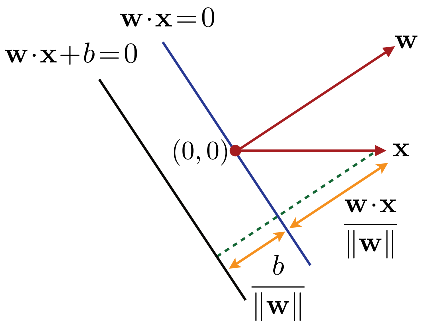
But, as can be seen from figure 5.1, there are then infinitely many such separating hyperplanes. Which hyperplane should a learning algorithm select? The definition of the SVM solution is based on the notion of *geometric margin*.

Definition 5.1 (Geometric margin) *The* geometric margin $\rho_h(\mathbf{x})$ *of a linear classifier* $h\colon \mathbf{x} \mapsto \mathbf{w}\cdot\mathbf{x}+b$ *at a point* $\mathbf{x}$ *is its Euclidean distance to the hyperplane* $\mathbf{w}\cdot\mathbf{x}+b = 0$*:*

$$\rho_h(x) = \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|_2}. \tag{5.4}$$

The geometric margin ρ_h *of a linear classifier* h *for a sample* $S = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ *is the minimum geometric margin over the points in the sample,* $\rho_h = \min_{i \in [m]} \rho_h(x_i)$*, that is the distance of the hyperplane defining* h *to the closest sample points.*

The SVM solution is the separating hyperplane with the maximum geometric margin and is thus known as the *maximum-margin hyperplane*. The right panel of figure 5.1 illustrates the maximum-margin hyperplane returned by the SVM



$\mathbf{w}\cdot\mathbf{x}=0$
$\mathbf{w}\cdot\mathbf{x}+b=0$
$\mathbf{w}$
$(0,0)$
$\mathbf{x}$
$\frac{\mathbf{w}\cdot\mathbf{x}}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$

Figure 5.2
An illustration of the geometric margin of a point $\mathbf{x}$ in the case $\mathbf{w}\cdot\mathbf{x} > 0$ and $b > 0$.

algorithm in the separable case. We will present later in this chapter a theory that provides a strong justification for this solution. We can observe already, however, that the SVM solution can also be viewed as the "safest" choice in the following sense: a test point is classified correctly by a separating hyperplane with geometric margin ρ even when it falls within a distance ρ of the training samples sharing the same label; for the SVM solution, ρ is the maximum geometric margin and thus the "safest" value.

5.2.1 Primal optimization problem

We now derive the equations and optimization problem that define the SVM solution. By definition of the geometric margin (see also figure 5.2), the maximum margin ρ of a separating hyperplane is given by

$$\rho = \max_{\mathbf{w},b\colon\, y_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 0} \ \min_{i\in[m]} \frac{|\mathbf{w}\cdot\mathbf{x}_i + b|}{\|\mathbf{w}\|} = \max_{\mathbf{w},b} \ \min_{i\in[m]} \frac{y_i\big(\mathbf{w}\cdot\mathbf{x}_i + b\big)}{\|\mathbf{w}\|}. \tag{5.5}$$

The second equality follows from the fact that, since the sample is linearly separable, for the maximizing pair $(\mathbf{w}, b)$, $y_i\big(\mathbf{w}\cdot\mathbf{x}_i + b\big)$ must be non-negative for all $i \in [m]$. Now, observe that the last expression is invariant to multiplication of $(\mathbf{w}, b)$ by a positive scalar. Thus, we can restrict ourselves to pairs $(\mathbf{w}, b)$ scaled such that $\min_{i\in[m]} y_i(\mathbf{w}\cdot\mathbf{x}_i + b) = 1$:

$$\rho = \max_{\substack{\mathbf{w},b\colon\\ \min_{i\in[m]} y_i(\mathbf{w}\cdot\mathbf{x}_i+b)=1}} \frac{1}{\|\mathbf{w}\|} = \max_{\substack{\mathbf{w},b\colon\\ \forall i\in[m], y_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 1}} \frac{1}{\|\mathbf{w}\|}. \tag{5.6}$$

The second equality results from the fact that for the maximizing pair $(\mathbf{w}, b)$, the minimum of $y_i(\mathbf{w}\cdot\mathbf{x}_i + b)$ is 1.

Figure 5.3 illustrates the solution $(\mathbf{w}, b)$ of the maximization (5.6). In addition to the maximum-margin hyperplane, it also shows the *marginal hyperplanes*, which are

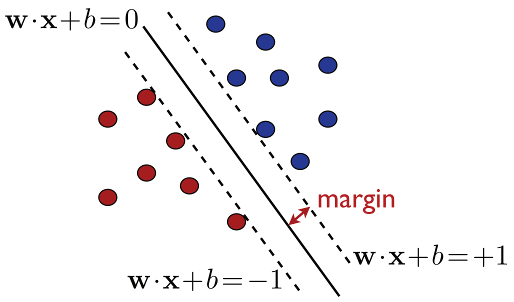

$\mathbf{w}\cdot\mathbf{x}+b=0$
margin
$\mathbf{w}\cdot\mathbf{x}+b=+1$
$\mathbf{w}\cdot\mathbf{x}+b=-1$

Figure 5.3
Maximum-margin hyperplane solution of (5.6). The marginal hyperplanes are represented by dashed lines on the figure.

the hyperplanes parallel to the separating hyperplane and passing through the closest points on the negative or positive sides. Since they are parallel to the separating hyperplane, they admit the same normal vector $\mathbf{w}$. Furthermore, since $|\mathbf{w}\cdot\mathbf{x}+b| = 1$ for the closest points, the equations of the marginal hyperplanes are $\mathbf{w}\cdot\mathbf{x}+b = \pm 1$.

Since maximizing $1/\|\mathbf{w}\|$ is equivalent to minimizing $\frac{1}{2}\|\mathbf{w}\|^2$, in view of (5.6), the pair $(\mathbf{w}, b)$ returned by SVM in the separable case is the solution of the following convex optimization problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{5.7}$$
$$\text{subject to: } y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1, \; \forall i \in [m] .$$

The objective function $F\colon \mathbf{w} \mapsto \frac{1}{2}\|\mathbf{w}\|^2$ is infinitely differentiable. Its gradient is $\nabla F(\mathbf{w}) = \mathbf{w}$ and its Hessian is the identity matrix $\nabla^2 F(\mathbf{w}) = \mathbf{I}$, whose eigenvalues are strictly positive. Therefore, $\nabla^2 F(\mathbf{w}) \succ \mathbf{0}$ and F is strictly convex. The constraints are all defined by affine functions $g_i\colon (\mathbf{w}, b) \mapsto 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)$ and are therefore qualified. Thus, in view of the results known for convex optimization (see appendix B for details), the optimization problem of (5.7) admits a unique solution, an important and favorable property that does not hold for all learning algorithms.

Moreover, since the objective function is quadratic and the constraints are affine, the optimization problem of (5.7) is in fact a specific instance of *quadratic programming* (QP), a family of problems extensively studied in optimization. A variety of commercial and open-source solvers are available for solving convex QP problems. Additionally, motivated by the empirical success of SVMs along with its rich theoretical underpinnings, specialized methods have been developed to more efficiently solve this particular convex QP problem, notably the block coordinate descent algorithms with blocks of just two coordinates.

5.2.2 Support vectors

Returning to the optimization problem (5.7), we note that the constraints are affine and thus qualified. The objective function as well as the affine constraints are convex and differentiable. Thus, the requirements of theorem B.30 hold and the KKT conditions apply at the optimum. We shall use these conditions to both analyze the algorithm and demonstrate several of its crucial properties, and subsequently derive the dual optimization problem associated to SVMs in section 5.2.3.

We introduce Lagrange variables $\alpha_i \geq 0$, $i \in [m]$, associated to the m constraints and denote by $\boldsymbol{\alpha}$ the vector $(\alpha_1, \ldots, \alpha_m)^\top$. The Lagrangian can then be defined for all $\mathbf{w} \in \mathbb{R}^N$, $b \in \mathbb{R}$, and $\boldsymbol{\alpha} \in \mathbb{R}_+^m$, by

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^m \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] . \tag{5.8}$$

The KKT conditions are obtained by setting the gradient of the Lagrangian with respect to the primal variables $\mathbf{w}$ and b to zero and by writing the complementarity conditions:

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i = 0 \quad \Longrightarrow \quad \mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i \tag{5.9}$$

$$\nabla_b \mathcal{L} = -\sum_{i=1}^m \alpha_i y_i = 0 \quad \Longrightarrow \quad \sum_{i=1}^m \alpha_i y_i = 0 \tag{5.10}$$

$$\forall i, \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0 \quad \Longrightarrow \quad \alpha_i = 0 \vee y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1. \tag{5.11}$$

By equation (5.9), the weight vector $\mathbf{w}$ at the solution of the SVM problem is a linear combination of the training set vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$. A vector $\mathbf{x}_i$ appears in that expansion iff $\alpha_i \neq 0$. Such vectors are called *support vectors*. By the complementarity conditions (5.11), if $\alpha_i \neq 0$, then $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$. Thus, support vectors lie on the marginal hyperplanes $\mathbf{w} \cdot \mathbf{x}_i + b = \pm 1$.

Support vectors fully define the maximum-margin hyperplane or SVM solution, which justifies the name of the algorithm. By definition, vectors not lying on the marginal hyperplanes do not affect the definition of these hyperplanes — in their absence, the solution to the SVM problem remains unchanged. Note that while the solution $\mathbf{w}$ of the SVM problem is unique, the support vectors are not. In dimension N, $N + 1$ points are sufficient to define a hyperplane. Thus, when more than $N + 1$ points lie on a marginal hyperplane, different choices are possible for the $N + 1$ support vectors.

5.2.3 Dual optimization problem

To derive the dual form of the constrained optimization problem (5.7), we plug into the Lagrangian the definition of $\mathbf{w}$ in terms of the dual variables as expressed in

(5.9) and apply the constraint (5.10). This yields

$$\mathcal{L} = \underbrace{\frac{1}{2}\|\sum_{i=1}^m \alpha_i y_i \mathbf{x}_i\|^2 - \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)}_{-\frac{1}{2}\sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)} - \underbrace{\sum_{i=1}^m \alpha_i y_i b}_{0} + \sum_{i=1}^m \alpha_i\,, \tag{5.12}$$

which simplifies to

$$\mathcal{L} = \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)\,. \tag{5.13}$$

This leads to the following dual optimization problem for SVMs in the separable case:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{5.14}$$

$$\text{subject to: } \alpha_i \geq 0 \wedge \sum_{i=1}^m \alpha_i y_i = 0,\ \forall i \in [m]\,.$$

The objective function $G\colon \boldsymbol{\alpha} \mapsto \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$ is infinitely differentiable. Its Hessian is given by $\nabla^2 G = -\mathbf{A}$, with $\mathbf{A} = \big(y_i \mathbf{x}_i \cdot y_j \mathbf{x}_j\big)_{ij}$. $\mathbf{A}$ is the Gram matrix associated to the vectors $y_1\mathbf{x}_1, \ldots, y_m\mathbf{x}_m$ and is therefore positive semidefinite (see section A.2.3), which shows that $\nabla^2 G \preceq \mathbf{0}$ and that G is a concave function. Since the constraints are affine and convex, the maximization problem (5.14) is a convex optimization problem. Since G is a quadratic function of $\boldsymbol{\alpha}$, this dual optimization problem is also a QP problem, as in the case of the primal optimization and once again both general-purpose and specialized QP solvers can be used to obtain the solution (see exercise 5.4 for details on the SMO algorithm, which is often used to solve the dual form of the SVM problem in the more general non-separable setting).

Moreover, since the constraints are affine, they are qualified and strong duality holds (see appendix B). Thus, the primal and dual problems are equivalent, i.e., the solution $\boldsymbol{\alpha}$ of the dual problem (5.14) can be used directly to determine the hypothesis returned by SVMs, using equation (5.9):

$$h(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x} + b) = \operatorname{sgn}\Big(\sum_{i=1}^m \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b\Big). \tag{5.15}$$

Since support vectors lie on the marginal hyperplanes, for any support vector $\mathbf{x}_i$, $\mathbf{w} \cdot \mathbf{x}_i + b = y_i$, and thus b can be obtained via

$$b = y_i - \sum_{j=1}^{m} \alpha_j y_j (\mathbf{x}_j \cdot \mathbf{x}_i)\,. \tag{5.16}$$

The dual optimization problem (5.14) and the expressions (5.15) and (5.16) reveal an important property of SVMs: the hypothesis solution depends only on inner products between vectors and not directly on the vectors themselves. This observation is key and its importance will become clear in Chapter 6 where we introduce kernel methods.

Equation (5.16) can now be used to derive a simple expression of the geometric margin ρ in terms of $\boldsymbol{\alpha}$. Since (5.16) holds for all i with $\alpha_i \neq 0$, multiplying both sides by $\alpha_i y_i$ and taking the sum leads to

$$\sum_{i=1}^{m} \alpha_i y_i b = \sum_{i=1}^{m} \alpha_i y_i^2 - \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)\,. \tag{5.17}$$

Using the fact that $y_i^2 = 1$ along with equation (5.9) then yields

$$0 = \sum_{i=1}^{m} \alpha_i - \|\mathbf{w}\|^2. \tag{5.18}$$

Noting that $\alpha_i \geq 0$, we obtain the following expression of the margin ρ in terms of the L_1 norm of $\boldsymbol{\alpha}$:

$$\rho^2 = \frac{1}{\|\mathbf{w}\|_2^2} = \frac{1}{\sum_{i=1}^{m} \alpha_i} = \frac{1}{\|\boldsymbol{\alpha}\|_1}. \tag{5.19}$$

5.2.4 Leave-one-out analysis

We now use the notion of *leave-one-out error* to derive a first learning guarantee for SVMs based on the fraction of support vectors in the training set.

Definition 5.2 (Leave-one-out error) *Let h_S denote the hypothesis returned by a learning algorithm $\mathcal{A}$, when trained on a fixed sample S. Then, the* leave-one-out error *of $\mathcal{A}$ on a sample S of size m is defined by*

$$\widehat{R}_{\text{LOO}}(\mathcal{A}) = \frac{1}{m} \sum_{i=1}^{m} 1_{h_{S-\{x_i\}}(x_i) \neq y_i}\,.$$

Thus, for each $i \in [m]$, $\mathcal{A}$ is trained on all the points in S except for x_i, i.e., $S - \{x_i\}$, and its error is then computed using x_i. The leave-one-out error is the average of these errors. We will use an important property of the leave-one-out error stated in the following lemma.

Lemma 5.3 *The average leave-one-out error for samples of size $m \geq 2$ is an unbiased estimate of the average generalization error for samples of size $m-1$:*

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[\widehat{R}_{\mathrm{LOO}}(\mathcal{A})] = \mathop{\mathbb{E}}_{S' \sim \mathcal{D}^{m-1}}[R(h_{S'})], \tag{5.20}$$

where $\mathcal{D}$ denotes the distribution according to which points are drawn.

Proof: By the linearity of expectation, we can write

$$\begin{aligned}
\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[\widehat{R}_{\mathrm{LOO}}(\mathcal{A})] &= \frac{1}{m}\sum_{i=1}^{m} \mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[1_{h_{S-\{x_i\}}(x_i) \neq y_i}] \\
&= \mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[1_{h_{S-\{x_1\}}(x_1) \neq y_1}] \\
&= \mathop{\mathbb{E}}_{S' \sim \mathcal{D}^{m-1}, x_1 \sim \mathcal{D}}[1_{h_{S'}(x_1) \neq y_1}] \\
&= \mathop{\mathbb{E}}_{S' \sim \mathcal{D}^{m-1}}[\mathop{\mathbb{E}}_{x_1 \sim \mathcal{D}}[1_{h_{S'}(x_1) \neq y_1}]] \\
&= \mathop{\mathbb{E}}_{S' \sim \mathcal{D}^{m-1}}[R(h_{S'})].
\end{aligned}$$

For the second equality, we used the fact that, since the points of S are drawn in an i.i.d. fashion, the expectation $\mathbb{E}_{S \sim \mathcal{D}^m}[1_{h_{S-\{x_i\}}(x_i) \neq y_i}]$ does not depend on the choice of $i \in [m]$ and is thus equal to $\mathbb{E}_{S \sim \mathcal{D}^m}[1_{h_{S-\{x_1\}}(x_1) \neq y_1}]$. □

In general, computing the leave-one-out error may be costly since it requires training m times on samples of size $m-1$. In some situations however, it is possible to derive the expression of $\widehat{R}_{\mathrm{LOO}}(\mathcal{A})$ much more efficiently (see exercise 11.9).

Theorem 5.4 *Let h_S be the hypothesis returned by SVMs for a sample S, and let $N_{SV}(S)$ be the number of support vectors that define h_S. Then,*

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[R(h_S)] \leq \mathop{\mathbb{E}}_{S \sim \mathcal{D}^{m+1}}\left[\frac{N_{\mathrm{SV}}(S)}{m+1}\right].$$

Proof: Let S be a linearly separable sample of $m+1$. If x is not a support vector for h_S, removing it does not change the SVM solution. Thus, $h_{S-\{x\}} = h_S$ and $h_{S-\{x\}}$ correctly classifies x. By contraposition, if $h_{S-\{x\}}$ misclassifies x, x must be a support vector, which implies

$$\widehat{R}_{\mathrm{LOO}}(\mathrm{SVM}) \leq \frac{N_{\mathrm{SV}}(S)}{m+1}. \tag{5.21}$$

Taking the expectation of both sides and using lemma 5.3 yields the result. □

Theorem 5.4 gives a sparsity argument in favor of SVMs: the average error of the algorithm is upper bounded by the average fraction of support vectors. One may hope that for many distributions seen in practice, a relatively small number of the training points will lie on the marginal hyperplanes. The solution will then be sparse in the sense that a small fraction of the dual variables α_i will be non-

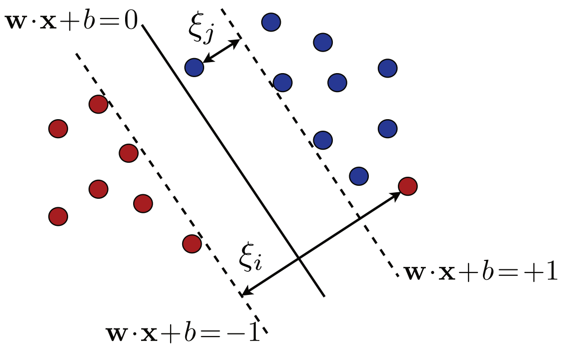

$\mathbf{w}\cdot\mathbf{x}+b=0$
ξ_j
ξ_i
$\mathbf{w}\cdot\mathbf{x}+b=+1$
$\mathbf{w}\cdot\mathbf{x}+b=-1$

Figure 5.4
A separating hyperplane with point $\mathbf{x}_i$ classified incorrectly and point $\mathbf{x}_j$ correctly classified, but with margin less than 1.

zero. Note, however, that this bound is relatively weak since it applies only to the average generalization error of the algorithm over all samples of size m. It provides no information about the variance of the generalization error. In section 5.4, we present stronger high-probability bounds using a different argument based on the notion of margin.

5.3 Non-separable case

In most practical settings, the training data is not linearly separable, which implies that for any hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$, there exists $\mathbf{x}_i \in S$ such that

$$y_i \left[\mathbf{w} \cdot \mathbf{x}_i + b\right] \not\geq 1 . \tag{5.22}$$

Thus, the constraints imposed in the linearly separable case discussed in section 5.2 cannot all hold simultaneously. However, a relaxed version of these constraints can indeed hold, that is, for each $i \in [m]$, there exist $\xi_i \geq 0$ such that

$$y_i \left[\mathbf{w} \cdot \mathbf{x}_i + b\right] \geq 1 - \xi_i . \tag{5.23}$$

The variables ξ_i are known as *slack variables* and are commonly used in optimization to define relaxed versions of constraints. Here, a slack variable ξ_i measures the distance by which vector $\mathbf{x}_i$ violates the desired inequality, $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$. Figure 5.4 illustrates the situation. For a hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$, a vector $\mathbf{x}_i$ with $\xi_i > 0$ can be viewed as an *outlier*. Each $\mathbf{x}_i$ must be positioned on the correct side of the appropriate marginal hyperplane to not be considered an outlier. As a consequence, a vector $\mathbf{x}_i$ with $0 < y_i(\mathbf{w} \cdot \mathbf{x}_i + b) < 1$ is correctly classified by the hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ but is nonetheless considered to be an outlier, that is, $\xi_i > 0$. If we omit the outliers, the training data is correctly separated by $\mathbf{w} \cdot \mathbf{x} + b = 0$ with a margin $\rho = 1/\|\mathbf{w}\|$ that we refer to as the *soft margin*, as opposed to the *hard margin* in the separable case.

How should we select the hyperplane in the non-separable case? One idea consists of selecting the hyperplane that minimizes the empirical error. But, that solution will not benefit from the large-margin guarantees we will present in section 5.4. Furthermore, the problem of determining a hyperplane with the smallest zero-one loss, that is the smallest number of misclassifications, is NP-hard as a function of the dimension N of the space.

Here, there are two conflicting objectives: on one hand, we wish to limit the total amount of slack due to outliers, which can be measured by $\sum_{i=1}^m \xi_i$, or, more generally by $\sum_{i=1}^m \xi_i^p$ for some $p \geq 1$; on the other hand, we seek a hyperplane with a large margin, though a larger margin can lead to more outliers and thus larger amounts of slack.

5.3.1 Primal optimization problem

This leads to the following general optimization problem defining SVMs in the non-separable case where the parameter $C \geq 0$ determines the trade-off between margin-maximization (or minimization of $\|\mathbf{w}\|^2$) and the minimization of the slack penalty $\sum_{i=1}^m \xi_i^p$:

$$\min_{\mathbf{w}, b, \boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i^p \tag{5.24}$$
$$\text{subject to } \ y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \ \wedge \ \xi_i \geq 0, i \in [m],$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m)^\top$. The parameter C is typically determined via n-fold cross-validation (see section 4.5).

As in the separable case, (5.24) is a convex optimization problem since the constraints are affine and thus convex and since the objective function is convex for any $p \geq 1$. In particular, $\boldsymbol{\xi} \mapsto \sum_{i=1}^m \xi_i^p = \|\boldsymbol{\xi}\|_p^p$ is convex in view of the convexity of the norm $\|\cdot\|_p$.

There are many possible choices for p leading to more or less aggressive penalizations of the slack terms (see exercise 5.1). The choices $p = 1$ and $p = 2$ lead to the most straightforward solutions and analyses. The loss functions associated with $p = 1$ and $p = 2$ are called the *hinge loss* and the *quadratic hinge loss*, respectively. Figure 5.5 shows the plots of these loss functions as well as that of the standard zero-one loss function. Both hinge losses are convex upper bounds on the zero-one loss, thus making them well suited for optimization. In what follows, the analysis is presented in the case of the hinge loss ($p = 1$), which is the most widely used loss function for SVMs.

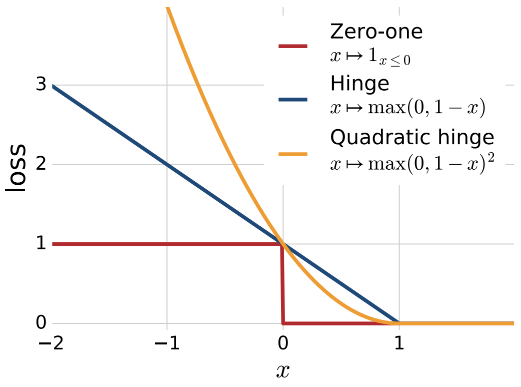


Figure 5.5
Both the hinge loss and the quadratic hinge loss provide convex upper bounds on the binary zero-one loss.

5.3.2 Support vectors

As in the separable case, the constraints are affine and thus qualified. The objective function as well as the affine constraints are convex and differentiable. Thus, the hypotheses of theorem B.30 hold and the KKT conditions apply at the optimum. We use these conditions to both analyze the algorithm and demonstrate several of its crucial properties, and subsequently derive the dual optimization problem associated to SVMs in section 5.3.3.

We introduce Lagrange variables $\alpha_i \geq 0$, $i \in [m]$, associated to the first m constraints and $\beta_i \geq 0$, $i \in [m]$ associated to the non-negativity constraints of the slack variables. We denote by $\boldsymbol{\alpha}$ the vector $(\alpha_1, \ldots, \alpha_m)^\top$ and by $\boldsymbol{\beta}$ the vector $(\beta_1, \ldots, \beta_m)^\top$. The Lagrangian can then be defined for all $\mathbf{w} \in \mathbb{R}^N$, $b \in \mathbb{R}$, and $\boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^m_+$, by

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^m \xi_i - \sum_{i=1}^m \alpha_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^m \beta_i \xi_i \,. \quad (5.25)$$

The KKT conditions are obtained by setting the gradient of the Lagrangian with respect to the primal variables $\mathbf{w}$, b, and ξ_is to zero and by writing the complementarity conditions:

$$\nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = 0 \quad \Longrightarrow \quad \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \tag{5.26}$$

$$\nabla_b \mathcal{L} = -\sum_{i=1}^{m} \alpha_i y_i = 0 \quad \Longrightarrow \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{5.27}$$

$$\nabla_{\xi_i}\mathcal{L} = C - \alpha_i - \beta_i = 0 \quad \Longrightarrow \quad \alpha_i + \beta_i = C \tag{5.28}$$

$$\forall i, \alpha_i[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] = 0 \quad \Longrightarrow \quad \alpha_i = 0 \vee y_i(\mathbf{w}\cdot\mathbf{x}_i + b) = 1 - \xi_i \tag{5.29}$$

$$\forall i, \beta_i \xi_i = 0 \quad \Longrightarrow \quad \beta_i = 0 \vee \xi_i = 0\,. \tag{5.30}$$

By equation (5.26), as in the separable case, the weight vector $\mathbf{w}$ at the solution of the SVM problem is a linear combination of the training set vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$. A vector $\mathbf{x}_i$ appears in that expansion iff $\alpha_i \neq 0$. Such vectors are called *support vectors*. Here, there are two types of support vectors. By the complementarity condition (5.29), if $\alpha_i \neq 0$, then $y_i(\mathbf{w}\cdot\mathbf{x}_i+b) = 1-\xi_i$. If $\xi_i = 0$, then $y_i(\mathbf{w}\cdot\mathbf{x}_i+b) = 1$ and $\mathbf{x}_i$ lies on a marginal hyperplane, as in the separable case. Otherwise, $\xi_i \neq 0$ and $\mathbf{x}_i$ is an outlier. In this case, (5.30) implies $\beta_i = 0$ and (5.28) then requires $\alpha_i = C$. Thus, support vectors $\mathbf{x}_i$ are either outliers, in which case $\alpha_i = C$, or vectors lying on the marginal hyperplanes. As in the separable case, note that while the weight vector $\mathbf{w}$ solution is unique, the support vectors are not.

5.3.3 Dual optimization problem

To derive the dual form of the constrained optimization problem (5.24), we plug into the Lagrangian the definition of $\mathbf{w}$ in terms of the dual variables (5.26) and apply the constraint (5.27). This yields

$$\mathcal{L} = \underbrace{\frac{1}{2}\|\sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i\|^2 - \sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j)}_{-\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j)} - \underbrace{\sum_{i=1}^{m}\alpha_i y_i b}_{0} + \sum_{i=1}^{m}\alpha_i\,. \tag{5.31}$$

Remarkably, we find that the objective function is no different than in the separable case:

$$\mathcal{L} = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j)\,. \tag{5.32}$$

However, here, in addition to $\alpha_i \geq 0$, we must impose the constraint on the Lagrange variables $\beta_i \geq 0$. In view of (5.28), this is equivalent to $\alpha_i \leq C$. This leads to the following dual optimization problem for SVMs in the non-separable case, which only differs from that of the separable case (5.14) by the constraints $\alpha_i \leq C$:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{5.33}$$
$$\text{subject to: } 0 \leq \alpha_i \leq C \wedge \sum_{i=1}^{m} \alpha_i y_i = 0, i \in [m].$$

Thus, our previous comments about the optimization problem (5.14) apply to (5.33) as well. In particular, the objective function is concave and infinitely differentiable and (5.33) is equivalent to a convex QP. The problem is equivalent to the primal problem (5.24).

The solution $\boldsymbol{\alpha}$ of the dual problem (5.33) can be used directly to determine the hypothesis returned by SVMs, using equation (5.26):

$$h(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x} + b) = \operatorname{sgn}\Big(\sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b\Big). \tag{5.34}$$

Moreover, b can be obtained from any support vector $\mathbf{x}_i$ lying on a marginal hyperplane, that is any vector $\mathbf{x}_i$ with $0 < \alpha_i < C$. For such support vectors, $\mathbf{w} \cdot \mathbf{x}_i + b = y_i$ and thus

$$b = y_i - \sum_{j=1}^{m} \alpha_j y_j (\mathbf{x}_j \cdot \mathbf{x}_i). \tag{5.35}$$

As in the separable case, the dual optimization problem (5.33) and the expressions (5.34) and (5.35) show an important property of SVMs: the hypothesis solution depends only on inner products between vectors and not directly on the vectors themselves. This fact can be used to extend SVMs to define non-linear decision boundaries, as we shall see in chapter 6.

5.4 Margin theory

This section presents generalization bounds which provide a strong theoretical justification for the SVM algorithm.

Recall that the VC-dimension of the family of hyperplanes or linear hypotheses in $\mathbb{R}^N$ is $N+1$. Thus, the application of the VC-dimension bound (3.29) of corollary 3.19 to this hypothesis set yields the following: for any $\delta > 0$, with probability at least $1-\delta$, for any $h \in \mathcal{H}$,

$$R(h) \leq \widehat{R}_S(h) + \sqrt{\frac{2(N+1)\log \frac{em}{N+1}}{m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}. \tag{5.36}$$

When the dimension of the feature space N is large compared to the sample size m, this bound is uninformative. Remarkably, the learning guarantees presented in this section are independent of the dimension N and thus hold regardless of its value.

The guarantees we will present hold for real-valued functions such as the function $\mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} + b$ returned by SVMs, as opposed to classification functions returning $+1$ or -1, such as $\mathbf{x} \mapsto \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$. They are based on the notion of *confidence margin*. The confidence margin of a real-valued function h at a point x labeled with y is the quantity $yh(x)$. Thus, when $yh(x) > 0$, h classifies x correctly but we interpret the magnitude of $|h(x)|$ as the *confidence* of the prediction made by h. The notion of confidence margin is distinct from that of geometric margin and does not require a linear separability assumption. But, the two notions are related as follows in the separable case: for $h \colon \mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} + b$ with geometric margin ρ_{geom}, the confidence margin at any point $\mathbf{x}$ of the training sample with label y is at least $\rho_{\text{geom}} \|\mathbf{w}\|$, i.e. $|yh(\mathbf{x})| \geq \rho_{\text{geom}} \|\mathbf{w}\|$.

In view of the definition of the confidence margin, for any parameter $\rho > 0$, we will define a *ρ-margin loss function* that, as with the zero-one loss, penalizes h with the cost of 1 when it misclassifies point x ($yh(x) \leq 0$), but also penalizes h (linearly) when it correctly classifies x with confidence less than or equal to ρ ($yh(x) \leq \rho$). The main margin-based generalization bounds of this section are presented in terms of this loss function, which is formally defined as follows.

Definition 5.5 (Margin loss function) *For any $\rho > 0$, the ρ-margin loss is the function $L_\rho \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ defined for all $y, y' \in \mathbb{R}$ by $L_\rho(y, y') = \Phi_\rho(yy')$ with,*

$$\Phi_\rho(x) = \min\left(1, \max\left(0, 1 - \frac{x}{\rho}\right)\right) = \begin{cases} 1 & \text{if } x \leq 0 \\ 1 - \frac{x}{\rho} & \text{if } 0 \leq x \leq \rho \\ 0 & \text{if } \rho \leq x. \end{cases}$$

This loss function is illustrated by figure 5.6. The parameter $\rho > 0$ can be interpreted as the confidence margin demanded from a hypothesis h. The empirical margin loss is similarly defined as the margin loss over the training sample.

Definition 5.6 (Empirical margin loss) *Given a sample $S = (x_1, \ldots, x_m)$ and a hypothesis h, the empirical margin loss is defined by*

$$\widehat{R}_{S,\rho}(h) = \frac{1}{m} \sum_{i=1}^{m} \Phi_\rho(y_i h(x_i)) . \tag{5.37}$$

Note that, for any $i \in [m]$, $\Phi_\rho(y_i h(x_i)) \leq 1_{y_i h(x_i) \leq \rho}$. Thus, the empirical margin loss can be upper-bounded as follows:

$$\widehat{R}_{S,\rho}(h) \leq \frac{1}{m} \sum_{i=1}^{m} 1_{y_i h(x_i) \leq \rho} . \tag{5.38}$$

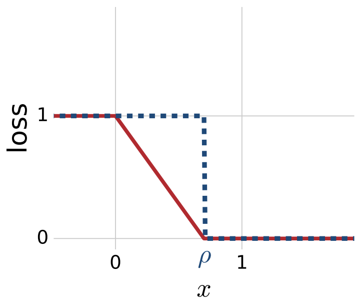


Figure 5.6
The margin loss illustrated in red, defined with respect to margin parameter $\rho = 0.7$.

In all the results that follow, the empirical margin loss can be replaced by this upper bound, which admits a simple interpretation: it is the fraction of the points in the training sample S that have been misclassified or classified with confidence less than ρ. In other words, the upper bound is then the fraction of the points in the training data with margin less than ρ. This corresponds to the loss function indicated by the blue dotted line in figure 5.6.

A key benefit of using a loss function based on Φ_ρ as opposed to the zero-one loss or the loss defined by the blue dotted line of figure 5.6 is that Φ_ρ is $1/\rho$-Lipschitz, since the absolute value of the slope of the function is at most $1/\rho$. The following lemma bounds the empirical Rademacher complexity of a hypothesis set $\mathcal{H}$ after composition with such a Lipschitz function in terms of the empirical Rademacher complexity of $\mathcal{H}$. It will be needed for the proof of the margin-based generalization bound.

Lemma 5.7 (Talagrand's lemma) *Let $\Phi_1, \ldots, \Phi_m$ be l-Lipschitz functions from $\mathbb{R}$ to $\mathbb{R}$ and $\sigma_1, \ldots, \sigma_m$ be Rademacher random variables. Then, for any hypothesis set $\mathcal{H}$ of real-valued functions, the following inequality holds:*

$$\frac{1}{m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\sigma_i(\Phi_i\circ h)(x_i))\Big] \leq \frac{l}{m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\sigma_i h(x_i)\Big] = l\,\widehat{\mathfrak{R}}_S(\mathcal{H})\,.$$

In particular, if $\Phi_i = \Phi$ for all $i \in [m]$, then the following holds:

$$\widehat{\mathfrak{R}}_S(\Phi\circ\mathcal{H}) \leq l\,\widehat{\mathfrak{R}}_S(\mathcal{H})\,.$$

Proof: First we fix a sample $S = (x_1, \ldots, x_m)$, then, by definition,

$$\frac{1}{m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\sigma_i(\Phi_m\circ h)(x_i)\Big] = \frac{1}{m}\mathop{\mathbb{E}}_{\sigma_1,\ldots,\sigma_{m-1}}\Big[\mathop{\mathbb{E}}_{\sigma_m}\Big[\sup_{h\in\mathcal{H}} u_{m-1}(h) + \sigma_m(\Phi_m\circ h)(x_m)\Big]\Big],$$

where $u_{m-1}(h) = \sum_{i=1}^{m-1} \sigma_i(\Phi_i \circ h)(x_i)$. By definition of the supremum, for any $\epsilon > 0$, there exist $h_1, h_2 \in \mathcal{H}$ such that

$$u_{m-1}(h_1) + (\Phi_m \circ h_1)(x_m) \geq (1-\epsilon)\Big[\sup_{h\in\mathcal{H}} u_{m-1}(h) + (\Phi_m \circ h)(x_m)\Big]$$
$$\text{and} \quad u_{m-1}(h_2) - (\Phi_m \circ h_2)(x_m) \geq (1-\epsilon)\Big[\sup_{h\in\mathcal{H}} u_{m-1}(h) - (\Phi_m \circ h)(x_m)\Big].$$

Thus, for any $\epsilon > 0$, by definition of $\mathbb{E}_{\sigma_m}$,

$$\begin{aligned}
&(1-\epsilon) \mathop{\mathbb{E}}_{\sigma_m}\Big[\sup_{h\in\mathcal{H}} u_{m-1}(h) + \sigma_m(\Phi_m \circ h)(x_m)\Big] \\
&= (1-\epsilon)\Big[\frac{1}{2}\sup_{h\in\mathcal{H}}\Big[u_{m-1}(h) + (\Phi_m \circ h)(x_m)\Big] + \frac{1}{2}\Big[\sup_{h\in\mathcal{H}} u_{m-1}(h) - (\Phi_m \circ h)(x_m)\Big]\Big] \\
&\leq \frac{1}{2}[u_{m-1}(h_1) + (\Phi_m \circ h_1)(x_m)] + \frac{1}{2}[u_{m-1}(h_2) - (\Phi_m \circ h_2)(x_m)].
\end{aligned}$$

Let $s = \text{sgn}(h_1(x_m) - h_2(x_m))$. Then, the previous inequality implies

$$\begin{aligned}
&(1-\epsilon) \mathop{\mathbb{E}}_{\sigma_m}\Big[\sup_{h\in\mathcal{H}} u_{m-1}(h) + \sigma_m(\Phi_m \circ h)(x_m)\Big] \\
&\leq \frac{1}{2}[u_{m-1}(h_1) + u_{m-1}(h_2) + sl(h_1(x_m) - h_2(x_m))] && \text{(Lipschitz property)} \\
&= \frac{1}{2}[u_{m-1}(h_1) + slh_1(x_m)] + \frac{1}{2}[u_{m-1}(h_2) - slh_2(x_m)] && \text{(rearranging)} \\
&\leq \frac{1}{2}\sup_{h\in\mathcal{H}}[u_{m-1}(h) + slh(x_m)] + \frac{1}{2}\sup_{h\in\mathcal{H}}[u_{m-1}(h) - slh(x_m)] && \text{(definition of sup)} \\
&= \mathop{\mathbb{E}}_{\sigma_m}\Big[\sup_{h\in\mathcal{H}} u_{m-1}(h) + \sigma_m l h(x_m)\Big]. && \text{(definition of } \mathop{\mathbb{E}}_{\sigma_m}\text{)}
\end{aligned}$$

Since the inequality holds for all $\epsilon > 0$, we have

$$\mathop{\mathbb{E}}_{\sigma_m}\Big[\sup_{h\in\mathcal{H}} u_{m-1}(h) + \sigma_m(\Phi_m \circ h)(x_m)\Big] \leq \mathop{\mathbb{E}}_{\sigma_m}\Big[\sup_{h\in\mathcal{H}} u_{m-1}(h) + \sigma_m l h(x_m)\Big].$$

Proceeding in the same way for all other σ_i $(i \neq m)$ proves the lemma. □

The following is a general margin-based generalization bound that will be used in the analysis of several algorithms.

Theorem 5.8 (Margin bound for binary classification) *Let $\mathcal{H}$ be a set of real-valued functions. Fix $\rho > 0$, then, for any $\delta > 0$, with probability at least $1-\delta$, each of the following holds for all $h \in \mathcal{H}$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}} \tag{5.39}$$

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\widehat{\mathfrak{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}}\,. \tag{5.40}$$

Proof: Let $\widetilde{\mathcal{H}} = \{z = (x, y) \mapsto yh(x) \colon h \in \mathcal{H}\}$. Consider the family of functions taking values in $[0, 1]$:

$$\widetilde{\mathcal{H}} = \{\Phi_\rho \circ f \colon f \in \widetilde{\mathcal{H}}\} .$$

By theorem 3.3, with probability at least $1 - \delta$, for all $g \in \widetilde{\mathcal{H}}$,

$$\mathbb{E}[g(z)] \leq \frac{1}{m} \sum_{i=1}^{m} g(z_i) + 2\mathfrak{R}_m(\widetilde{\mathcal{H}}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} ,$$

and thus, for all $h \in \mathcal{H}$,

$$\mathbb{E}[\Phi_\rho(yh(x))] \leq \widehat{R}_{S,\rho}(h) + 2\mathfrak{R}_m\big(\Phi_\rho \circ \widetilde{\mathcal{H}}\big) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} .$$

Since $1_{u \leq 0} \leq \Phi_\rho(u)$ for all $u \in \mathbb{R}$, we have $R(h) = \mathbb{E}[1_{yh(x) \leq 0}] \leq \mathbb{E}[\Phi_\rho(yh(x))]$, thus

$$R(h) \leq \widehat{R}_{S,\rho}(h) + 2\mathfrak{R}_m\big(\Phi_\rho \circ \widetilde{\mathcal{H}}\big) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} .$$

Since Φ_ρ is $1/\rho$-Lipschitz, by lemma 5.7, we have $\mathfrak{R}_m\big(\Phi_\rho \circ \widetilde{\mathcal{H}}\big) \leq \frac{1}{\rho}\mathfrak{R}_m(\widetilde{\mathcal{H}})$ and $\mathfrak{R}_m(\widetilde{\mathcal{H}})$ can be rewritten as follows:

$$\mathfrak{R}_m(\widetilde{\mathcal{H}}) = \frac{1}{m} \mathop{\mathbb{E}}_{S,\sigma} \Big[\sup_{h \in \mathcal{H}} \sum_{i=1}^{m} \sigma_i y_i h(x_i) \Big] = \frac{1}{m} \mathop{\mathbb{E}}_{S,\sigma} \Big[\sup_{h \in \mathcal{H}} \sum_{i=1}^{m} \sigma_i h(x_i) \Big] = \mathfrak{R}_m(\mathcal{H}) .$$

This proves (5.39). The second inequality, (5.40), can be derived in the same way by using the second inequality of theorem 3.3, (3.4), instead of (3.3). □

The generalization bounds of theorem 5.8 suggest a trade-off: a larger value of ρ decreases the complexity term (second term), but tends to increase the empirical margin-loss $\widehat{R}_{S,\rho}(h)$ (first term) by requiring from a hypothesis h a higher confidence margin. Thus, if for a relatively large value of ρ the empirical margin loss of h remains relatively small, then h benefits from a very favorable guarantee on its generalization error. For theorem 5.8, the margin parameter ρ must be selected beforehand. But, the bounds of the theorem can be generalized to hold uniformly for all $\rho \in (0, 1]$ at the cost of a modest additional term $\sqrt{\frac{\log \log_2 \frac{2}{\rho}}{m}}$, as shown in the following theorem (a version of this theorem with better constants can be derived, see exercise 5.2).

Theorem 5.9 *Let $\mathcal{H}$ be a set of real-valued functions. Fix $r > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, each of the following holds for all $h \in \mathcal{H}$ and $\rho \in (0, r]$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{4}{\rho}\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log\log_2 \frac{2r}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}} \tag{5.41}$$

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{4}{\rho}\widehat{\mathfrak{R}}_S(\mathcal{H}) + \sqrt{\frac{\log\log_2 \frac{2r}{\rho}}{m}} + 3\sqrt{\frac{\log \frac{4}{\delta}}{2m}}\,. \tag{5.42}$$

Proof: Consider two sequences $(\rho_k)_{k\geq 1}$ and $(\epsilon_k)_{k\geq 1}$, with $\epsilon_k \in (0, 1]$. By theorem 5.8, for any fixed $k \geq 1$,

$$\mathbb{P}\left[\sup_{h\in H} R(h) - \widehat{R}_{S,\rho_k}(h) > \frac{2}{\rho_k}\mathfrak{R}_m(\mathcal{H}) + \epsilon_k\right] \leq \exp(-2m\epsilon_k^2). \tag{5.43}$$

Choose $\epsilon_k = \epsilon + \sqrt{\frac{\log k}{m}}$, then, by the union bound, the following holds:

$$\begin{aligned}
\mathbb{P}\left[\sup_{\substack{h\in H\\ k\geq 1}} R(h) - \widehat{R}_{S,\rho_k}(h) - \frac{2}{\rho_k}\mathfrak{R}_m(\mathcal{H}) - \epsilon_k > 0\right]& \\
&\leq \sum_{k\geq 1} \exp(-2m\epsilon_k^2)\\
&= \sum_{k\geq 1} \exp\big[-2m(\epsilon + \sqrt{(\log k)/m})^2\big]\\
&\leq \sum_{k\geq 1} \exp(-2m\epsilon^2)\exp(-2\log k)\\
&= \big(\sum_{k\geq 1} 1/k^2\big)\exp(-2m\epsilon^2)\\
&= \frac{\pi^2}{6}\exp(-2m\epsilon^2) \leq 2\exp(-2m\epsilon^2).
\end{aligned}$$

We can choose $\rho_k = r/2^k$. For any $\rho \in (0, r]$, there exists $k \geq 1$ such that $\rho \in (\rho_k, \rho_{k-1}]$, with $\rho_0 = r$. For that k, $\rho \leq \rho_{k-1} = 2\rho_k$, thus $1/\rho_k \leq 2/\rho$ and $\sqrt{\log k} = \sqrt{\log\log_2(r/\rho_k)} \leq \sqrt{\log\log_2(2r/\rho)}$. Furthermore, for any $h \in \mathcal{H}$, $\widehat{R}_{S,\rho_k}(h) \leq \widehat{R}_{S,\rho}(h)$. Thus, the following inequality holds:

$$\mathbb{P}\left[\sup_{\substack{h\in H\\ \rho\in(0,r]}} R(h) - \widehat{R}_{S,\rho}(h) - \frac{4}{\rho}\mathfrak{R}_m(\mathcal{H}) - \sqrt{\frac{\log\log_2(2r/\rho)}{m}} - \epsilon > 0\right] \leq 2\exp(-2m\epsilon^2),$$

which proves the first statement. The second statement can be proven in a similar way. □

The Rademacher complexity of linear hypotheses with bounded weight vector can be bounded as follows.

Theorem 5.10 *Let $S \subseteq \{\mathbf{x} \colon \|\mathbf{x}\| \leq r\}$ be a sample of size m and let $\mathcal{H} = \{\mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} \colon \|\mathbf{w}\| \leq \Lambda\}$. Then, the empirical Rademacher complexity of $\mathcal{H}$ can be bounded as follows:*

$$\widehat{\mathfrak{R}}_S(\mathcal{H}) \leq \sqrt{\frac{r^2\Lambda^2}{m}} .$$

Proof: The proof follows through a series of inequalities:

$$\begin{aligned}
\widehat{\mathfrak{R}}_S(\mathcal{H}) &= \frac{1}{m} \mathop{\mathbb{E}}_{\sigma}\Bigg[\sup_{\|\mathbf{w}\|\leq\Lambda} \sum_{i=1}^m \sigma_i \mathbf{w} \cdot \mathbf{x}_i\Bigg] = \frac{1}{m} \mathop{\mathbb{E}}_{\sigma}\Bigg[\sup_{\|\mathbf{w}\|\leq\Lambda} \mathbf{w} \cdot \sum_{i=1}^m \sigma_i \mathbf{x}_i\Bigg] \\
&\leq \frac{\Lambda}{m} \mathop{\mathbb{E}}_{\sigma}\Bigg[\Big\|\sum_{i=1}^m \sigma_i \mathbf{x}_i\Big\|\Bigg] \leq \frac{\Lambda}{m} \Bigg[\mathop{\mathbb{E}}_{\sigma}\Bigg[\Big\|\sum_{i=1}^m \sigma_i \mathbf{x}_i\Big\|^2\Bigg]\Bigg]^{\frac{1}{2}} \\
&= \frac{\Lambda}{m} \Bigg[\mathop{\mathbb{E}}_{\sigma}\Bigg[\sum_{i,j=1}^m \sigma_i \sigma_j (\mathbf{x}_i \cdot \mathbf{x}_j)\Bigg]\Bigg]^{\frac{1}{2}} \leq \frac{\Lambda}{m} \Bigg[\sum_{i=1}^m \|\mathbf{x}_i\|^2\Bigg]^{\frac{1}{2}} \leq \frac{\Lambda\sqrt{mr^2}}{m} = \sqrt{\frac{r^2\Lambda^2}{m}} ,
\end{aligned}$$

The first inequality makes use of the Cauchy-Schwarz inequality and the bound on $\|\mathbf{w}\|$, the second follows by Jensen's inequality, the third by $\mathbb{E}[\sigma_i\sigma_j] = \mathbb{E}[\sigma_i]\,\mathbb{E}[\sigma_j] = 0$ for $i \neq j$, and the last one by $\|\mathbf{x}_i\| \leq r$. □

Combining theorem 5.10 and theorem 5.8 gives directly the following general margin bound for linear hypotheses with bounded weight vectors, presented in corollary 5.11.

Corollary 5.11 *Let $\mathcal{H} = \{\mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} \colon \|\mathbf{w}\| \leq \Lambda\}$ and assume that $\mathcal{X} \subseteq \{\mathbf{x} \colon \|\mathbf{x}\| \leq r\}$. Fix $\rho > 0$, then, for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of a sample S of size m, the following holds for any $h \in \mathcal{H}$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + 2\sqrt{\frac{r^2\Lambda^2/\rho^2}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}} . \tag{5.44}$$

As with theorem 5.8, the bound of this corollary can be generalized to hold uniformly for all $\rho \in (0, 1]$ at the cost of an additional term $\sqrt{\frac{\log\log_2\frac{2}{\rho}}{m}}$ by combining theorems 5.10 and 5.9.

This generalization bound for linear hypotheses is remarkable, since it does not depend directly on the dimension of the feature space, but only on the margin. It suggests that a small generalization error can be achieved when $\rho/(r\Lambda)$ is large (small second term) while the empirical margin loss is relatively small (first term). The latter occurs when few points are either classified incorrectly or correctly, but with margin less than ρ. When the training sample is linearly separable, for a linear hypothesis with geometric margin ρ_{geom} and the choice of the confidence margin parameter $\rho = \rho_{\text{geom}}$, the empirical margin loss term is zero. Thus, if ρ_{geom} is relatively large, this provides a strong guarantee for the generalization error of the corresponding linear hypothesis.

The fact that the guarantee does not explicitly depend on the dimension of the feature space may seem surprising and appears to contradict the VC-dimension lower bounds of theorems 3.20 and 3.23. Those lower bounds show that for any learning algorithm $\mathcal{A}$ there exists a *bad* distribution for which the error of the hypothesis returned by the algorithm is $\Omega(\sqrt{d/m})$ with a non-zero probability. The bound of the corollary does not rule out such *bad* cases, however: for such bad distributions, the empirical margin loss would be large even for a relatively small margin ρ, and thus the bound of the corollary would be loose in that case.

Thus, in some sense, the learning guarantee of the corollary hinges upon the hope of a good margin value ρ: if there exists a relatively large margin value $\rho > 0$ for which the empirical margin loss is small, then a small generalization error is guaranteed by the corollary. This favorable margin situation depends on the distribution: while the learning bound is distribution-independent, the existence of a good margin is in fact distribution-dependent. A favorable margin seems to appear relatively often in applications.

The bound of the corollary gives a strong justification for margin-maximization algorithms such as SVMs. Choosing $\Lambda = 1$, by the generalization of corollary 5.11 to a uniform bound over $\rho \in (0, r]$, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for all $h \in \{\mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} \colon \|\mathbf{w}\| \leq 1\}$ and $\rho \in (0, r]$:

$$R(h) \leq \widehat{R}_{S,\rho}(h) + 4\sqrt{\frac{r^2/\rho^2}{m}} + \sqrt{\frac{\log \log_2 \frac{2r}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}.$$

The inequality also trivially holds for ρ larger than r since in that case, by the Cauchy-Schwarz inequality, for any $\mathbf{w}$ with $\|\mathbf{w}\| \leq 1$, we have $y_i(\mathbf{w} \cdot \mathbf{x}_i) \leq r \leq \rho$ and $\widehat{R}_{S,\rho}(h)$ is equal to one for all h.

Now, for any $\rho > 0$, the ρ-margin loss function is upper bounded by the ρ-hinge loss:

$$\forall u \in \mathbb{R}, \Phi_\rho(u) = \min\left(1, \max\left(0, 1 - \frac{u}{\rho}\right)\right) \leq \max\left(0, 1 - \frac{u}{\rho}\right). \tag{5.45}$$

Thus, with probability at least $1 - \delta$, the following holds for all $h \in \{\mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} \colon \|\mathbf{w}\| \leq 1\}$ and all $\rho > 0$:

$$R(h) \leq \frac{1}{m} \sum_{i=1}^{m} \max\left(0, 1 - \frac{y_i(\mathbf{w} \cdot \mathbf{x}_i)}{\rho}\right) + 4\sqrt{\frac{r^2/\rho^2}{m}} + \sqrt{\frac{\log \log_2 \frac{2r}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}.$$

Since for any $\rho > 0$, h/ρ admits the same generalization error as h, with probability at least $1-\delta$, the following holds for all $h \in \big\{\mathbf{x} \mapsto \mathbf{w}\cdot\mathbf{x} \colon \|\mathbf{w}\| \leq 1/\rho\big\}$ and all $\rho > 0$:

$$R(h) \leq \frac{1}{m}\sum_{i=1}^{m} \max\big(0, 1-y_i(\mathbf{w}\cdot\mathbf{x}_i)\big) + 4\sqrt{\frac{r^2/\rho^2}{m}} + \sqrt{\frac{\log\log_2 \frac{2r}{\rho}}{m}} + \sqrt{\frac{\log\frac{2}{\delta}}{2m}}\,. \quad (5.46)$$

This inequality can be used to derive an algorithm that selects $\mathbf{w}$ and $\rho > 0$ to minimize the right-hand side. The minimization with respect to ρ does not lead to a convex optimization and depends on theoretical constant factors affecting the second and third terms, which may not be optimal. Thus, instead, ρ is left as a free parameter of the algorithm, typically determined via cross-validation.

Now, since only the first term of the right-hand side depends on $\mathbf{w}$, for any $\rho > 0$, the bound suggests selecting $\mathbf{w}$ as the solution of the following optimization problem:

$$\min_{\|\mathbf{w}\|^2 \leq \frac{1}{\rho^2}} \frac{1}{m}\sum_{i=1}^{m} \max\big(0, 1-y_i(\mathbf{w}\cdot\mathbf{x}_i)\big)\,. \quad (5.47)$$

Introducing a Lagrange variable $\lambda \geq 0$, the optimization problem can be written equivalently as

$$\min_{\mathbf{w}} \lambda\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max\big(0, 1-y_i(\mathbf{w}\cdot\mathbf{x}_i)\big)\,. \quad (5.48)$$

Since for any choice of ρ in the constraint of (5.47) there exists an equivalent dual variable λ in the formulation of (5.48) that achieves the same optimal $\mathbf{w}$, λ can be freely selected via cross-validation.[5] The resulting algorithm precisely coincides with SVMs. Note that an alternative objective function and thus algorithm would be based on the empirical margin loss instead of the hinge loss. However, the advantage of the hinge loss is that it is convex, while the margin loss is not.

As already pointed out, the bounds just discussed do not directly depend on the dimension of the feature space but guarantee good generalization when given a favorable margin. Thus, they suggest seeking large-margin separating hyperplanes in a very high-dimensional space. In view of the form of the dual optimization problems for SVMs, determining the solution of the optimization and using it for prediction both require computing many inner products in that space. For very high-dimensional spaces, the computation of these inner products could become very costly. The next chapter provides a solution to this problem which further provides a generalization of SVMs to non-vectorial input spaces.

[5] An equivalent analysis consists of choosing $\rho = 1/\|\mathbf{w}\|$ in (5.46).

5.5 Chapter notes

The maximum-margin or *optimal hyperplane* solution described in section 5.2 was introduced by Vapnik and Chervonenkis [1964]. The algorithm had limited applications since in most tasks in practice the data is not linearly separable. In contrast, the SVM algorithm of section 5.3 for the general non-separable case, introduced by Cortes and Vapnik [1995] under the name *support-vector networks*, has been widely adopted and been shown to be effective in practice. The algorithm and its theory have had a profound impact on theoretical and applied machine learning and inspired research on a variety of topics. Several specialized algorithms have been suggested for solving the specific QP that arises when solving the SVM problem, for example the SMO algorithm of Platt [1999] (see exercise 5.4) and a variety of other decomposition methods such as those used in the LibLinear software library [Hsieh et al., 2008], and [Allauzen et al., 2010] for solving the problem when using rational kernels (see chapter 6).

Much of the theory supporting the SVM algorithm ([Cortes and Vapnik, 1995, Vapnik, 1998]), in particular the margin theory presented in section 5.4, has been adopted in the learning theory and statistics communities and applied to a variety of other problems. The margin bound on the VC-dimension of canonical hyperplanes (exercise 5.7) is by Vapnik [1998], the proof is very similar to Novikoff's margin bound on the number of updates made by the Perceptron algorithm in the separable case. Our presentation of margin guarantees based on the Rademacher complexity follows the elegant analysis of Koltchinskii and Panchenko [2002] (see also Bartlett and Mendelson [2002], Shawe-Taylor et al. [1998]). Our proof of Talagrand's lemma 5.7 is a simpler and more concise version of a more general result given by Ledoux and Talagrand [1991, pp. 112–114]. See Höffgen et al. [1995] for hardness results related to the problem of finding a hyperplane with the minimal number of errors on a training sample.

5.6 Exercises

5.1 Soft margin hyperplanes. The function of the slack variables used in the optimization problem for soft margin hyperplanes has the form: $\xi \mapsto \sum_{i=1}^{m} \xi_i$. Instead, we could use $\xi \mapsto \sum_{i=1}^{m} \xi_i^p$, with $p > 1$.

(a) Give the dual formulation of the problem in this general case.

(b) How does this more general formulation ($p > 1$) compare to the standard setting ($p = 1$)? In the case $p = 2$ is the optimization still convex?

5.2 Tighter Rademacher Bound. Derive the following tighter version of the bound of theorem 5.9: for any $\delta > 0$, with probability at least $1-\delta$, for all $h \in \mathcal{H}$ and $\rho \in (0, 1]$ the following holds:

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2\gamma}{\rho}\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log\log_\gamma \frac{\gamma}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}} \tag{5.49}$$

for any $\gamma > 1$.

5.3 Importance weighted SVM. Suppose you wish to use SVMs to solve a learning problem where some training data points are more important than others. More formally, assume that each training point consists of a triplet (x_i, y_i, p_i), where $0 \leq p_i \leq 1$ is the importance of the ith point. Rewrite the primal SVM constrained optimization problem so that the penalty for mis-labeling a point x_i is scaled by the priority p_i. Then carry this modification through the derivation of the dual solution.

5.4 Sequential minimal optimization (SMO). The SMO algorithm is an optimization algorithm introduced to speed up the training of SVMs. SMO reduces a (potentially) large quadratic programming (QP) optimization problem into a series of small optimizations involving only two Lagrange multipliers. SMO reduces memory requirements, bypasses the need for numerical QP optimization and is easy to implement. In this question, we will derive the update rule for the SMO algorithm in the context of the dual formulation of the SVM problem.

(a) Assume that we want to optimize equation (5.33) only over α_1 and α_2. Show that the optimization problem reduces to

$$\max_{\alpha_1,\alpha_2} \underbrace{\alpha_1 + \alpha_2 - \frac{1}{2}K_{11}\alpha_1^2 - \frac{1}{2}K_{22}\alpha_2^2 - sK_{12}\alpha_1\alpha_2 - y_1\alpha_1 v_1 - y_2\alpha_2 v_2}_{\Psi_1(\alpha_1,\alpha_2)}$$

$$\text{subject to: } 0 \leq \alpha_1, \alpha_2 \leq C \wedge \alpha_1 + s\alpha_2 = \gamma\,,$$

where $\gamma = y_1 \sum_{i=3}^m y_i\alpha_i$, $s = y_1y_2 \in \{-1, +1\}$, $K_{ij} = (\mathbf{x}_i \cdot \mathbf{x}_j)$ and $v_i = \sum_{j=3}^m \alpha_j y_j K_{ij}$ for $i = 1, 2$.

(b) Substitute the linear constraint $\alpha_1 = \gamma - s\alpha_2$ into Ψ_1 to obtain a new objective function Ψ_2 that depends only on α_2. Show that the α_2 that maximizes Ψ_2 (without the constraints $0 \leq \alpha_1, \alpha_2 \leq C$) can be expressed as

$$\alpha_2 = \frac{s(K_{11} - K_{12})\gamma + y_2(v_1 - v_2) - s + 1}{\eta}\,,$$

where $\eta = K_{11} + K_{22} - 2K_{12}$.

(c) Show that

$$v_1 - v_2 = f(\mathbf{x}_1) - f(\mathbf{x}_2) + \alpha_2^* y_2 \eta - s y_2 \gamma (K_{11} - K_{12})$$

where $f(\mathbf{x}) = \sum_{i=1}^m \alpha_i^* y_i (\mathbf{x}_i \cdot \mathbf{x}) + b^*$ and α_i^* are values for the Lagrange multipliers prior to optimization over α_1 and α_2 (similarly, b^* is the previous value for the offset).

(d) Show that

$$\alpha_2 = \alpha_2^* + y_2 \frac{(y_2 - f(\mathbf{x}_2)) - (y_1 - f(\mathbf{x}_1))}{\eta}.$$

(e) For $s = +1$, define $L = \max\{0, \gamma - C\}$ and $H = \min\{C, \gamma\}$ as the lower and upper bounds on α_2. Similarly, for $s = -1$, define $L = \max\{0, -\gamma\}$ and $H = \min\{C, C - \gamma\}$. The update rule for SMO involves "clipping" the value of α_2, i.e.,

$$\alpha_2^{clip} = \begin{cases} \alpha_2 & \text{if } L < \alpha_2 < H \\ L & \text{if } \alpha_2 \le L \\ H & \text{if } \alpha_2 \ge H \end{cases}.$$

We subsequently solve for α_1 such that we satisfy the equality constraint, resulting in $\alpha_1 = \alpha_1^* + s(\alpha_2^* - \alpha_2^{clip})$. Why is "clipping" is required? How are L and H derived for the case $s = +1$?

5.5 SVMs hands-on.

(a) Download and install the `libsvm` software library from:

`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

(b) Download the `satimage` data set found at:

`http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

Merge the training and validation sets into one. We will refer to the resulting set as the training set from now on. Normalize both the training and test vectors.

(c) Consider the binary classification that consists of distinguishing class 6 from the rest of the data points. Use SVMs combined with polynomial kernels (see chapter 6) to solve this classification problem. To do so, randomly split the training data into ten equal-sized disjoint sets. For each value of the polynomial degree, $d = 1, 2, 3, 4$, plot the average cross-validation error plus or minus one standard deviation as a function of C (let the other parameters of polynomial kernels in `libsvm`, γ and c, be equal to their default values 1).

Report the best value of the trade-off constant C measured on the validation set.

(d) Let (C^*, d^*) be the best pair found previously. Fix C to be C^*. Plot the ten-fold cross-validation training and test errors for the hypotheses obtained as a function of d. Plot the average number of support vectors obtained as a function of d.

(e) How many of the support vectors lie on the margin hyperplanes?

(f) In the standard two-group classification, errors on positive or negative points are treated in the same manner. Suppose, however, that we wish to penalize an error on a negative point (false positive error) $k > 0$ times more than an error on a positive point. Give the dual optimization problem corresponding to SVMs modified in this way.

(g) Assume that k is an integer. Show how you can use `libsvm` without writing any additional code to find the solution of the modified SVMs just described.

(h) Apply the modified SVMs to the classification task previously examined and compare with your previous SVMs results for $k = 2, 4, 8, 16$.

5.6 Sparse SVM. One can give two types of arguments in favor of the SVM algorithm: one based on the sparsity of the support vectors, another based on the notion of margin. Suppose that instead of maximizing the margin, we choose instead to maximize sparsity by minimizing the L_p norm of the vector $\boldsymbol{\alpha}$ that defines the weight vector $\mathbf{w}$, for some $p \geq 1$. First, consider the case $p = 2$. This gives the following optimization problem:

$$\min_{\boldsymbol{\alpha}, b} \frac{1}{2} \sum_{i=1}^{m} \alpha_i^2 + C \sum_{i=1}^{m} \xi_i \tag{5.50}$$

$$\text{subject to } y_i \Big(\sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_i \cdot \mathbf{x}_j + b \Big) \geq 1 - \xi_i, i \in [m]$$

$$\xi_i, \alpha_i \geq 0, i \in [m].$$

(a) Show that modulo the non-negativity constraint on $\boldsymbol{\alpha}$, the problem coincides with an instance of the primal optimization problem of SVM.

(b) Derive the dual optimization of problem of (5.50).

(c) Setting $p = 1$ will induce a more sparse $\boldsymbol{\alpha}$. Derive the dual optimization in this case.

5.7 VC-dimension of canonical hyperplanes. The objective of this problem is derive a bound on the VC-dimension of canonical hyperplanes that does not depend on the dimension of feature space. Let $S \subseteq \{\mathbf{x} \colon \|\mathbf{x}\| \leq r\}$. We will show that the VC-dimension d of the set of canonical hyperplanes $\{x \mapsto \operatorname{sgn}(\mathbf{w} \cdot \mathbf{x}) \colon \min_{x \in S} |\mathbf{w} \cdot \mathbf{x}| = 1 \wedge \|\mathbf{w}\| \leq \Lambda\}$ verifies

$$d \leq r^2 \Lambda^2 \,. \tag{5.51}$$

(a) Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_d\}$ be a set that can be shattered. Show that for all $\mathbf{y} = (y_1, \ldots, y_d) \in \{-1, +1\}^d$, $d \leq \Lambda \| \sum_{i=1}^d y_i \mathbf{x}_i \|$.

(b) Use randomization over the labels $\mathbf{y}$ and Jensen's inequality to show that $d \leq \Lambda \sqrt{\sum_{i=1}^d \|\mathbf{x}_i\|^2}$.

(c) Conclude that $d \leq r^2 \Lambda^2$.

6 Kernel Methods

Kernel methods are widely used in machine learning. They are flexible techniques that can be used to extend algorithms such as SVMs to define non-linear decision boundaries. Other algorithms that only depend on inner products between sample points can be extended similarly, many of which will be studied in future chapters.

The main idea behind these methods is based on so-called *kernels* or *kernel functions*, which, under some technical conditions of symmetry and *positive-definiteness*, implicitly define an inner product in a high-dimensional space. Replacing the original inner product in the input space with positive definite kernels immediately extends algorithms such as SVMs to a linear separation in that high-dimensional space, or, equivalently, to a non-linear separation in the input space.

In this chapter, we present the main definitions and key properties of positive definite symmetric kernels, including the proof of the fact that they define an inner product in a Hilbert space, as well as their closure properties. We then extend the SVM algorithm using these kernels and present several theoretical results including general margin-based learning guarantees for hypothesis sets based on kernels. We also introduce *negative definite symmetric kernels* and point out their relevance to the construction of positive definite kernels, in particular from distances or metrics. Finally, we illustrate the design of kernels for non-vectorial discrete structures by introducing a general family of kernels for sequences, *rational kernels*. We describe an efficient algorithm for the computation of these kernels and illustrate them with several examples.

6.1 Introduction

In the previous chapter, we presented an algorithm for linear classification, SVMs, which is both effective in applications and benefits from a strong theoretical justification. In practice, linear separation is often not possible. Figure 6.1a shows an example where any hyperplane crosses both populations. However, one can use

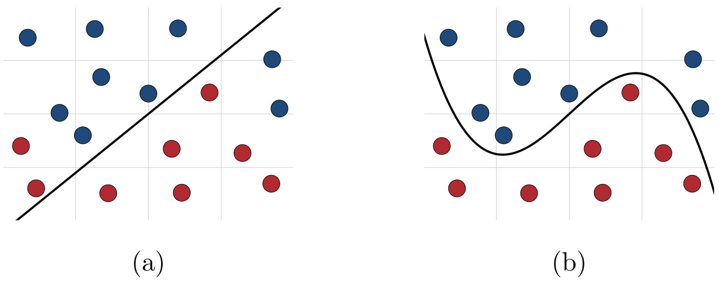


Figure 6.1
Non-linearly separable case. The classification task consists of discriminating between blue and red points. (a) No hyperplane can separate the two populations. (b) A non-linear mapping can be used instead.

more complex functions to separate the two sets as in figure 6.1b. One way to define such a non-linear decision boundary is to use a non-linear mapping Φ from the input space $\mathcal{X}$ to a higher-dimensional space $\mathbb{H}$, where linear separation is possible (see figure 6.2).

The dimension of $\mathbb{H}$ can truly be very large in practice. For example, in the case of document classification, one may wish to use as features sequences of three consecutive words, i.e., *trigrams*. Thus, with a vocabulary of just 100,000 words, the dimension of the feature space $\mathbb{H}$ reaches 10^{15}. On the positive side, the margin bounds presented in section 5.4 show that, remarkably, the generalization ability of large-margin classification algorithms such as SVMs do not depend on the dimension of the feature space, but only on the margin ρ and the number of training examples m. Thus, with a favorable margin ρ, such algorithms could succeed even in very high-dimensional space. However, determining the hyperplane solution requires multiple inner product computations in high-dimensional spaces, which can become be very costly.

A solution to this problem is to use *kernel methods*, which are based on *kernels* or *kernel functions*.

Definition 6.1 (Kernels) *A function $K: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a* kernel over $\mathcal{X}$.

The idea is to define a kernel K such that for any two points $x, x' \in \mathcal{X}$, $K(x, x')$ be equal to an inner product of vectors $\Phi(x)$ and $\Phi(y)$:[6]

$$\forall x, x' \in \mathcal{X}, \quad K(x, x') = \langle \Phi(x), \Phi(x') \rangle\,, \tag{6.1}$$

[6] To differentiate that inner product from the one of the input space, we will typically denote it by $\langle \cdot, \cdot \rangle$.

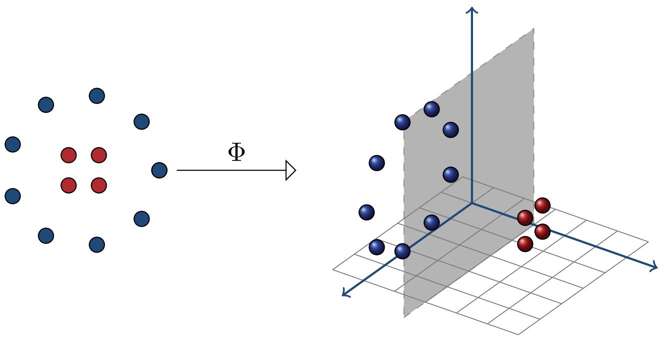


Figure 6.2
An example of a non-linear mapping from 2-dimensions to 3-dimensions, where the task becomes linearly seperable.

for some mapping $\Phi\colon \mathcal{X} \to \mathbb{H}$ to a Hilbert space $\mathbb{H}$ called a *feature space*. Since an inner product is a measure of the similarity of two vectors, K is often interpreted as a similarity measure between elements of the input space $\mathcal{X}$.

An important advantage of such a kernel K is efficiency: K is often significantly more efficient to compute than Φ and an inner product in $\mathbb{H}$. We will see several common examples where the computation of $K(x, x')$ can be achieved in $O(N)$ while that of $\langle \Phi(x), \Phi(x') \rangle$ typically requires $O(\dim(\mathbb{H}))$ work, with $\dim(\mathbb{H}) \gg N$. Furthermore, in some cases, the dimension of $\mathbb{H}$ is infinite.

Perhaps an even more crucial benefit of such a kernel function K is flexibility: there is no need to explicitly define or compute a mapping Φ. The kernel K can be arbitrarily chosen so long as the existence of Φ is guaranteed, i.e. K satisfies *Mercer's condition* (see theorem 6.2).

Theorem 6.2 (Mercer's condition) *Let $\mathcal{X} \subset \mathbb{R}^N$ be a compact set and let $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a continuous and symmetric function. Then, K admits a uniformly convergent expansion of the form*

$$K(x, x') = \sum_{n=0}^{\infty} a_n \phi_n(x) \phi_n(x'),$$

with $a_n > 0$ iff for any square integrable function c ($c \in L_2(\mathcal{X})$), the following condition holds:

$$\int \int_{\mathcal{X} \times \mathcal{X}} c(x) c(x') K(x, x') dx dx' \geq 0.$$

This condition is important to guarantee the convexity of the optimization problem for algorithms such as SVMs, thereby ensuring convergence to a global minimum. A condition that is equivalent to Mercer's condition under the assumptions of the theorem is that the kernel K be *positive definite symmetric* (PDS). This property is in fact more general since in particular it does not require any assumption about $\mathcal{X}$. In the next section, we give the definition of this property and present several commonly used examples of PDS kernels, then show that PDS kernels induce an inner product in a Hilbert space, and prove several general closure properties for PDS kernels.

6.2 Positive definite symmetric kernels

6.2.1 Definitions

Definition 6.3 (Positive definite symmetric kernels) *A kernel* $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *is said to be* positive definite symmetric *(PDS) if for any* $\{x_1, \ldots, x_m\} \subseteq \mathcal{X}$*, the matrix* $\mathbf{K} = [K(x_i, x_j)]_{ij} \in \mathbb{R}^{m \times m}$ *is symmetric positive semidefinite (SPSD).*

$\mathbf{K}$ is SPSD if it is symmetric and one of the following two equivalent conditions holds:

- the eigenvalues of $\mathbf{K}$ are non-negative;
- for any column vector $\mathbf{c} = (c_1, \ldots, c_m)^\top \in \mathbb{R}^{m \times 1}$,

$$\mathbf{c}^\top \mathbf{K} \mathbf{c} = \sum_{i,j=1}^{m} c_i c_j K(x_i, x_j) \geq 0. \tag{6.2}$$

For a sample $S = (x_1, \ldots, x_m)$, $\mathbf{K} = [K(x_i, x_j)]_{ij} \in \mathbb{R}^{m \times m}$ is called the *kernel matrix* or the *Gram matrix* associated to K and the sample S.

Let us insist on the terminology: the kernel matrix associated to a *positive definite kernel* is *positive semidefinite* . This is the correct mathematical terminology. Nevertheless, the reader should be aware that in the context of machine learning, some authors have chosen to use instead the term *positive definite kernel* to imply a *positive definite* kernel matrix or used new terms such as *positive semidefinite kernel*.

The following are some standard examples of PDS kernels commonly used in applications.

Example 6.4 (Polynomial kernels) For any constant $c > 0$, a *polynomial kernel of degree* $d \in \mathbb{N}$ is the kernel K defined over $\mathbb{R}^N$ by:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^N, \quad K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + c)^d. \tag{6.3}$$

Polynomial kernels map the input space to a higher-dimensional space of dimension $\binom{N+d}{d}$ (see exercise 6.12). As an example, for an input space of dimension $N = 2$, a second-degree polynomial ($d = 2$) corresponds to the following inner product in

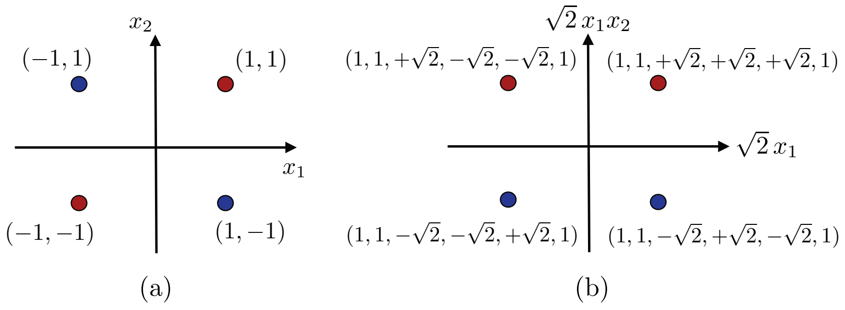

x_2
x_1
$(-1,1)$
$(1,1)$
$(-1,-1)$
$(1,-1)$
(a)
$\sqrt{2}\,x_1x_2$
$\sqrt{2}\,x_1$
$(1,1,+\sqrt{2},-\sqrt{2},-\sqrt{2},1)$
$(1,1,+\sqrt{2},+\sqrt{2},+\sqrt{2},1)$
$(1,1,-\sqrt{2},-\sqrt{2},+\sqrt{2},1)$
$(1,1,-\sqrt{2},+\sqrt{2},-\sqrt{2},1)$
(b)

Figure 6.3
Illustration of the XOR classification problem and the use of polynomial kernels. (a) XOR problem linearly non-separable in the input space. (b) Linearly separable using second-degree polynomial kernel.

dimension 6:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^2, \quad K(\mathbf{x}, \mathbf{x}') = (x_1x_1' + x_2x_2' + c)^2 = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\,x_1x_2 \\ \sqrt{2c}\,x_1 \\ \sqrt{2c}\,x_2 \\ c \end{bmatrix} \cdot \begin{bmatrix} x_1'^2 \\ x_2'^2 \\ \sqrt{2}\,x_1'x_2' \\ \sqrt{2c}\,x_1' \\ \sqrt{2c}\,x_2' \\ c \end{bmatrix}. \tag{6.4}$$

Thus, the features corresponding to a second-degree polynomial are the original features (x_1 and x_2), as well as products of these features, and the constant feature. More generally, the features associated to a polynomial kernel of degree d are all the monomials of degree at most d based on the original features. The explicit expression of polynomial kernels as inner products, as in (6.4), proves directly that they are PDS kernels.

To illustrate the application of polynomial kernels, consider the example of figure 6.3a which shows a simple data set in dimension two that is not linearly separable. This is known as the XOR problem due to its interpretation in terms of the exclusive OR (XOR) function: the label of a point is blue iff exactly one of its coordinates is 1. However, if we map these points to the six-dimensional space defined by a second-degree polynomial as described in (6.4), then the problem becomes separable by the hyperplane of equation $x_1x_2 = 0$. Figure 6.3b illustrates that by showing the projection of these points on the two-dimensional space defined by their third and fourth coordinates.

Example 6.5 (Gaussian kernels) For any constant $\sigma > 0$, a *Gaussian kernel* or *radial basis function (RBF)* is the kernel K defined over $\mathbb{R}^N$ by:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^N, \quad K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}' - \mathbf{x}\|^2}{2\sigma^2}\right). \tag{6.5}$$

Gaussian kernels are among the most frequently used kernels in applications. We will prove in section 6.2.3 that they are PDS kernels and that they can be derived by *normalization* from the kernels $K' \colon (\mathbf{x}, \mathbf{x}') \mapsto \exp\left(\frac{\mathbf{x} \cdot \mathbf{x}'}{\sigma^2}\right)$. Using the power series expansion of the exponential function, we can rewrite the expression of K' as follows:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^N, \quad K'(\mathbf{x}, \mathbf{x}') = \sum_{n=0}^{+\infty} \frac{(\mathbf{x} \cdot \mathbf{x}')^n}{\sigma^{2n}\, n!},$$

which shows that the kernels K', and thus Gaussian kernels, are positive linear combinations of polynomial kernels of all degrees $n \geq 0$.

Example 6.6 (Sigmoid kernels) For any real constants $a, b \geq 0$, a *sigmoid kernel* is the kernel K defined over $\mathbb{R}^N$ by:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^N, \quad K(\mathbf{x}, \mathbf{x}') = \tanh\big(a(\mathbf{x} \cdot \mathbf{x}') + b\big). \tag{6.6}$$

Using sigmoid kernels with SVMs leads to an algorithm that is closely related to learning algorithms based on simple neural networks, which are also often defined via a sigmoid function. When $a < 0$ or $b < 0$, the kernel is not PDS and the corresponding neural network does not benefit from the convergence guarantees of convex optimization (see exercise 6.18).

6.2.2 Reproducing kernel Hilbert space

Here, we prove the crucial property of PDS kernels, which is to induce an inner product in a Hilbert space. The proof will make use of the following lemma.

Lemma 6.7 (Cauchy-Schwarz inequality for PDS kernels) *Let K be a PDS kernel. Then, for any $x, x' \in \mathcal{X}$,*

$$K(x, x')^2 \leq K(x, x)K(x', x'). \tag{6.7}$$

Proof: Consider the matrix $\mathbf{K} = \begin{pmatrix} K(x,x) & K(x,x') \\ K(x',x) & K(x',x') \end{pmatrix}$. By definition, if K is PDS, then $\mathbf{K}$ is SPSD for all $x, x' \in \mathcal{X}$. In particular, the product of the eigenvalues of $\mathbf{K}$, $\det(\mathbf{K})$, must be non-negative, thus, using $K(x', x) = K(x, x')$, we have

$$\det(\mathbf{K}) = K(x, x)K(x', x') - K(x, x')^2 \geq 0,$$

which concludes the proof. □

The following is the main result of this section.

Theorem 6.8 (Reproducing kernel Hilbert space (RKHS)) *Let $K \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel. Then, there exists a Hilbert space $\mathbb{H}$ (see definition A.2) and a mapping Φ*

from $\mathcal{X}$ to $\mathbb{H}$ such that:

$$\forall x, x' \in \mathcal{X}, \quad K(x, x') = \langle \Phi(x), \Phi(x') \rangle . \tag{6.8}$$

Furthermore, $\mathbb{H}$ has the following property known as the reproducing property*:*

$$\forall h \in \mathbb{H}, \forall x \in \mathcal{X}, \quad h(x) = \langle h, K(x, \cdot) \rangle . \tag{6.9}$$

$\mathbb{H}$ is called a reproducing kernel Hilbert space *(RKHS) associated to K.*

Proof: For any $x \in \mathcal{X}$, define $\Phi(x) \colon \mathcal{X} \to \mathbb{R}^{\mathcal{X}}$ as follows:

$$\forall x' \in \mathcal{X}, \ \Phi(x)(x') = K(x, x').$$

We define $\mathbb{H}_0$ as the set of finite linear combinations of such functions $\Phi(x)$:

$$\mathbb{H}_0 = \Big\{ \sum_{i \in I} a_i \Phi(x_i) \colon a_i \in \mathbb{R}, x_i \in \mathcal{X}, |I| < \infty \Big\}.$$

Now, we introduce an operation $\langle \cdot, \cdot \rangle$ on $\mathbb{H}_0 \times \mathbb{H}_0$ defined for all $f, g \in \mathbb{H}_0$ with $f = \sum_{i \in I} a_i \Phi(x_i)$ and $g = \sum_{j \in J} b_j \Phi(x'_j)$ by

$$\langle f, g \rangle = \sum_{i \in I, j \in J} a_i b_j K(x_i, x'_j) = \sum_{j \in J} b_j f(x'_j) = \sum_{i \in I} a_i g(x_i).$$

By definition, $\langle \cdot, \cdot \rangle$ is symmetric. The last two equations show that $\langle f, g \rangle$ does not depend on the particular representations of f and g, and also show that $\langle \cdot, \cdot \rangle$ is bilinear. Further, for any $f = \sum_{i \in I} a_i \Phi(x_i) \in \mathbb{H}_0$, since K is PDS, we have

$$\langle f, f \rangle = \sum_{i, j \in I} a_i a_j K(x_i, x_j) \geq 0.$$

Thus, $\langle \cdot, \cdot \rangle$ is positive semidefinite bilinear form. This inequality implies more generally using the bilinearity of $\langle \cdot, \cdot \rangle$ that for any $f_1, \ldots, f_m$ and $c_1, \ldots, c_m \in \mathbb{R}$,

$$\sum_{i,j=1}^{m} c_i c_j \langle f_i, f_j \rangle = \Big\langle \sum_{i=1}^{m} c_i f_i, \sum_{j=1}^{m} c_j f_j \Big\rangle \geq 0.$$

Hence, $\langle \cdot, \cdot \rangle$ is a PDS kernel on $\mathbb{H}_0$. Thus, for any $f \in \mathbb{H}_0$ and any $x \in \mathcal{X}$, by lemma 6.7, we can write

$$\langle f, \Phi(x) \rangle^2 \leq \langle f, f \rangle \langle \Phi(x), \Phi(x) \rangle.$$

Further, we observe the reproducing property of $\langle \cdot, \cdot \rangle$: for any $f = \sum_{i \in I} a_i \Phi(x_i) \in \mathbb{H}_0$, by definition of $\langle \cdot, \cdot \rangle$,

$$\forall x \in \mathcal{X}, \quad f(x) = \sum_{i \in I} a_i K(x_i, x) = \langle f, \Phi(x) \rangle . \tag{6.10}$$

Thus, $[f(x)]^2 \le \langle f, f\rangle K(x,x)$ for all $x \in \mathcal{X}$, which shows the definiteness of $\langle \cdot, \cdot \rangle$. This implies that $\langle \cdot, \cdot \rangle$ defines an inner product on $\mathbb{H}_0$, which thereby becomes a pre-Hilbert space. $\mathbb{H}_0$ can be completed to form a Hilbert space $\mathbb{H}$ in which it is dense, following a standard construction. By the Cauchy-Schwarz inequality, for any $x \in \mathcal{X}$, $f \mapsto \langle f, \Phi(x)\rangle$ is Lipschitz, therefore continuous. Thus, since $\mathbb{H}_0$ is dense in $\mathbb{H}$, the reproducing property (6.10) also holds over $\mathbb{H}$. $\square$

The Hilbert space $\mathbb{H}$ defined in the proof of the theorem for a PDS kernel K is called *the reproducing kernel Hilbert space (RKHS) associated to K*. Any Hilbert space $\mathbb{H}$ such that there exists $\Phi\colon \mathcal{X} \to \mathbb{H}$ with $K(x,x') = \langle \Phi(x), \Phi(x')\rangle$ for all $x, x' \in \mathcal{X}$ is called a *feature space* associated to K and Φ is called a *feature mapping*. We will denote by $\|\cdot\|_{\mathbb{H}}$ the norm induced by the inner product in feature space $\mathbb{H}$: $\|\mathbf{w}\|_{\mathbb{H}} = \sqrt{\langle \mathbf{w}, \mathbf{w}\rangle}$ for all $\mathbf{w} \in \mathbb{H}$. Note that the feature spaces associated to K are in general not unique and may have different dimensions. In practice, when referring to the *dimension of the feature space* associated to K, we either refer to the dimension of the feature space based on a feature mapping described explicitly, or to that of the RKHS associated to K.

Theorem 6.8 implies that PDS kernels can be used to implicitly define a feature space or feature vectors. As already underlined in previous chapters, the role played by the features in the success of learning algorithms is crucial: with poor features, uncorrelated with the target labels, learning could become very challenging or even impossible; in contrast, good features could provide invaluable clues to the algorithm. Therefore, in the context of learning with PDS kernels and for a fixed input space, the problem of seeking useful features is replaced by that of finding useful PDS kernels. While features represented the user's prior knowledge about the task in the standard learning problems, here PDS kernels will play this role. Thus, in practice, an appropriate choice of PDS kernel for a task will be crucial.

6.2.3 Properties

This section highlights several important properties of PDS kernels. We first show that PDS kernels can be *normalized* and that the resulting normalized kernels are also PDS. We also introduce the definition of *empirical kernel maps* and describe their properties and extension. We then prove several important closure properties of PDS kernels, which can be used to construct complex PDS kernels from simpler ones.

To any kernel K, we can associate a *normalized kernel* K' defined by

$$\forall x, x' \in \mathcal{X}, \quad K'(x,x') = \begin{cases} 0 & \text{if } (K(x,x) = 0) \vee (K(x',x') = 0) \\ \frac{K(x,x')}{\sqrt{K(x,x)K(x',x')}} & \text{otherwise.} \end{cases} \tag{6.11}$$

By definition, for a normalized kernel K', $K'(x,x) = 1$ for all $x \in \mathcal{X}$ such that $K(x,x) \neq 0$. An example of normalized kernel is the Gaussian kernel with parameter $\sigma > 0$, which is the normalized kernel associated to $K'\colon (\mathbf{x}, \mathbf{x}') \mapsto \exp\left(\frac{\mathbf{x}\cdot\mathbf{x}'}{\sigma^2}\right)$:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^N, \frac{K'(\mathbf{x},\mathbf{x}')}{\sqrt{K'(\mathbf{x},\mathbf{x})K'(\mathbf{x}',\mathbf{x}')}} = \frac{e^{\frac{\mathbf{x}\cdot\mathbf{x}'}{\sigma^2}}}{e^{\frac{\|\mathbf{x}\|^2}{2\sigma^2}} e^{\frac{\|\mathbf{x}'\|^2}{2\sigma^2}}} = \exp\left(-\frac{\|\mathbf{x}'-\mathbf{x}\|^2}{2\sigma^2}\right). \tag{6.12}$$

Lemma 6.9 (Normalized PDS kernels) *Let K be a PDS kernel. Then, the normalized kernel K' associated to K is PDS.*

Proof: Let $\{x_1, \ldots, x_m\} \subseteq \mathcal{X}$ and let $\mathbf{c}$ be an arbitrary vector in $\mathbb{R}^m$. We will show that the sum $\sum_{i,j=1}^m c_i c_j K'(x_i, x_j)$ is non-negative. By lemma 6.7, if $K(x_i, x_i) = 0$ then $K(x_i, x_j) = 0$ and thus $K'(x_i, x_j) = 0$ for all $j \in [m]$. Thus, we can assume that $K(x_i, x_i) > 0$ for all $i \in [m]$. Then, the sum can be rewritten as follows:

$$\sum_{i,j=1}^m \frac{c_i c_j K(x_i,x_j)}{\sqrt{K(x_i,x_i)K(x_j,x_j)}} = \sum_{i,j=1}^m \frac{c_i c_j \langle \Phi(x_i), \Phi(x_j)\rangle}{\|\Phi(x_i)\|_{\mathbb{H}} \|\Phi(x_j)\|_{\mathbb{H}}} = \left\| \sum_{i=1}^m \frac{c_i \Phi(x_i)}{\|\Phi(x_i)\|_{\mathbb{H}}} \right\|_{\mathbb{H}}^2 \geq 0,$$

where Φ is a feature mapping associated to K, which exists by theorem 6.8. □

As indicated earlier, PDS kernels can be interpreted as a similarity measure since they induce an inner product in some Hilbert space $\mathbb{H}$. This is more evident for a normalized kernel K since $K(x, x')$ is then exactly the cosine of the angle between the feature vectors $\Phi(x)$ and $\Phi(x')$, provided that none of them is zero: $\Phi(x)$ and $\Phi(x')$ are then unit vectors since $\|\Phi(x)\|_{\mathbb{H}} = \|\Phi(x')\|_{\mathbb{H}} = \sqrt{K(x,x)} = 1$.

While one of the advantages of PDS kernels is an implicit definition of a feature mapping, in some instances, it may be desirable to define an explicit feature mapping based on a PDS kernel. This may be to work in the primal for various optimization and computational reasons, to derive an approximation based on an explicit mapping, or as part of a theoretical analysis where an explicit mapping is more convenient. The *empirical kernel map* Φ associated to a PDS kernel K is a feature mapping that can be used precisely in such contexts. Given a training sample containing points $x_1, \ldots, x_m \in \mathcal{X}$, $\Phi\colon \mathcal{X} \to \mathbb{R}^m$ is defined for all $x \in \mathcal{X}$ by

$$\Phi(x) = \begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_m) \end{bmatrix}.$$

Thus, $\Phi(x)$ is the vector of the K-similarity measures of x with each of the training points. Let $\mathbf{K}$ be the kernel matrix associated to K and $\mathbf{e}_i$ the ith unit vector. Note that for any $i \in [m]$, $\Phi(x_i)$ is the ith column of $\mathbf{K}$, that is $\Phi(x_i) = \mathbf{K}\mathbf{e}_i$. In particular, for all $i, j \in [m]$,

$$\langle \Phi(x_i), \Phi(x_j) \rangle = (\mathbf{K}\mathbf{e}_i)^\top (\mathbf{K}\mathbf{e}_j) = \mathbf{e}_i^\top \mathbf{K}^2 \mathbf{e}_j.$$

Thus, the kernel matrix $\mathbf{K}'$ associated to Φ is $\mathbf{K}^2$. It may desirable in some cases to define a feature mapping whose kernel matrix coincides with $\mathbf{K}$. Let $\mathbf{K}^{\dagger\frac{1}{2}}$ denote the SPSD matrix whose square is $\mathbf{K}^\dagger$, the pseudo-inverse of $\mathbf{K}$. $\mathbf{K}^{\dagger\frac{1}{2}}$ can be derived from $\mathbf{K}^\dagger$ via singular value decomposition and if the matrix $\mathbf{K}$ is invertible, $\mathbf{K}^{\dagger\frac{1}{2}}$ coincides with $\mathbf{K}^{-1/2}$ (see appendix A for properties of the pseudo-inverse). Then, Ψ can be defined as follows using the empirical kernel map Φ:

$$\forall x \in \mathcal{X}, \quad \Psi(x) = \mathbf{K}^{\dagger\frac{1}{2}} \Phi(x).$$

Using the identity $\mathbf{K}\mathbf{K}^\dagger\mathbf{K} = \mathbf{K}$ valid for any symmetric matrix $\mathbf{K}$, for all $i, j \in [m]$, the following holds:

$$\langle \Psi(x_i), \Psi(x_j) \rangle = (\mathbf{K}^{\dagger\frac{1}{2}}\mathbf{K}\mathbf{e}_i)^\top (\mathbf{K}^{\dagger\frac{1}{2}}\mathbf{K}\mathbf{e}_j) = \mathbf{e}_i^\top \mathbf{K}\mathbf{K}^\dagger\mathbf{K}\mathbf{e}_j = \mathbf{e}_i^\top \mathbf{K}\mathbf{e}_j.$$

Thus, the kernel matrix associated to Ψ is $\mathbf{K}$. Finally, note that for the feature mapping $\Omega\colon \mathcal{X} \to \mathbb{R}^m$ defined by

$$\forall x \in \mathcal{X}, \quad \Omega(x) = \mathbf{K}^\dagger \Phi(x),$$

for all $i, j \in [m]$, we have $\langle \Omega(x_i), \Omega(x_j) \rangle = \mathbf{e}_i^\top \mathbf{K}\mathbf{K}^\dagger\mathbf{K}^\dagger\mathbf{K}\mathbf{e}_j = \mathbf{e}_i^\top \mathbf{K}\mathbf{K}^\dagger\mathbf{e}_j$, using the identity $\mathbf{K}^\dagger\mathbf{K}^\dagger\mathbf{K} = \mathbf{K}^\dagger$ valid for any symmetric matrix $\mathbf{K}$. Thus, the kernel matrix associated to Ω is $\mathbf{K}\mathbf{K}^\dagger$, which reduces to the identity matrix $\mathbf{I} \in \mathbb{R}^{m\times m}$ when $\mathbf{K}$ is invertible, since $\mathbf{K}^\dagger = \mathbf{K}^{-1}$ in that case.

As pointed out in the previous section, kernels represent the user's prior knowledge about a task. In some cases, a user may come up with appropriate similarity measures or PDS kernels for some subtasks — for example, for different subcategories of proteins or text documents to classify. But how can the user combine these PDS kernels to form a PDS kernel for the entire class? Is the resulting combined kernel guaranteed to be PDS? In the following, we will show that PDS kernels are closed under several useful operations which can be used to design complex PDS kernels. These operations are the sum and the product of kernels, as well as the *tensor product* of two kernels K and K', denoted by $K \otimes K'$ and defined by

$$\forall x_1, x_2, x_1', x_2' \in \mathcal{X}, \quad (K \otimes K')(x_1, x_1', x_2, x_2') = K(x_1, x_2)K'(x_1', x_2').$$

They also include the pointwise limit: given a sequence of kernels $(K_n)_{n\in\mathbb{N}}$ such that for all $x, x' \in \mathcal{X}$ $(K_n(x, x'))_{n\in\mathbb{N}}$ admits a limit, the pointwise limit of $(K_n)_{n\in\mathbb{N}}$ is the kernel K defined for all $x, x' \in \mathcal{X}$ by $K(x, x') = \lim_{n\to+\infty}(K_n)(x, x')$. Similarly, if $\sum_{n=0}^\infty a_n x^n$ is a power series with radius of convergence $\rho > 0$ and K a kernel taking values in $(-\rho, +\rho)$, then $\sum_{n=0}^\infty a_n K^n$ is the kernel obtained by composition of K with that power series. The following theorem provides closure guarantees for all of these operations.

Theorem 6.10 (PDS kernels — closure properties) *PDS kernels are closed under sum, product, tensor product, pointwise limit, and composition with a power series $\sum_{n=0}^{\infty} a_n x^n$ with $a_n \geq 0$ for all $n \in \mathbb{N}$.*

Proof: We start with two kernel matrices, $\mathbf{K}$ and $\mathbf{K}'$, generated from PDS kernels K and K' for an arbitrary set of m points. By assumption, these kernel matrices are SPSD. Observe that for any $\mathbf{c} \in \mathbb{R}^{m \times 1}$,

$$(\mathbf{c}^\top \mathbf{K} \mathbf{c} \geq 0) \wedge (\mathbf{c}^\top \mathbf{K}' \mathbf{c} \geq 0) \Rightarrow \mathbf{c}^\top (\mathbf{K} + \mathbf{K}') \mathbf{c} \geq 0.$$

By (6.2), this shows that $\mathbf{K} + \mathbf{K}'$ is SPSD and thus that $K + K'$ is PDS. To show closure under product, we will use the fact that for any SPSD matrix $\mathbf{K}$ there exists $\mathbf{M}$ such that $\mathbf{K} = \mathbf{M}\mathbf{M}^\top$. The existence of $\mathbf{M}$ is guaranteed as it can be generated via, for instance, singular value decomposition of $\mathbf{K}$, or by Cholesky decomposition. The kernel matrix associated to KK' is $(\mathbf{K}_{ij}\mathbf{K}'_{ij})_{ij}$. For any $\mathbf{c} \in \mathbb{R}^{m \times 1}$, expressing $\mathbf{K}_{ij}$ in terms of the entries of $\mathbf{M}$, we can write

$$\begin{aligned}
\sum_{i,j=1}^{m} c_i c_j (\mathbf{K}_{ij}\mathbf{K}'_{ij}) &= \sum_{i,j=1}^{m} c_i c_j \bigg(\Big[\sum_{k=1}^{m} \mathbf{M}_{ik}\mathbf{M}_{jk} \Big] \mathbf{K}'_{ij} \bigg) \\
&= \sum_{k=1}^{m} \Bigg[\sum_{i,j=1}^{m} c_i c_j \mathbf{M}_{ik}\mathbf{M}_{jk}\mathbf{K}'_{ij} \Bigg] \\
&= \sum_{k=1}^{m} \mathbf{z}_k^\top \mathbf{K}' \mathbf{z}_k \geq 0,
\end{aligned}$$

with $\mathbf{z}_k = \begin{bmatrix} c_1\mathbf{M}_{1k} \\ \vdots \\ c_m\mathbf{M}_{mk} \end{bmatrix}$. This shows that PDS kernels are closed under product. The tensor product of K and K' is PDS as the product of the two PDS kernels $(x_1, x'_1, x_2, x'_2) \mapsto K(x_1, x_2)$ and $(x_1, x'_1, x_2, x'_2) \mapsto K'(x'_1, x'_2)$. Next, let $(K_n)_{n \in \mathbb{N}}$ be a sequence of PDS kernels with pointwise limit K. Let $\mathbf{K}$ be the kernel matrix associated to K and $\mathbf{K}_n$ the one associated to K_n for any $n \in \mathbb{N}$. Observe that

$$(\forall n, \mathbf{c}^\top \mathbf{K}_n \mathbf{c} \geq 0) \Rightarrow \lim_{n \to \infty} \mathbf{c}^\top \mathbf{K}_n \mathbf{c} = \mathbf{c}^\top \mathbf{K} \mathbf{c} \geq 0.$$

This shows the closure under pointwise limit. Finally, assume that K is a PDS kernel with $|K(x, x')| < \rho$ for all $x, x' \in \mathcal{X}$ and let $f \colon x \mapsto \sum_{n=0}^{\infty} a_n x^n, a_n \geq 0$ be a power series with radius of convergence ρ. Then, for any $n \in \mathbb{N}$, K^n and thus $a_n K^n$ are PDS by closure under product. For any $N \in \mathbb{N}$, $\sum_{n=0}^{N} a_n K^n$ is PDS by closure under sum of $a_n K^n$s and $f \circ K$ is PDS by closure under the limit of $\sum_{n=0}^{N} a_n K^n$ as N tends to infinity. □

The theorem implies in particular that for any PDS kernel matrix K, $\exp(K)$ is PDS, since the radius of convergence of exp is infinite. In particular, the kernel $K' \colon (\mathbf{x}, \mathbf{x}') \mapsto \exp\left(\frac{\mathbf{x} \cdot \mathbf{x}'}{\sigma^2}\right)$ is PDS since $(\mathbf{x}, \mathbf{x}') \mapsto \frac{\mathbf{x} \cdot \mathbf{x}'}{\sigma^2}$ is PDS. Thus, by lemma 6.9, this shows that a Gaussian kernel, which is the normalized kernel associated to K', is PDS.

6.3 Kernel-based algorithms

In this section we discuss how SVMs can be used with kernels and analyze the impact that kernels have on generalization.

6.3.1 SVMs with PDS kernels

In chapter 5, we noted that the dual optimization problem for SVMs as well as the form of the solution did not directly depend on the input vectors but only on inner products. Since a PDS kernel implicitly defines an inner product (theorem 6.8), we can extend SVMs and combine it with an arbitrary PDS kernel K by replacing each instance of an inner product $x \cdot x'$ with $K(x, x')$. This leads to the following general form of the SVM optimization problem and solution with PDS kernels extending (5.33):

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{6.13}$$

$$\text{subject to: } 0 \leq \alpha_i \leq C \wedge \sum_{i=1}^{m} \alpha_i y_i = 0, i \in [m].$$

In view of (5.34), the hypothesis h solution can be written as:

$$h(x) = \operatorname{sgn}\Big(\sum_{i=1}^{m} \alpha_i y_i K(x_i, x) + b\Big), \tag{6.14}$$

with $b = y_i - \sum_{j=1}^{m} \alpha_j y_j K(x_j, x_i)$ for any x_i with $0 < \alpha_i < C$. We can rewrite the optimization problem (6.13) in a vector form, by using the kernel matrix $\mathbf{K}$ associated to K for the training sample $(x_1, \ldots, x_m)$ as follows:

$$\max_{\boldsymbol{\alpha}} 2\, \mathbf{1}^\top \boldsymbol{\alpha} - (\boldsymbol{\alpha} \circ \mathbf{y})^\top \mathbf{K} (\boldsymbol{\alpha} \circ \mathbf{y}) \tag{6.15}$$

$$\text{subject to: } \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{C} \wedge \boldsymbol{\alpha}^\top \mathbf{y} = 0.$$

In this formulation, $\boldsymbol{\alpha} \circ \mathbf{y}$ is the Hadamard product or entry-wise product of the vectors $\boldsymbol{\alpha}$ and $\mathbf{y}$. Thus, it is the column vector in $\mathbb{R}^{m \times 1}$ whose ith component equals $\alpha_i y_i$. The solution in vector form is the same as in (6.14), but with $b = y_i - (\boldsymbol{\alpha} \circ \mathbf{y})^\top \mathbf{K} \mathbf{e}_i$ for any x_i with $0 < \alpha_i < C$.

This version of SVMs used with PDS kernels is the general form of SVMs we will consider in all that follows. The extension is important, since it enables an implicit non-linear mapping of the input points to a high-dimensional space where large-margin separation is sought.

Many other algorithms in areas including regression, ranking, dimensionality reduction or clustering can be extended using PDS kernels following the same scheme (see in particular chapters 9, 10, 11, 15).

6.3.2 Representer theorem

Observe that modulo the offset b, the hypothesis solution of SVMs can be written as a linear combination of the functions $K(x_i, \cdot)$, where x_i is a sample point. The following theorem known as the *representer theorem* shows that this is in fact a general property that holds for a broad class of optimization problems, including that of SVMs with no offset.

Theorem 6.11 (Representer theorem) *Let $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel and $\mathbb{H}$ its corresponding RKHS. Then, for any non-decreasing function $G\colon \mathbb{R} \to \mathbb{R}$ and any loss function $L\colon \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$, the optimization problem*

$$\underset{h \in \mathbb{H}}{\operatorname{argmin}}\, F(h) = \underset{h \in \mathbb{H}}{\operatorname{argmin}}\, G(\|h\|_{\mathbb{H}}) + L\big(h(x_1), \ldots, h(x_m)\big)$$

admits a solution of the form $h^* = \sum_{i=1}^m \alpha_i K(x_i, \cdot)$. If G is further assumed to be increasing, then any solution has this form.

Proof: Let $\mathbb{H}_1 = \operatorname{span}(\{K(x_i, \cdot)\colon i \in [m]\})$. Any $h \in \mathbb{H}$ admits the decomposition $h = h_1 + h^\perp$ according to $\mathbb{H} = \mathbb{H}_1 \oplus \mathbb{H}_1^\perp$, where $\oplus$ is the direct sum. Since G is non-decreasing, $G(\|h_1\|_{\mathbb{H}}) \leq G(\sqrt{\|h_1\|_{\mathbb{H}}^2 + \|h^\perp\|_{\mathbb{H}}^2}) = G(\|h\|_{\mathbb{H}})$. By the reproducing property, for all $i \in [m]$, $h(x_i) = \langle h, K(x_i, \cdot)\rangle = \langle h_1, K(x_i, \cdot)\rangle = h_1(x_i)$. Thus, $L\big(h(x_1), \ldots, h(x_m)\big) = L\big(h_1(x_1), \ldots, h_1(x_m)\big)$ and $F(h_1) \leq F(h)$. This proves the first part of the theorem. If G is further increasing, then $F(h_1) < F(h)$ when $\|h^\perp\|_{\mathbb{H}} > 0$ and any solution of the optimization problem must be in $\mathbb{H}_1$. □

6.3.3 Learning guarantees

Here, we present general learning guarantees for hypothesis sets based on PDS kernels, which hold in particular for SVMs combined with PDS kernels.

The following theorem gives a general bound on the empirical Rademacher complexity of kernel-based hypotheses with bounded norm, that is a hypothesis set of the form $\mathcal{H} = \{h \in \mathbb{H}\colon \|h\|_{\mathbb{H}} \leq \Lambda\}$, for some $\Lambda \geq 0$, where $\mathbb{H}$ is the RKHS associated to a kernel K. By the reproducing property, any $h \in \mathcal{H}$ is of the form $x \mapsto \langle h, K(x, \cdot)\rangle = \langle h, \Phi(x)\rangle$ with $\|h\|_{\mathbb{H}} \leq \Lambda$, where Φ is a feature mapping associated to K, that is of the form $x \mapsto \langle \mathbf{w}, \Phi(x)\rangle$ with $\|\mathbf{w}\|_{\mathbb{H}} \leq \Lambda$.

Theorem 6.12 (Rademacher complexity of kernel-based hypotheses) *Let $K \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel and let $\Phi \colon \mathcal{X} \to \mathbb{H}$ be a feature mapping associated to K. Let $S \subseteq \{x \colon K(x,x) \leq r^2\}$ be a sample of size m, and let $\mathcal{H} = \{x \mapsto \langle \mathbf{w}, \Phi(x) \rangle : \|\mathbf{w}\|_{\mathbb{H}} \leq \Lambda\}$ for some $\Lambda \geq 0$. Then*

$$\widehat{\mathfrak{R}}_S(\mathcal{H}) \leq \frac{\Lambda\sqrt{\mathrm{Tr}[\mathbf{K}]}}{m} \leq \sqrt{\frac{r^2\Lambda^2}{m}}. \tag{6.16}$$

Proof: The proof steps are as follows:

$$\begin{aligned}
\widehat{\mathfrak{R}}_S(\mathcal{H}) &= \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \Big[\sup_{\|\mathbf{w}\| \leq \Lambda} \Big\langle \mathbf{w}, \sum_{i=1}^m \sigma_i \Phi(x_i) \Big\rangle \Big] \\
&= \frac{\Lambda}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \Big[\Big\| \sum_{i=1}^m \sigma_i \Phi(x_i) \Big\|_{\mathbb{H}} \Big] && \text{(Cauchy-Schwarz, eq. case)} \\
&\leq \frac{\Lambda}{m} \Big[\mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \Big[\Big\| \sum_{i=1}^m \sigma_i \Phi(x_i) \Big\|_{\mathbb{H}}^2 \Big] \Big]^{1/2} && \text{(Jensen's ineq.)} \\
&= \frac{\Lambda}{m} \Big[\mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \Big[\sum_{i=1}^m \|\Phi(x_i)\|_{\mathbb{H}}^2 \Big] \Big]^{1/2} && (i \neq j \Rightarrow \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}[\sigma_i \sigma_j] = 0) \\
&= \frac{\Lambda}{m} \Big[\mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \Big[\sum_{i=1}^m K(x_i, x_i) \Big] \Big]^{1/2} \\
&= \frac{\Lambda\sqrt{\mathrm{Tr}[\mathbf{K}]}}{m} \leq \sqrt{\frac{r^2\Lambda^2}{m}}.
\end{aligned}$$

The initial equality holds by definition of the empirical Rademacher complexity (definition 3.1). The first inequality is due to the Cauchy-Schwarz inequality and $\|\mathbf{w}\|_{\mathbb{H}} \leq \Lambda$. The following inequality results from Jensen's inequality (theorem B.20) applied to the concave function $\sqrt{\cdot}$. The subsequent equality is a consequence of $\mathbb{E}_{\boldsymbol{\sigma}}[\sigma_i \sigma_j] = \mathbb{E}_{\boldsymbol{\sigma}}[\sigma_i] \, \mathbb{E}_{\boldsymbol{\sigma}}[\sigma_j] = 0$ for $i \neq j$, since the Rademacher variables σ_i and σ_j are independent. The statement of the theorem then follows by noting that $\mathrm{Tr}[\mathbf{K}] \leq mr^2$. □

The theorem indicates that the trace of the kernel matrix is an important quantity for controlling the complexity of hypothesis sets based on kernels. Observe that by the Khintchine-Kahane inequality (D.24), the empirical Rademacher complexity $\widehat{\mathfrak{R}}_S(\mathcal{H}) = \frac{\Lambda}{m} \mathbb{E}_{\boldsymbol{\sigma}}[\| \sum_{i=1}^m \sigma_i \Phi(x_i) \|_{\mathbb{H}}]$ can also be lower bounded by $\frac{1}{\sqrt{2}} \frac{\Lambda\sqrt{\mathrm{Tr}[\mathbf{K}]}}{m}$, which only differs from the upper bound found by the constant $\frac{1}{\sqrt{2}}$. Also, note that if $K(x,x) \leq r^2$ for all $x \in \mathcal{X}$, then the inequalities 6.16 hold for all samples S.

The bound of theorem 6.12 or the inequalities 6.16 can be plugged into any of the Rademacher complexity generalization bounds presented in the previous chapters. In particular, in combination with theorem 5.8, they lead directly to the following margin bound similar to that of corollary 5.11.

Corollary 6.13 (Margin bounds for kernel-based hypotheses) *Let $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel with $r^2 = \sup_{x \in \mathcal{X}} K(x, x)$. Let $\Phi\colon \mathcal{X} \to \mathbb{H}$ be a feature mapping associated to K and let $\mathcal{H} = \{\mathbf{x} \mapsto \mathbf{w} \cdot \Phi(x) \colon \|\mathbf{w}\|_{\mathbb{H}} \leq \Lambda\}$ for some $\Lambda \geq 0$. Fix $\rho > 0$. Then, for any $\delta > 0$, each of the following statements holds with probability at least $1 - \delta$ for any $h \in \mathcal{H}$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + 2\sqrt{\frac{r^2\Lambda^2/\rho^2}{m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \tag{6.17}$$

$$R(h) \leq \widehat{R}_{S,\rho}(h) + 2\frac{\sqrt{\mathrm{Tr}[\mathbf{K}]\Lambda^2/\rho^2}}{m} + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}\,. \tag{6.18}$$

6.4 Negative definite symmetric kernels

Often in practice, a natural distance or metric is available for the learning task considered. This metric could be used to define a similarity measure. As an example, Gaussian kernels have the form $\exp(-d^2)$, where d is a metric for the input vector space. Several natural questions arise such as: what other PDS kernels can we construct from a metric in a Hilbert space? What technical condition should d satisfy to guarantee that $\exp(-d^2)$ is PDS? A natural mathematical definition that helps address these questions is that of *negative definite symmetric (NDS) kernels*.

Definition 6.14 (Negative definite symmetric (NDS) kernels) *A kernel $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be* negative-definite symmetric (NDS) *if it is symmetric and if for all $\{x_1, \ldots, x_m\} \subseteq \mathcal{X}$ and $\mathbf{c} \in \mathbb{R}^{m \times 1}$ with $\mathbf{1}^\top \mathbf{c} = 0$, the following holds:*

$$\mathbf{c}^\top \mathbf{K} \mathbf{c} \leq 0.$$

Clearly, if K is PDS, then $-K$ is NDS, but the converse does not hold in general. The following gives a standard example of an NDS kernel.

Example 6.15 (Squared distance — NDS kernel) The squared distance $(x, x') \mapsto \|x' - x\|^2$ in $\mathbb{R}^N$ defines an NDS kernel. Indeed, let $\mathbf{c} \in \mathbb{R}^{m \times 1}$ with $\sum_{i=1}^m c_i = 0$. Then, for any $\{x_1, \ldots, x_m\} \subseteq \mathcal{X}$, we can write

$$\begin{aligned}
\sum_{i,j=1}^{m} c_i c_j ||\mathbf{x}_i - \mathbf{x}_j||^2 &= \sum_{i,j=1}^{m} c_i c_j (\|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - 2\mathbf{x}_i \cdot \mathbf{x}_j) \\
&= \sum_{i,j=1}^{m} c_i c_j (\|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2) - 2 \sum_{i=1}^{m} c_i \mathbf{x}_i \cdot \sum_{j=1}^{m} c_j \mathbf{x}_j \\
&= \sum_{i,j=1}^{m} c_i c_j (\|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2) - 2 \Big\| \sum_{i=1}^{m} c_i \mathbf{x}_i \Big\|^2 \\
&\leq \sum_{i,j=1}^{m} c_i c_j (\|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2) \\
&= \Big(\sum_{j=1}^{m} c_j\Big)\Big(\sum_{i=1}^{m} c_i (\|\mathbf{x}_i\|^2\Big) + \Big(\sum_{i=1}^{m} c_i\Big)\Big(\sum_{j=1}^{m} c_j \|\mathbf{x}_j\|^2\Big) = 0.
\end{aligned}$$

The next theorems show connections between NDS and PDS kernels. These results provide another series of tools for designing PDS kernels.

Theorem 6.16 *Let K' be defined for any x_0 by*

$$K'(x, x') = K(x, x_0) + K(x', x_0) - K(x, x') - K(x_0, x_0)$$

for all $x, x' \in \mathcal{X}$. Then K is NDS iff K' is PDS.

Proof: Assume that K' is PDS and define K such that for any x_0 we have $K(x, x') = K(x, x_0) + K(x_0, x') - K(x_0, x_0) - K'(x, x')$. Then for any $\mathbf{c} \in \mathbb{R}^m$ such that $\mathbf{c}^\top \mathbf{1} = 0$ and any set of points $(x_1, \ldots, x_m) \in \mathcal{X}^m$ we have

$$\begin{aligned}
\sum_{i,j=1}^{m} c_i c_j K(x_i, x_j) &= \Big(\sum_{i=1}^{m} c_i K(x_i, x_0)\Big)\Big(\sum_{j=1}^{m} c_j\Big) + \Big(\sum_{i=1}^{m} c_i\Big)\Big(\sum_{j=1}^{m} c_j K(x_0, x_j)\Big) \\
&- \Big(\sum_{i=1}^{m} c_i\Big)^2 K(x_0, x_0) - \sum_{i,j=1}^{m} c_i c_j K'(x_i, x_j) = - \sum_{i,j=1}^{m} c_i c_j K'(x_i, x_j) \leq 0\,.
\end{aligned}$$

which proves K is NDS.

Now, assume K is NDS and define K' for any x_0 as above. Then, for any $\mathbf{c} \in \mathbb{R}^m$, we can define $c_0 = -\mathbf{c}^\top \mathbf{1}$ and the following holds by the NDS property for any points $(x_1, \ldots, x_m) \in \mathcal{X}^m$ as well as x_0 defined previously: $\sum_{i,j=0}^{m} c_i c_j K(x_i, x_j) \leq 0$. This implies that

$$\begin{aligned}
&\Big(\sum_{i=0}^{m} c_i K(x_i, x_0)\Big)\Big(\sum_{j=0}^{m} c_j\Big) + \Big(\sum_{i=0}^{m} c_i\Big)\Big(\sum_{j=0}^{m} c_j K(x_0, x_j)\Big) \\
&\qquad - \Big(\sum_{i=0}^{m} c_i\Big)^2 K(x_0, x_0) - \sum_{i,j=0}^{m} c_i c_j K'(x_i, x_j) = - \sum_{i,j=0}^{m} c_i c_j K'(x_i, x_j) \leq 0\,,
\end{aligned}$$

which implies $2\sum_{i,j=1}^m c_i c_j K'(x_i, x_j) \geq -2c_0 \sum_{i=0}^m c_i K'(x_i, x_0) + c_0^2 K'(x_0, x_0) = 0$. The equality holds since $\forall x \in \mathcal{X}, K'(x, x_0) = 0$. □

This theorem is useful in showing other connections, such the following theorems, which are left as exercises (see exercises 6.17 and 6.18).

Theorem 6.17 *Let $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric kernel. Then, K is NDS iff $\exp(-tK)$ is a PDS kernel for all $t > 0$.*

The theorem provides another proof that Gaussian kernels are PDS: as seen earlier (Example 6.15), the squared distance $(x, x') \mapsto \|x - x'\|^2$ in $\mathbb{R}^N$ is NDS, thus $(x, x') \mapsto \exp(-t\|x - x'\|^2)$ is PDS for all $t > 0$.

Theorem 6.18 *Let $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be an NDS kernel such that for all $x, x' \in \mathcal{X}, K(x, x') = 0$ iff $x = x'$. Then, there exists a Hilbert space $\mathbb{H}$ and a mapping $\Phi\colon \mathcal{X} \to \mathbb{H}$ such that for all $x, x' \in \mathcal{X}$,*

$$K(x, x') = \|\Phi(x) - \Phi(x')\|^2.$$

Thus, under the hypothesis of the theorem, $\sqrt{K}$ defines a metric.

This theorem can be used to show that the kernel $(x, x') \mapsto \exp(-|x - x'|^p)$ in $\mathbb{R}$ is not PDS for $p > 2$. Otherwise, for any $t > 0$, $\{x_1, \ldots, x_m\} \subseteq \mathcal{X}$ and $\mathbf{c} \in \mathbb{R}^{m \times 1}$, we would have:

$$\sum_{i,j=1}^m c_i c_j e^{-t|x_i - x_j|^p} = \sum_{i,j=1}^m c_i c_j e^{-|t^{1/p} x_i - t^{1/p} x_j|^p} \geq 0.$$

This would imply that $(x, x') \mapsto |x - x'|^p$ is NDS for $p > 2$, which can be proven (via theorem 6.18) not to be valid.

6.5 Sequence kernels

The examples given in the previous sections, including the commonly used polynomial or Gaussian kernels, were all for PDS kernels over vector spaces. In many learning tasks found in practice, the input space $\mathcal{X}$ is not a vector space. The examples to classify in practice could be protein sequences, images, graphs, parse trees, finite automata, or other discrete structures which may not be directly given as vectors. PDS kernels provide a method for extending algorithms such as SVMs originally designed for a vectorial space to the classification of such objects. But, how can we define PDS kernels for these structures?

This section will focus on the specific case of *sequence kernels*, that is, kernels for sequences or strings. PDS kernels can be defined for other discrete structures in somewhat similar ways. Sequence kernels are particularly relevant to learning algorithms applied to computational biology or natural language processing, which are both important applications.

How can we define PDS kernels for sequences, which are similarity measures for sequences? One idea consists of declaring two sequences, e.g., two documents or two biosequences, as similar when they share common substrings or subsequences. One example could be the kernel between two sequences defined by the sum of the product of the counts of their common substrings. But which substrings should be used in that definition? Most likely, we would need some flexibility in the definition of the matching substrings. For computational biology applications, for example, the match could be imperfect. Thus, we may need to consider some number of mismatches, possibly gaps, or wildcards. More generally, we might need to allow various substitutions and might wish to assign different weights to common substrings to emphasize some matching substrings and deemphasize others.

As can be seen from this discussion, there are many different possibilities and we need a general framework for defining such kernels. In the following, we will introduce a general framework for sequence kernels, *rational kernels*, which will include all the kernels considered in this discussion. We will also describe a general and efficient algorithm for their computation and will illustrate them with some examples.

The definition of these kernels relies on that of *weighted transducers*. Thus, we start with the definition of these devices as well as some relevant algorithms.

6.5.1 Weighted transducers

Sequence kernels can be effectively represented and computed using *weighted transducers*. In the following definition, let Σ denote a finite input alphabet, Δ a finite output alphabet, and ϵ the *empty string* or null label, whose concatenation with any string leaves it unchanged.

Definition 6.19 *A* weighted transducer *T is a 7-tuple $T = (\Sigma, \Delta, Q, I, F, E, \rho)$ where Σ is a finite input alphabet, Δ a finite output alphabet, Q is a finite set of states, $I \subseteq Q$ the set of initial states, $F \subseteq Q$ the set of final states, E a finite multiset of transitions elements of $Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{R} \times Q$, and $\rho : F \to \mathbb{R}$ a final weight function mapping F to $\mathbb{R}$. The* size *of transducer T is the sum of its number of states and transitions and is denoted by $|T|$.*[7]

Thus, weighted transducers are finite automata in which each transition is labeled with both an input and an output label and carries some real-valued weight. Figure 6.4 shows an example of a weighted finite-state transducer. In this figure, the input and output labels of a transition are separated by a colon delimiter, and the weight is indicated after the slash separator. The initial states are represented by

[7] A multiset in the definition of the transitions is used to allow for the presence of several transitions from a state p to a state q with the same input and output label, and even the same weight, which may occur as a result of various operations.

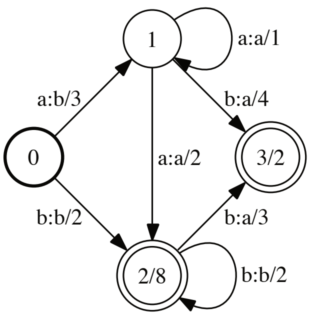


Figure 6.4
Example of weighted transducer.

a bold circle and final states by double circles. The final weight $\rho[q]$ at a final state q is displayed after the slash.

The input label of a path π is a string element of Σ^* obtained by concatenating input labels along π. Similarly, the output label of a path π is obtained by concatenating output labels along π. A path from an initial state to a final state is an *accepting path*. The weight of an accepting path is obtained by multiplying the weights of its constituent transitions and the weight of the final state of the path.

A weighted transducer defines a mapping from $\Sigma^* \times \Delta^*$ to $\mathbb{R}$. The weight associated by a weighted transducer T to a pair of strings $(x, y) \in \Sigma^* \times \Delta^*$ is denoted by $T(x, y)$ and is obtained by summing the weights of all accepting paths with input label x and output label y. For example, the transducer of figure 6.4 associates to the pair (aab, baa) the weight $3 \times 1 \times 4 \times 2 + 3 \times 2 \times 3 \times 2$, since there is a path with input label aab and output label baa and weight $3 \times 1 \times 4 \times 2$, and another one with weight $3 \times 2 \times 3 \times 2$.

The sum of the weights of all accepting paths of an acyclic transducer, that is a transducer T with no cycle, can be computed in linear time, that is $O(|T|)$, using a general *shortest-distance* or forward-backward algorithm. These are simple algorithms, but a detailed description would require too much of a digression from the main topic of this chapter.

Composition An important operation for weighted transducers is *composition*, which can be used to combine two or more weighted transducers to form more complex weighted transducers. As we shall see, this operation is useful for the creation and computation of sequence kernels. Its definition follows that of composition of relations. Given two weighted transducers $T_1 = (\Sigma, \Delta, Q_1, I_1, F_1, E_1, \rho_1)$ and $T_2 = (\Delta, \Omega, Q_2, I_2, F_2, E_2, \rho_2)$, the result of the composition of T_1 and T_2 is a weighted transducer denoted by $T_1 \circ T_2$ and defined for all $x \in \Sigma^*$ and $y \in \Omega^*$ by

$$(T_1 \circ T_2)(x, y) = \sum_{z \in \Delta^*} T_1(x, z) \cdot T_2(z, y), \tag{6.19}$$

where the sum runs over all strings z over the alphabet Δ. Thus, composition is similar to matrix multiplication with infinite matrices.

There exists a general and efficient algorithm to compute the composition of two weighted transducers. In the absence of ϵs on the input side of T_1 or the output side of T_2, the states of $T_1 \circ T_2 = (\Sigma, \Delta, Q, I, F, E, \rho)$ can be identified with pairs made of a state of T_1 and a state of T_2, $Q \subseteq Q_1 \times Q_2$. Initial states are those obtained by pairing initial states of the original transducers, $I = I_1 \times I_2$, and similarly final states are defined by $F = Q \cap (F_1 \times F_2)$. The final weight at a state $(q_1, q_2) \in F_1 \times F_2$ is $\rho(q) = \rho_1(q_1)\rho_2(q_2)$, that is the product of the final weights at q_1 and q_2. Transitions are obtained by matching a transition of T_1 with one of T_2 from appropriate transitions of T_1 and T_2:

$$E = \biguplus_{\substack{(q_1, a, b, w_1, q_2) \in E_1 \\ (q_1', b, c, w_2, q_2') \in E_2}} \Big\{ \Big((q_1, q_1'), a, c, w_1 \otimes w_2, (q_2, q_2') \Big) \Big\}.$$

Here, $\uplus$ denotes the standard join operation of multisets as in $\{1, 2\} \uplus \{1, 3\} = \{1, 1, 2, 3\}$, to preserve the multiplicity of the transitions.

In the worst case, all transitions of T_1 leaving a state q_1 match all those of T_2 leaving state q_1', thus the space and time complexity of composition is quadratic: $O(|T_1||T_2|)$. In practice, such cases are rare and composition is very efficient. Figure 6.5 illustrates the algorithm in a particular case.

As illustrated by figure 6.6, when T_1 admits output ϵ labels or T_2 input ϵ labels, the algorithm just described may create redundant ϵ-paths, which would lead to an incorrect result. The weight of the matching paths of the original transducers would be counted p times, where p is the number of redundant paths in the result of composition. To avoid with this problem, all but one ϵ-path must be filtered out of the composite transducer. Figure 6.6 indicates in boldface one possible choice for that path, which in this case is the shortest. Remarkably, that filtering mechanism itself can be encoded as a finite-state transducer F (figure 6.6b).

To apply that filter, we need to first augment T_1 and T_2 with auxiliary symbols that make the semantics of ϵ explicit: let $\tilde{T}_1$ ($\tilde{T}_2$) be the weighted transducer obtained from T_1 (respectively T_2) by replacing the output (respectively input) ϵ labels with ϵ_2 (respectively ϵ_1) as illustrated by figure 6.6. Thus, matching with the symbol ϵ_1 corresponds to remaining at the same state of T_1 and taking a transition of T_2 with input ϵ. ϵ_2 can be described in a symmetric way. The filter transducer F disallows a matching (ϵ_2, ϵ_2) immediately after (ϵ_1, ϵ_1) since this can be done

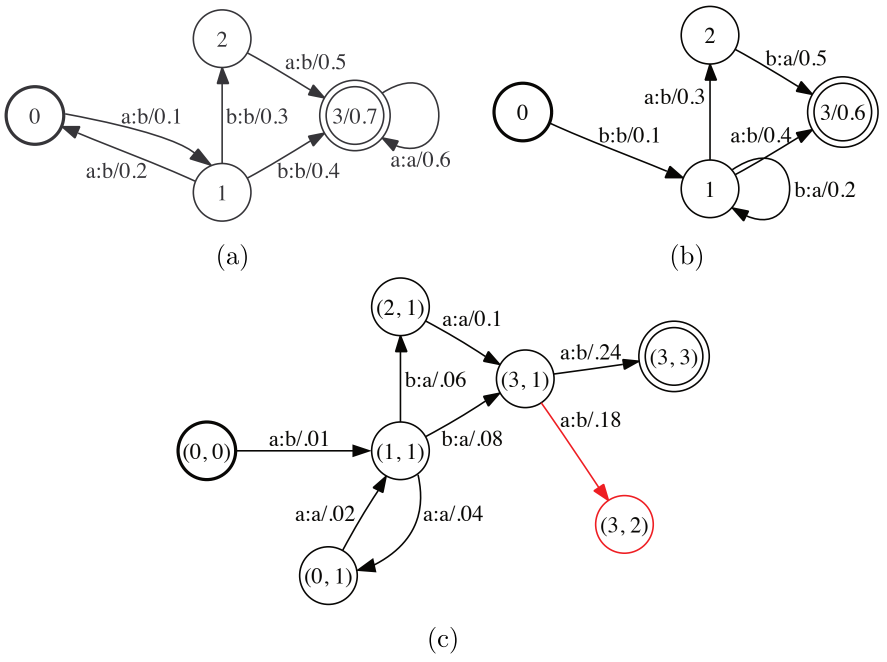


Figure 6.5
(a) Weighted transducer T_1. (b) Weighted transducer T_2. (c) Result of composition of T_1 and T_2, $T_1 \circ T_2$. Some states might be constructed during the execution of the algorithm that are not *co-accessible*, that is, they do not admit a path to a final state, e.g., $(3, 2)$. Such states and the related transitions (in red) can be removed by a trimming (or connection) algorithm in linear time.

instead via (ϵ_2, ϵ_1). By symmetry, it also disallows a matching (ϵ_1, ϵ_1) immediately after (ϵ_2, ϵ_2). In the same way, a matching (ϵ_1, ϵ_1) immediately followed by (ϵ_2, ϵ_1) is not permitted by the filter F since a path via the matchings $(\epsilon_2, \epsilon_1)(\epsilon_1, \epsilon_1)$ is possible. Similarly, $(\epsilon_2, \epsilon_2)(\epsilon_2, \epsilon_1)$ is ruled out. It is not hard to verify that the filter transducer F is precisely a finite automaton over pairs accepting the complement of the language

$$L = \sigma^*((\epsilon_1, \epsilon_1)(\epsilon_2, \epsilon_2) + (\epsilon_2, \epsilon_2)(\epsilon_1, \epsilon_1) + (\epsilon_1, \epsilon_1)(\epsilon_2, \epsilon_1) + (\epsilon_2, \epsilon_2)(\epsilon_2, \epsilon_1))\sigma^*,$$

where $\sigma = \{(\epsilon_1, \epsilon_1), (\epsilon_2, \epsilon_2), (\epsilon_2, \epsilon_1), x\}$. Thus, the filter F guarantees that exactly one ϵ-path is allowed in the composition of each ϵ sequences. To obtain the correct result of composition, it suffices then to use the ϵ-free composition algorithm already described and compute

$$\tilde{T}_1 \circ F \circ \tilde{T}_2. \tag{6.20}$$

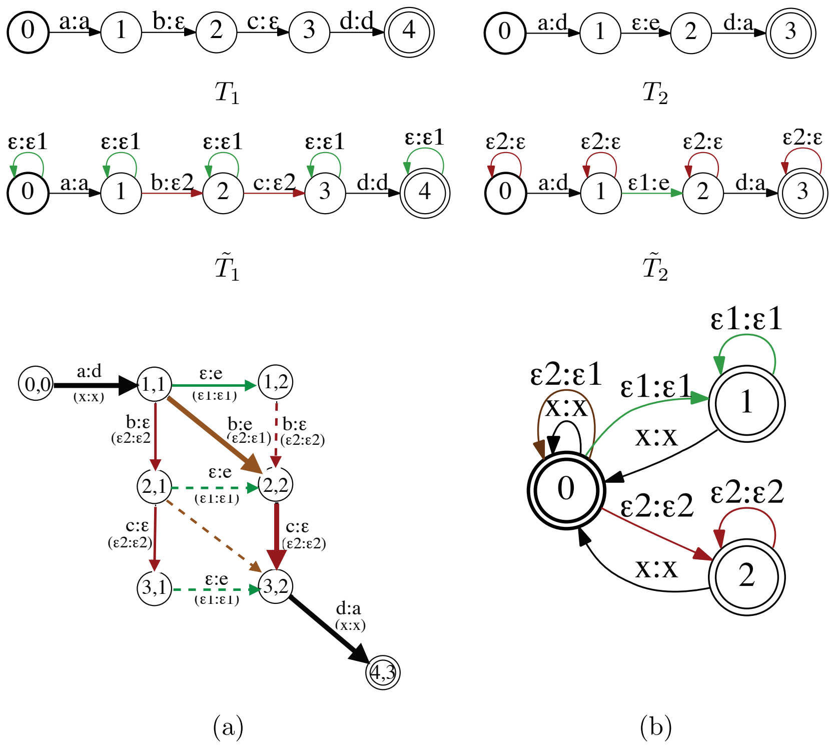


Figure 6.6
Redundant ϵ-paths in composition. All transition and final weights are equal to one. (a) A straightforward generalization of the ϵ-free case would generate all the paths from $(1, 1)$ to $(3, 2)$ when composing T_1 and T_2 and produce an incorrect results in non-idempotent semirings. (b) Filter transducer F. The shorthand x is used to represent an element of Σ.

Indeed, the two compositions in $\tilde{T}_1 \circ F \circ \tilde{T}_2$ no longer involve ϵs. Since the size of the filter transducer F is constant, the complexity of general composition is the same as that of ϵ-free composition, that is $O(|T_1||T_2|)$. In practice, the augmented transducers $\tilde{T}_1$ and $\tilde{T}_2$ are not explicitly constructed, instead the presence of the auxiliary symbols is simulated. Further filter optimizations help limit the number of non-coaccessible states created, for example, by examining more carefully the case of states with only outgoing non-ϵ-transitions or only outgoing ϵ-transitions.

6.5.2 Rational kernels

The following establishes a general framework for the definition of sequence kernels.

Definition 6.20 (Rational kernels) *A kernel $K\colon \Sigma^* \times \Sigma^* \to \mathbb{R}$ is said to be* rational *if it coincides with the mapping defined by some weighted transducer U: $\forall x, y \in \Sigma^*, K(x, y) = U(x, y)$.*

Note that we could have instead adopted a more general definition: instead of using weighted transducers, we could have used more powerful sequence mappings such as *algebraic transductions*, which are the functional counterparts of context-free languages, or even more powerful ones. However, an essential need for kernels is an efficient computation, and more complex definitions would lead to substantially more costly computational complexities for kernel computation. For rational kernels, there exists a general and efficient computation algorithm.

Computation We will assume that the transducer U defining a rational kernel K does not admit any ϵ-cycle with non-zero weight, otherwise the kernel value is infinite for all pairs. For any sequence x, let T_x denote a weighted transducer with just one accepting path whose input and output labels are both x and its weight equal to one. T_x can be straightforwardly constructed from x in linear time $O(|x|)$. Then, for any $x, y \in \Sigma^*$, $U(x, y)$ can be computed by the following two steps:

1. Compute $V = T_x \circ U \circ T_y$ using the composition algorithm in time $O(|U||T_x||T_y|)$.
2. Compute the sum of the weights of all accepting paths of V using a general shortest-distance algorithm in time $O(|V|)$.

By definition of composition, V is a weighted transducer whose accepting paths are precisely those accepting paths of U that have input label x and output label y. The second step computes the sum of the weights of these paths, that is, exactly $U(x, y)$. Since U admits no ϵ-cycle, V is acyclic, and this step can be performed in linear time. The overall complexity of the algorithm for computing $U(x, y)$ is then in $O(|U||T_x||T_y|)$. Since U is fixed for a rational kernel K and $|T_x| = O(|x|)$ for any x, this shows that the kernel values can be obtained in quadratic time $O(|x||y|)$. For some specific weighted transducers U, the computation can be more efficient, for example in $O(|x| + |y|)$ (see exercise 6.20).

PDS rational kernels For any transducer T, let T^{-1} denote the *inverse* of T, that is the transducer obtained from T by swapping the input and output labels of every transition. For all x, y, we have $T^{-1}(x, y) = T(y, x)$. The following theorem gives a general method for constructing a PDS rational kernel from an arbitrary weighted transducer.

Theorem 6.21 *For any weighted transducer $T = (\Sigma, \Delta, Q, I, F, E, \rho)$, the function $K = T \circ T^{-1}$ is a PDS rational kernel.*

Proof: By definition of composition and the inverse operation, for all $x, y \in \Sigma^*$,

$$K(x, y) = \sum_{z \in \Delta^*} T(x, z)\, T(y, z).$$

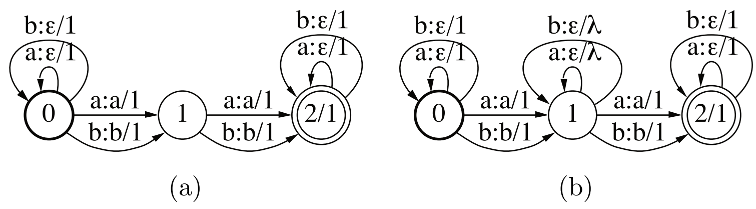


Figure 6.7
(a) Transducer T_{bigram} defining the bigram kernel $T_{\text{bigram}} \circ T^{-1}_{\text{bigram}}$ for $\Sigma = \{a, b\}$. (b) Transducer $T_{\text{gappy\_bigram}}$ defining the gappy bigram kernel $T_{\text{gappy\_bigram}} \circ T^{-1}_{\text{gappy\_bigram}}$ with gap penalty $\lambda \in (0, 1)$.

K is the pointwise limit of the kernel sequence $(K_n)_{n \geq 0}$ defined by:

$$\forall n \in \mathbb{N}, \forall x, y \in \Sigma^*, \quad K_n(x, y) = \sum_{|z| \leq n} T(x, z)\, T(y, z),$$

where the sum runs over all sequences in Δ^* of length at most n. K_n is PDS since its corresponding kernel matrix $\mathbf{K}_n$ for any sample $(x_1, \ldots, x_m)$ is SPSD. This can be see form the fact that $\mathbf{K}_n$ can be written as $\mathbf{K}_n = \mathbf{A}\mathbf{A}^\top$ with $\mathbf{A} = (K_n(x_i, z_j))_{i \in [m], j \in [N]}$, where $z_1, \ldots, z_N$ is some arbitrary enumeration of the set of strings in Σ^* with length at most n. Thus, K is PDS as the pointwise limit of the sequence of PDS kernels $(K_n)_{n \in \mathbb{N}}$. □

The sequence kernels commonly used in computational biology, natural language processing, computer vision, and other applications are all special instances of rational kernels of the form $T \circ T^{-1}$. All of these kernels can be computed efficiently using the same general algorithm for the computational of rational kernels presented in the previous paragraph. Since the transducer $U = T \circ T^{-1}$ defining such PDS rational kernels has a specific form, there are different options for the computation of the composition $T_x \circ U \circ T_y$:

- compute $U = T \circ T^{-1}$ first, then $V = T_x \circ U \circ T_y$;
- compute $V_1 = T_x \circ T$ and $V_2 = T_y \circ T$ first, then $V = V_1 \circ V_2^{-1}$;
- compute first $V_1 = T_x \circ T$, then $V_2 = V_1 \circ T^{-1}$, then $V = V_2 \circ T_y$, or the similar series of operations with x and y permuted.

All of these methods lead to the same result after computation of the sum of the weights of all accepting paths, and they all have the same worst-case complexity. However, in practice, due to the sparsity of intermediate compositions, there may be substantial differences between their time and space computational costs. An alternative method based on an *n-way composition* can further lead to significantly more efficient computations.

Example 6.22 (Bigram and gappy bigram sequence kernels) Figure 6.7a shows a weighted transducer T_{bigram} defining a common sequence kernel, the *bigram sequence kernel*, for the specific case of an alphabet reduced to $\Sigma = \{a, b\}$. The bigram kernel associates to any two sequences x and y the sum of the product of the counts of all bigrams in x and y. For any sequence $x \in \Sigma^*$ and any bigram $z \in \{aa, ab, ba, bb\}$, $T_{\text{bigram}}(x, z)$ is exactly the number of occurrences of the bigram z in x. Thus, by definition of composition and the inverse operation, $T_{\text{bigram}} \circ T^{-1}_{\text{bigram}}$ computes exactly the bigram kernel.

Figure 6.7b shows a weighted transducer $T_{\text{gappy\_bigram}}$ defining the so-called *gappy bigram kernel*. The gappy bigram kernel associates to any two sequences x and y the sum of the product of the counts of all gappy bigrams in x and y penalized by the length of their *gaps*. Gappy bigrams are sequences of the form aua, aub, bua, or bub, where $u \in \Sigma^*$ is called the gap. The count of a gappy bigram is multiplied by $\lambda^{|u|}$ for some fixed $\lambda \in (0, 1)$ so that gappy bigrams with longer gaps contribute less to the definition of the similarity measure. While this definition could appear to be somewhat complex, figure 6.7 shows that $T_{\text{gappy\_bigram}}$ can be straightforwardly derived from T_{bigram}. The graphical representation of rational kernels helps understanding or modifying their definition.

Counting transducers The definition of most sequence kernels is based on the counts of some common patterns appearing in the sequences. In the examples just examined, these were bigrams or gappy bigrams. There exists a simple and general method for constructing a weighted transducer counting the number of occurrences of patterns and using them to define PDS rational kernels. Let $\mathcal{X}$ be a finite automaton representing the set of patterns to count. In the case of bigram kernels with $\Sigma = \{a, b\}$, $\mathcal{X}$ would be an automaton accepting exactly the set of strings $\{aa, ab, ba, bb\}$. Then, the weighted transducer of figure 6.8 can be used to compute exactly the number of occurrences of each pattern accepted by $\mathcal{X}$.

Theorem 6.23 *For any $x \in \Sigma^*$ and any sequence z accepted by $\mathcal{X}$, $T_{count}(x, z)$ is the number of occurrences of z in x.*

Proof: Let $x \in \Sigma^*$ be an arbitrary sequence and let z be a sequence accepted by $\mathcal{X}$. Since all accepting paths of T_{count} have weight one, $T_{\text{count}}(x, z)$ is equal to the number of accepting paths in T_{count} with input label x and output z.

Now, an accepting path π in T_{count} with input x and output z can be decomposed as $\pi = \pi_0\, \pi_{01}\, \pi_1$, where π_0 is a path through the loops of state 0 with input label some prefix x_0 of x and output label ϵ, π_{01} an accepting path from 0 to 1 with input and output labels equal to z, and π_1 a path through the self-loops of state 1 with input label a suffix x_1 of x and output ϵ. Thus, the number of such paths is exactly

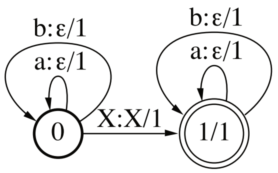


Figure 6.8
Counting transducer T_{count} for $\Sigma = \{a, b\}$. The "transition" $X : X/1$ stands for the weighted transducer created from the automaton $\mathcal{X}$ by adding to each transition an output label identical to the existing label, and by making all transition and final weights equal to one.

the number of distinct ways in which we can write sequence x as $x = x_0 z x_1$, which is exactly the number of occurrences of z in x. $\square$

The theorem provides a very general method for constructing PDS rational kernels $T_{\text{count}} \circ T_{\text{count}}^{-1}$ that are based on counts of some patterns that can be defined via a finite automaton, or equivalently a regular expression. Figure 6.8 shows the transducer for the case of an input alphabet reduced to $\Sigma = \{a, b\}$. The general case can be obtained straightforwardly by augmenting states 0 and 1 with other self-loops using other symbols than a and b. In practice, a lazy evaluation can be used to avoid the explicit creation of these transitions for all alphabet symbols and instead creating them on-demand based on the symbols found in the input sequence x. Finally, one can assign different weights to the patterns counted to emphasize or deemphasize some, as in the case of gappy bigrams. This can be done simply by changing the transitions weight or final weights of the automaton $\mathcal{X}$ used in the definition of T_{count}.

6.6 Approximate kernel feature maps

In the previous sections, we have seen the benefits that kernel methods can provide by implicitly and efficiently mapping a learning problem from the input space $\mathcal{X}$ to a richer feature space $\mathbb{H}$. One potential drawback when using kernel methods, is that the kernel function needs to be evaluated on all pairs of points in the training set. If this set contains a very large number of instances, then the $O(m^2)$ cost in memory and $O(m^2 C_K)$ cost in computation, where C_K is the cost of a single kernel function evaluation, may be prohibitive. Another consideration is the cost of making predictions with a trained model. Evaluating the kernelized function $h(x) = \sum_{i=1}^{m} \alpha_i K(x_i, x) + b$ requires $O(m)$ storage and $O(mC_K)$ computation cost (the exact amount of storage and number of operations depends on the number of support vectors).

Note that if we use explicit feature vectors $\mathbf{x} \in \mathbb{R}^N$, then the primal formulation of the SVM problem can be used for training. The primal formulation incurs only an $O(Nm)$ storage cost and evaluation requires only $O(N)$ storage and computation

Table 6.1
Examples of normalized shift-invariant kernels (defined over $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^N$) and their corresponding densities (defined over $\boldsymbol{\omega} \in \mathbb{R}^N$).

	$G(\mathbf{x}-\mathbf{x}')$	$p(\boldsymbol{\omega})$
Gaussian	$\exp\big(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2}\big)$	$(2\pi)^{\frac{-D}{2}}\exp\big(-\frac{\|\boldsymbol{\omega}\|^2}{2}\big)$
Laplacian	$\exp\big(-\|\mathbf{x}-\mathbf{x}'\|_1\big)$	$\prod_{i=1}^N \frac{1}{\pi(1+\omega_i^2)}$
Cauchy	$\prod_{i=1}^N \frac{2}{1+(x_i-x_i')^2}$	$\exp\big(-\|\boldsymbol{\omega}\|_1\big)$

time: $h(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}+b$. However, these observations are only useful if $N < m$, which is likely not the case when considering the explicit feature maps $\Phi(x)$ induced by a kernel function. For example, given an input feature space of dimension N, the dimension of the kernel feature map for a polynomial kernel of degree d is $O(N^d)$. In the case of Gaussian kernels the explicit feature map dimension is infinite. So clearly using explicit kernel feature maps in general is not possible and again emphasizes that using kernel functions to compute inner products implicitly is crucial.

In this section we show that a compromise is possible by constructing *approximate kernel feature maps*. These are feature maps with a user-specified dimension D, $\Psi(x) \in \mathbb{R}^D$, which guarantee $\Psi(x)\cdot\Psi(x') \approx K(x,x')$ when using a sufficiently large dimension D. To begin, we state a classical result from the field of harmonic analysis.

Theorem 6.24 (Bochner's theorem) *A continuous kernel of the form $K(x,x') = G(x-x')$ defined over a locally compact set $\mathcal{X}$ is positive definite if and only if G is the Fourier transform of a non-negative measure. That is,*

$$G(x) = \int_{\mathcal{X}} p(\omega)e^{i\omega\cdot x}d\omega,$$

where p is a non-negative measure.

Kernels of the form $K(x,x') = G(x-x')$ are called *shift-invariant kernels*. Note that if the kernel is scaled such that $G(0) = 1$, then p is in fact a probability distribution. Several examples of such kernels and their corresponding distributions are displayed in table 6.1. The next proposition provides a simplified expression in the case of real-valued kernels.

Proposition 6.25 *Let K be a continuous real-valued shift-invariant kernel and let p denote its corresponding non-negative measure as in theorem 6.24. Furthermore, assume that for all $x \in \mathcal{X}$ we have $K(x,x) = 1$ so that p is a probability distribution. Then, the following identity holds:*

$$\mathop{\mathbb{E}}_{\omega\sim p}\Big[\big[\cos(\omega\cdot x), \sin(\omega\cdot x)\big]^\top\big[\cos(\omega\cdot x'), \sin(\omega\cdot x')\big]\Big] = K(x,x')\,.$$

Proof: First, since both K and p are real-valued, it suffices to consider only the real portion of e^{ix} when invoking theorem 6.24. Thus, using $\mathrm{Re}[e^{ix}] = \mathrm{Re}[\cos(x) + i\sin(x)] = \cos(x)$, we have

$$K(x, x') = \mathrm{Re}[K(x, x')] = \int_{\mathcal{X}} p(\omega) \cos(\omega \cdot (x - x'))\, d\omega\,.$$

Next, by the standard trigonometric identity $\cos(a-b) = \cos(a)\cos(b) + \sin(a)\sin(b)$, we have

$$\begin{aligned}
&\int_{\mathcal{X}} p(\omega) \cos(\omega \cdot (x - x'))\, d\omega \\
&= \int_{\mathcal{X}} p(\omega) \big(\cos(\omega \cdot x) \cos(\omega \cdot x') + \sin(\omega \cdot x) \sin(\omega \cdot x') \big)\, d\omega \\
&= \mathop{\mathbb{E}}_{\omega \sim p} \Big[\big[\cos(\omega \cdot x), \sin(\omega \cdot x) \big]^\top \big[\cos(\omega \cdot x'), \sin(\omega \cdot x') \big] \Big]\,,
\end{aligned}$$

which completes the proof of the proposition. □

This proposition provides the motivation for a very simple method for generating for any $D \geq 1$, an approximate kernel map $\Psi \in \mathbb{R}^{2D}$, defined for all $x \in \mathcal{X}$ by

$$\Psi(x) = \sqrt{\frac{1}{D}} \Big[\cos(\omega_1 \cdot x), \sin(\omega_1 \cdot x), \ldots, \cos(\omega_D \cdot x), \sin(\omega_D \cdot x) \Big]^\top, \tag{6.21}$$

where ω_is, $i = 1, \ldots, D$, are sampled i.i.d. according to the measure p over $\mathcal{X}$ corresponding to kernel K considered. Thus,

$$\Psi(x) \cdot \Psi(x') = \frac{1}{D} \sum_{i=1}^{D} \Big[\cos(\omega_i \cdot x), \sin(\omega_i \cdot x) \Big]^\top \Big[\cos(\omega_i \cdot x'), \sin(\omega_i \cdot x') \Big]$$

is the empirical analog of the expectation computed in proposition 6.25. The following theorem shows that this empirical estimate converges uniformly over all points in a compact domain $\mathcal{X}$ as D grows.

Lemma 6.26 *Let K be a continuously differentiable kernel function that satisfies the conditions of proposition 6.25 and has associated measure p. Furthermore, assume $\mathcal{X}$ is compact and let N denote its dimension, R denote the radius of the Euclidean ball containing $\mathcal{X}$, and $\sigma_p^2 = \mathbb{E}_{\omega \sim p}[\|\omega\|^2] < \infty$. Then, for $\Psi \in \mathbb{R}^D$ as defined in* (6.21)*, the following holds for any $0 < r \leq 2R$ and $\epsilon > 0$:*

$$\mathbb{P}\bigg[\sup_{x, x' \in \mathcal{X}} \big| \Psi(x) \cdot \Psi(x') - K(x, x') \big| \geq \epsilon \bigg] \leq 2\mathcal{N}(2R, r) \exp\Big(-\frac{D\epsilon^2}{8} \Big) + \frac{4 r \sigma_p}{\epsilon}\,.$$

Where the probability is with respect to the draws of $\omega \sim p$ and $\mathcal{N}(R, r)$ denotes the minimal number of balls of radius r needed to cover a ball of radius R.

Proof: Define $\mathcal{Z} = \{z \colon z = x - x',\ x, x' \in \mathcal{X}\}$ and note that $\mathcal{Z}$ is contained in a ball of radius at most $2R$. $\mathcal{Z}$ is a closed set since $\mathcal{X}$ is closed and thus $\mathcal{Z}$ is a compact set. For convenience, define $B = \mathcal{N}(2R, r)$ the number of balls of radius r needed to cover $\mathcal{Z}$ and let z_j, for $j \in [B]$, denote the center of the covering balls. Thus, for any $z \in \mathcal{Z}$ there exists a j such that $z = z_j + \delta$ where $|\delta| < r$.

Next, define $S(z) = \Psi(x) \cdot \Psi(x') - K(x, x')$, where $z = x - x'$. Since S is continuously differentiable over the compact set $\mathcal{Z}$, it is L-Lipschitz with $L = \sup_{z \in \mathcal{Z}} \|\nabla S(z)\|$. Note that if $L < \frac{\epsilon}{2r}$ and for all $j \in [B]$ we have $|S(z_j)| < \frac{\epsilon}{2}$, then the following inequality holds for all $z = z_j + \delta \in \mathcal{Z}$:

$$|S(z)| = |S(z_j + \delta)| \leq L|z_j - (z_j + \delta)| + |S(z_j)| \leq rL + \frac{\epsilon}{2} < \epsilon\,. \tag{6.22}$$

The remainder of this proof bounds the probability of the events $L \geq \frac{\epsilon}{2r}$ and $|S(z_j)| \geq \frac{\epsilon}{2}$. Note, all following probabilities and expectations are with respect to the random variables $\omega_1, \ldots, \omega_D$.

To bound the probability of the first event, we use proposition 6.25 and the linearity of expectation, which implies the key fact $\mathbb{E}[\nabla(\Psi(x) \cdot \Psi(x'))] = \nabla K(x, x')$. We proceed with the following series of inequalities:

$$\begin{aligned}
\mathbb{E}[L^2] &= \mathbb{E}\left[\sup_{z \in \mathcal{Z}} \|\nabla S(z)\|^2\right] \\
&= \mathbb{E}\left[\sup_{x, x' \in \mathcal{X}} \|\nabla(\Psi(x) \cdot \Psi(x')) - \nabla K(x, x')\|^2\right] \\
&\leq 2\,\mathbb{E}\left[\sup_{x, x' \in \mathcal{X}} \|\nabla(\Psi(x) \cdot \Psi(x'))\|^2\right] + 2 \sup_{x, x' \in \mathcal{X}} \|\nabla K(x, x')\|^2 \\
&= 2\,\mathbb{E}\left[\sup_{x, x' \in \mathcal{X}} \|\nabla(\Psi(x) \cdot \Psi(x'))\|^2\right] + 2 \sup_{x, x' \in \mathcal{X}} \|\,\mathbb{E}[\nabla(\Psi(x) \cdot \Psi(x'))]\|^2 \\
&\leq 4\,\mathbb{E}\left[\sup_{x, x' \in \mathcal{X}} \|\nabla(\Psi(x) \cdot \Psi(x'))\|^2\right],
\end{aligned}$$

where the first inequality holds due to the the inequality $\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$ (which follows from Jensen's inequality) and the subadditivity of the supremum function. The second inequality also holds by Jensen's inequality (applied twice) and again the subadditivity of supremum function. Furthermore, using a sum-difference trigonometric identity and computing the gradient with respect to $z = x - x'$, yield the following for any $x, x' \in \mathcal{X}$:

$$\begin{aligned}
\nabla(\Psi(x) \cdot \Psi(x')) &= \nabla\left(\frac{1}{D} \sum_{i=1}^{D} \cos(\omega_i \cdot x) \cos(\omega_i \cdot x') + \sin(\omega_i \cdot x) \sin(\omega_i \cdot x')\right) \\
&= \nabla\left(\frac{1}{D} \sum_{i=1}^{D} \cos(\omega_i \cdot (x - x'))\right) = \frac{1}{D} \sum_{i=1}^{D} \omega_i \sin(\omega_i \cdot (x - x'))\,.
\end{aligned}$$

Combining the two previous results gives

$$\begin{aligned}
\mathbb{E}[L^2] &\leq 4\,\mathbb{E}\left[\sup_{x,x'\in\mathcal{X}}\left\|\frac{1}{D}\sum_{i=1}^{D}\omega_i\sin(\omega_i\cdot(x-x'))\right\|^2\right]\\
&\leq 4\underset{\omega_1,\ldots,\omega_N}{\mathbb{E}}\left[\Big(\frac{1}{D}\sum_{i=1}^{D}\|\omega_i\|\Big)^2\right]\\
&\leq 4\underset{\omega_1,\ldots,\omega_N}{\mathbb{E}}\left[\frac{1}{D}\sum_{i=1}^{D}\|\omega_i\|^2\right] = 4\,\underset{\omega}{\mathbb{E}}\left[\|\omega\|^2\right] = 4\sigma_p^2\,,
\end{aligned}$$

which follows from the triangle inequality, $|\sin(\cdot)|\leq 1$, Jensen's inequality and the fact that the ω_is are drawn i.i.d. derive the final expression. Thus, we can bound the probability of the first event via Markov's inequality:

$$\mathbb{P}\Big[L\geq\frac{\epsilon}{2r}\Big]\leq\Big(\frac{4r\sigma_p}{\epsilon}\Big)^2. \tag{6.23}$$

To bound the probability of the second event, note that, by definition, $S(z)$ is a sum of D i.i.d. variables, each bounded in absolute value by $\frac{2}{D}$ (since, for all x and x', we have $|K(x,x')|\leq 1$ and $|\Psi(x)\cdot\Psi(x')|\leq 1$), and $\mathbb{E}[S(z)]=0$. Thus, by Hoeffding's inequality and the union bound, we can write

$$\mathbb{P}\Big[\exists j\in[B]\colon|S(z_j)|\geq\frac{\epsilon}{2}\Big]\leq\sum_{i=1}^{B}\mathbb{P}\Big[|S(z_j)|\geq\frac{\epsilon}{2}\Big]\leq 2B\exp\Big(-\frac{D\epsilon^2}{8}\Big). \tag{6.24}$$

Finally, combining (6.22), (6.23), (6.24), and the definition of B we have

$$\mathbb{P}\Big[\sup_{z\in\mathcal{Z}}|S(z)|\geq\epsilon\Big]\leq 2\mathcal{N}(2R,r)\exp\Big(-\frac{D\epsilon^2}{8}\Big)+\Big(\frac{4r\sigma_p}{\epsilon}\Big)^2,$$

which completes the lemma. □

A key factor in the bound of the lemma is the covering number $\mathcal{N}(2R,r)$, which strongly depends on the dimension of the space N. In the following lemma, we make this dependency explicit for one especially simple case, although similar arguments hold for more general scenarios as well.

Lemma 6.27 *Let $\mathcal{X}\subset\mathbb{R}^N$ be a compact and let R denote the radius of the smallest enclosing ball. Then, the following inequality holds:*

$$\mathcal{N}(R,r)\leq\Big(\frac{3R}{r}\Big)^N.$$

Proof: First, by using the volume of balls in $\mathbb{R}^N$ we already see that $R^N/(r/3)^N=(3R/r)^N$ is a trivial upper bound on the number of balls of radius $r/3$ that can be packed into a ball of radius R without intersecting. Now, consider a maximal packing of at most $(3R/r)^N$ balls of radius $r/3$ into the ball of radius R. Every point in the ball of radius R is at distance at most r from the center of at least one of the packing balls. If this were not true, we would be able to fit another ball into the packing, thereby contradicting the assumption that it is a maximal packing. Thus, if we grow the radius of the at most $(3R/r)^N$ balls to r, they will then provide a (not necessarily minimal) cover of the ball of radius R. □

Finally, by combining the two previous lemmas, we can present an explicit finite sample approximation bound.

Theorem 6.28 *Let K be a continuously differentiable kernel function that satisfies the conditions of proposition 6.25 and has associated measure p. Furthermore, assume $\sigma_p^2 = \mathbb{E}_{\omega\sim p}[\|\omega\|^2] < \infty$ and $\mathcal{X} \subset \mathbb{R}^N$. Let R denote the radius of the Euclidean ball containing $\mathcal{X}$. Then, for $\Psi \in \mathbb{R}^D$ as defined in (6.21) and any $0 < \epsilon \leq 32R\sigma_p$, the following holds*

$$\mathbb{P}\left[\sup_{x,x'\in\mathcal{X}} \big|\Psi(x)\cdot\Psi(x') - K(x,x')\big| \geq \epsilon\right] \leq \left(\frac{48R\sigma_p}{\epsilon}\right)^2 \exp\left(-\frac{D\epsilon^2}{4(N+2)}\right).$$

Proof: We use lemma 6.27 in conjunction with lemma 6.26 with the following choice of r:

$$r = \left[\frac{2(6R)^N \exp(-\frac{D\epsilon^2}{8})}{\left(\frac{4\sigma_p}{\epsilon}\right)^2}\right]^{\frac{2}{N+2}},$$

which results in the following expression

$$\mathbb{P}\left[\sup_{z\in\mathcal{Z}} |S(z)| \geq \epsilon\right] \leq 4\left(\frac{24R\sigma_p}{\epsilon}\right)^{\frac{2N}{N+2}} \exp\left(-\frac{D\epsilon^2}{4(N+2)}\right).$$

Since $32R\sigma_p/\epsilon \geq 1$, the exponent $\frac{2N}{N+2}$ can be replaced by 2, which completes the proof. □

The previous theorem provides the guarantee that a good estimate of the kernel function can be found, with high probability, by sampling a finite number of coordinates D. In particular, for an absolute error of at most ϵ it suffices to sample $D = O\Big(\frac{N}{\epsilon^2}\log\Big(\frac{R\sigma_p}{\epsilon}\Big)\Big)$ coordinates.

6.7 Chapter notes

The mathematical theory of PDS kernels in a general setting originated with the fundamental work of Mercer [1909] who also proved the equivalence of a condition similar to that of theorem 6.2 for continuous kernels with the PDS property. The connection between PDS and NDS kernels, in particular theorems 6.18 and 6.17, are due to Schoenberg [1938]. A systematic treatment of the theory of reproducing kernel Hilbert spaces was presented in a long and elegant paper by Aronszajn [1950]. For an excellent mathematical presentation of PDS kernels and positive definite functions we refer the reader to Berg, Christensen, and Ressel [1984], which is also the source of several of the exercises given in this chapter.

The fact that SVMs could be extended by using PDS kernels was pointed out by Boser, Guyon, and Vapnik [1992]. The idea of kernel methods has been since then widely adopted in machine learning and applied in a variety of different tasks and settings. The following two books are in fact specifically devoted to the study of kernel methods: Schölkopf and Smola [2002] and Shawe-Taylor and Cristianini [2004]. The classical representer theorem is due to Kimeldorf and Wahba [1971]. A generalization to non-quadratic cost functions was stated by Wahba [1990]. The general form presented in this chapter was given by Schölkopf, Herbrich, Smola, and Williamson [2000].

Rational kernels were introduced by Cortes, Haffner, and Mohri [2004]. A general class of kernels, *convolution kernels*, was earlier introduced by Haussler [1999]. The convolution kernels for sequences described by Haussler [1999], as well as the pair-HMM string kernels described by Watkins [1999], are special instances of rational kernels. Rational kernels can be straightforwardly extended to define kernels for finite automata and even weighted automata [Cortes et al., 2004]. Cortes, Mohri, and Rostamizadeh [2008b] study the problem of *learning* rational kernels such as those based on counting transducers.

The composition of weighted transducers and the filter transducers in the presence of ϵ-paths are described in Pereira and Riley [1997], Mohri, Pereira, and Riley [2005], and Mohri [2009]. Composition can be further generalized to the *N-way composition* of weighted transducers [Allauzen and Mohri, 2009]. N-way composition of three or more transducers can substantially speed up computation, in particular for PDS rational kernels of the form $T \circ T^{-1}$. A generic *shortest-distance algorithm* which can be used with a large class of semirings and arbitrary queue disciplines is described by Mohri [2002]. A specific instance of that algorithm can be used to compute the sum of the weights of all paths as needed for the computation of rational kernels after composition. For a study of the class of languages linearly separable with rational kernels, see Cortes, Kontorovich, and Mohri [2007a].

The use of cosine-based approximate kernel feature maps was introduced by Rahimi and Recht [2007], as were the corresponding uniform convergence bounds, though their proofs were not complete. Sriperumbudur and Szabó [2015] gave an improved approximation bound that reduces the dependence on the radius of the data from $O(R^2)$ to only $O(\log(R))$. Bochner's theorem, which plays a central role in deriving an approximate map, is a classical result of harmonic analysis (for example, see Rudin [1990]). The general form of the theorem is due to Weil [1965], while Solomon Bochner recognized its importance to harmonic analysis.

6.8 Exercises

6.1 Let $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel, and let $\alpha\colon \mathcal{X} \to \mathbb{R}$ be a positive function. Show that the kernel K' defined for all $x, y \in \mathcal{X}$ by $K'(x, y) = \frac{K(x,y)}{\alpha(x)\alpha(y)}$ is a PDS kernel.

6.2 Show that the following kernels K are PDS:

(a) $K(x, y) = \cos(x - y)$ over $\mathbb{R} \times \mathbb{R}$.

(b) $K(x, y) = \cos(x^2 - y^2)$ over $\mathbb{R} \times \mathbb{R}$.

(c) For all integers $n > 0, K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} \cos^n(x_i^2 - y_i^2)$ over $\mathbb{R}^N \times \mathbb{R}^N$.

(d) $K(x, y) = (x + y)^{-1}$ over $(0, +\infty) \times (0, +\infty)$.

(e) $K(\mathbf{x}, \mathbf{x}') = \cos \angle(\mathbf{x}, \mathbf{x}')$ over $\mathbb{R}^n \times \mathbb{R}^n$, where $\angle(\mathbf{x}, \mathbf{x}')$ is the angle between $\mathbf{x}$ and $\mathbf{x}'$.

(f) $\forall \lambda > 0,\ K(x, x') = \exp\big(-\lambda[\sin(x' - x)]^2\big)$ over $\mathbb{R} \times \mathbb{R}$.
(*Hint*: rewrite $[\sin(x' - x)]^2$ as the square of the norm of the difference of two vectors.)

(g) $\forall \sigma > 0, K(x, y) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}}$ over $\mathbb{R}^N \times \mathbb{R}^N$.
(*Hint*: you could show that K is the normalized kernel of a kernel K' and show that K' is PDS using the following equality: $\|\mathbf{x} - \mathbf{y}\| = \frac{1}{2\Gamma(\frac{1}{2})} \int_0^{+\infty} \frac{1-e^{-t\|\mathbf{x}-\mathbf{y}\|^2}}{t^{\frac{3}{2}}} dt$ valid for all $\mathbf{x}, \mathbf{y}$.)

(h) $K(x, y) = \min(x, y) - xy$ over $[0, 1] \times [0, 1]$.
(*Hint*: you could consider the two integrals $\int_0^1 1_{t\in[0,x]} 1_{t\in[0,y]} dt$ and $\int_0^1 1_{t\in[x,1]} 1_{t\in[y,1]} dt$.)

(i) $K(x, x') = \frac{1}{\sqrt{1-(\mathbf{x}\cdot\mathbf{x}')}}$ over $\mathbf{x}, \mathbf{x}' \in \mathcal{X} = \{\mathbf{x} \in \mathbb{R}^N \colon \|\mathbf{x}\|_2 < 1\}$.
(*Hint*: one approach is to find an explicit expression of a feature mapping Φ by considering the Taylor expansion of the kernel function.)

(j) $\forall \sigma > 0, K(x, y) = \frac{1}{1+\frac{\|x-y\|^2}{\sigma^2}}$ over $\mathbb{R}^N \times \mathbb{R}^N$.
(*Hint*: the function $x \mapsto \int_0^{+\infty} e^{-sx} e^{-s} ds$ defined for all $x \geq 0$ could be useful for the proof.)

(k) $\forall \sigma > 0, K(x, y) = \exp\left(\frac{\sum_{i=1}^{N} \min(|x_i|, |y_i|)}{\sigma^2}\right)$ over $\mathbb{R}^N \times \mathbb{R}^N$.
(*Hint*: the function $(x_0, y_0) \mapsto \int_0^{+\infty} 1_{t\in[0,|x_0|]} 1_{t\in[0,|y_0|]} dt$ defined over $\mathbb{R} \times \mathbb{R}$ could be useful for the proof.)

6.3 Graph kernel. Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. $\mathcal{V}$ could represent a set of documents or biosequences and E the set of connections between them. Let $w[e] \in \mathbb{R}$ denote the weight assigned to edge $e \in \mathcal{E}$. The weight of a path is the product of the weights of its constituent edges. Show that the kernel K over $\mathcal{V} \times \mathcal{V}$ where $K(p, q)$ is the sum of the weights of all paths of length two between p and q is PDS (*Hint*: you could introduce the matrix $W = (W_{pq})$, where $W_{pq} = 0$ when there is no edge between p and q, W_{pq} equal to the weight of the edge between p and q otherwise).

6.4 Symmetric difference kernel. Let $\mathcal{X}$ be a finite set. Show that the kernel K defined over $2^{\mathcal{X}}$, the set of subsets of $\mathcal{X}$, by

$$\forall \mathcal{A}, \mathcal{B} \in 2^{\mathcal{X}}, K(\mathcal{A}, \mathcal{B}) = \exp\Big(-\frac{1}{2}|\mathcal{A}\Delta\mathcal{B}|\Big),$$

where $\mathcal{A}\Delta\mathcal{B}$ is the symmetric difference of $\mathcal{A}$ and $\mathcal{B}$ is PDS (*Hint*: you could use the fact that K is the result of the normalization of a kernel function K').

6.5 Set kernel. Let $\mathcal{X}$ be a finite set. Let K_0 be a PDS kernel over $\mathcal{X}$, show that K' defined by

$$\forall \mathcal{A}, \mathcal{B} \in 2^{\mathcal{X}}, K'(\mathcal{A}, \mathcal{B}) = \sum_{x \in \mathcal{A}, x' \in \mathcal{B}} K_0(x, x')$$

is a PDS kernel.

6.6 Show that the following kernels K are NDS:

(a) $K(x, y) = [\sin(x - y)]^2$ over $\mathbb{R} \times \mathbb{R}$.

(b) $K(x, y) = \log(x + y)$ over $(0, +\infty) \times (0, +\infty)$.

6.7 Define a *difference kernel* as $K(x, x') = |x - x'|$ for $x, x' \in \mathbb{R}$. Show that this kernel is not positive definite symmetric (PDS).

6.8 Is the kernel K defined over $\mathbb{R}^n \times \mathbb{R}^n$ by $K(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^{3/2}$ PDS? Is it NDS?

6.9 Let $\mathcal{H}$ be a Hilbert space with the corresponding dot product $\langle \cdot, \cdot \rangle$. Show that the kernel K defined over $\mathcal{H} \times \mathcal{H}$ by $K(x, y) = 1 - \langle x, y \rangle$ is negative definite.

6.10 For any $p > 0$, let K_p be the kernel defined over $\mathbb{R}_+ \times \mathbb{R}_+$ by

$$K_p(x, y) = e^{-(x+y)^p}. \tag{6.25}$$

Show that K_p is positive definite symmetric (PDS) iff $p \leq 1$. (*Hint*: you can use the fact that if K is NDS, then for any $0 < \alpha \leq 1$, K^α is also NDS.)

6.11 Explicit mappings.

(a) Denote a data set $x_1, \ldots, x_m$ and a kernel $K(x_i, x_j)$ with a Gram matrix $\mathbf{K}$. Assuming $\mathbf{K}$ is positive semidefinite, then give a map $\Phi(\cdot)$ such that $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$.

(b) Show the converse of the previous statement, i.e., if there exists a mapping $\Phi(x)$ from input space to some Hilbert space, then the corresponding matrix $\mathbf{K}$ is positive semidefinite.

6.12 Explicit polynomial kernel mapping. Let K be a polynomial kernel of degree d, i.e., $K \colon \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$, $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + c)^d$, with $c > 0$, Show that the dimension of the feature space associated to K is

$$\binom{N+d}{d}. \tag{6.26}$$

Write K in terms of kernels $k_i \colon (\mathbf{x}, \mathbf{x}') \mapsto (\mathbf{x} \cdot \mathbf{x}')^i$, $i \in \{0, \ldots, d\}$. What is the weight assigned to each k_i in that expression? How does it vary as a function of c?

6.13 High-dimensional mapping. Let $\Phi \colon \mathcal{X} \to \mathcal{H}$ be a feature mapping such that the dimension N of $\mathcal{H}$ is very large and let $K \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel defined by

$$K(x, x') = \mathop{\mathbb{E}}_{i \sim \mathcal{D}} \big[[\Phi(x)]_i [\Phi(x')]_i \big], \tag{6.27}$$

where $[\Phi(x)]_i$ is the ith component of $\Phi(x)$ (and similarly for $\Phi'(x)$) and where $\mathcal{D}$ is a distribution over the indices i. We shall assume that $|[\Phi(x)]_i| \leq R$ for all $x \in \mathcal{X}$ and $i \in [N]$. Suppose that the only method available to compute $K(x, x')$ involved direct computation of the inner product (6.27), which would require $O(N)$ time. Alternatively, an approximation can be computed based on random selection of a subset I of the N components of $\Phi(x)$ and $\Phi(x')$ according to $\mathcal{D}$, that is:

$$K'(x, x') = \frac{1}{n} \sum_{i \in I} \mathcal{D}(i) [\Phi(x)]_i [\Phi(x')]_i, \tag{6.28}$$

where $|I| = n$.

(a) Fix x and x' in $\mathcal{X}$. Prove that

$$\mathop{\mathbb{P}}_{I \sim \mathcal{D}^n} [|K(x, x') - K'(x, x')| > \epsilon] \leq 2 e^{\frac{-n\epsilon^2}{2r^2}}. \tag{6.29}$$

(*Hint*: use McDiarmid's inequality).

(b) Let $\mathbf{K}$ and $\mathbf{K}'$ be the kernel matrices associated to K and K'. Show that for any $\epsilon, \delta > 0$, for $n > \frac{r^2}{\epsilon^2} \log \frac{m(m+1)}{\delta}$, with probability at least $1 - \delta$, $|\mathbf{K}'_{ij} - \mathbf{K}_{ij}| \leq \epsilon$ for all $i, j \in [m]$.

6.14 Classifier based kernel. Let S be a training sample of size m. Assume that S has been generated according to some probability distribution $\mathcal{D}(x, y)$, where $(x, y) \in \mathcal{X} \times \{-1, +1\}$.

(a) Define the Bayes classifier $h^* \colon \mathcal{X} \to \{-1, +1\}$. Show that the kernel K^* defined by $K^*(x, x') = h^*(x)h^*(x')$ for any $x, x' \in \mathcal{X}$ is positive definite symmetric. What is the dimension of the natural feature space associated to K^*?

(b) Give the expression of the solution obtained using SVMs with this kernel. What is the number of support vectors? What is the value of the margin? What is the generalization error of the solution obtained? Under what condition are the data linearly separable?

(c) Let $h : \mathcal{X} \to \mathbb{R}$ be an arbitrary real-valued function. Under what condition on h is the kernel K defined by $K(x, x') = h(x)h(x')$, $x, x' \in \mathcal{X}$, positive definite symmetric?

6.15 Image classification kernel. For $\alpha \geq 0$, the kernel

$$K_\alpha \colon (\mathbf{x}, \mathbf{x}') \mapsto \sum_{k=1}^{N} \min(|x_k|^\alpha, |x'_k|^\alpha) \tag{6.30}$$

over $\mathbb{R}^N \times \mathbb{R}^N$ is used in image classification. Show that K_α is PDS for all $\alpha \geq 0$. To do so, proceed as follows.

(a) Use the fact that $(f, g) \mapsto \int_{t=0}^{+\infty} f(t)g(t)dt$ is an inner product over the set of measurable functions over $[0, +\infty)$ to show that $(x, x') \mapsto \min(x, x')$ is a PDS kernel. (*Hint*: associate an indicator function to x and another one to x'.)

(b) Use the result from (a) to first show that K_1 is PDS and similarly that K_α with other values of α is also PDS.

6.16 Fraud detection. To prevent fraud, a credit-card company decides to contact Professor Villebanque and provides him with a random list of several thousand fraudulent and non-fraudulent *events*. There are many different types of events, e.g., transactions of various amounts, changes of address or card-holder information, or requests for a new card. Professor Villebanque decides to use SVMs with an appropriate kernel to help predict fraudulent events accurately. It is difficult for Professor Villebanque to define relevant features for such a diverse set of events. However, the risk department of his company has created a complicated method to estimate a probability $\mathbb{P}[U]$ for any event U. Thus, Professor Villebanque decides to make use of that information and comes up with the following kernel defined over all pairs of events (U, V):

$$K(U, V) = \mathbb{P}[U \wedge V] - \mathbb{P}[U]\,\mathbb{P}[V]. \tag{6.31}$$

Help Professor Villebanque show that his kernel is positive definite symmetric.

6.17 Relationship between NDS and PDS kernels. Prove the statement of theorem 6.17. (*Hint*: Use the fact that if K is PDS then $\exp(K)$ is also PDS, along with theorem 6.16.)

6.18 Metrics and Kernels. Let $\mathcal{X}$ be a non-empty set and $K \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a negative definite symmetric kernel such that $K(x, x) = 0$ for all $x \in \mathcal{X}$.

(a) Show that there exists a Hilbert space $\mathbb{H}$ and a mapping $\Phi(x)$ from $\mathcal{X}$ to $\mathbb{H}$ such that:

$$K(x, y) = ||\Phi(x) - \Phi(x')||^2\,.$$

Assume that $K(x, x') = 0 \Rightarrow x = x'$. Use theorem 6.16 to show that $\sqrt{K}$ defines a metric on $\mathcal{X}$.

(b) Use this result to prove that the kernel $K(x, y) = \exp(-|x - x'|^p)$, $x, x' \in \mathbb{R}$, is not positive definite for $p > 2$.

(c) The kernel $K(x, x') = \tanh(a(x \cdot x') + b)$ was shown to be equivalent to a two-layer neural network when combined with SVMs. Show that K is not positive definite if $a < 0$ or $b < 0$. What can you conclude about the corresponding neural network when $a < 0$ or $b < 0$?

6.19 Sequence kernels. Let $\mathcal{X} = \{a, c, g, t\}$. To classify DNA sequences using SVMs, we wish to define a kernel between sequences defined over $\mathcal{X}$. We are given a finite set $\mathcal{I} \subset \mathcal{X}^*$ of non-coding regions (introns). For $x \in \mathcal{X}^*$, denote by $|x|$ the length of x and by $F(x)$ the set of factors of x, i.e., the set of subsequences of x with contiguous symbols. For any two strings $x, y \in \mathcal{X}^*$ define $K(x, y)$ by

$$K(x, y) = \sum_{z \in (F(x) \cap F(y)) - \mathcal{I}} \rho^{|z|}, \tag{6.32}$$

where $\rho \geq 1$ is a real number.

(a) Show that K is a rational kernel and that it is positive definite symmetric.

(b) Give the time and space complexity of the computation of $K(x, y)$ with respect to the size s of a minimal automaton representing $\mathcal{X}^* - \mathcal{I}$.

(c) Long common factors between x and y of length greater than or equal to n are likely to be important coding regions (exons). Modify the kernel K to assign weight $\rho_2^{|z|}$ to z when $|z| \geq n$, $\rho_1^{|z|}$ otherwise, where $1 \leq \rho_1 \ll \rho_2$. Show that the resulting kernel is still positive definite symmetric.

6.20 n-gram kernel. Show that for all $n \geq 1$, and any n-gram kernel K_n, $K_n(x, y)$ can be computed in linear time $O(|x| + |y|)$, for all $x, y \in \Sigma^*$ assuming n and the alphabet size are constants.

6.21 Mercer's condition. Let $\mathcal{X} \subset \mathbb{R}^N$ be a compact set and $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a continuous kernel function. Prove that if K verifies Mercer's condition (theorem 6.2), then it is PDS. (*Hint*: assume that K is not PDS and consider a set $\{x_1, \ldots, x_m\} \subseteq \mathcal{X}$ and a column-vector $c \in \mathbb{R}^{m \times 1}$ such that $\sum_{i,j=1}^m c_i c_j K(x_i, x_j) < 0$.)

6.22 Anomaly detection. For this problem, consider a Hilbert space $\mathbb{H}$ with associated feature map $\Phi : \mathcal{X} \to \mathbb{H}$ and kernel $K(x, x') = \Phi(x) \cdot \Phi(x')$.

(a) First, let us consider finding the smallest enclosing sphere for a given sample $S = (x_1, \ldots, x_m)$. Let $\mathbf{c} \in \mathbb{H}$ denote the center of the sphere and let $r > 0$ be its radius, then clearly the following optimization problem searches for the smallest enclosing sphere:

$$\min_{r>0, \mathbf{c} \in \mathbb{H}} r^2$$
$$\text{subject to: } \forall i \in [m], \|\Phi(x_i) - \mathbf{c}\|^2 \leq r^2.$$

Show how to derive the equivalent dual optimization

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^m \alpha_i K(x_i, x_i) - \sum_{i,j=1}^m \alpha_i \alpha_j K(x_i, x_j)$$
$$\text{subject to: } \boldsymbol{\alpha} \geq \mathbf{0} \ \wedge \ \sum_{i=1}^m \alpha_i = 1\,,$$

and prove that the optimal solution satisfies $\mathbf{c} = \sum_i \alpha_i \Phi(x_i)$. In other words the location of the sphere only depends on points x_i with non-zero coefficients α_i. These points are analogous to the support vectors of SVM.

(b) Consider the hypothesis class

$$\mathcal{H} = \{x \mapsto r^2 - \|\Phi(x) - \mathbf{c}\|^2 \colon \|\mathbf{c}\| \le \Lambda,\ 0 < r \le R\}.$$

A hypothesis $h \in \mathcal{H}$ can be used to detect anomalies in data, where $h(x) \ge 0$ indicates a non-anomalous point and $h(x) < 0$ indicates an anomaly.

Show that if $\sup_x \|\Phi(x)\| \le M$, then the solution to the optimization problem in part (a) is found in the hypothesis set $\mathcal{H}$ with $\Lambda \le M$ and $R \le 2M$.

(c) Let $\mathcal{D}$ denote the distribution of non-outlier points define the associated expected loss $R(h) = \mathbb{E}_{x\sim\mathcal{D}}[1_{h(x)<0}]$ and empirical margin loss $\widehat{R}_{S,\rho}(h) = \sum_{i=1}^m \frac{1}{m}\Phi_\rho(h(x_i)) \le \sum_{i=1}^m \frac{1}{m} 1_{h(x_i)<\rho}$. These losses measure errors caused by *false-positive* predictions, i.e. errors caused by incorrectly labeling a point anomalous.

i. Show that the empirical Rademacher complexity for the hypothesis class $\mathcal{H}$ from part (b) can be upper bound as follows:

$$\widehat{\mathfrak{R}}_S(\mathcal{H}) \le \frac{R^2 + \Lambda^2}{\sqrt{m}} + \Lambda\sqrt{\operatorname{Tr}[\mathbf{K}]},$$

where $\mathbf{K}$ is the kernel matrix constructed with the sample.

ii. Prove that with probability at least $1-\delta$, the following holds for all $h \in \mathcal{H}$ and $\rho \in (0, 1]$:

$$R(h) \le \widehat{R}_{S,\rho}(h) + \frac{4}{\rho}\Big(\frac{R^2 + \Lambda^2}{\sqrt{m}} + \Lambda\sqrt{\operatorname{Tr}[\mathbf{K}]}\Big) + \sqrt{\frac{\log\log_2 \frac{2}{\rho}}{m}} + 3\sqrt{\frac{\log \frac{4}{\delta}}{2m}}.$$

(d) Just as in the case of soft-margin SVM, we can also define a soft-margin objective for the smallest enclosing sphere that allows us tune the sensitivity to outliers in the training set by adjusting a regularization parameter C:

$$\min_{r>0, \mathbf{c}\in\mathbb{H}, \xi} r^2 + C\sum_{i=1}^m \xi_i$$
$$\text{subject to: } \forall i \in [m], \|\Phi(x_i) - \mathbf{c}\|^2 \le r^2 + \xi_i \ \wedge\ \xi_i \ge 0.$$

Show that the equivalent dual formulation of this problem is

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i K(x_i, x_i) - \sum_{i,j=1}^{m} \alpha_i \alpha_j K(x_i, x_j)$$

$$\text{subject to: } \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1} \ \wedge \ \sum_{i=1}^{m} \alpha_i = 1\,,$$

and that at the optimum we have $\mathbf{c} = \sum_{i=1}^{m} \alpha_i \Phi(x_i)$.

7 Boosting

Ensemble methods are general techniques in machine learning for combining several predictors to create a more accurate one. This chapter studies an important family of ensemble methods known as *boosting*, and more specifically the *AdaBoost* algorithm. This algorithm has been shown to be very effective in practice in some scenarios and is based on a rich theoretical analysis. We first introduce AdaBoost, show how it can rapidly reduce the empirical error as a function of the number of rounds of boosting, and point out its relationship with some known algorithms. Next, we present a theoretical analysis of the generalization properties of AdaBoost based on the VC-dimension of its hypothesis set and then based on the notion of margin. The margin theory developed in this context can be applied to other similar ensemble algorithms. A game-theoretic interpretation of AdaBoost further helps analyzing its properties and revealing the equivalence between the weak learning assumption and a separability condition. We end with a discussion of AdaBoost's benefits and drawbacks.

7.1 Introduction

It is often difficult, for a non-trivial learning task, to directly devise an accurate algorithm satisfying the strong PAC-learning requirements of chapter 2. But, there can be more hope for finding simple predictors guaranteed only to perform slightly better than random. The following gives a formal definition of such *weak learners*. As in the PAC-learning chapter, we let n be a number such that the computational cost of representing any element $x \in \mathcal{X}$ is at most $O(n)$ and denote by $\text{size}(c)$ the maximal cost of the computational representation of $c \in \mathcal{C}$.

Definition 7.1 (Weak learning) *A concept class $\mathcal{C}$ is said to be* weakly PAC-learnable *if there exists an algorithm $\mathcal{A}$, $\gamma > 0$, and a polynomial function poly$(\cdot, \cdot, \cdot)$ such that for any $\delta > 0$, for all distributions $\mathcal{D}$ on $\mathcal{X}$ and for any target concept $c \in \mathcal{C}$,*

```
AdaBoost($S = ((x_1, y_1), \ldots, (x_m, y_m))$)
 1  for $i \leftarrow 1$ to $m$ do
 2      $\mathcal{D}_1(i) \leftarrow \frac{1}{m}$
 3  for $t \leftarrow 1$ to $T$ do
 4      $h_t \leftarrow$ base classifier in $\mathcal{H}$ with small error $\epsilon_t = \mathbb{P}_{i \sim \mathcal{D}_t}\big[h_t(x_i) \neq y_i\big]$
 5      $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
 6      $Z_t \leftarrow 2\big[\epsilon_t(1-\epsilon_t)\big]^{\frac{1}{2}}$    ▷ normalization factor
 7      for $i \leftarrow 1$ to $m$ do
 8          $\mathcal{D}_{t+1}(i) \leftarrow \frac{\mathcal{D}_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
 9  $f \leftarrow \sum_{t=1}^{T} \alpha_t h_t$
10  return $f$
```

Figure 7.1
AdaBoost algorithm for a base classifier set $\mathcal{H} \subseteq \{-1, +1\}^{\mathcal{X}}$.

the following holds for any sample size $m \geq poly(1/\delta, n, size(c))$:

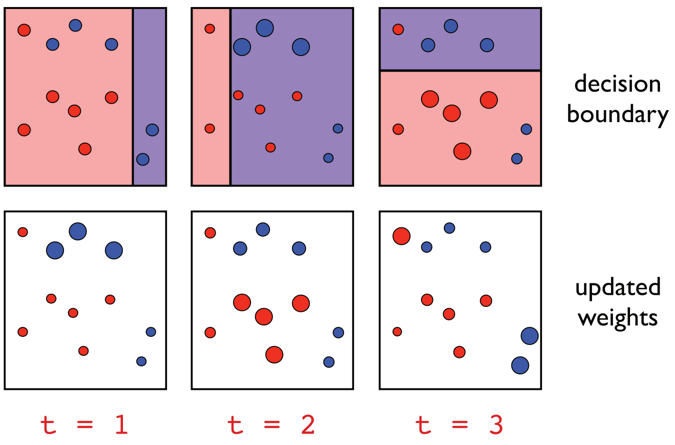
$$\mathop{\mathbb{P}}_{S \sim \mathcal{D}^m}\left[R(h_S) \leq \frac{1}{2} - \gamma\right] \geq 1 - \delta\,, \tag{7.1}$$

where h_S is the hypothesis returned by algorithm $\mathcal{A}$ when trained on sample S. When such an algorithm $\mathcal{A}$ exists, it is called a weak learning algorithm *for $\mathcal{C}$ or a* weak learner*. The hypotheses returned by a weak learning algorithm are called* base classifiers*.*

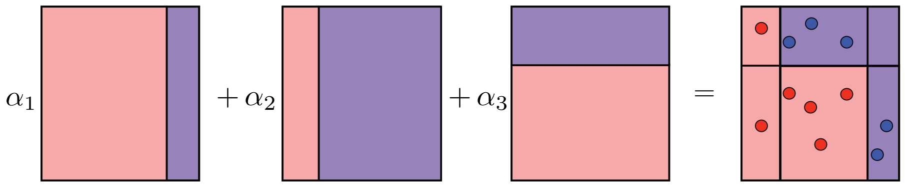
The key idea behind boosting techniques is to use a weak learning algorithm to build a *strong learner*, that is, an accurate PAC-learning algorithm. To do so, boosting techniques use an ensemble method: they combine different base classifiers returned by a weak learner to create a more accurate predictor. But which base classifiers should be used and how should they be combined? The next section addresses these questions by describing in detail one of the most prevalent and successful boosting algorithms, AdaBoost.

7.2 AdaBoost

We denote by $\mathcal{H}$ the hypothesis set out of which the base classifiers are selected, which we will sometimes refer to as the *base classifier set*. Figure 7.1 gives the



(a)



(b)

Figure 7.2
Example of AdaBoost with axis-aligned hyperplanes as base classifiers. (a) The top row shows decision boundaries at each boosting round. The bottom row shows how weights are updated at each round, with incorrectly (resp., correctly) points given increased (resp., decreased) weights. (b) Visualization of final classifier, constructed as a non-negative linear combination of base classifiers.

pseudocode of AdaBoost in the case where the base classifiers are functions mapping from $\mathcal{X}$ to $\{-1,+1\}$, thus $\mathcal{H} \subseteq \{-1,+1\}^{\mathcal{X}}$.

The algorithm takes as input a labeled sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, with $(x_i, y_i) \in \mathcal{X} \times \{-1,+1\}$ for all $i \in [m]$, and maintains a distribution over the indices $\{1, \ldots, m\}$. Initially (lines 1-2), the distribution is uniform ($\mathcal{D}_1$). At each *round of boosting*, that is each iteration $t \in [T]$ of the loop 3–8, a new base classifier $h_t \in \mathcal{H}$ is selected that minimizes the error on the training sample weighted by the distribution $\mathcal{D}_t$:

$$h_t \in \operatorname*{argmin}_{h \in \mathcal{H}} \mathop{\mathbb{P}}_{i \sim \mathcal{D}_t} \big[h(x_i) \neq y_i\big] = \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{i=1}^{m} \mathcal{D}_t(i) 1_{h(x_i) \neq y_i}.$$

Z_t is simply a normalization factor to ensure that the weights $\mathcal{D}_{t+1}(i)$ sum to one. The precise reason for the definition of the coefficient α_t will become clear later. For now, observe that if ϵ_t, the error of the base classifier, is less than $\frac{1}{2}$, then $\frac{1-\epsilon_t}{\epsilon_t} > 1$ and α_t is positive ($\alpha_t > 0$). Thus, the new distribution $\mathcal{D}_{t+1}$ is defined from $\mathcal{D}_t$ by substantially increasing the weight on i if point x_i is incorrectly classified ($y_i h_t(x_i) < 0$), and, on the contrary, decreasing it if x_i is correctly classified. This has the effect of focusing more on the points incorrectly classified at the next round of boosting, less on those correctly classified by h_t.

After T rounds of boosting, the classifier returned by AdaBoost is based on the sign of function f, which is a non-negative linear combination of the base classifiers h_t. The weight α_t assigned to h_t in that sum is a logarithmic function of the ratio of the accuracy $1 - \epsilon_t$ and error ϵ_t of h_t. Thus, more accurate base classifiers are assigned a larger weight in that sum. Figure 7.2 illustrates the AdaBoost algorithm. The size of the points represents the distribution weight assigned to them at each boosting round.

For any $t \in [T]$, we will denote by f_t the linear combination of the base classifiers after t rounds of boosting: $f_t = \sum_{s=1}^{t} \alpha_s h_s$. In particular, we have $f_T = f$. The distribution $\mathcal{D}_{t+1}$ can be expressed in terms of f_t and the normalization factors Z_s, $s \in [t]$, as follows:

$$\forall i \in [m], \quad \mathcal{D}_{t+1}(i) = \frac{e^{-y_i f_t(x_i)}}{m \prod_{s=1}^{t} Z_s}\,. \tag{7.2}$$

We will make use of this identity several times in the proofs of the following sections. It can be shown straightforwardly by repeatedly expanding the definition of the distribution over the point x_i:

$$\begin{aligned}\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t} &= \frac{\mathcal{D}_{t-1}(i) e^{-\alpha_{t-1} y_i h_{t-1}(x_i)} e^{-\alpha_t y_i h_t(x_i)}}{Z_{t-1} Z_t} \\ &= \frac{e^{-y_i \sum_{s=1}^{t} \alpha_s h_s(x_i)}}{m \prod_{s=1}^{t} Z_s}.\end{aligned}$$

The AdaBoost algorithm can be generalized in several ways:

- Instead of a hypothesis with minimal weighted error, h_t can be more generally the base classifier returned by a weak learning algorithm trained on $\mathcal{D}_t$;
- The range of the base classifiers could be $[-1, +1]$, or more generally a bounded subset of $\mathbb{R}$. The coefficients α_t can then be different and may not even admit a closed form. In general, they are chosen to minimize an upper bound on the empirical error, as discussed in the next section. Of course, in that general case, the hypotheses h_t are not binary *classifiers*, but their sign could define the label, and their magnitude could be interpreted as a measure of confidence.

In rest of this chapter, the range of the base classifiers in $\mathcal{H}$ will be assumed to be included in $[-1, +1]$. We now further analyze the properties of AdaBoost and discuss its typical use in practice.

7.2.1 Bound on the empirical error

We first show that the empirical error of AdaBoost decreases exponentially fast as a function of the number of rounds of boosting.

Theorem 7.2 *The empirical error of the classifier returned by AdaBoost verifies:*

$$\widehat{R}_S(f) \leq \exp\Big[-2\sum_{t=1}^{T}\Big(\frac{1}{2}-\epsilon_t\Big)^2\Big]. \tag{7.3}$$

Furthermore, if for all $t \in [T]$, $\gamma \leq \big(\frac{1}{2}-\epsilon_t\big)$, then

$$\widehat{R}_S(f) \leq \exp(-2\gamma^2 T)\,. \tag{7.4}$$

Proof: Using the general inequality $1_{u\leq 0} \leq \exp(-u)$ valid for all $u \in \mathbb{R}$ and identity 7.2, we can write:

$$\widehat{R}_S(f) = \frac{1}{m}\sum_{i=1}^{m} 1_{y_i f(x_i)\leq 0} \leq \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)} = \frac{1}{m}\sum_{i=1}^{m}\Big[m\prod_{t=1}^{T} Z_t\Big]\mathcal{D}_{T+1}(i) = \prod_{t=1}^{T} Z_t.$$

Since for all $t \in [T]$, Z_t is a normalization factor, it can be expressed in terms of ϵ_t by:

$$\begin{aligned} Z_t = \sum_{i=1}^{m}\mathcal{D}_t(i)e^{-\alpha_t y_i h_t(x_i)} &= \sum_{i:y_ih_t(x_i)=+1}\mathcal{D}_t(i)e^{-\alpha_t} + \sum_{i:y_ih_t(x_i)=-1}\mathcal{D}_t(i)e^{\alpha_t} \\ &= (1-\epsilon_t)e^{-\alpha_t} + \epsilon_t e^{\alpha_t} \\ &= (1-\epsilon_t)\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} + \epsilon_t\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = 2\sqrt{\epsilon_t(1-\epsilon_t)}\,. \end{aligned}$$

Thus, the product of the normalization factors can be expressed and upper bounded as follows:

$$\begin{aligned} \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_{t=1}^{T}\sqrt{1-4\big(\tfrac{1}{2}-\epsilon_t\big)^2} &\leq \prod_{t=1}^{T}\exp\Big[-2\big(\tfrac{1}{2}-\epsilon_t\big)^2\Big] \\ &= \exp\Big[-2\sum_{t=1}^{T}\big(\tfrac{1}{2}-\epsilon_t\big)^2\Big], \end{aligned}$$

where the inequality follows from the inequality $1-x \leq e^{-x}$ valid for all $x \in \mathbb{R}$. □

Note that the value of γ, which is known as the *edge*, and the accuracy of the base classifiers do not need to be known to the algorithm. The algorithm adapts to their

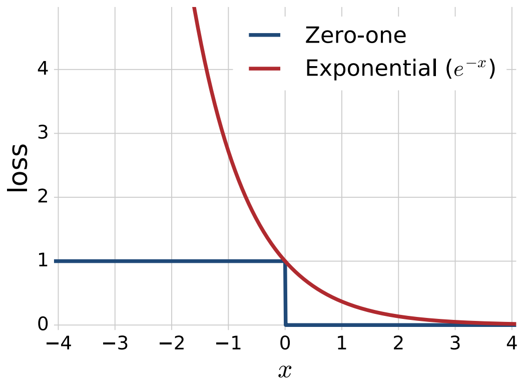


Figure 7.3
Visualization of the zero-one loss (blue) and the convex and differentiable upper bound on the zero-one loss (red) that is optimized by AdaBoost.

accuracy and defines a solution based on these values. This is the source of the extended name of AdaBoost: *adaptive boosting*.

The proof of theorem 7.2 reveals several other important properties. First, observe that α_t is the minimizer of the function $\varphi\colon \alpha \mapsto (1-\epsilon_t)e^{-\alpha} + \epsilon_t e^{\alpha}$. Indeed, φ is convex and differentiable, and setting its derivative to zero yields:

$$\varphi'(\alpha) = -(1-\epsilon_t)e^{-\alpha} + \epsilon_t e^{\alpha} = 0 \Leftrightarrow (1-\epsilon_t)e^{-\alpha} = \epsilon_t e^{\alpha} \Leftrightarrow \alpha = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}. \quad (7.5)$$

Thus, α_t is chosen to minimize $Z_t = \varphi(\alpha_t)$ and, in light of the bound $\widehat{R}_S(f) \leq \prod_{t=1}^{T} Z_t$ shown in the proof, these coefficients are selected to minimize an upper bound on the empirical error. In fact, for base classifiers whose range is $[-1,+1]$ or $\mathbb{R}$, α_t can be chosen in a similar fashion to minimize Z_t, and this is the way AdaBoost is extended to these more general cases.

Observe also that the equality $(1-\epsilon_t)e^{-\alpha_t} = \epsilon_t e^{\alpha_t}$ just shown in (7.5) implies that at each iteration, AdaBoost assigns equal distribution mass to correctly and incorrectly classified instances, since $(1-\epsilon_t)e^{-\alpha_t}$ is the total distribution assigned to correctly classified points and $\epsilon_t e^{\alpha_t}$ that of incorrectly classified ones. This may seem to contradict the fact that AdaBoost increases the weights of incorrectly classified points and decreases that of others, but there is in fact no inconsistency: the reason is that there are always fewer incorrectly classified points, since the base classifier's accuracy is better than random.

7.2.2 Relationship with coordinate descent

AdaBoost was originally designed to address the theoretical question of whether a weak learning algorithm could be used to derive a strong learning one. Here, we will show that it coincides in fact with a very simple algorithm, which consists of applying a general coordinate descent technique to a convex and differentiable objective function.

For simplicity, in this section, we assume that the base classifier set $\mathcal{H}$ is finite, with cardinality N: $\mathcal{H} = \{h_1, \ldots, h_N\}$. An ensemble function f such as the one returned by AdaBoost can then be written as $f = \sum_{j=1}^N \bar{\alpha}_j h_j$, with $\bar{\alpha}_j \geq 0$. Given a labeled sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, let F be the objective function defined for all $\bar{\boldsymbol{\alpha}} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_N) \in \mathbb{R}^N$ by

$$F(\bar{\boldsymbol{\alpha}}) = \frac{1}{m} \sum_{i=1}^m e^{-y_i f(x_i)} = \frac{1}{m} \sum_{i=1}^m e^{-y_i \sum_{j=1}^N \bar{\alpha}_j h_j(x_i)}. \tag{7.6}$$

Since the exponential loss $u \mapsto e^{-u}$ is an upper bound on the zero-one loss $u \mapsto 1_{u \leq 0}$ (see figure 7.3), F is an upper bound on the empirical error:

$$\widehat{R}_S(f) = \frac{1}{m} \sum_{i=1}^m 1_{y_i f(x_i) \leq 0} \leq \frac{1}{m} \sum_{i=1}^m e^{-y_i f(x_i)}. \tag{7.7}$$

F is a convex function of $\bar{\boldsymbol{\alpha}}$ since it is a sum of convex functions, each obtained by composition of the (convex) exponential function with an affine function of $\bar{\boldsymbol{\alpha}}$. F is also differentiable since the exponential function is differentiable. We will show that F is the objective function minimized by AdaBoost.

Different convex optimization techniques can be used to minimize F. Here, we will use a variant of the coordinate descent technique. Coordinate descent is applied over T rounds. Let $\bar{\boldsymbol{\alpha}}_0 = \mathbf{0}$ and let $\bar{\boldsymbol{\alpha}}_t$ denote the parameter vector at the end of iteration t. At each round $t \in [T]$, a direction $\mathbf{e}_k$ corresponding to the kth coordinate of $\bar{\boldsymbol{\alpha}}$ in $\mathbb{R}^N$ is selected, as well as a step size η along that direction. $\bar{\boldsymbol{\alpha}}_t$ is obtained from $\bar{\boldsymbol{\alpha}}_{t-1}$ according to the update $\bar{\boldsymbol{\alpha}}_t = \bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k$, where η is the step size chosen along the direction $\mathbf{e}_k$. Observe that if we denote by $\bar{g}_t$ the ensemble function defined by $\bar{\boldsymbol{\alpha}}_t$, that is $\bar{g}_t = \sum_{j=1}^N \bar{\alpha}_{t,j} h_j$, then the coordinate descent update coincides with the update $\bar{g}_t = \bar{g}_{t-1} + \eta h_k$, which is also the AdaBoost update. Thus, since both algorithms start with $\bar{g}_0 = 0$, to show that AdaBoost coincides with coordinate descent applied to F, it suffices to show at every iteration t, coordinate descent selects the same base hypothesis h_k and step η as AdaBoost. We will assume by induction that this holds up to iteration $t-1$, which implies the equality $\bar{g}_{t-1} = f_{t-1}$, and will show then that it also holds at iteration t.

The variant of coordinate descent we consider here consists of selecting, at each iteration, the maximum descent direction, that is the direction $\mathbf{e}_k$ along which the derivative of F is the largest in absolute value, and of selecting the best step along that direction, that is of choosing η to minimize $F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k)$. To give the expressions of the direction and the step at each iteration, we first introduce similar quantities to those appearing in the analysis of the boosting algorithm. For any $t \in [T]$, we define a distribution $\bar{\mathcal{D}}_t$ over the indices $\{1, \ldots, m\}$ as follows:

$$\bar{\mathcal{D}}_t(i) = \frac{e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1,j} h_j(x_i)}}{\bar{Z}_t} = \frac{e^{-y_i \bar{g}_{t-1}(x_i)}}{\bar{Z}_t},$$

where $\bar{Z}_t$ is the normalization factor $\bar{Z}_t = \sum_{i=1}^m e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1,j} h_j(x_i)}$. Observe that, since $\bar{g}_{t-1} = f_{t-1}$, $\bar{\mathcal{D}}_t$ coincides with $\mathcal{D}_t$. We also define for any base hypothesis h_j, $j \in [N]$, its expected error $\bar{\epsilon}_{t,j}$ with respect to the distribution $\bar{\mathcal{D}}_t$:

$$\bar{\epsilon}_{t,j} = \mathop{\mathbb{E}}_{i \sim \bar{\mathcal{D}}_t} \left[1_{y_i h_j(x_i) \leq 0}\right].$$

The directional derivative of F at $\bar{\boldsymbol{\alpha}}_{t-1}$ along $\mathbf{e}_k$ is denoted by $F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k)$ and defined by

$$F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k) = \lim_{\eta \to 0} \frac{F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k) - F(\bar{\boldsymbol{\alpha}}_{t-1})}{\eta}.$$

Since $F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k) = \sum_{i=1}^m e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1,j} h_j(x_i) - \eta y_i h_k(x_i)}$, the directional derivative along $\mathbf{e}_k$ can be expressed as follows:

$$\begin{aligned}
F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k) &= -\frac{1}{m} \sum_{i=1}^m y_i h_k(x_i) e^{-y_i \sum_{j=1}^N \bar{\alpha}_{t-1,j} h_j(x_i)} \\
&= -\frac{1}{m} \sum_{i=1}^m y_i h_k(x_i) \bar{\mathcal{D}}_t(i) \bar{Z}_t \\
&= -\left[\sum_{i=1}^m \bar{\mathcal{D}}_t(i) 1_{y_i h_k(x_i) = +1} - \sum_{i=1}^m \bar{\mathcal{D}}_t(i) 1_{y_i h_k(x_i) = -1}\right] \frac{\bar{Z}_t}{m} \\
&= -\left[(1 - \bar{\epsilon}_{t,k}) - \bar{\epsilon}_{t,k}\right] \frac{\bar{Z}_t}{m} = \left[2\bar{\epsilon}_{t,k} - 1\right] \frac{\bar{Z}_t}{m}.
\end{aligned}$$

Since $\frac{\bar{Z}_t}{m}$ does not depend on k, the maximum descent direction k is the one minimizing $\bar{\epsilon}_{t,k}$. Thus, the hypothesis h_k selected by coordinate descent at iteration t is the one with the smallest expected error on the sample S, where the expectation is taken with respect to $\bar{\mathcal{D}}_t = \mathcal{D}_t$. This matches exactly the choice made by AdaBoost at the tth round.

The step size η is selected to minimize the function along the direction $\mathbf{e}_k$ chosen: $\operatorname{argmin}_\eta F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta\, \mathbf{e}_k)$. Since $F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta\, \mathbf{e}_k)$ is a convex function of η, to find the

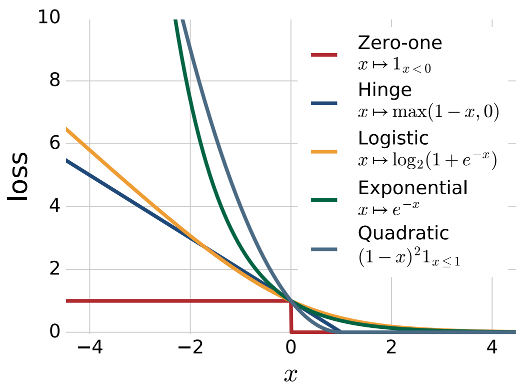


Figure 7.4
Examples of several convex upper bounds on the zero-one loss.

minimum, it suffices to set its derivative to zero:

$$\begin{aligned}
\frac{dF(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k)}{d\eta} = 0 &\Leftrightarrow -\sum_{i=1}^{m} y_i h_k(x_i) e^{-y_i \sum_{j=1}^{N} \bar{\alpha}_{t-1,j} h_j(x_i)} e^{-\eta y_i h_k(x_i)} = 0 \\
&\Leftrightarrow -\sum_{i=1}^{m} y_i h_k(x_i) \bar{\mathcal{D}}_t(i) \bar{Z}_t e^{-\eta y_i h_k(x_i)} = 0 \\
&\Leftrightarrow -\sum_{i=1}^{m} y_i h_k(x_i) \bar{\mathcal{D}}_t(i) e^{-\eta y_i h_k(x_i)} = 0 \\
&\Leftrightarrow -\Big[(1 - \bar{\epsilon}_{t,k}) e^{-\eta} - \bar{\epsilon}_{t,k} e^{\eta}\Big] = 0 \\
&\Leftrightarrow \eta = \frac{1}{2} \log \frac{1 - \bar{\epsilon}_{t,k}}{\bar{\epsilon}_{t,k}}.
\end{aligned}$$

This proves that the step size chosen by coordinate descent coincides with the weight α_t assigned by AdaBoost to the classifier chosen in the tth round. Thus, coordinate descent applied to exponential objective F precisely coincides with AdaBoost and F can be viewed as the objective function that AdaBoost seeks to minimize.

In light of this relationship, one may wish to consider similar applications of coordinate descent to other convex and differentiable functions of $\bar{\boldsymbol{\alpha}}$ upper-bounding the zero-one loss. In particular, the *logistic loss* $x \mapsto \log_2(1 + e^{-x})$ is convex and differentiable and upper bounds the zero-one loss. Figure 7.4 shows other examples of alternative convex loss functions upper-bounding the zero-one loss. Using the logistic loss, instead of the exponential loss used by AdaBoost, leads to an objective that coincides with *logistic regression*.

7.2.3 Practical use

Here, we briefly describe the standard practical use of AdaBoost. An important requirement for the algorithm is the choice of the base classifiers or that of the weak learner. The family of base classifiers typically used with AdaBoost in practice is that of *decision trees*, which are equivalent to hierarchical partitions of the space (see chapter 9, section 9.3.3). Among decision trees, those of depth one, also known as *stumps*, are by far the most frequently used base classifiers.

Boosting stumps are threshold functions associated to a single feature. Thus, a stump corresponds to a single axis-aligned partition of the space, as illustrated in figure 7.2. If the data is in $\mathbb{R}^N$, we can associate a stump to each of the N components. Thus, to determine the stump with the minimal weighted error at each round of boosting, the best component and the best threshold for each component must be computed.

To do so, we can first presort each component in $O(m \log m)$ time with a total computational cost of $O(mN \log m)$. For a given component, there are only $m+1$ possible distinct thresholds, since two thresholds between the same consecutive component values are equivalent. To find the best threshold at each round of boosting, all of these possible $m+1$ values can be compared, which can be done in $O(m)$ time. Thus, the total computational complexity of the algorithm for T rounds of boosting is $O(mN \log m + mNT)$.

Observe, however, that while boosting stumps are widely used in combination with AdaBoost and can perform well in practice, the algorithm that returns the stump with the minimal (weighted) empirical error is *not a weak learner* (see definition 7.1)! Consider, for example, the simple XOR example with four data points lying in $\mathbb{R}^2$ (see figure 6.3a), where points in the second and fourth quadrants are labeled positively and those in the first and third quadrants negatively. Then, no decision stump can achieve an accuracy better than $\frac{1}{2}$.

7.3 Theoretical results

In this section we present a theoretical analysis of the generalization properties of AdaBoost.

7.3.1 VC-dimension-based analysis

We start with an analysis of AdaBoost based on the VC-dimension of its hypothesis set. The family of functions $\mathcal{F}_T$ out of which AdaBoost selects its output after T rounds of boosting is

$$\mathcal{F}_T = \left\{ \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t \right) : \alpha_t \geq 0, h_t \in \mathcal{H}, t \in [T] \right\}. \tag{7.8}$$

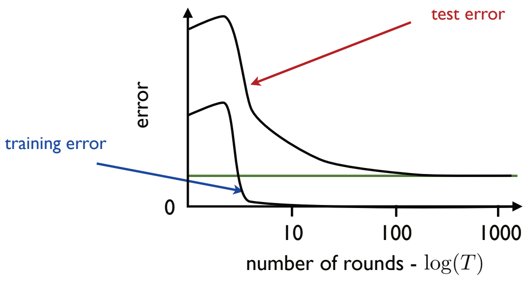


Figure 7.5
An empirical result using AdaBoost with C4.5 decision trees as base learners. In this example, the training error goes to zero after about 5 rounds of boosting ($T \approx 5$), yet the test error continues to decrease for larger values of T.

The VC-dimension of $\mathcal{F}_T$ can be bounded as follows in terms of the VC-dimension d of the family of base hypothesis $\mathcal{H}$ (exercise 7.1):

$$\text{VCdim}(\mathcal{F}_T) \leq 2(d+1)(T+1)\log_2((T+1)e)\,. \tag{7.9}$$

The upper bound grows as $O(dT\log T)$, thus, the bound suggests that AdaBoost could overfit for large values of T, and indeed this can occur. However, in many cases, it has been observed empirically that the generalization error of AdaBoost decreases as a function of the number of rounds of boosting T, as illustrated in figure 7.5! How can these empirical results be explained? The following sections present a margin-based analysis in support of AdaBoost that can serve as a theoretical explanation for these empirical observations.

7.3.2 L_1-geometric margin

In chapter 5, we introduced the definition of confidence margin and presented a series of general learning bounds based on that notion which, in particular, provided strong learning guarantees for SVMs. Here, we will similarly derive general learning bounds based on that same notion of confidence margin for ensemble methods, which we will use, in particular, to derive learning guarantees for AdaBoost.

Recall that the confidence margin of a real-valued function f at a point x labeled with y is the quantity $yf(x)$. For SVMs, we also defined the notion of geometric margin which, in the separable case, is a lower bound on the confidence margin of a linear hypothesis with a normalized weighted vector $\mathbf{w}$, $\|\mathbf{w}\|_2 = 1$. Here, we will also define a notion of geometric margin for linear hypotheses with a norm-1 constraint, such as the ensemble hypotheses returned by AdaBoost, and similarly relate that notion to that of confidence margin. This will also serve as an opportunity for us to point out the connection between several concepts and terminology used in the context of SVMs and those used in the context of boosting.

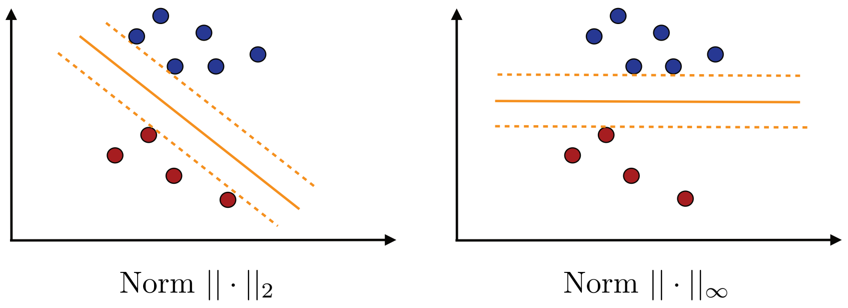


Figure 7.6
Maximum margin hyperplanes for norm-2 and norm-∞.

First note that a function $f = \sum_{t=1}^{T} \alpha_t h_t$ that is a linear combination of base hypotheses $h_1, \ldots, h_T$ can be equivalently expressed as an inner product $f = \boldsymbol{\alpha} \cdot \mathbf{h}$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_T)^\top$ and $\mathbf{h} = [h_1, \ldots, h_T]^\top$. This makes the similarity between the linear hypotheses considered in this chapter and those of chapter 5 and chapter 6 evident: the vector of base hypothesis values $\mathbf{h}(x)$ can be viewed as a feature vector associated to x, which was denoted by $\mathbf{\Phi}(x)$ in previous chapters, and $\boldsymbol{\alpha}$ is the weight vector that was denoted by $\mathbf{w}$. For ensemble linear combinations such as those returned by AdaBoost, additionally, the weight vector is non-negative: $\boldsymbol{\alpha} \geq 0$. Next, we introduce a notion of geometric margin for such ensemble functions which differs from the one introduced for SVMs only by the norm-1 used instead of norm-2, using the notation just introduced.

Definition 7.3 (L_1-geometric margin) *The* L_1-geometric margin $\rho_f(x)$ *of a linear function* $f = \sum_{t=1}^{T} \alpha_t h_t$ *with* $\boldsymbol{\alpha} \neq 0$ *at a point* $x \in \mathcal{X}$ *is defined by*

$$\rho_f(x) = \frac{|f(x)|}{\|\boldsymbol{\alpha}\|_1} = \frac{|\sum_{t=1}^{T} \alpha_t h_t(x)|}{\|\boldsymbol{\alpha}\|_1} = \frac{\big|\boldsymbol{\alpha} \cdot \mathbf{h}(x)\big|}{\|\boldsymbol{\alpha}\|_1}. \tag{7.10}$$

The L_1-margin *of* f *over a sample* $S = (x_1, \ldots, x_m)$ *is its minimum margin at the points in that sample:*

$$\rho_f = \min_{i \in [m]} \rho_f(x_i) = \min_{i \in [m]} \frac{\big|\boldsymbol{\alpha} \cdot \mathbf{h}(x_i)\big|}{\|\boldsymbol{\alpha}\|_1}. \tag{7.11}$$

This definition of geometric margin differs from definition 5.1 given in the context of the SVM algorithm only by the norm used for the weight vector: L_1-norm here, L_2-norm in definition 5.1. To distinguish them in the discussion that follows, let $\rho_1(x)$ denote the L_1-margin and $\rho_2(x)$ the L_2-margin at point x (definition 5.1):

$$\rho_1(x) = \frac{|\boldsymbol{\alpha} \cdot \mathbf{h}(x)|}{\|\boldsymbol{\alpha}\|_1} \quad \text{and} \quad \rho_2(x) = \frac{|\boldsymbol{\alpha} \cdot \mathbf{h}(x)|}{\|\boldsymbol{\alpha}\|_2}.$$

$\rho_2(x)$ is then the norm-2 distance of the vector $\mathbf{h}(x)$ to the hyperplane of equation $\boldsymbol{\alpha}\cdot\mathbf{x} = 0$ in $\mathbb{R}^T$. Similarly, $\rho_1(x)$ is the norm-∞ distance of $\mathbf{h}(x)$ to that hyperplane. This geometric difference is illustrated by figure 7.6.[8]

We will denote by

$$\bar{f} = \frac{f}{\sum_{t=1}^{T} \alpha_t} = \frac{f}{\|\boldsymbol{\alpha}\|_1}$$

the normalized version of the function f returned by AdaBoost. Note that if a point x with label y is correctly classified by f (or $\bar{f}$), then the confidence margin of $\bar{f}$ at x coincides with the L_1-geometric margin of f: $y\bar{f}(x) = \frac{yf(x)}{\|\boldsymbol{\alpha}\|_1} = \rho_f(x)$. Observe that, since the coefficients α_t are non-negative, $\rho_f(x)$ is then a convex combination of the base hypothesis values $h_t(x)$. In particular, if the base hypotheses h_t take values in $[-1, +1]$, then $\rho_f(x)$ is in $[-1, +1]$.

7.3.3 Margin-based analysis

To analyze the generalization properties of AdaBoost, we start by examining the Rademacher complexity of convex linear ensembles. For any hypothesis set $\mathcal{H}$ of real-valued functions, we denote by $\operatorname{conv}(\mathcal{H})$ its convex hull defined by

$$\operatorname{conv}(\mathcal{H}) = \left\{ \sum_{k=1}^{p} \mu_k h_k \colon p \geq 1, \forall k \in [p], \mu_k \geq 0, h_k \in \mathcal{H}, \sum_{k=1}^{p} \mu_k \leq 1 \right\}. \tag{7.12}$$

The following lemma shows that, remarkably, the empirical Rademacher complexity of $\operatorname{conv}(\mathcal{H})$, which in general is a strictly larger set including $\mathcal{H}$, coincides with that of $\mathcal{H}$.

Lemma 7.4 *Let $\mathcal{H}$ be a set of functions mapping from $\mathcal{X}$ to $\mathbb{R}$. Then, for any sample S, we have*

$$\widehat{\mathfrak{R}}_S\big(\operatorname{conv}(\mathcal{H})\big) = \widehat{\mathfrak{R}}_S(\mathcal{H})\,.$$

[8] More generally, for $p, q \geq 1$, p and q *conjugate*, that is $\frac{1}{p} + \frac{1}{q} = 1$, $\frac{|\boldsymbol{\alpha}\cdot\mathbf{h}(x)|}{\|\boldsymbol{\alpha}\|_p}$ is the norm-q distance of $\mathbf{h}(x)$ to the hyperplane of equation $\boldsymbol{\alpha}\cdot\mathbf{h}(x) = 0$.

Proof: The proof follows from a straightforward series of equalities:

$$\begin{aligned}
\widehat{\mathfrak{R}}_S\big(\operatorname{conv}(\mathcal{H})\big) &= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\Bigg[\sup_{h_1,\ldots,h_p\in\mathcal{H},\boldsymbol{\mu}\geq 0,\|\boldsymbol{\mu}\|_1\leq 1}\sum_{i=1}^{m}\sigma_i\sum_{k=1}^{p}\mu_k h_k(x_i)\Bigg]\\
&= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\Bigg[\sup_{h_1,\ldots,h_p\in\mathcal{H}}\ \sup_{\boldsymbol{\mu}\geq 0,\|\boldsymbol{\mu}\|_1\leq 1}\sum_{k=1}^{p}\mu_k\sum_{i=1}^{m}\sigma_i h_k(x_i)\Bigg]\\
&= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\Bigg[\sup_{h_1,\ldots,h_p\in\mathcal{H}}\ \max_{k\in[p]}\sum_{i=1}^{m}\sigma_i h_k(x_i)\Bigg]\\
&= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\Bigg[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\sigma_i h(x_i)\Bigg] = \widehat{\mathfrak{R}}_S(\mathcal{H}),
\end{aligned}$$

where the third equality follows the definition of the dual norm (see section A.1.2) or the observation that the maximizing vector $\boldsymbol{\mu}$ for a convex combination of p terms is the one placing all the weight on the largest term. □

This theorem can be used directly in combination with theorem 5.8 to derive the following Rademacher complexity generalization bound for convex combination ensembles of hypotheses. Recall that $\widehat{R}_{S,\rho}(h)$ denotes the empirical margin loss with margin ρ.

Corollary 7.5 (Ensemble Rademacher margin bound) *Let $\mathcal{H}$ denote a set of real-valued functions. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1-\delta$, each of the following holds for all $h \in \operatorname{conv}(\mathcal{H})$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\mathfrak{R}_m\big(\mathcal{H}\big) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}} \tag{7.13}$$

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\widehat{\mathfrak{R}}_S\big(\mathcal{H}\big) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}}\,. \tag{7.14}$$

Using corollary 3.8 and corollary 3.18 to bound the Rademacher complexity in terms of the VC-dimension yields immediately the following VC-dimension-based generalization bounds for convex combination ensembles of hypotheses.

Corollary 7.6 (Ensemble VC-Dimension margin bound) *Let $\mathcal{H}$ be a family of functions taking values in $\{+1,-1\}$ with VC-dimension d. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following holds for all $h \in \operatorname{conv}(\mathcal{H})$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\sqrt{\frac{2d\log\frac{em}{d}}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}\,. \tag{7.15}$$

These bounds can be generalized to hold uniformly for all $\rho \in (0,1]$, at the price of an additional term of the form $\sqrt{\frac{\log\log_2\frac{2}{\delta}}{m}}$ as in theorem 5.9. They cannot be directly applied to the function f returned by AdaBoost, since it is not a convex combination of base hypotheses, but they can be applied to its normalized version, $\bar{f} = \frac{\sum_{t=1}^T \alpha_t h_t}{\|\boldsymbol{\alpha}\|_1} \in \operatorname{conv}(\mathcal{H})$. Notice that from the point of view of binary classification, f and $\bar{f}$ are equivalent since $\operatorname{sgn}(f) = \operatorname{sgn}\big(\frac{f}{\|\boldsymbol{\alpha}\|_1}\big)$, thus $R(f) = R\big(\frac{f}{\|\boldsymbol{\alpha}\|_1}\big)$, but their empirical margin losses are distinct.

Let $f = \sum_{t=1}^T \alpha_t h_t$ denote the function defining the classifier returned by AdaBoost after T rounds of boosting when trained on sample S. Then, in view of (7.13), for any $\delta > 0$, the following holds with probability at least $1 - \delta$:

$$R(f) \leq \widehat{R}_{S,\rho}\left(\bar{f}\right) + \frac{2}{\rho}\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}. \tag{7.16}$$

Similar bounds can be derived from (7.14) and (7.15). Remarkably, the number of rounds of boosting T does not appear in the generalization bound (7.16). The bound depends only on the confidence margin ρ, the sample size m, and the Rademacher complexity of the family of base classifiers $\mathcal{H}$. Thus, the bound guarantees an effective generalization if the margin loss $R_\rho(\bar{f})$ is small for a relatively large ρ. Recall that the margin loss can be upper bounded by the fraction of the points x labeled with y in the training sample with confidence margin at most ρ, that is $\frac{yf(x)}{\|\boldsymbol{\alpha}\|_1} \leq \rho$ (see (5.38)). With our definition of L_1-margin, this can also be written as follows:

$$\widehat{R}_{S,\rho}\left(\bar{f}\right) \leq \frac{|\{i \in [m] \colon y_i\, \rho_f(x_i) \leq \rho\}|}{m}. \tag{7.17}$$

Additionally, the following theorem provides a bound on the empirical margin loss, which decreases with T under conditions discussed later.

Theorem 7.7 *Let $f = \sum_{t=1}^T \alpha_t h_t$ denote the function returned by AdaBoost after T rounds of boosting and assume for all $t \in [T]$ that $\epsilon_t < \frac{1}{2}$, which implies $\alpha_t > 0$. Then, for any $\rho > 0$, the following holds:*

$$\widehat{R}_{S,\rho}(\bar{f}) \leq 2^T \prod_{t=1}^T \sqrt{\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}}\,.$$

Proof: Using the general inequality $1_{u \leq 0} \leq \exp(-u)$ valid for all $u \in \mathbb{R}$, identity 7.2, that is $\mathcal{D}_{t+1}(i) = \frac{e^{-y_i f(x_i)}}{m \prod_{t=1}^T Z_t}$, the equality $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$ from the proof of theorem 7.2, and the definition of $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$ in AdaBoost, we can write:

$$\begin{aligned}
\frac{1}{m}\sum_{i=1}^{m} 1_{y_i f(x_i)-\rho\|\boldsymbol{\alpha}\|_1 \leq 0} &\leq \frac{1}{m}\sum_{i=1}^{m} \exp(-y_i f(x_i) + \rho\|\boldsymbol{\alpha}\|_1) \\
&= \frac{1}{m}\sum_{i=1}^{m} e^{\rho\|\boldsymbol{\alpha}\|_1}\left[m\prod_{t=1}^{T} Z_t\right]\mathcal{D}_{T+1}(i) \\
&= e^{\rho\|\boldsymbol{\alpha}\|_1}\prod_{t=1}^{T} Z_t = e^{\rho\sum_{t'}\alpha_{t'}}\prod_{t=1}^{T} Z_t \\
&= 2^T \prod_{t=1}^{T}\left[\sqrt{\tfrac{1-\epsilon_t}{\epsilon_t}}\right]^{\rho}\sqrt{\epsilon_t(1-\epsilon_t)}\,,
\end{aligned}$$

which concludes the proof. □

Moreover, if for all $t \in [T]$ we have $\gamma \leq (\frac{1}{2} - \epsilon_t)$ and $\rho \leq 2\gamma$, then the expression $4\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}$ is maximized at $\epsilon_t = \frac{1}{2} - \gamma$.[9] Thus, the upper bound on the empirical margin loss can then be bounded by

$$\widehat{R}_{S,\rho}(\bar{f}) \leq \left[(1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho}\right]^{\frac{T}{2}}. \tag{7.18}$$

Observe that $(1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho} = (1-4\gamma^2)\big(\frac{1+2\gamma}{1-2\gamma}\big)^{\rho}$. This is an increasing function of ρ since we have $\big(\frac{1+2\gamma}{1-2\gamma}\big) > 1$ as a consequence of $\gamma > 0$. Thus, if $\rho < \gamma$, it can be strictly upper bounded as follows

$$(1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho} < (1-2\gamma)^{1-\gamma}(1+2\gamma)^{1+\gamma}.$$

The function $\gamma \mapsto (1-2\gamma)^{1-\gamma}(1+2\gamma)^{1+\gamma}$ is strictly upper bounded by 1 over the interval $(0, 1/2)$, thus, if $\rho < \gamma$, then $(1-2\gamma)^{1-\rho}(1+2\gamma)^{1+\rho} < 1$ and the right-hand side of (7.18) decreases exponentially with T. Since the condition $\rho \gg O(1/\sqrt{m})$ is necessary in order for the given margin bounds to converge, this places a condition of $\gamma \gg O(1/\sqrt{m})$ on the edge value. In practice, the error ϵ_t of the base classifier at round t may increase as a function of t. Informally, this is because boosting presses the weak learner to concentrate on instances that are harder and harder to classify, for which even the best base classifier could not achieve an error significantly better than random. If ϵ_t becomes close to $\frac{1}{2}$ relatively fast as a function of t, then the bound of theorem 7.7 becomes uninformative.

[9] The differential of $f\colon \epsilon \mapsto \log[\epsilon^{1-\rho}(1-\epsilon)^{1+\rho}] = (1-\rho)\log\epsilon + (1+\rho)\log(1-\epsilon)$ over the interval $(0,1)$ is given by $f'(\epsilon) = \frac{1-\rho}{\epsilon} - \frac{1+\rho}{1-\epsilon} = 2\frac{(\frac{1}{2}-\frac{\rho}{2})-\epsilon}{\epsilon(1-e)}$. Thus, f is an increasing function over $(0, \frac{1}{2}-\frac{\rho}{2})$, which implies that it is increasing over $(0, \frac{1}{2}-\gamma)$ when $\gamma \geq \frac{\rho}{2}$.

The analysis and discussion that precede show that if AdaBoost admits a positive edge ($\gamma > 0$), then, for $\rho < \gamma$, the empirical margin loss $\widehat{R}_{S,\rho}(\bar{f})$ becomes zero for T sufficiently large (it decreases exponentially fast). Thus, AdaBoost achieves an L_1-geometric margin of γ over the training sample. In section 7.3.5, we will see that the edge γ is positive if and only if the training sample is separable. In that case, the edge can be chosen to be as large as half the maximum L_1-geometric margin $\rho_{\max}$ that can be achieved on the sample: $\gamma = \frac{\rho_{\max}}{2}$. Thus, for a separable data set, AdaBoost can asymptotically achieve a geometric margin that is at least half the maximum geometric margin, $\frac{\rho_{\max}}{2}$.

This analysis can serve as a theoretical explanation of the empirical observation that, in some tasks, the generalization error decreases as a function of T even after the error on the training sample is zero: the geometric margin continues to increase when the training sample is separable. In (7.16), for the ensemble function f determined by AdaBoost after T rounds, as T increases, ρ can be chosen as a larger quantity for which the first term on the right-hand side vanishes ($\widehat{R}_{S,\rho}(\bar{f}) = 0$) while the second term becomes more favorable since it decreases as $\frac{1}{\rho}$.

But, does AdaBoost achieve the maximum L_1-geometric margin $\rho_{\max}$? No. It has been shown that AdaBoost may converge, for a linearly separable sample, to a geometric margin that is significantly smaller than the maximum margin (e.g., $\frac{1}{3}$ instead of $\frac{3}{8}$).

7.3.4 Margin maximization

In view of these results, several algorithms have been devised with the explicit goal of maximizing the L_1-geometric margin. These algorithms correspond to different methods for solving a linear program (LP).

By definition of the L_1-margin, the maximum margin for a linearly separable sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$ is given by

$$\rho = \max_{\boldsymbol{\alpha}} \min_{i \in [m]} \frac{y_i\big(\boldsymbol{\alpha} \cdot \mathbf{h}(x_i)\big)}{\|\boldsymbol{\alpha}\|_1} . \tag{7.19}$$

By definition of the maximization, the optimization problem can be written as:

$$\max_{\boldsymbol{\alpha}} \ \rho$$
$$\text{subject to: } \frac{y_i\big(\boldsymbol{\alpha} \cdot \mathbf{h}(x_i)\big)}{\|\boldsymbol{\alpha}\|_1} \geq \rho, \ \forall i \in [m].$$

Since $\frac{\boldsymbol{\alpha} \cdot \mathbf{h}(x_i)}{\|\boldsymbol{\alpha}\|_1}$ is invariant to the scaling of $\boldsymbol{\alpha}$, we can restrict ourselves to $\|\boldsymbol{\alpha}\|_1 = 1$. Further seeking a non-negative $\boldsymbol{\alpha}$ as in the case of AdaBoost leads to the following optimization:

$$\begin{aligned}
&\max_{\boldsymbol{\alpha}} \ \rho \\
&\text{subject to: } y_i\big(\boldsymbol{\alpha} \cdot \mathbf{h}(x_i)\big) \geq \rho, \ \forall i \in [m]; \\
&\qquad \left(\sum_{t=1}^{T} \alpha_t = 1\right) \wedge \Big(\alpha_t \geq 0, \forall t \in [T]\Big).
\end{aligned}$$

This is a linear program (LP), that is, a convex optimization problem with a linear objective function and linear constraints. There are several different methods for solving relative large LPs in practice, using the simplex method, interior-point methods, or a variety of special-purpose solutions.

Note that the solution of this algorithm differs from the margin-maximization defining SVMs in the separable case only by the definition of the geometric margin used (L_1 versus L_2) and the non-negativity constraint on the weight vector. Figure 7.6 illustrates the margin-maximizing hyperplanes found using these two distinct margin definitions in a simple case. The left figure shows the SVM solution, where the distance to the closest points to the hyperplane is measured with respect to the norm $\|\cdot\|_2$. The right figure shows the solution for the L_1-margin, where the distance to the closest points to the hyperplane is measured with respect to the norm $\|\cdot\|_\infty$.

By definition, the solution of the LP just described admits an L_1-margin that is larger or equal to that of the AdaBoost solution. However, empirical results do not show a systematic benefit for the solution of the LP. In fact, it appears that in many cases, AdaBoost outperforms that algorithm. The margin theory described does not seem sufficient to explain that performance.

7.3.5 Game-theoretic interpretation

In this section, we show that AdaBoost admits a natural game-theoretic interpretation. The application of von Neumann's theorem then helps us relate the maximum margin and the optimal edge and clarify the connection of AdaBoost's weak-learning assumption with the notion of L_1-margin. We first introduce the definition of the edge of a base classifier for a particular distribution.

Definition 7.8 *The* edge of a base classifier h_t for a distribution $\mathcal{D}$ *over the training sample* $S = ((x_1, y_1), \ldots, (x_m, y_m))$ *is defined by*

$$\gamma_t(\mathcal{D}) = \frac{1}{2} - \epsilon_t = \frac{1}{2} \sum_{i=1}^{m} y_i h_t(x_i) \mathcal{D}(i). \tag{7.20}$$

AdaBoost's weak learning condition can now be formulated as follows: there exists $\gamma > 0$ such that for any distribution $\mathcal{D}$ over the training sample and any base

Table 7.1
The loss matrix for the standard rock-paper-scissors game.

	rock	paper	scissors
rock	0	+1	-1
paper	-1	0	+1
scissors	+1	-1	0

classifier h_t, the following holds:

$$\gamma_t(\mathcal{D}) \geq \gamma. \tag{7.21}$$

This condition is required for the analysis of theorem 7.2 and the non-negativity of the coefficients α_t. We will frame boosting as a two-person zero-sum game.

Definition 7.9 (Zero-sum game) *A* finite two-person zero-sum game *consists of a* loss matrix $\mathbf{M} \in \mathbb{R}^{m\times n}$*, where m is the number of possible* actions *(or* pure strategies*) for the row player and n the number of possible actions for the column player. The entry M_{ij} is the loss for the row player (or equivalently the payoff for the column payer) when the row player takes action i and the column player takes action j.*[10]

An example of a loss matrix for the familiar "rock-paper-scissors" game is shown in table 7.1.

Definition 7.10 (Mixed strategy) *A* mixed strategy *for the row player is a distribution p over the m possible row actions; a* mixed strategy *for the column player is a distribution q over the n possible column actions. The* expected loss *for the row player (expected payoff for the column player) with respect to the mixed strategies p and q is*

$$\operatorname*{\mathbb{E}}_{\substack{i\sim p\\ j\sim q}}[M_{ij}] = \sum_{i=1}^{m}\sum_{j=1}^{n} p_i M_{ij} q_j = \mathbf{p}^\top \mathbf{M}\mathbf{q}.$$

The following is a fundamental result in game theory proven in chapter 8.

Theorem 7.11 (Von Neumann's minimax theorem) *For any finite two-person zero-sum game defined by the matrix $\mathbf{M}$, the following equality holds:*

$$\min_{\mathbf{p}} \max_{\mathbf{q}} \mathbf{p}^\top \mathbf{M}\mathbf{q} = \max_{\mathbf{q}} \min_{\mathbf{p}} \mathbf{p}^\top \mathbf{M}\mathbf{q}\,. \tag{7.22}$$

The common value in (7.22) is called the *value of the game*. The theorem states that for any two-person zero-sum game, there exists a mixed strategy for each player

[10] To be consistent with the results discussed in other chapters, we consider the loss matrix as opposed to the payoff matrix (its opposite).

such that the expected loss for one is the same as the expected payoff for the other, both of which are equal to the value of the game.

Note that, given the row player's strategy, the column player can choose a pure strategy optimizes their payoff. That is, the column player can choose the single strategy corresponding the largest coordinate of the vector $\mathbf{p}^\top \mathbf{M}$. A similar comment applies to the reverse. Thus, an alternative and equivalent form of the minimax theorem is

$$\min_{\mathbf{p}} \max_{j \in [n]} \mathbf{p}^\top \mathbf{M} \mathbf{e}_j = \max_{\mathbf{q}} \min_{i \in [m]} \mathbf{e}_i^\top \mathbf{M} \mathbf{q}, \tag{7.23}$$

where $\mathbf{e}_i$ denotes the ith unit vector.

We can now view AdaBoost as a zero-sum game, where an action of the row player is the selection of a training instance x_i, $i \in [m]$, and an action of the column player the selection of a base learner h_t, $t \in [T]$. A mixed strategy for the row player is thus a distribution $\mathcal{D}$ over the training points' indices $[m]$. A mixed strategy for the column player is a distribution over the based classifiers' indices $[T]$. This can be defined from a non-negative vector $\boldsymbol{\alpha} \geq 0$: the weight assigned to $t \in [T]$ is $\alpha_t / \|\boldsymbol{\alpha}\|_1$. The loss matrix $\mathbf{M} \in \{-1, +1\}^{m \times T}$ for AdaBoost is defined by $M_{it} = y_i h_t(x_i)$ for all $(i, t) \in [m] \times [T]$. By von Neumann's theorem (7.23), the following holds:

$$\min_{\mathcal{D} \in \mathscr{D}} \max_{t \in [T]} \sum_{i=1}^{m} \mathcal{D}(i) y_i h_t(x_i) = \max_{\boldsymbol{\alpha} \geq 0} \min_{i \in [m]} \sum_{t=1}^{T} \frac{\alpha_t}{\|\boldsymbol{\alpha}\|_1} y_i h_t(x_i), \tag{7.24}$$

where $\mathscr{D}$ denotes the set of all distributions over the training sample. Let $\rho_{\boldsymbol{\alpha}}(x)$ denote the margin of point x for the classifier defined by $f = \sum_{t=1}^{T} \alpha_t h_t$. The result can be rewritten as follows in terms of the margins and edges:

$$2\gamma^* = 2 \min_{\mathcal{D}} \max_{t \in [T]} \gamma_t(\mathcal{D}) = \max_{\boldsymbol{\alpha}} \min_{i \in [m]} \rho_{\boldsymbol{\alpha}}(x_i) = \rho^*, \tag{7.25}$$

where ρ^* is the maximum margin of a classifier and γ^* the best possible edge. This result has several implications. First, it shows that the weak learning condition ($\gamma^* > 0$) implies $\rho^* > 0$ and thus the existence of a classifier with positive margin, which motivates the search for a non-zero margin. AdaBoost can be viewed as an algorithm seeking to achieve such a non-zero margin, though, as discussed earlier, AdaBoost does not always achieve an optimal margin and is thus suboptimal in that respect. Furthermore, we see that the "weak learning" assumption, which originally appeared to be the weakest condition one could require for an algorithm (that of performing better than random), is in fact a strong condition: it implies that the training sample is linearly separable with margin $2\gamma^* > 0$. Linear separability often does not hold for the data sets found in practice.

7.4 L_1-regularization

In practice, the training sample may not be linearly separable and AdaBoost may not admit a positive edge, in which case the weak learning condition does not hold. It may also be that AdaBoost does admit a positive edge but with γ very small. In such cases, running AdaBoost may result in large total mixture weights for some base classifiers h_j. This can be because the algorithm increasingly concentrates on a few examples that are hard to classify and whose weights keep growing. Only a few base classifiers might achieve the best performance for those examples and the algorithm keeps selecting them, thereby increasing their total mixture weights. These base classifiers with relatively large total mixture weight end up dominating in an ensemble f and therefore solely dictating the classification decision. The performance of the resulting ensemble is typically poor since it almost entirely hinges on that of a few base classifiers.

There are several methods for avoiding such situations. One consists of limiting the number of rounds of boosting T, which is also known as *early-stopping*. Another one consists of controlling the magnitude of the mixture weights. This can be done by augmenting the objective function of AdaBoost with a regularization term based on a norm of the vector of mixture weights. Using a norm-1 regularization leads to an algorithm that we will refer to as *L_1-regularized AdaBoost*. Given a labeled sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, the objective function G minimized by L_1-regularized AdaBoost is defined for all $\bar{\boldsymbol{\alpha}} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_N) \in \mathbb{R}^N$ by

$$G(\bar{\boldsymbol{\alpha}}) = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(x_i)} + \lambda\|\bar{\boldsymbol{\alpha}}\|_1 = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{N} \bar{\alpha}_j h_j(x_i)} + \lambda\|\bar{\boldsymbol{\alpha}}\|_1, \tag{7.26}$$

where, as for AdaBoost, f is an ensemble function defined by $f = \sum_{j=1}^{N} \bar{\alpha}_j h_j$, with $\bar{\alpha}_j \geq 0$. The objective function G is a convex function of $\bar{\boldsymbol{\alpha}}$ as the sum of the convex objective of AdaBoost and the norm-1 of $\bar{\boldsymbol{\alpha}}$. L_1-regularized AdaBoost consists of applying coordinate-descent to the objective function G.

We now show that the algorithm can be directed derived from the margin-based guarantee for ensemble methods of Corollary 7.5 or Corollary 7.6. Thus, in that way, L_1-regularized AdaBoost benefits from a more favorable and natural theoretical guarantee than AdaBoost.

By the generalization of Corollary 7.5 to a uniform convergence bound over ρ, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for all ensemble functions $f = \sum_{j=1}^{N} \bar{\alpha}_j h_j$ with $\|\bar{\boldsymbol{\alpha}}\|_1 \leq 1$ and all $\rho \in (0, 1]$:

$$R(f) \leq \frac{1}{m}\sum_{i=1}^{m} 1_{f(x_i) \leq \rho} + \frac{2}{\rho}\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log \log_2 \frac{2}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{7.27}$$

The inequality also trivially holds for $\rho > 1$ since, in that case, the first term on the right-hand side of the bound is equal to one. Indeed, in that case, by Hölder's inequality, for any $x \in \mathcal{X}$, we have $f(x) = \sum_{j=1}^{N} \bar{\alpha}_i h_j(x) \leq \|\bar{\boldsymbol{\alpha}}\|_1 \max_{j \in [N]} |h_j(x)| \leq \|\bar{\boldsymbol{\alpha}}\|_1 \leq 1 < \rho$.

Now, in view of the general upper bound $1_{u \leq 0} \leq e^{-u}$ valid for all $u \in \mathbb{R}$, with probability at least $1 - \delta$, the following holds for all $f = \sum_{j=1}^{N} \bar{\alpha}_j h_j$ with $\|\bar{\boldsymbol{\alpha}}\|_1 \leq 1$ and all $\rho > 0$:

$$R(f) \leq \frac{1}{m} \sum_{i=1}^{m} e^{1 - \frac{f(x_i)}{\rho}} + \frac{2}{\rho} \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log \log_2 \frac{2}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{7.28}$$

Since for any $\rho > 0$, f/ρ admits the same generalization error as f, with probability at least $1 - \rho$, the following inequality holds for all $f = \sum_{j=1}^{N} \bar{\alpha}_j h_j$ with $\|\bar{\boldsymbol{\alpha}}\|_1 \leq 1/\rho$ and all $\rho > 0$:

$$R(f) \leq \frac{1}{m} \sum_{i=1}^{m} e^{1 - f(x_i)} + \frac{2}{\rho} \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log \log_2 \frac{2}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{7.29}$$

This inequality can be used to derive an algorithm that selects $\bar{\boldsymbol{\alpha}}$ and $\rho > 0$ to minimize the right-hand side. The minimization with respect to ρ does not lead to a convex optimization and depends on theoretical constant factors affecting the second and third terms, which may not be optimal. Thus, instead, ρ is left as a free parameter of the algorithm, typically determined via cross-validation.

Now, since only the first term of the right-hand side depends on $\bar{\boldsymbol{\alpha}}$, the bound suggests selecting $\bar{\boldsymbol{\alpha}}$ as the solution of the following optimization problem:

$$\min_{\|\bar{\boldsymbol{\alpha}}\|_1 \leq \frac{1}{\rho}} \frac{1}{m} \sum_{i=1}^{m} e^{-f(x_i)} = \frac{1}{m} \sum_{i=1}^{m} e^{-\sum_{j=1}^{N} \bar{\alpha}_j h_j(x_i)}. \tag{7.30}$$

Introducing a Lagrange variable $\lambda \geq 0$, the optimization problem can be written equivalently as

$$\min_{\|\bar{\boldsymbol{\alpha}}\|_1 \leq \frac{1}{\rho}} \frac{1}{m} \sum_{i=1}^{m} e^{-\sum_{j=1}^{N} \bar{\alpha}_j h_j(x_i)} + \lambda \|\bar{\boldsymbol{\alpha}}\|_1. \tag{7.31}$$

Since for any choice of ρ in the constraint of (7.30) there exists an equivalent dual variable λ in the formulation (7.31) that achieves the same optimal $\bar{\boldsymbol{\alpha}}$, $\lambda \geq 0$ can be freely selected via cross-validation. The resulting objective function therefore precisely coincides with that of L_1-regularized AdaBoost.

7.5 Discussion

AdaBoost offers several advantages: it is simple, its implementation is straightforward, and the time complexity of each round of boosting as a function of the sample size is rather favorable. As already discussed, when using decision stumps, the time complexity of each round of boosting is in $O(mN)$. Of course, if the dimension of the feature space N is very large, then the algorithm could become in fact quite slow.

AdaBoost additionally benefits from a rich theoretical analysis. Nevertheless, there are still many theoretical questions related to the algorithm. For example, as we saw, the algorithm in fact does not maximize the margin, and yet algorithms that do maximize the margin do not always outperform it. This suggests that perhaps a finer analysis based on a notion different from that of minimal margin could shed more light on the properties of the algorithm.

A minor drawback of the algorithm is the need to select the parameter T and the base classifier set. The choice of the number of rounds of boosting T (stopping criterion) is crucial to the performance of the algorithm. As suggested by the VC-dimension analysis, larger values of T can lead to overfitting. In practice, T is typically determined via cross-validation. The choice of the base classifiers is also crucial. The complexity of the family of base classifiers $\mathcal{H}$ appeared in all the bounds presented and it is important to control it in order to guarantee generalization. On the other hand, insufficiently complex hypothesis sets could lead to low margins.

Probably the most serious disadvantage of AdaBoost is its performance in the presence of noise, at least in some tasks. The distribution weight assigned to examples that are harder to classify substantially increases with the number of rounds of boosting, by the nature of the algorithm. These examples may end up dominating the selection of the base classifiers, which, with a large enough number of rounds, will play a detrimental role in the definition of the linear combination defined by AdaBoost. Several solutions have been proposed to address these issues. One consists of using a "less aggressive" objective function than the exponential function of AdaBoost, such as the logistic loss, to penalize less incorrectly classified points. Another solution is based on a regularization, e.g., the L_1-regularized AdaBoost described in the previous section.

An empirical study of AdaBoost has shown that uniform noise severely damages its accuracy. This has also been corroborated by recent theoretical results showing that boosting algorithms based on convex potentials do not tolerate even low levels of random noise. Moreover, these issues have been shown to persist even when using L_1-regularization or early stopping. However, the uniform noise model used in those experiments or analysis is rather unrealistic and seems unlikely to appear in practice. The model assumes that a label corruption with some fixed probability affects all instances uniformly. Clearly, the performance of any algorithm should degrade in the presence of such noise. Empirical results suggest, however, that the performance of AdaBoost tends to degrade more than that of other algorithms for this uniform noise model.

Finally, notice that the behavior of AdaBoost in the presence of noise can be used, in fact, as a useful feature for detecting *outliers*, that is, examples that are incorrectly labeled or that are hard to classify. Examples with large weights after a certain number of rounds of boosting can be identified as outliers.

7.6 Chapter notes

The question of whether a weak learning algorithm could be *boosted* to derive a strong learning algorithm was first posed by Kearns and Valiant [1988, 1994], who also gave a negative proof of this result for a distribution-dependent setting. The first positive proof of this result in a distribution-independent setting was given by Schapire [1990], and later by Freund [1990].

These early boosting algorithms, boosting by filtering [Schapire, 1990] or boosting by majority [Freund, 1990, 1995] were not practical. The AdaBoost algorithm introduced by Freund and Schapire [1997] solved several of these practical issues. Freund and Schapire [1997] further gave a detailed presentation and analysis of the algorithm including the bound on its empirical error, a VC-dimension analysis, and its applications to multi-class classification and regression.

Early experiments with AdaBoost were carried out by Drucker, Schapire, and Simard [1993], who gave the first implementation in OCR with weak learners based on neural networks and Drucker and Cortes [1995], who reported the empirical performance of AdaBoost combined with decision trees, in particular decision stumps.

The fact that AdaBoost coincides with coordinate descent applied to an exponential objective function was later shown by Duffy and Helmbold [1999], Mason et al. [1999], and Friedman [2000]. Friedman, Hastie, and Tibshirani [2000] also gave an interpretation of boosting in terms of additive models. They also pointed out the close connections between AdaBoost and logistic regression, in particular the fact that their objective functions have a similar behavior near zero or the fact that their expectation admit the same minimizer, and derived an alternative boosting algorithm, LogitBoost, based on the logistic loss. Lafferty [1999] showed how an incremental family of algorithms, including LogitBoost, can be derived from Bregman divergences and designed to closely approximate AdaBoost when varying a parameter. Kivinen and Warmuth [1999] gave an equivalent view of AdaBoost as an entropy projection. They showed that the distribution over the sample found by Adaboost at each round is approximately the solution to the problem of finding the closest distribution to the one at the previous round, subject to the constraint that it be orthogonal to the vector of errors of the current base hypotheses. Here, closeness is measured by a Bregman divergence, which, for AdaBoost is the unnormalized relative entropy. Collins, Schapire, and Singer [2002] later showed that boosting and logistic regression were special instances of a common framework based on Bregman divergences and used that to give the first convergence proof of AdaBoost. Another direct relationship between AdaBoost and logistic regression is given by Lebanon and Lafferty [2001] who showed that the two algorithms minimize the same extended relative entropy objective function subject to the same feature constraints, except from an additional normalization constraint for logistic regression.

A margin-based analysis of AdaBoost was first presented by Schapire, Freund, Bartlett, and Lee [1997], including theorem 7.7 which gives a bound on the empirical margin loss. Our presentation is based on the elegant derivation of margin bounds by Koltchinskii and Panchenko [2002] using the notion of Rademacher complexity. Rudin et al. [2004] gave an example showing that, in general, AdaBoost does not maximize the L_1-margin. Rätsch and Warmuth [2002] provided asymptotic lower bounds for the margin achieved by AdaBoost under some conditions. The L_1-margin maximization based on an LP is due to Grove and Schuurmans [1998]. Rätsch, Onoda, and Müller [2001] suggested a modification of that algorithm using a soft-margin instead and pointed out its connections with SVMs. The game-theoretic interpretation of boosting and the application of von Neumann's minimax theorem [von Neumann, 1928] in that context were pointed out by Freund and Schapire [1996, 1999b]; see also Grove and Schuurmans [1998] and Breiman [1999].

The L_1-regularized AdaBoost algorithm described in Section 7.4 is presented and analyzed by Rätsch, Mika, and Warmuth [2001]. Cortes, Mohri, and Syed [2014] introduced a new boosting algorithm, *DeepBoost*, which they proved to benefit from finer learning guarantees, including favorable ones even when using as base classifier set relatively rich families, for example a family of very deep decision trees, or other similarly complex families. In DeepBoost, the decisions in each iteration of which classifier to add to the ensemble and which weight to assign to that classifier, depend on the (data-dependent) complexity of the sub-family to which the classifier belongs. Cortes, Mohri, and Syed [2014] further showed that empirically DeepBoost achieves a better performance than AdaBoost, Logistic Regression, and their L_1-regularized variants. Both AdaBoost and L_1-regularized AdaBoost can be viewed as special instances of DeepBoost.

Dietterich [2000] provided extensive empirical evidence for the fact that uniform noise can severely damage the accuracy of AdaBoost. This has been reported by a number of other authors since then. Long and Servedio [2010] further recently showed the failure of boosting algorithms based on convex potentials to tolerate random noise, even with L_1-regularization or early stopping.

There are several excellent surveys and tutorials related to boosting [Schapire, 2003, Meir and Rätsch, 2002, Meir and Rätsch, 2003], including the recent book of Schapire and Freund [2012] fully dedicated to this topic, with an extensive list of references and a detailed presentation.

7.7 Exercises

7.1 VC-dimension of the hypothesis set of AdaBoost. Prove the upper bound on the VC-dimension of the hypothesis set $\mathcal{F}_T$ of AdaBoost after T rounds of boosting, as stated in equation (7.9).

7.2 Alternative objective functions. This problem studies boosting-type algorithms defined with objective functions different from that of AdaBoost. We assume that the training data are given as m labeled examples $(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \{-1, +1\}$. We further assume that Φ is a strictly increasing convex and differentiable function over $\mathbb{R}$ such that: $\forall x \geq 0, \Phi(x) \geq 1$ and $\forall x < 0, \Phi(x) > 0$.

(a) Consider the loss function $L(\alpha) = \sum_{i=1}^{m} \Phi(-y_i f(x_i))$ where f is a linear combination of base classifiers, i.e., $f = \sum_{t=1}^{T} \alpha_t h_t$ (as in AdaBoost). Derive a new boosting algorithm using the objective function L. In particular, characterize the best base classifier h_u to select at each round of boosting if we use coordinate descent.

(b) Consider the following functions: (1) zero-one loss $\Phi_1(-u) = 1_{u \leq 0}$; (2) least squared loss $\Phi_2(-u) = (1-u)^2$; (3) SVM loss $\Phi_3(-u) = \max\{0, 1-u\}$; and (4) logistic loss $\Phi_4(-u) = \log(1 + e^{-u})$. Which functions satisfy the assumptions on Φ stated earlier in this problem?

(c) For each loss function satisfying these assumptions, derive the corresponding boosting algorithm. How do the algorithm(s) differ from AdaBoost?

7.3 Update guarantee. Assume that the main weak learner assumption of AdaBoost holds. Let h_t be the base learner selected at round t. Show that the base learner h_{t+1} selected at round $t+1$ must be different from h_t.

7.4 Weighted instances. Let the training sample be $S = ((x_1, y_1), \ldots, (x_m, y_m))$. Suppose we wish to penalize differently errors made on x_i versus x_j. To do that, we associate some non-negative importance weight w_i to each point x_i and define the objective function $F(\boldsymbol{\alpha}) = \sum_{i=1}^m w_i e^{-y_i f(x_i)}$, where $f = \sum_{t=1}^T \alpha_t h_t$. Show that this function is convex and differentiable and use it to derive a boosting-type algorithm.

7.5 Define the unnormalized correlation of two vectors $\mathbf{x}$ and $\mathbf{x}'$ as the inner product between these vectors. Prove that the distribution vector $(\mathcal{D}_{t+1}(1), \ldots, \mathcal{D}_{t+1}(m))$ defined by AdaBoost and the vector of components $y_i h_t(x_i)$ are uncorrelated.

7.6 Fix $\epsilon \in (0, 1/2)$. Let the training sample be defined by m points in the plane with $\frac{m}{4}$ negative points all at coordinate $(1, 1)$, another $\frac{m}{4}$ negative points all at coordinate $(-1, -1)$, $\frac{m(1-\epsilon)}{4}$ positive points all at coordinate $(1, -1)$, and $\frac{m(1+\epsilon)}{4}$ positive points all at coordinate $(-1, +1)$. Describe the behavior of AdaBoost when run on this sample using boosting stumps. What solution does the algorithm return after T rounds?

7.7 Noise-tolerant AdaBoost. AdaBoost may be significantly overfitting in the presence of noise, in part due to the high penalization of misclassified examples. To reduce this effect, one could use instead the following objective function:

$$F = \sum_{i=1}^m G(-y_i f(x_i)), \tag{7.32}$$

where G is the function defined on $\mathbb{R}$ by

$$G(x) = \begin{cases} e^x & \text{if } x \le 0 \\ x + 1 & \text{otherwise.} \end{cases} \tag{7.33}$$

(a) Show that the function G is convex and differentiable.

(b) Use F and greedy coordinate descent to derive an algorithm similar to AdaBoost.

(c) Compare the reduction of the empirical error rate of this algorithm with that of AdaBoost.

7.8 Simplified AdaBoost. Suppose we simplify AdaBoost by setting the parameter α_t to a fixed value $\alpha_t = \alpha > 0$, independent of the boosting round t.

(a) Let γ be such that $(\frac{1}{2} - \epsilon_t) \ge \gamma > 0$. Find the best value of α as a function of γ by analyzing the empirical error.

(b) For this value of α, does the algorithm assign the same probability mass to correctly classified and misclassified examples at each round? If not, which set is assigned a higher probability mass?

```
AdaBoost($\mathbf{M}, t_{\max}$)
1  $\lambda_{1,j} \leftarrow 0$ for $i = 1, \ldots, m$
2  for $t \leftarrow 1$ to $t_{\max}$ do
3      $d_{t,i} \leftarrow \frac{\exp(-(\mathbf{M}\boldsymbol{\lambda}_t)_i)}{\sum_{k=1}^m \exp(-(\mathbf{M}\boldsymbol{\lambda}_t)_k)}$ for $i = 1, \ldots, m$
4      $j_t \leftarrow \operatorname{argmax}_j (\mathbf{d}_t^\top \mathbf{M})_j$
5      $r_t \leftarrow (\mathbf{d}_t^\top \mathbf{M})_{j_t}$
6      $\alpha_t \leftarrow \frac{1}{2} \log\left(\frac{1+r_t}{1-r_t}\right)$
7      $\boldsymbol{\lambda}_{t+1} \leftarrow \boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t}$, where $\mathbf{e}_{j_t}$ is 1 in position $j_t$ and 0 elsewhere.
8  return $\frac{\boldsymbol{\lambda}_{t_{\max}}}{\|\boldsymbol{\lambda}_{t_{\max}}\|_1}$
```

Figure 7.7
AdaBoost defined with respect to a matrix $\mathbf{M}$, which encodes the accuracy of each weak classifier on each training point.

(c) Using the previous value of α, give a bound on the empirical error of the algorithm that depends only on γ and the number of rounds of boosting T.

(d) Using the previous bound, show that for $T > \frac{\log m}{2\gamma^2}$, the resulting hypothesis is consistent with the sample of size m.

(e) Let s be the VC-dimension of the base learners used. Give a bound on the generalization error of the consistent hypothesis obtained after $T = \left\lfloor \frac{\log m}{2\gamma^2} \right\rfloor +$ 1 rounds of boosting. (*Hint*: Use the fact that the VC-dimension of the family of functions $\{\text{sgn}(\sum_{t=1}^T \alpha_t h_t) : \alpha_t \in \mathbb{R}\}$ is bounded by $2(s+1)T \log_2(eT)$). Suppose now that γ varies with m. Based on the bound derived, what can be said if $\gamma(m) = O(\sqrt{\frac{\log m}{m}})$?)

7.9 AdaBoost example.

In this exercise we consider a concrete example that consists of eight training points and eight weak classifiers.

(a) Define an $m \times n$ matrix $\mathbf{M}$ where $\mathbf{M}_{ij} = y_i h_j(\mathbf{x}_i)$, i.e., $\mathbf{M}_{ij} = +1$ if training example i is classified correctly by weak classifier h_j, and -1 otherwise. Let $\mathbf{d}_t, \boldsymbol{\lambda}_t \in \mathbb{R}^n$, $\|\mathbf{d}_t\|_1 = 1$ and $d_{t,i}$ (respectively $\lambda_{t,i}$) equal the i^{th} component of $\mathbf{d}_t$ (respectively $\boldsymbol{\lambda}_t$). Now, consider AdaBoost as described in figure 7.7

and define $\mathbf{M}$ as below with eight training points and eight weak classifiers.

$$\mathbf{M} = \begin{pmatrix} -1 & 1 & 1 & 1 & 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 \end{pmatrix}$$

Assume that we start with the following initial distribution over the datapoints:

$$\mathbf{d}_1 = \left(\frac{3-\sqrt{5}}{8}, \frac{3-\sqrt{5}}{8}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{\sqrt{5}-1}{8}, \frac{\sqrt{5}-1}{8}, 0\right)^\top$$

Compute the first few steps of the AdaBoost algorithm using $\mathbf{M}$, $\mathbf{d}_1$, and $t_{max} = 7$. What weak classifier is picked at each round of boosting? Do you notice any pattern?

(b) What is the L_1 norm margin produced by AdaBoost for this example?

(c) Instead of using AdaBoost, imagine we combined our classifiers using the following coefficients: $[2, 3, 4, 1, 2, 2, 1, 1] \times \frac{1}{16}$. What is the margin in this case? Does AdaBoost maximize the margin?

7.10 Boosting in the presence of unknown labels. Consider the following variant of the classification problem where, in addition to the positive and negative labels $+1$ and -1, points may be labeled with 0. This can correspond to cases where the true label of a point is unknown, a situation that often arises in practice, or more generally to the fact that the learning algorithm incurs no loss for predicting -1 or $+1$ for such a point. Let $\mathcal{X}$ be the input space and let $\mathcal{Y} = \{-1, 0, +1\}$. As in standard binary classification, the loss of $f \colon \mathcal{X} \to \mathbb{R}$ on a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is defined by $1_{yf(x)<0}$.

Consider a sample $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ and a hypothesis set H of base functions taking values in $\{-1, 0, +1\}$. For a base hypothesis $h_t \in \mathcal{H}$ and a distribution $\mathcal{D}_t$ over indices $i \in [m]$, define ϵ_t^s for $s \in \{-1, 0, +1\}$ by $\epsilon_t^s = \mathbb{E}_{i \sim \mathcal{D}_t}[1_{y_i h_t(x_i)=s}]$.

(a) Derive a boosting-style algorithm for this setting in terms of ϵ_t^ss, using the same objective function as that of AdaBoost. You should carefully justify the definition of the algorithm.

(b) What is the weak-learning assumption in this setting?

(c) Write the full pseudocode of the algorithm.

(d) Give an upper bound on the training error of the algorithm as a function of the number of rounds of boosting and ϵ_t^ss.

7.11 HingeBoost. As discussed in the chapter, AdaBoost can be viewed as coordinate descent applied to an exponential objective function. Here, we consider an alternative ensemble method algorithm, HingeBoost, that consists of applying coordinate descent to an objective function based on the hinge loss. Consider the function F defined for all $\boldsymbol{\alpha} \in \mathbb{R}^N$ by

$$F(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \max\left(0, 1 - y_i \sum_{j=1}^{N} \alpha_j h_j(x_i)\right), \tag{7.34}$$

where the h_js are base classifiers belonging to a hypothesis set H of functions taking values -1 or $+1$.

(a) Show that F is convex and admits a right- and left-derivative along any direction.

(b) For any $j \in [N]$, let $\boldsymbol{e}_j$ denote the direction corresponding to the base hypothesis h_j. Let $\boldsymbol{\alpha}_t$ denote the vector of coefficients $\alpha_{t,j}$, $j \in [N]$ obtained after $t \geq 0$ iterations of coordinate descent and $f_t = \sum_{j=1}^{N} \alpha_{t,j} h_j$ the predictor obtained after t iterations.

Give the expression of the right-derivative $F'_+(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j)$ and the left-derivative $F'_-(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j)$ after $t-1$ iterations in terms of f_{t-1}.

(c) For any $j \in [N]$, define the maximum directional derivative $\delta F(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j)$ at $\boldsymbol{\alpha}_{t-1}$ as follows:

$$\delta F(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) =
\begin{cases}
0 & \text{if } F'_-(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) \leq 0 \leq F'_+(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) \\
F'_+(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) & \text{if } F'_-(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) \leq F'_+(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) \leq 0 \\
F'_-(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) & \text{if } 0 \leq F'_-(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j) \leq F'_+(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j).
\end{cases}$$

The direction $\boldsymbol{e}_j$ considered by the coordinate descent considered here is the one maximizing $|\delta F(\boldsymbol{\alpha}_{t-1}, \boldsymbol{e}_j)|$. Once the best direction j is selected, the step η can be determined by minimizing $F(\boldsymbol{\alpha}_{t-1} + \eta \mathbf{e}_j)$ using a grid search. Give the pseudocode of HingeBoost.

7.12 Empirical margin loss boosting. As discussed in the chapter, AdaBoost can be viewed as coordinate descent applied to a convex upper bound on the empirical error. Here, we consider an algorithm seeking to minimize the empirical margin loss. For any $0 \leq \rho < 1$ let $\widehat{R}_{S,\rho}(f) = \frac{1}{m}\sum_{i=1}^m 1_{y_i f(x_i) \leq \rho}$ denote the empirical margin loss of a function f of the form $f = \frac{\sum_{t=1}^T \alpha_t h_t}{\sum_{t=1}^T \alpha_t}$ for a labeled sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$.

(a) Show that $\widehat{R}_{S,\rho}(f)$ can be upper bounded as follows:

$$\widehat{R}_{S,\rho}(f) \leq \frac{1}{m}\sum_{i=1}^m \exp\left(-y_i \sum_{t=1}^T \alpha_t h_t(x_i) + \rho \sum_{t=1}^T \alpha_t\right).$$

(b) For any $\rho > 0$, let G_ρ be the objective function defined for all $\boldsymbol{\alpha} \geq 0$ by

$$G_\rho(\boldsymbol{\alpha}) = \frac{1}{m}\sum_{i=1}^m \exp\left(-y_i \sum_{j=1}^N \alpha_j h_j(x_i) + \rho \sum_{j=1}^N \alpha_j\right),$$

with $h_j \in H$ for all $j \in [N]$, with the notation used in class in the boosting lecture. Show that G_ρ is convex and differentiable.

(c) Derive a boosting-style algorithm $\mathcal{A}_\rho$ by applying (maximum) coordinate descent to G_ρ. You should justify in detail the derivation of the algorithm, in particular the choice of the base classifier selected at each round and that of the step. Compare both to their counterparts in AdaBoost.

(d) What is the equivalent of the weak learning assumption for $\mathcal{A}_\rho$ (*Hint*: use non-negativity of the step value)?

(e) Give the full pseudocode of the algorithm $\mathcal{A}_\rho$. What can you say about the $\mathcal{A}_0$ algorithm?

(f) Provide a bound on $\widehat{R}_{S,\rho}(f)$.

i. Prove the upper bound $\widehat{R}_{S,\rho}(f) \leq \exp\left(\sum_{t=1}^T \alpha_t \rho\right) \prod_{t=1}^T Z_t$, where the normalization factors Z_t are defined as in the case of AdaBoost (with α_t the step chosen by $\mathcal{A}_\rho$ at round t).

ii. Give the expression of Z_t as a function of ρ and ϵ_t, where ϵ_t is the weighted error of the hypothesis found by $\mathcal{A}_\rho$ at round t (defined in the same way as for AdaBoost in class). Use that to prove the following upper bound

$$\widehat{R}_{S,\rho}(f) \leq \left(u^{\frac{1+\rho}{2}} + u^{-\frac{1-\rho}{2}}\right)^T \prod_{t=1}^{T} \sqrt{\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}},$$

where $u = \frac{1-\rho}{1+\rho}$.

iii. Assume that for all $t \in [T]$, $\frac{1-\rho}{2} - \epsilon_t > \gamma > 0$. Use the result of the previous question to show that

$$\widehat{R}_{S,\rho}(f) \leq \exp\left(-\frac{2\gamma^2 T}{1-\rho^2}\right).$$

(*Hint*: you can use without proof the following identity:

$$\left(u^{\frac{1+\rho}{2}} + u^{-\frac{1-\rho}{2}}\right)\sqrt{\epsilon_t^{1-\rho}(1-\epsilon_t)^{1+\rho}} \leq 1 - 2\frac{\left(\frac{1-\rho}{2} - \epsilon_t\right)^2}{1-\rho^2},$$

valid for $\frac{1-\rho}{2} - \epsilon_t > 0$.) Show that for $T \geq \frac{(\log m)(1-\rho^2)}{2\gamma^2}$, all points of the training data have margin at least ρ.

8 On-Line Learning

This chapter presents an introduction to on-line learning, an important area with a rich literature and multiple connections with game theory and optimization that is increasingly influencing the theoretical and algorithmic advances in machine learning. In addition to the intriguing novel learning theory questions that they raise, on-line learning algorithms are particularly attractive in modern applications since they provide an efficient solution for large-scale problems.

These algorithms process one sample at a time with an update per iteration that is often computationally cheap and easy to implement. As a result, they are typically significantly more efficient both in time and space and more practical than batch algorithms, when processing modern data sets of several million or billion points. They are also typically easy to implement. Moreover, on-line algorithms do not require any distributional assumption; their analysis assumes an adversarial scenario. This makes them applicable in a variety of scenarios where the sample points are not drawn i.i.d. or according to a fixed distribution.

We first introduce the general scenario of on-line learning, then present and analyze several key algorithms for on-line learning with expert advice, including the deterministic and randomized weighted majority algorithms for the zero-one loss and an extension of these algorithms for convex losses. We also describe and analyze two standard on-line algorithms for linear classification, the Perceptron and Winnow algorithms, as well as some extensions. While on-line learning algorithms are designed for an adversarial scenario, they can be used, under some assumptions, to derive accurate predictors for a distributional scenario. We derive learning guarantees for this on-line to batch conversion. Finally, we briefly point out the connection of on-line learning with game theory by describing its use to derive a simple proof of von Neumann's minimax theorem.

8.1 Introduction

The learning framework for on-line algorithms is in stark contrast to the PAC learning or stochastic models discussed up to this point. First, instead of learning from a training set and then testing on a test set, the on-line learning scenario mixes the training and test phases. Second, PAC learning follows the key assumption that the distribution over data points is fixed over time, both for training and test points, and that points are sampled in an i.i.d. fashion. Under this assumption, the natural goal is to learn a hypothesis with a small expected loss or generalization error. In contrast, with on-line learning, *no distributional assumption* is made, and thus there is no notion of generalization. Instead, the performance of on-line learning algorithms is measured using a *mistake model* and the notion of *regret*. To derive guarantees in this model, theoretical analyses are based on a worst-case or adversarial assumption.

The general on-line setting involves T rounds. At the tth round, the algorithm receives an instance $x_t \in \mathcal{X}$ and makes a prediction $\widehat{y}_t \in \mathcal{Y}$. It then receives the true label $y_t \in \mathcal{Y}$ and incurs a loss $L(\widehat{y}_t, y_t)$, where $L\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a loss function. More generally, the prediction domain for the algorithm may be $\mathcal{Y}' \neq \mathcal{Y}$ and the loss function defined over $\mathcal{Y}' \times \mathcal{Y}$. For classification problems, we often have $\mathcal{Y} = \{0, 1\}$ and $L(y, y') = |y' - y|$, while for regression $\mathcal{Y} \subseteq \mathbb{R}$ and typically $L(y, y') = (y' - y)^2$. The objective in the on-line setting is to minimize the cumulative loss: $\sum_{t=1}^{T} L(\widehat{y}_t, y_t)$ over T rounds.

8.2 Prediction with expert advice

We first discuss the setting of online learning with expert advice, and the associated notion of *regret*. In this setting, at the tth round, in addition to receiving $x_t \in \mathcal{X}$, the algorithm also receives *advice* $y_{t,i} \in \mathcal{Y}$, $i \in [N]$, from N experts. Following the general framework of on-line algorithms, it then makes a prediction, receives the true label, and incurs a loss. After T rounds, the algorithm has incurred a cumulative loss. The objective in this setting is to minimize the *regret* R_T, also called *external regret*, which compares the cumulative loss of the algorithm to that of the best expert in hindsight after T rounds:

$$R_T = \sum_{t=1}^{T} L(\widehat{y}_t, y_t) - \min_{i=1}^{N} \sum_{t=1}^{T} L(\widehat{y}_{t,i}, y_t). \tag{8.1}$$

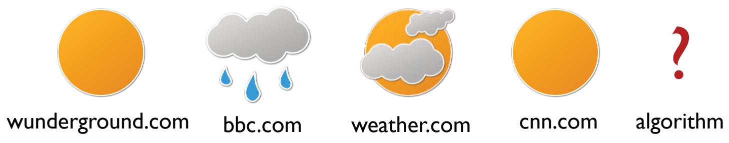


Figure 8.1
Weather forecast: an example of a prediction problem based on expert advice.

This problem arises in a variety of different domains and applications. Figure 8.1 illustrates the problem of predicting the weather using several forecasting sources as experts.

8.2.1 Mistake bounds and Halving algorithm

Here, we assume that the loss function is the standard zero-one loss used in classification. To analyze the expert advice setting, we first consider the realizable case, that is the setting where at least one of the experts makes no errors. As such, we discuss the *mistake bound model*, which asks the simple question "How many mistakes before we learn a particular concept?" Since we are in the realizable case, after some number of rounds T, we will learn the concept and no longer make errors in subsequent rounds. For any fixed concept c, we define the maximum number of mistakes a learning algorithm $\mathcal{A}$ makes as

$$M_{\mathcal{A}}(c) = \max_{x_1,\ldots,x_T} |\text{mistakes}(\mathcal{A}, c)|. \tag{8.2}$$

Further, for any concept in a concept class $\mathcal{C}$, the maximum number of mistakes a learning algorithm makes is

$$M_{\mathcal{A}}(\mathcal{C}) = \max_{c\in\mathcal{C}} M_{\mathcal{A}}(c). \tag{8.3}$$

Our goal in this setting is to derive *mistake bounds*, that is, a bound M on $M_{\mathcal{A}}(\mathcal{C})$. We will first do this for the Halving algorithm, an elegant and simple algorithm for which we can guarantee surprisingly favorable mistake bounds. At each round, the Halving algorithm makes its prediction by taking the majority vote over all *active* experts. After any incorrect prediction, it deactivates all experts that gave faulty advice. Initially, all experts are active, and by the time the algorithm has converged to the correct concept, the active set contains only those experts that are consistent with the target concept. The pseudocode for this algorithm is shown in figure 8.2. We also present straightforward mistake bounds in theorems 8.1 and 8.2, where the former deals with finite hypothesis sets and the latter relates mistake bounds to VC-dimension. Note that the hypothesis complexity term in theorem 8.1 is identical to the corresponding complexity term in the PAC model bound of theorem 2.5.

```
Halving($\mathcal{H}$)
1  $\mathcal{H}_1 \leftarrow \mathcal{H}$
2  for $t \leftarrow 1$ to $T$ do
3      Receive($x_t$)
4      $\widehat{y}_t \leftarrow$ MajorityVote($\mathcal{H}_t, x_t$)
5      Receive($y_t$)
6      if ($\widehat{y}_t \neq y_t$) then
7          $\mathcal{H}_{t+1} \leftarrow \{c \in \mathcal{H}_t \colon c(x_t) = y_t\}$
8      else $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t$
9  return $\mathcal{H}_{T+1}$
```

Figure 8.2
Halving algorithm.

Theorem 8.1 *Let $\mathcal{H}$ be a finite hypothesis set. Then*

$$M_{\text{Halving}}(\mathcal{H}) \leq \log_2 |\mathcal{H}|. \tag{8.4}$$

Proof: Since at each round the algorithm makes predictions using majority vote from the active set, at each mistake, the active set is reduced by at least half. Hence, after $\log_2 |\mathcal{H}|$ mistakes, there can only remain one active hypothesis, and since we are in the realizable case, this hypothesis must coincide with the target concept. □

Theorem 8.2 *Let $opt(\mathcal{H})$ be the optimal mistake bound for $\mathcal{H}$. Then,*

$$\text{VCdim}(\mathcal{H}) \leq opt(\mathcal{H}) \leq M_{\text{Halving}}(\mathcal{H}) \leq \log_2 |\mathcal{H}|. \tag{8.5}$$

Proof: The second inequality is true by definition and the third inequality holds based on theorem 8.1. To prove the first inequality, we let $d = \text{VCdim}(\mathcal{H})$. Then there exists a shattered set of d points, for which we can form a complete binary tree of the mistakes with height d, and we can choose labels at each round of learning to ensure that d mistakes are made. Note that this adversarial argument is valid since the on-line setting makes no statistical assumptions about the data. □

```
WEIGHTED-MAJORITY($N$)
 1  for $i \leftarrow 1$ to $N$ do
 2      $w_{1,i} \leftarrow 1$
 3  for $t \leftarrow 1$ to $T$ do
 4      RECEIVE($x_t$)
 5      if $\sum_{i\colon y_{t,i}=1} w_{t,i} \geq \sum_{i\colon y_{t,i}=0} w_{t,i}$ then
 6          $\widehat{y}_t \leftarrow 1$
 7      else $\widehat{y}_t \leftarrow 0$
 8      RECEIVE($y_t$)
 9      if $(\widehat{y}_t \neq y_t)$ then
10          for $i \leftarrow 1$ to $N$ do
11              if $(y_{t,i} \neq y_t)$ then
12                  $w_{t+1,i} \leftarrow \beta w_{t,i}$
13              else $w_{t+1,i} \leftarrow w_{t,i}$
14  return $\mathbf{w}_{T+1}$
```

Figure 8.3
Weighted majority algorithm, $y_t, y_{t,i} \in \{0, 1\}$.

8.2.2 Weighted majority algorithm

In the previous section, we focused on the realizable setting in which the Halving algorithm simply discarded experts after a single mistake. We now move to the non-realizable setting and use a more general and less extreme algorithm, the Weighted Majority (WM) algorithm, that *weights* the importance of experts as a function of their mistake rate. The WM algorithm begins with uniform weights over all N experts. At each round, it generates predictions using a weighted majority vote. After receiving the true label, the algorithm then reduces the weight of each incorrect expert by a factor of $\beta \in [0, 1)$. Note that this algorithm reduces to the Halving algorithm when $\beta = 0$. The pseudocode for the WM algorithm is shown in figure 8.3.

Since we are not in the realizable setting, the mistake bounds of theorem 8.1 cannot apply. However, the following theorem presents a bound on the number of mistakes m_T made by the WM algorithm after $T \geq 1$ rounds of on-line learning as a function of the number of mistakes made by the best expert, that is the expert who achieves the smallest number of mistakes for the sequence $y_1, \ldots, y_T$. Let us emphasize that this is the best expert *in hindsight*.

Theorem 8.3 *Fix $\beta \in (0, 1)$. Let m_T be the number of mistakes made by algorithm WM after $T \geq 1$ rounds, and m_T^* be the number of mistakes made by the best of the N experts. Then, the following inequality holds:*

$$m_T \leq \frac{\log N + m_T^* \log \frac{1}{\beta}}{\log \frac{2}{1+\beta}}. \tag{8.6}$$

Proof: To prove this theorem, we first introduce a potential function. We then derive upper and lower bounds for this function, and combine them to obtain our result. This potential function method is a general proof technique that we will use throughout this chapter.

For any $t \geq 1$, we define our potential function as $W_t = \sum_{i=1}^{N} w_{t,i}$. Since predictions are generated using weighted majority vote, if the algorithm makes an error at round t, this implies that

$$W_{t+1} \leq \big[1/2 + (1/2)\beta\big] W_t = \left[\frac{1+\beta}{2}\right] W_t. \tag{8.7}$$

Since $W_1 = N$ and m_T mistakes are made after T rounds, we thus have the following upper bound:

$$W_T \leq \left[\frac{1+\beta}{2}\right]^{m_T} N. \tag{8.8}$$

Next, since the weights are all non-negative, it is clear that for any expert i, $W_T \geq w_{T,i} = \beta^{m_{T,i}}$, where $m_{T,i}$ is the number of mistakes made by the ith expert after T rounds. Applying this lower bound to the best expert and combining it with the upper bound in (8.8) gives us:

$$\begin{aligned} \beta^{m_T^*} &\leq W_T \leq \left[\frac{1+\beta}{2}\right]^{m_T} N \\ \Rightarrow m_T^* \log \beta &\leq \log N + m_T \log \left[\frac{1+\beta}{2}\right] \\ \Rightarrow m_T \log \left[\frac{2}{1+\beta}\right] &\leq \log N + m_T^* \log \frac{1}{\beta}, \end{aligned}$$

which concludes the proof. □

Thus, the theorem guarantees a bound of the following form for algorithm WM:

$$m_T \leq O(\log N) + \text{constant} \times |\text{mistakes of best expert}|.$$

Since the first term varies only logarithmically as a function of N, the theorem guarantees that the number of mistakes is roughly a constant times that of the best expert in hindsight. This is a remarkable result, especially because it requires no assumption about the sequence of points and labels generated. In particular, the sequence could be chosen adversarially. In the realizable case where $m_T^* = 0$, the bound reduces to $m_T \leq O(\log N)$ as for the Halving algorithm.

8.2.3 Randomized weighted majority algorithm

In spite of the guarantees just discussed, the WM algorithm admits a drawback that affects all deterministic algorithms in the case of the zero-one loss: no deterministic algorithm can achieve a regret $R_T = o(T)$ over all sequences. Clearly, for any deterministic algorithm $\mathcal{A}$ and any $t \in [T]$, we can adversarially select y_t to be 1 if the algorithm predicts 0, and choose it to be 0 otherwise. Thus, $\mathcal{A}$ errs at every point of such a sequence and its cumulative mistake is $m_T = T$. Assume for example that $N = 2$ and that one expert always predicts 0, the other one always 1. The error of the best expert over that sequence (and in fact any sequence of that length) is then at most $m_T^* \leq T/2$. Thus, for that sequence, we have

$$R_T = m_T - m_T^* \geq T/2,$$

which shows that $R_T = o(T)$ cannot be achieved in general. Note that this does not contradict the bound proven in the previous section, since for any $\beta \in (0, 1)$, $\frac{\log \frac{1}{\beta}}{\log \frac{2}{1+\beta}} \geq 2$. As we shall see in the next section, this negative result does not hold for any loss that is convex with respect to one of its arguments. But for the zero-one loss, this leads us to consider randomized algorithms instead.

In the randomized scenario of on-line learning, we assume that a set $\mathcal{A} = \{1, \ldots, N\}$ of N actions is available. At each round $t \in [T]$, an on-line algorithm $\mathcal{A}$ selects a distribution $\mathbf{p}_t$ over the set of actions, receives a loss vector $\mathbf{l}_t$, whose ith component $l_{t,i} \in [0, 1]$ is the loss associated with action i, and incurs the expected loss $L_t = \sum_{i=1}^N p_{t,i}\, l_{t,i}$. The total loss incurred by the algorithm over T rounds is $\mathcal{L}_T = \sum_{t=1}^T L_t$. The total loss associated to action i is $\mathcal{L}_{T,i} = \sum_{t=1}^T l_{t,i}$. The minimal loss of a single action is denoted by $\mathcal{L}_T^{\min} = \min_{i \in \mathcal{A}} \mathcal{L}_{T,i}$. The regret R_T of the algorithm after T rounds is then typically defined by the difference of the loss of the algorithm and that of the best single action:[11]

$$R_T = \mathcal{L}_T - \mathcal{L}_T^{\min}.$$

Here, we consider specifically the case of zero-one losses and assume that $l_{t,i} \in \{0, 1\}$ for all $t \in [T]$ and $i \in \mathcal{A}$.

[11] Alternative definitions of the regret with comparison classes different from the set of single actions can be considered.

```
RANDOMIZED-WEIGHTED-MAJORITY ($N$)
 1  for $i \leftarrow 1$ to $N$ do
 2      $w_{1,i} \leftarrow 1$
 3      $p_{1,i} \leftarrow 1/N$
 4  for $t \leftarrow 1$ to $T$ do
 5      RECEIVE($\mathbf{l}_t$)
 6      for $i \leftarrow 1$ to $N$ do
 7          if ($l_{t,i} = 1$) then
 8              $w_{t+1,i} \leftarrow \beta w_{t,i}$
 9          else $w_{t+1,i} \leftarrow w_{t,i}$
10      $W_{t+1} \leftarrow \sum_{i=1}^{N} w_{t+1,i}$
11      for $i \leftarrow 1$ to $N$ do
12          $p_{t+1,i} \leftarrow w_{t+1,i}/W_{t+1}$
13  return $\mathbf{w}_{T+1}$
```

Figure 8.4
Randomized weighted majority algorithm.

The WM algorithm admits a straightforward randomized version, the randomized weighted majority (RWM) algorithm. The pseudocode of this algorithm is given in figure 8.4. The algorithm updates the weight $w_{t,i}$ of expert i as in the case of the WM algorithm by multiplying it by β. The following theorem gives a strong guarantee on the regret R_T of the RWM algorithm, showing that it is in $O(\sqrt{T \log N})$.

Theorem 8.4 *Fix $\beta \in [1/2, 1)$. Then, for any $T \geq 1$, the loss of algorithm RWM on any sequence can be bounded as follows:*

$$\mathcal{L}_T \leq \frac{\log N}{1-\beta} + (2-\beta)\mathcal{L}_T^{\min}. \tag{8.9}$$

In particular, for $\beta = \max\{1/2, 1 - \sqrt{(\log N)/T}\}$, the loss can be bounded as:

$$\mathcal{L}_T \leq \mathcal{L}_T^{\min} + 2\sqrt{T \log N}. \tag{8.10}$$

Proof: As in the proof of theorem 8.3, we derive upper and lower bounds for the potential function $W_t = \sum_{i=1}^{N} w_{t,i}$, $t \in [T]$, and combine these bounds to obtain the result. By definition of the algorithm, for any $t \in [T]$, W_{t+1} can be expressed as follows in terms of W_t:

$$\begin{aligned} W_{t+1} = \sum_{i\colon l_{t,i}=0} w_{t,i} + \beta \sum_{i\colon l_{t.i}=1} w_{t,i} &= W_t + (\beta - 1) \sum_{i\colon l_{t,i}=1} w_{t,i} \\ &= W_t + (\beta - 1) W_t \sum_{i\colon l_{t,i}=1} p_{t,i} \\ &= W_t + (\beta - 1) W_t L_t \\ &= W_t (1 - (1-\beta) L_t). \end{aligned}$$

Thus, since $W_1 = N$, it follows that $W_{T+1} = N \prod_{t=1}^{T} (1 - (1-\beta)L_t)$. On the other hand, the following lower bound clearly holds: $W_{T+1} \geq \max_{i \in [N]} w_{T+1,i} = \beta^{\mathcal{L}_T^{\min}}$. This leads to the following inequality and series of derivations after taking the log and using the inequalities $\log(1-x) \leq -x$ valid for all $x < 1$, and $-\log(1-x) \leq x + x^2$ valid for all $x \in [0, 1/2]$:

$$\begin{aligned} \beta^{\mathcal{L}_T^{\min}} \leq N \prod_{t=1}^{T} (1 - (1-\beta)L_t) &\implies \mathcal{L}_T^{\min} \log \beta \leq \log N + \sum_{t=1}^{T} \log(1 - (1-\beta)L_t) \\ &\implies \mathcal{L}_T^{\min} \log \beta \leq \log N - (1-\beta) \sum_{t=1}^{T} L_t \\ &\implies \mathcal{L}_T^{\min} \log \beta \leq \log N - (1-\beta) \mathcal{L}_T \\ &\implies \mathcal{L}_T \leq \frac{\log N}{1-\beta} - \frac{\log \beta}{1-\beta} \mathcal{L}_T^{\min} \\ &\implies \mathcal{L}_T \leq \frac{\log N}{1-\beta} - \frac{\log(1 - (1-\beta))}{1-\beta} \mathcal{L}_T^{\min} \\ &\implies \mathcal{L}_T \leq \frac{\log N}{1-\beta} + (2-\beta) \mathcal{L}_T^{\min}. \end{aligned}$$

This shows the first statement. Since $\mathcal{L}_T^{\min} \leq T$, this also implies

$$\mathcal{L}_T \leq \frac{\log N}{1-\beta} + (1-\beta)T + \mathcal{L}_T^{\min}. \tag{8.11}$$

Differentiating the upper bound with respect to β and setting it to zero gives $\frac{\log N}{(1-\beta)^2} - T = 0$, that is $\beta = 1 - \sqrt{(\log N)/T} < 1$. Thus, if $1 - \sqrt{(\log N)/T} \geq 1/2$, $\beta_0 = 1 - \sqrt{(\log N)/T}$ is the minimizing value of β, otherwise the boundary value $\beta_0 = 1/2$ is the optimal value. The second statement follows by replacing β with β_0 in (8.11). □

The bound (8.10) assumes that the algorithm additionally receives as a parameter the number of rounds T. As we shall see in the next section, however, there exists a general *doubling trick* that can be used to relax this requirement at the price of a small constant factor increase. Inequality 8.10 can be written directly in terms of the regret R_T of the RWM algorithm:

$$R_T \leq 2\sqrt{T \log N}. \tag{8.12}$$

Thus, for N constant, the regret verifies $R_T = O(\sqrt{T})$ and the *average regret* or *regret per round* R_T/T decreases as $O(1/\sqrt{T})$. These results are optimal, as shown by the following theorem.

Theorem 8.5 *Let $N = 2$. There exists a stochastic sequence of losses for which the regret of any on-line learning algorithm verifies $\mathbb{E}[R_T] \geq \sqrt{T/8}$.*

Proof: For any $t \in [T]$, let the vector of losses $\mathbf{l}_t$ take the values $\mathbf{l}_{01} = (0,1)^\top$ and $\mathbf{l}_{10} = (1,0)^\top$ with equal probability. Then, the expected loss of any randomized algorithm $\mathcal{A}$ is

$$\mathbb{E}[\mathcal{L}_T] = \mathbb{E}\Big[\sum_{t=1}^{T} \mathbf{p}_t \cdot \mathbf{l}_t\Big] = \sum_{t=1}^{T} \mathbf{p}_t \cdot \mathbb{E}[\mathbf{l}_t] = \sum_{t=1}^{T} \frac{1}{2} p_{t,1} + \frac{1}{2}(1 - p_{t,1}) = T/2,$$

where we denoted by $\mathbf{p}_t$ the distribution selected by $\mathcal{A}$ at round t. By definition, $\mathcal{L}_T^{\min}$ can be written as follows:

$$\mathcal{L}_T^{\min} = \min\{\mathcal{L}_{T,1}, \mathcal{L}_{T,2}\} = \frac{1}{2}(\mathcal{L}_{T,1} + \mathcal{L}_{T,2} - |\mathcal{L}_{T,1} - \mathcal{L}_{T,2}|) = T/2 - |\mathcal{L}_{T,1} - T/2|,$$

using the fact that $\mathcal{L}_{T,1} + \mathcal{L}_{T,2} = T$. Thus, the expected regret of $\mathcal{A}$ is

$$\mathbb{E}[R_T] = \mathbb{E}[\mathcal{L}_T] - \mathbb{E}[\mathcal{L}_T^{\min}] = \mathbb{E}[|\mathcal{L}_{T,1} - T/2|].$$

Let σ_t, $t \in [T]$, denote Rademacher variables taking values in $\{-1, +1\}$, then $\mathcal{L}_{T,1}$ can be rewritten as $\mathcal{L}_{T,1} = \sum_{t=1}^{T} \frac{1+\sigma_t}{2} = T/2 + \frac{1}{2}\sum_{t=1}^{T} \sigma_t$. Thus, introducing scalars $x_t = 1/2$, $t \in [T]$, by the Khintchine-Kahane inequality, (D.24) we have:

$$\mathbb{E}[R_T] = \mathbb{E}\Big[|\sum_{t=1}^{T} \sigma_t x_t|\Big] \geq \sqrt{\frac{1}{2}\sum_{t=1}^{T} x_t^2} = \sqrt{T/8},$$

which concludes the proof. □

More generally, for $T \geq N$, a lower bound of $R_T = \Omega(\sqrt{T \log N})$ can be proven for the regret of any algorithm.

8.2.4 Exponential weighted average algorithm

The WM algorithm can be extended to other loss functions L taking values in $[0,1]$. The Exponential Weighted Average algorithm presented here can be viewed as that extension for the case where L is convex in its first argument. Note that this algorithm is deterministic and yet, as we shall see, admits a very favorable regret

```
Exponential-Weighted-Average $(N)$
1  for $i \leftarrow 1$ to $N$ do
2      $w_{1,i} \leftarrow 1$
3  for $t \leftarrow 1$ to $T$ do
4      Receive$(x_t)$
5      $\widehat{y}_t \leftarrow \frac{\sum_{i=1}^N w_{t,i} y_{t,i}}{\sum_{i=1}^N w_{t,i}}$
6      Receive$(y_t)$
7      for $i \leftarrow 1$ to $N$ do
8          $w_{t+1,i} \leftarrow w_{t,i}\, e^{-\eta L(\widehat{y}_{t,i}, y_t)}$
9  return $\mathbf{w}_{T+1}$
```

Figure 8.5
Exponential weighted average, $L(\widehat{y}_{t,i}, y_t) \in [0, 1]$.

guarantee. Figure 8.5 gives its pseudocode. At round $t \in [T]$, the algorithm's prediction is

$$\widehat{y}_t = \frac{\sum_{i=1}^N w_{t,i} y_{t,i}}{\sum_{i=1}^N w_{t,i}}, \tag{8.13}$$

where $y_{t,i}$ is the prediction by expert i and $w_{t,i}$ the weight assigned by the algorithm to that expert. Initially, all weights are set to one. The algorithm then updates the weights at the end of round t according to the following rule:

$$w_{t+1,i} \leftarrow w_{t,i}\, e^{-\eta L(\widehat{y}_{t,i}, y_t)} = e^{-\eta L_{t,i}}, \tag{8.14}$$

where $L_{t,i}$ is the total loss incurred by expert i after t rounds. Note that this algorithm, as well as the others presented in this chapter, are simple, since they do not require keeping track of the losses incurred by each expert at all previous rounds but only of their cumulative performance. Furthermore, this property is also computationally advantageous. The following theorem presents a regret bound for this algorithm.

Theorem 8.6 *Assume that the loss function L is convex in its first argument and takes values in $[0, 1]$. Then, for any $\eta > 0$ and any sequence $y_1, \ldots, y_T \in \mathcal{Y}$, the regret of the Exponential Weighted Average algorithm after T rounds satisfies*

$$R_T \leq \frac{\log N}{\eta} + \frac{\eta T}{8}. \tag{8.15}$$

In particular, for $\eta = \sqrt{8 \log N/T}$, the regret is bounded as

$$R_T \leq \sqrt{(T/2) \log N}. \tag{8.16}$$

Proof: We apply the same potential function analysis as in previous proofs but using as potential $\Phi_t = \log \sum_{i=1}^N w_{t,i}$, $t \in [T]$. Let $\mathbf{p}_t$ denote the distribution over $\{1, \ldots, N\}$ with $p_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^N w_{t,i}}$. To derive an upper bound on Φ_t, we first examine the difference of two consecutive potential values:

$$\Phi_{t+1} - \Phi_t = \log \frac{\sum_{i=1}^N w_{t,i}\, e^{-\eta L(\widehat{y}_{t,i}, y_t)}}{\sum_{i=1}^N w_{t,i}} = \log \big(\mathop{\mathbb{E}}_{\mathbf{p}_t}[e^{\eta X}] \big),$$

with $X = -L(\widehat{y}_{t,i}, y_t) \in [-1, 0]$. To upper bound the expression appearing in the right-hand side, we apply Hoeffding's lemma (lemma D.1) to the centered random variable $X - \mathbb{E}_{\mathbf{p}_t}[X]$, then Jensen's inequality (theorem B.20) using the convexity of L with respect to its first argument:

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \log \big(\mathop{\mathbb{E}}_{\mathbf{p}_t} \big[e^{\eta(X - \mathbb{E}[X]) + \eta\, \mathbb{E}[X]} \big] \big) \\
&\leq \frac{\eta^2}{8} + \eta \mathop{\mathbb{E}}_{\mathbf{p}_t}[X] = \frac{\eta^2}{8} - \eta \mathop{\mathbb{E}}_{\mathbf{p}_t}[L(\widehat{y}_{t,i}, y_t)] && \text{(Hoeffding's lemma)} \\
&\leq -\eta L\big(\mathop{\mathbb{E}}_{\mathbf{p}_t}[\widehat{y}_{t,i}], y_t \big) + \frac{\eta^2}{8} && \text{(convexity of first arg. of } L\text{)} \\
&= -\eta L(\widehat{y}_t, y_t) + \frac{\eta^2}{8}.
\end{aligned}$$

Summing up these inequalities yields the following upper bound:

$$\Phi_{T+1} - \Phi_1 \leq -\eta \sum_{t=1}^T L(\widehat{y}_t, y_t) + \frac{\eta^2 T}{8}. \tag{8.17}$$

We obtain a lower bound for the same quantity as follows:

$$\Phi_{T+1} - \Phi_1 = \log \sum_{i=1}^N e^{-\eta L_{T,i}} - \log N \geq \log \max_{i=1}^N e^{-\eta L_{T,i}} - \log N = -\eta \min_{i=1}^N L_{T,i} - \log N.$$

Combining the upper and lower bounds yields:

$$\begin{aligned}
& -\eta \min_{i=1}^N L_{T,i} - \log N \leq -\eta \sum_{t=1}^T L(\widehat{y}_t, y_t) + \frac{\eta^2 T}{8} \\
\implies & \sum_{t=1}^T L(\widehat{y}_t, y_t) - \min_{i=1}^N L_{T,i} \leq \frac{\log N}{\eta} + \frac{\eta T}{8},
\end{aligned}$$

and concludes the proof. □

The optimal choice of η in theorem 8.6 requires knowledge of the horizon T, which is an apparent disadvantage of this analysis. However, we can use a standard *doubling trick* to eliminate this requirement, at the price of a small constant factor. This consists of dividing time into periods $[2^k, 2^{k+1}-1]$ of length 2^k with $k = 0, \ldots, n$ and $T \geq 2^n - 1$, and then choosing $\eta_k = \sqrt{\frac{8 \log N}{2^k}}$ in each period. The following theorem presents a regret bound when using the doubling trick to select η. A more general method consists of interpreting η as a function of time, i.e., $\eta_t = \sqrt{(8 \log N)/t}$, which can lead to a further constant factor improvement over the regret bound of the following theorem.

Theorem 8.7 *Assume that the loss function L is convex in its first argument and takes values in $[0, 1]$. Then, for any $T \geq 1$ and any sequence $y_1, \ldots, y_T \in \mathcal{Y}$, the regret of the Exponential Weighted Average algorithm after T rounds is bounded as follows:*

$$R_T \leq \frac{\sqrt{2}}{\sqrt{2}-1} \sqrt{(T/2) \log N} + \sqrt{\log N/2}. \tag{8.18}$$

Proof: Let $T \geq 1$ and let $\mathcal{I}_k = [2^k, 2^{k+1} - 1]$, for $k \in [0, n]$, with $n = \lfloor \log(T + 1) \rfloor$. Let $L_{\mathcal{I}_k}$ denote the loss incurred in the interval $\mathcal{I}_k$. By theorem 8.6 (8.16), for any $k \in \{0, \ldots, n\}$, we have

$$L_{\mathcal{I}_k} - \min_{i=1}^{N} L_{\mathcal{I}_k,i} \leq \sqrt{2^k/2 \, \log N}. \tag{8.19}$$

Thus, we can bound the total loss incurred by the algorithm after T rounds as:

$$\begin{aligned} L_T = \sum_{k=0}^{n} L_{\mathcal{I}_k} &\leq \sum_{k=0}^{n} \min_{i=1}^{N} L_{\mathcal{I}_k,i} + \sum_{k=0}^{n} \sqrt{2^k \, (\log N)/2} \\ &\leq \min_{i=1}^{N} L_{T,i} + \sqrt{(\log N)/2} \cdot \sum_{k=0}^{n} 2^{\frac{k}{2}}, \end{aligned} \tag{8.20}$$

where the second inequality follows from the super-additivity of min, that is $\min_i X_i + \min_i Y_i \leq \min_i (X_i + Y_i)$ for any sequences $(X_i)_i$ and $(Y_i)_i$, which implies $\sum_{k=0}^{n} \min_{i=1}^{N} L_{\mathcal{I}_k,i} \leq \min_{i=1}^{N} \sum_{k=0}^{n} L_{\mathcal{I}_k,i}$. The geometric sum appearing in the right-hand side of (8.20) can be expressed as follows:

$$\sum_{k=0}^{n} 2^{\frac{k}{2}} = \frac{2^{(n+1)/2} - 1}{\sqrt{2} - 1} \leq \frac{\sqrt{2}\sqrt{T+1} - 1}{\sqrt{2} - 1} \leq \frac{\sqrt{2}(\sqrt{T} + 1) - 1}{\sqrt{2} - 1} = \frac{\sqrt{2}\sqrt{T}}{\sqrt{2} - 1} + 1.$$

Plugging back into (8.20) and rearranging terms yields (8.18). □

The $O(\sqrt{T})$ dependency on T presented in this bound cannot be improved for general loss functions.

```
Perceptron($\mathbf{w}_0$)
1  $\mathbf{w}_1 \leftarrow \mathbf{w}_0$        $\triangleright$ typically $\mathbf{w}_0 = \mathbf{0}$
2  for $t \leftarrow 1$ to $T$ do
3      Receive($\mathbf{x}_t$)
4      $\widehat{y}_t \leftarrow \mathrm{sgn}(\mathbf{w}_t \cdot \mathbf{x}_t)$
5      Receive($y_t$)
6      if ($\widehat{y}_t \neq y_t$) then
7          $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t\mathbf{x}_t$    $\triangleright$ more generally $\eta y_t \mathbf{x}_t, \eta > 0$.
8      else $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
9  return $\mathbf{w}_{T+1}$
```

Figure 8.6
Perceptron algorithm.

8.3 Linear classification

This section presents two well-known on-line learning algorithms for linear classification: the Perceptron and Winnow algorithms.

8.3.1 Perceptron algorithm

The Perceptron algorithm is one of the earliest machine learning algorithms. It is an on-line linear classification algorithm. Thus, it learns a decision function based on a hyperplane by processing training points one at a time. Figure 8.6 gives its pseudocode.

The algorithm maintains a weight vector $\mathbf{w}_t \in \mathbb{R}^N$ defining the hyperplane learned, starting with an arbitrary vector $\mathbf{w}_0$. At each round $t \in [T]$, it predicts the label of the point $\mathbf{x}_t \in \mathbb{R}^N$ received, using the current vector $\mathbf{w}_t$ (line 4). When the prediction made does not match the correct label (lines 6-7), it updates $\mathbf{w}_t$ by adding $y_t\mathbf{x}_t$. More generally, when a learning rate $\eta > 0$ is used, the vector added is $\eta y_t \mathbf{x}_t$. This update can be partially motivated by examining the inner product of the current weight vector with $y_t\mathbf{x}_t$, whose sign determines the classification of $\mathbf{x}_t$. Just before an update, $\mathbf{x}_t$ is misclassified and thus $y_t\mathbf{w}_t \cdot \mathbf{x}_t$ is negative; afterward, $y_t\mathbf{w}_{t+1} \cdot \mathbf{x}_t = y_t\mathbf{w}_t \cdot \mathbf{x}_t + \eta\|\mathbf{x}_t\|^2$, thus, the update corrects the weight vector in the direction of making the inner product $y_t\mathbf{w}_t \cdot \mathbf{x}_t$ positive by augmenting it with the quantity $\eta\|\mathbf{x}_t\|^2 > 0$.

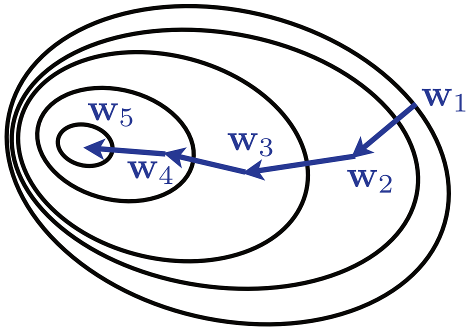


Figure 8.7
An example path followed by the iterative stochastic gradient descent technique. Each inner contour indicates a region of lower elevation.

The Perceptron algorithm can be shown in fact to seek a weight vector $\mathbf{w}$ minimizing an objective function F precisely based on the quantities $(-y_t\mathbf{w}\cdot\mathbf{x}_t)$, $t\in[T]$. Since $(-y_t\mathbf{w}\cdot\mathbf{x}_t)$ is positive when $\mathbf{x}_t$ is misclassified by $\mathbf{w}$, F is defined for all $\mathbf{w}\in\mathbb{R}^N$ by

$$F(\mathbf{w}) = \frac{1}{T}\sum_{t=1}^{T}\max\Big(0, -y_t(\mathbf{w}\cdot\mathbf{x}_t)\Big) = \mathop{\mathbb{E}}_{\mathbf{x}\sim\widehat{\mathcal{D}}}[\widetilde{F}(\mathbf{w},\mathbf{x})], \tag{8.21}$$

where $\widetilde{F}(\mathbf{w},\mathbf{x}) = \max\big(0, -f(\mathbf{x})(\mathbf{w}\cdot\mathbf{x})\big)$ with $f(\mathbf{x})$ denoting the label of $\mathbf{x}$, and $\widehat{\mathcal{D}}$ is the empirical distribution associated with the sample $(\mathbf{x}_1,\ldots,\mathbf{x}_T)$. For any $t\in[T]$, $\mathbf{w}\mapsto -y_t(\mathbf{w}\cdot\mathbf{x}_t)$ is linear and thus convex. Since the max operator preserves convexity, this shows that F is convex. However, F is not differentiable. Nevertheless, the Perceptron algorithm coincides with the application of the *stochastic subgradient descent* technique to F.

The stochastic (or on-line) subgradient descent technique examines one point $\mathbf{x}_t$ at a time. Note, the function $\widetilde{F}(\cdot,\mathbf{x}_t)$ is non-differentiable for any $\mathbf{w}_t$ where $\mathbf{w}_t\cdot\mathbf{x}_t = 0$. In such a case any subgradient of $\widetilde{F}$, i.e. any vector in the convex hull of $\mathbf{0}$ and $-y_t\mathbf{x}_t$, may be used for the update step (see B.4.1). Choosing the subgradient $-y_t\mathbf{x}_t$, we arrive at the following general update for each point $\mathbf{x}_t$:

$$\mathbf{w}_{t+1} \leftarrow \begin{cases}\mathbf{w}_t - \eta\nabla_{\mathbf{w}}\widetilde{F}(\mathbf{w}_t,\mathbf{x}_t) & \text{if } \mathbf{w}_t\cdot\mathbf{x}_t \neq 0\\ \mathbf{w}_t + \eta y_t\mathbf{x}_t & \text{otherwise,}\end{cases} \tag{8.22}$$

where $\eta > 0$ is a learning rate parameter. Figure 8.7 illustrates an example path the gradient descent follows. In the specific case we are considering, $\mathbf{w}\mapsto\widetilde{F}(\mathbf{w},\mathbf{x}_t)$ is differentiable at any $\mathbf{w}$ such that $y_t(\mathbf{w}\cdot\mathbf{x}_t)\neq 0$ with $\nabla_{\mathbf{w}}\widetilde{F}(\mathbf{w},\mathbf{x}_t) = -y\mathbf{x}_t$ if $y_t(\mathbf{w}\cdot\mathbf{x}_t) < 0$ and $\nabla_{\mathbf{w}}\widetilde{F}(\mathbf{w},\mathbf{x}_t) = 0$ if $y_t(\mathbf{w}\cdot\mathbf{x}_t) > 0$. Thus, the stochastic gradient descent update becomes

$$\mathbf{w}_{t+1} \leftarrow \begin{cases} \mathbf{w}_t + \eta y_t \mathbf{x}_t & \text{if } y_t(\mathbf{w}_t \cdot \mathbf{x}_t) \leq 0; \\ \mathbf{w}_t & \text{if } y_t(\mathbf{w}_t \cdot \mathbf{x}_t) > 0, \end{cases} \tag{8.23}$$

which coincides exactly with the update of the Perceptron algorithm.

The following theorem gives a margin-based upper bound on the number of mistakes or updates made by the Perceptron algorithm when processing a sequence of T points that can be linearly separated by a hyperplane with margin $\rho > 0$.

Theorem 8.8 *Let $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^N$ be a sequence of T points with $\|\mathbf{x}_t\| \leq r$ for all $t \in [T]$, for some $r > 0$. Assume that there exist $\rho > 0$ and $\mathbf{v} \in \mathbb{R}^N$ such that for all $t \in [T]$, $\rho \leq \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{\|\mathbf{v}\|}$. Then, the number of updates made by the Perceptron algorithm when processing $\mathbf{x}_1, \ldots, \mathbf{x}_T$ is bounded by r^2/ρ^2.*

Proof: Let $\mathcal{I}$ be the subset of the T rounds at which there is an update, and let M be the total number of updates, i.e., $|\mathcal{I}| = M$. Summing up the assumption inequalities yields:

$$\begin{aligned} M\rho \leq \frac{\mathbf{v} \cdot \sum_{t \in \mathcal{I}} y_t \mathbf{x}_t}{\|\mathbf{v}\|} &\leq \Big\| \sum_{t \in \mathcal{I}} y_t \mathbf{x}_t \Big\| && \text{(Cauchy-Schwarz inequality)} \\ &= \Big\| \sum_{t \in \mathcal{I}} (\mathbf{w}_{t+1} - \mathbf{w}_t) \Big\| && \text{(definition of updates)} \\ &= \|\mathbf{w}_{T+1}\| && \text{(telescoping sum, } \mathbf{w}_0 = 0) \\ &= \sqrt{\sum_{t \in \mathcal{I}} \|\mathbf{w}_{t+1}\|^2 - \|\mathbf{w}_t\|^2} && \text{(telescoping sum, } \mathbf{w}_0 = 0) \\ &= \sqrt{\sum_{t \in \mathcal{I}} \|\mathbf{w}_t + y_t \mathbf{x}_t\|^2 - \|\mathbf{w}_t\|^2} && \text{(definition of updates)} \\ &= \sqrt{\sum_{t \in \mathcal{I}} 2 \underbrace{y_t \mathbf{w}_t \cdot \mathbf{x}_t}_{\leq 0} + \|\mathbf{x}_t\|^2} \\ &\leq \sqrt{\sum_{t \in \mathcal{I}} \|\mathbf{x}_t\|^2} \leq \sqrt{M r^2}. \end{aligned}$$

Comparing the left- and right-hand sides gives $\sqrt{M} \leq r/\rho$, that is, $M \leq r^2/\rho^2$. □

By definition of the algorithm, the weight vector $\mathbf{w}_T$ after processing T points is a linear combination of the vectors $\mathbf{x}_t$ at which an update was made: $\mathbf{w}_T = \sum_{t \in \mathcal{I}} y_t \mathbf{x}_t$. Thus, as in the case of SVMs, these vectors can be referred to as *support vectors* for the Perceptron algorithm.

The bound of theorem 8.8 is remarkable, since it depends only on the normalized margin ρ/r and not on the dimension N of the space. This bound can be shown to be tight, that is the number of updates can be equal to r^2/ρ^2 in some instances (see exercise 8.3 to show the upper bound is tight).

The theorem required no assumption about the sequence of points $\mathbf{x}_1, \ldots, \mathbf{x}_T$. A standard setting for the application of the Perceptron algorithm is one where a finite sample S of size $m < T$ is available and where the algorithm makes multiple passes over these m points. The result of the theorem implies that when S is linearly separable, the Perceptron algorithm converges after a finite number of updates and thus passes. For a small margin ρ, the convergence of the algorithm can be quite slow, however. In fact, for some samples, regardless of the order in which the points in S are processed, the number of updates made by the algorithm is in $\Omega(2^N)$ (see exercise 8.1). Of course, if S is not linearly separable, the Perceptron algorithm does not converge. In practice, it is stopped after some number of passes over S.

There are many variants of the standard Perceptron algorithm which are used in practice and have been theoretically analyzed. One notable example is the *voted Perceptron algorithm*, which predicts according to the rule $\operatorname{sgn}\big((\sum_{t \in \mathcal{I}} c_t \mathbf{w}_t) \cdot \mathbf{x}\big)$, where c_t is a weight proportional to the number of iterations that $\mathbf{w}_t$ survives, i.e., the number of iterations between $\mathbf{w}_t$ and $\mathbf{w}_{t+1}$.

For the following theorem, we consider the case where the Perceptron algorithm is trained via multiple passes till convergence over a finite sample that is linearly separable. In view of theorem 8.8, convergence occurs after a finite number of updates.

For a linearly separable sample S, we denote by r_S the radius of the smallest origin-centered sphere containing all points in S and by ρ_S the largest margin of a separating hyperplane for S. We also denote by $M(S)$ the number of updates made by the algorithm after training over S.

Theorem 8.9 *Assume that the data is linearly separable. Let h_S be the hypothesis returned by the Perceptron algorithm after training over a sample S of size m drawn according to some distribution $\mathcal{D}$. Then, the expected error of h_S is bounded as follows:*

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[R(h_S)] \leq \mathop{\mathbb{E}}_{S \sim \mathcal{D}^{m+1}} \left[\frac{\min\big(M(S), r_S^2/\rho_S^2\big)}{m+1} \right].$$

Proof: Let S be a linearly separable sample of size $m+1$ drawn i.i.d. according to $\mathcal{D}$ and let $\mathbf{x}$ be a point in S. If $h_{S-\{\mathbf{x}\}}$ misclassifies $\mathbf{x}$, then $\mathbf{x}$ must be a support vector for h_S. Thus, the leave-one-out error of the Perceptron algorithm on sample

S is at most $\frac{M(S)}{m+1}$. The result then follows lemma 5.3, which relates the expected leave-one-out error to the expected error, along with the upper bound on $M(S)$ given by theorem 8.8. □

This result can be compared with a similar one given for the SVM algorithm (with no offset) in the following theorem, which is an extension of theorem 5.4. We denote by $N_{\text{SV}}(S)$ the number of support vectors that define the hypothesis h_S returned by SVMs when trained on a sample S.

Theorem 8.10 *Assume that the data is linearly separable. Let h_S be the hypothesis returned by SVMs used with no offset ($b = 0$) after training over a sample S of size m drawn according to some distribution $\mathcal{D}$. Then, the expected error of h_S is bounded as follows:*

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[R(h_S)] \leq \mathop{\mathbb{E}}_{S\sim\mathcal{D}^{m+1}}\left[\frac{\min\left(N_{SV}(S), r_S^2/\rho_S^2\right)}{m+1}\right].$$

Proof: The fact that the expected error can be upper bounded by the average fraction of support vectors ($N_{\text{SV}}(S)/(m+1)$) was already shown by theorem 5.4. Thus, it suffices to show that it is also upper bounded by the expected value of $(r_S^2/\rho_S^2)/(m+1)$. To do so, we will bound the leave-one-out error of the SVM algorithm for a sample S of size $m+1$ by $(r_S^2/\rho_S^2)/(m+1)$. The result will then follow by lemma 5.3, which relates the expected leave-one-out error to the expected error.

Let $S = (\mathbf{x}_1, \ldots, \mathbf{x}_{m+1})$ be a linearly separable sample drawn i.i.d. according to $\mathcal{D}$ and let $\mathbf{x}$ be a point in S that is misclassified by $h_{S-\{\mathbf{x}\}}$. We will analyze the case where $\mathbf{x} = \mathbf{x}_{m+1}$, the analysis of other cases is similar. We denote by S' the sample $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$.

For any $q \in [m+1]$, let G_q denote the function defined over $\mathbb{R}^q$ by $G_q \colon \boldsymbol{\alpha} \mapsto \sum_{i=1}^q \alpha_i - \frac{1}{2}\sum_{i,j=1}^q \alpha_i\alpha_j y_i y_j(\mathbf{x}_i \cdot \mathbf{x}_j)$. Then, G_{m+1} is the objective function of the dual optimization problem for SVMs associated to the sample S and G_m the one for the sample S'. Let $\boldsymbol{\alpha} \in \mathbb{R}^{m+1}$ denote a solution of the dual SVM problem $\max_{\boldsymbol{\alpha}\geq 0} G_{m+1}(\boldsymbol{\alpha})$ and $\boldsymbol{\alpha}' \in \mathbb{R}^{m+1}$ the vector such that $(\alpha'_1, \ldots, \alpha'_m)^\top \in \mathbb{R}^m$ is a solution of $\max_{\boldsymbol{\alpha}\geq 0} G_m(\boldsymbol{\alpha})$ and $\alpha'_{m+1} = 0$. Let $\mathbf{e}_{m+1}$ denote the $(m+1)$th unit vector in $\mathbb{R}^{m+1}$. By definition of $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ as maximizers, $\max_{\beta\geq 0} G_{m+1}(\boldsymbol{\alpha}' + \beta\mathbf{e}_{m+1}) \leq G_{m+1}(\boldsymbol{\alpha})$ and $G_{m+1}(\boldsymbol{\alpha} - \alpha_{m+1}\mathbf{e}_{m+1}) \leq G_m(\boldsymbol{\alpha}')$. Thus, the quantity $A = G_{m+1}(\boldsymbol{\alpha}) - G_m(\boldsymbol{\alpha}')$ admits the following lower and upper bounds:

$$\max_{\beta\geq 0} G_{m+1}(\boldsymbol{\alpha}' + \beta\mathbf{e}_{m+1}) - G_m(\boldsymbol{\alpha}') \leq A \leq G_{m+1}(\boldsymbol{\alpha}) - G_{m+1}(\boldsymbol{\alpha} - \alpha_{m+1}\mathbf{e}_{m+1}).$$

Let $\mathbf{w} = \sum_{i=1}^{m+1} y_i\alpha_i\mathbf{x}_i$ denote the weight vector returned by SVMs for the sample S. Since $h_{S'}$ misclassifies $\mathbf{x}_{m+1}$, $\mathbf{x}_{m+1}$ must be a support vector for h_S, thus

$y_{m+1}\mathbf{w} \cdot \mathbf{x}_{m+1} = 1$. In view of that, the upper bound can be rewritten as follows:

$$\begin{aligned}
&G_{m+1}(\boldsymbol{\alpha}) - G_{m+1}(\boldsymbol{\alpha} - \alpha_{m+1}\mathbf{e}_{m+1}) \\
&\qquad = \alpha_{m+1} - \sum_{i=1}^{m+1} (y_i \alpha_i \mathbf{x}_i) \cdot (y_{m+1}\alpha_{m+1}\mathbf{x}_{m+1}) + \frac{1}{2}\alpha_{m+1}^2 \|\mathbf{x}_{m+1}\|^2 \\
&\qquad = \alpha_{m+1}(1 - y_{m+1}\mathbf{w} \cdot \mathbf{x}_{m+1}) + \frac{1}{2}\alpha_{m+1}^2 \|\mathbf{x}_{m+1}\|^2 \\
&\qquad = \frac{1}{2}\alpha_{m+1}^2 \|\mathbf{x}_{m+1}\|^2.
\end{aligned}$$

Similarly, let $\mathbf{w}' = \sum_{i=1}^{m} y_i \alpha'_i \mathbf{x}_i$. Then, for any $\beta \geq 0$, the quantity maximized in the lower bound can be written as

$$\begin{aligned}
&G_{m+1}(\boldsymbol{\alpha}' + \beta\mathbf{e}_{m+1}) - G_m(\boldsymbol{\alpha}') \\
&\qquad = \beta\big(1 - y_{m+1}(\mathbf{w}' + \beta y_{m+1}\mathbf{x}_{m+1}) \cdot \mathbf{x}_{m+1}\big) + \frac{1}{2}\beta^2 \|\mathbf{x}_{m+1}\|^2 \\
&\qquad = \beta(1 - y_{m+1}\mathbf{w}' \cdot \mathbf{x}_{m+1}) - \frac{1}{2}\beta^2 \|\mathbf{x}_{m+1}\|^2.
\end{aligned}$$

The right-hand side is maximized for the following value of β: $\frac{1 - y_{m+1}\mathbf{w}' \cdot \mathbf{x}_{m+1}}{\|\mathbf{x}_{m+1}\|^2}$. Plugging in this value in the right-hand side gives $\frac{1}{2}\frac{(1 - y_{m+1}\mathbf{w}' \cdot \mathbf{x}_{m+1})^2}{\|\mathbf{x}_{m+1}\|^2}$. Thus,

$$A \geq \frac{1}{2}\frac{(1 - y_{m+1}\mathbf{w}' \cdot \mathbf{x}_{m+1})^2}{\|\mathbf{x}_{m+1}\|^2} \geq \frac{1}{2\|\mathbf{x}_{m+1}\|^2},$$

using the fact that $y_{m+1}\mathbf{w}' \cdot \mathbf{x}_{m+1} < 0$, since $\mathbf{x}_{m+1}$ is misclassified by $\mathbf{w}'$. Comparing this lower bound on A with the upper bound previously derived leads to $\frac{1}{2\|\mathbf{x}_{m+1}\|^2} \leq \frac{1}{2}\alpha_{m+1}^2 \|\mathbf{x}_{m+1}\|^2$, that is

$$\alpha_{m+1} \geq \frac{1}{\|\mathbf{x}_{m+1}\|^2} \geq \frac{1}{r_S^2}.$$

The analysis carried out in the case $\mathbf{x} = \mathbf{x}_{m+1}$ holds similarly for any $\mathbf{x}_i$ in S that is misclassified by $h_{S - \{\mathbf{x}_i\}}$. Let $\mathcal{I}$ denote the set of such indices i. Then, we can write:

$$\sum_{i \in \mathcal{I}} \alpha_i \geq \frac{|\mathcal{I}|}{r_S^2}.$$

By (5.19), the following simple expression holds for the margin: $\sum_{i=1}^{m+1} \alpha_i = 1/\rho_S^2$. Using this identity leads to

$$|\mathcal{I}| \leq r_S^2 \sum_{i \in \mathcal{I}} \alpha_i \leq r_S^2 \sum_{i=1}^{m+1} \alpha_i = \frac{r_S^2}{\rho_S^2}.$$

Since by definition $|\mathcal{I}|$ is the total number of leave-one-out errors, this concludes the proof. □

Thus, the guarantees given by theorem 8.9 and theorem 8.10 in the separable case have a similar form. These bounds do not seem sufficient to distinguish the effectiveness of the SVM and Perceptron algorithms. Note, however, that while the same margin quantity ρ_S appears in both bounds, the radius r_S can be replaced by a finer quantity that is different for the two algorithms: in both cases, instead of the radius of the sphere containing all sample points, r_S can be replaced by the radius of the sphere containing the support vectors, as can be seen straightforwardly from the proof of the theorems. Thus, the position of the support vectors in the case of SVMs can provide a more favorable guarantee than that of the support vectors (update vectors) for the Perceptron algorithm. Finally, the guarantees given by these theorems are somewhat weak. These are not high probability bounds, they hold only for the expected error of the hypotheses returned by the algorithms and in particular provide no information about the variance of their error.

The following two theorems give bounds on the number of updates or mistakes made by the Perceptron algorithm in the more general scenario of a non-linearly separable sample in terms of the ρ-Hinge losses of an arbitrary weight vector $\mathbf{v}$.

Theorem 8.11 *Let $\mathcal{I}$ denote the set of indices $t \in [T]$ at which the Perceptron algorithm makes an update when processing a sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$ with $\|\mathbf{x}_t\| \leq r$ for some $r > 0$. Then, the number of updates $M = |\mathcal{I}|$ made by the algorithm can be bounded as follows:*

$$M \leq \inf_{\rho>0, \|\mathbf{v}\|_2 \leq 1} \left[\frac{\frac{r}{\rho} + \sqrt{\frac{r^2}{\rho^2} + 4\|\mathbf{l}_\rho\|_1}}{2} \right]^2 \leq \inf_{\rho>0, \|v\|_2 \leq 1} \left(\frac{r}{\rho} + \sqrt{\|\mathbf{l}_\rho\|_1} \right)^2 ,$$

where $\mathbf{l}_\rho = (l_t)_{t \in \mathcal{I}}$ with $l_t = \max\left\{0, 1 - \frac{y_t(\mathbf{v}\cdot\mathbf{x}_t)}{\rho}\right\}$.

Proof: Fix $\rho > 0$ and $\mathbf{v}$ with $\|\mathbf{v}\|_2 = 1$. By definition of l_t, for any t, we have $1 - \frac{y_t(\mathbf{v}\cdot\mathbf{x}_t)}{\rho} \leq l_t$. Summing up these inequalities over all $t \in \mathcal{I}$ yields

$$\begin{aligned} M &\leq \sum_{t\in\mathcal{I}} l_t + \sum_{t\in\mathcal{I}} \frac{y_t(\mathbf{v}\cdot\mathbf{x}_t)}{\rho} \\ &= \|\mathbf{l}_\rho\|_1 + \sum_{t\in\mathcal{I}} \frac{y_t(\mathbf{v}\cdot\mathbf{x}_t)}{\rho} \leq \|\mathbf{l}_\rho\|_1 + \frac{\sqrt{Mr^2}}{\rho} , \end{aligned} \tag{8.24}$$

where the last inequality holds by the bound shown in the proof of the separable case (theorem 8.8): $\frac{\mathbf{v}\cdot\sum_{t\in\mathcal{I}} y_t\mathbf{x}_t}{\|\mathbf{v}\|} \leq \sqrt{Mr^2}$. Now, solving the resulting second-degree inequality $M \leq \|\mathbf{l}_\rho\|_1 + \frac{\sqrt{Mr^2}}{\rho}$ gives $\sqrt{M} \leq \frac{1}{2}\left(\frac{r}{\rho} + \sqrt{\frac{r^2}{\rho^2} + 4\|\mathbf{l}_\rho\|_1}\right)$, which proves the first inequality. The second inequality follows from the sub-additivity of the square-root function. □

Theorem 8.12 *Let $\mathcal{I}$ denote the set of indices $t \in [T]$ at which the Perceptron algorithm makes an update when processing a sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$ with $\|\mathbf{x}_t\| \leq r$ for some $r > 0$. Then, the number of updates $M = |\mathcal{I}|$ made by the algorithm can be bounded as follows:*

$$M \leq \inf_{\rho>0, \|v\|_2 \leq 1} \left(\frac{r}{\rho} + \|\mathbf{l}_\rho\|_2 \right)^2 ,$$

where $\mathbf{l}_\rho = (l_t)_{t \in \mathcal{I}}$ with $l_t = \max\left\{0, 1 - \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{\rho}\right\}$.

Proof: Fix $\rho > 0$ and $\mathbf{v}$ with $\|\mathbf{v}\|_2 = 1$. Starting with line (8.24) of theorem 8.11 and using $\|\mathbf{l}_\rho\|_1 \leq \sqrt{M} \|\mathbf{l}_\rho\|_2$, which holds by the Cauchy-Schwarz inequality, give

$$M \leq \|\mathbf{l}_\rho\|_1 + \frac{\sqrt{Mr^2}}{\rho} \leq \sqrt{M} \|\mathbf{l}_\rho\|_2 + \frac{\sqrt{Mr^2}}{\rho} .$$

This implies $\sqrt{M} \leq \|\mathbf{l}_\rho\|_2 + \frac{\sqrt{r^2}}{\rho}$ and proves the statement. □

These bounds strictly generalize the bounds given in the separable case (theorem 8.8) since in that case the vector $\mathbf{v}$ can be chosen to be that of a maximum-margin hyperplane with no Hinge loss at any point. The main difference between the two bounds is the L_1-norm of the vector of Hinge losses in Theorem 8.11 versus the L_2-norm in Theorem 8.12. Note that, since the L_2-norm bound follows from upper bounding inequality (8.24), which is equivalent to the first inequality of Theorem 8.11, the first L_1-norm bound of Theorem 8.11 is always tighter than the L_2-norm bound of Theorem 8.12.

The Perceptron algorithm can be generalized, as in the case of SVMs, to define a linear separation in a high-dimensional space. It admits an equivalent dual form, the dual Perceptron algorithm, which is presented in figure 8.8. The dual Perceptron algorithm maintains a vector $\boldsymbol{\alpha} \in \mathbb{R}^T$ of coefficients assigned to each point $\mathbf{x}_t$, $t \in [T]$. The label of a point $\mathbf{x}_t$ is predicted according to the rule $\text{sgn}(\mathbf{w} \cdot \mathbf{x}_t)$, where $\mathbf{w} = \sum_{s=1}^T \alpha_s y_s \mathbf{x}_s$. The coefficient α_t is incremented by one when this prediction does not match the correct label. Thus, an update for $\mathbf{x}_t$ is equivalent to augmenting the weight vector $\mathbf{w}$ with $y_t \mathbf{x}_t$, which shows that the dual algorithm matches exactly the standard Perceptron algorithm. The dual Perceptron algorithm can be written solely in terms of inner products between training instances. Thus, as in the case of SVMs, instead of the inner product between points in the input space, an arbitrary PDS kernel can be used, which leads to the kernel Perceptron algorithm detailed in figure 8.9. The kernel Perceptron algorithm and its average variant, i.e., voted Perceptron with uniform weights c_t, are commonly used algorithms in a variety of applications.

```
DualPerceptron($\boldsymbol{\alpha}_0$)
1  $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}_0$        $\triangleright$ typically $\boldsymbol{\alpha}_0 = \mathbf{0}$
2  for $t \leftarrow 1$ to $T$ do
3      Receive($\mathbf{x}_t$)
4      $\widehat{y}_t \leftarrow \mathrm{sgn}(\sum_{s=1}^{T} \alpha_s y_s (\mathbf{x}_s \cdot \mathbf{x}_t))$
5      Receive($y_t$)
6      if ($\widehat{y}_t \neq y_t$) then
7          $\alpha_t \leftarrow \alpha_t + 1$
8      else $\alpha_t \leftarrow \alpha_t$
9  return $\boldsymbol{\alpha}$
```

Figure 8.8
Dual Perceptron algorithm.

8.3.2 Winnow algorithm

This section presents an alternative on-line linear classification algorithm, the *Winnow algorithm*. Thus, it learns a weight vector defining a separating hyperplane by sequentially processing the training points. As suggested by the name, the algorithm is particularly well suited to cases where a relatively small number of dimensions or experts can be used to define an accurate weight vector. Many of the other dimensions may then be irrelevant.

The Winnow algorithm is similar to the Perceptron algorithm, but, instead of the additive update of the weight vector in the Perceptron case, Winnow's update is multiplicative. The pseudocode of the algorithm is given in figure 8.10. The algorithm takes as input a learning parameter $\eta > 0$. It maintains a non-negative weight vector $\mathbf{w}_t$ with components summing to one ($\|\mathbf{w}_t\|_1 = 1$) starting with the uniform weight vector (line 1). At each round $t \in [T]$, if the prediction does not match the correct label (line 6), each component $w_{t,i}$, $i \in [N]$, is updated by multiplying it by $\exp(\eta y_t x_{t,i})$ and dividing by the normalization factor Z_t to ensure that the weights sum to one (lines 7–9). Thus, if the label y_t and $x_{t,i}$ share the same sign, then $w_{t,i}$ is increased, while, in the opposite case, it is significantly decreased.

The Winnow algorithm is closely related to the WM algorithm: when $\mathbf{x}_{t,i} \in \{-1, +1\}$, $\mathrm{sgn}(\mathbf{w}_t \cdot \mathbf{x}_t)$ coincides with the majority vote, since multiplying the weight of correct or incorrect experts by e^{η} or $e^{-\eta}$ is equivalent to multiplying the weight

```
KernelPerceptron($\boldsymbol{\alpha}_0$)
1  $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}_0$    $\triangleright$ typically $\boldsymbol{\alpha}_0 = \mathbf{0}$
2  for $t \leftarrow 1$ to $T$ do
3      Receive($x_t$)
4      $\widehat{y}_t \leftarrow \operatorname{sgn}(\sum_{s=1}^{T} \alpha_s y_s K(x_s, x_t))$
5      Receive($y_t$)
6      if ($\widehat{y}_t \neq y_t$) then
7          $\alpha_t \leftarrow \alpha_t + 1$
8      else $\alpha_t \leftarrow \alpha_t$
9  return $\boldsymbol{\alpha}$
```

Figure 8.9
Kernel Perceptron algorithm for PDS kernel K.

of incorrect ones by $\beta = e^{-2\eta}$. The multiplicative update rule of Winnow is of course also similar to that of AdaBoost.

The following theorem gives a mistake bound for the Winnow algorithm in the separable case, which is similar in form to the bound of theorem 8.8 for the Perceptron algorithm.

Theorem 8.13 *Let $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^N$ be a sequence of T points with $\|x_t\|_\infty \leq r_\infty$ for all $t \in [T]$, for some $r_\infty > 0$. Assume that there exist $\mathbf{v} \in \mathbb{R}^N$, $\mathbf{v} \geq 0$, and $\rho_\infty > 0$ such that for all $t \in [T]$, $\rho_\infty \leq \frac{y_t(\mathbf{v} \cdot \mathbf{x}_t)}{\|\mathbf{v}\|_1}$. Then, for $\eta = \frac{\rho_\infty}{r_\infty^2}$, the number of updates made by the Winnow algorithm when processing $\mathbf{x}_1, \ldots, \mathbf{x}_T$ is upper bounded by $2\,(r_\infty^2/\rho_\infty^2) \log N$.*

Proof: Let $\mathcal{I} \subseteq [T]$ be the set of iterations at which there is an update, and let M be the total number of updates, i.e., $|\mathcal{I}| = M$. The potential function Φ_t, $t \in [T]$, used for this proof is the relative entropy of the distribution defined by the normalized weights $v_i/\|\mathbf{v}\|_1 \geq 0$, $i \in [N]$, and the one defined by the components of the weight vector $w_{t,i}$, $i \in [N]$:

$$\Phi_t = \sum_{i=1}^{N} \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{v_i/\|\mathbf{v}\|_1}{w_{t,i}} .$$

To derive an upper bound on Φ_t, we analyze the difference of the potential functions at two consecutive rounds. For all $t \in \mathcal{I}$, this difference can be expressed and

```
WINNOW($\eta$)
 1  $w_1 \leftarrow \mathbf{1}/N$
 2  for $t \leftarrow 1$ to $T$ do
 3      RECEIVE($\mathbf{x}_t$)
 4      $\widehat{y}_t \leftarrow \operatorname{sgn}(\mathbf{w}_t \cdot \mathbf{x}_t)$
 5      RECEIVE($y_t$)
 6      if ($\widehat{y}_t \neq y_t$) then
 7          $Z_t \leftarrow \sum_{i=1}^N w_{t,i} \exp(\eta y_t x_{t,i})$
 8          for $i \leftarrow 1$ to $N$ do
 9              $w_{t+1,i} \leftarrow \frac{w_{t,i} \exp(\eta y_t x_{t,i})}{Z_t}$
10      else $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
11  return $\mathbf{w}_{T+1}$
```

Figure 8.10
Winnow algorithm, with $y_t \in \{-1, +1\}$ for all $t \in [T]$.

bounded as follows:

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \sum_{i=1}^N \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{w_{t,i}}{w_{t+1,i}} \\
&= \sum_{i=1}^N \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{Z_t}{\exp(\eta y_t x_{t,i})} \\
&= \log Z_t - \eta \sum_{i=1}^N \frac{v_i}{\|\mathbf{v}\|_1} y_t x_{t,i} \\
&\leq \log \Big[\sum_{i=1}^N w_{t,i} \exp(\eta y_t x_{t,i}) \Big] - \eta \rho_\infty \\
&= \log \mathop{\mathbb{E}}_{i \sim \mathbf{w}_t} \big[\exp(\eta y_t x_{t,i}) \big] - \eta \rho_\infty \\
&= \log \mathop{\mathbb{E}}_{i \sim \mathbf{w}_t} \big[\exp(\eta y_t x_{t,i} - \eta y_t \mathbf{w}_t \cdot x_t + \eta y_t \mathbf{w}_t \cdot x_t) \big] - \eta \rho_\infty \\
&\leq \log \big[\exp(\eta^2 (2r_\infty)^2/8) \big] + \underbrace{\eta y_t (\mathbf{w}_t \cdot x_t)}_{\leq 0} - \eta \rho_\infty \\
&\leq \eta^2 r_\infty^2 / 2 - \eta \rho_\infty .
\end{aligned}$$

The first inequality follows the definition of ρ_∞. The subsequent equality rewrites the summation as an expectation over the distribution defined by $\mathbf{w}_t$. The next inequality uses Hoeffding's lemma (lemma D.1) and the last one the fact that there has been an update at t, which implies $y_t(\mathbf{w}_t \cdot x_t) \leq 0$. Summing up these inequalities over all $t \in \mathcal{I}$ yields:

$$\Phi_{T+1} - \Phi_1 \leq M(\eta^2 r_\infty^2/2 - \eta\rho_\infty).$$

Next, we derive a lower bound by noting that

$$\Phi_1 = \sum_{i=1}^N \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{v_i/\|\mathbf{v}\|_1}{1/N} = \log N + \sum_{i=1}^N \frac{v_i}{\|\mathbf{v}\|_1} \log \frac{v_i}{\|\mathbf{v}\|_1} \leq \log N\,.$$

Additionally, since the relative entropy is always non-negative, we have $\Phi_{T+1} \geq 0$. This yields the following lower bound:

$$\Phi_{T+1} - \Phi_1 \geq 0 - \log N = -\log N\,.$$

Combining the upper and lower bounds we see that $-\log N \leq M(\eta^2 r_\infty^2/2 - \eta\rho_\infty)$. Setting $\eta = \frac{\rho_\infty}{r_\infty^2}$ yields the statement of the theorem. □

The margin-based mistake bounds of theorem 8.8 and theorem 8.13 for the Perceptron and Winnow algorithms have a similar form, but they are based on different norms. For both algorithms, the norm $\|\cdot\|_p$ used for the input vectors $\mathbf{x}_t$, $t \in [T]$, is the dual of the norm $\|\cdot\|_q$ used for the margin vector $\mathbf{v}$, that is p and q are conjugate: $1/p + 1/q = 1$: in the case of the Perceptron algorithm $p = q = 2$, while for Winnow $p = \infty$ and $q = 1$.

These bounds imply different types of guarantees. The bound for Winnow is favorable when a sparse set of the experts $i \in [N]$ can predict well. For example, if $\mathbf{v} = \mathbf{e}_1$ where $\mathbf{e}_1$ is the unit vector along the first axis in $\mathbb{R}^N$ and if $\mathbf{x}_t \in \{-1, +1\}^N$ for all t, then the upper bound on the number of mistakes given for Winnow by theorem 8.13 is only $2\log N$, while the upper bound of theorem 8.8 for the Perceptron algorithm is N. The guarantee for the Perceptron algorithm is more favorable in the opposite situation, where sparse solutions are not effective.

8.4 On-line to batch conversion

The previous sections presented several algorithms for the scenario of on-line learning, including the Perceptron and Winnow algorithms, and analyzed their behavior within the mistake model, where no assumption is made about the way the training sequence is generated. Can these algorithms be used to derive hypotheses with small generalization error in the standard stochastic setting? How can the intermediate hypotheses they generate be combined to form an accurate predictor? These are the questions addressed in this section.

Let $\mathcal{H}$ be a hypothesis of functions mapping $\mathcal{X}$ to $\mathcal{Y}'$, and let $L\colon \mathcal{Y}' \times \mathcal{Y} \to \mathbb{R}_+$ be a bounded loss function, that is $L \leq M$ for some $M \geq 0$. We assume a standard supervised learning setting where a labeled sample $S = ((x_1, y_1), \ldots, (x_T, y_T)) \in (\mathcal{X} \times \mathcal{Y})^T$ is drawn i.i.d. according to some fixed but unknown distribution $\mathcal{D}$. The sample is sequentially processed by an on-line learning algorithm $\mathcal{A}$. The algorithm starts with an initial hypothesis $h_1 \in \mathcal{H}$ and generates a new hypothesis $h_{t+1} \in \mathcal{H}$, after processing pair (x_t, y_t), $t \in [m]$. The regret of the algorithm is defined as before by

$$R_T = \sum_{t=1}^{T} L(h_t(x_t), y_t) - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} L(h(x_t), y_t). \tag{8.25}$$

The generalization error of a hypothesis $h \in \mathcal{H}$ is its expected loss $R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(h(x), y)]$.

The following lemma gives a bound on the average of the generalization errors of the hypotheses generated by $\mathcal{A}$ in terms of its average loss $\frac{1}{T}\sum_{t=1}^{T} L(h_t(x_t), y_t)$.

Lemma 8.14 *Let $S = ((x_1, y_1), \ldots, (x_T, y_T)) \in (\mathcal{X} \times \mathcal{Y})^T$ be a labeled sample drawn i.i.d. according to $\mathcal{D}$, L a loss bounded by M and $h_1, \ldots, h_T$ the sequence of hypotheses generated by an on-line algorithm $\mathcal{A}$ sequentially processing S. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds:*

$$\frac{1}{T}\sum_{t=1}^{T} R(h_t) \leq \frac{1}{T}\sum_{t=1}^{T} L(h_t(x_t), y_t) + M\sqrt{\frac{2\log\frac{1}{\delta}}{T}}. \tag{8.26}$$

Proof: For any $t \in [T]$, let V_t be the random variable defined by $V_t = R(h_t) - L(h_t(x_t), y_t)$. Observe that for any $t \in [T]$,

$$\mathbb{E}[V_t | x_1, \ldots, x_{t-1}] = R(h_t) - \mathbb{E}[L(h_t(x_t), y_t) | h_t] = R(h_t) - R(h_t) = 0.$$

Since the loss is bounded by M, V_t takes values in the interval $[-M, +M]$ for all $t \in [T]$. Thus, by Azuma's inequality (theorem D.7), $\mathbb{P}[\frac{1}{T}\sum_{t=1}^{T} V_t \geq \epsilon] \leq \exp(-2T\epsilon^2/(2M)^2))$. Setting the right-hand side to be equal to $\delta > 0$ yields the statement of the lemma. □

When the loss function is convex with respect to its first argument, the lemma can be used to derive a bound on the generalization error of the average of the hypotheses generated by $\mathcal{A}$, $\frac{1}{T}\sum_{t=1}^{T} h_t$, in terms of the average loss of $\mathcal{A}$ on S, or in terms of the regret R_T and the infimum error of hypotheses in $\mathcal{H}$.

Theorem 8.15 *Let $S = ((x_1, y_1), \ldots, (x_T, y_T)) \in (\mathcal{X} \times \mathcal{Y})^T$ be a labeled sample drawn i.i.d. according to $\mathcal{D}$, L a loss bounded by M and convex with respect to its first argument, and $h_1, \ldots, h_T$ the sequence of hypotheses generated by an on-line algorithm $\mathcal{A}$ sequentially processing S. Then, for any $\delta > 0$, with probability at least*

$1-\delta$, *each of the following holds:*

$$R\Big(\frac{1}{T}\sum_{t=1}^{T}h_t\Big) \le \frac{1}{T}\sum_{t=1}^{T}L(h_t(x_t),y_t) + M\sqrt{\frac{2\log\frac{1}{\delta}}{T}} \tag{8.27}$$

$$R\Big(\frac{1}{T}\sum_{t=1}^{T}h_t\Big) \le \inf_{h\in\mathcal{H}}R(h) + \frac{R_T}{T} + 2M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}. \tag{8.28}$$

Proof: By the convexity of L with respect to its first argument, for any $(x,y)\in\mathcal{X}\times\mathcal{Y}$, we have $L(\frac{1}{T}\sum_{t=1}^{T}h_t(x),y)\le\frac{1}{T}\sum_{t=1}^{T}L(h_t(x),y)$. Taking the expectation gives $R(\frac{1}{T}\sum_{t=1}^{T}h_t)\le\frac{1}{T}\sum_{t=1}^{T}R(h_t)$. The first inequality then follows by lemma 8.14. Thus, by definition of the regret R_T, for any $\delta>0$, the following holds with probability at least $1-\delta/2$:

$$\begin{aligned}R\Big(\frac{1}{T}\sum_{t=1}^{T}h_t\Big) &\le \frac{1}{T}\sum_{t=1}^{T}L(h_t(x_t),y_t) + M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}\\ &\le \min_{h\in\mathcal{H}}\frac{1}{T}\sum_{t=1}^{T}L(h(x_t),y_t) + \frac{R_T}{T} + M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}.\end{aligned}$$

By definition of $\inf_{h\in\mathcal{H}}R(h)$, for any $\epsilon>0$, there exists $h^*\in\mathcal{H}$ with $R(h^*)\le\inf_{h\in\mathcal{H}}R(h)+\epsilon$. By Hoeffding's inequality, for any $\delta>0$, with probability at least $1-\delta/2$, $\frac{1}{T}\sum_{t=1}^{T}L(h^*(x_t),y_t)\le R(h^*)+M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}$. Thus, for any $\epsilon>0$, by the union bound, the following holds with probability at least $1-\delta$:

$$\begin{aligned}R\Big(\frac{1}{T}\sum_{t=1}^{T}h_t\Big) &\le \frac{1}{T}\sum_{t=1}^{T}L(h^*(x_t),y_t) + \frac{R_T}{T} + M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}\\ &\le R(h^*) + M\sqrt{\frac{2\log\frac{2}{\delta}}{T}} + \frac{R_T}{T} + M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}\\ &= R(h^*) + \frac{R_T}{T} + 2M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}\\ &\le \inf_{h\in\mathcal{H}}R(h) + \epsilon + \frac{R_T}{T} + 2M\sqrt{\frac{2\log\frac{2}{\delta}}{T}}.\end{aligned}$$

Since this inequality holds for all $\epsilon>0$, it implies the second statement of the theorem. □

The theorem can be applied to a variety of on-line regret minimization algorithms, for example when $R_T/T=O(1/\sqrt{T})$. In particular, we can apply the theorem to the exponential weighted average algorithm. Assuming that the loss L is bounded by $M = 1$ and that the number of rounds T is known to the algorithm, we can use the regret bound of theorem 8.6. The doubling trick (used in theorem 8.7) can be used to derive a similar bound if T is not known in advance. Thus, for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for the generalization error of the average of the hypotheses generated by exponential weighted average:

$$R\bigg(\frac{1}{T}\sum_{t=1}^{T} h_t\bigg) \leq \inf_{h \in \mathcal{H}} R(h) + \sqrt{\frac{\log N}{2T}} + 2\sqrt{\frac{2\log\frac{2}{\delta}}{T}},$$

where N is the number of experts, or the dimension of the weight vectors.

8.5 Game-theoretic connection

The existence of regret minimization algorithms can be used to give a simple proof of von Neumann's theorem. For any $m \geq 1$, we will denote by Δ_m the set of all distributions over $\{1, \ldots, m\}$, that is $\Delta_m = \{\mathbf{p} \in \mathbb{R}^m \colon \mathbf{p} \geq 0 \wedge \|\mathbf{p}\|_1 = 1\}$.

Theorem 8.16 (Von Neumann's minimax theorem) *Let $m, n \geq 1$. Then, for any two-person zero-sum game defined by matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$,*

$$\min_{\mathbf{p} \in \Delta_m} \max_{\mathbf{q} \in \Delta_n} \mathbf{p}^\top \mathbf{M} \mathbf{q} = \max_{\mathbf{q} \in \Delta_n} \min_{\mathbf{p} \in \Delta_m} \mathbf{p}^\top \mathbf{M} \mathbf{q}\,. \tag{8.29}$$

Proof: The inequality $\max_{\mathbf{q}} \min_{\mathbf{p}} \mathbf{p}^\top \mathbf{M} \mathbf{q} \leq \min_{\mathbf{p}} \max_{\mathbf{q}} \mathbf{p}^\top \mathbf{M} \mathbf{q}$ is straightforward, since by definition of min, for all $\mathbf{p} \in \Delta_m, \mathbf{q} \in \Delta_n$, we have $\min_{\mathbf{p}} \mathbf{p}^\top \mathbf{M} \mathbf{q} \leq \mathbf{p}^\top \mathbf{M} \mathbf{q}$. Taking the maximum over $\mathbf{q}$ of both sides gives: $\max_{\mathbf{q}} \min_{\mathbf{p}} \mathbf{p}^\top \mathbf{M} \mathbf{q} \leq \max_{\mathbf{q}} \mathbf{p}^\top \mathbf{M} \mathbf{q}$ for all $\mathbf{p}$, subsequently taking the minimum over $\mathbf{p}$ proves the inequality.[12]

To show the reverse inequality, consider an on-line learning setting where at each round $t \in [T]$, algorithm $\mathcal{A}$ returns $\mathbf{p}_t$ and incurs loss $\mathbf{M}\mathbf{q}_t$. We can assume that $\mathbf{q}_t$ is selected in the optimal adversarial way, that is $\mathbf{q}_t \in \operatorname{argmax}_{q \in \Delta_m} \mathbf{p}_t^\top \mathbf{M} \mathbf{q}$, and that $\mathcal{A}$ is a regret minimization algorithm, that is $R_T/T \to 0$, where $R_T = \sum_{t=1}^T \mathbf{p}_t^\top \mathbf{M} \mathbf{q}_t - \min_{\mathbf{p} \in \Delta_m} \sum_{t=1}^T \mathbf{p}^\top \mathbf{M} \mathbf{q}_t$. Then, the following holds:

$$\min_{\mathbf{p} \in \Delta_m} \max_{\mathbf{q} \in \Delta_n} \mathbf{p}^\top \mathbf{M} \mathbf{q} \leq \max_{\mathbf{q}} \Big(\frac{1}{T}\sum_{t=1}^{T} \mathbf{p}_t\Big)^{\!\top} \mathbf{M} \mathbf{q} \leq \frac{1}{T}\sum_{t=1}^{T} \max_{\mathbf{q}} \mathbf{p}_t^\top \mathbf{M} \mathbf{q} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{p}_t^\top \mathbf{M} \mathbf{q}_t.$$

[12] More generally, the maxmin is always upper bounded by the minmax for any function or two arguments and any constraint sets, following the same proof.

By definition of regret, the right-hand side can be expressed and bounded as follows:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbf{p}_t^\top \mathbf{M}\mathbf{q}_t = \min_{\mathbf{p}\in\Delta_m}\frac{1}{T}\sum_{t=1}^{T}\mathbf{p}^\top \mathbf{M}\mathbf{q}_t + \frac{R_T}{T} = \min_{\mathbf{p}\in\Delta_m}\mathbf{p}^\top\mathbf{M}\Big(\frac{1}{T}\sum_{t=1}^{T}\mathbf{q}_t\Big) + \frac{R_T}{T}$$
$$\leq \max_{\mathbf{q}\in\Delta_n}\min_{\mathbf{p}\in\Delta_m}\mathbf{p}^\top\mathbf{M}\mathbf{q} + \frac{R_T}{T}.$$

This implies that the following bound holds for the minmax for all $T \geq 1$:

$$\min_{\mathbf{p}\in\Delta_m}\max_{\mathbf{q}\in\Delta_n}\mathbf{p}^\top\mathbf{M}\mathbf{q} \leq \max_{\mathbf{q}\in\Delta_n}\min_{\mathbf{p}\in\Delta_m}\mathbf{p}^\top\mathbf{M}\mathbf{q} + \frac{R_T}{T}$$

Since $\lim_{T\to+\infty}\frac{R_T}{T} = 0$, this shows that $\min_{\mathbf{p}}\max_{\mathbf{q}}\mathbf{p}^\top\mathbf{M}\mathbf{q} \leq \max_{\mathbf{q}}\min_{\mathbf{p}}\mathbf{p}^\top\mathbf{M}\mathbf{q}$.$\square$

8.6 Chapter notes

Algorithms for regret minimization were initiated with the pioneering work of Hannan [1957] who gave an algorithm whose regret decreases as $O(\sqrt{T})$ as a function of T but whose dependency on N is linear. The weighted majority algorithm and the randomized weighted majority algorithm, whose regret is only logarithmic in N, are due to Littlestone and Warmuth [1989]. The exponential weighted average algorithm and its analysis, which can be viewed as an extension of the WM algorithm to convex non-zero-one losses is due to the same authors [Littlestone and Warmuth, 1989, 1994]. The analysis we presented follows Cesa-Bianchi [1999] and Cesa-Bianchi and Lugosi [2006]. The doubling trick technique appears in Vovk [1990] and Cesa-Bianchi et al. [1997]. The algorithm of exercise 8.7 and the analysis leading to a second-order bound on the regret are due to Cesa-Bianchi et al. [2005]. The lower bound presented in theorem 8.5 is from Blum and Mansour [2007].

While the regret bounds presented are logarithmic in the number of the experts N, when N is exponential in the size of the input problem, the computational complexity of an expert algorithm could be exponential. For example, in the online shortest paths problem, N is the number of paths between two vertices of a directed graph. However, several computationally efficient algorithms have been presented for broad classes of such problems by exploiting their structure [Takimoto and Warmuth, 2002, Kalai and Vempala, 2003, Zinkevich, 2003].

The notion of regret (or *external regret*) presented in this chapter can be generalized to that of *internal regret* or even *swap regret*, by comparing the loss of the algorithm not just to that of the best expert in retrospect, but to that of any modification of the actions taken by the algorithm by replacing each occurrence of some specific action with another one (internal regret), or even replacing actions via an arbitrary mapping (swap regret) [Foster and Vohra, 1997, Hart and Mas-Colell, 2000, Lehrer, 2003]. Several algorithms for low internal regret have been given

[Foster and Vohra, 1997, 1998, 1999, Hart and Mas-Colell, 2000, Cesa-Bianchi and Lugosi, 2001, Stoltz and Lugosi, 2003], including a conversion of low external regret to low swap regret by Blum and Mansour [2005].

The Perceptron algorithm was introduced by Rosenblatt [1958]. The algorithm raised a number of reactions, in particular by Minsky and Papert [1969], who objected that the algorithm could not be used to recognize the XOR function. Of course, the kernel Perceptron algorithm already given by Aizerman et al. [1964] could straightforwardly succeed to do so using second-degree polynomial kernels. The margin bound for the Perceptron algorithm was proven by Novikoff [1962] and is one of the first results in learning theory. We presented two extensions of Novikoff's result which hold in the more general non-separable case: Theorem 8.12 due to Freund and Schapire [1999a] and Theorem 8.11 due to Mohri and Rostamizadeh [2013]. Our proof of Theorem 8.12 is significantly more concise that the original proof given by Freund and Schapire [1999a] and shows that the bound of Theorem 8.11 is always tighter than that of Theorem 8.12. See [Mohri and Rostamizadeh, 2013] for other more general data-dependent upper bounds on the number of updates made by the Perceptron algorithm in the non-separable case. The leave-one-out analysis for SVMs is described by Vapnik [1998]. The Winnow algorithm was introduced by Littlestone [1987].

The analysis of the on-line to batch conversion and exercises 8.10 and 8.11 are from Cesa-Bianchi et al. [2001, 2004] (see also Littlestone [1989]). Von Neumann's minimax theorem admits a number of different generalizations. See Sion [1958] for a generalization to quasi-concave-convex functions semi-continuous in each argument and the references therein. The simple proof of von Neumann's theorem presented here is entirely based on learning-related techniques. A proof of a more general version using multiplicative updates was presented by Freund and Schapire [1999b].

On-line learning is a very broad and fast-growing research area in machine learning. The material presented in this chapter should be viewed only as an introduction to the topic, but the proofs and techniques presented should indicate the flavor of most results in this area. For a more comprehensive presentation of on-line learning and related game theory algorithms and techniques, the reader could consult the book of Cesa-Bianchi and Lugosi [2006].

8.7 Exercises

8.1 Perceptron lower bound. Let S be a labeled sample of m points in $\mathbb{R}^N$ with

$$x_i = (\underbrace{(-1)^i, \ldots, (-1)^i, (-1)^{i+1}}_{i \text{ first components}}, 0, \ldots, 0) \quad \text{and} \quad y_i = (-1)^{i+1}. \tag{8.30}$$

```
On-line-SVM($\mathbf{w}_0$)
1  $\mathbf{w}_1 \leftarrow \mathbf{w}_0$    ▷ typically $\mathbf{w}_0 = \mathbf{0}$
2  for $t \leftarrow 1$ to $T$ do
3      Receive($\mathbf{x}_t, y_t$)
4      if $y_t(\mathbf{w}_t \cdot \mathbf{x}_t) < 1$ then
5          $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta(\mathbf{w}_t - C y_t \mathbf{x}_t)$
6      elseif $y_t(\mathbf{w}_t \cdot \mathbf{x}_t) > 1$ then
7          $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \mathbf{w}_t$
8      else $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
9  return $\mathbf{w}_{T+1}$
```

Figure 8.11
On-line SVM algorithm.

Show that the Perceptron algorithm makes $\Omega(2^N)$ updates before finding a separating hyperplane, regardless of the order in which it receives the points.

8.2 Generalized mistake bound. Theorem 8.8 presents a margin bound on the maximum number of updates for the Perceptron algorithm for the special case $\eta = 1$. Consider now the general Perceptron update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta y_t \mathbf{x}_t$, where $\eta > 0$. Prove a bound on the maximum number of mistakes. How does η affect the bound?

8.3 Sparse instances. Suppose each input vector $\mathbf{x}_t$, $t \in [T]$, coincides with the tth unit vector of $\mathbb{R}^T$. How many updates are required for the Perceptron algorithm to converge? Show that the number of updates matches the margin bound of theorem 8.8.

8.4 Tightness of lower bound. Is the lower bound of theorem 8.5 tight? Explain why or show a counter-example.

8.5 On-line SVM algorithm. Consider the algorithm described in figure 8.11. Show that this algorithm corresponds to the stochastic gradient descent technique applied to the SVM problem (5.24) with hinge loss and no offset (i.e., fix $p = 1$ and $b = 0$).

```
MarginPerceptron()
1  $\mathbf{w}_1 \leftarrow \mathbf{0}$
2  for $t \leftarrow 1$ to $T$ do
3      Receive($\mathbf{x}_t$)
4      Receive($y_t$)
5      if $\big((\mathbf{w}_t = 0)$ or $(\frac{y_t \mathbf{w}_t \cdot \mathbf{x}_t}{\|\mathbf{w}_t\|} < \frac{\rho}{2})\big)$ then
6          $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t \mathbf{x}_t$
7      else $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
8  return $\mathbf{w}_{T+1}$
```

Figure 8.12
Margin Perceptron algorithm.

8.6 Margin Perceptron. Given a training sample S that is linearly separable with a maximum margin $\rho > 0$, theorem 8.8 states that the Perceptron algorithm run cyclically over S is guaranteed to converge after at most R^2/ρ^2 updates, where R is the radius of the sphere containing the sample points. However, this theorem does not guarantee that the hyperplane solution of the Perceptron algorithm achieves a margin close to ρ. Suppose we modify the Perceptron algorithm to ensure that the margin of the hyperplane solution is at least $\rho/2$. In particular, consider the algorithm described in figure 8.12. In this problem we show that this algorithm converges after at most $16R^2/\rho^2$ updates. Let $\mathcal{I}$ denote the set of times $t \in [T]$ at which the algorithm makes an update and let $M = |\mathcal{I}|$ be the total number of updates.

(a) Using an analysis similar to the one given for the Perceptron algorithm, show that $M\rho \leq \|\mathbf{w}_{T+1}\|$. Conclude that if $\|\mathbf{w}_{T+1}\| < \frac{4R^2}{\rho}$, then $M < 4R^2/\rho^2$. (For the remainder of this problem, we will assume that $\|\mathbf{w}_{T+1}\| \geq \frac{4R^2}{\rho}$.)

(b) Show that for any $t \in \mathcal{I}$ (including $t = 0$), the following holds:

$$\|\mathbf{w}_{t+1}\|^2 \leq (\|\mathbf{w}_t\| + \rho/2)^2 + R^2.$$

(c) From (b), infer that for any $t \in \mathcal{I}$ we have

$$\|\mathbf{w}_{t+1}\| \leq \|\mathbf{w}_t\| + \rho/2 + \frac{R^2}{\|\mathbf{w}_t\| + \|\mathbf{w}_{t+1}\| + \rho/2}.$$

(d) Using the inequality from (c), show that for any $t \in \mathcal{I}$ such that either $\|\mathbf{w}_t\| \geq \frac{4R^2}{\rho}$ or $\|\mathbf{w}_{t+1}\| \geq \frac{4R^2}{\rho}$, we have

$$\|\mathbf{w}_{t+1}\| \leq \|\mathbf{w}_t\| + \frac{3}{4}\rho.$$

(e) Show that $\|\mathbf{w}_1\| \leq R \leq 4R^2/\rho$. Since by assumption we have $\|\mathbf{w}_{T+1}\| \geq \frac{4R^2}{\rho}$, conclude that there must exist a largest time $t_0 \in \mathcal{I}$ such that $\|\mathbf{w}_{t_0}\| \leq \frac{4R^2}{\rho}$ and $\|\mathbf{w}_{t_0+1}\| \geq \frac{4R^2}{\rho}$.

(f) Show that $\|\mathbf{w}_{T+1}\| \leq \|\mathbf{w}_{t_0}\| + \frac{3}{4}M\rho$. Conclude that $M \leq 16R^2/\rho^2$.

8.7 Second-order regret bound. Consider the randomized algorithm that differs from the RWM algorithm only by the weight update, i.e., $w_{t+1,i} \leftarrow (1-(1-\beta)l_{t,i})w_{t,i}$, $t \in [T]$, which is applied to all $i \in [N]$ with $1/2 \leq \beta < 1$. This algorithm can be used in a more general setting than RWM since the losses $l_{t,i}$ are only assumed to be in $[0,1]$. The objective of this problem is to show that a similar upper bound can be shown for the regret.

(a) Use the same potential W_t as for the RWM algorithm and derive a simple upper bound for $\log W_{T+1}$:

$$\log W_{T+1} \leq \log N - (1-\beta)\mathcal{L}_T\,.$$

(*Hint*: Use the inequality $\log(1-x) \leq -x$ for $x \in [0, 1/2]$.)

(b) Prove the following lower bound for the potential for all $i \in [N]$:

$$\log W_{T+1} \geq -(1-\beta)\mathcal{L}_{T,i} - (1-\beta)^2 \sum_{t=1}^{T} l_{t,i}^2\,.$$

(*Hint*: Use the inequality $\log(1-x) \geq -x - x^2$, which is valid for all $x \in [0, 1/2]$.)

(c) Use upper and lower bounds to derive the following regret bound for the algorithm: $R_T \leq 2\sqrt{T \log N}$.

8.8 Polynomial weighted algorithm. The objective of this problem is to show how another regret minimization algorithm can be defined and studied. Let L be a loss function convex in its first argument and taking values in $[0, M]$.

We will assume $N > e^2$ and then for any expert $i \in [N]$, we denote by $r_{t,i}$ the instantaneous regret of that expert at time $t \in [T]$, $r_{t,i} = L(\widehat{y}_t, y_t) - L(y_{t,i}, y_t)$, and by $R_{t,i}$ its cumulative regret up to time t: $R_{t,i} = \sum_{s=1}^{t} r_{t,i}$. For convenience, we also define $R_{0,i} = 0$ for all $i \in [N]$. For any $x \in \mathbb{R}$, $(x)_+$ denotes $\max(x, 0)$, that is the positive part of x, and for $\mathbf{x} = (x_1, \ldots, x_N)^\top \in \mathbb{R}^N$, $(\mathbf{x})_+ = ((x_1)_+, \ldots, (x_N)_+)^\top$.

Let $\alpha > 2$ and consider the algorithm that predicts at round $t \in [T]$ according to $\widehat{y}_t = \frac{\sum_{i=1}^{n} w_{t,i} y_{t,i}}{\sum_{i=1}^{n} w_{t,i}}$, with the weight $w_{t,i}$ defined based on the αth power of the regret up to time $(t-1)$: $w_{t,i} = (R_{t-1,i})_+^{\alpha-1}$. The potential function we use to analyze the algorithm is based on the function Φ defined over $\mathbb{R}^N$ by $\Phi \colon \mathbf{x} \mapsto \|(\mathbf{x})_+\|_\alpha^2 = \big[\sum_{i=1}^{N} (x_i)_+^\alpha\big]^{\frac{2}{\alpha}}$.

(a) Show that Φ is twice differentiable over $\mathbb{R}^N - B$, where B is defined as follows:

$$B = \{\mathbf{u} \in \mathbb{R}^N \colon (\mathbf{u})_+ = 0\}.$$

(b) For any $t \in [T]$, let $\mathbf{r}_t$ denote the vector of instantaneous regrets, $\mathbf{r}_t = (r_{t,1}, \ldots, r_{t,N})^\top$, and similarly $\mathbf{R}_t = (R_{t,1}, \ldots, R_{t,N})^\top$. We define the potential function as $\Phi(\mathbf{R}_t) = \|(\mathbf{R}_t)_+\|_\alpha^2$. Compute $\nabla\Phi(\mathbf{R}_{t-1})$ for $\mathbf{R}_{t-1} \notin B$ and show that $\nabla\Phi(\mathbf{R}_{t-1}) \cdot \mathbf{r}_t \leq 0$ (*Hint*: use the convexity of the loss with respect to the first argument).

(c) Prove the inequality $\mathbf{r}^\top[\nabla^2\Phi(\mathbf{u})]\mathbf{r} \leq 2(\alpha-1)\|\mathbf{r}\|_\alpha^2$ valid for all $\mathbf{r} \in \mathbb{R}^N$ and $\mathbf{u} \in \mathbb{R}^N - B$ (*Hint*: write the Hessian $\nabla^2\Phi(\mathbf{u})$ as a sum of a diagonal matrix and a positive semidefinite matrix multiplied by $(2-\alpha)$. Also, use Hölder's inequality generalizing Cauchy-Schwarz: for any $p > 1$ and $q > 1$ with $\frac{1}{p} + \frac{1}{q} = 1$ and $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N$, $|\mathbf{u} \cdot \mathbf{v}| \leq \|\mathbf{u}\|_p \|\mathbf{v}\|_q$).

(d) Using the answers to the two previous questions and Taylor's formula, show that for all $t \geq 1$, $\Phi(\mathbf{R}_t) - \Phi(\mathbf{R}_{t-1}) \leq (\alpha-1)\|\mathbf{r}_t\|_\alpha^2$, if $\gamma\mathbf{R}_{t-1} + (1-\gamma)\mathbf{R}_t \notin B$ for all $\gamma \in [0, 1]$.

(e) Suppose there exists $\gamma \in [0, 1]$ such that $(1-\gamma)\mathbf{R}_{t-1} + \gamma\mathbf{R}_t \in B$. Show that $\Phi(\mathbf{R}_t) \leq (\alpha-1)\|\mathbf{r}_t\|_\alpha^2$.

(f) Using the two previous questions, derive an upper bound on $\Phi(\mathbf{R}_T)$ expressed in terms of T, N, and M.

(g) Show that $\Phi(\mathbf{R}_T)$ admits as a lower bound the square of the regret R_T of the algorithm.

(h) Using the two previous questions give an upper bound on the regret R_T. For what value of α is the bound the most favorable? Give a simple expression of the upper bound on the regret for a suitable approximation of that optimal value.

8.9 General inequality. In this exercise we generalize the result of exercise 8.7 by using a more general inequality: $\log(1-x) \geq -x - \frac{x^2}{\alpha}$ for some $0 < \alpha < 2$.

(a) First prove that the inequality is true for $x \in [0, 1 - \frac{\alpha}{2}]$. What does this imply about the valid range of β?

(b) Give a generalized version of the regret bound derived in exercise 8.7 in terms of α, which shows:

$$R_T \leq \frac{\log N}{1-\beta} + \frac{1-\beta}{\alpha} T .$$

What is the optimal choice of β and the resulting bound in this case?

(c) Explain how α may act as a regularization parameter. What is the optimal choice of α?

8.10 On-line to batch — non-convex loss.

The on-line to batch result of theorem 8.15 heavily relies on the fact that the loss is convex in order to provide a generalization guarantee for the uniformly averaged hypothesis $\frac{1}{T}\sum_{i=1}^{T} h_i$. For general losses, instead of using the averaged hypothesis we will use a different strategy and try to estimate the best single base hypothesis and show the expected loss of this hypothesis is bounded.

Let m_i denote the cumulative loss of hypothesis h_i on the points $(x_i, \ldots, x_T)$, that is $m_i = \sum_{t=i}^{T} L(h_i(x_t), y_t)$. Then we define the *penalized risk estimate* of hypothesis h_i as,

$$\frac{m_i}{T-i+1} + c_\delta(T-i+1) \quad \text{where} \quad c_\delta(x) = \sqrt{\frac{1}{2x}\log\frac{T(T+1)}{\delta}} .$$

The term c_δ penalizes the empirical error when the test sample is small. Define $\widehat{h} = h_{i^*}$ where $i^* = \operatorname{argmin}_i m_i/(T-i+1) + c_\delta(T-i+1)$. We will then show under the same conditions of theorem 8.15 (with $M = 1$ for simplicity), but without requiring the convexity of L, that the following holds with probability at least $1-\delta$:

$$R(\widehat{h}) \leq \frac{1}{T}\sum_{i=1}^{T} L(h_i(x_i), y_i) + 6\sqrt{\frac{1}{T}\log\frac{2(T+1)}{\delta}} . \tag{8.31}$$

(a) Prove the following inequality:

$$\min_{i\in[T]}\big(R(h_i) + 2c_\delta(T-i+1)\big) \leq \frac{1}{T}\sum_{i=1}^{T} R(h_i) + 4\sqrt{\frac{1}{T}\log\frac{T+1}{\delta}} .$$

(b) Use part (a) to show that with probability at least $1-\delta$,

$$\min_{i\in[T]}(R(h_i)+2c_\delta(T-i+1)) < \sum_{i=1}^{T} L(h_i(x_i),y_i)+\sqrt{\frac{2}{T}\log\frac{1}{\delta}}+4\sqrt{\frac{1}{T}\log\frac{T+1}{\delta}}\,.$$

(c) By design, the definition of c_δ ensures that with probability at least $1-\delta$

$$R(\widehat{h}) \leq \min_{i\in[T]}(R(h_i)+2c_\delta(T-i+1))\,.$$

Use this property to complete the proof of (8.31).

8.11 On-line to batch — kernel Perceptron margin bound. In this problem, we give a margin-based generalization guarantee for the kernel Perceptron algorithm. Let $h_1,\ldots,h_T$ be the sequence of hypotheses generated by the kernel Perceptron algorithm and let $\widehat{h}$ be defined as in exercise 8.10. Finally, let L denote the zero-one loss. We now wish to more precisely bound the generalization error of $\widehat{h}$ in this setting.

(a) First, show that

$$\sum_{i=1}^{T} L(h_i(x_i),y_i) \leq \inf_{h\in\mathbb{H}:\|h\|\leq 1}\sum_{i=1}^{T}\max\left(0,1-\frac{y_i h(x_i)}{\rho}\right)+\frac{1}{\rho}\sqrt{\sum_{i\in I}K(x_i,x_i)},$$

where I is the set of indices where the kernel Perceptron makes an update and where δ and ρ are defined as in theorem 8.12.

(b) Now, use the result of exercise 8.10 to derive a generalization guarantee for $\widehat{h}$ in the case of kernel Perceptron, which states that for any $0<\delta\leq 1$, the following holds with probability at least $1-\delta$:

$$R(\widehat{h}) \leq \inf_{h\in\mathbb{H}:\|h\|\leq 1}\widehat{R}_{S,\rho}(h)+\frac{1}{\rho T}\sqrt{\sum_{i\in I}K(x_i,x_i)}+6\sqrt{\frac{1}{T}\log\frac{2(T+1)}{\delta}},$$

where $\widehat{R}_{S,\rho}(h)=\frac{1}{T}\sum_{i=1}^{T}\max\left(0,1-\frac{y_i h(x_i)}{\rho}\right)$. Compare this result with the margin bounds for kernel-based hypotheses given by corollary 6.13.

9 Multi-Class Classification

The classification problems we examined in the previous chapters were all binary. However, in most real-world classification problems the number of classes is greater than two. The problem may consist of assigning a topic to a text document, a category to a speech utterance or a function to a biological sequence. In all of these tasks, the number of classes may be on the order of several hundred or more.

In this chapter, we analyze the problem of multi-class classification. We first introduce the multi-class classification learning problem and discuss its multiple settings, and then derive generalization bounds for it using the notion of Rademacher complexity. Next, we describe and analyze a series of algorithms for tackling the multi-class classification problem. We will distinguish between two broad classes of algorithms: *uncombined algorithms* that are specifically designed for the multi-class setting such as multi-class SVMs, decision trees, or multi-class boosting, and *aggregated algorithms* that are based on a reduction to binary classification and require training multiple binary classifiers. We will also briefly discuss the problem of structured prediction, which is a related problem arising in a variety of applications.

9.1 Multi-class classification problem

Let $\mathcal{X}$ denote the input space and $\mathcal{Y}$ denote the output space, and let $\mathcal{D}$ be an unknown distribution over $\mathcal{X}$ according to which input points are drawn. We will distinguish between two cases: the *mono-label case*, where $\mathcal{Y}$ is a finite set of classes that we mark with numbers for convenience, $\mathcal{Y} = \{1, \ldots, k\}$, and the *multi-label case* where $\mathcal{Y} = \{-1, +1\}^k$. In the mono-label case, each example is labeled with a single class, while in the multi-label case it can be labeled with several. The latter can be illustrated by the case of text documents, which can be labeled with several different relevant topics, e.g., *sports*, *business*, and *society*. The positive components of a vector in $\{-1, +1\}^k$ indicate the classes associated with an example.

In either case, the learner receives a labeled sample $S = \big((x_1, y_1), \ldots, (x_m, y_m)\big) \in (\mathcal{X} \times \mathcal{Y})^m$ with $x_1, \ldots, x_m$ drawn i.i.d. according to $\mathcal{D}$, and $y_i = f(x_i)$ for all $i \in [m]$, where $f \colon \mathcal{X} \to \mathcal{Y}$ is the target labeling function. Thus, we consider a deterministic scenario, which, as discussed in section 2.4.1, can be straightforwardly extended to a stochastic one that admits a distribution over $\mathcal{X} \times \mathcal{Y}$.

Given a hypothesis set $\mathcal{H}$ of functions mapping $\mathcal{X}$ to $\mathcal{Y}$, the multi-class classification problem consists of using the labeled sample S to find a hypothesis $h \in \mathcal{H}$ with small generalization error $R(h)$ with respect to the target f:

$$R(h) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}}[1_{h(x) \neq f(x)}] \qquad \text{mono-label case} \tag{9.1}$$

$$R(h) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}}\Big[\sum_{l=1}^{k} 1_{[h(x)]_l \neq [f(x)]_l}\Big] \qquad \text{multi-label case.} \tag{9.2}$$

The notion of *Hamming distance* d_H, that is, the number of corresponding components in two vectors that differ, can be used to give a common formulation for both errors:

$$R(h) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}}\Big[d_H(h(x), f(x))\Big]. \tag{9.3}$$

The empirical error of $h \in \mathcal{H}$ is denoted by $\widehat{R}_S(h)$ and defined by

$$\widehat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m} d_H(h(x_i), y_i)\,. \tag{9.4}$$

Several issues, both computational and learning-related, often arise in the multi-class setting. Computationally, dealing with a large number of classes can be problematic. The number of classes k directly enters the time complexity of the algorithms we will present. Even for a relatively small number of classes such as $k = 100$ or $k = 1{,}000$, some techniques may become prohibitive to use in practice. This dependency is even more critical in the case where k is very large or even infinite as in the case of some structured prediction problems.

A learning-related issue that commonly appears in the multi-class setting is the existence of unbalanced classes. Some classes may be represented by less than 5 percent of the labeled sample, while others may dominate a very large fraction of the data. When separate binary classifiers are used to define the multi-class solution, we may need to train a classifier distinguishing between two classes with only a small representation in the training sample. This implies training on a small sample, with poor performance guarantees. Alternatively, when a large fraction of the training instances belong to one class, it may be tempting to propose a hypothesis always returning that class, since its generalization error as defined earlier is likely to be relatively low. However, this trivial solution is typically not the one intended. Instead, the loss function may need to be reformulated by assigning different misclassification weights to each pair of classes.

Another learning-related issue is the relationship between classes, which can be hierarchical. For example, in the case of document classification, the error of misclassifying a document dealing with *world politics* as one dealing with *real estate* should naturally be penalized more than the error of labeling a document with *sports* instead of the more specific label *baseball*. Thus, a more complex and more useful multi-class classification formulation would take into consideration the hierarchical relationships between classes and define the loss function in accordance with this hierarchy. More generally, there may be a graph relationship between classes as in the case of gene ontology in computational biology. The use of hierarchical relationships between classes leads to a richer and more complex multi-class classification problem.

9.2 Generalization bounds

In this section, we present margin-based generalization bounds for multi-class classification in the mono-label case. In the binary setting, classifiers are often defined based on the sign of a scoring function. In the multi-class setting, a hypothesis is defined based on a scoring function $h\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. The label associated to point x is the one resulting in the largest score $h(x, y)$, which defines the following mapping from $\mathcal{X}$ to $\mathcal{Y}$:

$$x \mapsto \operatorname*{argmax}_{y \in \mathcal{Y}} h(x, y).$$

This naturally leads to the following definition of the *margin $\rho_h(x, y)$ of the function h at a labeled example (x, y)*:

$$\rho_h(x, y) = h(x, y) - \max_{y' \neq y} h(x, y').$$

Thus, h misclassifies (x, y) iff $\rho_h(x, y) \leq 0$. For any $\rho > 0$, we can define the *empirical margin loss of a hypothesis h* for multi-class classification as

$$\widehat{R}_{S,\rho}(h) = \frac{1}{m} \sum_{i=1}^{m} \Phi_\rho(\rho_h(x_i, y_i)), \tag{9.5}$$

where Φ_ρ is the margin loss function (definition 5.5). Thus, the empirical margin loss for multi-class classification is upper bounded by the fraction of the training points misclassified by h or correctly classified but with confidence less than or equal to ρ:

$$\widehat{R}_{S,\rho}(h) \leq \frac{1}{m}\sum_{i=1}^{m} 1_{\rho_h(x_i,y_i)\leq\rho}. \tag{9.6}$$

The following lemma will be used in the proof of the main result of this section.

Lemma 9.1 *Let $\mathcal{F}_1, \ldots, \mathcal{F}_l$ be l hypothesis sets in $\mathbb{R}^{\mathcal{X}}$, $l \geq 1$, and let $\mathcal{G} = \{\max\{h_1, \ldots, h_l\} \colon h_i \in \mathcal{F}_i, i \in [l]\}$. Then, for any sample S of size m, the empirical Rademacher complexity of $\mathcal{G}$ can be upper bounded as follows:*

$$\widehat{\mathfrak{R}}_S(\mathcal{G}) \leq \sum_{j=1}^{l} \widehat{\mathfrak{R}}_S(\mathcal{F}_j). \tag{9.7}$$

Proof: Let $S = (x_1, \ldots, x_m)$ be a sample of size m. We first prove the result in the case $l = 2$. By definition of the max operator, for any $h_1 \in \mathcal{F}_1$ and $h_2 \in \mathcal{F}_2$,

$$\max\{h_1, h_2\} = \frac{1}{2}[h_1 + h_2 + |h_1 - h_2|].$$

Thus, we can write:

$$\begin{aligned}
\widehat{\mathfrak{R}}_S(\mathcal{G}) &= \frac{1}{m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{\substack{h_1\in\mathcal{F}_1\\h_2\in\mathcal{F}_2}}\sum_{i=1}^{m}\sigma_i \max\{h_1(x_i), h_2(x_i)\}\Big] \\
&= \frac{1}{2m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{\substack{h_1\in\mathcal{F}_1\\h_2\in\mathcal{F}_2}}\sum_{i=1}^{m}\sigma_i\big(h_1(x_i) + h_2(x_i) + |(h_1 - h_2)(x_i)|\big)\Big] \\
&\leq \frac{1}{2}\widehat{\mathfrak{R}}_S(\mathcal{F}_1) + \frac{1}{2}\widehat{\mathfrak{R}}_S(\mathcal{F}_2) + \frac{1}{2m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{\substack{h_1\in\mathcal{F}_1\\h_2\in\mathcal{F}_2}}\sum_{i=1}^{m}\sigma_i|(h_1 - h_2)(x_i)|\Big], \qquad (9.8)
\end{aligned}$$

using the sub-additivity of sup. Since $x \mapsto |x|$ is 1-Lipschitz, by Talagrand's lemma (lemma 5.7), the last term can be bounded as follows

$$\begin{aligned}
\frac{1}{2m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{\substack{h_1\in\mathcal{F}_1\\h_2\in\mathcal{F}_2}}\sum_{i=1}^{m}\sigma_i|(h_1 - h_2)(x_i)|\Big] &\leq \frac{1}{2m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{\substack{h_1\in\mathcal{F}_1\\h_2\in\mathcal{F}_2}}\sum_{i=1}^{m}\sigma_i(h_1 - h_2)(x_i)\Big] \\
&\leq \frac{1}{2}\widehat{\mathfrak{R}}_S(\mathcal{F}_1) + \frac{1}{2m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sup_{h_2\in\mathcal{F}_2}\sum_{i=1}^{m}-\sigma_i h_2(x_i)\Big] \\
&= \frac{1}{2}\widehat{\mathfrak{R}}_S(\mathcal{F}_1) + \frac{1}{2}\widehat{\mathfrak{R}}_S(\mathcal{F}_2), \qquad (9.9)
\end{aligned}$$

where we again use the sub-additivity of sup for the second inequality and the fact that σ_i and $-\sigma_i$ have the same distribution for any $i \in [m]$ for the last equality. Combining (9.8) and (9.9) yields $\widehat{\mathfrak{R}}_S(\mathcal{G}) \leq \widehat{\mathfrak{R}}_S(\mathcal{F}_1) + \widehat{\mathfrak{R}}_S(\mathcal{F}_2)$. The general case can be derived from the case $l = 2$ using $\max\{h_1, \ldots, h_l\} = \max\{h_1, \max\{h_2, \ldots, h_l\}\}$ and an immediate recurrence. □

For any family of hypotheses mapping $\mathcal{X} \times \mathcal{Y}$ to $\mathbb{R}$, we define $\Pi_1(\mathcal{H})$ by

$$\Pi_1(\mathcal{H}) = \{x \mapsto h(x, y) \colon y \in \mathcal{Y}, h \in \mathcal{H}\}.$$

The following theorem gives a general margin bound for multi-class classification.

Theorem 9.2 (Margin bound for multi-class classification) *Let $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X} \times \mathcal{Y}}$ be a hypothesis set with $\mathcal{Y} = \{1, \ldots, k\}$. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following multi-class classification generalization bound holds for all $h \in \mathcal{H}$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{4k}{\rho}\mathfrak{R}_m(\Pi_1(\mathcal{H})) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}\,. \tag{9.10}$$

Proof: We will need the following definition for this proof:

$$\rho_{\theta,h}(x, y) = \min_{y'}(h(x, y) - h(x, y') + \theta 1_{y'=y}),$$

where $\theta > 0$ is an arbitrary constant. Observe that $\mathbb{E}[1_{\rho_h(x,y) \leq 0}] \leq \mathbb{E}[1_{\rho_{\theta,h}(x,y) \leq 0}]$ since the inequality $\rho_{\theta,h}(x, y) \leq \rho_h(x, y)$ holds for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$:

$$\begin{aligned}\rho_{\theta,h}(x, y) &= \min_{y'} \big(h(x, y) - h(x, y') + \theta 1_{y'=y}\big)\\ &\leq \min_{y' \neq y} \big(h(x, y) - h(x, y') + \theta 1_{y'=y}\big)\\ &= \min_{y' \neq y} \big(h(x, y) - h(x, y')\big) = \rho_h(x, y),\end{aligned}$$

where the inequality follows from taking the minimum over a smaller set.

Now, similar to the proof of theorem 5.8, let $\widetilde{\mathcal{H}} = \{(x, y) \mapsto \rho_{\theta,h}(x, y) \colon h \in \mathcal{H}\}$ and $\widetilde{\mathcal{H}} = \{\Phi_\rho \circ \widetilde{h} \colon \widetilde{h} \in \widetilde{\mathcal{H}}\}$. By theorem 3.3, with probability at least $1 - \delta$, for all $h \in \mathcal{H}$,

$$\mathbb{E}\left[\Phi_\rho(\rho_{\theta,h}(x, y))\right] \leq \frac{1}{m}\sum_{i=1}^{m} \Phi_\rho(\rho_{\theta,h}(x_i, y_i)) + 2\mathfrak{R}_m(\widetilde{\mathcal{H}}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$

Since $1_{u \leq 0} \leq \Phi_\rho(u)$ for all $u \in \mathbb{R}$, the generalization error $R(h)$ is a lower bound on the left-hand side, $R(h) = \mathbb{E}[1_{\rho_h(x,y) \leq 0}] \leq \mathbb{E}[1_{\rho_{\theta,h}(x,y) \leq 0}] \leq \mathbb{E}\left[\Phi_\rho(\rho_{\theta,h}(x, y))\right]$, and we can write:

$$R(h) \leq \frac{1}{m}\sum_{i=1}^{m} \Phi_\rho(\rho_{\theta,h}(x_i, y_i)) + 2\mathfrak{R}_m(\widetilde{\mathcal{H}}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$

Fixing $\theta = 2\rho$, we observe that $\Phi_\rho(\rho_{\theta,h}(x_i, y_i)) = \Phi_\rho(\rho_h(x_i, y_i))$. Indeed, either $\rho_{\theta,h}(x_i, y_i) = \rho_h(x_i, y_i)$ or $\rho_{\theta,h}(x_i, y_i) = 2\rho \leq \rho_h(x_i, y_i)$, which implies the desired result. Furthermore, Talagrand's lemma (lemma 5.7) yields $\mathfrak{R}_m(\widetilde{\mathcal{H}}) \leq \frac{1}{\rho}\mathfrak{R}_m(\widetilde{\mathcal{H}})$ since Φ_ρ is a $\frac{1}{\rho}$-Lipschitz function. Therefore, for any $\delta > 0$, with probability at least $1-\delta$, for all $h \in \mathcal{H}$:

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\mathfrak{R}_m(\widetilde{\mathcal{H}}) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}.$$

and to complete the proof it suffices to show that $\mathfrak{R}_m(\widetilde{\mathcal{H}}) \leq 2k\mathfrak{R}_m(\Pi_1(\mathcal{H}))$.

Here $\mathfrak{R}_m(\widetilde{\mathcal{H}})$ can be upper-bounded as follows:

$$\begin{aligned}\mathfrak{R}_m(\widetilde{\mathcal{H}}) &= \frac{1}{m}\mathop{\mathbb{E}}_{S,\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i(h(x_i,y_i) - \max_y(h(x_i,y) - 2\rho 1_{y=y_i}))\right]\\ &\leq \frac{1}{m}\mathop{\mathbb{E}}_{S,\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i h(x_i,y_i)\right] + \frac{1}{m}\mathop{\mathbb{E}}_{S,\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i \max_y(h(x_i,y) - 2\rho 1_{y=y_i})\right].\end{aligned}$$

Now we bound the first term above. Observe that

$$\begin{aligned}\frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i h(x_i,y_i)\right] &= \frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m\sum_{y\in\mathcal{Y}} \sigma_i h(x_i,y)1_{y_i=y}\right]\\ &\leq \frac{1}{m}\sum_{y\in\mathcal{Y}}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i h(x_i,y)1_{y_i=y}\right]\\ &= \sum_{y\in\mathcal{Y}}\frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i h(x_i,y)\left(\frac{\epsilon_i}{2}+\frac{1}{2}\right)\right],\end{aligned}$$

where $\epsilon_i = 2\cdot 1_{y_i=y} - 1$. Since $\epsilon_i \in \{-1,+1\}$, we have that σ_i and $\sigma_i\epsilon_i$ admit the same distribution and, for any $y \in \mathcal{Y}$, each of the terms of the right-hand side can be bounded as follows:

$$\begin{aligned}&\frac{1}{m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i h(x_i,y)\left(\frac{\epsilon_i}{2}+\frac{1}{2}\right)\right]\\ &\leq \frac{1}{2m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i\epsilon_i h(x_i,y)\right] + \frac{1}{2m}\mathop{\mathbb{E}}_{\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i h(x_i,y)\right]\\ &\leq \widehat{\mathfrak{R}}_m(\Pi_1(\mathcal{H})).\end{aligned}$$

Thus, we can write $\frac{1}{m}\mathbb{E}_{S,\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i h(x_i,y_i)\right] \leq k\,\mathfrak{R}_m(\Pi_1(\mathcal{H}))$. To bound the second term, we first apply lemma 9.1 which immediately yields that

$$\begin{aligned}&\frac{1}{m}\mathop{\mathbb{E}}_{S,\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i \max_y(h(x_i,y) - 2\rho 1_{y=y_i})\right]\\ &\qquad\qquad \leq \sum_{y\in\mathcal{Y}}\frac{1}{m}\mathop{\mathbb{E}}_{S,\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i(h(x_i,y) - 2\rho 1_{y=y_i})\right]\end{aligned}$$

and since Rademacher variables are mean zero, we observe that

$$\begin{aligned}\mathop{\mathbb{E}}_{S,\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\sigma_i(h(x_i,y)-2\rho 1_{y=y_i})\right] &= \mathop{\mathbb{E}}_{S,\sigma}\left[\sup_{h\in\mathcal{H}}\left(\sum_{i=1}^{m}\sigma_i h(x_i,y)\right)-2\rho\sum_{i=1}^{m}\sigma_i 1_{y=y_i}\right]\\ &= \mathop{\mathbb{E}}_{S,\sigma}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\sigma_i h(x_i,y)\right]\leq \mathfrak{R}_m(\Pi_1(\mathcal{H}))\end{aligned}$$

which completes the proof. □

These bounds can be generalized to hold uniformly for all $\rho > 0$ at the cost of an additional term $\sqrt{(\log\log_2(2/\rho))/m}$, as in theorem 5.9 and exercise 5.2. As for other margin bounds presented in previous sections, they show the conflict between two terms: the larger the desired pairwise ranking margin ρ, the smaller the middle term, at the price of a larger empirical multi-class classification margin loss $\widehat{R}_{S,\rho}$. Note, however, that here there is additionally a dependency on the number of classes k. This suggests either weaker guarantees when learning with a large number of classes or the need for even larger margins ρ for which the empirical margin loss would be small.

For some hypothesis sets, a simple upper bound can be derived for the Rademacher complexity of $\Pi_1(\mathcal{H})$, thereby making theorem 9.2 more explicit. We will show this for kernel-based hypotheses. Let $K\colon \mathcal{X}\times\mathcal{X}\to\mathbb{R}$ be a PDS kernel and let $\mathbf{\Phi}\colon \mathcal{X}\to\mathbb{H}$ be a feature mapping associated to K. In multi-class classification, a kernel-based hypothesis is based on k weight vectors $\mathbf{w}_1,\ldots,\mathbf{w}_k\in\mathbb{H}$. Each weight vector $\mathbf{w}_l$, $l\in[k]$, defines a scoring function $x\mapsto \mathbf{w}_l\cdot\mathbf{\Phi}(x)$ and the class associated to point $x\in\mathcal{X}$ is given by

$$\operatorname*{argmax}_{y\in\mathcal{Y}} \mathbf{w}_y\cdot\mathbf{\Phi}(x).$$

We denote by $\mathbf{W}$ the matrix formed by these weight vectors: $\mathbf{W}=(\mathbf{w}_1,\ldots,\mathbf{w}_k)^\top$ and for any $p\geq 1$ denote by $\|\mathbf{W}\|_{\mathbb{H},p}$ the $L_{\mathbb{H},p}$ group norm of $\mathbf{W}$ defined by

$$\|\mathbf{W}\|_{\mathbb{H},p}=\big(\sum_{l=1}^{k}\|\mathbf{w}_l\|_{\mathbb{H}}^p\big)^{1/p}.$$

For any $p\geq 1$, the family of kernel-based hypotheses we will consider is[13]

$$\mathcal{H}_{K,p}=\{(x,y)\in\mathcal{X}\times\{1,\ldots,k\}\mapsto \mathbf{w}_y\cdot\mathbf{\Phi}(x)\colon \mathbf{W}=(\mathbf{w}_1,\ldots,\mathbf{w}_k)^\top, \|\mathbf{W}\|_{\mathbb{H},p}\leq\Lambda\}.$$

Proposition 9.3 (Rademacher complexity of multi-class kernel-based hypotheses) *Let $K\colon \mathcal{X}\times\mathcal{X}\to\mathbb{R}$ be a PDS kernel and let $\mathbf{\Phi}\colon \mathcal{X}\to\mathbb{H}$ be a feature mapping associated to*

[13] The hypothesis set $\mathcal{H}$ can also be defined via $\mathcal{H}=\{h\in\mathbb{R}^{\mathcal{X}\times\mathcal{Y}}\colon h(\cdot,y)\in\mathbb{H}\wedge\|h\|_{K,p}\leq\Lambda\}$, where $\|h\|_{K,p}=\big(\sum_{y=1}^{k}\|h(\cdot,y)\|_{\mathbb{H}}^p\big)^{1/p}$, without referring to a feature mapping for K.

K. Assume that there exists $r > 0$ such that $K(x, x) \leq r^2$ for all $x \in \mathcal{X}$. Then, for any $m \geq 1$, $\mathfrak{R}_m(\Pi_1(\mathcal{H}_{K,p}))$ can be bounded as follows:

$$\mathfrak{R}_m(\Pi_1(\mathcal{H}_{K,p})) \leq \sqrt{\frac{r^2\Lambda^2}{m}}.$$

Proof: Let $S = (x_1, \ldots, x_m)$ denote a sample of size m. Observe that for all $l \in [k]$, the inequality $\|\mathbf{w}_l\|_{\mathbb{H}} \leq \big(\sum_{l=1}^k \|\mathbf{w}_l\|_{\mathbb{H}}^p\big)^{1/p} = \|\mathbf{W}\|_{\mathbb{H},p}$ holds. Thus, the condition $\|\mathbf{W}\|_{\mathbb{H},p} \leq \Lambda$ implies that $\|\mathbf{w}_l\|_{\mathbb{H}} \leq \Lambda$ for all $l \in [k]$. In view of that, the Rademacher complexity of the hypothesis set $\Pi_1(\mathcal{H}_{K,p})$ can be expressed and bounded as follows:

$$\begin{aligned}
\mathfrak{R}_m(\Pi_1(\mathcal{H}_{K,p})) &= \frac{1}{m} \mathop{\mathbb{E}}_{S,\boldsymbol{\sigma}}\Bigg[\sup_{\substack{y\in\mathcal{Y}\\ \|\mathbf{W}\|\leq\Lambda}} \Big\langle \mathbf{w}_y, \sum_{i=1}^m \sigma_i \mathbf{\Phi}(x_i)\Big\rangle\Bigg] \\
&\leq \frac{1}{m} \mathop{\mathbb{E}}_{S,\boldsymbol{\sigma}}\Bigg[\sup_{\substack{y\in\mathcal{Y}\\ \|\mathbf{W}\|\leq\Lambda}} \|\mathbf{w}_y\|_{\mathbb{H}} \Big\|\sum_{i=1}^m \sigma_i \Phi(x_i)\Big\|_{\mathbb{H}}\Bigg] && \text{(Cauchy-Schwarz ineq.)} \\
&\leq \frac{\Lambda}{m} \mathop{\mathbb{E}}_{S,\boldsymbol{\sigma}}\Bigg[\Big\|\sum_{i=1}^m \sigma_i \Phi(x_i)\Big\|_{\mathbb{H}}\Bigg] \\
&\leq \frac{\Lambda}{m}\Bigg[\mathop{\mathbb{E}}_{S,\boldsymbol{\sigma}}\Bigg[\Big\|\sum_{i=1}^m \sigma_i \Phi(x_i)\Big\|_{\mathbb{H}}^2\Bigg]\Bigg]^{1/2} && \text{(Jensen's inequality)} \\
&= \frac{\Lambda}{m}\Bigg[\mathop{\mathbb{E}}_{S,\boldsymbol{\sigma}}\Bigg[\sum_{i=1}^m \|\Phi(x_i)\|_{\mathbb{H}}^2\Bigg]\Bigg]^{1/2} && (i \neq j \Rightarrow \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}[\sigma_i\sigma_j] = 0) \\
&= \frac{\Lambda}{m}\Bigg[\mathop{\mathbb{E}}_{S,\boldsymbol{\sigma}}\Bigg[\sum_{i=1}^m K(x_i, x_i)\Bigg]\Bigg]^{1/2} \\
&\leq \frac{\Lambda\sqrt{mr^2}}{m} = \sqrt{\frac{r^2\Lambda^2}{m}},
\end{aligned}$$

which concludes the proof. □

Combining theorem 9.2 and proposition 9.3 yields directly the following result.

Corollary 9.4 (Margin bound for multi-class classification with kernel-based hypotheses) *Let $K\colon \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ be a PDS kernel and let $\mathbf{\Phi}\colon \mathcal{X} \to \mathbb{H}$ be a feature mapping associated to K. Assume that there exists $r > 0$ such that $K(x, x) \leq r^2$ for all $x \in \mathcal{X}$. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following multi-class classification generalization bound holds for all $h \in \mathcal{H}_{K,p}$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + 4k\sqrt{\frac{r^2\Lambda^2/\rho^2}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}. \tag{9.11}$$

In the next two sections, we describe multi-class classification algorithms that belong to two distinct families: *uncombined algorithms*, which are defined by a single optimization problem, and *aggregated algorithms*, which are obtained by training multiple binary classifications and by combining their outputs.

9.3 Uncombined multi-class algorithms

In this section, we describe three algorithms designed specifically for multi-class classification. We start with a multi-class version of SVMs, then describe a boosting-type multi-class algorithm, and conclude with *decision trees*, which are often used as base learners in boosting.

9.3.1 Multi-class SVMs

We describe an algorithm that can be derived directly from the theoretical guarantees presented in the previous section. Proceeding as in section 5.4 for classification, the guarantee of corollary 9.4 can be expressed as follows: for any $\delta > 0$, with probability at least $1 - \delta$, for all $h \in \mathcal{H}_{K,2} = \{(x, y) \to \mathbf{w}_y \cdot \mathbf{\Phi}(x) \colon \mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_k)^\top, \sum_{l=1}^k \|\mathbf{w}_l\|^2 \le \Lambda^2\}$,

$$R(h) \le \frac{1}{m} \sum_{i=1}^m \xi_i + 4k\sqrt{\frac{r^2\Lambda^2}{m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}, \tag{9.12}$$

where $\xi_i = \max\big(1 - [\mathbf{w}_{y_i} \cdot \mathbf{\Phi}(x_i) - \max_{y' \neq y_i} \mathbf{w}_{y'} \cdot \mathbf{\Phi}(x_i)], 0\big)$ for all $i \in [m]$.

An algorithm based on this theoretical guarantee consists of minimizing the right-hand side of (9.12), that is, minimizing an objective function with a term corresponding to the sum of the slack variables ξ_i, and another one minimizing $\|\mathbf{W}\|_{\mathbb{H},2}$ or equivalently $\sum_{l=1}^k \|\mathbf{w}_l\|^2$. This is precisely the optimization problem defining the *multi-class SVM* algorithm:

$$\begin{aligned}
&\min_{\mathbf{W}, \boldsymbol{\xi}} \frac{1}{2} \sum_{l=1}^k \|\mathbf{w}_l\|^2 + C \sum_{i=1}^m \xi_i \\
&\text{subject to: } \forall i \in [m], \forall l \in \mathcal{Y} - \{y_i\}, \\
&\qquad \mathbf{w}_{y_i} \cdot \mathbf{\Phi}(x_i) \ge \mathbf{w}_l \cdot \mathbf{\Phi}(x_i) + 1 - \xi_i, \\
&\qquad \xi_i \ge 0.
\end{aligned}$$

The decision function learned is of the form $x \mapsto \operatorname{argmax}_{l \in \mathcal{Y}} \mathbf{w}_l \cdot \mathbf{\Phi}(x)$. As with the primal problem of SVMs, this is a convex optimization problem: the objective function is convex, since it is a sum of convex functions, and the constraints are affine and thus qualified. The objective and constraint functions are differentiable, and the KKT conditions hold at the optimum. Defining the Lagrangian and applying these conditions leads to the equivalent dual optimization problem, which can be expressed in terms of the kernel function K alone:

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^{m \times k}} \sum_{i=1}^{m} \boldsymbol{\alpha}_i \cdot \mathbf{e}_{y_i} - \frac{1}{2} \sum_{i=1}^{m} (\boldsymbol{\alpha}_i \cdot \boldsymbol{\alpha}_j) K(x_i, x_j)$$
$$\text{subject to: } \forall i \in [m], (0 \leq \alpha_{iy_i} \leq C) \wedge (\forall j \neq y_i, \alpha_{ij} \leq 0) \wedge (\boldsymbol{\alpha}_i \cdot \mathbf{1} = 0).$$

Here, $\boldsymbol{\alpha} \in \mathbb{R}^{m \times k}$ is a matrix, $\boldsymbol{\alpha}_i$ denotes the ith row of $\boldsymbol{\alpha}$, and $\mathbf{e}_l$ the lth unit vector in $\mathbb{R}^k$, $l \in [k]$. Both the primal and dual problems are simple QPs generalizing those of the standard SVM algorithm. However, the size of the solution and the number of constraints for both problems is in $\Omega(mk)$, which, for a large number of classes k, can make it difficult to solve. However, there exist specific optimization solutions designed for this problem based on a decomposition of the problem into m disjoint sets of constraints.

9.3.2 Multi-class boosting algorithms

We describe a boosting algorithm for multi-class classification called *AdaBoost.MH*, which in fact coincides with a special instance of AdaBoost. An alternative multi-class classification algorithm based on similar boosting ideas, AdaBoost.MR, is described and analyzed in exercise 9.4. AdaBoost.MH applies to the multi-label setting where $\mathcal{Y} = \{-1, +1\}^k$. As in the binary case, it returns a convex combination of base classifiers selected from a hypothesis set $\mathcal{H} = \{h_1, \ldots, h_N\}$. Let F be the following objective function defined for all samples $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ and $\bar{\boldsymbol{\alpha}} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_N) \in \mathbb{R}^N$, $N \geq 1$, by

$$F(\bar{\boldsymbol{\alpha}}) = \sum_{i=1}^{m} \sum_{l=1}^{k} e^{-y_i[l] f_N(x_i, l)} = \sum_{i=1}^{m} \sum_{l=1}^{k} e^{-y_i[l] \sum_{j=1}^{N} \bar{\alpha}_j h_j(x_i, l)}, \tag{9.13}$$

where $f_N = \sum_{j=1}^{N} \bar{\alpha}_j h_j$ and where $y_i[l]$ denotes the lth coordinate of y_i for any $i \in [m]$ and $l \in [k]$. F is a convex and differentiable upper bound on the multi-class multi-label loss:

$$\sum_{i=1}^{m} \sum_{l=1}^{k} 1_{y_i[l] \neq f_N(x_i, l)} \leq \sum_{i=1}^{m} \sum_{l=1}^{k} e^{-y_i[l] f_N(x_i, l)}, \tag{9.14}$$

since for any $x \in \mathcal{X}$ with label $y = f(x)$ and any $l \in [k]$, the inequality $1_{y[l] \neq f_N(x,l)} \leq e^{-y[l] f_N(x,l)}$ holds. Using the same arguments as in section 7.2.2, we see that AdaBoost.MH coincides exactly with the application of coordinate descent to the objective function F. Figure 9.1 gives the pseudocode of the algorithm in the case where the base classifiers are functions mapping from $\mathcal{X} \times \mathcal{Y}$ to $\{-1, +1\}$. The algorithm takes as input a labeled sample $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ and maintains a distribution $\mathcal{D}_t$ over $\{1, \ldots, m\} \times \mathcal{Y}$. The remaining details of the

```
AdaBoost.MH($S = ((x_1, y_1), \ldots, (x_m, y_m))$)
 1  for $i \leftarrow 1$ to $m$ do
 2      for $l \leftarrow 1$ to $k$ do
 3          $\mathcal{D}_1(i, l) \leftarrow \frac{1}{mk}$
 4  for $j \leftarrow 1$ to $N$ do
 5      $h_j \leftarrow$ base classifier in $\mathcal{H}$ with small error $\epsilon_j = \mathbb{P}_{(i,l)\sim\mathcal{D}_j}[h_j(x_i, l) \neq y_i[l]]$
 6      $\bar{\alpha}_j \leftarrow \frac{1}{2}\log\frac{1-\epsilon_j}{\epsilon_j}$
 7      $Z_t \leftarrow 2[\epsilon_j(1-\epsilon_j)]^{\frac{1}{2}}$    ▷ normalization factor
 8      for $i \leftarrow 1$ to $m$ do
 9          for $l \leftarrow 1$ to $k$ do
10              $\mathcal{D}_{j+1}(i, l) \leftarrow \frac{\mathcal{D}_j(i,l)\exp(-\bar{\alpha}_j y_i[l] h_j(x_i, l))}{Z_j}$
11  $f_N \leftarrow \sum_{j=1}^{N} \bar{\alpha}_j h_j$
12  return $h = \mathrm{sgn}(f_N)$
```

Figure 9.1
AdaBoost.MH algorithm, for $\mathcal{H} \subseteq (\{-1, +1\}^k)^{\mathcal{X}\times\mathcal{Y}}$.

algorithm are similar to AdaBoost. In fact, AdaBoost.MH exactly coincides with AdaBoost applied to the training sample derived from S by splitting each labeled point (x_i, y_i) into k labeled examples $((x_i, l), y_i[l])$, with each example (x_i, l) in $\mathcal{X} \times \mathcal{Y}$ and its label in $\{-1, +1\}$:

$$(x_i, y_i) \to ((x_i, 1), y_i[1]), \ldots, ((x_i, k), y_i[k]), i \in [m].$$

Let S' denote the resulting sample, then $S' = ((x_1, 1), y_1[1]), \ldots, (x_m, k), y_m[k]))$. S' contains mk examples and the expression of the objective function F in (9.13) coincides exactly with that of the objective function of AdaBoost for the sample S'. In view of this connection, the theoretical analysis along with the other observations we presented for AdaBoost in chapter 7 also apply here. Hence, we will focus on aspects related to the computational efficiency and to the weak learning condition that are specific to the multi-class scenario.

The complexity of the algorithm is that of AdaBoost applied to a sample of size mk. For $\mathcal{X} \subseteq \mathbb{R}^d$, using boosting stumps as base classifiers, the complexity of the algorithm is therefore in $O((mk)\log(mk) + mkdN)$. Thus, for a large number of classes k, the algorithm may become impractical using a single processor. The

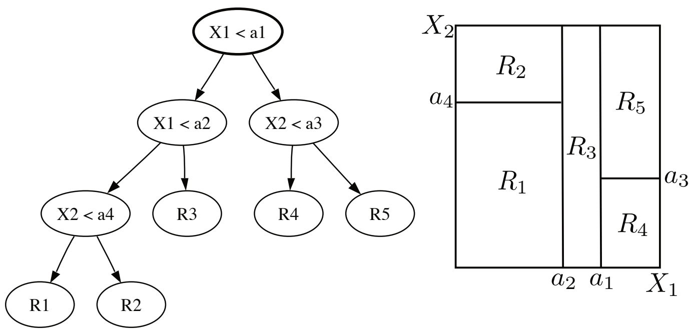


Figure 9.2
Left: example of a decision tree with numerical questions based on two variables X_1 and X_2. Here, each leaf is marked with the region it defines. The class labeling for a leaf is obtained via majority vote based on the training points falling in the region it defines. Right: Partition of the two-dimensional space induced by that decision tree.

weak learning condition for the application of AdaBoost in this scenario requires that at each round there exists a base classifier $h_j : \mathcal{X} \times \mathcal{Y} \to \{-1, +1\}$ such that $\mathbb{P}_{(i,l)\sim \mathcal{D}_j}[h_j(x_i, l) \neq y_i[l]] < 1/2$. This may be hard to achieve if some classes difficult to distinguish between. It is also more difficult in this context to come up with "rules of thumb" h_j defined over $\mathcal{X} \times \mathcal{Y}$.

9.3.3 Decision trees

We present and discuss the general learning method of *decision trees* that can be used in multi-class classification, but also in other learning problems such as regression (chapter 11) and clustering. Although the empirical performance of decision trees often is not state-of-the-art, decision trees can be used as weak learners with boosting to define effective learning algorithms. Decision trees are also typically fast to train and evaluate and relatively easy to interpret.

Definition 9.5 (Binary decision tree) *A* binary decision tree *is a tree representation of a partition of the feature space. Figure 9.2 shows a simple example in the case of a two-dimensional space based on two features X_1 and X_2, as well as the partition it represents. Each interior node of a decision tree corresponds to a question related to the features. It can be a* numerical question *of the form $X_i \leq a$ for a feature variable X_i, $i \in [N]$, and some threshold $a \in \mathbb{R}$, as in the example of figure 9.2, or a* categorical question *such as $X_i \in \{blue, white, red\}$, when feature X_i takes a categorical value such as a color. Each leaf is labeled with a label $l \in \mathcal{Y}$.*

Decision trees can be defined using more complex node questions, resulting in partitions based on more complex decision surfaces. For example, *binary space*

```
GreedyDecisionTrees($S = ((x_1, y_1), \ldots, (x_m, y_m))$)
1  tree ← {n_0}  ▷ root node.
2  for $t \leftarrow 1$ to $T$ do
3      $(\mathsf{n}_t, \mathsf{q}_t) \leftarrow \operatorname{argmax}_{(\mathsf{n},\mathsf{q})} \widetilde{F}(\mathsf{n}, \mathsf{q})$
4      Split(tree, $\mathsf{n}_t$, $\mathsf{q}_t$)
5  return tree
```

Figure 9.3
Greedy algorithm for building a decision tree from a labeled sample S. The procedure Split(tree, $\mathsf{n}_t, \mathsf{q}_t$) splits node n_t by making it an internal node with question q_t and leaf children $\mathsf{n}_-(\mathsf{n}, \mathsf{q})$ and $\mathsf{n}_+(\mathsf{n}, \mathsf{q})$, each labeled with the dominating class of the region it defines, with ties broken arbitrarily.

partition (BSP) trees partition the space with convex polyhedral regions, based on questions of the form $\sum_{i=1}^{n} \alpha_i X_i \leq a$, and *sphere trees* partition with pieces of spheres based on questions of the form $\|X - a_0\| \leq a$, where X is a feature vector, a_0 a fixed vector, and a is a fixed positive real number. More complex tree questions lead to richer partitions and thus hypothesis sets, which can cause overfitting in the absence of a sufficiently large training sample. They also increase the computational complexity of prediction and training. Decision trees can also be generalized to branching factors greater than two, but binary trees are most commonly used due their more limited computational cost.

Prediction/partitioning: To predict the label of any point $x \in \mathcal{X}$ we start at the root node of the decision tree and go down the tree until a leaf is found, by moving to the right child of a node when the response to the node question is positive, and to the left child otherwise. When we reach a leaf, we associate x with the label of this leaf.

Thus, each leaf defines a *region* of $\mathcal{X}$ formed by the set of points corresponding exactly to the same node responses and thus the same traversal of the tree. By definition, no two regions intersect and all points belong to exactly one region. Thus, leaf regions define a partition of $\mathcal{X}$, as shown in the example of figure 9.2. In multi-class classification, the label of a leaf is determined using the training sample: the class with the majority representation among the training points falling in a leaf region defines the label of that leaf, with ties broken arbitrarily.

Learning: We will discuss two different methods for learning a decision tree using a labeled sample. The first method is a greedy technique. This is motivated by the fact that the general problem of finding a decision tree with the smallest error is NP-hard. The method consists of starting with a tree reduced to a single (root) node, which is a leaf whose label is the class that has majority over the entire sample. Next, at each round, a node n_t is split based on some question q_t. The pair $(\mathsf{n}_t, \mathsf{q}_t)$ is chosen so that the *node impurity* is maximally decreased according to some measure of impurity F. We denote by $F(\mathsf{n})$ the impurity of n. The decrease in node impurity after a split of node n based on question q is defined as follows. Let $\mathsf{n}_+(\mathsf{n}, \mathsf{q})$ denote the right child of n after the split, $\mathsf{n}_-(\mathsf{n}, \mathsf{q})$ the left child, and $\eta(\mathsf{n}, \mathsf{q})$ the fraction of the points in the region defined by n that are moved to $\mathsf{n}_-(\mathsf{n}, \mathsf{q})$. The total impurity of the leaves $\mathsf{n}_-(\mathsf{n}, \mathsf{q})$ and $\mathsf{n}_+(\mathsf{n}, \mathsf{q})$ is therefore $\eta(\mathsf{n}, \mathsf{q})F(\mathsf{n}_-(\mathsf{n}, \mathsf{q})) + (1-\eta(\mathsf{n}, \mathsf{q}))F(\mathsf{n}_+(\mathsf{n}, \mathsf{q}))$. Thus, the decrease in impurity $\widetilde{F}(\mathsf{n}, \mathsf{q})$ by that split is given by

$$\widetilde{F}(\mathsf{n}, \mathsf{q}) = F(\mathsf{n}) - [\eta(\mathsf{n}, \mathsf{q})F(\mathsf{n}_-(\mathsf{n}, \mathsf{q})) + (1 - \eta(\mathsf{n}, \mathsf{q}))F(\mathsf{n}_+(\mathsf{n}, \mathsf{q}))].$$

Figure 9.3 shows the pseudocode of this greedy construction based on $\widetilde{F}$. In practice, the algorithm is stopped once all nodes have reached a sufficient level of purity, when the number of points per leaf has become too small for further splitting or based on some other similar heuristic.

For any node n and class $l \in [k]$, let $p_l(\mathsf{n})$ denote the fraction of points at n that belong to class l. Then, the three most commonly used measures of node impurity F are defined as follows:

$$F(\mathsf{n}) = \begin{cases} 1 - \max_{l \in [k]} p_l(\mathsf{n}) & \textit{misclassification}; \\ -\sum_{l=1}^{k} p_l(\mathsf{n}) \log_2 p_l(\mathsf{n}) & \textit{entropy}; \\ \sum_{l=1}^{k} p_l(\mathsf{n})(1 - p_l(\mathsf{n})) & \textit{Gini index}. \end{cases}$$

Figure 9.4 illustrates these definitions in the special cases of two classes ($k = 2$). The entropy and Gini index impurity functions are upper bounds on the misclassification impurity function. All three functions are concave, which ensures that

$$F(\mathsf{n}) - [\eta(\mathsf{n}, \mathsf{q})F(\mathsf{n}_-(\mathsf{n}, \mathsf{q})) + (1 - \eta(\mathsf{n}, \mathsf{q}))F(\mathsf{n}_+(\mathsf{n}, \mathsf{q}))] \geq 0.$$

However, the misclassification function is piecewise linear, so $\widetilde{F}(\mathsf{n}, \mathsf{q})$ is zero if the fraction of positive points remains less than (or more than) half after a split. In some cases, the impurity cannot be decreased by any split using that criterion. In contrast, the entropy and Gini functions are strictly concave, which guarantees a strict decrease in impurity. Furthermore, they are differentiable which is a useful feature for numerical optimization. Thus, the Gini index and the entropy criteria are typically preferred in practice.

The greedy method just described faces some issues. One issue relates to the greedy nature of the algorithm: a seemingly bad split may dominate subsequent useful splits, which could lead to trees with less impurity overall. This can be

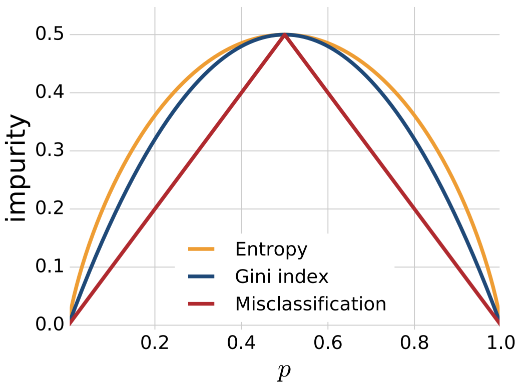


Figure 9.4
Three node impurity definitions plotted as a function of the fraction of positive examples in the binary case: misclassification, entropy (scaled by 0.5 to set the maximum to the same value for all three functions), and the Gini index.

addressed to a certain extent by using a look-ahead of some depth d to determine the splitting decisions, but such look-aheads can be computationally very costly. Another issue relates to the size of the resulting tree. To achieve some desired level of impurity, trees of relatively large sizes may be needed. However, larger trees define overly complex hypotheses with high VC-dimensions (see exercise 9.5) and thus could overfit.

An alternative method for learning decision trees using a labeled training sample is based on the so-called *grow-then-prune strategy*. First a very large tree is grown until it fully fits the training sample or until no more than a very small number of points are left at each leaf. Then, the resulting tree, denoted as tree, is pruned back to minimize an objective function defined (based on generalization bounds) as the sum of an empirical error and a complexity term. The complexity can be expressed in terms of the size of $\widetilde{\mathsf{tree}}$, the set of leaves of tree. The resulting objective is

$$G_\lambda(\mathsf{tree}) = \sum_{\mathsf{n} \in \widetilde{\mathsf{tree}}} |\mathsf{n}| F(\mathsf{n}) + \lambda |\widetilde{\mathsf{tree}}|, \tag{9.15}$$

where $\lambda \geq 0$ is a regularization parameter determining the trade-off between misclassification, or more generally impurity, versus tree complexity. For any tree tree', we denote by $\widehat{R}(\mathsf{tree}')$ the total empirical error $\sum_{\mathsf{n} \in \widetilde{\mathsf{tree}}'} |\mathsf{n}| F(\mathsf{n})$. We seek a sub-tree tree_λ of tree that minimizes G_λ and that has the smallest size. tree_λ can be shown to be unique. To determine tree_λ, the following pruning method is used, which defines a finite sequence of nested sub-trees $\mathsf{tree}^{(0)}, \ldots, \mathsf{tree}^{(n)}$. We start with the full tree $\mathsf{tree}^{(0)} = \mathsf{tree}$ and for any $i \in \{0, \ldots, n-1\}$, define $\mathsf{tree}^{(i+1)}$ from $\mathsf{tree}^{(i)}$ by collapsing an internal node n' of $\mathsf{tree}^{(i)}$, that is by replacing the sub-tree rooted at n' with a leaf, or equivalently by combining the regions of all the leaves dominated by n'. n' is chosen so that collapsing it causes the smallest per node increase in $\widehat{R}(\mathsf{tree}^{(i)})$, that is the smallest $r(\mathsf{tree}^{(i)}, \mathsf{n}')$ defined by

$$r(\mathsf{tree}^{(i)}, \mathsf{n}') = \frac{|\mathsf{n}'|F(\mathsf{n}') - \widehat{R}(\mathsf{tree}')}{|\widetilde{\mathsf{tree}}'| - 1},$$

where n' is an internal node of $\mathsf{tree}^{(i)}$. If several nodes n' in $\mathsf{tree}^{(i)}$ cause the same smallest increase per node $r(\mathsf{tree}^{(i)}, \mathsf{n}')$, then all of them are pruned to define $\mathsf{tree}^{(i+1)}$ from $\mathsf{tree}^{(i)}$. This procedure continues until the tree $\mathsf{tree}^{(n)}$ obtained has a single node. The sub-tree tree_λ can be shown to be among the elements of the sequence $\mathsf{tree}^{(0)}, \ldots, \mathsf{tree}^{(n)}$. The parameter λ is determined via n-fold cross-validation.

Decision trees seem relatively easy to interpret, and this is often underlined as one of their most useful features. However, such interpretations should be carried out with care since decision trees are *unstable*: small changes in the training data may lead to very different splits and thus entirely different trees, as a result of their hierarchical nature. Decision trees can also be used in a natural manner to deal with the problem of *missing features*, which often appears in learning applications; in practice, some features values may be missing because the proper measurements were not taken or because of some noise source causing their systematic absence. In such cases, only those variables available at a node can be used in prediction. Finally, decision trees can be used and learned from data in a similar way in the *regression* setting (see chapter 11).[14]

9.4 Aggregated multi-class algorithms

In this section, we discuss a different approach to multi-class classification that reduces the problem to that of multiple binary classification tasks. A binary classification algorithm is then trained for each of these tasks independently, and the multi-class predictor is defined as a combination of the hypotheses returned by each of these algorithms. We first discuss two standard techniques for the reduction of multi-class classification to binary classification, and then show that they are both special instances of a more general framework.

[14] The only changes to the description for classification are the following. For prediction, the label of a leaf is defined as the mean squared average of the labels of the points falling in that region. For learning, the impurity function is the mean squared error.

9.4.1 One-versus-all

Let $S = ((x_1, y_1), \ldots, x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ be a labeled training sample. A straightforward reduction of the multi-class classification to binary classification is based on the so-called *one-versus-all (OVA) or one-versus-the-rest technique*. This technique consists of learning k binary classifiers $h_l \colon \mathcal{X} \to \{-1, +1\}$, $l \in \mathcal{Y}$, each seeking to discriminate one class $l \in \mathcal{Y}$ from all the others. For any $l \in \mathcal{Y}$, h_l is obtained by training a binary classification algorithm on the full sample S after relabeling points in class l with 1 and all others with -1. For $l \in \mathcal{Y}$, assume that h_l is derived from the sign of a scoring function $f_l \colon \mathcal{X} \to \mathbb{R}$, that is $h_l = \operatorname{sgn}(f_l)$, as in the case of many of the binary classification algorithms discussed in the previous chapters. Then, the multi-class hypothesis $h \colon \mathcal{X} \to \mathcal{Y}$ defined by the OVA technique is given by:

$$\forall x \in \mathcal{X}, \quad h(x) = \operatorname*{argmax}_{l \in \mathcal{Y}} f_l(x). \tag{9.16}$$

This formula may seem similar to those defining a multi-class classification hypothesis in the case of uncombined algorithms. Note, however, that for uncombined algorithms the functions f_l are learned together, while here they are learned independently. Formula (9.16) is well-founded when the scores given by functions f_l can be interpreted as confidence scores, that is when $f_l(x)$ is learned as an estimate of the probability of x conditioned on class l. However, in general, the scores given by functions f_l, $l \in \mathcal{Y}$, are not comparable and the OVA technique based on (9.16) admits *no principled justification*. This is sometimes referred to as a *calibration problem*. Clearly, this problem cannot be corrected by simply normalizing the scores of each function to make their magnitudes uniform, or by applying other similar heuristics. When it is justifiable, the OVA technique is simple and its computational cost is k times that of training a binary classification algorithm, which is similar to the computation costs for many uncombined algorithms.

9.4.2 One-versus-one

An alternative technique, known as the *one-versus-one (OVO) technique*, consists of using the training data to learn (independently), for each pair of distinct classes $(l, l') \in \mathcal{Y}^2$, $l \neq l'$, a binary classifier $h_{ll'} \colon \mathcal{X} \to \{-1, 1\}$ discriminating between classes l and l'. For any $(l, l') \in \mathcal{Y}^2$, $h_{ll'}$ is obtained by training a binary classification algorithm on the sub-sample containing exactly the points labeled with l or l', with the value $+1$ returned for class l' and -1 for class l. This requires training $\binom{k}{2} = k(k-1)/2$ classifiers, which are combined to define a multi-class classification hypothesis h via majority vote:

$$\forall x \in \mathcal{X}, \quad h(x) = \operatorname*{argmax}_{l' \in \mathcal{Y}} \big|\{l \colon h_{ll'}(x) = 1\}\big|. \tag{9.17}$$

Table 9.1
Comparison of the time complexity the OVA and OVO techniques for both training and testing. The table assumes a full training sample of size m with each class represented by m/k points. The time for training a binary classification algorithm on a sample of size n is assumed to be in $O(n^\alpha)$. Thus, the training time for the OVO technique is in $O(k^2(m/k)^\alpha) = O(k^{2-\alpha}m^\alpha)$. c_t denotes the cost of testing a single classifier.

	Training	Testing
OVA	$O(km^\alpha)$	$O(kc_t)$
OVO	$O(k^{2-\alpha}m^\alpha)$	$O(k^2c_t)$

Thus, for a fixed point $x \in \mathcal{X}$, if we describe the prediction values $h_{ll'}(x)$ as the results of the matches in a tournament between two players l and l', with $h_{ll'}(x) = 1$ indicating l' winning over l, then the class predicted by h can be interpreted as the one with the largest number of wins in that tournament.

Let $x \in \mathcal{X}$ be a point belonging to class l'. By definition of the OVO technique, if $h_{ll'}(x) = 1$ for all $l \neq l'$, then the class associated to x by OVO is the correct class l' since $\big|\{l \colon h_{ll'}(x) = 1\}\big| = k - 1$ and no other class can reach $(k-1)$ wins. By contraposition, if the OVO hypothesis misclassifies x, then at least one of the $(k-1)$ binary classifiers $h_{ll'}$, $l \neq l'$, incorrectly classifies x. Assume that the generalization error of all binary classifiers $h_{ll'}$ used by OVO is at most r, then, in view of this discussion, the generalization error of the hypothesis returned by OVO is at most $(k-1)r$.

The OVO technique is not subject to the calibration problem pointed out in the case of the OVA technique. However, when the size of the sub-sample containing members of the classes l and l' is relatively small, $h_{ll'}$ may be learned without sufficient data or with increased risk of overfitting. Another concern often raised for the use of this technique is the computational cost of training $k(k-1)/2$ binary classifiers versus that of the OVA technique.

Taking a closer look at the computational requirements of these two methods reveals, however, that the disparity may not be so great and that in fact under some assumptions the time complexity of training for OVO could be less than that of OVA. Table 9.1 compares the computational complexity of these methods both for training and testing assuming that the complexity of training a binary classifier on a sample of size m is in $O(m^\alpha)$ and that each class is equally represented in the training set, that is by m/k points. Under these assumptions, if $\alpha \in [2, 3)$ as in the case of some algorithms solving a QP problem, such as SVMs, then the time complexity of training for the OVO technique is in fact more favorable than that of OVA. For $\alpha = 1$, the two are comparable and it is only for sub-linear algorithms that the OVA technique would benefit from a better complexity. In all cases, at test time, OVO requires $k(k-1)/2$ classifier evaluations, which is $(k-1)$

times more than OVA. However, for some algorithms the evaluation time for each classifier could be much smaller for OVO. For example, in the case of SVMs, the average number of support vectors may be significantly smaller for OVO, since each classifier is trained on a significantly smaller sample. If the number of support vectors is k times smaller and if sparse feature representations are used, then the time complexities of both techniques for testing are comparable.

9.4.3 Error-correcting output codes

A more general method for the reduction of multi-class to binary classification is based on the idea of *error-correcting output codes (ECOC)*. This technique consists of assigning to each class $l \in \mathcal{Y}$ a *code word* of length $c \geq 1$, which in the simplest case is a binary vector $\mathbf{M}_l \in \{-1, +1\}^c$. $\mathbf{M}_l$ serves as a signature for class l, and together these vectors define a matrix $\mathbf{M} \in \{-1, +1\}^{k \times c}$ whose lth row is $\mathbf{M}_l$, as illustrated by figure 9.5. Next, for each column $j \in [c]$, a binary classifier $h_j \colon \mathcal{X} \to \{-1, +1\}$ is learned using the full training sample S, after all points that belong to a class represented by $+1$ in column j are labeled with $+1$, while all other points are labeled with -1. For any $x \in \mathcal{X}$, let $\mathbf{h}(x)$ denote the vector $\mathbf{h}(x) = (h_1(x), \ldots, h_c(x))^\top$. Then, the multi-class hypothesis $h \colon \mathcal{X} \to \mathcal{Y}$ is defined by

$$\forall x \in \mathcal{X}, \quad h(x) = \operatorname*{argmin}_{l \in \mathcal{Y}} d_H\big(\mathbf{M}_l, \mathbf{h}(x)\big). \tag{9.18}$$

Thus, the class predicted is the one whose signatures is the closest to $\mathbf{h}(x)$ in Hamming distance. Figure 9.5 illustrates this definition: no row of matrix $\mathbf{M}$ matches the vector of predictions $\mathbf{h}(x)$ in that case, but the third row shares the largest number of components with $\mathbf{h}(x)$.

The success of the ECOC technique depends on the minimal Hamming distance between the class code words. Let d denote that distance, then up to $r_0 = \left\lfloor \frac{d-1}{2} \right\rfloor$ binary classification errors can be corrected by this technique: by definition of d, even if $r < r_0$ binary classifiers h_l misclassify $x \in \mathcal{X}$, $\mathbf{h}(x)$ is closest to the code word of the correct class of x. For a fixed c, the design of error-correction matrix $\mathbf{M}$ is subject to a trade-off, since larger d values may imply substantially more difficult binary classification tasks. In practice, each column may correspond to a class feature determined based on domain knowledge.

The ECOC technique just described can be extended in two ways. First, instead of using only the label predicted by each classifier h_j the magnitude of the scores defining h_j is used. Thus, if $h_j = \operatorname{sgn}(f_j)$ for some function f_j whose values can be interpreted as confidence scores, then the multi-class hypothesis $h \colon \mathcal{X} \to \mathcal{Y}$ is codes

classes	1	2	3	4	5	6
1	-1	-1	-1	+1	-1	-1
2	+1	-1	-1	-1	-1	-1
3	-1	+1	+1	-1	+1	-1
4	+1	+1	-1	-1	-1	-1
5	+1	+1	-1	-1	+1	-1
6	-1	-1	+1	+1	-1	+1
7	-1	-1	+1	-1	-1	-1
8	-1	+1	-1	+1	-1	-1

$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
-1	+1	+1	-1	+1	+1

new example x

Figure 9.5
Illustration of error-correcting output codes for multi-class classification. Left: binary code matrix $\mathbf{M}$, with each row representing the code word of length $c = 6$ of a class $l \in [8]$. Right: vector of predictions $\mathbf{h}(x)$ for a test point x. The ECOC classifier assigns label 3 to x, since the binary code for the third class yields the minimal Hamming distance with $\mathbf{h}(x)$ (distance of 1).

defined by

$$\forall x \in \mathcal{X}, \quad h(x) = \operatorname*{argmin}_{l \in \mathcal{Y}} \sum_{j=1}^{c} L(m_{lj} f_j(x)), \tag{9.19}$$

where (m_{lj}) are the entries of $\mathbf{M}$ and where $L\colon \mathbb{R} \to \mathbb{R}_+$ is a loss function. When L is defined by $L(x) = \frac{1-\operatorname{sgn}(x)}{2}$ for all $x \in \mathcal{X}$ and $h_l = f_l$, we can write:

$$\sum_{j=1}^{c} L(m_{lj} f_j(x)) = \sum_{j=1}^{c} \frac{1 - \operatorname{sgn}(m_{lj} h_j(x))}{2} = d_h(\mathbf{M}_l, \mathbf{h}(x)),$$

and (9.19) coincides with (9.18). Furthermore, ternary codes can be used with matrix entries in $\{-1, 0, +1\}$ so that examples in classes labeled with 0 are disregarded when training a binary classifier for each column. With these extensions, both OVA and OVO become special instances of the ECOC technique. The matrix $\mathbf{M}$ for OVA is a square matrix, that is $c = k$, with all terms equal to -1 except from the diagonal ones which are all equal to $+1$. The matrix $\mathbf{M}$ for OVO has $c = k(k-1)/2$ columns. Each column corresponds to a pair of distinct classes (l, l'), $l \neq l'$, with all entries equal to 0 except from the one with row l, which is -1, and the one with row l', which is $+1$.

Since the values of the scoring functions are assumed to be confidence scores, $m_{lj} f_j(x)$ can be interpreted as the margin of classifier j on point x and (9.19) is thus based on some loss L defined with respect to the binary classifier's margin.

A further extension of ECOC consists of extending discrete codes to continuous ones by letting the matrix entries take arbitrary real values and by using the training sample to *learn* matrix $\mathbf{M}$. Starting with a discrete version of $\mathbf{M}$, c binary classifiers with scoring functions f_l, $l \in [c]$, are first learned as described previously. We will denote by $\mathbf{F}(x)$ the vector $(f_1(x), \ldots, f_c(x))^\top$ for any $x \in \mathcal{X}$. Next, the entries of $\mathbf{M}$ are relaxed to take real values and learned from the training sample with the objective of making the row of $\mathbf{M}$ corresponding to the class of any point $x \in \mathcal{X}$ more similar to $\mathbf{F}(x)$ than other rows. The similarity can be measured using any PDS kernel K. An example of an algorithm for learning $\mathbf{M}$ using a PDS kernel K and the idea just discussed is in fact multi-class SVMs, which, in this context, can be formulated as follows:

$$\begin{aligned} &\min_{\mathbf{M}, \boldsymbol{\xi}} \|\mathbf{M}\|_F^2 + C \sum_{i=1}^{m} \xi_i \\ &\text{subject to: } \forall (i, l) \in [m] \times \mathcal{Y}, \\ &\qquad K(\mathbf{f}(x_i), \mathbf{M}_{y_i}) \geq K(\mathbf{f}(x_i), \mathbf{M}_l) + 1 - \xi_i. \end{aligned}$$

Similar algorithms can be defined using other matrix norms. The resulting multi-class classification decision function has the following form:

$$h \colon x \mapsto \underset{l \in \{1, \ldots, k\}}{\operatorname{argmax}} K(\mathbf{f}(x), \mathbf{M}_l).$$

9.5 Structured prediction algorithms

In this section, we briefly discuss an important class of problems related to multi-class classification that frequently arises in computer vision, computational biology, and natural language processing. These include all sequence labeling problems and complex problems such as parsing, machine translation, and speech recognition.

In these applications, the output labels have a rich internal structure. For example, in *part-of-speech tagging* the problem consists of assigning a part-of-speech tag such as N (noun), V (verb), or A (adjective), to every word of a sentence. Thus, the label of the sentence $\omega_1 \ldots \omega_n$ made of the words ω_i is a sequence of part-of-speech tags $t_1 \ldots t_n$. This can be viewed as a multi-class classification problem where each sequence of tags is a possible label. However, several critical aspects common to such *structured output* problems make them distinct from the standard multi-class classification.

First, the label set is exponentially large as a function of the size of the output. For example, if Σ denotes the alphabet of part-of-speech tags, for a sentence of length n there are $|\Sigma|^n$ possible tag sequences. Second, there are dependencies between the substructures of a label that are important to take into account for an accurate prediction. For example, in part-of-speech tagging, some tag sequences may be ungrammatical or unlikely. Finally, the loss function used is typically not a zero-one loss, but one that depends on the substructures. Let $L \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ denote a loss function such that $L(y', y)$ measures the penalty of predicting the label $y' \in \mathcal{Y}$ instead of the correct label $y \in \mathcal{Y}$.[15] In part-of-speech tagging, $L(y', y)$ could be for example the Hamming distance between y' and y.

The relevant features in structured output problems often depend on both the input and the output. Thus, we will denote by $\mathbf{\Phi}(x, y) \in \mathbb{R}^N$ the feature vector associated to a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

To model the label structures and their dependency, the label set $\mathcal{Y}$ is typically assumed to be endowed with a *graphical model* structure, that is, a graph giving a probabilistic model of the conditional dependence between the substructures. It is also assumed that both the feature vector $\mathbf{\Phi}(x, y)$ associated to an input $x \in \mathcal{X}$ and output $y \in \mathcal{Y}$ and the loss $L(y', y)$ factorize according to the cliques of that graphical model.[16] A detailed treatment of this topic would require a further background in graphical models, and is thus beyond the scope of this section.

The hypothesis set used by most structured prediction algorithms is then defined as the set of functions $h \colon \mathcal{X} \to \mathcal{Y}$ such that

$$\forall x \in \mathcal{X}, \quad h(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbf{w} \cdot \mathbf{\Phi}(x, y), \tag{9.20}$$

for some vector $\mathbf{w} \in \mathbb{R}^N$. Let $S = ((x_1, y_1), \ldots, x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ be an i.i.d. labeled sample. Since the hypothesis set is linear, we can seek to define an algorithm similar to multi-class SVMs. The optimization problem for multi-class SVMs can be rewritten equivalently as follows:

$$\min_{\mathbf{w}} \; \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \max_{y \neq y_i} \max \Big(0, 1 - \mathbf{w} \cdot [\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)]\Big), \tag{9.21}$$

However, here we need to take into account the loss function L, that is $L(y, y_i)$ for each $i \in [m]$ and $y \in \mathcal{Y}$, and there are multiple ways to proceed. One possible way is to let the margin violation be penalized additively with $L(y, y_i)$. Thus, in that case $L(y, y_i)$ is added to the margin violation. Another natural method consists of penalizing the margin violation by multiplying it with $L(y, y_i)$. A margin violation with a larger loss is then penalized more than one with a smaller loss.

[15] More generally, in some applications, the loss function could also depend on the input. Thus, L is then a function mapping $L \colon \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, with $L(x, y', y)$ measuring the penalty of predicting the label y' instead of y given the input x.

[16] In an undirected graph, a *clique* is a set of fully connected vertices.

The additive penalization leads to the following algorithm known as *Maximum Margin Markov Networks (M³N)*:

$$\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \max_{y \neq y_i} \max\Big(0, L(y_i, y) - \mathbf{w}\cdot[\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)]\Big). \qquad (9.22)$$

An advantage of this algorithm is that, as in the case of SVMs, it admits a natural use of PDS kernels. As already indicated, the label set $\mathcal{Y}$ is assumed to be endowed with a graph structure with a Markov property, typically a chain or a tree, and the loss function is assumed to be decomposable in the same way. Under these assumptions, by exploiting the graphical model structure of the labels, a polynomial-time algorithm can be given to determine its solution.

A multiplicative combination of the loss with the margin leads to the following algorithm known as *SVMStruct*:

$$\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \max_{y \neq y_i} L(y_i, y) \max\Big(0, 1 - \mathbf{w}\cdot[\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)]\Big). \qquad (9.23)$$

This problem can be equivalently written as a QP with an infinite number of constraints. In practice, it is solved iteratively by augmenting at each round the finite set of constraints of the previous round with the most violating constraint. This method can be applied in fact under very general assumptions and for arbitrary loss definitions. As in the case of M³N, SVMStruct naturally admits the use of PDS kernels and thus an extension to non-linear models for the solution.

Another standard algorithm for structured prediction problems is *Conditional Random Fields (CRFs)*. We will not describe this algorithm in detail, but point out its similarity with the algorithms just described, in particular M³N. The optimization problem for CRFs can be written as

$$\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \log \sum_{y \in \mathcal{Y}} \exp\Big(L(y_i, y) - \mathbf{w}\cdot[\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)]\Big). \qquad (9.24)$$

Assume for simplicity that $\mathcal{Y}$ is finite and has cardinality k and let f denote the function $(x_1, \ldots, x_k) \mapsto \log(\sum_{j=1}^{k} e^{x_j})$. f is a convex function known as the *soft-max*, since it provides a smooth approximation of $(x_1, \ldots, x_k) \mapsto \max(x_1, \ldots, x_k)$. Then, problem (9.24) is similar to (9.22) modulo the replacement of the max operator with the soft-max function just described.

9.6 Chapter notes

The margin-based generalization bound for multi-class classification presented in theorem 9.2 is due to Kuznetsov, Mohri, and Syed [2014]. It admits only a linear dependency on the number of classes. This improves over a similar result by Koltchinskii and Panchenko [2002], which admits a quadratic dependency on the number of classes. Proposition 9.3 bounding the Rademacher complexity of multi-class kernel-based hypotheses and corollary 9.4 are new.

An algorithm generalizing SVMs to the multi-class classification setting was first introduced by Weston and Watkins [1999]. The optimization problem for that algorithm was based on $k(k-1)/2$ slack variables for a problem with k classes and thus could be inefficient for a relatively large number of classes. A simplification of that algorithm by replacing the sum of the slack variables $\sum_{j \neq i} \xi_{ij}$ related to point x_i by its maximum $\xi_i = \max_{j \neq i} \xi_{ij}$ considerably reduces the number of variables and leads to the multi-class SVM algorithm presented in this chapter [Crammer and Singer, 2001, 2002].

The AdaBoost.MH algorithm is presented and discussed by Schapire and Singer [1999, 2000]. As we showed in this chapter, the algorithm is a special instance of AdaBoost. Another boosting-type algorithm for multi-class classification, AdaBoost.MR, is presented by Schapire and Singer [1999, 2000]. That algorithm is also a special instance of the RankBoost algorithm presented in chapter 10. See exercise 10.5 for a detailed analysis of this algorithm, including generalization bounds.

The most commonly used tools for learning decision trees are CART (classification and regression tree) [Breiman et al., 1984] and C4.5 [Quinlan, 1986, 1993]. The greedy technique we described for learning decision trees benefits in fact from an interesting analysis: remarkably, it has been shown by Kearns and Mansour [1999], Mansour and McAllester [1999] that, under a weak learner hypothesis assumption, such decision tree algorithms produce a strong hypothesis. The grow-then-prune method is from CART. It has been analyzed by a variety of different studies, in particular by Kearns and Mansour [1998] and Mansour and McAllester [2000], who give generalization bounds for the resulting decision trees with respect to the error and size of the best sub-tree of the original tree pruned. Hardness of ERM for decision trees of a fixed size was shown by Grigni et al. [2000].

The idea of the ECOC framework for multi-class classification is due to Dietterich and Bakiri [1995]. Allwein et al. [2000] further extended and analyzed this method to margin-based losses, for which they presented a bound on the empirical error and a generalization bound in the more specific case of boosting. While the OVA technique is in general subject to a calibration issue and does not have any justification, it is very commonly used in practice. Rifkin [2002] reports the results of extensive experiments with several multi-class classification algorithms that are rather favorable to the OVA technique, with performances often very close or better than for those of several uncombined algorithms, unlike what has been claimed by some authors (see also Rifkin and Klautau [2004]).

The CRFs algorithm was introduced by Lafferty, McCallum, and Pereira [2001]. M^{3}N is due to Taskar, Guestrin, and Koller [2003] and StructSVM was presented by Tsochantaridis, Joachims, Hofmann, and Altun [2005]. An alternative technique for tackling structured prediction as a regression problem was presented and analyzed by Cortes, Mohri, and Weston [2007c].

9.7 Exercises

9.1 Generalization bounds for multi-label case. Use similar techniques to those used in the proof of theorem 9.2 to derive a margin-based learning bound in the multi-label case.

9.2 Multi-class classification with kernel-based hypotheses constrained by an L_p norm. Use corollary 9.4 to define alternative multi-class classification algorithms with kernel-based hypotheses constrained by an L_p norm with $p \neq 2$. For which value of $p \geq 1$ is the bound of proposition 9.3 tightest? Derive the dual optimization of the multi-class classification algorithm defined with $p = \infty$.

9.3 Alternative multi-class boosting algorithm. Consider the objective function G defined for any sample $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$, $n \geq 1$, by

$$G(\boldsymbol{\alpha}) = \sum_{i=1}^{m} e^{-\frac{1}{k}\sum_{l=1}^{k} y_i[l] f_n(x_i, l)} = \sum_{i=1}^{m} e^{-\frac{1}{k}\sum_{l=1}^{k} y_i[l] \sum_{t=1}^{n} \alpha_t h_t(x_i, l)}. \tag{9.25}$$

Use the convexity of the exponential function to compare G with the objective function F defining AdaBoost.MH. Show that G is a convex function upper bounding the multi-label multi-class error. Discuss the properties of G and derive an algorithm defined by the application of coordinate descent to G. Give theoretical guarantees for the performance of the algorithm and analyze its running-time complexity when using boosting stumps.

9.4 Multi-class algorithm based on RankBoost. This problem requires familiarity with the material presented both in this chapter and in chapter 10. An alternative boosting-type multi-class classification algorithm is one based on a ranking criterion. We will define and examine that algorithm in the mono-label setting. Let $\mathcal{H}$ be a family of base hypotheses mapping $\mathcal{X} \times \mathcal{Y}$ to $\{-1, +1\}$. Let F be the following objective function defined for all samples $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ and $\bar{\boldsymbol{\alpha}} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_N) \in \mathbb{R}^N$, $N \geq 1$, by

$$F(\bar{\boldsymbol{\alpha}}) = \sum_{i=1}^{m} \sum_{l \neq y_i} e^{-(f_N(x_i, y_i) - f_N(x_i, l))} = \sum_{i=1}^{m} \sum_{l \neq y_i} e^{-\sum_{j=1}^{N} \bar{\alpha}_j (h_j(x_i, y_i) - h_j(x_i, l))}. \tag{9.26}$$

where $f_N = \sum_{j=1}^{N} \bar{\alpha}_j h_j$.

(a) Show that F is convex and differentiable.

(b) Show that $\frac{1}{m} \sum_{i=1}^{m} 1_{\rho_{f_N}(x_i, y_i)} \leq \frac{1}{k-1} F(\bar{\boldsymbol{\alpha}})$, where $f_N = \sum_{j=1}^{N} \bar{\alpha}_j h_j$.

(c) Give the pseudocode of the algorithm obtained by applying coordinate descent to F. The resulting algorithm is known as AdaBoost.MR. Show that AdaBoost.MR exactly coincides with the RankBoost algorithm applied to the problem of ranking pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Describe exactly the ranking target for these pairs.

(d) Use question (9.4b) and the learning bounds of this chapter to derive margin-based generalization bounds for this algorithm.

(e) Use the connection of the algorithm with RankBoost and the learning bounds of chapter 10 to derive alternative generalization bounds for this algorithm. Compare these bounds with those of the previous question.

9.5 Decision trees. Show that VC-dimension of a binary decision tree with n nodes in dimension N is in $O(n \log N)$.

9.6 Give an example where the generalization error of each of the $k(k-1)/2$ binary classifiers $h_{ll'}$, $l \neq l'$, used in the definition of the OVO technique is r and that of the OVO hypothesis $(k-1)r$.

10 Ranking

The learning problem of ranking arises in many modern applications, including the design of search engines, information extraction platforms, and movie recommendation systems. In these applications, the ordering of the documents or movies returned is a critical aspect of the system. The main motivation for ranking over classification in the binary case is the limitation of resources: for very large data sets, it may be impractical or even impossible to display or process all items labeled as relevant by a classifier. A standard user of a search engine is not willing to consult all the documents returned in response to a query, but only the top ten or so. Similarly, a member of the fraud detection department of a credit card company cannot investigate thousands of transactions classified as potentially fraudulent, but only a few dozens of the most suspicious ones.

In this chapter, we study in depth the learning problem of ranking. We distinguish two general settings for this problem: the score-based and the preference-based settings. For the score-based setting, which is the most widely explored one, we present margin-based generalization bounds using the notion of Rademacher complexity. We then describe an SVM-based ranking algorithm that can be derived from these bounds and describe and analyze RankBoost, a boosting algorithm for ranking. We further study specifically the bipartite setting of the ranking problem where, as in binary classification, each point belongs to one of two classes. We discuss an efficient implementation of RankBoost in that setting and point out its connections with AdaBoost. We also introduce the notions of ROC curves and area under the ROC curve (AUC) which are directly relevant to bipartite ranking. For the preference-based setting, we present a series of results, in particular regret-based guarantees for both a deterministic and a randomized algorithm, as well as a lower bound in the deterministic case.

10.1 The problem of ranking

We first introduce the most commonly studied scenario of the ranking problem in machine learning. We will refer to this scenario as the *score-based setting* of the ranking problem. In section 10.6, we present and analyze an alternative setting, the *preference-based setting*.

The general supervised learning problem of ranking consists of using labeled information to define an accurate ranking prediction function for all points. In the scenario examined here, the labeled information is supplied only for pairs of points and the quality of a predictor is similarly measured in terms of its average pairwise misranking. The predictor is a real-valued function, a *scoring function*: the scores assigned to input points by this function determine their ranking.

Let $\mathcal{X}$ denote the input space. We denote by $\mathcal{D}$ an unknown distribution over $\mathcal{X} \times \mathcal{X}$ according to which pairs of points are drawn and by $f : \mathcal{X} \times \mathcal{X} \to \{-1, 0, +1\}$ a target labeling function or *preference function*. The three values assigned by f are interpreted as follows: $f(x, x') = +1$ if x' is preferred to x or ranked higher than x, $f(x, x') = -1$ if x is preferred to x', and $f(x, x') = 0$ if both x and x' have the same preference or ranking, or if there is no information about their respective ranking. This formulation corresponds to a deterministic scenario which we adopt for simplification. As discussed in section 2.4.1, it can be straightforwardly extended to a stochastic scenario where we have a distribution over $\mathcal{X} \times \mathcal{X} \times \{-1, 0, +1\}$.

Note that in general no particular assumption is made about the transitivity of the order induced by f: we may have $f(x, x') = 1$ and $f(x', x'') = 1$ but $f(x, x'') = -1$ for three points x, x', and x''. While this may contradict an intuitive notion of preference, such preference orders are in fact commonly encountered in practice, in particular when they are based on human judgments. This is sometimes because the preference between two items are decided based on different features: for example, an individual may prefer movie x' to x because x' is an action movie and x a musical, and prefer x'' to x' because x'' is an action movie with more special effects than x'. Nevertheless, they may prefer x to x'' because the cost of the movie ticket for x'' is signicantly higher. Thus, in this example, two features, the genre and the price, are invoked, each affecting the decision for different pairs. In fact, in general, no assumption is made about the preference function, not even the antisymmetry of the order induced; thus, we may have $f(x, x') = 1$ and $f(x', x) = 1$ and yet $x \neq x'$.

The learner receives a labeled sample $S = \big((x_1, x'_1, y_1), \ldots, (x_m, x'_m, y_m)\big) \in \mathcal{X} \times \mathcal{X} \times \{-1, 0, +1\}$ with $(x_1, x'_1), \ldots, (x_m, x'_m)$ drawn i.i.d. according to $\mathcal{D}$ and $y_i = f(x_i, x'_i)$ for all $i \in [m]$. Given a hypothesis set $\mathcal{H}$ of functions mapping $\mathcal{X}$ to $\mathbb{R}$, the ranking problem consists of selecting a hypothesis $h \in \mathcal{H}$ with small expected pairwise misranking or generalization error $R(h)$ with respect to the target f:

$$R(h) = \underset{(x,x')\sim\mathcal{D}}{\mathbb{P}} \Big[\big(f(x,x') \neq 0\big) \wedge \big(f(x,x')(h(x') - h(x)) \leq 0\big)\Big]. \tag{10.1}$$

The empirical pairwise misranking or empirical error of h is denoted by $\widehat{R}_S(h)$ and defined by

$$\widehat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m} 1_{(y_i \neq 0)\wedge(y_i(h(x'_i)-h(x_i))\leq 0)}\,. \tag{10.2}$$

Note that while the target preference function f is in general not transitive, the linear ordering induced by a scoring function $h \in \mathcal{H}$ is by definition transitive. This is a drawback of the score-based setting for the ranking problem since, regardless of the complexity of the hypothesis set $\mathcal{H}$, if the preference function is not transitive, no hypothesis $h \in \mathcal{H}$ can faultlessly predict the target pairwise ranking.

10.2 Generalization bound

In this section, we present margin-based generalization bounds for ranking. To simplify the presentation, we will assume for the results of this section that the pairwise labels are in $\{-1, +1\}$. Thus, if a pair (x, x') is drawn according to $\mathcal{D}$, then either x is preferred to x' or the opposite. The learning bounds for the general case have a very similar form but require more details. As in the case of classification, for any $\rho > 0$, we can define the empirical margin loss of a hypothesis h for pairwise ranking as

$$\widehat{R}_{S,\rho}(h) = \frac{1}{m}\sum_{i=1}^{m} \Phi_\rho(y_i(h(x'_i) - h(x_i))), \tag{10.3}$$

where Φ_ρ is the margin loss function (definition 5.5). Thus, the empirical margin loss for ranking is upper bounded by the fraction of the pairs (x_i, x'_i) that h is misranking or correctly ranking but with confidence less than ρ:

$$\widehat{R}_{S,\rho}(h) \leq \frac{1}{m}\sum_{i=1}^{m} 1_{y_i(h(x'_i)-h(x_i))\leq\rho}. \tag{10.4}$$

We denote by $\mathcal{D}_1$ the marginal distribution of the first element of the pairs in $\mathcal{X}\times\mathcal{X}$ derived from $\mathcal{D}$, and by $\mathcal{D}_2$ the marginal distribution with respect to the second element of the pairs. Similarly, S_1 is the sample derived from S by keeping only the first element of each pair: $S_1 = \big((x_1, y_1), \ldots, (x_m, y_m)\big)$ and S_2 the one obtained by keeping only the second element: $S_2 = \big((x'_1, y_1), \ldots, (x'_m, y_m)\big)$. We also denote by $\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H})$ the Rademacher complexity of $\mathcal{H}$ with respect to the marginal distribution $\mathcal{D}_1$, that is $\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H}) = \mathbb{E}[\widehat{\mathfrak{R}}_{S_1}(\mathcal{H})]$, and similarly $\mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H}) = \mathbb{E}[\widehat{\mathfrak{R}}_{S_2}(\mathcal{H})]$. Clearly, if the distribution $\mathcal{D}$ is symmetric, the marginal distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ coincide and $\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H}) = \mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H})$.

Theorem 10.1 (Margin bound for ranking) *Let $\mathcal{H}$ be a set of real-valued functions. Fix $\rho > 0$; then, for any $\delta > 0$, with probability at least $1-\delta$ over the choice of a sample S of size m, each of the following holds for all $h \in \mathcal{H}$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\big(\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H}) + \mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H})\big) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \tag{10.5}$$

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\big(\widehat{\mathfrak{R}}_{S_1}(\mathcal{H}) + \widehat{\mathfrak{R}}_{S_2}(\mathcal{H})\big) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}\,. \tag{10.6}$$

Proof: The proof is similar to that of theorem 5.8. Let $\widetilde{\mathcal{H}}$ be the family of hypotheses mapping $(\mathcal{X} \times \mathcal{X}) \times \{-1, +1\}$ to $\mathbb{R}$ defined by $\widetilde{\mathcal{H}} = \{z = ((x, x'), y) \mapsto y[h(x') - h(x)] \colon h \in \mathcal{H}\}$. Consider the family of functions $\widetilde{\mathcal{H}} = \{\Phi_\rho \circ f \colon f \in \widetilde{\mathcal{H}}\}$ derived from $\widetilde{\mathcal{H}}$ which are taking values in $[0, 1]$. By theorem 3.3, for any $\delta > 0$ with probability at least $1 - \delta$, for all $h \in \mathcal{H}$,

$$\mathbb{E}\left[\Phi_\rho(y[h(x') - h(x)])\right] \leq \widehat{R}_{S,\rho}(h) + 2\mathfrak{R}_m\big(\Phi_\rho \circ \widetilde{\mathcal{H}}\big) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}\,.$$

Since $1_{u \leq 0} \leq \Phi_\rho(u)$ for all $u \in \mathbb{R}$, the generalization error $R(h)$ is a lower bound on left-hand side, $R(h) = \mathbb{E}[1_{y[h(x') - h(x)] \leq 0}] \leq \mathbb{E}\left[\Phi_\rho(y[h(x') - h(x)])\right]$, and we can write:

$$R(h) \leq \widehat{R}_{S,\rho}(h) + 2\mathfrak{R}_m\big(\Phi_\rho \circ \widetilde{\mathcal{H}}\big) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}\,.$$

Since Φ_ρ is a $(1/\rho)$-Lipschitz function, by Talagrand's lemma (lemma 5.7) we have that $\mathfrak{R}_m\big(\Phi_\rho \circ \widetilde{\mathcal{H}}\big) \leq \frac{1}{\rho}\mathfrak{R}_m(\widetilde{\mathcal{H}})$. Here, $\mathfrak{R}_m(\widetilde{\mathcal{H}})$ can be upper bounded as follows:

$$\begin{aligned}
\mathfrak{R}_m(\widetilde{\mathcal{H}}) &= \frac{1}{m} \mathop{\mathbb{E}}_{S,\sigma}\Big[\sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i y_i (h(x'_i) - h(x_i))\Big] \\
&= \frac{1}{m} \mathop{\mathbb{E}}_{S,\sigma}\Big[\sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i (h(x'_i) - h(x_i))\Big] && (y_i\sigma_i \text{ and } \sigma_i\text{: same distrib.}) \\
&\leq \frac{1}{m} \mathop{\mathbb{E}}_{S,\sigma}\Big[\sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i h(x'_i) + \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i h(x_i)\Big] && (\text{by sub-additivity of sup}) \\
&= \mathop{\mathbb{E}}_{S}\Big[\mathfrak{R}_{S_2}(\mathcal{H}) + \mathfrak{R}_{S_1}(\mathcal{H})\Big] && (\text{definition of } S_1 \text{ and } S_2) \\
&= \mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H}) + \mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H})\,,
\end{aligned}$$

which proves (10.5). The second inequality, (10.6), can be derived in the same way by using the second inequality of theorem 3.3, (3.4), instead of (3.3). □

These bounds can be generalized to hold uniformly for all $\rho > 0$ at the cost of an additional term $\sqrt{(\log\log_2(2/\rho))/m}$, as in theorem 5.9 and exercise 5.2. As for other margin bounds presented in previous sections, they show the conflict between two terms: the larger the desired pairwise ranking margin ρ, the smaller the middle term. However, the first term, the empirical pairwise ranking margin loss $\widehat{R}_{S,\rho}$, increases as a function of ρ.

Known upper bounds for the Rademacher complexity of a hypothesis set $\mathcal{H}$, including bounds in terms of VC-dimension, can be used directly to make theorem 10.1 more explicit. In particular, using theorem 10.1, we obtain immediately the following margin bound for pairwise ranking using kernel-based hypotheses.

Corollary 10.2 (Margin bounds for ranking with kernel-based hypotheses) *Let $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel with $r = \sup_{x\in\mathcal{X}} K(x,x)$. Let $\Phi\colon \mathcal{X} \to \mathbb{H}$ be a feature mapping associated to K and let $\mathcal{H} = \{x \mapsto \mathbf{w}\cdot\mathbf{\Phi}(x)\colon \|\mathbf{w}\|_{\mathbb{H}} \leq \Lambda\}$ for some $\Lambda \geq 0$. Fix $\rho > 0$. Then, for any $\delta > 0$, the following pairwise margin bound holds with probability at least $1-\delta$ for any $h \in \mathcal{H}$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + 4\sqrt{\frac{r^2\Lambda^2/\rho^2}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}. \tag{10.7}$$

As with theorem 5.8, the bound of this corollary can be generalized to hold uniformly for all $\rho > 0$ at the cost of an additional term $\sqrt{(\log\log_2(2/\rho))/m}$. This generalization bound for kernel-based hypotheses is remarkable, since it does not depend directly on the dimension of the feature space, but only on the pairwise ranking margin. It suggests that a small generalization error can be achieved when ρ/r is large (small second term) while the empirical margin loss is relatively small (first term). The latter occurs when few points are either classified incorrectly or correctly but with margin less than ρ.

10.3 Ranking with SVMs

In this section, we discuss an algorithm that is derived directly from the theoretical guarantees just presented. The algorithm turns out to be a special instance of the SVM algorithm.

Proceeding as in section 5.4 for classification, the guarantee of corollary 10.2 can be expressed as follows: for any $\delta > 0$, with probability at least $1-\delta$, for all $h \in \mathcal{H} = \{x \mapsto \mathbf{w}\cdot\mathbf{\Phi}(x)\colon \|\mathbf{w}\| \leq \Lambda\}$,

$$R(h) \leq \frac{1}{m}\sum_{i=1}^{m}\xi_i + 4\sqrt{\frac{r^2\Lambda^2}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}\,, \tag{10.8}$$

where $\xi_i = \max\big(1 - y_i\big[\mathbf{w}\cdot\big(\mathbf{\Phi}(x_i') - \mathbf{\Phi}(x_i)\big)\big], 0\big)$ for all $i \in [m]$, and where $\mathbf{\Phi}\colon \mathcal{X} \to \mathbb{H}$ is a feature mapping associated to a PDS kernel K. An algorithm based on this theoretical guarantee consists of minimizing the right-hand side of (10.8), that is minimizing an objective function with a term corresponding to the sum of the slack variables ξ_i, and another one minimizing $\|\mathbf{w}\|$ or equivalently $\|\mathbf{w}\|^2$. Its optimization problem can thus be formulated as

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i \tag{10.9}$$

$$\text{subject to: } y_i\Big[\mathbf{w}\cdot\big(\mathbf{\Phi}(x'_i) - \mathbf{\Phi}(x_i)\big)\Big] \geq 1 - \xi_i$$

$$\xi_i \geq 0, \quad \forall i \in [m]\,.$$

This coincides exactly with the primal optimization problem of SVMs, with a feature mapping $\mathbf{\Psi}\colon \mathcal{X}\times\mathcal{X}\to\mathbb{H}$ defined by $\mathbf{\Psi}(x,x') = \mathbf{\Phi}(x') - \mathbf{\Phi}(x)$ for all $(x,x')\in\mathcal{X}\times\mathcal{X}$, and with a hypothesis set of functions of the form $(x,x')\mapsto \mathbf{w}\cdot\mathbf{\Psi}(x,x')$. Thus, clearly, all the properties already presented for SVMs apply in this instance. In particular, the algorithm can benefit from the use of PDS kernels. Problem (10.9) admits an equivalent dual that can be expressed in terms of the kernel matrix $\mathbf{K}'$ defined by

$$\mathbf{K}'_{ij} = \mathbf{\Psi}(x_i,x'_i)\cdot\mathbf{\Psi}(x_j,x'_j) = K(x_i,x_j)+K(x'_i,x'_j)-K(x'_i,x_j)-K(x_i,x'_j), \tag{10.10}$$

for all $i,j\in[m]$. This algorithm can provide an effective solution for pairwise ranking in practice. The algorithm can also be used and extended to the case where the labels are in $\{-1,0,+1\}$. The next section presents an alternative algorithm for ranking in the score-based setting.

10.4 RankBoost

This section presents a boosting algorithm for pairwise ranking, *RankBoost*, similar to the AdaBoost algorithm for binary classification. RankBoost is based on ideas analogous to those discussed for classification: it consists of combining different *base rankers* to create a more accurate predictor. The base rankers are hypotheses returned by a *weak learning algorithm* for ranking. As for classification, these base hypotheses must satisfy a minimal accuracy condition that will be described precisely later.

Let $\mathcal{H}$ denote the hypothesis set from which the base rankers are selected. Algorithm 10.1 gives the pseudocode of the RankBoost algorithm when $\mathcal{H}$ is a set of functions mapping from $\mathcal{X}$ to $\{0,1\}$. For any $s\in\{-1,0,+1\}$, we define ϵ_t^s by

$$\epsilon_t^s = \sum_{i=1}^{m}\mathcal{D}_t(i)1_{y_i(h_t(x'_i)-h_t(x_i))=s} = \mathop{\mathbb{E}}_{i\sim\mathcal{D}_t}\big[1_{y_i(h_t(x'_i)-h_t(x_i))=s}\big], \tag{10.11}$$

```
RankBoost($S = ((x_1, x'_1, y_1) \ldots, (x_m, x'_m, y_m))$)
 1  for $i \leftarrow 1$ to $m$ do
 2      $\mathcal{D}_1(i) \leftarrow \frac{1}{m}$
 3  for $t \leftarrow 1$ to $T$ do
 4      $h_t \leftarrow$ base ranker in $\mathcal{H}$ with smallest $\epsilon_t^- - \epsilon_t^+ = -\mathop{\mathbb{E}}_{i\sim\mathcal{D}_t}\Big[y_i\big(h_t(x'_i) - h_t(x_i)\big)\Big]$
 5      $\alpha_t \leftarrow \frac{1}{2}\log\frac{\epsilon_t^+}{\epsilon_t^-}$
 6      $Z_t \leftarrow \epsilon_t^0 + 2[\epsilon_t^+\epsilon_t^-]^{\frac{1}{2}}$    ▷ normalization factor
 7      for $i \leftarrow 1$ to $m$ do
 8          $\mathcal{D}_{t+1}(i) \leftarrow \frac{\mathcal{D}_t(i)\exp\big[-\alpha_t y_i\big(h_t(x'_i)-h_t(x_i)\big)\big]}{Z_t}$
 9  $f \leftarrow \sum_{t=1}^T \alpha_t h_t$
10  return $f$
```

Figure 10.1
RankBoost algorithm for $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$.

and simplify the notation ϵ_t^{+1} into ϵ_t^+ and similarly write ϵ_t^- instead of ϵ_t^{-1}. With these definitions, clearly the following equality holds: $\epsilon_t^0 + \epsilon_t^+ + \epsilon_t^- = 1$.

The algorithm takes as input a labeled sample $S = \big((x_1, x'_1, y_1), \ldots, (x_m, x'_m, y_m)\big)$ with elements in $\mathcal{X} \times \mathcal{X} \times \{-1, 0, +1\}$, and maintains a distribution over the subset of the indices $i \in \{1, \ldots, m\}$ for which $y_i \neq 0$. To simplify the presentation, we will assume that $y_i \neq 0$ for all $i \in \{1, \ldots, m\}$ and consider distributions defined over $\{1, \ldots, m\}$. This can be guaranteed by simply first removing from the sample the pairs labeled with zero.

Initially (lines 1–2), the distribution is uniform ($\mathcal{D}_1$). At each round of boosting, that is at each iteration $t \in [T]$ of the loop 3–8, a new base ranker $h_t \in \mathcal{H}$ is selected with the smallest difference $\epsilon_t^- - \epsilon_t^+$, that is one with the smallest pairwise misranking error and largest correct pairwise ranking accuracy for the distribution $\mathcal{D}_t$:

$$h_t \in \operatorname*{argmin}_{h\in\mathcal{H}}\Big\{-\mathop{\mathbb{E}}_{i\sim\mathcal{D}_t}\Big[y_i\big(h(x'_i) - h(x_i)\big)\Big]\Big\}.$$

Note that $\epsilon_t^- - \epsilon_t^+ = \epsilon_t^- - (1 - \epsilon_t^- - \epsilon_t^0) = 2\epsilon_t^- + \epsilon_t^0 - 1$. Thus, finding the smallest difference $\epsilon_t^- - \epsilon_t^+$ is equivalent to seeking the smallest $2\epsilon_t^- + \epsilon_t^0$, which itself coincides with seeking the smallest ϵ_t^- when $\epsilon_t^0 = 0$. Z_t is simply a normalization factor to ensure that the weights $\mathcal{D}_{t+1}(i)$ sum to one. RankBoost relies on the assumption that at each round $t \in [T]$, for the hypothesis h_t found, the inequality $\epsilon_t^+ - \epsilon_t^- > 0$ holds; thus, the probability mass of the pairs correctly ranked by h_t (ignoring pairs with label zero) is larger than that of misranked pairs. We denote by γ_t the *edge* of the base ranker h_t: $\gamma_t = \frac{\epsilon_t^+ - \epsilon_t^-}{2}$.

The precise reason for the definition of the coefficient α_t (line 5) will become clear later. For now, observe that if $\epsilon_t^+ - \epsilon_t^- > 0$, then $\epsilon_t^+ / \epsilon_t^- > 1$ and $\alpha_t > 0$. Thus, the new distribution $\mathcal{D}_{t+1}$ is defined from $\mathcal{D}_t$ by increasing the weight on i if the pair (x_i, x_i') is misranked $(y_i(h_t(x_i') - h_t(x_i) < 0)$, and, on the contrary, decreasing it if (x_i, x_i') is ranked correctly $(y_i(h_t(x_i') - h_t(x_i) > 0)$. The relative weight is unchanged for a pair with $h_t(x_i') - h_t(x_i) = 0$. This distribution update has the effect of focusing more on misranked points at the next round of boosting.

After T rounds of boosting, the hypothesis returned by RankBoost is f, which is a linear combination of the base classifiers h_t. The weight α_t assigned to h_t in that sum is a logarithmic function of the ratio of ϵ_t^+ and ϵ_t^-. Thus, more accurate base rankers are assigned a larger weight in that sum.

For any $t \in [T]$, we will denote by f_t the linear combination of the base rankers after t rounds of boosting: $f_t = \sum_{s=1}^t \alpha_t h_t$. In particular, we have $f_T = f$. The distribution $\mathcal{D}_{t+1}$ can be expressed in terms of f_t and the normalization factors Z_s, $s \in [t]$, as follows:

$$\forall i \in [m], \quad \mathcal{D}_{t+1}(i) = \frac{e^{-y_i(f_t(x_i')) - f_t(x_i)}}{m \prod_{s=1}^t Z_s}. \tag{10.12}$$

We will make use of this identity several times in the proofs of the following sections. It can be shown straightforwardly by repeatedly expanding the definition of the distribution over the point x_i:

$$\begin{aligned}
\mathcal{D}_{t+1}(i) &= \frac{\mathcal{D}_t(i) e^{-\alpha_t y_i(h_t(x_i') - h_t(x_i))}}{Z_t} \\
&= \frac{\mathcal{D}_{t-1}(i) e^{-\alpha_{t-1} y_i(h_{t-1}(x_i') - h_{t-1}(x_i))} e^{-\alpha_t y_i(h_t(x_i') - h_t(x_i))}}{Z_{t-1} Z_t} \\
&= \frac{e^{-y_i \sum_{s=1}^t \alpha_s(h_s(x_i') - h_s(x_i))}}{m \prod_{s=1}^t Z_s}.
\end{aligned}$$

10.4.1 Bound on the empirical error

We first show that the empirical error of RankBoost decreases exponentially fast as a function of the number of rounds of boosting when the edge γ_t of each base ranker h_t is lower bounded by some positive value $\gamma > 0$.

Theorem 10.3 *The empirical error of the hypothesis $h\colon \mathcal{X} \to \{0,1\}$ returned by RankBoost verifies:*

$$\widehat{R}_S(h) \leq \exp\Big[-2\sum_{t=1}^{T}\Big(\frac{\epsilon_t^+ - \epsilon_t^-}{2}\Big)^2\Big]. \tag{10.13}$$

Furthermore, if there exists γ such that for all $t \in [T]$, $0 < \gamma \leq \frac{\epsilon_t^+ - \epsilon_t^-}{2}$, then

$$\widehat{R}_S(h) \leq \exp(-2\gamma^2 T)\,. \tag{10.14}$$

Proof: Using the general inequality $1_{u \leq 0} \leq \exp(-u)$ valid for all $u \in \mathbb{R}$ and identity 10.12, we can write:

$$\begin{aligned}\widehat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m} 1_{y_i(f(x'_i)-f(x_i))\leq 0} &\leq \frac{1}{m}\sum_{i=1}^{m} e^{-y_i(f(x'_i)-f(x_i))}\\ &\leq \frac{1}{m}\sum_{i=1}^{m}\Big[m\prod_{t=1}^{T} Z_t\Big]\mathcal{D}_{T+1}(i) = \prod_{t=1}^{T} Z_t.\end{aligned}$$

By the definition of normalization factor, for all $t \in [T]$, we have $Z_t = \sum_{i=1}^m \mathcal{D}_t(i)\, e^{-\alpha_t y_i(h_t(x'_i)-h_t(x_i))}$. By grouping together the indices i for which $y_i(h_t(x'_i)-h_t(x_i))$ takes the values in $+1$, -1, or 0, Z_t can be rewritten as

$$Z_t = \epsilon_t^+ e^{-\alpha_t} + \epsilon_t^- e^{\alpha_t} + \epsilon_t^0 = \epsilon_t^+\sqrt{\frac{\epsilon_t^-}{\epsilon_t^+}} + \epsilon_t^-\sqrt{\frac{\epsilon_t^+}{\epsilon_t^-}} + \epsilon_t^0 = 2\sqrt{\epsilon_t^+\epsilon_t^-} + \epsilon_t^0\,.$$

Since $\epsilon_t^+ = 1 - \epsilon_t^- - \epsilon_t^0$, we have

$$4\epsilon_t^+\epsilon_t^- = (\epsilon_t^+ + \epsilon_t^-)^2 - (\epsilon_t^+ - \epsilon_t^-)^2 = (1-\epsilon_t^0)^2 - (\epsilon_t^+ - \epsilon_t^-)^2.$$

Thus, assuming that $\epsilon_t^0 < 1$, Z_t can be upper bounded as follows:

$$\begin{aligned} Z_t &= \sqrt{(1-\epsilon_t^0)^2 - (\epsilon_t^+ - \epsilon_t^-)^2} + \epsilon_t^0\\ &= (1-\epsilon_t^0)\sqrt{1 - \frac{(\epsilon_t^+ - \epsilon_t^-)^2}{(1-\epsilon_t^0)^2}} + \epsilon_t^0\\ &\leq \sqrt{1 - \frac{(\epsilon_t^+ - \epsilon_t^-)^2}{(1-\epsilon_t^0)}}\\ &\leq \exp\left(-\frac{(\epsilon_t^+ - \epsilon_t^-)^2}{2(1-\epsilon_t^0)}\right) \leq \exp\left(-\frac{(\epsilon_t^+ - \epsilon_t^-)^2}{2}\right) = \exp\left(-2[(\epsilon_t^+ - \epsilon_t^-)/2]^2\right),\end{aligned}$$

where we used for the first inequality the concavity of the square-root function and the fact that $0 < 1 - \epsilon_t^0 \leq 1$, and the inequality $1 - x \leq e^{-x}$ valid for all $x \in \mathbb{R}$ for the second inequality. This upper bound on Z_t also trivially holds when $\epsilon_t^0 = 1$ since in that case $\epsilon_t^+ = \epsilon_t^- = 0$. This concludes the proof. □

As can be seen from the proof of the theorem, the weak ranking assumption $\gamma \leq \frac{\epsilon_t^+ - \epsilon_t^-}{2}$ with $\gamma > 0$ can be replaced with the somewhat weaker requirement $\gamma \leq \frac{\epsilon_t^+ - \epsilon_t^-}{2\sqrt{1-\epsilon_t^0}}$, with $\epsilon_t^0 \neq 1$, which can be rewritten as $\gamma \leq \frac{1}{2}\frac{\epsilon_t^+ - \epsilon_t^-}{\sqrt{\epsilon_t^+ + \epsilon_t^-}}$, with $\epsilon_t^+ + \epsilon_t^- \neq 0$, where the quantity $\frac{\epsilon_t^+ - \epsilon_t^-}{\sqrt{\epsilon_t^+ + \epsilon_t^-}}$ can be interpreted as a (normalized) relative difference between ϵ_t^+ and ϵ_t^-.

The proof of the theorem also shows that the coefficient α_t is selected to minimize Z_t. Thus, overall, these coefficients are chosen to minimize the upper bound on the empirical error $\prod_{t=1}^T Z_t$, as for AdaBoost. The RankBoost algorithm can be generalized in several ways:

- Instead of a hypothesis with minimal difference $\epsilon_t^- - \epsilon_t^+$, h_t can be more generally a base ranker returned by a weak ranking algorithm trained on $\mathcal{D}_t$ with $\epsilon_t^+ > \epsilon_t^-$;
- The range of the base rankers could be $[0, +1]$, or more generally $\mathbb{R}$. The coefficients α_t can then be different and may not even admit a closed form. However, in general, they are chosen to minimize the upper bound $\prod_{t=1}^T Z_t$ on the empirical error.

10.4.2 Relationship with coordinate descent

RankBoost coincides with the application of the coordinate descent technique to a convex and differentiable objective function F defined for all samples $S = \big((x_1, x_1', y_1), \ldots, (x_m, x_m', y_m)\big) \in \mathcal{X} \times \mathcal{X} \times \{-1, 0, +1\}$ and $\bar{\boldsymbol{\alpha}} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_n) \in \mathbb{R}^N$, $N \geq 1$ by

$$F(\bar{\boldsymbol{\alpha}}) = \sum_{i=1}^m e^{-y_i[f_N(x_i') - f_N(x_i)]} = \sum_{i=1}^m e^{-y_i \sum_{j=1}^N \bar{\alpha}_j [h_j(x_i') - h_j(x_i)]}, \tag{10.15}$$

where $f_N = \sum_{j=1}^N \bar{\alpha}_j h_j$. This loss function is a convex upper bound on the zero-one pairwise loss function $\bar{\boldsymbol{\alpha}} \mapsto \sum_{i=1}^m 1_{y_i[f_N(x_i') - f_N(x_i)] \leq 0}$, which is not convex. Let $\mathbf{e}_k$ denote the unit vector corresponding to the kth coordinate in $\mathbb{R}^N$ and let $\bar{\boldsymbol{\alpha}}_{t-1} = \bar{\boldsymbol{\alpha}}_t + \eta \mathbf{e}_k$ denote the parameter vector after iteration t (with $\bar{\boldsymbol{\alpha}}_0 = \mathbf{0}$). Also, for any $t \in [T]$, we define the function $\bar{f}_t = \sum_{j=1}^N \bar{\alpha}_{t,j} h_j$ and distribution $\bar{\mathcal{D}}_t$ over the indices $\{1, \ldots, m\}$ as follows:

$$\bar{\mathcal{D}}_{t+1}(i) = \frac{e^{-y_i(\bar{f}_t(x_i')) - \bar{f}_t(x_i))}}{m \prod_{s=1}^t \bar{Z}_s}, \tag{10.16}$$

where $\bar{Z}_t$ is the analogue of Z_t computed as a function of $\bar{\mathcal{D}}_t$ instead of $\mathcal{D}_t$. Similarly, let $\bar{\epsilon}_t^+, \bar{\epsilon}_t^-$, and $\bar{\epsilon}_t$ denote the analogues of $\epsilon_t^+, \epsilon_t^-$, and ϵ_t defined with respect to $\bar{\mathcal{D}}_t$ instead of $\mathcal{D}_t$.

Following very similar steps as in section 7.2.2, we will use an inductive argument argument to show that coordinate descent on F and the RankBoost algorithm are in fact equivalent. Clearly, $\bar{\mathcal{D}}_{t+1} = \mathcal{D}_{t+1}$ if we have $\bar{f}_t = f_t$ for all t. We trivially have $\bar{f}_0 = f_0$, so we will make the inductive assumption that $\bar{f}_{t-1} = f_{t-1}$ and show that this implies $\bar{f}_t = f_t$.

At each iteration $t \geq 1$, the direction $\mathbf{e}_k$ selected by coordinate descent is the one minimizing the directional derivative, which is defined as:

$$F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k) = \lim_{\eta \to 0} \frac{F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k) - F(\bar{\boldsymbol{\alpha}}_{t-1})}{\eta}.$$

Since $F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k) = \sum_{i=1}^m e^{-y_i \sum_{j=1}^{t-1} \bar{\alpha}_{t-1,j}(h_j(x_i') - h_j(x_i)) - \eta y_i (h_k(x_i') - h_k(x_i))}$, the directional derivative along $\mathbf{e}_k$ can be expressed as follows:

$$\begin{aligned}
&F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k) \\
&= -\sum_{i=1}^m y_i (h_k(x_i') - h_k(x_i)) \exp\Big[- y_i \sum_{j=1}^N \bar{\alpha}_{t-1,j}(h_j(x_i') - h_j(x_i))\Big] \\
&= -\sum_{i=1}^m y_i (h_k(x_i') - h_k(x_i)) \bar{\mathcal{D}}_t(i) \Big[m \prod_{s=1}^{t-1} \bar{Z}_s \Big] \\
&= -\Big[\sum_{i=1}^m \bar{\mathcal{D}}_t(i) 1_{y_i(h_t(x_i') - h_t(x_i)) = +1} - \sum_{i=1}^m \bar{\mathcal{D}}_t(i) 1_{y_i(h_t(x_i') - h_t(x_i)) = -1} \Big] \Big[m \prod_{s=1}^{t-1} \bar{Z}_s \Big] \\
&= -[\bar{\epsilon}_t^+ - \bar{\epsilon}_t^-] \Big[m \prod_{s=1}^{t-1} \bar{Z}_s \Big].
\end{aligned}$$

The first equality holds by differentiation and evaluation at $\eta = 0$ and the second one follows from (10.12). In view of the final equality, since $m \prod_{s=1}^{t-1} \bar{Z}_s$ is fixed, the direction $\mathbf{e}_k$ selected by coordinate descent is the one minimizing $\bar{\epsilon}_t$. By the inductive hypothesis $\bar{\mathcal{D}}_t = \mathcal{D}_t$ and $\bar{\epsilon}_t = \epsilon_t$, thus, the chosen base ranker corresponds exactly to the base ranker h_t selected by RankBoost.

The step size η is identified by setting the derivative to zero in order to minimize the function in the chosen direction $\mathbf{e}_k$. Thus, using identity 10.12 and the definition of $\bar{\epsilon}_t$, we can write:

$$\begin{aligned}
&\frac{dF(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k)}{d\eta} = 0 \\
&\Leftrightarrow -\sum_{i=1}^{m} y_i(h_t(x'_i) - h_t(x_i))e^{-y_i \sum_{j=1}^{N} \bar{\alpha}_{t-1,j}(h_j(x'_i)-h_j(x_i))} e^{-\eta y_i(h_k(x'_i)-h_k(x_i))} = 0 \\
&\Leftrightarrow -\sum_{i=1}^{m} y_i(h_t(x'_i) - h_t(x_i))\bar{\mathcal{D}}_t(i)\Big[m \prod_{s=1}^{t-1} \bar{Z}_s\Big] e^{-\eta y_i(h_k(x'_i)-h_k(x_i))} = 0 \\
&\Leftrightarrow -\sum_{i=1}^{m} y_i(h_t(x'_i) - h_t(x_i))\bar{\mathcal{D}}_t(i)\, e^{-\eta y_i(h_t(x'_i)-h_t(x_i))} = 0 \\
&\Leftrightarrow -[\bar{\epsilon}_t^{+} e^{-\eta} - \bar{\epsilon}_t^{-} e^{\eta}] = 0 \\
&\Leftrightarrow \eta = \frac{1}{2} \log \frac{\bar{\epsilon}_t^{+}}{\bar{\epsilon}_t^{-}}.
\end{aligned}$$

By the inductive hypothesis, we have $\bar{\epsilon}_t^{+} = \epsilon_t^{+}$ and $\bar{\epsilon}_t^{-} = \epsilon_t^{-}$ and this proves that the step size chosen by coordinate descent matches the base ranker weight α_t of RankBoost. Thus, by combining the previous results we have $\bar{f}_t = f_t$ and the proof by induction is complete. This shows that coordinate descent applied to F precisely coincides with the RankBoost algorithm.

As in the classification case, other convex loss functions upper bounding the zero-one pairwise misranking loss can be used. In particular, the following objective function based on the logistic loss can be used: $\bar{\boldsymbol{\alpha}} \mapsto \sum_{i=1}^{m} \log(1 + e^{-y_i[f_N(x'_i) - f_N(x_i)]})$ to derive an alternative boosting-type algorithm.

10.4.3 Margin bound for ensemble methods in ranking

To simplify the presentation, we will assume for the results of this section, as in section 10.2, that the pairwise labels are in $\{-1, +1\}$. By lemma 7.4, the empirical Rademacher complexity of the convex hull $\operatorname{conv}(\mathcal{H})$ equals that of $\mathcal{H}$. Thus, theorem 10.1 immediately implies the following guarantee for ensembles of hypotheses in ranking.

Corollary 10.4 *Let $\mathcal{H}$ be a set of real-valued functions. Fix $\rho > 0$; then, for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of a sample S of size m, each of the following ranking guarantees holds for all $h \in \operatorname{conv}(\mathcal{H})$:*

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\big(\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H}) + \mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H})\big) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \tag{10.17}$$

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\big(\widehat{\mathfrak{R}}_{S_1}(\mathcal{H}) + \widehat{\mathfrak{R}}_{S_2}(\mathcal{H})\big) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}\,. \tag{10.18}$$

For RankBoost, these bounds apply to $f/\|\boldsymbol{\alpha}\|_1$, where f is the hypothesis returned by the algorithm. Since f and $f/\|\boldsymbol{\alpha}\|_1$ induce the same ordering of the points, for any $\delta > 0$, the following holds with probability at least $1-\delta$:

$$R(f) \leq \widehat{R}_{S,\rho}(f/\|\boldsymbol{\alpha}\|_1) + \frac{2}{\rho}\big(\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H}) + \mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H})\big) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}} \tag{10.19}$$

Remarkably, the number of rounds of boosting T does not appear in this bound. The bound depends only on the margin ρ, the sample size m, and the Rademacher complexity of the family of base classifiers $\mathcal{H}$. Thus, the bound guarantees an effective generalization if the pairwise margin loss $\widehat{R}_{S,\rho}(f/\|\boldsymbol{\alpha}\|_1)$ is small for a relatively large ρ. A bound similar to that of theorem 7.7 for AdaBoost can be derived for the empirical pairwise ranking margin loss of RankBoost (see exercise 10.3) and similar comments on that result apply here.

These results provide a margin-based analysis in support of ensemble methods in ranking and RankBoost in particular. As in the case of AdaBoost, however, RankBoost in general does not achieve a maximum margin. But, in practice, it has been observed to obtain excellent pairwise ranking performances.

10.5 Bipartite ranking

This section examines an important ranking scenario within the score-based setting, the *bipartite ranking problem*. In this scenario, the set of points $\mathcal{X}$ is partitioned into two classes: $\mathcal{X}_+$ the class of positive points, and $\mathcal{X}_-$ that of negative ones. The problem consists of ranking positive points higher than negative ones. For example, for a fixed search engine query, the task consists of ranking relevant (positive) documents higher than irrelevant (negative) ones.

The bipartite problem could be treated in the way already discussed in the previous sections with exactly the same theory and algorithms. However, the setup typically adopted for this problem is different: instead of assuming that the learner receives a sample of random pairs, here pairs of positive and negative elements, it is assumed that it receives a sample of positive points from some distribution and a sample of negative points from another. This leads to the set of all pairs made of a positive point of the first sample and a negative point of the second.

More formally, the learner receives a sample $S_+ = (x'_1, \ldots, x'_m)$ drawn i.i.d. according to some distribution $\mathcal{D}_+$ over $\mathcal{X}_+$, and a sample $S_- = (x_1, \ldots, x_n)$ drawn i.i.d. according to some distribution $\mathcal{D}_-$ over $\mathcal{X}_-$.[17] Given a hypothesis set $\mathcal{H}$ of

[17] This two-distribution formulation also avoids a potential dependency issue that can arise for some modeling of the problem: if pairs are drawn according to some distribution $\mathcal{D}$ over $\mathcal{X}_- \times \mathcal{X}_+$

functions mapping $\mathcal{X}$ to $\mathbb{R}$, the learning problem consists of selecting a hypothesis $h \in \mathcal{H}$ with small expected bipartite misranking or generalization error $R(h)$:

$$R(h) = \underset{\substack{x \sim \mathcal{D}_- \\ x' \sim \mathcal{D}_+}}{\mathbb{P}} \left[h(x') < h(x) \right]. \tag{10.20}$$

The empirical pairwise misranking or empirical error of h is denoted by $\widehat{R}_{S_+, S_-}(h)$ and defined by

$$\widehat{R}_{S_+, S_-}(h) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} 1_{h(x'_i) < h(x_j)}. \tag{10.21}$$

Note that while the bipartite ranking problem bears some similarity with binary classification, in particular, the presence of two classes, they are distinct problems, since their objectives and measures of success clearly differ.

By the definition of the formulation of the bipartite ranking just presented, the learning algorithm must typically deal with mn pairs. For example, the application of SVMs to ranking in this scenario leads to an optimization with mn slack variables or constraints. With just a thousand positive and a thousand negative points, one million pairs would need to be considered. This can lead to a prohibitive computational cost for some learning algorithms. The next section shows that RankBoost admits an efficient implementation in the bipartite scenario.

10.5.1 Boosting in bipartite ranking

This section shows the efficiency of RankBoost in the bipartite scenario and discusses the connection between AdaBoost and RankBoost in this context.

The key property of RankBoost leading to an efficient algorithm in the bipartite setting is the fact that its objective function is based on the exponential function. As a result, it can be decomposed into the product of two functions, one depending on only the positive and the other on only the negative points. Similarly, the distribution $\mathcal{D}_t$ maintained by the algorithm can be factored as the product of two distributions $\mathcal{D}_t^+$ and $\mathcal{D}_t^-$. This is clear for the uniform distribution $\mathcal{D}_1$ at the first round as for any $i \in [m]$ and $j \in [n]$, $\mathcal{D}_1(i, j) = 1/(mn) = \mathcal{D}_1^+(i)\mathcal{D}_1^-(j)$ with $\mathcal{D}_1^+(i) = 1/m$ and $\mathcal{D}_1^-(j) = 1/n$. This property is recursively preserved since, in view of the following, the decomposition of $\mathcal{D}_t$ implies that of $\mathcal{D}_{t+1}$ for any $t \in [T]$.

and the learner makes use of this information to augment its training sample, then the resulting sample is in general not i.i.d. This is because if (x_1, x'_1) and (x_2, x'_2) are in the sample, then so are the pairs (x_1, x'_2) and (x_2, x'_1) and thus the pairs are not independent. However, without sample augmentation, the points are i.i.d., and this issue does not arise.

For any $i \in [m]$ and $j \in [n]$, by definition of the update, we can write:

$$\mathcal{D}_{t+1}(i,j) = \frac{\mathcal{D}_t(i,j)e^{-\alpha_t[h_t(x'_i)-h_t(x_j)]}}{Z_t} = \frac{\mathcal{D}^+_t(i)e^{-\alpha_t h_t(x'_i)}}{Z_{t,+}} \frac{\mathcal{D}^-_t(j)e^{\alpha_t h_t(x_j)}}{Z_{t,-}},$$

since the normalization factor Z_t can also be decomposed as $Z_t = Z^-_t Z^+_t$, with $Z^+_t = \sum_{i=1}^m \mathcal{D}^+_t(i)e^{-\alpha_t h_t(x'_i)}$ and $Z^-_t = \sum_{j=1}^n \mathcal{D}^-_t(j)e^{\alpha_t h_t(x_j)}$. Furthermore, the pairwise misranking of a hypothesis $h \in \mathcal{H}$ based on the distribution $\mathcal{D}_t$ used to determine h_t can also be computed as the difference of two quantities, one depending only on positive points, the other only on negative ones:

$$\mathop{\mathbb{E}}_{(i,j)\sim\mathcal{D}_t}[h(x'_i) - h(x_j)] = \mathop{\mathbb{E}}_{i\sim\mathcal{D}^+_t}[\mathop{\mathbb{E}}_{j\sim\mathcal{D}^-_t}[h(x'_i) - h(x_j)]] = \mathop{\mathbb{E}}_{i\sim\mathcal{D}^+_t}[h(x'_i)] - \mathop{\mathbb{E}}_{j\sim\mathcal{D}^-_t}[h(x_j)].$$

Thus, the time and space complexity of RankBoost depends only on the total number of points $m+n$ and not the number of pairs mn. More specifically, ignoring the call to the weak ranker or the cost of determining h_t, the time and space complexity of each round is linear, that is, in $O(m+n)$. Furthermore, the cost of determining h_t is a function of $O(m+n)$ and not $O(mn)$. Figure 10.2 gives the pseudocode of the algorithm adapted to the bipartite scenario.

In the bipartite scenario, a connection can be made between the classification algorithm AdaBoost and the ranking algorithm RankBoost. In particular, the objective function of RankBoost can be expressed as follows for any $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_T) \in \mathbb{R}^T$, $T \geq 1$:

$$\begin{aligned} F_{\text{RankBoost}}(\boldsymbol{\alpha}) &= \sum_{j=1}^m \sum_{i=1}^n \exp(-[f(x'_i) - f(x_j)]) \\ &= \Big(\sum_{i=1}^m e^{-\sum_{t=1}^T \alpha_t h_t(x'_i)}\Big)\Big(\sum_{j=1}^n e^{+\sum_{t=1}^T \alpha_t h_t(x_j)}\Big) \\ &= F_+(\boldsymbol{\alpha})F_-(\boldsymbol{\alpha}), \end{aligned}$$

where F_+ denotes the function defined by the sum over the positive points and F_- the function defined over the negative points. The objective function of AdaBoost can be defined in terms of these same two functions as follows:

$$\begin{aligned} F_{\text{AdaBoost}}(\boldsymbol{\alpha}) &= \sum_{i=1}^m \exp(-y'_i f(x'_i)) + \sum_{j=1}^n \exp(-y_j f(x_j)) \\ &= \sum_{i=1}^m e^{-\sum_{t=1}^T \alpha_t h_t(x'_i)} + \sum_{j=1}^n e^{+\sum_{t=1}^T \alpha_t h_t(x_j)} \\ &= F_+(\boldsymbol{\alpha}) + F_-(\boldsymbol{\alpha}). \end{aligned}$$

```
BipartiteRankBoost($S = (x'_1, \ldots, x'_m, x_1, \ldots, x_n)$)
 1  for $j \leftarrow 1$ to $m$ do
 2      $\mathcal{D}^+_1(j) \leftarrow \frac{1}{m}$
 3  for $i \leftarrow 1$ to $n$ do
 4      $\mathcal{D}^-_1(i) \leftarrow \frac{1}{n}$
 5  for $t \leftarrow 1$ to $T$ do
 6      $h_t \leftarrow$ base ranker in $\mathcal{H}$ with smallest $\epsilon^-_t - \epsilon^+_t = \mathbb{E}_{j \sim \mathcal{D}^-_t}[h(x_j)] - \mathbb{E}_{i \sim \mathcal{D}^+_t}[h(x'_i)]$
 7      $\alpha_t \leftarrow \frac{1}{2} \log \frac{\epsilon^+_t}{\epsilon^-_t}$
 8      $Z^+_t \leftarrow 1 - \epsilon^+_t + \sqrt{\epsilon^+_t \epsilon^-_t}$
 9      for $i \leftarrow 1$ to $m$ do
10          $\mathcal{D}^+_{t+1}(i) \leftarrow \frac{\mathcal{D}^+_t(i) \exp\left[-\alpha_t h_t(x'_i)\right]}{Z^+_t}$
11      $Z^-_t \leftarrow 1 - \epsilon^-_t + \sqrt{\epsilon^+_t \epsilon^-_t}$
12      for $j \leftarrow 1$ to $n$ do
13          $\mathcal{D}^-_{t+1}(j) \leftarrow \frac{\mathcal{D}^-_t(j) \exp\left[+\alpha_t h_t(x_j)\right]}{Z_t}$
14  $f \leftarrow \sum_{t=1}^T \alpha_t h_t$
15  return $f$
```

Figure 10.2
Pseudocode of RankBoost in a bipartite setting, with $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$, $\epsilon^+_t = \mathbb{E}_{i \sim \mathcal{D}^+_t}[h(x'_i)]$ and $\epsilon^-_t = \mathbb{E}_{j \sim \mathcal{D}^-_t}[h(x_j)]$.

Note that the gradient of the objective function of RankBoost can be expressed in terms of AdaBoost as follows:

$$\begin{aligned}
\nabla_{\boldsymbol{\alpha}} F_{\text{RankBoost}}(\boldsymbol{\alpha}) &= F_-(\boldsymbol{\alpha}) \nabla_{\boldsymbol{\alpha}} F_+(\boldsymbol{\alpha}) + F_+(\boldsymbol{\alpha}) \nabla_{\boldsymbol{\alpha}} F_-(\boldsymbol{\alpha}) \qquad (10.22)\\
&= F_-(\boldsymbol{\alpha})(\nabla_{\boldsymbol{\alpha}} F_+(\boldsymbol{\alpha}) + \nabla_{\boldsymbol{\alpha}} F_-(\boldsymbol{\alpha})) + (F_+(\boldsymbol{\alpha}) - F_-(\boldsymbol{\alpha}))\nabla_{\boldsymbol{\alpha}} F_-(\boldsymbol{\alpha})\\
&= F_-(\boldsymbol{\alpha}) \nabla_{\boldsymbol{\alpha}} F_{\text{AdaBoost}}(\boldsymbol{\alpha}) + (F_+(\boldsymbol{\alpha}) - F_-(\boldsymbol{\alpha}))\nabla_{\boldsymbol{\alpha}} F_-(\boldsymbol{\alpha}).
\end{aligned}$$

If $\boldsymbol{\alpha}$ is a minimizer of F_{AdaBoost}, then $\nabla_{\boldsymbol{\alpha}} F_{\text{AdaBoost}}(\boldsymbol{\alpha}) = 0$ and it can be shown that the equality $F_+(\boldsymbol{\alpha}) - F_-(\boldsymbol{\alpha}) = 0$ also holds for $\boldsymbol{\alpha}$, provided that the family of base hypotheses $\mathcal{H}$ used for AdaBoost includes the constant hypothesis $h_0 \colon x \mapsto 1$, which often is the case in practice. Then, by (10.22), this implies that $\nabla_{\boldsymbol{\alpha}} F_{\text{RankBoost}}(\boldsymbol{\alpha}) = 0$ and therefore that $\boldsymbol{\alpha}$ is also a minimizer of the convex

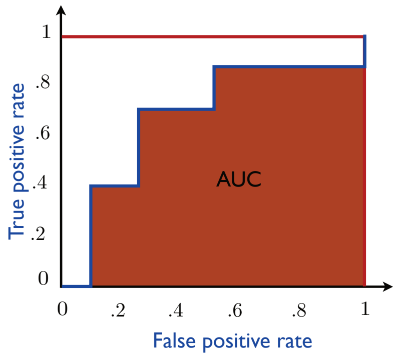


Figure 10.3
The AUC (area under the ROC curve) is a measure of the performance of a bipartite ranking.

function $F_{\text{RankBoost}}$. In general, F_{AdaBoost} does not admit a minimizer. Nevertheless, it can be shown that if $\lim_{k\to\infty} F_{\text{AdaBoost}}(\boldsymbol{\alpha}_k) = \inf_{\boldsymbol{\alpha}} F_{\text{AdaBoost}}(\boldsymbol{\alpha})$ for some sequence $(\boldsymbol{\alpha}_k)_{k\in\mathbb{N}}$, then, under the same assumption on the use of a constant base hypothesis and for a non-linearly separable dataset, the following holds: $\lim_{k\to\infty} F_{\text{RankBoost}}(\boldsymbol{\alpha}_k) = \inf_{\boldsymbol{\alpha}} F_{\text{RankBoost}}(\boldsymbol{\alpha})$.

The connections between AdaBoost and RankBoost just mentioned suggest that AdaBoost could achieve a good ranking performance as well. This is often observed empirically, a fact that brings strong support to the use of AdaBoost both as a classifier and a ranking algorithm. Nevertheless, RankBoost may converge faster and achieve a good ranking faster than AdaBoost.

10.5.2 Area under the ROC curve

The performance of a bipartite ranking algorithm is typically reported in terms of the *area under the receiver operating characteristic (ROC) curve*, or the *area under the curve (AUC)* for short.

Let U be a test sample used to evaluate the performance of h (or a training sample) with m positive points $z'_1, \ldots, z'_m$ and n negative points $z_1, \ldots, z_n$. For any $h \in \mathcal{H}$, let $\widehat{R}(h, U)$ denote the average pairwise misranking of h over U. Then, the AUC of h for the sample U is precisely $1 - \widehat{R}(h, U)$, that is, its average pairwise ranking accuracy on U:

$$\text{AUC}(h, U) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} 1_{h(z'_i) \geq h(z_j)} = \mathop{\mathbb{P}}_{\substack{z \sim \widehat{\mathcal{D}}_U^- \\ z' \sim \widehat{\mathcal{D}}_U^+}} [h(z') \geq h(z)].$$

Here, $\widehat{\mathcal{D}}_U^+$ denotes the empirical distribution corresponding to the positive points in U and $\widehat{\mathcal{D}}_U^+$ the empirical distribution corresponding to the negative ones. AUC(h, U)

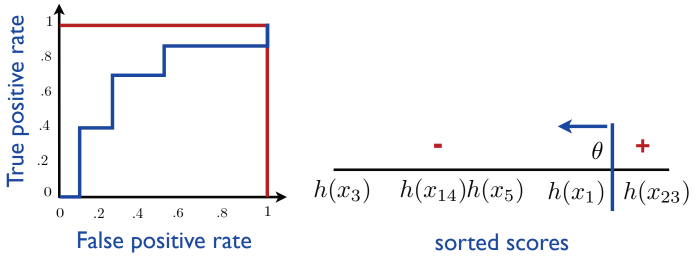


Figure 10.4
An example ROC curve and illustrated threshold. Varying the value of θ from one extreme to the other generates points on the curve.

is thus an empirical estimate of the pairwise ranking accuracy based on the sample U, and by definition it is in $[0, 1]$. Higher AUC values correspond to a better ranking performance. In particular, an AUC of one indicates that the points of U are ranked perfectly using h. $\text{AUC}(h, U)$ can be computed in linear time from a sorted array containing the $m+n$ elements $h(z'_i)$ and $h(z_j)$, for $i \in [m]$ and $j \in [n]$. Assuming that the array is sorted in increasing order (with a positive point placed higher than a negative one if they both have the same scores) the total number of correctly ranked pairs r can be computed as follows. Starting with $r = 0$, the array is inspected in increasing order of the indices while maintaining at any time the number of negative points seen n and incrementing the current value of r with n whenever a positive point is found. After full inspection of the array, the AUC is given by $r/(mn)$. Thus, assuming that a comparison-based sorting algorithm is used, the complexity of the computation of the AUC is in $O((m+n)\log(m+n))$.

As indicated by its name, the AUC coincides with the area under the ROC curve (figure 10.3). An ROC curve plots the *true positive rate*, that is, the percentage of positive points correctly predicted as positive as a function of the *false positive rate*, that is, the percentage of negative points incorrectly predicted as positive. Figure 10.4 illustrates the definition and construction of an ROC curve.

Points are generated along the curve by varying a threshold value θ as in the right panel of figure 10.4, from higher values to lower ones. The threshold is used to determine the label of any point x (positive or negative) based on $\text{sgn}(h(x) - \theta)$. At one extreme, all points are predicted as negative; thus, the false positive rate is zero, but the true positive rate is zero as well. This gives the first point $(0, 0)$ of the plot. At the other extreme, all points are predicted as positive; thus, both the true and the false positive rates are equal to one, which gives the point $(1, 1)$. In the ideal case, as already discussed, the AUC value is one, and, with the exception of $(0, 0)$, the curve coincides with a horizontal line reaching $(1, 1)$.

10.6 Preference-based setting

This section examines a different setting for the problem of learning to rank: the *preference-based setting*. In this setting, the objective is to rank as accurately as possible any test subset $X \subseteq \mathcal{X}$, typically a finite set that we refer to as a *finite query subset*. This is close to the query-based scenario of search engines or information extraction systems and the terminology stems from the fact that X could be a set of items needed to rank in response to a particular query. The advantage of this setting over the score-based setting is that here the learning algorithm is not required to return a linear ordering of all points of $\mathcal{X}$, which may be impossible to achieve faultlessly in accordance with a general possibly non-transitive pairwise preference labeling. Supplying a correct linear ordering for a query subset is more likely to be achievable exactly or at least with a better approximation.

The preference-based setting consists of two stages. In the first stage, a sample of labeled pairs S, exactly as in the score-based setting, is used to learn a *preference function* $h\colon \mathcal{X} \times \mathcal{X} \mapsto [0,1]$, that is, a function that assigns a higher value to a pair (u,v) when u is preferred to v or is to be ranked higher than v, and smaller values in the opposite case. This preference function can be obtained as the output of a standard classification algorithm trained on S. A crucial difference with the score-based setting is that, in general, the preference function h is not required to induce a linear ordering. The relation it induces may be non-transitive; thus, we may have, for example, $h(u,v) = h(v,w) = h(w,u) = 1$ for three distinct points u, v, and w.

In the second stage, given a query subset $X \subseteq \mathcal{X}$, the preference function h is used to determine a ranking of X. How can h be used to generate an accurate ranking? This will be the main focus of this section. The computational complexity of the algorithm determining the ranking is also crucial. Here, we will measure its running time complexity in terms of the number of calls to h.

When the preference function is obtained as the output of a binary classification algorithm, the preference-based setting can be viewed as a reduction of ranking to classification: the second stage specifies how a ranking is obtained from a classifier's output.

10.6.1 Second-stage ranking problem

The ranking problem of the second stage is modeled as follows. We assume that a preference function h is given. From the point of view of this stage, the way the function h has been determined is immaterial, it can be viewed as a black box. As already discussed, h is not assumed to be transitive. But, we will assume that it is *pairwise consistent*, that is $h(u,v) + h(v,u) = 1$, for all $u, v \in \mathcal{X}$.

Let $\mathcal{D}$ be an unknown distribution according to which pairs (X, σ^*) are drawn where $X \subseteq \mathcal{X}$ is a query subset and σ^* a target ranking or permutation of X, that is, a bijective function from X to $\{1, \ldots, |X|\}$. Thus, we consider a stochastic scenario, and σ^* is a random variable. The objective of a second-stage algorithm $\mathcal{A}$ consists of using the preference function h to return an accurate ranking $\mathcal{A}(X)$ for any query subset X. The algorithm may be deterministic, in which case $\mathcal{A}(X)$ is uniquely determined from X or it may be randomized, in which case we denote by s the randomization seed it may depend on.

The following loss function L can be used to measure the disagreement between a ranking σ and a desired one σ^* over a set X of $n \geq 1$ elements:

$$L(\sigma, \sigma^*) = \frac{2}{n(n-1)} \sum_{u \neq v} 1_{\sigma(u) < \sigma(v)} 1_{\sigma^*(v) < \sigma^*(u)}, \tag{10.23}$$

where the sum runs over all pairs (u, v) with u and v distinct elements of X. All the results presented in the following hold for a broader set of loss functions described later. Abusing the notation, we also define the loss of the preference function h with respect to a ranking σ^* of a set X of $n \geq 1$ elements by

$$L(h, \sigma^*) = \frac{2}{n(n-1)} \sum_{u \neq v} h(u, v) 1_{\sigma^*(v) < \sigma^*(u)}. \tag{10.24}$$

The expected loss for a deterministic algorithm $\mathcal{A}$ is thus $\mathbb{E}_{(X,\sigma^*) \sim \mathcal{D}}[L(\mathcal{A}(X), \sigma^*)]$. The *regret* of algorithm $\mathcal{A}$ is then defined as the difference between its loss and that of the best fixed global ranking. This can be written as follows:

$$\text{Reg}(\mathcal{A}) = \mathop{\mathbb{E}}_{(X,\sigma^*) \sim \mathcal{D}}[L(\mathcal{A}(X), \sigma^*)] - \min_{\sigma'} \mathop{\mathbb{E}}_{(X,\sigma^*) \sim \mathcal{D}}[L(\sigma'_{|X}, \sigma^*)], \tag{10.25}$$

where $\sigma'_{|X}$ denotes the ranking induced on X by a global ranking σ' of $\mathcal{X}$. Similarly, we define the regret of the preference function as follows

$$\text{Reg}(h) = \mathop{\mathbb{E}}_{(X,\sigma^*) \sim \mathcal{D}}[L(h_{|X}, \sigma^*)] - \min_{h'} \mathop{\mathbb{E}}_{(X,\sigma^*) \sim \mathcal{D}}[L(h'_{|X}, \sigma^*)], \tag{10.26}$$

where $h_{|X}$ denotes the restriction of h to $X \times X$, and similarly with h'. The regret results presented in this section hold assuming the following *pairwise independence on irrelevant alternatives* property:

$$\mathop{\mathbb{E}}_{\sigma^* | X_1}[1_{\sigma^*(v) < \sigma^*(u)}] = \mathop{\mathbb{E}}_{\sigma^* | X_2}[1_{\sigma^*(v) < \sigma^*(u)}], \tag{10.27}$$

for any $u, v \in \mathcal{X}$ and any two sets X_1 and X_2 containing u and v, and where $\sigma^*|X$ denotes the random variable σ^* conditioned on X.[18] Similar regret definitions can be given for a randomized algorithm additionally taking the expectation over s.

Clearly, the quality of the ranking output by the second-stage algorithm intimately depends on that of the preference function h. In the next sections, we discuss both a deterministic and a randomized second-stage algorithm for which the regret can be upper bounded in terms of the regret of the preference function.

10.6.2 Deterministic algorithm

A natural deterministic algorithm for the second-stage is based on the *sort-by-degree algorithm*. This consists of ranking each element of X based on the number of other elements it is preferred to according to the preference function h. Let $\mathcal{A}_{\text{sort-by-degree}}$ denote this algorithm. In the bipartite setting, the following bounds can be proven for the expected loss of this algorithm and its regret:

$$\mathop{\mathbb{E}}_{X,\sigma^*}[L(\mathcal{A}_{\text{sort-by-degree}}(X), \sigma^*)] \leq 2 \mathop{\mathbb{E}}_{X,\sigma^*}[L(h, \sigma^*)] \tag{10.30}$$

$$\text{Reg}(\mathcal{A}_{\text{sort-by-degree}}(X)) \leq 2\,\text{Reg}(h)\,. \tag{10.31}$$

These results show that the sort-by-degree algorithm can achieve an accurate ranking when the loss or the regret of the preference function h is small. They also bound the *ranking* loss or regret of the algorithm in terms of the *classification* loss or regret of h, which can be viewed as a guarantee for the reduction of ranking to classification using the sort-by-degree algorithm.

Nevertheless, in some cases, the guarantee given by these results is weak or uninformative owing to the presence of the factor of two. Consider the case of a binary classifier h with an error rate of just 25 percent, which is quite reasonable in many applications. Assume that the Bayes error is close to zero for the classification problem and, similarly, that for the ranking problem the regret and loss approximately coincide. Then, using the bound in (10.30) guarantees a worst-case pairwise misranking error of at most 50 percent for the ranking algorithm, which is the pairwise misranking error of random ranking.

[18] More generally, they hold without that assumption using the following weaker notions of regret:

$$\text{Reg}'(\mathcal{A}) = \mathop{\mathbb{E}}_{(X,\sigma^*)\sim\mathcal{D}}[L(\mathcal{A}(X), \sigma^*)] - \mathop{\mathbb{E}}_{X}\Big[\min_{\sigma'} \mathop{\mathbb{E}}_{\sigma^*|X}[L(\sigma', \sigma^*)]\Big] \tag{10.28}$$

$$\text{Reg}'(h) = \mathop{\mathbb{E}}_{(X,\sigma^*)\sim\mathcal{D}}[L(h_{|X}, \sigma^*)] - \mathop{\mathbb{E}}_{X}\Big[\min_{h'} \mathop{\mathbb{E}}_{\sigma^*|X}[L(h', \sigma^*)]\Big]\,, \tag{10.29}$$

where σ' denotes a ranking of X and h' a preference function defined over $X \times X$.

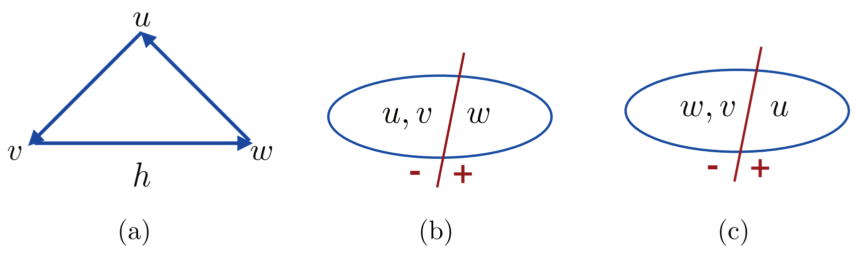


Figure 10.5
Illustration of the proof of theorem 10.5.

Furthermore, the running time complexity of the algorithm quadratic, that is in $\Omega(|X|^2)$ of a query set X, since it requires calling the preference function for every pair (u, v) with u and v in X.

As shown by the following theorem, no deterministic algorithm can improve upon the factor of two appearing in the regret guarantee of the sort-by-degree algorithm.

Theorem 10.5 (Lower bound for deterministic algorithms) *For any deterministic algorithm $\mathcal{A}$, there is a bipartite distribution for which*

$$\operatorname{Reg}(\mathcal{A}) \geq 2 \operatorname{Reg}(h). \tag{10.32}$$

Proof: Consider the simple case where $\mathcal{X} = X = \{u, v, w\}$ and where the preference function induces a cycle as illustrated by figure 10.5a. An arrow from u to v indicates that v is preferred to u according to h. The proof is based on an adversarial choice of the target σ^*.

Without loss of generality, either $\mathcal{A}$ returns the ranking u, v, w (figure 10.5b) or w, v, u (figure 10.5c). In the first case, let σ^* be defined by the labeling indicated in the figure. In that case, we have $L(h, \sigma^*) = 1/3$, since u is preferred to w according to h while w is labeled positively and u negatively. The loss of the algorithm is $L(\mathcal{A}, \sigma^*) = 2/3$, since both u and v are ranked higher than the positively labeled w by the algorithm. Similarly, σ^* can be defined as in figure 10.5c in the second case, and we find again that $L(h, \sigma^*) = 1/3$ and $L(\mathcal{A}, \sigma^*) = 2/3$. This concludes the proof. □

The theorem suggests that randomization is necessary in order to achieve a better guarantee. In the next section, we present a randomized algorithm that benefits both from better guarantees and a better time complexity.

10.6.3 Randomized algorithm

The general idea of the algorithm described in this section is to use a straightforward extension of the randomized QuickSort algorithm in the second stage. Unlike in

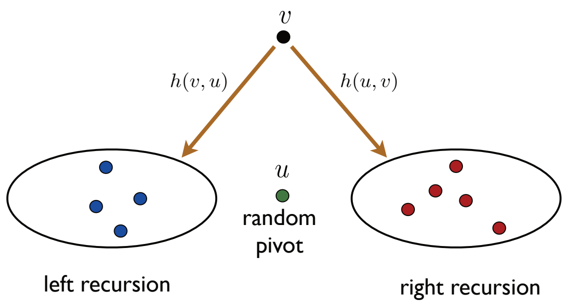


Figure 10.6
Illustration of randomized QuickSort based on a preference function h (not necessarily transitive).

the standard version of QuickSort, here the comparison function is based on the preference function, which in general is not transitive. Nevertheless, it can be shown here, too, that the expected time complexity of the algorithm is in $O(n \log n)$ when applied to an array of size n.

The algorithm works as follows, as illustrated by figure 10.6. At each recursive step, a *pivot* element u is selected uniformly at random from X. For each $v \neq u$, v is placed on the left of u with probability $h(v, u)$ and to its right with the remaining probability $h(u, v)$. The algorithm proceeds recursively with the array to the left of u and the one to its right and returns the concatenation of the permutation returned by the left recursion, u, and the permutation returned by the right recursion.

Let $\mathcal{A}_{\text{QuickSort}}$ denote this algorithm. In the bipartite setting, the following guarantees can be proven:

$$\mathop{\mathbb{E}}_{X,\sigma^*,s}[L(\mathcal{A}_{\text{QuickSort}}(X, s), \sigma^*)] = \mathop{\mathbb{E}}_{X,\sigma^*}[L(h, \sigma^*)] \tag{10.33}$$

$$\text{Reg}(\mathcal{A}_{\text{QuickSort}}) \leq \text{Reg}(h)\,. \tag{10.34}$$

Thus, here, the factor of two of the bounds in the deterministic case has vanished, which is substantially more favorable. Furthermore, the guarantee for the loss is an equality. Moreover, the expected time complexity of the algorithm is only in $O(n \log n)$, and, if only the top k items are needed to be ranked, as in many applications, the time complexity is reduced to $O(n + k \log k)$.

For the QuickSort algorithm, the following guarantee can also be proven in the case of general ranking setting (not necessarily bipartite setting):

$$\mathop{\mathbb{E}}_{X,\sigma^*,s}[L(\mathcal{A}_{\text{QuickSort}}(X, s), \sigma^*)] \leq 2 \mathop{\mathbb{E}}_{X,\sigma^*}[L(h, \sigma^*)]. \tag{10.35}$$

10.6.4 Extension to other loss functions

All of the results just presented hold for a broader class of loss functions L_ω defined in terms of a *weight function* or *emphasis function* ω. L_ω is similar to (10.23), but measures the weighted disagreement between a ranking σ and a desired one σ^* over a set X of $n \geq 1$ elements as follows:

$$L_\omega(\sigma, \sigma^*) = \frac{2}{n(n-1)} \sum_{u \neq v} \omega(\sigma^*(v), \sigma^*(u))\, 1_{\sigma(u)<\sigma(v)}\, 1_{\sigma^*(v)<\sigma^*(u)}, \tag{10.36}$$

where the sum runs over all pairs (u, v) with u and v distinct elements of X, and where ω is a symmetric function whose properties are described below. Thus, the loss counts the number of pairwise misrankings of σ with respect to σ^*, each weighted by ω. The function ω is assumed to satisfy the following three natural axioms:

- symmetry: $\omega(i, j) = \omega(j, i)$ for all i, j;
- monotonicity: $\omega(i, j) \leq \omega(i, k)$ if either $i < j < k$ or $i > j > k$;
- triangle inequality: $\omega(i, j) \leq \omega(i, k) + \omega(k, j)$.

The motivation for this last property stems from the following: if correctly ordering items in positions (i, k) and (k, j) is not of great importance, then the same should hold for items in positions (i, j).

Using different functions ω, the family of functions L_ω can cover several familiar and important losses. Here are some examples. Setting $\omega(i, j) = 1$ for all $i \neq j$ yields the unweighted pairwise misranking measure. For a fixed integer $k \geq 1$, the function ω defined by $\omega(i, j) = 1_{((i \leq k) \vee (j \leq k)) \wedge (i \neq j)}$ for all (i, j) can be used to emphasize ranking at the top k elements. Misranking of pairs with at least one element ranked among the top k is penalized by this function. This can be of interest in applications such as information extraction or search engines where the ranking of the top documents matters more. For this emphasis function, all elements ranked below k are in a tie. Any tie relation can be encoded using ω. Finally, in a bipartite ranking scenario with m^+ positive and m^- negative points and $m^+ + m^- = n$, choosing $\omega(i, j) = \frac{n(n-1)}{2m^-m^+}$ yields the standard loss function coinciding with $1 - \text{AUC}$.

10.7 Other ranking criteria

The objective function for the ranking problems discussed in this chapter were all based on pairwise misranking. Other ranking criteria have been introduced in information retrieval and used to derive alternative ranking algorithms. Here, we briefly present several of these criteria.

- Precision, precision@n, average precision, recall. All of these criteria assume that points are partitioned into two classes (positives and negatives), as in the bipartite ranking setting. *Precision* is the fraction of positively predicted points that are in fact positive. Whereas precision takes into account all positive predictions, *precision@n* only considers the top n predictions. For example, precision@5 considers only the top 5 positively predicted points. *Average precision* involves computing precision@n for each value of n, and averaging across these values. Each precision@n computation can be interpreted as computing precision for a fixed value of *recall*, or the fraction of positive points that are predicted to be positive (recall coincides with the notion of true positive rate).
- DCG, NDCG. These criteria assume the existence of relevance scores associated with the points to be ranked, e.g., given a web search query, each website returned by a search engine has an associated relevance score. Moreover, these criteria measure the extent to which points with large relevance scores appear at or near the beginning of a ranking. Define $(c_i)_{i \in \mathbb{N}}$ as a predefined sequence of non-increasing and non-negative discount factors, e.g., $c_i = \log(i)^{-1}$. Then, given a ranking of m points and defining r_i as the relevance score of the ith point in this ranking, the *discounted cumulative gain (DCG)* is defined as $DCG = \sum_{i=1}^{m} c_i r_i$. Note that DCG is an increasing function of m. In contrast, the *normalized discounted cumulative gain (NDCG)* normalizes the DCG across values of m by dividing the DCG by the IDCG, or the ideal DCG that would result from an optimal ordering of the points.

10.8 Chapter notes

The problem of learning to rank is distinct from the purely algorithmic one of rank aggregation, which, as shown by Dwork, Kumar, Naor, and Sivakumar [2001], is NP-hard even for $k = 4$ rankings. The Rademacher complexity and margin-based generalization bounds for pairwise ranking given in theorem 10.1 and corollary 6.13 are novel. Margin bounds based on covering numbers were also given by Rudin, Cortes, Mohri, and Schapire [2005]. Other learning bounds in the score-based setting of ranking, including VC-dimension and stability-based learning bounds, have been given by Agarwal and Niyogi [2005], Agarwal et al. [2005] and Cortes et al. [2007b].

The ranking algorithm based on SVMs presented in section 10.3 has been used and discussed by several researchers. One early and specific discussion of its use can be found in Joachims [2002]. The fact that the algorithm is simply a special instance of SVMs seems not to be clearly stated in the literature. The theoretical justification presented here for its use in ranking is novel.

RankBoost was introduced by Freund et al. [2003]. The version of the algorithm presented here is the coordinate descent RankBoost from Rudin et al. [2005]. RankBoost in general does not achieve a maximum margin and may not increase the margin at each iteration. A Smooth Margin ranking algorithm [Rudin et al., 2005] based on a modified version of the objective function of RankBoost can be shown to increase the smooth margin at every iteration, but the comparison of its empirical performance with that of RankBoost has not been reported. For the empirical ranking quality of AdaBoost and the connections between AdaBoost and RankBoost in the bipartite setting, see Cortes and Mohri [2003] and Rudin et al. [2005].

The Receiver Operating Characteristics (ROC) curves were originally developed in signal detection theory [Egan, 1975] in connection with radio signals during World War II. They also had applications to psychophysics [Green and Swets, 1966] and have been used since then in a variety of other applications, in particular for medical decision making. The area under an ROC curve (AUC) is equivalent to the Wilcoxon-Mann-Whitney statistic [Hanley and McNeil, 1982] and is closely related to the Gini index [Breiman et al., 1984] (see also chapter 9). For a statistical analysis of the AUC and confidence intervals depending on the error rate, see Cortes and Mohri [2003, 2005]. The deterministic algorithm in the preference-based setting discussed in this chapter was presented and analyzed by Balcan et al. [2008]. The randomized algorithm as well as much of the results presented in section 10.6 are due to Ailon and Mohri [2008].

A somewhat related problem of *ordinal regression* has been studied by some authors [McCullagh, 1980, McCullagh and Nelder, 1983, Herbrich et al., 2000] which consists of predicting the correct label of each item out of a finite set, as in multi-class classification, with the additional assumption of an ordering among the labels. This problem is distinct, however, from the pairwise ranking problem discussed in this chapter.

The DCG ranking criterion was introduced by Järvelin and Kekäläinen [2000], and has been used and discussed in a number of subsequent studies, in particular Cossock and Zhang [2008] who consider a subset ranking problem formulated in terms of DCG, for which they consider a regression-based solution.

10.9 Exercises

10.1 Uniform margin-bound for ranking. Use theorem 10.1 to derive a margin-based learning bound for ranking that holds uniformly for all $\rho > 0$ (see similar binary classification bounds of theorem 5.9 and exercise 5.2).

10.2 On-line ranking. Give an on-line version of the SVM-based ranking algorithm presented in section 10.3.

10.3 Empirical margin loss of RankBoost. Derive an upper bound on the empirical pairwise ranking margin loss of RankBoost similar to that of theorem 7.7 for AdaBoost.

10.4 Margin maximization and RankBoost. Give an example showing that RankBoost does not achieve the maximum margin, as in the case of AdaBoost.

10.5 RankPerceptron. Adapt the Perceptron algorithm to derive a pairwise ranking algorithm based on a linear scoring function. Assume that the training sample is linear separable for pairwise ranking. Give an upper bound on the number of updates made by the algorithm in terms of the ranking margin.

10.6 Margin-maximization ranking. Give a linear programming (LP) algorithm returning a linear hypothesis for pairwise ranking based on margin maximization.

10.7 Bipartite ranking. Suppose that we use a binary classifier for ranking in the bipartite setting. Prove that if the error of the binary classifier is ϵ, then that of the ranking it induces is also at most ϵ. Show that the converse does not hold.

10.8 Multipartite ranking. Consider the ranking scenario in a k-partite setting where $\mathcal{X}$ is partitioned into k subsets $\mathcal{X}_1, \ldots, \mathcal{X}_k$ with $k \geq 1$. The bipartite case ($k = 2$) is already specifically examined in the chapter. Give a precise formulation of the problem in terms of k distributions. Does RankBoost admit an efficient implementation in this case? Give the pseudocode of the algorithm.

10.9 Deviation bound for the AUC. Let h be a fixed scoring function used to rank the points of $\mathcal{X}$. Use Hoeffding's bound to show that with high probability the AUC of h for a finite sample is close to its average.

10.10 k-partite weight function. Show how the weight function ω can be defined so that L_ω encodes the natural loss function associated to a k-partite ranking scenario.

11 Regression

This chapter discusses in depth the learning problem of *regression*, which consists of using data to predict, as closely as possible, the correct real-valued labels of the points or items considered. Regression is a common task in machine learning with a variety of applications, which justifies the specific chapter we reserve to its analysis.

The learning guarantees presented in the previous sections focused largely on classification problems. Here we present generalization bounds for regression, both for finite and infinite hypothesis sets. Several of these learning bounds are based on the familiar notion of Rademacher complexity, which is useful for characterizing the complexity of hypothesis sets in regression as well. Others are based on a combinatorial notion of complexity tailored to regression that we will introduce, *pseudo-dimension*, which can be viewed as an extension of the VC-dimension to regression. We describe a general technique for reducing regression problems to classification and deriving generalization bounds based on the notion of pseudo-dimension. We present and analyze several regression algorithms, including *linear regression*, *kernel ridge regression*, *support-vector regression*, *Lasso*, and several on-line versions of these algorithms. We discuss in detail the properties of these algorithms, including the corresponding learning guarantees.

11.1 The problem of regression

We first introduce the learning problem of regression. Let $\mathcal{X}$ denote the input space and $\mathcal{Y}$ a measurable subset of $\mathbb{R}$. Here, we will adopt the stochastic scenario and will denote by $\mathcal{D}$ a distribution over $\mathcal{X} \times \mathcal{Y}$. As discussed in section 2.4.1, the deterministic scenario is a straightforward special case where input points admit a unique label determined by a target function $f\colon \mathcal{X} \to \mathcal{Y}$.

As in all supervised learning problems, the learner receives a labeled sample $S = \big((x_1, y_1), \ldots, (x_m, y_m)\big) \in (\mathcal{X} \times \mathcal{Y})^m$ drawn i.i.d. according to $\mathcal{D}$. Since the labels are real numbers, it is not reasonable to hope that the learner could predict precisely the correct label when it is unique, or precisely its average label. Instead, we can require that its predictions be close to the correct ones. This is the key difference between regression and classification: in regression, the measure of error is based on the magnitude of the difference between the real-valued label predicted and the true or correct one, and not based on the equality or inequality of these two values. We denote by $L\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ the *loss function* used to measure the magnitude of error. The most common loss function used in regression is the *squared loss* L_2 defined by $L(y, y') = |y' - y|^2$ for all $y, y' \in \mathcal{Y}$, or, more generally, an L_p loss defined by $L(y, y') = |y' - y|^p$, for some $p \geq 1$ and all $y, y' \in \mathcal{Y}$.

Given a hypothesis set $\mathcal{H}$ of functions mapping $\mathcal{X}$ to $\mathcal{Y}$, the regression problem consists of using the labeled sample S to find a hypothesis $h \in \mathcal{H}$ with small expected loss or generalization error $R(h)$ with respect to the target f:

$$R(h) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \big[L\big(h(x), y\big)\big]. \tag{11.1}$$

As in the previous chapters, the empirical loss or error of $h \in \mathcal{H}$ is denoted by $\widehat{R}_S(h)$ and defined by

$$\widehat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m} L\big(h(x_i), y_i\big). \tag{11.2}$$

In the common case where L is the squared loss, this represents the *mean squared error* of h on the sample S.

When the loss function L is bounded by some $M > 0$, that is $L(y', y) \leq M$ for all $y, y' \in \mathcal{Y}$ or, more strictly, $L(h(x), y) \leq M$ for all $h \in \mathcal{H}$ and $(x, y) \in \mathcal{X} \times \mathcal{Y}$, the problem is referred to as a *bounded regression problem.* Much of the theoretical results presented in the following sections are based on that assumption. The analysis of *unbounded regression problems* is technically more elaborate and typically requires some other types of assumptions.

11.2 Generalization bounds

This section presents learning guarantees for bounded regression problems. We start with the simple case of a finite hypothesis set.

11.2.1 Finite hypothesis sets

In the case of a finite hypothesis, we can derive a generalization bound for regression by a straightforward application of Hoeffding's inequality and the union bound.

Theorem 11.1 *Let L be a bounded loss function. Assume that the hypothesis set $\mathcal{H}$ is finite. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following inequality* *holds for all $h \in \mathcal{H}$:*

$$R(h) \leq \widehat{R}_S(h) + M\sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2m}}\,.$$

Proof: By Hoeffding's inequality, since L takes values in $[0, M]$, for any $h \in \mathcal{H}$, the following holds:

$$\mathbb{P}\Big[R(h) - \widehat{R}_S(h) > \epsilon\Big] \leq e^{-\frac{2m\epsilon^2}{M^2}}\,.$$

Thus, by the union bound, we can write

$$\mathbb{P}\Big[\exists h \in \mathcal{H} \colon R(h) - \widehat{R}_S(h) > \epsilon\Big] \leq \sum_{h \in \mathcal{H}} \mathbb{P}\Big[R(h) - \widehat{R}_S(h) > \epsilon\Big] \leq |\mathcal{H}| e^{-\frac{2m\epsilon^2}{M^2}}\,.$$

Setting the right-hand side to be equal to δ yields the statement of the theorem. □

With the same assumptions and using the same proof, a two-sided bound can be derived: with probability at least $1 - \delta$, for all $h \in \mathcal{H}$,

$$|R(h) - \widehat{R}_S(h)| \leq M\sqrt{\frac{\log |\mathcal{H}| + \log \frac{2}{\delta}}{2m}}\,.$$

These learning bounds are similar to those derived for classification. In fact, they coincide with the classification bounds given in the inconsistent case when $M = 1$. Thus, all the remarks made in that context apply identically here. In particular, a larger sample size m guarantees better generalization; the bound increases as a function of $\log |\mathcal{H}|$ and suggests selecting, for the same empirical error, a smaller hypothesis set. This is an instance of Occam's razor principle for regression. In the next sections, we present other instances of this principle for the general case of infinite hypothesis sets using the notions of Rademacher complexity and pseudo-dimension.

11.2.2 Rademacher complexity bounds

Here, we show how the Rademacher complexity bounds of theorem 3.3 can be used to derive generalization bounds for regression in the case of the family of L_p loss functions. We first show an upper bound for the Rademacher complexity of a relevant family of functions.

Proposition 11.2 (Rademacher complexity of μ-Lipschitz loss functions) *Let $L \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a non-negative loss upper bounded by $M > 0$ ($L(y, y') \leq M$ for all $y, y' \in \mathcal{Y}$) and such that for any fixed $y' \in \mathcal{Y}$, $y \mapsto L(y, y')$ is μ-Lipschitz for some $\mu > 0$. Then, for any sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, the Rademacher complexity of the family $\mathcal{G} = \{(x, y) \mapsto L(h(x), y) \colon h \in \mathcal{H}\}$ is upper bounded as follows:*

$$\widehat{\mathfrak{R}}_S(\mathcal{G}) \leq \mu\, \widehat{\mathfrak{R}}_S(\mathcal{H})\,.$$

Proof: Since for any fixed y_i, $y \mapsto L(y, y_i)$ is μ-Lipschitz, by Talagrand's contraction lemma (lemma 5.7), we can write

$$\widehat{R}_S(\mathcal{G}) = \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sum_{i=1}^{m} \sigma_i L(h(x_i), y_i) \right] \leq \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sum_{i=1}^{m} \sigma_i \mu\, h(x_i) \right] = \mu\, \widehat{\mathfrak{R}}_S(\mathcal{H}),$$

which completes the proof. □

Theorem 11.3 (Rademacher complexity regression bounds) *Let $L\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a non-negative loss upper bounded by $M > 0$ ($L(y, y') \leq M$ for all $y, y' \in \mathcal{Y}$) and such that for any fixed $y' \in \mathcal{Y}$, $y \mapsto L(y, y')$ is μ-Lipschitz for some $\mu > 0$.*

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \big[L(x, y) \big] \leq \frac{1}{m} \sum_{i=1}^{m} L(x_i, y_i) + 2\mu\, \mathfrak{R}_m(\mathcal{H}) + M \sqrt{\frac{\log \frac{1}{\delta}}{2m}}$$

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \big[L(x, y) \big] \leq \frac{1}{m} \sum_{i=1}^{m} L(x_i, y_i) + 2\mu\, \widehat{\mathfrak{R}}_S(\mathcal{H}) + 3M \sqrt{\frac{\log \frac{2}{\delta}}{2m}}\,.$$

Proof: Since for any fixed y_i, $y \mapsto L(y, y_i)$ is μ-Lipschitz, by Talagrand's contraction lemma (lemma 5.7), we can write

$$\widehat{R}_S(\mathcal{G}) = \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sum_{i=1}^{m} \sigma_i L(h(x_i), y_i) \right] \leq \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sum_{i=1}^{m} \sigma_i \mu\, h(x_i) \right] = \mu\, \widehat{\mathfrak{R}}_S(\mathcal{H}).$$

Combining this inequality with the general Rademacher complexity learning bound of theorem 3.3 completes the proof. □

Let $p \geq 1$ and assume that $|h(x) - y| \leq M$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and $h \in \mathcal{H}$. Then, since for any y' the function $y \mapsto |y - y'|^p$ is pM^{p-1}-Lipschitz for $(y - y') \in [-M, M]$, the theorem applies to any L_p-loss. As an example, for any $\delta > 0$, with probability at least $1 - \delta$ over a sample S of size m, each of the following inequalities holds for all $h \in \mathcal{H}$:

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[\big|h(x) - y\big|^p \Big] \leq \frac{1}{m} \sum_{i=1}^{m} \big|h(x_i) - y_i\big|^p + 2pM^{p-1} \mathfrak{R}_m(\mathcal{H}) + M^p \sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$

As in the case of classification, these generalization bounds suggest a trade-off between reducing the empirical error, which may require more complex hypothesis sets, and controlling the Rademacher complexity of $\mathcal{H}$, which may increase the empirical error. An important benefit of the last learning bound of the theorem is that it is data-dependent. This can lead to more accurate learning guarantees. The upper bounds on $\mathfrak{R}_m(\mathcal{H})$ or $\mathfrak{R}_S(\mathcal{H})$ for kernel-based hypotheses (theorem 6.12) can be used directly here to derive generalization bounds in terms of the trace of the kernel matrix or the maximum diagonal entry.

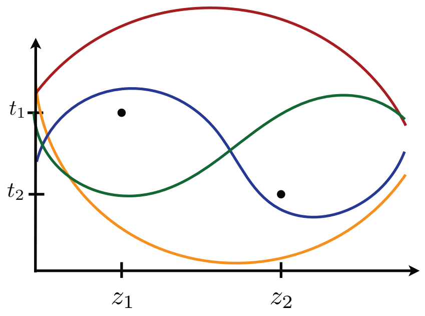


Figure 11.1
Illustration of the shattering of a set of two points $\{z_1, z_2\}$ with witnesses t_1 and t_2.

11.2.3 Pseudo-dimension bounds

As previously discussed in the case of classification, it is sometimes computationally hard to estimate the empirical Rademacher complexity of a hypothesis set. In chapter 3, we introduce other measures of the complexity of a hypothesis set such as the VC-dimension, which are purely combinatorial and typically easier to compute or upper bound. However, the notion of shattering or that of VC-dimension introduced for binary classification are not readily applicable to real-valued hypothesis classes.

We first introduce a new notion of *shattering* for families of real-valued functions. As in previous chapters, we will use the notation $\mathcal{G}$ for a family of functions, whenever we intend to later interpret it (at least in some cases) as the family of loss functions associated to some hypothesis set $\mathcal{H}$: $\mathcal{G} = \{z = (x, y) \mapsto L(h(x), y) : h \in \mathcal{H}\}$.

Definition 11.4 (Shattering) *Let $\mathcal{G}$ be a family of functions from a set $\mathcal{Z}$ to $\mathbb{R}$. A set $\{z_1, \ldots, z_m\} \subseteq \mathcal{X}$ is said to be* shattered *by $\mathcal{G}$ if there exist $t_1, \ldots, t_m \in \mathbb{R}$ such that,*

$$\left| \left\{ \begin{bmatrix} \operatorname{sgn}\big(g(z_1) - t_1\big) \\ \vdots \\ \operatorname{sgn}\big(g(z_m) - t_m\big) \end{bmatrix} : g \in \mathcal{G} \right\} \right| = 2^m .$$

When they exist, the threshold values $t_1, \ldots, t_m$ are said to witness *the shattering.*

Thus, $\{z_1, \ldots, z_m\}$ is shattered if for some witnesses $t_1, \ldots, t_m$, the family of functions $\mathcal{G}$ is rich enough to contain a function going above a subset $\mathcal{A}$ of the set of points $\mathcal{I} = \{(z_i, t_i) : i \in [m]\}$ and below the others $(\mathcal{I} - \mathcal{A})$, for any choice of the subset $\mathcal{A}$. Figure 11.1 illustrates this shattering in a simple case. The notion of shattering naturally leads to the following definition.

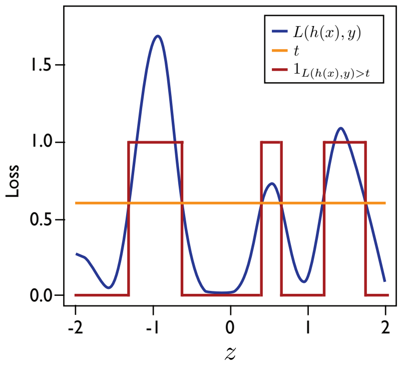


Figure 11.2
A function $g\colon z = (x, y) \mapsto L(h(x), y)$ (in blue) defined as the loss of some fixed hypothesis $h \in \mathcal{H}$, and its thresholded version $(x, y) \mapsto 1_{L(h(x),y)>t}$ (in red) with respect to the threshold t (in yellow).

Definition 11.5 (Pseudo-dimension) *Let $\mathcal{G}$ be a family of functions mapping from $\mathcal{X}$ to $\mathbb{R}$. Then, the* pseudo-dimension of $\mathcal{G}$, *denoted by* $\text{Pdim}(\mathcal{G})$, *is the size of the largest set shattered by $\mathcal{G}$.*

By definition of the shattering just introduced, the notion of pseudo-dimension of a family of real-valued functions $\mathcal{G}$ coincides with that of the VC-dimension of the corresponding thresholded functions mapping $\mathcal{X}$ to $\{0, 1\}$:

$$\text{Pdim}(\mathcal{G}) = \text{VCdim}\Big(\big\{(x, t) \mapsto 1_{(g(x)-t)>0}\colon g \in \mathcal{G}\big\}\Big) . \tag{11.3}$$

Figure 11.2 illustrates this interpretation. In view of this interpretation, the following two results follow directly the properties of the VC-dimension.

Theorem 11.6 *The pseudo-dimension of hyperplanes in $\mathbb{R}^N$ is given by*

$$\text{Pdim}(\{\mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} + b\colon \mathbf{w} \in \mathbb{R}^N, b \in \mathbb{R}\}) = N + 1 .$$

Theorem 11.7 *The pseudo-dimension of a vector space of real-valued functions $\mathcal{H}$ is equal to the dimension of the vector space:*

$$\text{Pdim}(\mathcal{H}) = \dim(\mathcal{H}) .$$

The following theorem gives a generalization bound for bounded regression in terms of the pseudo-dimension of a family of loss function $\mathcal{G} = \{z = (x, y) \mapsto L(h(x), y)\colon h \in \mathcal{H}\}$ associated to a hypothesis set $\mathcal{H}$. The key technique to derive these bounds consists of reducing the problem to that of classification by making use of the following general identity for the expectation of a random variable X:

$$\mathbb{E}[X] = -\int_{-\infty}^{0} \mathbb{P}[X < t]dt + \int_{0}^{+\infty} \mathbb{P}[X > t]dt , \tag{11.4}$$

which holds by definition of the Lebesgue integral. In particular, for any distribution $\mathcal{D}$ and any non-negative measurable function f, we can write

$$\mathop{\mathbb{E}}_{z\sim\mathcal{D}}[f(z)] = \int_0^\infty \mathop{\mathbb{P}}_{z\sim\mathcal{D}}[f(z) > t]dt\,. \tag{11.5}$$

Theorem 11.8 *Let $\mathcal{H}$ be a family of real-valued functions and $\mathcal{G} = \{(x, y) \mapsto L(h(x), y)\colon h \in \mathcal{H}\}$ the family of loss functions associated to $\mathcal{H}$. Assume that* $\mathrm{Pdim}(\mathcal{G}) = d$ *and that the loss function L is non-negative and bounded by M. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of am i.i.d. sample S of size m drawn from $\mathcal{D}^m$, the following inequality holds for all $h \in \mathcal{H}$:*

$$R(h) \leq \widehat{R}_S(h) + M\sqrt{\frac{2d\log\frac{em}{d}}{m}} + M\sqrt{\frac{\log\frac{1}{\delta}}{2m}}\,. \tag{11.6}$$

Proof: Let S be a sample of size m drawn i.i.d. according to $\mathcal{D}$ and let $\widehat{\mathcal{D}}$ denote the empirical distribution defined by S. For any $h \in \mathcal{H}$ and $t \geq 0$, we denote by $c(h, t)$ the classifier defined by $c(h, t)\colon (x, y) \mapsto 1_{L(h(x),y)>t}$. The error of $c(h, t)$ can be defined by

$$R(c(h,t)) = \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}}[c(h,t)(x,y) = 1] = \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}}[L(h(x),y) > t],$$

and, similarly, its empirical error is $\widehat{R}_S(c(h,t)) = \mathbb{P}_{(x,y)\sim\widehat{\mathcal{D}}}[L(h(x),y) > t]$.

Now, in view of the identity (11.5) and the fact that the loss function L is bounded by M, we can write:

$$\begin{aligned}
|R(h) - \widehat{R}_S(h)| &= \left|\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[L(h(x),y)] - \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}}[L(h(x),y)]\right| \\
&= \left|\int_0^M \left(\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}}[L(h(x),y) > t] - \mathop{\mathbb{P}}_{(x,y)\sim\widehat{\mathcal{D}}}[L(h(x),y) > t]\right) dt\right| \\
&\leq M \sup_{t\in[0,M]} \left|\mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}}[L(h(x),y) > t] - \mathop{\mathbb{P}}_{(x,y)\sim\widehat{\mathcal{D}}}[L(h(x),y) > t]\right| \\
&= M \sup_{t\in[0,M]} \left|R(c(h,t)) - \widehat{R}_S(c(h,t))\right|\,.
\end{aligned}$$

This implies the following inequality:

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}} |R(h) - \widehat{R}_S(h)| > \epsilon\right] \leq \mathbb{P}\left[\sup_{\substack{h\in\mathcal{H}\\ t\in[0,M]}} \left|R(c(h,t)) - \widehat{R}_S(c(h,t))\right| > \frac{\epsilon}{M}\right].$$

The right-hand side can be bounded using a standard generalization bound for classification (corollary 3.19) in terms of the VC-dimension of the family of hy-

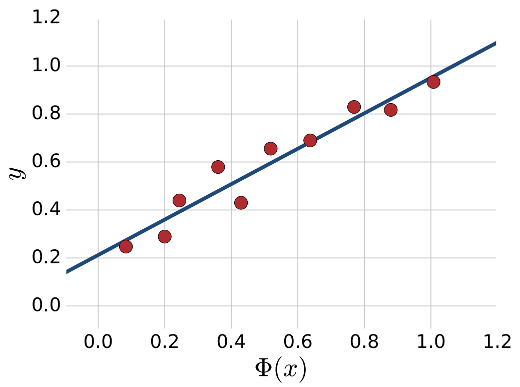


Figure 11.3
For $N = 1$, linear regression consists of finding the line of best fit, measured in terms of the squared loss.

potheses $\{c(h, t): h \in \mathcal{H}, t \in [0, M]\}$, which, by definition of the pseudo-dimension, is precisely $\text{Pdim}(\mathcal{G}) = d$. The resulting bound coincides with (11.6). $\square$

The notion of pseudo-dimension is suited to the analysis of regression as demonstrated by the previous theorem; however, it is not a scale-sensitive notion. There exists an alternative complexity measure, the *fat-shattering dimension*, that is scale-sensitive and that can be viewed as a natural extension of the pseudo-dimension. Its definition is based on the notion of γ-shattering.

Definition 11.9 (γ-shattering) *Let $\mathcal{G}$ be a family of functions from $\mathcal{Z}$ to $\mathbb{R}$ and let $\gamma > 0$. A set $\{z_1, \ldots, z_m\} \subseteq \mathcal{X}$ is said to be* γ-shattered *by $\mathcal{G}$ if there exist $t_1, \ldots, t_m \in \mathbb{R}$ such that for all $\mathbf{y} \in \{-1, +1\}^m$, there exists $g \in \mathcal{G}$ such that:*

$$\forall i \in [m], \quad y_i(g(z_i) - t_i) \geq \gamma\,.$$

Thus, $\{z_1, \ldots, z_m\}$ is γ-shattered if for some witnesses $t_1, \ldots, t_m$, the family of functions $\mathcal{G}$ is rich enough to contain a function going at least γ above a subset $\mathcal{A}$ of the set of points $\mathcal{I} = \{(z_i, t_i): i \in [m]\}$ and at least γ below the others $(\mathcal{I} - \mathcal{A})$, for any choice of the subset $\mathcal{A}$.

Definition 11.10 (γ-fat-dimension) *The* γ-fat-dimension *of $\mathcal{G}$, $\text{fat}_\gamma(\mathcal{G})$, is the size of the largest set that is γ-shattered by $\mathcal{G}$.*

Finer generalization bounds than those based on the pseudo-dimension can be derived in terms of the γ-fat-dimension. However, the resulting learning bounds, are not more informative than those based on the Rademacher complexity, which is also a scale-sensitive complexity measure. Thus, we will not detail an analysis based on the γ-fat-dimension.

11.3 Regression algorithms

The results of the previous sections show that, for the same empirical error, hypothesis sets with smaller complexity measured in terms of the Rademacher complexity or in terms of pseudo-dimension benefit from better generalization guarantees. One family of functions with relatively small complexity is that of linear hypotheses. In this section, we describe and analyze several algorithms based on that hypothesis set: *linear regression*, *kernel ridge regression* (KRR), *support vector regression* (SVR), and *Lasso*. These algorithms, in particular the last three, are extensively used in practice and often lead to state-of-the-art performance results.

11.3.1 Linear regression

We start with the simplest algorithm for regression known as *linear regression*. Let $\mathbf{\Phi}\colon \mathcal{X} \to \mathbb{R}^N$ be a feature mapping from the input space $\mathcal{X}$ to $\mathbb{R}^N$ and consider the family of linear hypotheses

$$\mathcal{H} = \{x \mapsto \mathbf{w} \cdot \mathbf{\Phi}(x) + b \colon \mathbf{w} \in \mathbb{R}^N, b \in \mathbb{R}\}. \tag{11.7}$$

Linear regression consists of seeking a hypothesis in $\mathcal{H}$ with the smallest empirical mean squared error. Thus, for a sample $S = \big((x_1, y_1), \ldots, (x_m, y_m)\big) \in (\mathcal{X} \times \mathcal{Y})^m$, the following is the corresponding optimization problem:

$$\min_{\mathbf{w}, b} \frac{1}{m} \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{\Phi}(x_i) + b - y_i)^2 . \tag{11.8}$$

Figure 11.3 illustrates the algorithm in the simple case where $N = 1$. The optimization problem admits the simpler formulation:

$$\min_{\mathbf{W}} F(\mathbf{W}) = \frac{1}{m} \|\mathbf{X}^\top \mathbf{W} - \mathbf{Y}\|^2, \tag{11.9}$$

using the notation $\mathbf{X} = \begin{bmatrix} \Phi(x_1) & \ldots & \Phi(x_m) \\ 1 & \ldots & 1 \end{bmatrix}$, $\mathbf{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \\ b \end{bmatrix}$ and $\mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$. The objective function F is convex, by composition of the convex function $\mathbf{u} \mapsto \|\mathbf{u}\|^2$ with the affine function $\mathbf{W} \mapsto \mathbf{X}^\top \mathbf{W} - \mathbf{Y}$, and it is differentiable. Thus, F admits a global minimum at $\mathbf{W}$ if and only if $\nabla F(\mathbf{W}) = 0$, that is if and only if

$$\frac{2}{m} \mathbf{X}(\mathbf{X}^\top \mathbf{W} - \mathbf{Y}) = 0 \Leftrightarrow \mathbf{X}\mathbf{X}^\top \mathbf{W} = \mathbf{X}\mathbf{Y}. \tag{11.10}$$

When $\mathbf{X}\mathbf{X}^\top$ is invertible, this equation admits a unique solution. Otherwise, the equation admits a family of solutions that can be given in terms of the pseudo-inverse of matrix $\mathbf{X}\mathbf{X}^\top$ (see appendix A) by $\mathbf{W} = (\mathbf{X}\mathbf{X}^\top)^\dagger \mathbf{X}\mathbf{Y} + (I - (\mathbf{X}\mathbf{X}^\top)^\dagger (\mathbf{X}\mathbf{X}^\top))\mathbf{W}_0$, where $\mathbf{W}_0$ is an arbitrary matrix in $\mathbb{R}^{N \times N}$. Among these, the solution $\mathbf{W} = (\mathbf{X}\mathbf{X}^\top)^\dagger \mathbf{X}\mathbf{Y}$ is the one with the minimal norm and is often preferred for that reason. Thus, we will write the solutions as

$$\mathbf{W} = \begin{cases} (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{Y} & \text{if } \mathbf{X}\mathbf{X}^\top \text{ is invertible,} \\ (\mathbf{X}\mathbf{X}^\top)^\dagger \mathbf{X}\mathbf{Y} & \text{otherwise.} \end{cases} \tag{11.11}$$

The matrix $\mathbf{X}\mathbf{X}^\top$ can be computed in $O(mN^2)$. The cost of its inversion or that of computing its pseudo-inverse is in $O(N^3)$.[19] Finally, the multiplication with $\mathbf{X}$ and $\mathbf{Y}$ takes $O(mN^2)$. Therefore, the overall complexity of computing the solution $\mathbf{W}$ is in $O(mN^2 + N^3)$. Thus, when the dimension of the feature space N is not too large, the solution can be computed efficiently.

While linear regression is simple and admits a straightforward implementation, it does not benefit from a strong generalization guarantee, since it is limited to minimizing the empirical error without controlling the norm of the weight vector and without any other regularization. Its performance is also typically poor in most applications. The next sections describe algorithms with both better theoretical guarantees and improved performance in practice.

11.3.2 Kernel ridge regression

We first present a learning guarantee for regression with bounded linear hypotheses in a feature space defined by a PDS kernel. This will provide a strong theoretical support for the *kernel ridge regression* algorithm presented in this section. The learning bounds of this section are given for the squared loss. Thus, in particular, the generalization error of a hypothesis h is defined by $R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\big[(h(x) - y)^2\big]$.

Theorem 11.11 *Let $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel, $\Phi\colon \mathcal{X} \to \mathbb{H}$ a feature mapping associated to K, and $\mathcal{H} = \{x \mapsto \mathbf{w} \cdot \mathbf{\Phi}(x) \colon \|\mathbf{w}\|_{\mathbb{H}} \le \Lambda\}$. Assume that there exists $r > 0$ such that $K(x, x) \le r^2$ and $M > 0$ such that $|h(x) - y| < M$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, each of the following inequalities holds for all $h \in \mathcal{H}$:*

$$R(h) \le \widehat{R}_S(h) + 4M\sqrt{\frac{r^2\Lambda^2}{m}} + M^2\sqrt{\frac{\log\frac{1}{\delta}}{2m}}$$

$$R(h) \le \widehat{R}_S(h) + \frac{4M\Lambda\sqrt{\operatorname{Tr}[\mathbf{K}]}}{m} + 3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2m}}\,.$$

[19] In the analysis of the computational complexity of the algorithms discussed in this chapter, the cubic-time complexity of matrix inversion can be replaced by a more favorable complexity $O(N^{2+\omega})$, with $\omega = .376$ using asymptotically faster matrix inversion methods such as that of Coppersmith and Winograd.

Proof: By the bound on the empirical Rademacher complexity of kernel-based hypotheses (theorem 6.12), the following holds for any sample S of size m:

$$\widehat{\mathfrak{R}}_S(\mathcal{H}) \leq \frac{\Lambda\sqrt{\operatorname{Tr}[\mathbf{K}]}}{m} \leq \sqrt{\frac{r^2\Lambda^2}{m}},$$

which implies that $\mathfrak{R}_m(\mathcal{H}) \leq \sqrt{\frac{r^2\Lambda^2}{m}}$. Combining these inequalities with the learning bounds of Theorem 11.3 yield immediately the inequalities claimed. □

The learning bounds of the theorem suggests minimizing a trade-off between the empirical squared loss (first term on the right-hand side), and the norm of the weight vector (upper bound Λ on the norm appearing in the second term), or equivalently the norm squared. Kernel ridge regression is defined by the minimization of an objective function that has precisely this form and thus is directly motivated by the theoretical analysis just presented:

$$\min_{\mathbf{w}} F(\mathbf{w}) = \lambda\|\mathbf{w}\|^2 + \sum_{i=1}^{m} \left(\mathbf{w} \cdot \mathbf{\Phi}(x_i) - y_i\right)^2. \tag{11.12}$$

Here, λ is a positive parameter determining the trade-off between the regularization term $\|\mathbf{w}\|^2$ and the empirical mean squared error. The objective function differs from that of linear regression only by the first term, which controls the norm of $\mathbf{w}$. As in the case of linear regression, the problem can be rewritten in a more compact form as

$$\min_{\mathbf{W}} F(\mathbf{W}) = \lambda\|\mathbf{W}\|^2 + \|\mathbf{X}^\top\mathbf{W} - \mathbf{Y}\|^2, \tag{11.13}$$

where $\mathbf{X} \in \mathbb{R}^{N\times m}$ is the matrix formed by the feature vectors, $\mathbf{X} = \left[\begin{smallmatrix}\Phi(x_1) & \ldots & \Phi(x_m)\end{smallmatrix}\right]$, $\mathbf{W} = \mathbf{w}$, and $\mathbf{Y} = (y_1, \ldots, y_m)^\top$. Here too, F is convex, by the convexity of $\mathbf{w} \mapsto \|\mathbf{w}\|^2$ and that of the sum of two convex functions, and is differentiable. Thus F admits a global minimum at $\mathbf{W}$ if and only if

$$\nabla F(\mathbf{W}) = 0 \Leftrightarrow (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})\mathbf{W} = \mathbf{X}\mathbf{Y} \Leftrightarrow \mathbf{W} = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{Y}. \tag{11.14}$$

Note that the matrix $\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}$ is always invertible, since its eigenvalues are the sum of the non-negative eigenvalues of the symmetric positive semidefinite matrix $\mathbf{X}\mathbf{X}^\top$ and $\lambda > 0$. Thus, kernel ridge regression admits a closed-form solution.

An alternative formulation of the optimization problem for kernel ridge regression equivalent to (11.12) is

$$\min_{\mathbf{w}} \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{\Phi}(x_i) - y_i)^2 \quad \text{subject to: } \|\mathbf{w}\|^2 \leq \Lambda^2.$$

This makes the connection with the bounded linear hypothesis set of theorem 11.11 even more evident. Using slack variables ξ_i, for all $i \in [m]$, the problem can be equivalently written as

$$\min_{\mathbf{w}} \sum_{i=1}^{m} \xi_i^2 \quad \text{subject to: } (\|\mathbf{w}\|^2 \leq \Lambda^2) \wedge \big(\forall i \in [m],\ \xi_i = y_i - \mathbf{w} \cdot \mathbf{\Phi}(x_i)\big).$$

This is a convex optimization problem with differentiable objective function and constraints. To derive the equivalent dual problem, we introduce the Lagrangian $\mathcal{L}$, which is defined for all $\boldsymbol{\xi}, \mathbf{w}, \boldsymbol{\alpha}'$, and $\lambda \geq 0$ by

$$\mathcal{L}(\boldsymbol{\xi}, \mathbf{w}, \boldsymbol{\alpha}', \lambda) = \sum_{i=1}^{m} \xi_i^2 + \sum_{i=1}^{m} \alpha_i'(y_i - \xi_i - \mathbf{w} \cdot \mathbf{\Phi}(x_i)) + \lambda(\|\mathbf{w}\|^2 - \Lambda^2)\,.$$

The KKT conditions lead to the following equalities:

$$\begin{aligned}
&\nabla_{\mathbf{w}}\mathcal{L} = -\sum_{i=1}^{m} \alpha_i' \mathbf{\Phi}(x_i) + 2\lambda \mathbf{w} = 0 && \implies && \mathbf{w} = \frac{1}{2\lambda}\sum_{i=1}^{m} \alpha_i' \mathbf{\Phi}(x_i)\\
&\nabla_{\xi_i}\mathcal{L} = 2\xi_i - \alpha_i' = 0 && \implies && \xi_i = \alpha_i'/2\\
&\forall i \in [m], \alpha_i'(y_i - \xi_i - \mathbf{w} \cdot \mathbf{\Phi}(x_i)) = 0\\
&\lambda(\|\mathbf{w}\|^2 - \Lambda^2) = 0.
\end{aligned}$$

Plugging in the expressions of $\mathbf{w}$ and ξ_is in that of $\mathcal{L}$ gives

$$\begin{aligned}
\mathcal{L} &= \sum_{i=1}^{m} \frac{\alpha_i'^2}{4} + \sum_{i=1}^{m} \alpha_i' y_i - \sum_{i=1}^{m} \frac{\alpha_i'^2}{2} - \frac{1}{2\lambda}\sum_{i,j=1}^{m} \alpha_i' \alpha_j' \mathbf{\Phi}(x_i)^\top \mathbf{\Phi}(x_j)\\
&\quad + \lambda\Big(\frac{1}{4\lambda^2}\|\sum_{i=1}^{m} \alpha_i' \mathbf{\Phi}(x_i)\|^2 - \Lambda^2\Big)\\
&= -\frac{1}{4}\sum_{i=1}^{m} \alpha_i'^2 + \sum_{i=1}^{m} \alpha_i' y_i - \frac{1}{4\lambda}\sum_{i,j=1}^{m} \alpha_i' \alpha_j' \mathbf{\Phi}(x_i)^\top \mathbf{\Phi}(x_j) - \lambda\Lambda^2\\
&= -\lambda\sum_{i=1}^{m} \alpha_i^2 + 2\sum_{i=1}^{m} \alpha_i y_i - \sum_{i,j=1}^{m} \alpha_i \alpha_j \mathbf{\Phi}(x_i)^\top \mathbf{\Phi}(x_j) - \lambda\Lambda^2,
\end{aligned}$$

with $\alpha_i' = 2\lambda\alpha_i$. Thus, the equivalent dual optimization problem for KRR can be written as follows:

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^m} -\lambda \boldsymbol{\alpha}^\top \boldsymbol{\alpha} + 2\boldsymbol{\alpha}^\top \mathbf{Y} - \boldsymbol{\alpha}^\top (\mathbf{X}^\top \mathbf{X})\boldsymbol{\alpha}\,, \tag{11.15}$$

or, more compactly, as

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^m} G(\boldsymbol{\alpha}) = -\boldsymbol{\alpha}^\top (\mathbf{K} + \lambda \mathbf{I})\boldsymbol{\alpha} + 2\boldsymbol{\alpha}^\top \mathbf{Y}\,, \tag{11.16}$$

where $\mathbf{K} = \mathbf{X}^\top \mathbf{X}$ is the kernel matrix associated to the training sample. The objective function G is concave and differentiable. The optimal solution is obtained by differentiating the function and setting it to zero:

$$\nabla G(\boldsymbol{\alpha}) = 0 \iff 2(\mathbf{K} + \lambda \mathbf{I})\boldsymbol{\alpha} = 2\mathbf{Y} \iff \boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{Y}\,. \tag{11.17}$$

Note that $(\mathbf{K}+\lambda\mathbf{I})$ is invertible, since its eigenvalues are the sum of the eigenvalues of the SPSD matrix $\mathbf{K}$ and $\lambda > 0$. Thus, as in the primal case, the dual optimization problem admits a closed-form solution. By the first KKT equation, $\mathbf{w}$ can be determined from $\boldsymbol{\alpha}$ by

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \boldsymbol{\Phi}(\mathbf{x}_i) = \mathbf{X}\boldsymbol{\alpha} = \mathbf{X}(\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{Y}. \tag{11.18}$$

The hypothesis h solution can be given as follows in terms of $\boldsymbol{\alpha}$:

$$\forall x \in \mathcal{X}, \quad h(x) = \mathbf{w} \cdot \boldsymbol{\Phi}(x) = \sum_{i=1}^{m} \alpha_i K(x_i, x)\,. \tag{11.19}$$

Note that the form of the solution, $h = \sum_{i=1}^{m} \alpha_i K(x_i, \cdot)$, could be immediately predicted using the Representer theorem, since the objective function minimized by KRR falls within the general framework of theorem 6.11. This also could show that $\mathbf{w}$ could be written as $\mathbf{w} = \mathbf{X}\alpha$. This fact, combined with the following simple lemma, can be used to determine $\boldsymbol{\alpha}$ in a straightforward manner, without the intermediate derivation of the dual problem.

Lemma 11.12 *The following identity holds for any matrix* $\mathbf{X}$*:*

$$(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I})^{-1}\mathbf{X} = \mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda \mathbf{I})^{-1}\,.$$

Proof: Observe that $(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})\mathbf{X} = \mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})$. Left-multiplying by $(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}$ this equality and right-multiplying it by $(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}$ yields the statement of the lemma. □

Now, using this lemma, the primal solution of $\mathbf{w}$ can be rewritten as follows:

$$\mathbf{w} = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{Y} = \mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{Y} = \mathbf{X}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{Y}.$$

Comparing with $\mathbf{w} = \mathbf{X}\alpha$ gives immediately $\boldsymbol{\alpha} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{Y}$.

Our presentation of the KRR algorithm was given for linear hypotheses with no offset, that is we implicitly assumed $b = 0$. It is common to use this formulation and to extend it to the general case by augmenting the feature vector $\boldsymbol{\Phi}(x)$ with an extra component equal to one for all $x \in \mathcal{X}$ and the weight vector $\mathbf{w}$ with an extra component $b \in \mathbb{R}$. For the augmented feature vector $\boldsymbol{\Phi}'(x) \in \mathbb{R}^{N+1}$ and weight vector $\mathbf{w}' \in \mathbb{R}^{N+1}$, we have $\mathbf{w}' \cdot \boldsymbol{\Phi}'(x) = \mathbf{w} \cdot \boldsymbol{\Phi}(x) + b$. Nevertheless, this formulation does not coincide with the general KRR algorithm where a solution of the form $x \mapsto \mathbf{w} \cdot \boldsymbol{\Phi}(x) + b$ is sought. This is because for the general KRR, the regularization term is $\lambda\|\mathbf{w}\|$, while for the extension just described it is $\lambda\|\mathbf{w}'\|$.

Table 11.1
Comparison of the running-time complexity of KRR for computing the solution or the prediction value of a point in both the primal and the dual case. κ denotes the time complexity of computing a kernel value; for polynomial and Gaussian kernels, $\kappa = O(N)$.

	Solution	Prediction
Primal	$O(mN^2 + N^3)$	$O(N)$
Dual	$O(\kappa m^2 + m^3)$	$O(\kappa m)$

In both the primal and dual cases, KRR admits a closed-form solution. Table 11.1 gives the time complexity of the algorithm for computing the solution and the one for determining the prediction value of a point in both cases. In the primal case, determining the solution $\mathbf{w}$ requires computing matrix $\mathbf{X}\mathbf{X}^\top$, which takes $O(mN^2)$, the inversion of $(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})$, which is in $O(N^3)$, and multiplication with $\mathbf{X}$, which is in $O(mN^2)$. Prediction requires computing the inner product of $\mathbf{w}$ with a feature vector of the same dimension that can be achieved in $O(N)$. The dual solution first requires computing the kernel matrix $\mathbf{K}$. Let κ be the maximum cost of computing $K(x, x')$ for all pairs $(x, x') \in \mathcal{X} \times \mathcal{X}$. Then, $\mathbf{K}$ can be computed in $O(\kappa m^2)$. The inversion of matrix $\mathbf{K} + \lambda\mathbf{I}$ can be achieved in $O(m^3)$ and multiplication with $\mathbf{Y}$ takes $O(m^2)$. Prediction requires computing the vector $(K(x_1, x), \ldots, K(x_m, x))^\top$ for some $x \in \mathcal{X}$, which requires $O(\kappa m)$, and the inner product with $\boldsymbol{\alpha}$, which is in $O(m)$.

Thus, in both cases, the main step for computing the solution is a matrix inversion, which takes $O(N^3)$ in the primal case, $O(m^3)$ in the dual case. When the dimension of the feature space is relatively small, solving the primal problem is advantageous, while for high-dimensional spaces and medium-sized training sets, solving the dual is preferable. Note that for relatively large matrices, the space complexity could also be an issue: the size of relatively large matrices could be prohibitive for memory storage and the use of external memory could significantly affect the running time of the algorithm.

For sparse matrices, there exist several techniques for faster computations of the matrix inversion. This can be useful in the primal case where the features can be relatively sparse. On the other hand, the kernel matrix $\mathbf{K}$ is typically dense; thus, there is less hope for benefiting from such techniques in the dual case. In such cases, or, more generally, to deal with the time and space complexity issues arising when m and N are large, approximation methods using low-rank approximations via the Nyström method or the partial Cholesky decomposition can be used very effectively.

The KRR algorithm admits several advantages: it benefits from favorable theoretical guarantees since it can be derived directly from the generalization bound we

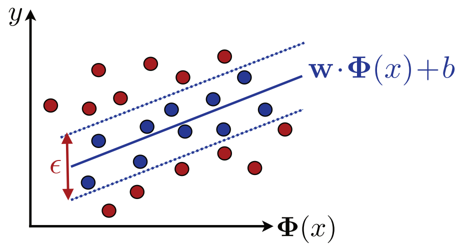


Figure 11.4
SVR attempts to fit a "tube" with width ϵ to the data. Training data within the "epsilon tube" (blue points) incur no loss.

presented; it admits a closed-form solution, which can make the analysis of many of its properties convenient; and it can be used with PDS kernels, which extends its use to non-linear regression solutions and more general features spaces. KRR also admits favorable stability properties that we discuss in chapter 14.

The algorithm can be generalized to learning a mapping from $\mathcal{X}$ to $\mathbb{R}^p$, $p > 1$. This can be done by formulating the problem as p independent regression problems, each consisting of predicting one of the p target components. Remarkably, the computation of the solution for this generalized algorithm requires only a single matrix inversion, e.g., $(\mathbf{K} + \lambda \mathbf{I})^{-1}$ in the dual case, regardless of the value of p.

One drawback of the KRR algorithm, in addition to the computational issues for determining the solution for relatively large matrices, is the fact that the solution it returns is typically not sparse. The next two sections present two sparse algorithms for linear regression.

11.3.3 Support vector regression

In this section, we present the *support vector regression* (SVR) algorithm, which is inspired by the SVM algorithm presented for classification in chapter 5. The main idea of the algorithm consists of fitting a tube of width $\epsilon > 0$ to the data, as illustrated by figure 11.4. As in binary classification, this defines two sets of points: those falling inside the tube, which are ϵ-close to the function predicted and thus not penalized, and those falling outside, which are penalized based on their distance to the predicted function, in a way that is similar to the penalization used by SVMs in classification.

Using a hypothesis set $\mathcal{H}$ of linear functions: $\mathcal{H} = \{x \mapsto \mathbf{w} \cdot \mathbf{\Phi}(x) + b \colon \mathbf{w} \in \mathbb{R}^N, b \in \mathbb{R}\}$, where $\mathbf{\Phi}$ is the feature mapping corresponding some PDS kernel K, the optimization problem for SVR can be written as follows:

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \big|y_i - (\mathbf{w} \cdot \mathbf{\Phi}(x_i) + b)\big|_{\epsilon}, \tag{11.20}$$

where $|\cdot|_\epsilon$ denotes the *ϵ-insensitive loss*:

$$\forall y, y' \in \mathcal{Y}, \quad |y' - y|_\epsilon = \max(0, |y' - y| - \epsilon). \tag{11.21}$$

The use of this loss function leads to sparse solutions with a relatively small number of support vectors. Using slack variables $\xi_i \geq 0$ and $\xi'_i \geq 0$, $i \in [m]$, the optimization problem can be equivalently written as

$$\begin{aligned} \min_{\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}'} \ & \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^m (\xi_i + \xi'_i) \\ \text{subject to } & (\mathbf{w} \cdot \boldsymbol{\Phi}(x_i) + b) - y_i \leq \epsilon + \xi_i \\ & y_i - (\mathbf{w} \cdot \boldsymbol{\Phi}(x_i) + b) \leq \epsilon + \xi'_i \\ & \xi_i \geq 0, \xi'_i \geq 0, \ \forall i \in [m]. \end{aligned} \tag{11.22}$$

This is a convex quadratic program (QP) with affine constraints. Introducing the Lagrangian and applying the KKT conditions leads to the following equivalent dual problem in terms of the kernel matrix $\mathbf{K}$:

$$\begin{aligned} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}'} \ & -\epsilon(\boldsymbol{\alpha}' + \boldsymbol{\alpha})^\top \mathbf{1} + (\boldsymbol{\alpha}' - \boldsymbol{\alpha})^\top \mathbf{y} - \frac{1}{2}(\boldsymbol{\alpha}' - \boldsymbol{\alpha})^\top \mathbf{K}(\boldsymbol{\alpha}' - \boldsymbol{\alpha}) \\ \text{subject to: } & (\mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{C}) \wedge (\mathbf{0} \leq \boldsymbol{\alpha}' \leq \mathbf{C}) \wedge ((\boldsymbol{\alpha}' - \boldsymbol{\alpha})^\top \mathbf{1} = 0). \end{aligned} \tag{11.23}$$

Any PDS kernel K can be used with SVR, which extends the algorithm to non-linear regression solutions. Problem (11.23) is a convex QP similar to the dual problem of SVMs and can be solved using similar optimization techniques. The solutions $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ define the hypothesis h returned by SVR as follows:

$$\forall x \in \mathcal{X}, \quad h(x) = \sum_{i=1}^m (\alpha'_i - \alpha_i) K(\mathbf{x}_i, \mathbf{x}) + b, \tag{11.24}$$

where the offset b can be obtained from a point x_j with $0 < \alpha_j < C$ by

$$b = -\sum_{i=1}^m (\alpha'_i - \alpha_i) K(x_i, x_j) + y_j + \epsilon, \tag{11.25}$$

or from a point x_j with $0 < \alpha'_j < C$ via

$$b = -\sum_{i=1}^m (\alpha'_i - \alpha_i) K(x_i, x_j) + y_j - \epsilon. \tag{11.26}$$

By the complementarity conditions, for all $i \in [m]$, the following equalities hold:

$$\begin{aligned} \alpha_i \big((\mathbf{w} \cdot \boldsymbol{\Phi}(x_i) + b) - y_i - \epsilon - \xi_i\big) &= 0 \\ \alpha'_i \big((\mathbf{w} \cdot \boldsymbol{\Phi}(x_i) + b) - y_i + \epsilon + \xi'_i\big) &= 0. \end{aligned}$$

Thus, if $\alpha_i \neq 0$ or $\alpha'_i \neq 0$, that is if x_i is a support vector, then, either $(\mathbf{w} \cdot \mathbf{\Phi}(x_i) + b) - y_i - \epsilon = \xi_i$ holds or $y_i - (\mathbf{w} \cdot \mathbf{\Phi}(x_i) + b) - \epsilon = \xi'_i$. This shows that support vectors points lying outside the ϵ-tube. Of course, at most one of α_i or α'_i is non-zero for any point x_i: the hypothesis either overestimates or underestimates the true label by more than ϵ. For the points within the ϵ-tube, we have $\alpha_j = \alpha'_j = 0$; thus, these points do not contribute to the definition of the hypothesis returned by SVR. Thus, when the number of points inside the tube is relatively large, the hypothesis returned by SVR is relatively sparse. The choice of the parameter ϵ determines a trade-off between sparsity and accuracy: larger ϵ values provide sparser solutions, since more points can fall within the ϵ-tube, but may ignore too many key points for determining an accurate solution.

The following generalization bounds hold for the ϵ-insensitive loss and kernel-based hypotheses and thus for the SVR algorithm. We denote by $\mathcal{D}$ the distribution according to which sample points are drawn and by $\widehat{\mathcal{D}}$ the empirical distribution defined by a training sample of size m.

Theorem 11.13 *Let $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel, let $\Phi\colon \mathcal{X} \to \mathbb{H}$ be a feature mapping associated to K and let $\mathcal{H} = \{x \mapsto \mathbf{w} \cdot \mathbf{\Phi}(x)\colon \|\mathbf{w}\|_{\mathbb{H}} \leq \Lambda\}$. Assume that there exists $r > 0$ such that $K(x, x) \leq r^2$ and $M > 0$ such that $|h(x) - y| \leq M$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Fix $\epsilon > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, each of the following inequalities holds for all $h \in \mathcal{H}$,*

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\big[|h(x) - y|_\epsilon\big] \leq \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}}\big[|h(x) - y|_\epsilon\big] + 2\sqrt{\frac{r^2\Lambda^2}{m}} + M\sqrt{\frac{\log\frac{1}{\delta}}{2m}}$$

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\big[|h(x) - y|_\epsilon\big] \leq \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}}\big[|h(x) - y|_\epsilon\big] + \frac{2\Lambda\sqrt{\mathrm{Tr}[\mathbf{K}]}}{m} + 3M\sqrt{\frac{\log\frac{2}{\delta}}{2m}}.$$

Proof: Since for any $y' \in \mathcal{Y}$, the function $y \mapsto |y - y'|_\epsilon$ is 1-Lipschitz, the result follows Theorem 11.3 and the bound on the empirical Rademacher complexity of $\mathcal{H}$. □

These results provide theoretical guarantees for the SVR algorithm. Notice, however, that the theorem does not provide guarantees for the expected loss of the hypotheses in terms of the squared loss. For $0 < \epsilon < 1/4$, the inequality $|x|^2 \leq |x|_\epsilon$ holds for all x in $[-\eta'_\epsilon, -\eta_\epsilon] \cup [\eta_\epsilon, \eta'_\epsilon]$ with $\eta_\epsilon = \frac{1-\sqrt{1-4\epsilon}}{2}$ and $\eta'_\epsilon = \frac{1+\sqrt{1-4\epsilon}}{2}$. For small values of ϵ, $\eta_\epsilon \approx 0$ and $\eta'_\epsilon \approx 1$, thus, if $M = 2r\lambda \leq 1$, then, the squared loss can be upper bounded by the ϵ-insensitive loss for almost all values of $(h(x) - y)$ in $[-1, 1]$ and the theorem can be used to derive a useful generalization bound for the squared loss.

More generally, if the objective is to achieve a small squared loss, then, SVR can be modified by using the *quadratic ϵ-insensitive loss*, that is the square of the ϵ-insensitive loss, which also leads to a convex QP. We will refer by *quadratic SVR* to this version of the algorithm. Introducing the Lagrangian and applying the KKT conditions leads to the following equivalent dual optimization problem for quadratic SVR in terms of the kernel matrix $\mathbf{K}$:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}'} \; -\epsilon(\boldsymbol{\alpha}'+\boldsymbol{\alpha})^\top \mathbf{1} + (\boldsymbol{\alpha}'-\boldsymbol{\alpha})^\top \mathbf{y} - \frac{1}{2}(\boldsymbol{\alpha}'-\boldsymbol{\alpha})^\top \Big(\mathbf{K}+\frac{1}{C}\mathbf{I}\Big)(\boldsymbol{\alpha}'-\boldsymbol{\alpha}) \tag{11.27}$$

$$\text{subject to: } (\boldsymbol{\alpha} \geq \mathbf{0}) \wedge (\boldsymbol{\alpha}' \geq \mathbf{0}) \wedge (\boldsymbol{\alpha}'-\boldsymbol{\alpha})^\top \mathbf{1} = 0) \,.$$

Any PDS kernel K can be used with quadratic SVR, which extends the algorithm to non-linear regression solutions. Problem (11.27) is a convex QP similar to the dual problem of SVMs in the separable case and can be solved using similar optimization techniques. The solutions $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ define the hypothesis h returned by SVR as follows:

$$h(x) = \sum_{i=1}^m (\alpha'_i - \alpha_i) K(\mathbf{x}_i, \mathbf{x}) + b \,, \tag{11.28}$$

where the offset b can be obtained from a point x_j with $0 < \alpha_j < C$ or $0 < \alpha'_j < C$ exactly as in the case of SVR with (non-quadratic) ϵ-insensitive loss. Note that for $\epsilon = 0$, the quadratic SVR algorithm coincides with KRR as can be seen from the dual optimization problem (the additional constraint $(\boldsymbol{\alpha}'-\boldsymbol{\alpha})^\top \mathbf{1} = 0$ appears here due to use of an offset b). The following generalization bound holds for quadratic SVR. It can be shown in a way that is similar to the proof of theorem 11.13 using the fact that the quadratic ϵ-insensitive function $x \mapsto |x|_\epsilon^2$ is $2M$-Lipschitz over the interval $[-M, +M]$.

Theorem 11.14 *Let $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel, let $\Phi\colon \mathcal{X} \to \mathbb{H}$ be a feature mapping associated to K and let $\mathcal{H} = \{x \mapsto \mathbf{w} \cdot \boldsymbol{\Phi}(x)\colon \|\mathbf{w}\|_{\mathbb{H}} \leq \Lambda\}$. Assume that there exists $r > 0$ such that $K(x,x) \leq r^2$ and $M > 0$ such that $|h(x) - y| \leq M$ for all $(x,y) \in \mathcal{X} \times \mathcal{Y}$. Fix $\epsilon > 0$. Then, for any $\delta > 0$, with probability at least $1-\delta$, each of the following inequalities holds for all $h \in \mathcal{H}$,*

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \big[|h(x)-y|_\epsilon^2\big] \leq \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}} \big[|h(x)-y|_\epsilon^2\big] + 4M\sqrt{\frac{r^2\Lambda^2}{m}} + M^2\sqrt{\frac{\log\frac{1}{\delta}}{2m}}$$

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \big[|h(x)-y|_\epsilon^2\big] \leq \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}} \big[|h(x)-y|_\epsilon^2\big] + \frac{4M\Lambda\sqrt{\mathrm{Tr}[\mathbf{K}]}}{m} + 3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2m}}.$$

This theorem provides a strong justification for the quadratic SVR algorithm. Alternative convex loss functions can be used to define regression algorithms, in particular the *Huber loss* (see figure 11.5), which penalizes smaller errors quadratically and larger ones only linearly.

SVR admits several advantages: the algorithm is based on solid theoretical guarantees, the solution returned is sparse, and it allows a natural use of PDS kernels,

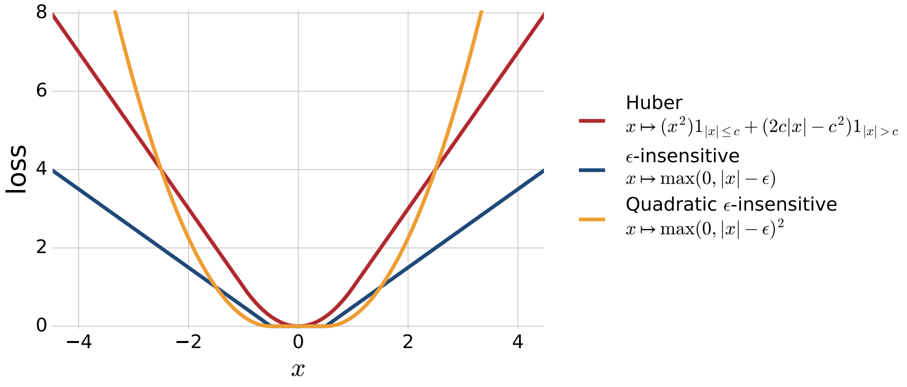


Figure 11.5
Alternative loss functions that can be used in conjunction with SVR.

which extend the algorithm to non-linear regression solutions. SVR also admits favorable stability properties that we discuss in chapter 14. However, one drawback of the algorithm is that it requires the selection of two parameters, C and ϵ. These can be selected via cross-validation, as in the case of SVMs, but this requires a relatively larger validation set. Some heuristics are often used to guide the search for their values: C is searched near the maximum value of the labels in the absence of an offset ($b = 0$) and for a normalized kernel, and ϵ is chosen close to the average difference of the labels. As already discussed, the value of ϵ determines the number of support vectors and the sparsity of the solution. Another drawback of SVR is that, as in the case of SVMs or KRR, it may be computationally expensive when dealing with large training sets. One effective solution in such cases, as for KRR, consists of approximating the kernel matrix using low-rank approximations via the Nyström method or the partial Cholesky decomposition. In the next section, we discuss an alternative sparse algorithm for regression.

11.3.4 Lasso

Unlike the KRR and SVR algorithms, the Lasso (least absolute shrinkage and selection operator) algorithm does not admit a natural use of PDS kernels. Thus, here, we assume that the input space $\mathcal{X}$ is a subset of $\mathbb{R}^N$ and consider a family of linear hypotheses $\mathcal{H} = \{x \mapsto \mathbf{w} \cdot \mathbf{x} + b \colon \mathbf{w} \in \mathbb{R}^N, b \in \mathbb{R}\}$.

Let $S = \big((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\big) \in (\mathcal{X} \times \mathcal{Y})^m$ be a labeled training sample. Lasso is based on the minimization of the empirical squared error on S with a regularization term depending on the norm of the weight vector, as in the case of the ridge regression, but using the L_1 norm instead of the L_2 norm and without squaring the

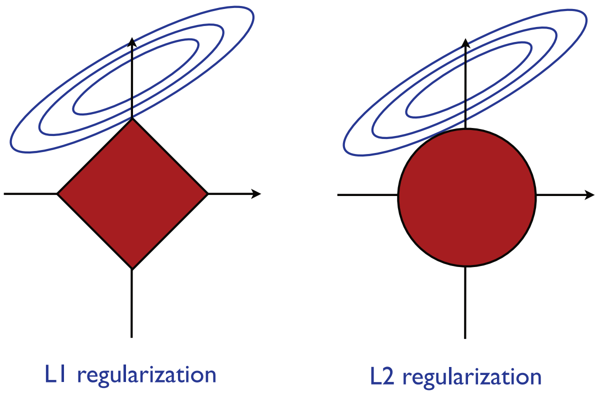


Figure 11.6
Comparison of the Lasso and ridge regression solutions.

norm:

$$\min_{\mathbf{w},b} F(\mathbf{w}, b) = \lambda\|\mathbf{w}\|_1 + \sum_{i=1}^{m} (\mathbf{w}\cdot\mathbf{x}_i + b - y_i)^2 \ . \tag{11.29}$$

Here λ denotes a positive parameter as for ridge regression. This is a convex optimization problem, since $\|\cdot\|_1$ is convex as with all norms and since the empirical error term is convex, as already discussed for linear regression. The optimization for Lasso can be written equivalently as

$$\min_{\mathbf{w},b} \sum_{i=1}^{m} (\mathbf{w}\cdot\mathbf{x}_i + b - y_i)^2 \quad \text{subject to: } \|\mathbf{w}\|_1 \leq \Lambda_1, \tag{11.30}$$

where Λ_1 is a positive parameter.

The key property of Lasso as in the case of other algorithms using the L_1 norm constraint is that it leads to a sparse solution $\mathbf{w}$, that is one with few non-zero components. Figure 11.6 illustrates the difference between the L_1 and L_2 regularizations in dimension two. The objective function of (11.30) is a quadratic function, thus its contours are ellipsoids, as illustrated by the figure (in blue). The areas corresponding to L_1 and L_2 balls of a fixed radius Λ_1 are also shown in the left and right panel (in red). The Lasso solution is the point of intersection of the contours with the L_1 ball. As can be seen form the figure, this can typically occur at a corner of the L_1 ball where some coordinates are zero. In contrast, the ridge regression solution is at the point of intersection of the contours and the L_2 ball, where none of the coordinates is typically zero.

The following results show that Lasso also benefits from strong theoretical guarantees. We first give a general upper bound on the empirical Rademacher complexity of L_1 norm-constrained linear hypotheses .

Theorem 11.15 (Rademacher complexity of linear hypotheses with bounded L_1 norm) *Let $\mathcal{X} \subseteq \mathbb{R}^N$ and let $S = \big((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\big) \in (\mathcal{X} \times \mathcal{Y})^m$ be a sample of size m. Assume that for all $i \in [m]$, $\|\mathbf{x}_i\|_\infty \leq r_\infty$ for some $r_\infty > 0$, and let $\mathcal{H} = \{\mathbf{x} \in \mathcal{X} \mapsto \mathbf{w} \cdot \mathbf{x} \colon \|\mathbf{w}\|_1 \leq \Lambda_1\}$. Then, the empirical Rademacher complexity of $\mathcal{H}$ can be bounded as follows:*

$$\widehat{\mathfrak{R}}_S(\mathcal{H}) \leq \sqrt{\frac{2r_\infty^2 \Lambda_1^2 \log(2N)}{m}}. \tag{11.31}$$

Proof: For any $i \in [m]$ we denote by x_{ij} the jth component of $\mathbf{x}_i$.

$$\begin{aligned}
\widehat{\mathfrak{R}}_S(\mathcal{H}) &= \frac{1}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\|\mathbf{w}\|_1 \leq \Lambda_1} \sum_{i=1}^m \sigma_i \mathbf{w} \cdot \mathbf{x}_i \right] \\
&= \frac{\Lambda_1}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\Big\| \sum_{i=1}^m \sigma_i \mathbf{x}_i \Big\|_\infty \right] && \text{(by definition of the dual norm)} \\
&= \frac{\Lambda_1}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\max_{j \in [N]} \left| \sum_{i=1}^m \sigma_i x_{ij} \right| \right] && \text{(by definition of } \| \cdot \|_\infty) \\
&= \frac{\Lambda_1}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\max_{j \in [N]} \max_{s \in \{-1,+1\}} s \sum_{i=1}^m \sigma_i x_{ij} \right] && \text{(by definition of } \| \cdot \|_\infty) \\
&= \frac{\Lambda_1}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\mathbf{z} \in A} \sum_{i=1}^m \sigma_i z_i \right],
\end{aligned}$$

where A denotes the set of N vectors $\{s(x_{1j}, \ldots, x_{mj})^\top : j \in [N], s \in \{-1,+1\}\}$. For any $\mathbf{z} \in A$, we have $\|\mathbf{z}\|_2 \leq \sqrt{mr_\infty^2} = r_\infty\sqrt{m}$. Thus, by Massart's lemma (theorem 3.7), since A contains at most $2N$ elements, the following inequality holds:

$$\widehat{\mathfrak{R}}_S(\mathcal{H}) \leq \Lambda_1 r_\infty \sqrt{m} \frac{\sqrt{2\log(2N)}}{m} = r_\infty \Lambda_1 \sqrt{\frac{2\log(2N)}{m}},$$

which concludes the proof. □

Note that dependence of the bound on the dimension N is only logarithmic, which suggests that using very high-dimensional feature spaces does not significantly affect generalization.

Combining the Rademacher complexity bound just proven and the general result of Theorem 11.3 yields the following generalization bound for the hypothesis set used by Lasso, using the squared loss.

Theorem 11.16 *Let $\mathcal{X} \subseteq \mathbb{R}^N$ and $\mathcal{H} = \{\mathbf{x} \in \mathcal{X} \mapsto \mathbf{w} \cdot \mathbf{x} \colon \|\mathbf{w}\|_1 \leq \Lambda_1\}$. Assume that there exists $r_\infty > 0$ such for all $\mathbf{x} \in \mathcal{X}$, $\|\mathbf{x}\|_\infty \leq r_\infty$ and $M > 0$ such that*

$|h(x) - y| \leq M$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, each of the following inequalities holds for all $h \in \mathcal{H}$:

$$R(h) \leq \widehat{R}_S(h) + 2r_\infty \Lambda_1 M \sqrt{\frac{2\log(2N)}{m}} + M^2 \sqrt{\frac{\log \frac{1}{\delta}}{2m}}. \tag{11.32}$$

As in the case of ridge regression, we observe that the objective function minimized by Lasso has the same form as the right-hand side of this generalization bound.

There exist a variety of different methods for solving the optimization problem of Lasso, including an efficient algorithm (LARS) for computing the entire *regularization path* of solutions, that is, the Lasso solutions for all values of the regularization parameter λ, and other on-line solutions that apply more generally to optimization problems with an L_1 norm constraint.

Here, we show that the Lasso problems (11.29) or (11.30) are equivalent to a quadratic program (QP), and therefore that any QP solver can be used to compute the solution. Observe that any weight vector $\mathbf{w}$ can be written as $\mathbf{w} = \mathbf{w}^+ - \mathbf{w}^-$, with $\mathbf{w}^+ \geq 0$, $\mathbf{w}^- \geq 0$, and $w_j^+ = 0$ or $w_j^- = 0$ for any $j \in [N]$, which implies $\|\mathbf{w}\|_1 = \sum_{j=1}^N w_j^+ + w_j^-$. This can be done by defining the jth component of $\mathbf{w}^+$ as w_j if $w_j \geq 0$, 0 otherwise, and similarly the jth component of $\mathbf{w}^-$ as $-w_j$ if $w_j \leq 0$, 0 otherwise, for any $j \in [N]$. With the replacement $\mathbf{w} = \mathbf{w}^+ - \mathbf{w}^-$, with $\mathbf{w}^+ \geq 0$, $\mathbf{w}^- \geq 0$, and $\|\mathbf{w}\|_1 = \sum_{j=1}^N w_j^+ + w_j^-$, the Lasso problem (11.29) becomes

$$\min_{\mathbf{w}^+ \geq 0, \mathbf{w}^- \geq 0, b} \lambda \sum_{j=1}^N (w_j^+ + w_j^-) + \sum_{i=1}^m \big((\mathbf{w}^+ - \mathbf{w}^-) \cdot \mathbf{x}_i + b - y_i\big)^2 . \tag{11.33}$$

Conversely, a solution $\mathbf{w} = \mathbf{w}^+ - \mathbf{w}^-$ of (11.33) verifies the condition $w_j^+ = 0$ or $w_j^- = 0$ for any $j \in [N]$, thus $w_j = w_j^+$ when $w_j \geq 0$ and $w_j = -w_j^-$ when $w_j \leq 0$. This is because if $\delta_j = \min(w_j^+, w_j^-) > 0$ for some $j \in [N]$, replacing w_j^+ with $(w_j^+ - \delta_j)$ and w_j^- with $(w_j^- - \delta_j)$ would not affect $w_j^+ - w_j^- = (w_j^+ - \delta) - (w_j^- - \delta)$, but would reduce the term $(w_j^+ + w_j^-)$ in the objective function by $2\delta_j > 0$ and provide a better solution. In view of this analysis, problems (11.29) and (11.33) admit the same optimal solution and are equivalent. Problem (11.33) is a QP since the objective function is quadratic in $\mathbf{w}^+$, $\mathbf{w}^-$, and b, and since the constraints are affine. With this formulation, the problem can be straightforwardly shown to admit a natural online algorithmic solution (exercise 11.10).[20]

Thus, Lasso has several advantages: it benefits from strong theoretical guarantees and returns a sparse solution, which is advantageous when there are accurate solutions based on few features. The sparsity of the solution is also computationally

[20] The technique we described to avoid absolute values in the objective function can be used similarly in other optimization problems.

attractive; sparse feature representations of the weight vector can be used to make the inner product with a new vector more efficient. The algorithm's sparsity can also be used for feature selection. The main drawback of the algorithm is that it does not admit a natural use of PDS kernels and thus an extension to non-linear regression, unlike KRR and SVR. One solution is then to use empirical kernel maps, as discussed in chapter 6. Also, Lasso's solution does not admit a closed-form solution. This is not a critical property from the optimization point of view but one that can make some mathematical analyses very convenient.

11.3.5 Group norm regression algorithms

Other types of regularization aside from the L_1 or L_2 norm can be used to define regression algorithms. For instance, in some situations, the feature space may be naturally partitioned into subsets, and it may be desirable to find a sparse solution that selects or omits entire subsets of features. A natural norm in this setting is the group or mixed norm $L_{2,1}$, which is a combination of the L_1 and L_2 norms. Imagine that we partition $\mathbf{w} \in \mathbb{R}^N$ as $\mathbf{w}_1, \ldots, \mathbf{w}_k$, where $\mathbf{w}_j \in \mathbb{R}^{N_j}$ for $1 \leq j \leq k$ and $\sum_j N_j = N$, and define $\mathbf{W} = (\mathbf{w}_1^\top, \ldots, \mathbf{w}_k^\top)^\top$. Then the $L_{2,1}$ norm of $\mathbf{W}$ is defined as

$$\|\mathbf{W}\|_{2,1} = \sum_{j=1}^{k} \|\mathbf{w}_j\| .$$

Combining the $L_{2,1}$ norm with the empirical mean squared error leads to the *Group Lasso* formulation. More generally, an $L_{q,p}$ group norm regularization can be used for $q, p \geq 1$ (see appendix A for the definition of group norms).

11.3.6 On-line regression algorithms

The regression algorithms presented in the previous sections admit natural on-line versions. Here, we briefly present two examples of these algorithms. These algorithms are particularly useful for applications to very large data sets for which a batch solution can be computationally too costly to derive and more generally in all of the on-line learning settings discussed in chapter 8.

Our first example is known as the *Widrow-Hoff algorithm* and coincides with the application of stochastic gradient descent techniques to the linear regression objective function. Figure 11.7 gives the pseudocode of the algorithm. A similar algorithm can be derived by applying the stochastic gradient technique to ridge regression. At each round, the weight vector is augmented with a quantity that depends on the prediction error $(\mathbf{w}_t \cdot \mathbf{x}_t - y_t)$.

Our second example is an online version of the SVR algorithm, which is obtained by application of stochastic gradient descent to the dual objective function of SVR. Figure 11.8 gives the pseudocode of the algorithm for an arbitrary PDS kernel K

```
WidrowHoff($\mathbf{w}_0$)
1  $\mathbf{w}_1 \leftarrow \mathbf{w}_0$      $\triangleright$ typically $\mathbf{w}_0 = \mathbf{0}$
2  for $t \leftarrow 1$ to $T$ do
3      Receive($\mathbf{x}_t$)
4      $\widehat{y}_t \leftarrow \mathbf{w}_t \cdot \mathbf{x}_t$
5      Receive($y_t$)
6      $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + 2\eta(\mathbf{w}_t \cdot \mathbf{x}_t - y_t)\mathbf{x}_t$    $\triangleright$ learning rate $\eta > 0$.
7  return $\mathbf{w}_{T+1}$
```

Figure 11.7
The Widrow-Hoff algorithm.

in the absence of any offset ($b = 0$). Another on-line regression algorithm is given by exercise 11.10 for Lasso.

11.4 Chapter notes

The generalization bounds presented in this chapter are for bounded regression problems. When $\{x \mapsto L(h(x), y)\colon h \in \mathcal{H}\}$, the family of losses of the hypotheses, is not bounded, a single function can take arbitrarily large values with arbitrarily small probabilities. This is the main issue for deriving uniform convergence bounds for unbounded losses. This problem can be avoided either by assuming the existence of an *envelope*, that is a single non-negative function with a finite expectation lying above the absolute value of the loss of every function in the hypothesis set [Dudley, 1984, Pollard, 1984, Dudley, 1987, Pollard, 1989, Haussler, 1992], or by assuming that some moment of the loss functions is bounded [Vapnik, 1998, 2006]. Cortes, Greenberg, and Mohri [2013] (see also [Cortes et al., 2010a]) give two-sided generalization bounds for unbounded losses with finite second moments. The one-sided version of their bounds coincides with that of Vapnik [1998, 2006] modulo a constant factor, but the proofs given by Vapnik in both books seem to be incomplete and incorrect.

The notion of pseudo-dimension is due to Pollard [1984]. Its equivalent definition in terms of VC-dimension is discussed by Vapnik [2000]. The notion of fat-shattering was introduced by Kearns and Schapire [1990]. The linear regression algorithm is a classical algorithm in statistics that dates back at least to the nine-

```
OnLineDualSVR()
1  $\boldsymbol{\alpha} \leftarrow \mathbf{0}$
2  $\boldsymbol{\alpha}' \leftarrow \mathbf{0}$
3  for $t \leftarrow 1$ to $T$ do
4      Receive($x_t$)
5      $\widehat{y}_t \leftarrow \sum_{s=1}^{t} (\alpha'_s - \alpha_s) K(x_s, x_t)$
6      Receive($y_t$)
7      $\alpha'_{t+1} \leftarrow \alpha'_t + \min(\max(\eta(y_t - \widehat{y}_t - \epsilon), -\alpha'_t), C - \alpha'_t)$
8      $\alpha_{t+1} \leftarrow \alpha_t + \min(\max(\eta(\widehat{y}_t - y_t - \epsilon), -\alpha_t), C - \alpha_t)$
9  return $\sum_{t=1}^{T} (\alpha'_t - \alpha_t) K(x_t, \cdot)$
```

Figure 11.8
An on-line version of dual SVR.

teenth century. The ridge regression algorithm is due to Hoerl and Kennard [1970]. Its kernelized version (KRR) was introduced and discussed by Saunders, Gammerman, and Vovk [1998]. An extension of KRR to outputs in $\mathbb{R}^p$ with $p > 1$ with possible constraints on the regression is presented and analyzed by Cortes, Mohri, and Weston [2007c]. The support vector regression (SVR) algorithm is discussed in Vapnik [2000]. Lasso was introduced by Tibshirani [1996]. The LARS algorithm for solving its optimization problem was later presented by Efron et al. [2004]. The Widrow-Hoff on-line algorithm is due to Widrow and Hoff [1988]. The dual on-line SVR algorithm was first introduced and analyzed by Vijayakumar and Wu [1999]. The kernel stability analysis of exercise 10.3 is from Cortes et al. [2010b].

For large-scale problems where a straightforward batch optimization of a primal or dual objective function is intractable, general iterative stochastic gradient descent methods similar to those presented in section 11.3.6, or quasi-Newton methods such as the limited-memory BFGS (Broyden-Fletcher-Goldfard-Shanno) algorithm [Nocedal, 1980] can be practical alternatives in practice.

In addition to the linear regression algorithms presented in this chapter and their kernel-based non-linear extensions, there exist many other algorithms for regression, including decision trees for regression (see chapter 9), boosting trees for regression, and artificial neural networks.

11.5 Exercises

11.1 Pseudo-dimension and monotonic functions.

Assume that ϕ is a strictly monotonic function and let $\phi \circ \mathcal{H}$ be the family of functions defined by $\phi \circ \mathcal{H} = \{\phi(h(\cdot)) : h \in \mathcal{H}\}$, where $\mathcal{H}$ is some set of real-valued functions. Show that $\mathrm{Pdim}(\phi \circ \mathcal{H}) = \mathrm{Pdim}(\mathcal{H})$.

11.2 Pseudo-dimension of linear functions. Let $\mathcal{H}$ be the set of all linear functions in dimension d, i.e. $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ for some $\mathbf{w} \in \mathbb{R}^d$. Show that $\mathrm{Pdim}(\mathcal{H}) = d$.

11.3 Linear regression.

(a) What condition is required on the data $\mathbf{X}$ in order to guarantee that $\mathbf{X}\mathbf{X}^\top$ is invertible?

(b) Assume the problem is under-determined. Then, we can choose a solution $\mathbf{w}$ such that the equality $\mathbf{X}^\top \mathbf{w} = \mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top)^\dagger \mathbf{X}\mathbf{y}$ (which can be shown to equal $\mathbf{X}^\dagger \mathbf{X}\mathbf{y}$) holds. One particular choice that satisfies this equality is $\mathbf{w}^* = (\mathbf{X}\mathbf{X}^\top)^\dagger \mathbf{X}\mathbf{y}$. However, this is not the unique solution. As a function of $\mathbf{w}^*$, characterize all choices of $\mathbf{w}$ that satisfy $\mathbf{X}^\top \mathbf{w} = \mathbf{X}^\dagger \mathbf{X}\mathbf{y}$ (*Hint*: use the fact that $\mathbf{X}\mathbf{X}^\dagger \mathbf{X} = \mathbf{X}$).

11.4 Perturbed kernels. Suppose two different kernel matrices, $\mathbf{K}$ and $\mathbf{K}'$, are used to train two kernel ridge regression hypothesis with the same regularization parameter λ. In this problem, we will show that the difference in the optimal dual variables, $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ respectively, is bounded by a quantity that depends on $\|\mathbf{K}' - \mathbf{K}\|_2$.

(a) Show $\boldsymbol{\alpha}' - \boldsymbol{\alpha} = \big((\mathbf{K}' + \lambda \mathbf{I})^{-1}(\mathbf{K}' - \mathbf{K})(\mathbf{K} + \lambda \mathbf{I})^{-1}\big)\mathbf{y}$. (*Hint*: Show that for any invertible matrix $\mathbf{M}$, $\mathbf{M}'^1 - \mathbf{M}^1 = -\mathbf{M}'^{-1}(\mathbf{M}' - \mathbf{M})\mathbf{M}^{-1}$.)

(b) Assuming $\forall y \in \mathcal{Y}, |y| \leq M$, show that

$$\|\boldsymbol{\alpha}' - \boldsymbol{\alpha}\| \leq \frac{\sqrt{m} M \|\mathbf{K}' - \mathbf{K}\|_2}{\lambda^2}.$$

11.5 Huber loss. Derive the primal and dual optimization problem used to solve the SVR problem with the Huber loss:

$$L_c(\xi_i) = \begin{cases} \frac{1}{2}\xi_i^2, & \text{if } |\xi_i| \leq c \\ c\xi_i - \frac{1}{2}c^2, & \text{otherwise} \end{cases},$$

where $\xi_i = \mathbf{w} \cdot \Phi(\mathbf{x}_i) + b - y_i$.

```
OnLineLasso($\mathbf{w}_0^+, \mathbf{w}_0^-$)
1  $\mathbf{w}_1^+ \leftarrow \mathbf{w}_0^+$        $\triangleright\, \mathbf{w}_0^+ \geq 0$
2  $\mathbf{w}_1^- \leftarrow \mathbf{w}_0^-$        $\triangleright\, \mathbf{w}_0^- \geq 0$
3  for $t \leftarrow 1$ to $T$ do
4      Receive($\mathbf{x}_t, y_t$)
5      for $j \leftarrow 1$ to $N$ do
6          $w_{t+1j}^+ \leftarrow \max\Big(0, w_{tj}^+ - \eta\Big[\lambda - \big[y_t - (\mathbf{w}_t^+ - \mathbf{w}_t^-) \cdot \mathbf{x}_t\big]\mathbf{x}_{tj}\Big]\Big)$
7          $w_{t+1j}^- \leftarrow \max\Big(0, w_{tj}^- - \eta\Big[\lambda + \big[y_t - (\mathbf{w}_t^+ - \mathbf{w}_t^-) \cdot \mathbf{x}_t\big]\mathbf{x}_{tj}\Big]\Big)$
8  return $\mathbf{w}_{T+1}^+ - \mathbf{w}_{T+1}^-$
```

Figure 11.9
On-line algorithm for Lasso.

11.6 SVR and squared loss. Assuming that $2r\Lambda \leq 1$, use theorem 11.13 to derive a generalization bound for the squared loss.

11.7 SVR dual formulations. Give a detailed and carefully justified derivation of the dual formulations of the SVR algorithm both for the ϵ-insensitive loss and the quadratic ϵ-insensitive loss.

11.8 Optimal kernel matrix. Suppose in addition to optimizing the dual variables $\alpha \in \mathbb{R}^m$, as in (11.16), we also wish to optimize over the entries of the PDS kernel matrix $\mathbf{K} \in \mathbb{R}^{m \times m}$.

$$\min_{\mathbf{K} \succeq 0} \max_{\boldsymbol{\alpha}} -\lambda\boldsymbol{\alpha}^\top\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha} + 2\boldsymbol{\alpha}^\top\mathbf{y}\,, \quad \text{s.t.} \quad \|\mathbf{K}\|_2 \leq 1$$

(a) What is the closed-form solution for the optimal $\mathbf{K}$ for the joint optimization?

(b) Optimizing over the choice of kernel matrix will provide a better value of the objective function. Explain, however, why the resulting kernel matrix is not useful in practice.

11.9 Leave-one-out error. In general, the computation of the leave-one-out error can be very costly since, for a sample of size m, it requires training the algorithm m times. The objective of this problem is to show that, remarkably, in the case of kernel ridge regression, the leave-one-out error can be computed efficiently by training the algorithm only once.

Let $S = ((x_1, y_1), \ldots, (x_m, y_m))$ denote a training sample of size m and for any $i \in [m]$, let S_i denote the sample of size $m-1$ obtained from S by removing (x_i, y_i): $S_i = S - \{(x_i, y_i)\}$. For any sample T, let h_T denote a hypothesis obtained by training T. By definition (see definition 5.2), for the squared loss, the leave-one-out error with respect to S is defined by

$$\widehat{R}_{\mathrm{LOO}}(\mathrm{KRR}) = \frac{1}{m}\sum_{i=1}^{m}(h_{S_i}(x_i) - y_i)^2 .$$

(a) Let $S'_i = ((x_1, y_1), \ldots, (x_i, h_{S_i}(y_i)), \ldots, (x_m, y_m))$. Show that $h_{S_i} = h_{S'_i}$.

(b) Define $\mathbf{y}_i = \mathbf{y} - y_i \boldsymbol{e}_i + h_{S_i}(x_i)\boldsymbol{e}_i$, that is the vector of labels with the ith component replaced with $h_{S_i}(x_i)$. Prove that for KRR $h_{S_i}(x_i) = \mathbf{y}_i^\top(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{K}\boldsymbol{e}_i$.

(c) Prove that the leave-one-out error admits the following simple expression in terms of h_S:

$$\widehat{R}_{\mathrm{LOO}}(\mathrm{KRR}) = \frac{1}{m}\sum_{i=1}^{m}\left[\frac{h_S(x_i) - y_i}{\boldsymbol{e}_i^\top(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{K}\boldsymbol{e}_i}\right]^2 . \tag{11.34}$$

(d) Suppose that the diagonal entries of matrix $\mathbf{M} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{K}$ are all equal to γ. How do the empirical error $\widehat{R}_S$ of the algorithm and the leave-one-out error $\widehat{R}_{\mathrm{LOO}}$ relate? Is there any value of γ for which the two errors coincide?

11.10 On-line Lasso. Use the formulation (11.33) of the optimization problem of Lasso and stochastic gradient descent (see section 8.3.1) to show that the problem can be solved using the on-line algorithm of figure 11.9.

11.11 On-line quadratic SVR. Derive an on-line algorithm for the quadratic SVR algorithm (provide the full pseudocode).

12 Maximum Entropy Models

In this chapter, we introduce and discuss maximum entropy models, also known as *Maxent models*, a widely used family of algorithms for density estimation that can exploit rich feature sets. We first introduce the standard density estimation problem and briefly describe the Maximum Likelihood and Maximum a Posteriori solutions. Next, we describe a richer density estimation problem where the learner additionally has access to features. This is the problem addressed by Maxent models.

We introduce the key principle behind Maxent models and formulate their primal optimization problem. Next, we prove a duality theorem showing that Maxent models coincide with Gibbs distribution solutions of a regularized Maximum Likelihood problem. We present generalization guarantees for these models and also give an algorithm for solving their dual optimization problem using a coordinate descent technique. We further extend these models to the case where an arbitrary Bregman divergence is used with other norms, and prove a general duality theorem leading to an equivalent optimization problem with alternative regularizations. We also give a specific theoretical analysis of Maxent models with L_2-regularization, which are commonly used in applications.

12.1 Density estimation problem

Let $S = (x_1, \ldots, x_m)$ be a sample of size m drawn i.i.d. from an unknown distribution $\mathcal{D}$. Then, the density estimation problem consists of using that sample to select out of a family of possible distributions $\mathcal{P}$ a distribution p that is close to $\mathcal{D}$.

The choice of the family $\mathcal{P}$ is critical. A relatively small family may not contain $\mathcal{D}$ or even any distribution close to $\mathcal{D}$. On the other hand, a very rich family defined by a large set of parameters may make the task of selecting p very difficult if only a sample of a relatively modest size m is available.

12.1.1 Maximum Likelihood (ML) solution

One common solution adopted for selecting a distribution p is based on the *maximum likelihood principle*. This consists of choosing a distribution out of the family $\mathcal{P}$ that assigns the largest probability to the sample S observed. Thus, using the fact that the sample is drawn i.i.d., the solution p_{ML} selected by maximum likelihood is defined by

$$\mathsf{p}_{\mathrm{ML}} = \operatorname*{argmax}_{\mathsf{p}\in\mathcal{P}} \prod_{i=1}^{m} \mathsf{p}(x_i) = \operatorname*{argmax}_{\mathsf{p}\in\mathcal{P}} \sum_{i=1}^{m} \log \mathsf{p}(x_i). \tag{12.1}$$

The maximum likelihood principle can be equivalently formulated in terms of the relative entropy. Let $\widehat{\mathcal{D}}$ denote the empirical distribution corresponding to the sample S. Then, p_{ML} coincides with the distribution p with respect to which the empirical distribution $\widehat{\mathcal{D}}$ admits the smallest relative entropy:

$$\mathsf{p}_{\mathrm{ML}} = \operatorname*{argmin}_{\mathsf{p}\in\mathcal{P}} \mathsf{D}(\widehat{\mathcal{D}} \,\|\, \mathsf{p}). \tag{12.2}$$

This can be seen straightforwardly from the following:

$$\begin{aligned}
\mathsf{D}(\widehat{\mathcal{D}} \,\|\, \mathsf{p}) &= \sum_x \widehat{\mathcal{D}}(x) \log \widehat{\mathcal{D}}(x) - \sum_x \widehat{\mathcal{D}}(x) \log \mathsf{p}(x)\\
&= -\mathsf{H}(\widehat{\mathcal{D}}) - \sum_x \frac{\sum_{i=1}^m 1_{x=x_i}}{m} \log \mathsf{p}(x)\\
&= -\mathsf{H}(\widehat{\mathcal{D}}) - \sum_{i=1}^m \sum_x \frac{1_{x=x_i}}{m} \log \mathsf{p}(x)\\
&= -\mathsf{H}(\widehat{\mathcal{D}}) - \sum_{i=1}^m \frac{\log \mathsf{p}(x_i)}{m},
\end{aligned}$$

since the first term of the last expression, the negative entropy of the empirical distribution, does not vary with p.

As an example of application of the maximum likelihood principle, suppose we wish to estimate the bias p_0 of a coin from an i.i.d. sample $S = (x_1, \ldots, x_m)$ where $x_i \in \{\mathsf{h}, \mathsf{t}\}$ with h denoting heads and t tails. $p_0 \in [0, 1]$ is the probability of h according to the unknown distribution $\mathcal{D}$. Let $\mathcal{P}$ be the family of all distributions $\mathsf{p} = (p, 1-p)$ where $p \in [0, 1]$ is an arbitrary possible bias value. Let n_{h} denote the number of occurrences of h in S. Then, choosing $\mathsf{p} = (\widehat{p}_S, 1 - \widehat{p}_S) = \widehat{\mathcal{D}}$ where $\widehat{p}_S = \frac{n_{\mathsf{h}}}{m}$ leads to $\mathsf{D}(\widehat{\mathcal{D}} \,\|\, \mathsf{p}) = 0$, which, by (12.2), shows that $\mathsf{p}_{\mathrm{ML}} = \widehat{\mathcal{D}}$. Thus, the maximum likelihood estimate p_{ML} of the bias is the empirical value

$$p_{\mathrm{ML}} = \frac{n_{\mathsf{h}}}{m}. \tag{12.3}$$

12.1.2 Maximum a Posteriori (MAP) solution

An alternative solution based on the so-called *Maximum a Posteriori* solution consists of selecting a distribution $\mathsf{p} \in \mathcal{P}$ that is the most likely, given the observed sample S and a prior $\mathbb{P}[\mathsf{p}]$ over the distributions $\mathsf{p} \in \mathcal{P}$. By the Bayes rule, the problem can be formulated as follows:

$$\mathsf{p}_{\text{MAP}} = \operatorname*{argmax}_{\mathsf{p} \in \mathcal{P}} \mathbb{P}[\mathsf{p}|S] = \operatorname*{argmax}_{\mathsf{p} \in \mathcal{P}} \frac{\mathbb{P}[S|\mathsf{p}]\,\mathbb{P}[\mathsf{p}]}{\mathbb{P}[S]} = \operatorname*{argmax}_{\mathsf{p} \in \mathcal{P}} \mathbb{P}[S|\mathsf{p}]\,\mathbb{P}[\mathsf{p}]. \tag{12.4}$$

Notice that, for a uniform prior, $\mathbb{P}[\mathsf{p}]$ is a constant and the Maximum a Posteriori solution then coincides with the Maximum Likelihood solution. The following is a standard example illustrating the MAP solution and its difference with the ML solution.

Example 12.1 (Application of the MAP solution) Suppose we need to determine if a patient has a rare disease, given a laboratory test of that patient. We consider a set of two simple distributions: d (disease with probability one) and $\bar{\mathsf{d}}$ (no disease with probability one), thus $\mathcal{P} = \{\mathsf{d}, \bar{\mathsf{d}}\}$. The laboratory test is either pos (positive) or neg (negative), thus $S \in \{\text{pos}, \text{neg}\}$.

Suppose that the disease is rare, say $\mathbb{P}[\mathsf{d}] = .005$ and that the laboratory is relatively accurate: $\mathbb{P}[\text{pos}|\mathsf{d}] = .98$, and $\mathbb{P}[\text{neg}|\bar{\mathsf{d}}] = .95$. Then, if the test is positive, what should be the diagnosis? We can compute the right-hand side of (12.4) for both outcomes, given the positive test result, to determine the MAP estimate:

$$\begin{aligned}\mathbb{P}[\text{pos}|\mathsf{d}]\,\mathbb{P}[\mathsf{d}] &= .98 \times .005 = .0049\\ \mathbb{P}[\text{pos}|\bar{\mathsf{d}}]\,\mathbb{P}[\bar{\mathsf{d}}] &= (1 - .95) \times (1 - .005) = .04975 > .0049.\end{aligned}$$

Thus, in this case, the MAP prediction is no disease: according to the MAP solution, with the values indicated, a patient with a positive test result is nonetheless more likely not to have the disease!

We will not analyze the properties of the Maximum Likelihood and Maximum a Posteriori solutions here, which depend on the size of the sample and the choice of the family $\mathcal{P}$. Instead, we will consider a richer density estimation problem where the learner has access to features, which is the learning problem addressed by Maximum Entropy (Maxent) models.

12.2 Density estimation problem augmented with features

As with the standard density estimation problem, we consider a scenario where the learner receives a sample $S = (x_1, \ldots, x_m) \subseteq \mathcal{X}$ of size m drawn i.i.d. according to some distribution $\mathcal{D}$. But, here, additionally, we assume that the learner has access to a feature mapping $\mathbf{\Phi}$ from $\mathcal{X}$ to $\mathbb{R}^N$ with $\|\mathbf{\Phi}\|_\infty \leq r$. In the most general case, we may have $N = +\infty$. We will denote by $\mathcal{H}$ a family of real-valued functions containing the component feature functions Φ_j with $j \in [N]$. Different feature functions can be considered in practice. $\mathcal{H}$ may be the family of threshold functions $\mathbf{x} \mapsto 1_{x_i \leq \theta}$, $\mathbf{x} \in \mathbb{R}^n$, $\theta \in \mathbb{R}$, defined over n variables as for boosting stumps, or it may be a family of functions defined by more complex decision trees or regression trees. Other features often used in practice are monomials of degree k based on the input variables. To simplify the presentation, in what follows, we will assume that the input set $\mathcal{X}$ is finite.

12.3 Maxent principle

Maxent models are derived from a principle based on the key property that, with high probability, the empirical average of any feature is close to its true average. By the Rademacher complexity bound, for any $\delta > 0$, the following inequality holds with probability at least $1 - \delta$ over the choice of a sample S of size m:

$$\left\| \mathop{\mathbb{E}}_{x \sim \mathcal{D}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x \sim \widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] \right\|_\infty \leq 2\mathfrak{R}_m(\mathcal{H}) + r\sqrt{\frac{\log \frac{2}{\delta}}{2m}}, \tag{12.5}$$

where we denote by $\widehat{\mathcal{D}}$ the empirical distribution defined by the sample S. This is the theoretical guarantee that guides the definition of the Maxent principle.

Let p_0 be a distribution over $\mathcal{X}$ with $\mathsf{p}_0(x) > 0$ for all $x \in \mathcal{X}$, which is often chosen to be the uniform distribution. Then, the *Maxent principle* consists of seeking a distribution p that is as agnostic as possible, that is as close as possible to the uniform distribution or, more generally, to a prior p_0, while verifying an inequality similar to (12.5):

$$\left\| \mathop{\mathbb{E}}_{x \sim \mathsf{p}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x \sim \widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] \right\|_\infty \leq \lambda, \tag{12.6}$$

where $\lambda \geq 0$ is a parameter. Here, closeness is measured using the relative entropy. Choosing $\lambda = 0$ corresponds to standard *Maxent* or *unregularized Maxent* and to requiring the expectation of the features with respect to p to precisely match the empirical averages. As we will see later, its relaxation, that is the inequality case ($\lambda \neq 0$), translates into a regularization. Notice that, unlike Maximum likelihood, the Maxent principle does not require specifying a family of probability distributions $\mathcal{P}$ to choose from.

12.4 Maxent models

Let Δ denote the simplex of all distributions over $\mathcal{X}$, then, the Maxent principle can be formulated as the following optimization problem:

$$\min_{\mathsf{p}\in\Delta} \mathsf{D}(\mathsf{p}\,\|\,\mathsf{p}_0) \tag{12.7}$$
$$\text{subject to: } \left\| \mathop{\mathbb{E}}_{x\sim\mathsf{p}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] \right\|_\infty \leq \lambda.$$

This defines a convex optimization problem since the relative entropy D is convex with respect to its arguments (appendix E), since the constraints are affine, and since Δ is a convex set. The solution is in fact unique since the relative entropy is strictly convex. The empirical distribution is clearly a feasible point, thus problem (12.7) is feasible.

For a uniform prior p_0, problem (12.7) can be equivalently formulated as an entropy maximization, which explains the name given to these models. Let $\mathsf{H}(\mathsf{p}) = -\sum_{x\in\mathcal{X}} \mathsf{p}(x)\log\mathsf{p}(x)$ denote the entropy of p. Then, the objective function of (12.7) can be rewritten as follows:

$$\begin{aligned}\mathsf{D}(\mathsf{p}\,\|\,\mathsf{p}_0) &= \sum_{x\in\mathcal{X}} \mathsf{p}(x)\log\frac{\mathsf{p}(x)}{\mathsf{p}_0(x)}\\ &= -\sum_{x\in\mathcal{X}} \mathsf{p}(x)\log\mathsf{p}_0(x) + \sum_{x\in\mathcal{X}} \mathsf{p}(x)\log\mathsf{p}(x)\\ &= \log|\mathcal{X}| - \mathsf{H}(\mathsf{p}).\end{aligned}$$

Thus, since $\log|\mathcal{X}|$ is a constant, minimizing the relative entropy $\mathsf{D}(\mathsf{p}\,\|\,\mathsf{p}_0)$ is then equivalent to maximizing $\mathsf{H}(\mathsf{p})$.

Maxent models are the solutions of the optimization problem just described. As already discussed, they admit two important benefits: they are based on a fundamental theoretical guarantee of closeness of empirical and true feature averages, and they do not require specifying a particular family of distributions $\mathcal{P}$. In the next sections, we will further analyze the properties of Maxent models.

12.5 Dual problem

Here, we derive an equivalent dual problem for (12.7) which, as we will show, can be formulated as a regularized maximum likelihood problem over the family of *Gibbs distributions*.

For any convex set K, let I_K denote the function defined by $I_K(x) = 0$ if $x \in K$, $I_K(x) = +\infty$ otherwise. Then, the Maxent optimization problem (12.7) can be equivalently expressed as the unconstrained optimization problem $\min_{\mathsf{p}} F(\mathsf{p})$ with, for all $\mathsf{p} \in \mathbb{R}^{\mathcal{X}}$,

$$F(\mathsf{p}) = \widetilde{\mathsf{D}}(\mathsf{p} \,\|\, \mathsf{p}_0) + I_{\mathcal{C}}(\mathop{\mathbb{E}}_{\mathsf{p}}[\mathbf{\Phi}]), \tag{12.8}$$

with $\widetilde{\mathsf{D}}(\mathsf{p} \,\|\, \mathsf{p}_0) = \mathsf{D}(\mathsf{p} \,\|\, \mathsf{p}_0)$ if p is in the simplex Δ, $\widetilde{\mathsf{D}}(\mathsf{p} \,\|\, \mathsf{p}_0) = +\infty$ otherwise, and with $\mathcal{C} \subseteq \mathbb{R}^N$ the convex set defined by $\mathcal{C} = \{\mathbf{u} \colon \|\mathbf{u} - \mathbb{E}_{(x,y)\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x,y)]\,\|_\infty \leq \lambda\}$.

The general form of a Gibbs distribution $\mathsf{p}_{\mathbf{w}}$ with prior p_0, parameter $\mathbf{w}$, and feature vector $\mathbf{\Phi}$ is

$$\mathsf{p}_{\mathbf{w}}[x] = \frac{\mathsf{p}_0[x] e^{\mathbf{w}\cdot\mathbf{\Phi}(x)}}{Z(\mathbf{w})} \tag{12.9}$$

where $Z(\mathbf{w}) = \sum_{x\in\mathcal{X}} \mathsf{p}_0[x] e^{\mathbf{w}\cdot\mathbf{\Phi}(x)}$ is a normalization factor also known as the *partition function*. Let G be the function defined for all $\mathbf{w} \in \mathbb{R}^N$ by

$$G(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \log\left[\frac{\mathsf{p}_{\mathbf{w}}[x_i]}{\mathsf{p}_0[x_i]}\right] - \lambda \|\mathbf{w}\|_1. \tag{12.10}$$

Then, the following theorem shows the equivalence of the primal problem (12.7) or (12.8) and a dual problem based on G.

Theorem 12.2 (Maxent duality) *Problems* (12.7) *or* (12.8) *are equivalent to the optimization problem* $\sup_{\mathbf{w}\in\mathbb{R}^N} G(\mathbf{w})$*:*

$$\sup_{\mathbf{w}\in\mathbb{R}^N} G(\mathbf{w}) = \min_{\mathsf{p}} F(\mathsf{p}). \tag{12.11}$$

Furthermore, let $\mathsf{p}^* = \operatorname{argmin}_{\mathsf{p}} F(\mathsf{p})$ *and* $d^* = \sup_{\mathbf{w}\in\mathbb{R}^N} G(\mathbf{w})$*, then, for any* $\epsilon > 0$ *and any* $\mathbf{w}$ *such that* $|G(\mathbf{w}) - d^*| < \epsilon$*, the following inequality holds:* $\mathsf{D}(\mathsf{p}^* \,\|\, \mathsf{p}_{\mathbf{w}}) \leq \epsilon$.

Proof: The first part of the proof follows by application of the Fenchel duality theorem (theorem B.39) to the optimization problem (12.8) with the functions f, g, and A defined for all $\mathsf{p} \in \mathbb{R}^{\mathcal{X}}$ and $\mathbf{u} \in \mathbb{R}^N$ by $f(\mathsf{p}) = \widetilde{\mathsf{D}}(\mathsf{p} \,\|\, \mathsf{p}_0)$, $g(\mathbf{u}) = I_{\mathcal{C}}(\mathbf{u})$ and $A\mathsf{p} = \sum_{x\in\mathcal{X}} \mathsf{p}(x)\mathbf{\Phi}(x)$. A is a bounded linear map since for any p, we have $\|A\mathsf{p}\| \leq \|\mathsf{p}\|_1 \sup_x \|\mathbf{\Phi}(x)\|_\infty \leq r\|\mathsf{p}\|_1$. Also, notice that for all $\mathbf{w} \in \mathbb{R}^N$, $A^*\mathbf{w} = \mathbf{w} \cdot \mathbf{\Phi}$.

Consider $\mathbf{u}_0 \in \mathbb{R}^N$ defined by $\mathbf{u}_0 = \mathbb{E}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] = A\widehat{\mathcal{D}}$. Since $\widehat{\mathcal{D}}$ is in $\Delta = \operatorname{dom}(f)$, $\mathbf{u}_0$ is in $A(\operatorname{dom}(f))$. Furthermore, since $\lambda > 0$, $\mathbf{u}_0$ is in $\operatorname{int}(\mathcal{C})$. $g = I_{\mathcal{C}}$ equals zero over $\operatorname{int}(\mathcal{C})$ and is therefore continuous over $\operatorname{int}(\mathcal{C})$, thus g is continuous at $\mathbf{u}_0$ and we have $\mathbf{u}_0 \in A(\operatorname{dom}(f)) \cap \operatorname{cont}(g)$. Thus, the assumptions of Theorem B.39 hold.

By Lemma B.37, the conjugate of f is the function $f^* \colon \mathbb{R}^{\mathcal{X}} \to \mathbb{R}$ defined by $f^*(\mathsf{q}) = \log\big(\sum_{x\in\mathcal{X}} \mathsf{p}_0[x] e^{\mathsf{q}[x]}\big)$ for all $\mathsf{q} \in \mathbb{R}^{\mathcal{X}}$. The conjugate function of $g = I_{\mathcal{C}}$ is the function g^* defined for all $\mathbf{w} \in \mathbb{R}^N$ by

$$\begin{aligned} g^*(\mathbf{w}) &= \sup_{\mathbf{u}} \big(\mathbf{w} \cdot \mathbf{u} - I_{\mathcal{C}}(\mathbf{u})\big) = \sup_{\mathbf{u} \in \mathcal{C}} (\mathbf{w} \cdot \mathbf{u}) \\ &= \sup_{\|\mathbf{u} - \mathbb{E}_{\widehat{\mathcal{D}}}[\mathbf{\Phi}]\|_\infty \leq \lambda} (\mathbf{w} \cdot \mathbf{u}) \\ &= \mathbf{w} \cdot \mathop{\mathbb{E}}_{\widehat{\mathcal{D}}}[\mathbf{\Phi}] + \sup_{\|\mathbf{u}\|_\infty \leq \lambda} (\mathbf{w} \cdot \mathbf{u}) \\ &= \mathop{\mathbb{E}}_{\widehat{\mathcal{D}}}[\mathbf{w} \cdot \mathbf{\Phi}] + \lambda \|\mathbf{w}\|_1, \end{aligned}$$

where the last equality holds by definition of the dual norm. In view of these identities, we can write

$$\begin{aligned} -f^*(A^*\mathbf{w}) - g^*(-\mathbf{w}) &= -\log\Big(\sum_{x \in \mathcal{X}} \mathsf{p}_0[x] e^{\mathbf{w} \cdot \mathbf{\Phi}(x)}\Big) + \mathop{\mathbb{E}}_{\widehat{\mathcal{D}}}[\mathbf{w} \cdot \mathbf{\Phi}] - \lambda \|\mathbf{w}\|_1 \\ &= -\log Z(\mathbf{w}) + \frac{1}{m} \sum_{i=1}^m \mathbf{w} \cdot \mathbf{\Phi}(x_i) - \lambda \|\mathbf{w}\|_1 \\ &= \frac{1}{m} \sum_{i=1}^m \log \frac{e^{\mathbf{w} \cdot \mathbf{\Phi}(x_i)}}{Z(\mathbf{w})} - \lambda \|\mathbf{w}\|_1 \\ &= \frac{1}{m} \sum_{i=1}^m \log \left[\frac{\mathsf{p}_{\mathbf{w}}[x_i]}{\mathsf{p}_0[x_i]}\right] - \lambda \|\mathbf{w}\|_1 = G(\mathbf{w}), \end{aligned}$$

which proves that $\sup_{\mathbf{w} \in \mathbb{R}^N} G(\mathbf{w}) = \min_{\mathsf{p}} F(\mathsf{p})$.

Now, for any $\mathbf{w} \in \mathbb{R}^N$, we can write

$$\begin{aligned} & G(\mathbf{w}) - \mathsf{D}(\mathsf{p}^* \,\|\, \mathsf{p}_0) + \mathsf{D}(\mathsf{p}^* \,\|\, \mathsf{p}_{\mathbf{w}}) \\ &= \mathop{\mathbb{E}}_{x \sim \widehat{\mathcal{D}}} \left[\log \frac{\mathsf{p}_{\mathbf{w}}[x]}{\mathsf{p}_0[x]}\right] - \lambda \|\mathbf{w}\|_1 - \mathop{\mathbb{E}}_{x \sim \mathsf{p}^*} \left[\log \frac{\mathsf{p}^*[x]}{\mathsf{p}_0[x]}\right] + \mathop{\mathbb{E}}_{x \sim \mathsf{p}^*} \left[\log \frac{\mathsf{p}^*[x]}{\mathsf{p}_{\mathbf{w}}[x]}\right] \\ &= -\lambda \|\mathbf{w}\|_1 + \mathop{\mathbb{E}}_{x \sim \widehat{\mathcal{D}}} \left[\log \frac{\mathsf{p}_{\mathbf{w}}[x]}{\mathsf{p}_0[x]}\right] - \mathop{\mathbb{E}}_{x \sim \mathsf{p}^*} \left[\log \frac{\mathsf{p}_{\mathbf{w}}(x)}{\mathsf{p}_0(x)}\right] \\ &= -\lambda \|\mathbf{w}\|_1 + \mathop{\mathbb{E}}_{x \sim \widehat{\mathcal{D}}} [\mathbf{w} \cdot \mathbf{\Phi}(x) - \log Z(\mathbf{w})] - \mathop{\mathbb{E}}_{x \sim \mathsf{p}^*} [\mathbf{w} \cdot \mathbf{\Phi}(x) - \log Z(\mathbf{w})] \\ &= -\lambda \|\mathbf{w}\|_1 + \mathbf{w} \cdot \Big[\mathop{\mathbb{E}}_{x \sim \widehat{\mathcal{D}}} [\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x \sim \mathsf{p}^*} [\mathbf{\Phi}(x)]\Big]. \end{aligned}$$

The solution of the primal optimization, p^*, verifies the constraint $I_{\mathcal{C}}(\mathbb{E}_{\mathsf{p}^*}[\mathbf{\Phi}]) = 0$, that is $\| \mathbb{E}_{x \sim \widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathbb{E}_{x \sim \mathsf{p}^*}[\mathbf{\Phi}(x)]\|_\infty \leq \lambda$. By Hölder's inequality, this implies the following inequality:

$$-\lambda \|\mathbf{w}\|_1 + \mathbf{w} \cdot \Big[\mathop{\mathbb{E}}_{x \sim \widehat{\mathcal{D}}} [\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x \sim \mathsf{p}^*} [\mathbf{\Phi}(x)]\Big] \leq -\lambda \|\mathbf{w}\|_1 + \lambda \|\mathbf{w}\|_1 = 0.$$

Thus, we can write, for any $\mathbf{w} \in \mathbb{R}^N$,

$$\mathsf{D}(\mathsf{p}^* \,\|\, \mathsf{p}_\mathbf{w}) \leq \mathsf{D}(\mathsf{p}^* \,\|\, \mathsf{p}_0) - G(\mathbf{w}).$$

Now, assume that $\mathbf{w}$ verifies $|G(\mathbf{w}) - \sup_{\mathbf{w} \in \mathbb{R}^N} G(\mathbf{w})| \leq \epsilon$ for some $\epsilon > 0$. Then, $\mathsf{D}(\mathsf{p}^* \,\|\, \mathsf{p}_0) - G(\mathbf{w}) = (\sup_\mathbf{w} G(\mathbf{w})) - G(\mathbf{w}) \leq \epsilon$ implies $\mathsf{D}(\mathsf{p}^* \,\|\, \mathsf{p}_\mathbf{w}) \leq \epsilon$. This concludes the proof of the theorem. □

In view of the theorem, if $\mathbf{w}$ is an ϵ-solution of the dual optimization problem, then $\mathsf{D}(\mathsf{p}^* \,\|\, \mathsf{p}_\mathbf{w}) \leq \epsilon$, which, by Pinsker's inequality (Proposition E.7) implies that $\mathsf{p}_\mathbf{w}$ is $\sqrt{2\epsilon}$-close in L_1-norm to the optimal solution of the primal: $\|\mathsf{p}^* - \mathsf{p}_\mathbf{w}\|_1 \leq \sqrt{2\epsilon}$. Thus, the solution of our Maxent problem can be determined by solving the dual problem, which can be written equivalently as follows:

$$\inf_{\mathbf{w} \in \mathbb{R}^N} \lambda \|\mathbf{w}\|_1 - \frac{1}{m} \sum_{i=1}^{m} \log \mathsf{p}_\mathbf{w}[x_i]. \tag{12.12}$$

Notice that the solution may not be achieved for any finite $\mathbf{w}$ for $\lambda = 0$, which is why the infimum is needed. This result may seem surprising since it shows that Maxent coincides with Maximum Likelihood ($\lambda = 0$) or regularized Maximum Likelihood ($\lambda > 0$) over a specific family $\mathcal{P}$ of distributions, that of Gibbs distributions, while, as pointed out earlier, the Maxent principle does not explicitly specify any family $\mathcal{P}$. What can then explain that the solution of Maxent belongs to the specific family of Gibbs distributions? The reason is the specific choice of the relative entropy as the measure of closeness of p to the prior distribution p_0. Other measures of closeness between distributions lead to different forms for the solution. Thus, in some sense, the choice of the measure of closeness is the (dual) counterpart of that of the family of distributions $\mathcal{P}$ in maximum likelihood.

Gibbs distributions form a very rich family. In particular, when $\mathcal{X}$ is a subset of a vector space and the features $\Phi_j(\mathbf{x})$ associated to $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X}$ are monomials of degree at most 2 based on the input variables x_j, that is $x_j x_k$, x_j, or the constant $a \in \mathbb{R}$, then $\mathbf{w} \cdot \mathbf{\Phi}(x)$ is a quadratic form as a function of the x_js. Thus, Gibbs distributions include the family of distributions defined by the normalized exponential of a quadratic form, which includes as a special case Gaussian distributions but also bi-modal distributions and normalized exponentials of non-positive definite quadratic forms. More complex multi-modal distributions can be further defined using higher-order monomials or more complex functions of the input variables. Figure 12.1 shows two examples of Gibbs distributions illustrating the richness of this family.

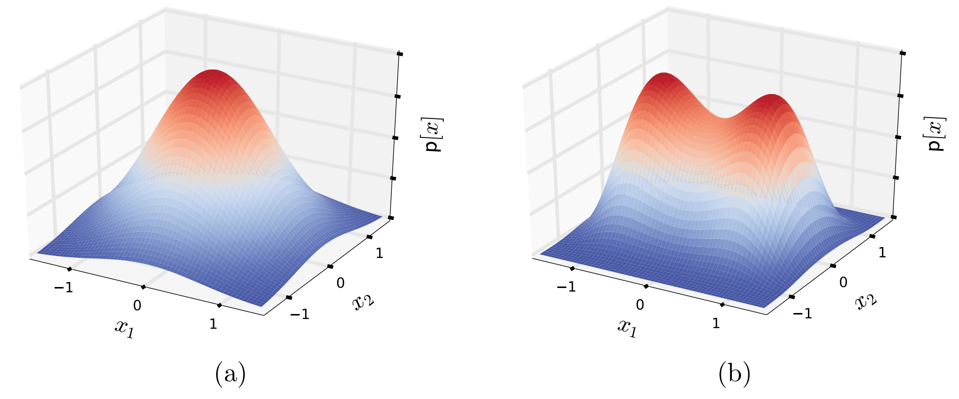


Figure 12.1
Examples of Gibbs distributions in $\mathbb{R}^2$. (a) Unimodal Gaussian distribution $\mathsf{p}[(x_1, x_2)] = \frac{e^{-(x_1^2+x_2^2)}}{Z}$; (b) Bimodal distribution $\mathsf{p}[(x_1, x_2)] = \frac{e^{-(x_1^4+x_2^4)+x_1^2-x_2^2}}{Z}$. In each case, Z is a normalization factor.

12.6 Generalization bound

Let $\mathcal{L}_{\mathcal{D}}(\mathbf{w})$ denote the log-loss of the distribution $\mathsf{p}_{\mathbf{w}}$ with respect to a distribution $\mathcal{D}$, $\mathcal{L}_{\mathcal{D}}(\mathbf{w}) = \mathbb{E}_{x\sim\mathcal{D}}[-\log \mathsf{p}_{\mathbf{w}}[x]]$, and similarly $\mathcal{L}_S(\mathbf{w})$ its log-loss with respect to the empirical distribution defined by a sample S.

Theorem 12.3 *Fix $\delta > 0$. Let $\widehat{\mathbf{w}}$ be a solution of the optimization* (12.12) *for $\lambda = 2\mathfrak{R}_m(\mathcal{H}) + r\sqrt{\frac{\log\frac{2}{\delta}}{2m}}$. Then, with probability at least $1-\delta$ over the draw of an i.i.d. sample S of size m from $\mathcal{D}$, the following inequality holds:*

$$\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}}) \leq \inf_{\mathbf{w}} \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + 2\|\mathbf{w}\|_1 \left[2\mathfrak{R}_m(\mathcal{H}) + r\sqrt{\frac{\log\frac{2}{\delta}}{2m}}\right].$$

Proof: Using the definition of $\mathcal{L}_{\mathcal{D}}(\mathbf{w})$ and $\mathcal{L}_S(\mathbf{w})$, Hölder's inequality, and inequality (12.5), with probability at least $1-\delta$, the following holds:

$$\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}}) - \mathcal{L}_S(\widehat{\mathbf{w}}) = \widehat{\mathbf{w}} \cdot [\mathop{\mathbb{E}}_{\widehat{\mathcal{D}}}[\mathbf{\Phi}] - \mathop{\mathbb{E}}_{\mathcal{D}}[\mathbf{\Phi}]] \leq \|\widehat{\mathbf{w}}\|_1 \, \| \mathop{\mathbb{E}}_{\widehat{\mathcal{D}}}[\mathbf{\Phi}] - \mathop{\mathbb{E}}_{\mathcal{D}}[\mathbf{\Phi}]\|_\infty \leq \lambda\|\widehat{\mathbf{w}}\|_1.$$

Thus, since $\widehat{\mathbf{w}}$ is a minimizer, we can write, for any $\mathbf{w}$,

$$\begin{aligned}
\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}}) - \mathcal{L}_{\mathcal{D}}(\mathbf{w}) &= \mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}}) - \mathcal{L}_S(\widehat{\mathbf{w}}) + \mathcal{L}_S(\widehat{\mathbf{w}}) - \mathcal{L}_{\mathcal{D}}(\mathbf{w}) \\
&\leq \lambda\|\widehat{\mathbf{w}}\|_1 + \mathcal{L}_S(\widehat{\mathbf{w}}) - \mathcal{L}_{\mathcal{D}}(\mathbf{w}) \\
&\leq \lambda\|\mathbf{w}\|_1 + \mathcal{L}_S(\mathbf{w}) - \mathcal{L}_{\mathcal{D}}(\mathbf{w}) \leq 2\lambda\|\mathbf{w}\|_1,
\end{aligned}$$

where we used for the last inequality the left inequality counterpart of inequality (12.5). This concludes the proof. □

Assume that $\mathbf{w}^*$ achieves the infimum of the loss, that is $\mathcal{L}_{\mathcal{D}}(\mathbf{w}^*) = \inf_{\mathbf{w}} \mathcal{L}_{\mathcal{D}}(\mathbf{w})$ and that $\mathfrak{R}_m(\mathcal{H}) = (1/\sqrt{m})$. Then, the theorem shows that, with high probability, the following inequality holds:

$$\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}}) \leq \inf_{\mathbf{w}} \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + O\bigg(\frac{\|\mathbf{w}^*\|_1}{\sqrt{m}}\bigg).$$

12.7 Coordinate descent algorithm

The dual objective function in the optimization (12.12) is convex since the Lagrange dual is always concave (appendix B). Ignoring the constant term $-\frac{1}{m}\sum_{i=1}^m \log \mathsf{p}_0[x_i]$, the optimization problem (12.12) can be rewritten as $\inf_{\mathbf{w}} J(\mathbf{w})$ with

$$J(\mathbf{w}) = \lambda\|\mathbf{w}\|_1 - \mathbf{w} \cdot \mathop{\mathbb{E}}_{\widehat{\mathcal{D}}}[\mathbf{\Phi}] + \log\left[\sum_{x\in\mathcal{X}} \mathsf{p}_0[x]e^{\mathbf{w}\cdot\mathbf{\Phi}(x)}\right].$$

Note in particular that the function $\mathbf{w} \mapsto \log\big[\sum_{x\in\mathcal{X}} \mathsf{p}_0[x]e^{\mathbf{w}\cdot\mathbf{\Phi}(x)}\big]$ is convex as the conjugate function f^* of the function f defined in the proof of Theorem 12.2.

Different optimization techniques can be used to solve this convex optimization problem, including standard stochastic gradient descent and several special-purpose techniques. In this section, we will describe a solution based on coordinate descent which is particularly advantageous in presence of a very large number of features.

Function J is not differentiable but since it is convex, it admits a subdifferential at any point. The Maxent algorithm we describe consists of applying coordinate descent to the objective function (12.12).

Direction Let $\mathbf{w}_{t-1}$ denote the weight vector defined after $(t-1)$ iterations. At each iteration $t \in [T]$, the direction $\mathbf{e}_j$, $j \in [N]$ considered by coordinate descent is $\delta J(\mathbf{w}_{t-1}, \mathbf{e}_j)$. If $w_{t-1,j} \neq 0$, then J admits a directional derivative along $\mathbf{e}_j$ given by

$$J'(\mathbf{w}_{t-1}, \mathbf{e}_j) = \lambda \operatorname{sgn}(w_{t-1,j}) + \epsilon_{t-1,j}.$$

where $\epsilon_{t-1,j} = \mathbb{E}_{\mathsf{p}_{\mathbf{w}_{t-1}}}[\Phi_j] - \mathbb{E}_{\widehat{\mathcal{D}}}[\Phi_j]$. If $w_{t-1,j} = 0$, J admits right and left directional derivatives along $\mathbf{e}_j$:

$$J'_+(\mathbf{w}_{t-1}, \mathbf{e}_j) = \lambda + \epsilon_{t-1,j} \qquad J'_-(\mathbf{w}_{t-1}, \mathbf{e}_j) = -\lambda + \epsilon_{t-1,j}.$$

```
CDMaxent($S = (x_1, \ldots, x_m)$)
 1  for $t \leftarrow 1$ to $T$ do
 2      for $j \leftarrow 1$ to $N$ do
 3          if $(w_{t-1,j} \neq 0)$ then
 4              $d_j \leftarrow \lambda \operatorname{sgn}(w_{t-1,j}) + \epsilon_{t-1,j}$
 5          elseif $|\epsilon_{t-1,j}| \leq \lambda$ then
 6              $d_j \leftarrow 0$
 7          else $d_j \leftarrow -\lambda \operatorname{sgn}(\epsilon_{t-1,j}) + \epsilon_{t-1,j}$
 8      $j \leftarrow \underset{j \in [N]}{\operatorname{argmax}} |d_j|$
 9      if $(|w_{t-1,j} r^2 - \epsilon_{t-1,j}| \leq \lambda)$ then
10          $\eta \leftarrow -w_{t-1,j}$
11      elseif $(w_{t-1,j} r^2 - \epsilon_{t-1,j} > \lambda)$ then
12          $\eta \leftarrow \frac{1}{r^2}[-\lambda - \epsilon_{t-1,j}]$
13      else $\eta \leftarrow \frac{1}{r^2}[\lambda - \epsilon_{t-1,j}]$
14      $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} + \eta \mathbf{e}_j$
15      $\mathsf{p}_{\mathbf{w}_t} \leftarrow \frac{\mathsf{p}_0[x] e^{\mathbf{w}_t \cdot \mathbf{\Phi}(x)}}{\sum_{x \in \mathcal{X}} \mathsf{p}_0[x] e^{\mathbf{w}_t \cdot \mathbf{\Phi}(x)}}$
16  return $\mathsf{p}_{\mathbf{w}_t}$
```

Figure 12.2
Pseudocode of the Coordinate Descent Maxent algorithm. For all $j \in [N]$, $\epsilon_{t-1,j} = \mathbb{E}_{\mathsf{p}_{\mathbf{w}_{t-1}}}[\Phi_j] - \mathbb{E}_{\widehat{\mathcal{D}}}[\Phi_j]$.

Thus, in summary, we can define, for all $j \in [N]$,

$$\delta J(\mathbf{w}_{t-1}, \mathbf{e}_j) = \begin{cases} \lambda \operatorname{sgn}(w_{t-1,j}) + \epsilon_{t-1,j} & \text{if } (w_{t-1,j} \neq 0) \\ 0 & \text{else if } \left|\epsilon_{t-1,j}\right| \leq \lambda \\ -\lambda \operatorname{sgn}(\epsilon_{t-1,j}) + \epsilon_{t-1,j} & \text{otherwise.} \end{cases}$$

The coordinate descent algorithm selects the direction $\mathbf{e}_j$ with the largest absolute value of $\delta J(\mathbf{w}_{t-1}, \mathbf{e}_j)$.

Step size Given the direction $\mathbf{e}_j$, the optimal step value η is given by $\operatorname{argmin}_\eta J(\mathbf{w}_{t-1} + \eta\, \mathbf{e}_j)$. η can be found via a line search or other numerical methods. A closed-form expression for the step can also be derived by minimizing an upper bound on $J(\mathbf{w}_{t-1} + \eta\, \mathbf{e}_j)$. Notice that we can write

$$J(\mathbf{w}_{t-1}+\eta\, \mathbf{e}_j) - J(\mathbf{w}_{t-1}) = \lambda(|w_j+\eta| - |w_j|) - \eta \mathop{\mathbb{E}}_{S}[\Phi_j] + \log\left[\mathop{\mathbb{E}}_{\mathsf{p}_{\mathbf{w}_{t-1}}}[e^{\eta\Phi_j}]\right]. \quad (12.13)$$

In view of $\Phi_j \in [-r, +r]$, by Hoeffding's lemma, the following inequality holds:

$$\log \mathop{\mathbb{E}}_{\mathsf{p}_{\mathbf{w}_{t-1}}}[e^{\eta\Phi_j}] \leq \eta \mathop{\mathbb{E}}_{\mathsf{p}_{\mathbf{w}_{t-1}}}[\Phi_j] + \frac{\eta^2 r^2}{2}.$$

Combining this inequality with (12.13) and disregarding constant terms, minimizing the resulting upper bound on $J(\mathbf{w}_{t-1} + \eta\, \mathbf{e}_j) - J(\mathbf{w}_{t-1})$ becomes equivalent to minimizing $\varphi(\eta)$ defined for all $\eta \in \mathbb{R}$ by

$$\varphi(\eta) = \lambda|w_j + \eta| + \eta\epsilon_{t-1,j} + \frac{\eta^2 r^2}{2}.$$

Let η^* denote the minimizer of $\varphi(\eta)$. If $w_{t-1,j} + \eta^* = 0$, then the subdifferential of $|w_{t-1,j} + \eta|$ at η^* is the set $\{\nu\colon \nu \in [-1, +1]\}$. Thus, in that case, the subdifferential $\partial\varphi(\eta^*)$ contains 0 iff there exists $\nu \in [-1, +1]$ such that

$$\lambda\nu + \epsilon_{t-1,j} + \eta^* r^2 = 0 \Leftrightarrow w_{t-1,j} r^2 - \epsilon_{t-1,j} = \lambda\nu.$$

The condition is therefore equivalent to $|w_{t-1,j} r^2 - \epsilon_{t-1,j}| \leq \lambda$. If $w_{t-1,j} + \eta^* > 0$, then φ is differentiable at η^* and $\varphi'(\eta^*) = 0$, that is

$$\lambda + \epsilon_{t-1,j} + \eta^* r^2 = 0 \Leftrightarrow \eta^* = \frac{1}{r^2}[-\lambda - \epsilon_{t-1,j}].$$

In view of that expression, the condition $w_{t-1,j} + \eta^* > 0$ is equivalent to $w_{t-1,j} r^2 - \epsilon_{t-1,j} > \lambda$. Similarly, if $w_{t-1,j} + \eta^* < 0$, φ is differentiable at η^* and $\varphi'(\eta^*) = 0$, which gives

$$\eta^* = \frac{1}{r^2}[\lambda - \epsilon_{t-1,j}].$$

Figure 12.2 shows the pseudocode of the Coordinate Descent Maxent algorithm using the closed-form solution for the step size just presented. Note that we do not need to update distribution $\mathsf{p}_{\mathbf{w}_t}$ at every iteration of the algorithm (line 15) and we only need to be able to compute $\mathbb{E}_{\mathsf{p}_{\mathbf{w}_t}}[\Phi_j]$ which defines $\epsilon_{t,j}$. Various approximation strategies can be used to do this efficiently, including for instance rejection sampling techniques.

12.8 Extensions

As already pointed out, the Gibbs distribution form of the Maxent models is tightly related to the choice of the divergence (relative entropy) used to measure closeness in the Maxent principle. For distributions, the relative entropy coincides with the unnormalized relative entropy, which is a Bregman divergence. Maxent models can be generalized by using an arbitrary Bregman divergence B_Ψ instead (appendix E), where Ψ is a convex function. Moreover, other norms $\|\cdot\|$ can be used to bound the difference of the empirical and true average feature vectors. This leads to the following general primal optimization problem for Maxent models:

$$\min_{\mathsf{p}\in\Delta} \mathsf{B}_\Psi(\mathsf{p}\,\|\,\mathsf{p}_0) \qquad (12.14)$$
$$\text{subject to: } \Big\| \mathop{\mathbb{E}}_{x\sim\mathsf{p}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)]\Big\| \leq \lambda,$$

which, as with (12.7), is a convex optimization problem since B_Ψ is convex with respect to its first argument and in fact strictly convex if Ψ is strictly convex. The following general duality theorem gives the form of the dual problem equivalent to (12.14) in terms of the conjugate function Ψ^* of Ψ. Here, $\|\cdot\|$ is an arbitrary norm over $\mathbb{R}^N$ and $\|\cdot\|_*$ its conjugate. We will assume here that $\sup_x \|\mathbf{\Phi}(x)\| \leq r$.

Theorem 12.4 *Let Ψ be a convex function defined over $\mathbb{R}^{\mathcal{X}}$. Then, problem (12.14) admits the following equivalent dual:*

$$\min_{\mathsf{p}\in\Delta} \mathsf{B}_\Psi(\mathsf{p}\,\|\,\mathsf{p}_0)$$
$$\textit{subject to: } \Big\| \mathop{\mathbb{E}}_{x\sim\mathsf{p}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)]\Big\| \leq \lambda$$
$$= \sup_{\mathbf{w}\in\mathbb{R}^N} -\Psi^*\big(\mathbf{w}\cdot\mathbf{\Phi} + \nabla\Psi(\mathsf{p}_0)\big) + \mathbf{w}\cdot \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \lambda\|\mathbf{w}\|_* - C(\mathsf{p}_0),$$

where $C(\mathsf{p}_0) = \Psi(\mathsf{p}_0) - \langle\nabla\Psi(\mathsf{p}_0), \mathsf{p}_0\rangle$.

Proof: The proof is similar to that of Theorem 12.2 and follows by application of the Fenchel duality theorem (Theorem B.39) to the following optimization problem:

$$\min_{\mathsf{p}} f(\mathsf{p}) + g(A\mathsf{p}), \qquad (12.15)$$

with the functions f, g, and A defined for all $\mathsf{p} \in \mathbb{R}^{\mathcal{X}}$ and $\mathbf{u} \in \mathbb{R}^N$ by $f(p) = \mathsf{B}_\Psi(\mathsf{p}\,\|\,\mathsf{p}_0) + I_\Delta(p)$, $g(\mathbf{u}) = I_{\mathcal{C}}(\mathbf{u})$, and $A\mathsf{p} = \sum_{x\in\mathcal{X}} \mathsf{p}(x)\mathbf{\Phi}(x)$. Given these definitions, problem (12.15) is equivalent to (12.14). A is a bounded linear map since for any p, we have $\|A\mathsf{p}\| \leq \|\mathsf{p}\|_1 \sup_x \|\mathbf{\Phi}(x)\| \leq r\|\mathsf{p}\|_1$. Also, notice that for all $\mathbf{w} \in \mathbb{R}^N$, $A^*\mathbf{w} = \mathbf{w}\cdot\mathbf{\Phi}$.

Consider $\mathbf{u}_0 \in \mathbb{R}^N$ defined by $\mathbf{u}_0 = \mathbb{E}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] = A\widehat{\mathcal{D}}$. Since $\widehat{\mathcal{D}}$ is in $\Delta = \mathrm{dom}(f)$, $\mathbf{u}_0$ is in $A(\mathrm{dom}(f))$. Furthermore, since $\lambda > 0$, $\mathbf{u}_0$ is in $\mathrm{int}(\mathcal{C})$. $g = I_{\mathcal{C}}$ equals zero over $\mathrm{int}(\mathcal{C})$ and is therefore continuous over $\mathrm{int}(\mathcal{C})$, thus g is continuous at $\mathbf{u}_0$ and we have $\mathbf{u}_0 \in A(\mathrm{dom}(f)) \cap \mathrm{cont}(g)$. Thus, the assumptions of Theorem B.39 hold.

The conjugate function of f is defined for all $\mathsf{q} \in \mathbb{R}^{\mathcal{X}}$ by

$$\begin{aligned}
f^*(\mathsf{q}) &= \sup_{\mathsf{p}} \langle \mathsf{p}, \mathsf{q}\rangle - \mathsf{B}_\Psi(\mathsf{p} \,\|\, \mathsf{p}_0) - I_\Delta(\mathsf{p}) \\
&= \sup_{\mathsf{p}\in\Delta} \langle \mathsf{p}, \mathsf{q}\rangle - \mathsf{B}_\Psi(\mathsf{p} \,\|\, \mathsf{p}_0) \\
&= \sup_{\mathsf{p}\in\Delta} \langle \mathsf{p}, \mathsf{q}\rangle - \Psi(\mathsf{p}) + \Psi(\mathsf{p}_0) + \langle \nabla\Psi(\mathsf{p}_0), \mathsf{p} - \mathsf{p}_0\rangle \\
&= \sup_{\mathsf{p}\in\Delta} \langle \mathsf{p}, \mathsf{q} + \nabla\Psi(\mathsf{p}_0)\rangle - \Psi(\mathsf{p}) + \Psi(\mathsf{p}_0) - \langle \nabla\Psi(\mathsf{p}_0), \mathsf{p}_0\rangle \\
&= \Psi^*(\mathsf{q} + \nabla\Psi(\mathsf{p}_0)) + \Psi(\mathsf{p}_0) - \langle \nabla\Psi(\mathsf{p}_0), \mathsf{p}_0\rangle.
\end{aligned}$$

The conjugate function of $g = I_{\mathcal{C}}$ is defined for all $\mathbf{w} \in \mathbb{R}^N$ by

$$\begin{aligned}
g^*(\mathbf{w}) &= \sup_{\mathbf{u}} \langle \mathbf{w}, \mathbf{u}\rangle - I_{\mathcal{C}}(\mathbf{u}) \\
&= \sup_{\mathbf{u}\in\mathcal{C}} \langle \mathbf{w}, \mathbf{u}\rangle \\
&= \sup_{\|\mathbf{u} - \mathbb{E}_{\widehat{\mathcal{D}}}[\mathbf{\Phi}]\| \le \lambda} \langle \mathbf{w}, \mathbf{u}\rangle \\
&= \langle \mathbf{w}, \mathop{\mathbb{E}}_{\widehat{\mathcal{D}}}[\mathbf{\Phi}]\rangle + \sup_{\|\mathbf{u}\| \le \lambda} \langle \mathbf{w}, \mathbf{u}\rangle = \langle \mathbf{w}, \mathop{\mathbb{E}}_{\widehat{\mathcal{D}}}[\mathbf{\Phi}]\rangle + \lambda\|\mathbf{w}\|_*,
\end{aligned}$$

where the last equality holds by definition of the dual norm. In view of these identities, by Theorem B.39, we have

$$\begin{aligned}
\min_{\mathsf{p}} f(\mathsf{p}) + g(A\mathsf{p}) &= \sup_{\mathbf{w}\in\mathbb{R}^N} -f^*(A^*\mathbf{w}) - g^*(\mathbf{w}) \\
&= \sup_{\mathbf{w}\in\mathbb{R}^N} -\Psi^*(\mathbf{w}\cdot\mathbf{\Phi} + \nabla\Psi(\mathsf{p}_0)) + \mathbf{w}\cdot \mathop{\mathbb{E}}_{\widehat{\mathcal{D}}}[\mathbf{\Phi}] - \lambda\|\mathbf{w}\|_* \\
&\qquad - \Psi(\mathsf{p}_0) + \langle \nabla\Psi(\mathsf{p}_0), \mathsf{p}_0\rangle,
\end{aligned}$$

which completes the proof. □

Note, the previous proof and its use of Fenchel duality holds even when considering norms that are not inner product norms and, more generally, Banach spaces are considered (as mentioned in section B.4).

Much of the analysis and theoretical guarantees presented in previous sections in the special case of the unnormalized relative entropy straightforwardly extend to a broad family of Bregman divergences.

12.9 L_2-regularization

In this section, we study a common variant of the Maxent algorithm where a regularization based on the norm-2 squared of the weight vector $\mathbf{w}$ is used. Observe that this is not covered by the general framework discussed in the previous section where the regularization was based on some norm of $\mathbf{w}$. The corresponding (dual) optimization problem is the following:

$$\min_{\mathbf{w}\in\mathbb{R}^N} \lambda\|\mathbf{w}\|_2^2 - \frac{1}{m}\sum_{i=1}^{m} \log \mathsf{p}_{\mathbf{w}}[x_i]. \tag{12.16}$$

Let $\mathcal{L}_{\mathcal{D}}(\mathbf{w})$ denote the log-loss of the distribution $\mathsf{p}_{\mathbf{w}}$ with respect to a distribution $\mathcal{D}$, $\mathcal{L}_{\mathcal{D}}(\mathbf{w}) = \mathbb{E}_{x\sim\mathcal{D}}[-\log \mathsf{p}_{\mathbf{w}}[x]]$, and similarly $\mathcal{L}_S(\mathbf{w})$ its log-loss with respect to the empirical distribution defined by a sample S. Then, the algorithm admits the following guarantee.

Theorem 12.5 *Let $\widehat{\mathbf{w}}$ be a solution of the optimization problem* (12.16)*. Then, for any $\delta > 0$, with probability at least $1-\delta$ over the draw of an i.i.d. sample S of size m from $\mathcal{D}$, the following inequality holds:*

$$\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}}) \leq \inf_{\mathbf{w}} \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2 + \frac{r^2}{\lambda m}\left(1+\sqrt{\log\frac{1}{\delta}}\right)^2.$$

Proof: Let $\widehat{D}$ denote the empirical distribution defined by the sample S. Then, the optimization problem (12.16) can be formulated as follows:

$$\min_{\mathbf{w}\in\mathbb{R}^N} \lambda\|\mathbf{w}\|_2^2 - \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}\big[\log \mathsf{p}_{\mathbf{w}}[x]\big] = \lambda\|\mathbf{w}\|_2^2 - \mathbf{w}\cdot \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] + \log Z(\mathbf{w}),$$

where $Z(\mathbf{w}) = \big(\sum_x \exp(\mathbf{w}\cdot\mathbf{\Phi}(x))\big)$. Similarly, let $\mathbf{w}_{\mathcal{D}}$ denote the solution of the minimization problem with the distribution $\mathcal{D}$:

$$\min_{\mathbf{w}\in\mathbb{R}^N} \lambda\|\mathbf{w}\|_2^2 - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\big[\log \mathsf{p}_{\mathbf{w}}[x]\big] = \lambda\|\mathbf{w}\|_2^2 - \mathbf{w}\cdot \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)] + \log Z(\mathbf{w}).$$

We first give an upper-bound on $\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}})$ valid for all $\mathbf{w}\in\mathbb{R}^N$, starting with a decomposition of $\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}})$ as a sum of terms, next using the expression of $\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}}) - \mathcal{L}_S(\widehat{\mathbf{w}})$ in terms of average feature values, then the optimality of $\widehat{\mathbf{w}}$, next the expression of $\mathcal{L}_S(\mathbf{w}_{\mathcal{D}}) - \mathcal{L}_{\mathcal{D}}(\mathbf{w}_{\mathcal{D}})$ in terms of average feature values, and finally the Cauchy-Schwarz inequality and the optimality of $\mathbf{w}_{\mathcal{D}}$:

$$\begin{aligned}
&\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}})\\
&= \mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}}) - \mathcal{L}_S(\widehat{\mathbf{w}}) + \mathcal{L}_S(\widehat{\mathbf{w}}) - \mathcal{L}_{\mathcal{D}}(\mathbf{w}_{\mathcal{D}}) + \mathcal{L}_{\mathcal{D}}(\mathbf{w}_{\mathcal{D}}) + \lambda\|\widehat{\mathbf{w}}\|_2^2 - \lambda\|\widehat{\mathbf{w}}\|_2^2\\
&= \widehat{\mathbf{w}}\cdot\Big[\mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)]\Big] + \mathcal{L}_S(\widehat{\mathbf{w}}) - \mathcal{L}_{\mathcal{D}}(\mathbf{w}_{\mathcal{D}}) + \mathcal{L}_{\mathcal{D}}(\mathbf{w}_{\mathcal{D}}) + \lambda\|\widehat{\mathbf{w}}\|_2^2 - \lambda\|\widehat{\mathbf{w}}\|_2^2\\
&\leq \widehat{\mathbf{w}}\cdot\Big[\mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)]\Big] + \mathcal{L}_S(\mathbf{w}_{\mathcal{D}}) - \mathcal{L}_{\mathcal{D}}(\mathbf{w}_{\mathcal{D}}) + \mathcal{L}_{\mathcal{D}}(\mathbf{w}_{\mathcal{D}}) + \lambda\|\mathbf{w}_{\mathcal{D}}\|_2^2 - \lambda\|\widehat{\mathbf{w}}\|_2^2\\
&\leq [\widehat{\mathbf{w}} - \mathbf{w}_{\mathcal{D}}]\cdot\Big[\mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)]\Big] + \mathcal{L}_{\mathcal{D}}(\mathbf{w}_{\mathcal{D}}) + \lambda\|\mathbf{w}_{\mathcal{D}}\|_2^2 - \lambda\|\widehat{\mathbf{w}}\|_2^2\\
&\leq \|\widehat{\mathbf{w}} - \mathbf{w}_{\mathcal{D}}\|_2\Big\|\mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)]\Big\|_2 + \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2.
\end{aligned}$$

Next, we bound $\|\widehat{\mathbf{w}} - \mathbf{w}_{\mathcal{D}}\|_2$ using the fact that $\widehat{\mathbf{w}}$ and $\mathbf{w}_{\mathcal{D}}$ are solutions of the minimization of convex and differentiable objectives functions, whose gradients must be zero at the minimizing values:

$$2\lambda\widehat{\mathbf{w}} - \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] + \nabla \log Z(\widehat{\mathbf{w}}) = 0$$
$$2\lambda\mathbf{w}_{\mathcal{D}} - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)] + \nabla \log Z(\mathbf{w}_{\mathcal{D}}) = 0,$$

which implies

$$2\lambda(\widehat{\mathbf{w}} - \mathbf{w}_{\mathcal{D}}) = \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)] + \nabla \log Z(\mathbf{w}_{\mathcal{D}}) - \nabla \log Z(\widehat{\mathbf{w}}).$$

Multiplying both sides by $(\widehat{\mathbf{w}} - \mathbf{w}_{\mathcal{D}})$ gives

$$\begin{aligned}
&2\lambda\|\widehat{\mathbf{w}} - \mathbf{w}_{\mathcal{D}}\|_2^2 \\
&= \Big[\mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)]\Big] \cdot [\widehat{\mathbf{w}} - \mathbf{w}_{\mathcal{D}}] - [\nabla \log Z(\widehat{\mathbf{w}}) - \nabla \log Z(\mathbf{w}_{\mathcal{D}})] \cdot [\widehat{\mathbf{w}} - \mathbf{w}_{\mathcal{D}}] \\
&\leq \Big[\mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)]\Big] \cdot [\widehat{\mathbf{w}} - \mathbf{w}_{\mathcal{D}}],
\end{aligned}$$

in view of the convexity of $\mathbf{w} \mapsto \log Z(\mathbf{w})$. Using the Cauchy-Schwarz inequality and simplifying, we obtain

$$\|\widehat{\mathbf{w}} - \mathbf{w}_{\mathcal{D}}\|_2 \leq \frac{\Big\|\mathbb{E}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathbb{E}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)]\Big\|_2}{2\lambda}.$$

Plugging this back in the upper bound previously derived for $\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}})$ yields

$$\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}}) \leq \frac{\Big\|\mathbb{E}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathbb{E}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)]\Big\|_2^2}{2\lambda} + \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2.$$

We now use McDiarmid's inequality to bound $\Big\|\mathbb{E}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathbb{E}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)]\Big\|_2$. Let $\Psi(S)$ denote this quantity for a sample S. Let S' be a sample differing from S by one point, say x_m for S, x'_m for S'. Then, by the triangle inequality,

$$\begin{aligned}
|\Psi(S') - \Psi(S)| &= \bigg|\Big\|\mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}'}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)]\Big\|_2 - \Big\|\mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathbf{\Phi}(x)]\Big\|_2\bigg| \\
&\leq \Big\|\mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}'}[\mathbf{\Phi}(x)] - \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x)]\Big\|_2 \\
&\leq \Big\|\frac{\mathbf{\Phi}(x'_m) - \mathbf{\Phi}(x_m)}{m}\Big\|_2 \leq \frac{2r}{m}.
\end{aligned}$$

Thus, for any $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds

$$\Psi(S) \leq \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[\Psi(S)] + 2r\sqrt{\frac{\log\frac{1}{\delta}}{2m}}.$$

For any $i \in [m]$, let $\mathbf{Z}_i$ denote the random variable $\mathbb{E}_{x \sim \widehat{\mathcal{D}}}[\mathbf{\Phi}(x_i)] - \mathbb{E}_{x \sim \mathcal{D}}[\mathbf{\Phi}(x)]$. Then, by Jensen's inequality, $\mathbb{E}_{S \sim \mathcal{D}^m}[\Psi(S)]$ can be upper-bounded as follows:

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[\Psi(S)] = \mathbb{E}\left[\left\|\frac{1}{m}\sum_{i=1}^{m} \mathbf{Z}_i\right\|_2\right] \leq \sqrt{\mathbb{E}\left[\left\|\frac{1}{m}\sum_{i=1}^{m} \mathbf{Z}_i\right\|_2^2\right]}.$$

Since the random variables $\mathbf{Z}_i$s are i.i.d. and centered ($\mathbb{E}[\mathbf{Z}_i] = 0$), we have

$$\begin{aligned}
\mathbb{E}\left[\left\|\frac{1}{m}\sum_{i=1}^{m} \mathbf{Z}_i\right\|^2\right] &= \frac{1}{m^2}\left[\sum_{i=1}^{m} \mathbb{E}\left[\|\mathbf{Z}_i\|^2\right] + \sum_{i \neq j} \mathbb{E}[\mathbf{Z}_i] \cdot \mathbb{E}[\mathbf{Z}_j]\right] \\
&= \frac{\mathbb{E}\left[\|\mathbf{Z}_1\|^2\right]}{m} \\
&= \frac{\mathbb{E}\left[\|\mathbf{Z}_1\|^2 + \|\mathbf{Z}_2\|^2\right]}{2m} \\
&= \frac{\mathbb{E}\left[\|\mathbf{Z}_1 - \mathbf{Z}_2\|^2\right]}{2m}.
\end{aligned}$$

where, for the last equality, we used the fact that $\mathbb{E}[\mathbf{Z}_1 \cdot \mathbf{Z}_2] = \mathbb{E}[\mathbf{Z}_1] \cdot \mathbb{E}[\mathbf{Z}_2] = 0$. This shows that $\mathbb{E}[\Psi(S)] \leq \frac{2r}{\sqrt{2m}}$ and that, with probability at least $1 - \delta$, the following holds

$$\Psi(S) \leq \frac{2r}{\sqrt{2m}}\left(1 + \sqrt{\log \frac{1}{\delta}}\right),$$

and therefore also

$$\begin{aligned}
\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}}) &\leq \frac{1}{2\lambda} \frac{2r^2}{m}\left(1 + \sqrt{\log \frac{1}{\delta}}\right)^2 + \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2 \\
&\leq \frac{r^2}{\lambda m}\left(1 + \sqrt{\log \frac{1}{\delta}}\right)^2 + \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2,
\end{aligned}$$

which ends the proof. □

Assume that $\mathbf{w}^*$ achieves the infimum of the loss, that is $\mathcal{L}_{\mathcal{D}}(\mathbf{w}^*) = \inf_{\mathbf{w}} \mathcal{L}_{\mathcal{D}}(\mathbf{w})$ and that we are given an upper bound Λ_2 on its norm: $\|\mathbf{w}^*\|_2 \leq \Lambda_2$. Then, we can use that upper bound and choose λ to minimize the two terms containing λ: $\lambda \Lambda_2^2 = \frac{r^2}{\lambda m}$, that is $\lambda = \frac{r}{\Lambda_2 \sqrt{m}}$ and the theorem would then guarantee the following inequality with probability $1 - \delta$ for $\widehat{\mathbf{w}}$:

$$\mathcal{L}_{\mathcal{D}}(\widehat{\mathbf{w}}) \leq \inf_{\mathbf{w}} \mathcal{L}_{\mathcal{D}}(\mathbf{w}) + \frac{r\Lambda_2}{\sqrt{m}}\left[1 + \left(1 + \sqrt{\log \frac{1}{\delta}}\right)^2\right].$$

12.10 Chapter notes

The Maxent principle was first explicitly advocated by Jaynes [1957] (see also Jaynes [1983]) who referred to Shannon's notion of entropy (appendix E) to support this principle. As seen in Section 12.5, standard Maxent models coincide with Gibbs distributions, as in the original Boltzmann models in statistical mechanics. In fact, Jaynes [1957] argued that statistical mechanics could be viewed as a form of statistical inference, as opposed to a physical theory, and that the thermodynamic notion of entropy could be replaced by the information-theoretical notion. The justification of the Maxent principle presented in this chapter is instead based upon learning theory arguments.

Maximum entropy models, commonly referred to as Maxent models, are used in a variety of tasks in natural language processing [Berger et al., 1996, Rosenfeld, 1996, Pietra et al., 1997, Malouf, 2002, Manning and Klein, 2003, Ratnaparkhi, 2010] and in many other applications, including species habitat modeling [Phillips et al., 2004, 2006, Dudík et al., 2007, Elith et al., 2011]. One key benefit of Maxent models is that they allow the use of diverse features that can be selected and augmented by the user. The richness of the features used in many tasks as well as small sample sizes have motivated the use of regularized Maxent models where the L_1-norm [Kazama and Tsujii, 2003] or the L_2-norm [Chen and Rosenfeld, 2000, Lebanon and Lafferty, 2001] of the parameter vector defining the Gibbs distribution is controlled. This can be shown to be equivalent to the introduction of a Laplacian or Gaussian prior over the parameter vectors in a Bayesian interpretation [Williams, 1994, Goodman, 2004], thereby making Maxent models coincide with Maximum a Posteriori solutions with specific choices of the prior.

An extensive theoretical study of these regularizations and the introduction of other more general ones were given by Dudík, Phillips, and Schapire [2007] and by Altun and Smola [2006] who studied the extensions to arbitrary Bregman divergences and norms (Section 12.8) using Fenchel duality (see also [Lafferty, Pietra, and Pietra, 1997]). Cortes, Kuznetsov, Mohri, and Syed [2015] give a more general family of density estimation models, *Structural Maxent models*, with feature functions selected from a union of possibly very complex sub-families for which they also give a duality theorem, strong learning guarantees, and algorithms. These models can also be viewed as Maxent with a more general type of regularization.

The Maxent duality theorem is due to Pietra, Pietra, and Lafferty [1997] (see also [Dudík et al., 2007] and [Altun and Smola, 2006]). Theorem 12.2 is a slight extension giving a guarantee for an ϵ-solution of the dual and is a special instance of a more general theorem given for Structural Maxent models [Cortes et al., 2015]. The generalization bounds of Sections 12.6 and 12.9 and their proofs are variants of results due to Dudík et al. [2007]. The stability analysis used in the proof of Theorem 12.5 is equivalent to the one described in Chapter 14 using Bregman divergences.

A variety of different techniques have been suggested to solve the Maxent optimization problem including standard gradient descent and stochastic gradient descent. Some specific algorithms were introduced for this problem, including *generalized iterative scaling* (GIS) [Darroch and Ratcliff, 1972] and *improved iterative scaling* (IIS) [Pietra et al., 1997]. It was shown by Malouf [2002] that these algorithms perform poorly in several natural language processing tasks in comparison with conjugate gradient techniques and limited-memory BFGS methods (see also [Andrew and Gao, 2007]). The coordinate descent solution presented in this chapter is due to Cortes et al. [2015]. It is a simpler version of an algorithm of Dudík et al. [2007] which uses a tighter upper bound on $J(\mathbf{w}_{t-1} + \eta\, \mathbf{e}_j)$ but which is subject to various technical conditions. Both algorithms benefit from a similar asymptotic convergence rate [Cortes et al., 2015] and are particularly adapted to cases where the number of features is very large and where updating all feature weights is impractical. A sequential greedy approximation due to Zhang [2003b] is also advocated by Altun and Smola [2006] as a general algorithm for general forms of the Maxent problem.

12.11 Exercises

12.1 Convexity. Prove directly that the function $\mathbf{w} \mapsto \log Z(\mathbf{w}) = \log(\sum_{x \in \mathcal{X}} e^{\mathbf{w} \cdot \mathbf{\Phi}(x)})$ is convex (*Hint*: compute its Hessian).

12.2 Lagrange duality. Derive the dual problem of the Maxent problem and justify it carefully in the case of the stricter constraint of positivity for the distribution p: $\mathsf{p}(x) > 0$ for all $x \in \mathcal{X}$.

12.3 Dual of norm-2 squared regularized Maxent. Derive the dual formulation of the norm-2 squared regularized Maxent optimization shown in equation (12.16).

12.4 Extension to Bregman divergences. Derive theoretical guarantees for the extensions discussed in Section 12.8. What additional property is needed for the Bregman divergence so that your learning guarantees hold?

12.5 L_2-regularization. Let $\mathbf{w}$ be the solution of Maxent with a norm-2 squared regularization.

(a) Prove the following inequality: $\|\mathbf{w}\|_2 \leq \frac{2r}{\lambda}$ (*Hint*: you could compare the values of the objective function at $\mathbf{w}$ and 0.). Generalize this result to other $\|\cdot\|_p^p$-regularizations with $p > 1$.

(b) Use the previous question to derive an explicit learning guarantee for Maxent with norm-2 squared regularization (*Hint*: you could use the last inequality given in Section 12.9 and derive an explicit expression for Λ_2).

13 Conditional Maximum Entropy Models

This chapter presents algorithms for estimating the conditional probability of a class given an example, rather than only predicting the class label for that example. This is motivated by several applications where confidence values are sought, in addition to the class prediction. The algorithms discussed, *conditional Maxent models*, also known as *multinomial logistic regression* algorithms, are among the most well-known and most widely used multi-class classification algorithms. In the special case of two classes, the algorithm is known as *logistic regression*.

As suggested by their name, these algorithms can be viewed as Maxent models for conditional probabilities. To introduce them, we will extend the ideas discussed in the previous chapter (Chapter 12), starting from an extension of the Maxent principle to the conditional case. Next, we will prove a duality theorem leading to an equivalent dual optimization problem for conditional Maxent. We will specifically discuss different aspects of multi-class classification using conditional Maxent and reserve a special section to the analysis of logistic regression.

13.1 Learning problem

We consider a multi-class classification problem with c classes, $c \geq 1$. Let $\mathcal{Y} = \{1, \ldots, c\}$ denote the output space and $\mathcal{D}$ a distribution over $\mathcal{X} \times \mathcal{Y}$. The learner receives a labeled training sample $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$ drawn i.i.d. according to $\mathcal{D}$. As in Chapter 12, we assume that, additionally, the learner has access to a feature mapping $\mathbf{\Phi} \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^N$ with $\mathbb{R}^N$ a normed vector space and with $\|\mathbf{\Phi}\|_\infty \leq r$. We will denote by $\mathcal{H}$ a family of real-valued functions containing the component feature functions Φ_j with $j \in [N]$. Note that in the most general case, we may have $N = +\infty$. The problem consists of using the training sample S to learn an accurate conditional probability $\mathsf{p}[\cdot|x]$, for any $x \in \mathcal{X}$.

13.2 Conditional Maxent principle

As for Maxent models, conditional Maxent or logistic regression models can be derived from a key concentration inequality. By the general Rademacher complexity bound (Theorem 3.3), for any $\delta > 0$, the following inequality holds with probability at least $1 - \delta$ over the choice of a sample of size m:

$$\left\| \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[\mathbf{\Phi}(x,y)] - \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x,y)] \right\|_\infty \leq 2\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\log\frac{2}{\delta}}{2m}}, \tag{13.1}$$

where we denote by $\widehat{\mathcal{D}}$ the empirical distribution defined by the sample S. We will also denote by $\widehat{\mathcal{D}}^1(x)$ the empirical distribution of x in the sample S. For any $x \in \mathcal{X}$, let $\mathsf{p}_0[\cdot|x]$ denote a conditional probability, often chosen to be the uniform distribution. Then, the *conditional Maxent principle* consists of seeking conditional probabilities $\mathsf{p}[\cdot|x]$ that are as agnostic as possible, that is as close as possible to the uniform distribution, or, more generally, to priors $\mathsf{p}_0[\cdot|x]$, while verifying an inequality similar to (13.1):

$$\left\| \mathop{\mathbb{E}}_{\substack{x\sim\widehat{\mathcal{D}}^1\\ y\sim\mathsf{p}[\cdot|x]}}[\mathbf{\Phi}(x,y)] - \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x,y)] \right\|_\infty \leq \lambda, \tag{13.2}$$

where $\lambda \geq 0$ is a parameter. Here, closeness is defined via the *conditional relative entropy* (appendix E) based on the empirical marginal distribution $\widehat{\mathcal{D}}^1$ of input points. Choosing $\lambda = 0$ corresponds to standard *conditional Maxent* or *unregularized conditional Maxent* and to requiring the expectation of the features based on $\mathcal{D}^1$ and the conditional probabilities $\mathsf{p}[\cdot|x]$ to precisely match the empirical averages. As we will see later, its relaxation, that is the inequality case ($\lambda \neq 0$), translates into a regularization. Notice that the conditional Maxent principle does not require specifying a family of conditional probability distributions $\mathcal{P}$ to choose from.

13.3 Conditional Maxent models

Let Δ denote the simplex of the probability distributions over $\mathcal{Y}$, $\mathcal{X}_1 = \text{supp}(\widehat{\mathcal{D}}^1)$ the support of $\widehat{\mathcal{D}}^1$, and $\bar{\mathsf{p}} \in \Delta^{\mathcal{X}_1}$ the family of conditional probabilities, $\bar{\mathsf{p}} = (\mathsf{p}[\cdot|x])_{x\in\mathcal{X}_1}$. Then, the conditional Maxent principle can be formulated as the following optimization problem:

$$\min_{\bar{\mathsf{p}} \in \Delta^{\mathcal{X}_1}} \sum_{x \in \mathcal{X}_1} \widehat{\mathcal{D}}^1(x)\, \mathsf{D}\big(\mathsf{p}[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]\big) \tag{13.3}$$

$$\text{s.t.} \left\| \mathop{\mathbb{E}}_{\substack{x \sim \widehat{\mathcal{D}}^1 \\ y \sim \mathsf{p}[\cdot|x]}} [\mathbf{\Phi}(x,y)] - \mathop{\mathbb{E}}_{(x,y)\sim \widehat{\mathcal{D}}} [\mathbf{\Phi}(x,y)] \right\|_\infty \leq \lambda.$$

This defines a convex optimization problem since the objective is a positive sum of relative entropies and since the relative entropy D is convex with respect to its arguments (appendix E), since the constraints are affine functions of $\bar{\mathsf{p}}$, and since $\Delta^{\mathcal{X}_1}$ is a convex set. The solution is in fact unique, since the objective is strictly convex as a positive sum of relative entropies, each strictly convex. The empirical conditional probabilities $\widehat{\mathcal{D}}^1(\cdot|x)$, $x \in \mathcal{X}_1$, clearly form a feasible solution, thus problem (12.7) is feasible.

For uniform priors $\mathsf{p}_0[\cdot|x]$, problem (13.3) can be equivalently formulated as a conditional entropy maximization, which explains the name given to these models. Let $\bar{\mathsf{H}}(\bar{\mathsf{p}}) = -\mathbb{E}_{x\sim\widehat{\mathcal{D}}^1}\big[\sum_{y\in\mathcal{Y}} \mathsf{p}[y|x]\log \mathsf{p}[y|x]\big]$ denote the conditional entropy of p with respect to the marginal $\widehat{\mathcal{D}}^1$. Then, the objective function of (12.7) can be rewritten as follows:

$$\begin{aligned}
\mathsf{D}\big(\mathsf{p}[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]\big) &= \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}^1}\left[\sum_{y\in\mathcal{Y}} \mathsf{p}[y|x] \log \frac{\mathsf{p}[y|x]}{\mathsf{p}_0[y|x]}\right] \\
&= \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}^1}\left[-\sum_{y\in\mathcal{Y}} \mathsf{p}[y|x]\log(1/c) + \sum_{y\in\mathcal{Y}} \mathsf{p}[y|x]\log \mathsf{p}[y|x]\right] \\
&= \log(c) - \bar{\mathsf{H}}(\bar{\mathsf{p}}).
\end{aligned}$$

Thus, since $\log(c)$ is a constant, minimizing the objective is then equivalent to maximizing $\bar{\mathsf{H}}(\bar{\mathsf{p}})$.

Conditional Maxent models are the solutions of the optimization problem just described. As in the non-conditional case, they admit two important benefits: they are based on a fundamental theoretical guarantee of closeness of empirical and true feature averages, and they do not require specifying a particular family of distributions $\mathcal{P}$. In the next sections, we will further analyze the properties of conditional Maxent models.

13.4 Dual problem

Here, we derive an equivalent dual problem for (13.3) which, as we will show, can be formulated as a regularized conditional maximum likelihood problem over the family of *Gibbs distributions*.

The Maxent optimization problem (13.3) can be equivalently expressed as the unconstrained optimization problem $\min_{\bar{\mathsf{p}}} F(\bar{\mathsf{p}})$ with, for all $\bar{\mathsf{p}} = (\mathsf{p}[\cdot|x] \in (\mathbb{R}^{\mathcal{Y}})^{\mathcal{X}_1}$,

$$F(\bar{\mathsf{p}}) = \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}^1}\Big[\widetilde{\mathsf{D}}\big(\mathsf{p}[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]\big)\Big] + I_{\mathcal{C}}\Bigg(\mathop{\mathbb{E}}_{\substack{x\sim\widehat{\mathcal{D}}^1\\ y\sim\mathsf{p}[\cdot|x]}}[\mathbf{\Phi}(x,y)]\Bigg), \tag{13.4}$$

with $\widetilde{\mathsf{D}}\big(\mathsf{p}[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]\big) = \mathsf{D}\big(\mathsf{p}[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]\big)$ if $\mathsf{p}[\cdot|x]$ is in Δ, $\widetilde{\mathsf{D}}\big(\mathsf{p}[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]\big) = +\infty$ otherwise, and with $\mathcal{C} = \{\mathbf{u} \in \mathbb{R}^N\colon \|\mathbf{u} - \mathbb{E}_{(x,y)\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x,y)]\,\|_\infty \leq \lambda\}$, which is a convex set.

Let G be the function defined for all $\mathbf{w} \in \mathbb{R}^N$ by

$$G(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^m \log\left[\frac{\mathsf{p}_{\mathbf{w}}[y_i|x_i]}{\mathsf{p}_0[y_i|x_i]}\right] - \lambda\|\mathbf{w}\|_1\,, \tag{13.5}$$

with, for all $x \in \mathcal{X}_1$ and $y \in \mathcal{Y}$,

$$\mathsf{p}_{\mathbf{w}}[y|x] = \frac{\mathsf{p}_0[y|x]e^{\mathbf{w}\cdot\mathbf{\Phi}(x,y)}}{Z(\mathbf{w},x)} \quad \text{and} \quad Z(\mathbf{w},x) = \sum_{y\in\mathcal{Y}} \mathsf{p}_0[y|x]e^{\mathbf{w}\cdot\mathbf{\Phi}(x,y)}. \tag{13.6}$$

Then, the following theorem gives a result similar to the duality theorem presented in the non-conditional case (Theorem 12.2, Section 12.5).

Theorem 13.1 *Problem* (13.3) *is equivalent to the dual optimization problem* $\sup_{\mathbf{w}\in\mathbb{R}^N} G(\mathbf{w})$*:*

$$\sup_{\mathbf{w}\in\mathbb{R}^N} G(\mathbf{w}) = \min_{\bar{\mathsf{p}}\in(\mathbb{R}^{\mathcal{Y}})^{\mathcal{X}_1}} F(\bar{\mathsf{p}}). \tag{13.7}$$

Furthermore, let $\bar{\mathsf{p}}^* = \operatorname{argmin}_{\bar{\mathsf{p}}} F(\bar{\mathsf{p}})$*. Then, for any* $\epsilon > 0$ *and any* $\mathbf{w}$ *such that* $|G(\mathbf{w}) - \sup_{\mathbf{w}\in\mathbb{R}^N} G(\mathbf{w})| < \epsilon$*, we have* $\mathbb{E}_{x\sim\widehat{\mathcal{D}}^1}\big[\mathsf{D}\big(\bar{\mathsf{p}}^*[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]\big)\big] \leq \epsilon$*.*

The proof is similar to that of Theorem 12.2 and is given at the end of this chapter since it is somewhat longer (Section 13.9).

In view of the theorem, if $\mathbf{w}$ is an ϵ-solution of the dual optimization problem, then $\mathbb{E}_{x\sim\widehat{\mathcal{D}}^1}\big[\mathsf{D}\big(\mathsf{p}^*[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]\big)\big] \leq \epsilon$, which, by Jensen's inequality and Pinsker's inequality (Proposition E.7) implies that

$$\mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}^1}\Big[\big\|\mathsf{p}^*[\cdot|x] - \mathsf{p}_{\mathbf{w}}[\cdot|x]\big\|_1\Big] \leq \sqrt{\mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}^1}\Big[\big\|\mathsf{p}^*[\cdot|x] - \mathsf{p}_{\mathbf{w}}[\cdot|x]\big\|_1^2\Big]} \leq \sqrt{2\epsilon}.$$

Thus, $\mathsf{p}_{\mathbf{w}}[\cdot|x]$ is then $\sqrt{2\epsilon}$-close in $\widehat{\mathcal{D}}^1$-averaged L_1-norm to the optimal solution of the primal and the theorem suggests that the solution of the conditional Maxent problem can be determined by solving the dual problem, which can be written equivalently as follows for a uniform prior:

$$\inf_{\mathbf{w}} \lambda\|\mathbf{w}\|_1 - \frac{1}{m}\sum_{i=1}^m \log\big[\mathsf{p}_{\mathbf{w}}[y_i|x_i]\big]. \tag{13.8}$$

Similar remarks to those made for non-conditional Maxent models apply here. In particular, the solution may not be achieved for any finite $\mathbf{w}$ for $\lambda = 0$, which is why the infimum is needed. Also, this result may seem surprising since it shows that conditional Maxent coincides with conditional Maximum Likelihood ($\lambda = 0$) or regularized conditional Maximum Likelihood ($\lambda > 0$) using for the family $\mathcal{P}$ of conditional probabilities to choose from that of Gibbs distributions, while the conditional Maxent principle does not explicitly specify any family of conditional probabilities $\mathcal{P}$. The reason is the specific choice of the conditional relative entropy as the measure of closeness of $\mathsf{p}[\cdot|x]$ to the prior conditional distributions $\mathsf{p}_0[\cdot|x]$. Other measures of closeness between distributions lead to different forms for the solution. Thus, in some sense, the choice of the measure of closeness is the (dual) counterpart of that of the family of conditional distributions in maximum likelihood. Also, as already mentioned in the standard Maxent case, Gibbs distributions form a very rich family.

Notice that both the primal and the dual optimization problems for conditional Maxent involve only conditional probabilities $\mathsf{p}[\cdot|x]$ for x in $\mathcal{X}_1$, that is for x in the training sample. Thus, they do not provide us with any information about other conditional probabilities. However, the dual shows that, for x in $\mathcal{X}_1$, the solution admits the same general form $\mathsf{p}_{\mathbf{w}}[\cdot|x]$, which only depends on the weight vector $\mathbf{w}$. In view of that, we extend the definition of Maxent conditional probabilities to all $x \in \mathcal{X}$ by using the same general form $\mathsf{p}_{\mathbf{w}}[\cdot|x]$ and the same vector $\mathbf{w}$ for all x.

Observe also that in the definition of the primal or dual problems we could have used some other distribution $\mathcal{Q}$ over $\mathcal{X}$ in lieu of $\widehat{\mathcal{D}}^1$. It is straightforward to verify that the duality theorem would continue to hold in that case using the same proof. In fact, ideally, we would have chosen $\mathcal{Q}$ to be $\mathcal{D}^1$. However, that optimization problem would require knowledge of the feature vectors for all $x \in \operatorname{supp}(\mathcal{D}^1)$, which of course is not accessible to us given a finite sample. The weighted vector $\mathbf{w}$ found when using $\widehat{\mathcal{D}}^1$ can be viewed as an approximation of the one obtained if using $\mathcal{D}^1$.

13.5 Properties

In this section, we discuss several aspects of conditional Maxent models, including the form of the dual optimization problems, the feature vectors used, and prediction with these models.

13.5.1 Optimization problem

L_1-regularized conditional Maxent models are therefore conditional probability models solutions of the primal problem (13.3) or, equivalently, models defined by

$$\mathsf{p}_{\mathbf{w}}[y|x] = \frac{e^{\mathbf{w}\cdot\mathbf{\Phi}(x,y)}}{Z(x)} \quad \text{and} \quad Z(x) = \sum_{y\in\mathcal{Y}} e^{\mathbf{w}\cdot\mathbf{\Phi}(x,y)}, \tag{13.9}$$

where $\mathbf{w}$ is solution of the dual problem

$$\min_{\mathbf{w}\in\mathbb{R}^N} \ \lambda\|\mathbf{w}\|_1 - \frac{1}{m}\sum_{i=1}^{m} \log \mathsf{p}_{\mathbf{w}}[y_i|x_i],$$

with $\lambda \geq 0$ is a parameter. Using the expression of the conditional probabilities, this optimization problem can be written more explicitly as

$$\min_{\mathbf{w}\in\mathbb{R}^N} \lambda\|\mathbf{w}\|_1 + \frac{1}{m}\sum_{i=1}^{m} \log\left[\sum_{y\in\mathcal{Y}} e^{\mathbf{w}\cdot\mathbf{\Phi}(x_i,y)-\mathbf{w}\cdot\mathbf{\Phi}(x_i,y_i)}\right]. \tag{13.10}$$

or, equivalently, as

$$\min_{\mathbf{w}\in\mathbb{R}^N} \lambda\|\mathbf{w}\|_1 - \mathbf{w}\cdot\frac{1}{m}\sum_{i=1}^{m}\mathbf{\Phi}(x_i,y_i) + \frac{1}{m}\sum_{i=1}^{m}\log\left[\sum_{y\in\mathcal{Y}} e^{\mathbf{w}\cdot\mathbf{\Phi}(x_i,y)}\right]. \tag{13.11}$$

By definition of the dual problem, this is an unconstrained convex optimization problem in $\mathbf{w}$. This can be also seen from the fact that the log-sum function $\mathbf{w} \mapsto \log\left[\sum_{y\in\mathcal{Y}} e^{\mathbf{w}\cdot\mathbf{\Phi}(x,y)}\right]$ is convex for any $x \in \mathcal{X}$.

There are many optimization solutions available for this problem, including several special-purpose algorithms, general first-order and second-order solutions, and special-purpose distributed solutions. One common method is simply to use stochastic gradient descent (SGD), which has been reported to be more efficient than most special-purpose methods in applications. When the dimension of the feature vectors $\mathbf{\Phi}$ (or the cardinality of the family of feature functions $\mathcal{H}$) is very large, these methods are typically inefficient. An alternative method then consists of applying coordinate descent to solve this problem. In that case, the resulting algorithm coincides with the version of L_1-regularized boosting where, instead of the exponential function, the logistic function is used.

13.5.2 Feature vectors

Using feature vectors $\mathbf{\Phi}(x,y)$ depending on both the input x and the output y is often important in applications. For example, in machine translation, it is convenient to use features whose values may depend on the presence of some words in the input sentence and some others in the output sequence. A common choice of the feature vector is however one where the column vectors $\mathbf{\Phi}(x,y)$ and $\mathbf{w}$ admit c

blocks of equal size and where only the block in $\mathbf{\Phi}(x, y)$ corresponding to the class y is non-zero and equal to a feature vector $\mathbf{\Gamma}(x)$ independent of the class labels:

$$\mathbf{\Phi}(x, y) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \mathbf{\Gamma}(x) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_{y-1} \\ \mathbf{w}_y \\ \mathbf{w}_{y+1} \\ \vdots \\ \mathbf{w}_c \end{bmatrix}.$$

In view of that, the inner product of $\mathbf{w}$ and $\mathbf{\Phi}(x, y)$ can be expressed in terms of the feature vector $\mathbf{\Gamma}(x)$, which only depends on x, but with a distinct parameter vector $\mathbf{w}_y$:

$$\mathbf{w} \cdot \mathbf{\Phi}(x, y) = \mathbf{w}_y \cdot \mathbf{\Gamma}(x).$$

The optimization problem for L_1-regularized conditional Maxent can then be written in terms of the vectors $\mathbf{w}_y$ as follows:

$$\min_{\mathbf{w} \in \mathbb{R}^N} \lambda \sum_{y \in \mathcal{Y}} \|\mathbf{w}_y\|_1 + \frac{1}{m} \sum_{i=1}^{m} \log \left[\sum_{y \in \mathcal{Y}} e^{\mathbf{w}_y \cdot \mathbf{\Gamma}(x_i) - \mathbf{w}_{y_i} \cdot \mathbf{\Gamma}(x_i)} \right]. \tag{13.12}$$

Notice that, if the vectors $\mathbf{w}_y$ were not correlated via the second term of the objective function (for example if, instead of the log of the sum, this term were replaced by the sum of the logs), then the problem would be reduced to c separate optimization functions learning a distinct weight vector for each class, as in the one-vs-all setup of multi-class classification.

13.5.3 Prediction

Finally, note that the class $\widehat{y}(x)$ predicted by a conditional Maxent model with parameter $\mathbf{w}$ is given by

$$\widehat{y}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathsf{p}_{\mathbf{w}}[y|x] = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbf{w} \cdot \mathbf{\Phi}(x, y). \tag{13.13}$$

Thus, conditional Maxent models define linear classifiers. Conditional Maxent models are also sometimes referred to as log-*linear models*.

13.6 Generalization bounds

In this section, we will present learning guarantees for conditional Maxent models in two different settings: one where the dimension of the feature vectors $\mathbf{\Phi}$ (or the cardinality of the family of feature functions $\mathcal{H}$) is infinite or extremely large and where a coordinate-descent or boosting-type algorithm is more suitable, and another one where the dimension of the feature vectors $\mathbf{\Phi}$ is finite and not too large.

We start with the case where the dimension of the feature vectors $\mathbf{\Phi}$ is very large. The following margin-based guarantee holds in that case.

Theorem 13.2 *For any $\delta > 0$, with probability at least $1-\delta$ over the draw of an i.i.d. sample S of size m, the following holds for all $\rho > 0$ and $f \in \mathcal{F} = \{(x, y) \mapsto \mathbf{w} \cdot \mathbf{\Phi}(x, y) \colon \|\mathbf{w}\|_1 \leq 1\}$:*

$$R(f) \leq \frac{1}{m} \sum_{i=1}^{m} \log_{u_0}\Big(\sum_{y \in \mathcal{Y}} e^{\frac{f(x_i, y) - f(x_i, y_i)}{\rho}}\Big) + \frac{8c}{\rho} \mathfrak{R}_m(\Pi_1(\mathcal{H})) + \sqrt{\frac{\log \log_2 \frac{4r}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}},$$

where $u_0 = \log(1 + 1/e)$ and $\Pi_1(\mathcal{H}) = \{x \mapsto \phi(x, y) \colon \phi \in \mathcal{H}, y \in \mathcal{Y}\}$.

Proof: For any $f \colon (x, y) \mapsto \mathbf{w} \cdot \mathbf{\Phi}(x, y)$ and $i \in [m]$, let $\rho_f(x_i, y_i)$ denote the margin of f at (x_i, y_i):

$$\rho_f(x_i, y_i) = \min_{y \neq y_i} f(x_i, y_i) - f(x_i, y) = \min_{y \neq y_i} \mathbf{w} \cdot (\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)).$$

Fix $\rho > 0$. Then, by Theorem 9.2, for any $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds for all $f \in \mathcal{H}$ and $\rho \in (0, 2r]$:

$$R(f) \leq \frac{1}{m} \sum_{i=1}^{m} 1_{\rho_f(x_i, y_i) \leq \rho} + \frac{4c}{\rho} \mathfrak{R}_m(\Pi_1(\mathcal{F})) + \sqrt{\frac{\log \log_2 \frac{4r}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}},$$

where $\Pi_1(\mathcal{F}) = \{x \mapsto f(x, y) \colon y \in \mathcal{Y}, f \in \mathcal{H}\}$. The inequality trivially holds for all $\rho > 0$ since for $\rho \geq 2r$, by Hölder's inequality, we have $|\mathbf{w} \cdot \mathbf{\Phi}(x, y)| \leq \|\mathbf{w}\|_1 \, \|\mathbf{\Phi}(x, y)\|_\infty \leq r$ for $\|\mathbf{w}\|_1 \leq 1$, and thus $\min_{y \neq y_i} f(x_i, y_i) - f(x_i, y) \leq 2r \leq \rho$ for all $i \in [m]$ and $y \in \mathcal{Y}$. Now, for any $\rho > 0$, the ρ-margin loss can be upper bounded by the ρ-logistic loss:

$$\forall u \in \mathbb{R}, 1_{u \leq \rho} = 1_{\frac{u}{\rho} - 1 \leq 0} \leq \log_{u_0}(1 + e^{-\frac{u}{\rho}}).$$

Thus, the ρ-margin loss of f at (x_i, y_i) can be upper bounded as follows:

$$\begin{aligned}
1_{\rho_f(x_i, y_i) \leq \rho} &\leq \log_{u_0}(1 + e^{-\frac{\rho(f, x_i, y_i)}{\rho}}) \\
&= \log_{u_0}(1 + \max_{y \neq y_i} e^{\frac{f(x_i, y) - f(x_i, y_i)}{\rho}}) \\
&\leq \log_{u_0}\Big(1 + \sum_{y \neq y_i} e^{\frac{f(x_i, y) - f(x_i, y_i)}{\rho}}\Big) = \log_{u_0}\Big(\sum_{y \in \mathcal{Y}} e^{\frac{f(x_i, y) - f(x_i, y_i)}{\rho}}\Big).
\end{aligned}$$

Thus, with probability at least $1 - \delta$, the following inequality holds for all $f \in \mathcal{H}$ and $\rho > 0$:

$$R(f) \leq \frac{1}{m} \sum_{i=1}^{m} \log_{u_0}\Big(\sum_{y \in \mathcal{Y}} e^{\frac{f(x_i, y) - f(x_i, y_i)}{\rho}}\Big) + \frac{4c}{\rho} \mathfrak{R}_m(\Pi_1(\mathcal{F})) + \sqrt{\frac{\log \log_2 \frac{4r}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}.$$

For any sample $S = (x_1, \ldots, x_m)$ of size m, the empirical Rademacher complexity of $\Pi_1(\mathcal{F})$ can be bounded as follows:

$$\begin{aligned}
\widehat{\mathfrak{R}}_S(\Pi_1(\mathcal{F})) &= \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\substack{\|\mathbf{w}\|_1 \leq 1 \\ y \in \mathcal{Y}}} \sum_{i=1}^{m} \sigma_i \sum_{j=1}^{N} w_j \Phi_j(x_i, y) \right] \\
&= \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\substack{\|\mathbf{w}\|_1 \leq 1 \\ y \in \mathcal{Y}}} \sum_{j=1}^{N} w_j \sum_{i=1}^{m} \sigma_i \Phi_j(x_i, y) \right] \\
&= \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\substack{j \in [N] \\ y \in \mathcal{Y}}} \left| \sum_{i=1}^{m} \sigma_i \Phi_j(x_i, y) \right| \right] \\
&\leq \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\substack{\Phi \in \mathcal{H} \\ y \in \mathcal{Y}}} \left| \sum_{i=1}^{m} \sigma_i \Phi(x_i, y) \right| \right] \leq 2\,\widehat{\mathfrak{R}}_S(\Pi_1(\mathcal{H})),
\end{aligned}$$

which completes the proof. □

The learning guarantee of the theorem is remarkable since it does not depend on the dimension N and since it only depends on the complexity of the family $\mathcal{H}$ of feature functions (or base hypotheses). Since for any $\rho > 0$, f/ρ admits the same generalization error as f, the theorem implies that with probability at least $1 - \delta$, the following inequality holds for all $f \in \{(x, y) \mapsto \mathbf{w} \cdot \boldsymbol{\Phi}(x, y) \colon \|\mathbf{w}\|_1 \leq \frac{1}{\rho}\}$ and $\rho > 0$:

$$R(f) \leq \frac{1}{m} \sum_{i=1}^{m} \log_{u_0}\!\left(\sum_{y \in \mathcal{Y}} e^{f(x_i, y) - f(x_i, y_i)} \right) + \frac{8c}{\rho} \mathfrak{R}_m(\Pi_1(\mathcal{H})) + \sqrt{\frac{\log \log_2 \frac{4r}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}.$$

This inequality can be used to derive an algorithm that selects $\mathbf{w}$ and $\rho > 0$ to minimize the right-hand side. The minimization with respect to ρ does not lead to a convex optimization and depends on theoretical constant factors affecting the second and third term. Thus, instead, ρ is left as a free parameter of the algorithm, typically determined via cross-validation.

Now, since only the first term of the right-hand side depends on $\mathbf{w}$, for any $\rho > 0$, the bound suggests selecting $\mathbf{w}$ as the solution of the following optimization problem:

$$\min_{\|\mathbf{w}\|_1 \leq \frac{1}{\rho}} \frac{1}{m} \sum_{i=1}^{m} \log \left(\sum_{y \in \mathcal{Y}} e^{\mathbf{w} \cdot \boldsymbol{\Phi}(x_i, y) - \mathbf{w} \cdot \boldsymbol{\Phi}(x_i, y_i)} \right). \tag{13.14}$$

Introducing a Lagrange variable $\lambda \geq 0$, the optimization problem can be written equivalently as

$$\min_{\mathbf{w}} \lambda \|\mathbf{w}\|_1 + \frac{1}{m} \sum_{i=1}^{m} \log \left(\sum_{y \in \mathcal{Y}} e^{\mathbf{w} \cdot \mathbf{\Phi}(x_i, y) - \mathbf{w} \cdot \mathbf{\Phi}(x_i, y_i)} \right). \tag{13.15}$$

Since for any choice of ρ in the constraint of (13.14), there exists an equivalent dual variable λ in the formulation of (13.15) that achieves the same optimal $\mathbf{w}$, λ can be freely selected via cross-validation. The resulting algorithm precisely coincides with conditional Maxent.

When the dimension N of the feature vectors $\mathbf{\Phi}$ is finite, the following margin-based guarantee holds.

Theorem 13.3 *For any $\delta > 0$, with probability at least $1 - \delta$ over the draw of an i.i.d. sample S of size m, the following holds for all $\rho > 0$ and $f \in \mathcal{F} = \{(x, y) \mapsto \mathbf{w} \cdot \mathbf{\Phi}(x, y) \colon \|\mathbf{w}\|_1 \leq 1\}$:*

$$R(f) \leq \frac{1}{m} \sum_{i=1}^{m} \log_{u_0}\left(\sum_{y \in \mathcal{Y}} e^{\frac{f(x_i, y) - f(x_i, y_i)}{\rho}} \right) + \frac{4cr\sqrt{2\log(2cN)}}{\rho} + \sqrt{\frac{\log \log_2 \frac{4r}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}},$$

where $u_0 = \log(1 + 1/e)$.

Proof: The proof coincides with that of Theorem 13.2, modulo the upper bound on $\mathfrak{R}_m(\Pi_1(\mathcal{F}))$. For any sample $S = (x_1, \ldots, x_m)$ of size m, the empirical Rademacher complexity of $\Pi_1(\mathcal{F})$ can be bounded as follows:

$$\begin{aligned}
\widehat{\mathfrak{R}}_S(\Pi_1(\mathcal{F})) &= \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\substack{\|\mathbf{w}\|_1 \leq 1 \\ y \in \mathcal{Y}}} \sum_{i=1}^{m} \sigma_i \mathbf{w} \cdot \mathbf{\Phi}(x_i, y) \right] \\
&= \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\substack{\|\mathbf{w}\|_1 \leq 1 \\ y \in \mathcal{Y}}} \mathbf{w} \cdot \sum_{i=1}^{m} \sigma_i \mathbf{\Phi}(x_i, y) \right] \\
&= \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{y \in \mathcal{Y}} \left\| \sum_{i=1}^{m} \sigma_i \mathbf{\Phi}(x_i, y) \right\|_\infty \right] \\
&= \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{\substack{j \in [N] \\ y \in \mathcal{Y}, s \in \{-1, +1\}}} s \sum_{i=1}^{m} \sigma_i \Phi_j(x_i, y) \right] \\
&\leq r\sqrt{2\log(2cN)},
\end{aligned}$$

where the third equality holds by definition of the dual norm, and the last inequality by the maximal inequality (Corollary D.11), since the supremum is taken over $2cN$ choices. □

This learning guarantee of the theorem is very favorable even for relatively high-dimensional problems since its dependency on the dimension N is only logarithmic.

13.7 Logistic regression

The binary case of conditional Maxent models ($c = 2$) is known as *logistic regression* and is one of the most well-known algorithms for binary classification.

13.7.1 Optimization problem

In the binary case, the sum appearing in the optimization problem of conditional Maxent models can be simplified as follows:

$$\begin{aligned}\sum_{y \in \mathcal{Y}} e^{\mathbf{w}\cdot\mathbf{\Phi}(x_i,y)-\mathbf{w}\cdot\mathbf{\Phi}(x_i,y_i)} &= e^{\mathbf{w}\cdot\mathbf{\Phi}(x_i,+1)-\mathbf{w}\cdot\mathbf{\Phi}(x_i,y_i)} + e^{\mathbf{w}\cdot\mathbf{\Phi}(x_i,-1)-\mathbf{w}\cdot\mathbf{\Phi}(x_i,y_i)} \\ &= 1 + e^{-y_i\mathbf{w}\cdot[\mathbf{\Phi}(x_i,+1)-\mathbf{\Phi}(x_i,-1)]} \\ &= 1 + e^{-y_i\mathbf{w}\cdot\mathbf{\Psi}(x_i)},\end{aligned}$$

where for all $x \in \mathcal{X}$, $\mathbf{\Psi}(x) = \mathbf{\Phi}(x,+1) - \mathbf{\Phi}(x,-1)$. This leads to the following optimization problem, which defines L_1*-regularized logistic regression*:

$$\min_{\mathbf{w}\in\mathbb{R}^N} \lambda\|\mathbf{w}\|_1 + \frac{1}{m}\sum_{i=1}^{m}\log\Big[1 + e^{-y_i\mathbf{w}\cdot\mathbf{\Psi}(x_i)}\Big]. \tag{13.16}$$

As discussed in the general case, this is a convex optimization problem which admits a variety of different solutions. A common solution is SGD, another one is coordinate descent. When coordinate descent is used, then the algorithm coincides with the alternative to AdaBoost where the logistic loss is used instead of the exponential loss ($\phi(-u) = \log_2(1 + e^{-u}) \geq 1_{u\leq 0}$).

13.7.2 Logistic model

In the binary case, the conditional probability defined by the weight vector $\mathbf{w}$ can be expressed as follows:

$$\mathsf{p}_{\mathbf{w}}[y = +1 \mid x] = \frac{e^{\mathbf{w}\cdot\mathbf{\Phi}(x,+1)}}{Z(x)}, \tag{13.17}$$

with $Z(x) = e^{\mathbf{w}\cdot\mathbf{\Phi}(x,+1)} + e^{\mathbf{w}\cdot\mathbf{\Phi}(x,-1)}$. Thus, prediction is based on a linear decision rule defined by the sign of log-odds ratio:

$$\log\frac{\mathsf{p}_{\mathbf{w}}[y = +1 \mid x]}{\mathsf{p}_{\mathbf{w}}[y = -1 \mid x]} = \mathbf{w}\cdot\big(\mathbf{\Phi}(x,+1) - \mathbf{\Phi}(x,-1)\big) = \mathbf{w}\cdot\mathbf{\Psi}(x).$$

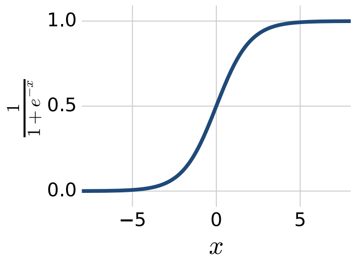


Figure 13.1
Plot of the logistic function f_{logistic}.

This is why logistic regression is also known as a log-*linear model*. Observe also that the conditional probability admits the following *logistic form*:

$$\mathsf{p}_{\mathbf{w}}[y=+1 \mid x] = \frac{1}{1+e^{-\mathbf{w}\cdot[\mathbf{\Phi}(x,+1)-\mathbf{\Phi}(x,-1)]}} = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{\Psi}(x)}} = f_{\text{logistic}}\big(\mathbf{w}\cdot\mathbf{\Psi}(x)\big),$$

where f_{logistic} is the function defined over $\mathbb{R}$ by $f_{\text{logistic}}\colon x \mapsto \frac{1}{1+e^{-x}}$. Figure 13.1 shows the plot of this function. The logistic function maps the images of the linear function $x \mapsto \mathbf{\Psi}(x)$ to the interval $[0,1]$, which makes them interpretable as probabilities.

L_1-regularized logistic regression benefits from the strong learning guarantees already presented for conditional maxent models, in the special case of two classes ($c = 2$). The learning guarantees for L_2-regularized logistic regression will be similarly special cases of those presented in the next section.

13.8 L_2-regularization

A common variant of conditional Maxent models is one where the dimension N is finite and where the regularization is based on the norm-2 squared of the weight vector $\mathbf{w}$. The optimization problem is thus given by

$$\min_{\mathbf{w}\in\mathbb{R}^N} \ \lambda\|\mathbf{w}\|_2^2 - \frac{1}{m}\sum_{i=1}^{m} \log \mathsf{p}_{\mathbf{w}}[y_i|x_i],$$

where for all $(x,y) \in \mathcal{X}\times\mathcal{Y}$,

$$\mathsf{p}_{\mathbf{w}}[y|x] = \frac{\exp(\mathbf{w}\cdot\mathbf{\Phi}(x,y))}{Z(x)} \quad \text{and} \quad Z(x) = \sum_{y\in\mathcal{Y}} \exp(\mathbf{w}\cdot\mathbf{\Phi}(x,y)). \tag{13.18}$$

As for the norm-1 regularization, there are many optimization solutions available for this problem, including special-purpose algorithms, general first-order and second-order solutions, and special-purpose distributed solutions. Here, the objective is additionally differentiable. A common optimization method is simply stochastic gradient descent (SGD).

In contrast to norm-1-regularized conditional Maxent models, which lead to sparser weight vectors, norm-2 conditional Maxent models lead to non-sparse solutions, which may be preferable and lead to more accurate solutions in some applications such as natural language processing. The following margin-based guarantee holds for norm-2 regularized conditional Maxent, assuming that the norm-2 of the feature vector is bounded.

Theorem 13.4 *For any $\delta > 0$, with probability at least $1 - \delta$ over the draw of an i.i.d. sample S of size m, the following holds for all $\rho > 0$ and $f \in \mathcal{F} = \{(x, y) \mapsto \mathbf{w} \cdot \mathbf{\Phi}(x, y) \colon \|\mathbf{w}\|_2 \leq 1\}$:*

$$R(f) \leq \frac{1}{m} \sum_{i=1}^{m} \log_{u_0}\Big(\sum_{y \in \mathcal{Y}} e^{\frac{f(x_i, y) - f(x_i, y_i)}{\rho}}\Big) + \frac{4 r_2 c^2}{\rho \sqrt{m}} + \sqrt{\frac{\log \log_2 \frac{4 r_2}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}},$$

where $u_0 = \log(1 + 1/e)$ and $r_2 = \sup_{(x,y)} \|\mathbf{\Phi}(x, y)\|_2$.

Proof: The proof is similar to that of Theorem 13.3, modulo the observation that here $|\mathbf{w} \cdot \mathbf{\Phi}(x, y)| \leq \|\mathbf{w}\|_2 \|\mathbf{\Phi}(x, y)\|_2 \leq r_2$ and modulo the upper bound on $\mathfrak{R}_m(\Pi_1(\mathcal{F}))$. For any sample $S = (x_1, \ldots, x_m)$ of size m, the empirical Rademacher complexity of $\Pi_1(\mathcal{F})$ can be bounded as follows:

$$\begin{aligned}
\widehat{\mathfrak{R}}_S(\Pi_1(\mathcal{F})) &= \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Bigg[\sup_{\substack{\|\mathbf{w}\|_2 \leq 1 \\ y \in \mathcal{Y}}} \sum_{i=1}^{m} \sigma_i \mathbf{w} \cdot \mathbf{\Phi}(x_i, y)\Bigg] \\
&= \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Bigg[\sup_{\substack{\|\mathbf{w}\|_2 \leq 1 \\ y \in \mathcal{Y}}} \mathbf{w} \cdot \sum_{i=1}^{m} \sigma_i \mathbf{\Phi}(x_i, y)\Bigg] \\
&= \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Bigg[\sup_{y \in \mathcal{Y}} \Big\|\sum_{i=1}^{m} \sigma_i \mathbf{\Phi}(x_i, y)\Big\|_2\Bigg] \\
&\leq \frac{1}{m} \sum_{y \in \mathcal{Y}} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Bigg[\Big\|\sum_{i=1}^{m} \sigma_i \mathbf{\Phi}(x_i, y)\Big\|_2\Bigg] \\
&\leq \frac{1}{m} \sum_{y \in \mathcal{Y}} \sqrt{\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Bigg[\Big\|\sum_{i=1}^{m} \sigma_i \mathbf{\Phi}(x_i, y)\Big\|_2^2\Bigg]} \\
&= \frac{1}{m} \sum_{y \in \mathcal{Y}} \sqrt{\sum_{i=1}^{m} \|\mathbf{\Phi}(x_i, y)\|_2^2\Bigg]} \leq \frac{r_2 c}{\sqrt{m}},
\end{aligned}$$

where the third equality holds by definition of the dual norm, and the second inequality by Jensen's inequality. □

The learning guarantee of the theorem for L_2-regularized conditional maxent models admits the advantage that the bound does not depend on the dimension. It can be very favorable for r_2 relatively small. The algorithm can then be very effective, provided that a small error can be achieved by a non-sparse weight vector.

13.9 Proof of the duality theorem

In this section, we give the full proof of Theorem 13.1.

Proof: The proof is similar to that of Theorem 12.2 and follows by application of the Fenchel duality theorem (theorem B.39) to the optimization problem (13.4) with the functions f and g defined for all $\bar{\mathsf{p}} \in (\mathbb{R}^{\mathcal{Y}})^{\mathcal{X}_1}$ and $\mathbf{u} \in \mathbb{R}^N$ by $f(\bar{\mathsf{p}}) = \mathbb{E}_{x\sim\widehat{\mathcal{D}}^1}\Big[\widetilde{\mathsf{D}}\big(\mathsf{p}[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]\big)\Big]$, $g(\mathbf{u}) = I_{\mathcal{C}}(\mathbf{u})$ and $A\mathsf{p} = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \widehat{\mathcal{D}}^1(x)\mathsf{p}[y|x]\mathbf{\Phi}(x,y)$. A is a bounded linear map since we have $\|A\bar{\mathsf{p}}\| \leq \|\bar{\mathsf{p}}\|_1 \sup_{x\in\mathcal{X},y\in\mathcal{Y}} \|\mathbf{\Phi}(x,y)\|_\infty \leq r\|\bar{\mathsf{p}}\|_1$ for any $\bar{\mathsf{p}} \in (\mathbb{R}^{\mathcal{Y}})^{\mathcal{X}_1}$. Also, notice that the conjugate of A is given for all $\mathbf{w} \in \mathbb{R}^N$ and $(x,y) \in \mathcal{X}_1 \times \mathcal{Y}$ by $(A^*\mathbf{w})(x,y) = \mathbf{w}\cdot\big(\widehat{\mathcal{D}}^1(x)\mathbf{\Phi}(x,y)\big)$.

Consider $\mathbf{u}_0 \in \mathbb{R}^N$ defined by $\mathbf{u}_0 = \mathbb{E}_{(x,y)\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x,y)] = A\bar{\mathsf{p}}_0$ with $\bar{\mathsf{p}}_0 = (\mathcal{D}(\cdot|x))_{x\in\mathcal{X}_1}$. Since $\bar{\mathsf{p}}_0$ is in $\operatorname{dom}(f) = \Delta^{\mathcal{X}_1}$, $\mathbf{u}_0$ is in $A(\operatorname{dom}(f))$. Furthermore, since λ is positive, $\mathbf{u}_0$ is contained in $\operatorname{int}(\mathcal{C})$. $g = I_{\mathcal{C}}$ equals zero over $\operatorname{int}(\mathcal{C})$ and is therefore continuous over $\operatorname{int}(\mathcal{C})$, thus g is continuous at $\mathbf{u}_0$ and we have $\mathbf{u}_0 \in A(\operatorname{dom}(f)) \cap \operatorname{cont}(g)$. Thus, the assumptions of Theorem B.39 hold.

The conjugate function of f is defined for all $\bar{\mathsf{q}} = (\mathsf{q}[\cdot|x])_{x\in\mathcal{X}_1} \in (\mathbb{R}^{\mathcal{Y}})^{\mathcal{X}_1}$ by

$$\begin{aligned}
f^*(\bar{\mathsf{q}}) &= \sup_{\bar{\mathsf{p}}\in(\mathbb{R}^{\mathcal{Y}})^{\mathcal{X}_1}} \Big\{ \langle \mathsf{p}, \mathsf{q}\rangle - \sum_{x\in\mathcal{X}} \widehat{\mathcal{D}}^1(x)\, \widetilde{\mathsf{D}}(\mathsf{p}[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]) \Big\} \\
&= \sup_{\bar{\mathsf{p}}\in(\mathbb{R}^{\mathcal{Y}})^{\mathcal{X}_1}} \Bigg\{ \sum_{x\in\mathcal{X}_1} \widehat{\mathcal{D}}^1[x] \sum_{y\in\mathcal{Y}} \frac{\mathsf{p}[y|x]\mathsf{q}[y|x]}{\widehat{\mathcal{D}}^1[x]} - \sum_{x\in\mathcal{X}_1} \widehat{\mathcal{D}}^1[x]\, \widetilde{\mathsf{D}}(\mathsf{p}[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]) \Bigg\} \\
&= \sum_{x\in\mathcal{X}_1} \widehat{\mathcal{D}}^1(x) \sup_{\bar{\mathsf{p}}\in(\mathbb{R}^{\mathcal{Y}})^{\mathcal{X}_1}} \Bigg\{ \sum_{y\in\mathcal{Y}} \mathsf{p}[y|x]\left[\frac{\mathsf{q}[y|x]}{\widehat{\mathcal{D}}^1(x)}\right] - \widetilde{\mathsf{D}}(\mathsf{p}[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x]) \Bigg\} \\
&= \sum_{x\in\mathcal{X}_1} \widehat{\mathcal{D}}^1(x) f_x^*\left(\frac{\mathsf{q}[y|x]}{\widehat{\mathcal{D}}^1(x)}\right),
\end{aligned}$$

where, for all $x \in \mathcal{X}_1$ and $\mathsf{p} \in \mathbb{R}^{\mathcal{X}_1}$, f_x is defined by $f_x(\bar{\mathsf{p}}) = \widetilde{\mathsf{D}}(\mathsf{p}[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x])$. By Lemma B.37, the conjugate function f_x^* is given for all $\bar{\mathsf{q}} \in (\mathbb{R}^{\mathcal{Y}})^{\mathcal{X}_1}$ by $f_x^*\left(\frac{\mathsf{q}[y|x]}{\widehat{\mathcal{D}}^1(x)}\right) = \log\left(\sum_{y\in\mathcal{Y}} \mathsf{p}_0[y|x] e^{\frac{\mathsf{q}[y|x]}{\widehat{\mathcal{D}}^1(x)}}\right)$. Thus, f^* is given for all $\bar{\mathsf{q}} \in (\mathbb{R}^{\mathcal{Y}})^{\mathcal{X}_1}$ by

$$f^*(\mathsf{q}) = \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}^1}\left[\log\Big(\sum_{y\in\mathcal{Y}} \mathsf{p}_0[y|x] e^{\frac{\mathsf{q}[y|x]}{\widehat{\mathcal{D}}^1(x)}}\Big)\right].$$

As in the proof of Theorem 12.2, the conjugate function of $g = I_{\mathcal{C}}$ is given for all $\mathbf{w} \in \mathbb{R}^N$ by $g^*(\mathbf{w}) = \mathbb{E}_{(x,y)\sim\widehat{\mathcal{D}}}[\mathbf{w} \cdot \mathbf{\Phi}(x, y)] + \lambda\|\mathbf{w}\|_1$. In view of these identities, we can write, for all $\mathbf{w} \in \mathbb{R}^N$,

$$\begin{aligned}
&- f^*(A^*\mathbf{w}) - g^*(-\mathbf{w})\\
&= - \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}^1}\left[\log\Big(\sum_{y\in\mathcal{Y}} \mathsf{p}_0[y|x]e^{\mathbf{w}\cdot\mathbf{\Phi}(x,y)}\Big)\right] + \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}}[\mathbf{w}\cdot\mathbf{\Phi}(x,y)] - \lambda\|\mathbf{w}\|_1\\
&= - \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}^1}[\log Z(\mathbf{w}, x)] + \frac{1}{m}\sum_{i=1}^m \mathbf{w}\cdot\mathbf{\Phi}(x_i, y_i) - \lambda\|\mathbf{w}\|_1\\
&= \frac{1}{m}\sum_{i=1}^m \log\frac{e^{\mathbf{w}\cdot\mathbf{\Phi}(x_i,y_i)}}{Z(\mathbf{w}, x_i)} - \lambda\|\mathbf{w}\|_1\\
&= \frac{1}{m}\sum_{i=1}^m \log\left[\frac{\mathsf{p}_{\mathbf{w}}[y_i|x_i]}{\mathsf{p}_0[y_i|x_i]}\right] - \lambda\|\mathbf{w}\|_1 = G(\mathbf{w}),
\end{aligned}$$

which proves that $\sup_{\mathbf{w}\in\mathbb{R}^N} G(\mathbf{w}) = \min_{\bar{\mathsf{p}}\in(\mathbb{R}^{\mathcal{Y}})^{\mathcal{X}_1}} F(\bar{\mathsf{p}})$.

The second part of the proof is similar to that of Theorem 12.2. For any $\mathbf{w} \in \mathbb{R}^N$, we can write

$$\begin{aligned}
&G(\mathbf{w}) - \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}^1}[\mathsf{D}(\mathsf{p}^*[\cdot|x] \,\|\, \mathsf{p}_0[\cdot|x])] + \mathop{\mathbb{E}}_{x\sim\widehat{\mathcal{D}}^1}[\mathsf{D}(\mathsf{p}^*[\cdot|x] \,\|\, \mathsf{p}_{\mathbf{w}}[\cdot|x])]\\
&= \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}}\left[\log\frac{\mathsf{p}_{\mathbf{w}}[y|x]}{\mathsf{p}_0[y|x]}\right] - \lambda\|\mathbf{w}\|_1 - \mathop{\mathbb{E}}_{\substack{x\sim\widehat{\mathcal{D}}^1\\ y\sim\mathsf{p}^*[\cdot|x]}}\left[\log\frac{\mathsf{p}^*[y|x]}{\mathsf{p}_0[y|x]}\right] + \mathop{\mathbb{E}}_{\substack{x\sim\widehat{\mathcal{D}}^1\\ y\sim\mathsf{p}^*[\cdot|x]}}\left[\log\frac{\mathsf{p}^*[y|x]}{\mathsf{p}_{\mathbf{w}}[y|x]}\right]\\
&= -\lambda\|\mathbf{w}\|_1 + \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}}\left[\log\frac{\mathsf{p}_{\mathbf{w}}[y|x]}{\mathsf{p}_0[y|x]}\right] - \mathop{\mathbb{E}}_{\substack{x\sim\widehat{\mathcal{D}}^1\\ y\sim\mathsf{p}^*[\cdot|x]}}\left[\log\frac{\mathsf{p}_{\mathbf{w}}[y|x]}{\mathsf{p}_0[y|x]}\right]\\
&= -\lambda\|\mathbf{w}\|_1 + \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}}[\mathbf{w}\cdot\mathbf{\Phi}(x,y) - \log Z(\mathbf{w}, x)] - \mathop{\mathbb{E}}_{\substack{x\sim\widehat{\mathcal{D}}^1\\ y\sim\mathsf{p}^*[\cdot|x]}}[\mathbf{w}\cdot\mathbf{\Phi}(x,y) - \log Z(\mathbf{w}, x)]\\
&= -\lambda\|\mathbf{w}\|_1 + \mathbf{w}\cdot\Big[\mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x,y)] - \mathop{\mathbb{E}}_{\substack{x\sim\widehat{\mathcal{D}}^1\\ y\sim\mathsf{p}^*[\cdot|x]}}[\mathbf{\Phi}(x,y)]\Big].
\end{aligned}$$

As the solution of the primal optimization, $\bar{\mathsf{p}}^*$ verifies $I_{\mathcal{C}}\bigg(\mathop{\mathbb{E}}_{\substack{x\sim\widehat{\mathcal{D}}^1\\ y\sim\mathsf{p}^*[\cdot|x]}}[\mathbf{\Phi}(x,y)]\bigg) = 0$, that is $\bigg\|\mathop{\mathbb{E}}_{\substack{x\sim\widehat{\mathcal{D}}^1\\ y\sim\mathsf{p}^*[\cdot|x]}}[\mathbf{\Phi}(x,y)] - \mathbb{E}_{(x,y)\sim\widehat{\mathcal{D}}}[\mathbf{\Phi}(x,y)]\bigg\|_\infty \leq \lambda$. By Hölder's inequality, this implies the following inequality:

$$-\|\mathbf{w}\|_1 + \mathbf{w}\cdot\left[\mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}}[\mathbf{w}\cdot\mathbf{\Phi}(x,y)] - \mathop{\mathbb{E}}_{\substack{x\sim\widehat{\mathcal{D}}^1\\ y\sim\mathsf{p}^*[\cdot|x]}}[\mathbf{w}\cdot\mathbf{\Phi}(x,y)]\right] \leq -\|\mathbf{w}\|_1 + \|\mathbf{w}\|_1 = 0.$$

Thus, we can write, for any $\mathbf{w} \in \mathbb{R}^N$,

$$\mathop{\mathbb{E}}_{x \sim \widehat{\mathcal{D}}^1}\Big[\mathsf{D}(\mathsf{p}^*[\cdot|x] \parallel \mathsf{p}_{\mathbf{w}}[\cdot|x])\Big] \leq \mathop{\mathbb{E}}_{x \sim \widehat{\mathcal{D}}^1}\Big[\mathsf{D}(\mathsf{p}^*[\cdot|x] \parallel \mathsf{p}_0[\cdot|x])\Big] - G(\mathbf{w}).$$

Now, assume that $\mathbf{w}$ verifies $|G(\mathbf{w}) - \sup_{\mathbf{w} \in \mathbb{R}^N} G(\mathbf{w})| \leq \epsilon$ for some $\epsilon > 0$. Then, $\mathbb{E}_{x \sim \widehat{\mathcal{D}}^1}[\mathsf{D}(\mathsf{p}^*[\cdot|x] \parallel \mathsf{p}_0[\cdot|x])] - G(\mathbf{w}) = (\sup_{\mathbf{w}} G(\mathbf{w})) - G(\mathbf{w}) \leq \epsilon$ implies the inequality $\mathbb{E}_{x \sim \widehat{\mathcal{D}}^1}[\mathsf{D}(\mathsf{p}^*[\cdot|x] \parallel \mathsf{p}_{\mathbf{w}}[\cdot|x])] \leq \epsilon$. This concludes the proof of the theorem. □

13.10 Chapter notes

The logistic regression model is a classical model in statistics. The term *logistic* was introduced by the Belgian mathematician Verhulst [1838, 1845]. An early reference for logistic regression is the publication of Berkson [1944] who advocated the use of the logistic function, instead of the cumulative distribution function of the standard normal distribution (*probit model*).

Conditional maximum entropy models in natural language processing were introduced by Berger et al. [1996] and were widely adopted for a variety of different tasks, including part-of-speech tagging, parsing, machine translation, and text categorization (see tutorial by Manning and Klein [2003]). Our presentation of the conditional Maxent principle, including their regularized variants, the duality theorem for conditional Maxent models (Theorem 13.1) and their theoretical justifications are based on [Cortes, Kuznetsov, Mohri, and Syed, 2015]. This chapter provided two types of justification for these models: one based on the conditional Maxent principle, another based on standard generalization bounds.

As in the case of Maxent models for density estimation, conditional Maxent models can be extended by using other Bregman divergences [Lafferty, Pietra, and Pietra, 1997] and other regularizations. Lafferty [1999] presented a general framework for incremental algorithms based on Bregman divergences that admits logistic regression as as special case, see also [Collins et al., 2002] who showed that boosting and logistic regression were special instances of a common framework based on Bregman divergences. The regularized conditional Maxent models presented in this chapter can be extended similarly using other Bregman divergences. In the binary classification case, when coordinate descent is used to solve the optimization problem of regularized conditional Maxent models, the algorithm coincides with L_1-regularized AdaBoost modulo the use of the logistic loss instead of the exponential loss.

Cortes, Kuznetsov, Mohri, and Syed [2015] presented a more general family of conditional probability models, *conditional structural Maxent models*, for which they also presented a duality theorem and gave strong learning guarantees. These Maxent models are based on feature functions selected from a union of possibly very complex sub-families. The resulting algorithms coincide with the DeepBoost algorithms of Cortes, Mohri, and Syed [2014] in the binary classification case or the multi-class DeepBoost algorithm of Kuznetsov, Mohri, and Syed [2014] in the multi-class classification case, when the logistic function is used as a convex surrogate loss function.

13.11 Exercises

13.1 Extension to Bregman divergences.

(a) Show how conditional Maxent models can be extended by using arbitrary Bregman divergences instead of the (unnormalized) relative entropy.

(b) Prove a duality theorem similar to Theorem 13.1 for theses extensions.

(c) Derive theoretical guarantees for these extensions. What additional property is needed for the Bregman divergence so that your learning guarantees hold?

13.2 Stability analysis for L_2-regularized conditional Maxent.

(a) Give an upper bound on the stability of the L_2-regularized conditional Maxent in terms of the sample size and λ (*Hint*: use the techniques and results of Chapter 14).

(b) Use the previous question to derive a stability-based generalization guarantee for the algorithm.

13.3 Maximum conditional Maxent. An alternative measure of closeness, instead of the conditional relative entropy, is the maximum relative entropy over all $x \in \mathcal{X}_1$.

(a) Write the primal optimization problem for this maximum conditional Maxent formulation. Show that it is a convex optimization problem, and discuss its feasibility and the uniqueness of its solution.

(b) Prove a duality theorem for maximum conditional Maxent and write the equivalent dual problem.

(c) Analyze the properties of maximum conditional Maxent and give a generalization bound for the algorithm.

13.4 Conditional Maxent with other marginal distributions: discuss and analyze conditional Maxent models when using a distribution $\mathcal{Q}$ over $\mathcal{X}$ instead of $\widehat{\mathcal{D}}^1$. Prove that a duality theorem similar to Theorem 13.1 holds.

14 Algorithmic Stability

In chapters 2–5 and several subsequent chapters, we presented a variety of generalization bounds based on different measures of the complexity of the hypothesis set $\mathcal{H}$ used for learning, including the Rademacher complexity, the growth function, and the VC-dimension. These bounds ignore the specific algorithm used, that is, they hold for any algorithm using $\mathcal{H}$ as a hypothesis set.

One may ask if an analysis of the properties of a specific algorithm could lead to finer guarantees. Such an algorithm-dependent analysis could have the benefit of a more informative guarantee. On the other hand, it could be inapplicable to other algorithms using the same hypothesis set. Alternatively, as we shall see in this chapter, a more general property of the learning algorithm could be used to incorporate algorithm-specific properties while extending the applicability of the analysis to other learning algorithms with similar properties.

This chapter uses the property of *algorithmic stability* to derive *algorithm-dependent* learning guarantees. We first present a generalization bound for any algorithm that is sufficiently stable. Then, we show that the wide class of kernel-based regularization algorithms enjoys this property and derive a general upper bound on their stability coefficient. Finally, we illustrate the application of these results to the analysis of several algorithms both in the regression and classification settings, including kernel ridge regression (KRR), SVR, and SVMs.

14.1 Definitions

We start by introducing the notation and definitions relevant to our analysis of algorithmic stability. We denote by z a labeled example $(x, y) \in \mathcal{X} \times \mathcal{Y}$. The hypotheses h we consider map $\mathcal{X}$ to a set $\mathcal{Y}'$ sometimes different from $\mathcal{Y}$. In particular, for classification, we may have $\mathcal{Y} = \{-1, +1\}$ while the hypothesis h learned takes values in $\mathbb{R}$. The loss functions L we consider are therefore defined over $\mathcal{Y}' \times \mathcal{Y}$, with $\mathcal{Y}' = \mathcal{Y}$ in most cases. For a loss function $L \colon \mathcal{Y}' \times \mathcal{Y} \to \mathbb{R}_+$, we denote the loss of a hypothesis h at point z by $L_z(h) = L(h(x), y)$. We denote by $\mathcal{D}$ the distribution according to which samples are drawn and by $\mathcal{H}$ the hypothesis set. The empirical error or loss of $h \in \mathcal{H}$ on a sample $S = (z_1, \ldots, z_m)$ and its generalization error are defined, respectively, by

$$\widehat{R}_S(h) = \frac{1}{m} \sum_{i=1}^{m} L_{z_i}(h) \quad \text{and} \quad R(h) = \mathop{\mathbb{E}}_{z \sim \mathcal{D}}[L_z(h)].$$

Given an algorithm $\mathcal{A}$, we denote by h_S the hypothesis $h_S \in \mathcal{H}$ returned by $\mathcal{A}$ when trained on sample S. We will say that the loss function L is bounded by $M \geq 0$ if for all $h \in \mathcal{H}$ and $z \in \mathcal{X} \times \mathcal{Y}$, $L_z(h) \leq M$. For the results presented in this chapter, a weaker condition suffices, namely that $L_z(h_S) \leq M$ for all hypotheses h_S returned by the algorithm $\mathcal{A}$.

We are now able to define the notion of uniform stability, the algorithmic property used in the analyses of this chapter.

Definition 14.1 (Uniform stability) *Let S and S' be any two training samples that differ by a single point. Then, a learning algorithm $\mathcal{A}$ is* uniformly β-stable *if the hypotheses it returns when trained on any such samples S and S' satisfy*

$$\forall z \in \mathcal{Z}, \quad |L_z(h_S) - L_z(h_{S'})| \leq \beta.$$

The smallest such β satisfying this inequality is called the stability coefficient *of $\mathcal{A}$.*

In other words, when $\mathcal{A}$ is trained on two similar training sets, the losses incurred by the corresponding hypotheses returned by $\mathcal{A}$ should not differ by more than β. Note that a uniformly β-stable algorithm is often referred to as being *β-stable* or even just *stable* (for some unspecified β). In general, the coefficient β depends on the sample size m. We will see in section 14.2 that $\beta = o(1/\sqrt{m})$ is necessary for the convergence of the stability-based learning bounds presented in this chapter. In section 14.3, we will show that a more favorable condition holds, that is, $\beta = O(1/m)$, for a wide family of algorithms.

14.2 Stability-based generalization guarantee

In this section, we show that exponential bounds can be derived for the generalization error of stable learning algorithms. The main result is presented in theorem 14.2.

Theorem 14.2 *Assume that the loss function L is bounded by $M \geq 0$. Let $\mathcal{A}$ be a β-stable learning algorithm and let S be a sample of m points drawn i.i.d. according* *to distribution $\mathcal{D}$. Then, with probability at least $1-\delta$ over the sample S drawn, the following holds:*

$$R(h_S) \le \widehat{R}_S(h_S) + \beta + (2m\beta + M)\sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$

Proof: The proof is based on the application of McDiarmid's inequality (theorem D.8) to the function Φ defined for all samples S by $\Phi(S) = R(h_S) - \widehat{R}_S(h_S)$. Let S' be another sample of size m with points drawn i.i.d. according to $\mathcal{D}$ that differs from S by exactly one point. We denote that point by z_m in S, z'_m in S', i.e.,

$$S = (z_1, \ldots, z_{m-1}, z_m) \quad \text{and} \quad S' = (z_1, \ldots, z_{m-1}, z'_m).$$

By definition of Φ, the following inequality holds:

$$|\Phi(S') - \Phi(S)| \le |R(h_{S'}) - R(h_S)| + |\widehat{R}_{S'}(h_{S'}) - \widehat{R}_S(h_S)|. \tag{14.1}$$

We bound each of these two terms separately. By the β-stability of $\mathcal{A}$, we have

$$|R(h_S) - R(h_{S'})| = |\mathop{\mathbb{E}}_z[L_z(h_S)] - \mathop{\mathbb{E}}_z[L_z(h_{S'})]| \le \mathop{\mathbb{E}}_z[|L_z(h_S) - L_z(h_{S'})|] \le \beta.$$

Using the boundedness of L along with β-stability of $\mathcal{A}$, we also have

$$\begin{aligned}
|\widehat{R}_S(h_S) - \widehat{R}_{S'}(h_{S'})| &= \frac{1}{m}\left|\left(\sum_{i=1}^{m-1} L_{z_i}(h_S) - L_{z_i}(h_{S'})\right) + L_{z_m}(h_S) - L_{z'_m}(h_{S'})\right| \\
&\le \frac{1}{m}\left[\left(\sum_{i=1}^{m-1} |L_{z_i}(h_S) - L_{z_i}(h_{S'})|\right) + |L_{z_m}(h_S) - L_{z'_m}(h_{S'})|\right] \\
&\le \frac{m-1}{m}\beta + \frac{M}{m} \le \beta + \frac{M}{m}.
\end{aligned}$$

Thus, in view of (14.1), Φ satisfies the condition $|\Phi(S) - \Phi(S')| \le 2\beta + \frac{M}{m}$. By applying McDiarmid's inequality to $\Phi(S)$, we can bound the deviation of Φ from its mean as

$$\mathbb{P}\left[\Phi(S) \ge \epsilon + \mathop{\mathbb{E}}_S[\Phi(S)]\right] \le \exp\left(\frac{-2m\epsilon^2}{(2m\beta + M)^2}\right),$$

or, equivalently, with probability $1-\delta$,

$$\Phi(S) < \epsilon + \mathop{\mathbb{E}}_S[\Phi(S)], \tag{14.2}$$

where $\delta = \exp\left(\frac{-2m\epsilon^2}{(2m\beta+M)^2}\right)$. If we solve for ϵ in this expression for δ, plug into (14.2) and rearrange terms, then, with probability $1-\delta$, we have

$$\Phi(S) \le \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[\Phi(S)] + (2m\beta + M)\sqrt{\frac{\log \frac{1}{\delta}}{2m}}. \tag{14.3}$$

We now bound the expectation term, first noting that by linearity of expectation $\mathbb{E}_S[\Phi(S)] = \mathbb{E}_S[R(h_S)] - \mathbb{E}_S[\widehat{R}_S(h_S)]$. By definition of the generalization error,

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[R(h_S)] = \mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}\Big[\mathop{\mathbb{E}}_{z \sim \mathcal{D}}[L_z(h_S)]\Big] = \mathop{\mathbb{E}}_{S, z \sim \mathcal{D}^{m+1}}[L_z(h_S)]. \tag{14.4}$$

By the linearity of expectation,

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[\widehat{R}_S(h_S)] = \frac{1}{m}\sum_{i=1}^{m} \mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[L_{z_i}(h_S)] = \mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[L_{z_1}(h_S)], \tag{14.5}$$

where the second equality follows from the fact that the z_i are drawn i.i.d. and thus the expectations $\mathbb{E}_{S \sim \mathcal{D}^m}[L_{z_i}(h_S)]$, $i \in [m]$, are all equal. The last expression in (14.5) is the expected loss of a hypothesis on one of its training points. We can rewrite it as $\mathbb{E}_{S \sim D^m}[L_{z_1}(h_S)] = \mathbb{E}_{S,z \sim \mathcal{D}^{m+1}}[L_z(h_{S'})]$, where S' is a sample of m points containing z extracted from the $m+1$ points formed by S and z. Thus, in view of (14.4) and by the β-stability of $\mathcal{A}$, it follows that

$$\begin{aligned}
\big|\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[\Phi(S)]\big| &= \Big|\mathop{\mathbb{E}}_{S,z \sim \mathcal{D}^{m+1}}[L_z(h_S)] - \mathop{\mathbb{E}}_{S,z \sim \mathcal{D}^{m+1}}[L_z(h_{S'})]\Big| \\
&\leq \mathop{\mathbb{E}}_{S,z \sim \mathcal{D}^{m+1}}\Big[|L_z(h_S) - L_z(h_{S'})|\Big] \\
&\leq \mathop{\mathbb{E}}_{S,z \sim \mathcal{D}^{m+1}}[\beta] = \beta.
\end{aligned}$$

We can thus replace $\mathbb{E}_S[\Phi(S)]$ by β in (14.3), which completes the proof. □

The bound of the theorem converges for $(m\beta)/\sqrt{m} = o(1)$, that is $\beta = o(1/\sqrt{m})$. In particular, when the stability coefficient β is in $O(1/m)$, the theorem guarantees that $R(h_S) - \widehat{R}_S(h_S) = O(1/\sqrt{m})$ with high probability. In the next section, we show that kernel-based regularization algorithms precisely admit this property under some general assumptions.

14.3 Stability of kernel-based regularization algorithms

Let K be a positive definite symmetric kernel, $\mathbb{H}$ the reproducing kernel Hilbert space associated to K, and $\|\cdot\|_K$ the norm induced by K in $\mathbb{H}$. A kernel-based regularization algorithm is defined by the minimization over $\mathbb{H}$ of an objective function F_S based on a training sample $S = (z_1, \ldots, z_m)$ and defined for all $h \in \mathbb{H}$ by:

$$F_S(h) = \widehat{R}_S(h) + \lambda\|h\|_K^2. \tag{14.6}$$

In this equation, $\widehat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m} L_{z_i}(h)$ is the empirical error of hypothesis h with respect to a loss function L and $\lambda \geq 0$ a trade-off parameter balancing the emphasis on the empirical error versus the regularization term $\|h\|_K^2$. The hypothesis set $\mathcal{H}$

is the subset of $\mathbb{H}$ formed by the hypotheses possibly returned by the algorithm. Algorithms such as KRR, SVR and SVMs all fall under this general model.

We first introduce some definitions and tools needed for a general proof of an upper bound on the stability coefficient of kernel-based regularization algorithms. Our analysis will assume that the loss function L is convex and that it further verifies the following Lipschitz-like smoothness condition.

Definition 14.3 (σ-admissibility) *A loss function L is σ-*admissible *with respect to the hypothesis class $\mathcal{H}$ if there exists $\sigma \in \mathbb{R}_+$ such that for any two hypotheses $h, h' \in \mathcal{H}$ and for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$,*

$$|L(h'(x), y) - L(h(x), y)| \leq \sigma |h'(x) - h(x)|. \tag{14.7}$$

This assumption holds for the quadratic loss and most other loss functions where the hypothesis set and the set of output labels are bounded by some $M \in \mathbb{R}_+$: $\forall h \in \mathcal{H}, \forall x \in \mathcal{X}, |h(x)| \leq M$ and $\forall y \in \mathcal{Y}, |y| \leq M$.

We will use the notion of *Bregman divergence*, B_F which can be defined for any convex and differentiable function $F \colon \mathbb{H} \to \mathbb{R}$ as follows: for all $f, g \in \mathbb{H}$,

$$\mathsf{B}_F(f \| g) = F(f) - F(g) - \langle f - g, \nabla F(g) \rangle .$$

Section E.4 presents the properties of the Bregman divergence in more detail and also contains figure E.2 which illustrates the geometric interpretation of the Bregman divergence. We generalize the definition of Bregman divergence to cover the case of convex but non-differentiable loss functions F by using the notion of *subgradient*. For a convex function $F \colon \mathbb{H} \to \mathbb{R}$, we denote by $\partial F(h)$ the *subdifferential* of F at h, which is defined as follows:

$$\partial F(h) = \{g \in \mathbb{H} \colon \forall h' \in \mathbb{H}, F(h') - F(h) \geq \langle h' - h, g \rangle\}.$$

Thus, $\partial F(h)$ is the set of vectors g defining a hyperplane supporting function F at point h (see figure 14.1). Elements of the subdifferential are called subgradients (see section B.4.1 for more discussion). Note, the subgradient found in $\partial F(h)$ coincides with $\nabla F(h)$ when F is differentiable at h, i.e. $\partial F(h) = \{\nabla F(h)\}$. Furthermore, at a point h where F is minimal, 0 is an element of $\partial F(h)$. The subgradient is additive, that is, for two convex function F_1 and F_2, $\partial(F_1 + F_2)(h) = \{g_1 + g_2 \colon g_1 \in \partial F_1(h), g_2 \in \partial F_2(h)\}$. For any $h \in \mathbb{H}$, we fix $\delta F(h)$ to be an (arbitrary) element of $\partial F(h)$. For any such choice of δF, we can define the *generalized Bregman divergence* associated to F by:

$$\forall h', h \in \mathbb{H}, \mathsf{B}_F(h' \parallel h) = F(h') - F(h) - \langle h' - h, \delta F(h) \rangle . \tag{14.8}$$

Note that by definition of the subgradient, $\mathsf{B}_F(h' \parallel h) \geq 0$ for all $h', h \in \mathbb{H}$.

Starting from (14.6), we can now define the generalized Bregman divergence of F_S. Let N denote the convex function $h \to \|h\|_K^2$. Since N is differentiable,

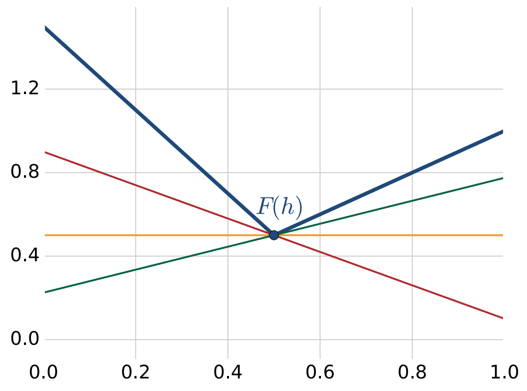


Figure 14.1
Illustration of the notion of subgradient: supporting hyperplanes (shown in red, orange and green) for the function F (shown in blue) at point h are defined by elements of the subdifferential $\partial F(h)$.

$\delta N(h) = \nabla N(h)$ for all $h \in \mathbb{H}$, and thus δN (as well as B_N) is uniquely defined. To make the definition of the Bregman divergences for F_S and $\widehat{R}_S$ compatible so that $\mathsf{B}_{F_S} = \mathsf{B}_{\widehat{R}_S} + \lambda \mathsf{B}_N$, we define $\delta \widehat{R}_S$ in terms of δF_S by: $\delta \widehat{R}_S(h) = \delta F_S(h) - \lambda \nabla N(h)$ for all $h \in \mathbb{H}$. Furthermore, we choose $\delta F_S(h)$ to be 0 for any point h where F_S is minimal and let $\delta F_S(h)$ be an arbitrary element of $\partial F_S(h)$ for all other $h \in \mathbb{H}$. We proceed in a similar way to define the Bregman divergences for $F_{S'}$ and $\widehat{R}_{S'}$ so that $\mathsf{B}_{F_{S'}} = \mathsf{B}_{\widehat{R}_{S'}} + \lambda \mathsf{B}_N$.

We will use the notion of generalized Bregman divergence for the proof of the following general upper bound on the stability coefficient of kernel-based regularization algorithms.

Proposition 14.4 *Let K be a positive definite symmetric kernel such that for all $x \in \mathcal{X}$, $K(x, x) \leq r^2$ for some $r \in \mathbb{R}_+$ and let L be a convex and σ-admissible loss function. Then, the kernel-based regularization algorithm defined by the minimization* (14.6) *is β-stable with the following upper bound on β:*

$$\beta \leq \frac{\sigma^2 r^2}{m\lambda}.$$

Proof: Let h be a minimizer of F_S and h' a minimizer of $F_{S'}$, where samples S and S' differ exactly by one point, z_m in S and z'_m in S'. Since the generalized Bregman divergence is non-negative and since $\mathsf{B}_{F_S} = \mathsf{B}_{\widehat{R}_S} + \lambda \mathsf{B}_N$ and $\mathsf{B}_{F_{S'}} = \mathsf{B}_{\widehat{R}_{S'}} + \lambda \mathsf{B}_N$, we can write

$$\mathsf{B}_{F_S}(h' \| h) + \mathsf{B}_{F_{S'}}(h \| h') \geq \lambda\big(\mathsf{B}_N(h' \| h) + \mathsf{B}_N(h \| h')\big).$$

Observe that $\mathsf{B}_N(h'\|h) + \mathsf{B}_N(h\|h') = -\langle h' - h, 2h\rangle - \langle h - h', 2h'\rangle = 2\|h' - h\|_K^2$. Let Δh denote $h' - h$, then we can write

$$\begin{aligned}
&2\lambda||\Delta h\|_K^2\\
&\quad\leq \mathsf{B}_{F_S}(h' \,\|\, h) + \mathsf{B}_{F_{S'}}(h \,\|\, h')\\
&\quad= F_S(h') - F_S(h) - \langle h' - h, \delta F_S(h)\rangle + F_{S'}(h) - F_{S'}(h') - \langle h - h', \delta F_{S'}(h')\rangle\\
&\quad= F_S(h') - F_S(h) + F_{S'}(h) - F_{S'}(h')\\
&\quad= \widehat{R}_S(h') - \widehat{R}_S(h) + \widehat{R}_{S'}(h) - \widehat{R}_{S'}(h').
\end{aligned}$$

The second equality follows from the definition of h' and h as minimizers and our choice of the subgradients for minimal points which together imply $\delta F_{S'}(h') = 0$ and $\delta F_S(h) = 0$. The last equality follows from the definitions of F_S and $F_{S'}$. Next, we express the resulting inequality in terms of the loss function L and use the fact that S and S' differ by only one point along with the σ-admissibility of L to get

$$\begin{aligned}
2\lambda\|\Delta h\|_K^2 &\leq \frac{1}{m}[L_{z_m}(h') - L_{z_m}(h) + L_{z'_m}(h) - L_{z'_m}(h')]\\
&\leq \frac{\sigma}{m}[|\Delta h(x_m)| + |\Delta h(x'_m)|]. \qquad (14.9)
\end{aligned}$$

By the reproducing kernel property and the Cauchy-Schwarz inequality, for all $x \in \mathcal{X}$,

$$\Delta h(x) = \langle \Delta h, K(x,\cdot)\rangle \leq \|\Delta h\|_K \|K(x,\cdot)\|_K = \sqrt{K(x,x)}\|\Delta h\|_K \leq r\|\Delta h\|_K.$$

In view of (14.9), this implies $\|\Delta h\|_K \leq \frac{\sigma r}{\lambda m}$. By the σ-admissibility of L and the reproducing property, the following holds:

$$\forall z \in \mathcal{X} \times \mathcal{Y}, |L_z(h') - L_z(h)| \leq \sigma|\Delta h(x)| \leq r\sigma\|\Delta h\|_K,$$

which gives

$$\forall z \in \mathcal{X} \times \mathcal{Y}, |L_z(h') - L_z(h)| \leq \frac{\sigma^2 r^2}{m\lambda},$$

and concludes the proof. □

Thus, under the assumptions of the proposition, for a fixed λ, the stability coefficient of kernel-based regularization algorithms is in $O(1/m)$.

14.3.1 Application to regression algorithms: SVR and KRR

Here, we analyze more specifically two widely used regression algorithms, Support Vector Regression (SVR) and Kernel Ridge Regression (KRR), which are both special instances of the family of kernel-based regularization algorithms.

SVR is based on the ϵ-insensitive loss L_ϵ defined for all $(y, y') \in \mathcal{Y} \times \mathcal{Y}$ by:

$$L_\epsilon(y', y) = \begin{cases} 0 & \text{if } |y' - y| \leq \epsilon; \\ |y' - y| - \epsilon & \text{otherwise.} \end{cases} \tag{14.10}$$

We now present a stability-based bound for SVR assuming that L_ϵ is bounded for the hypotheses returned by SVR (which, as we shall later see in lemma 14.7, is indeed the case when the label set $\mathcal{Y}$ is bounded).

Corollary 14.5 (Stability-based learning bound for SVR) *Assume that $K(x, x) \leq r^2$ for all $x \in \mathcal{X}$ for some $r \geq 0$ and that L_ϵ is bounded by $M \geq 0$. Let h_S denote the hypothesis returned by SVR when trained on an i.i.d. sample S of size m. Then, for any $\delta > 0$, the following inequality holds with probability at least $1 - \delta$:*

$$R(h_S) \leq \widehat{R}_S(h_S) + \frac{r^2}{m\lambda} + \Big(\frac{2r^2}{\lambda} + M\Big)\sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$

Proof: We first show that $L_\epsilon(\cdot) = L_\epsilon(\cdot, y)$ is 1-Lipschitz for any $y \in \mathcal{Y}$. For any $y', y'' \in \mathcal{Y}$, we must consider four cases. First, if $|y' - y| \leq \epsilon$ and $|y'' - y| \leq \epsilon$, then $|L_\epsilon(y'') - L_\epsilon(y')| = 0$. Second, if $|y' - y| > \epsilon$ and $|y'' - y| > \epsilon$, then $|L_\epsilon(y'') - L_\epsilon(y')| = ||y'' - y| - |y' - y|| \leq |y'' - y'|$, by the triangle inequality. Third, if $|y' - y| \leq \epsilon$ and $|y'' - y| > \epsilon$, then $|L_\epsilon(y'') - L_\epsilon(y')| = ||y'' - y| - \epsilon| = |y'' - y| - \epsilon \leq |y'' - y| - |y' - y| \leq |y'' - y'|$. Fourth, if $|y'' - y| \leq \epsilon$ and $|y' - y| > \epsilon$, by symmetry the same inequality is obtained as in the previous case.

Thus, in all cases, $|L_\epsilon(y'', y) - L_\epsilon(y', y)| \leq |y'' - y'|$. This implies in particular that L_ϵ is σ-admissible with $\sigma = 1$ for any hypothesis set $\mathcal{H}$. By proposition 14.4, under the assumptions made, SVR is β-stable with $\beta \leq \frac{r^2}{m\lambda}$. Plugging this expression into the bound of theorem 14.2 yields the result. ☐

We next present a stability-based bound for KRR, which is based on the square loss L_2 defined for all $y', y \in \mathcal{Y}$ by:

$$L_2(y', y) = (y' - y)^2. \tag{14.11}$$

As in the SVR setting, we assume in our analysis that L_2 is bounded for the hypotheses returned by KRR (which, as we shall later see again in lemma 14.7, is indeed the case when the label set $\mathcal{Y}$ is bounded).

Corollary 14.6 (Stability-based learning bound for KRR) *Assume that $K(x, x) \leq r^2$ for all $x \in \mathcal{X}$ for some $r \geq 0$ and that L_2 is bounded by $M \geq 0$. Let h_S denote the hypothesis returned by KRR when trained on an i.i.d. sample S of size m. Then, for any $\delta > 0$, the following inequality holds with probability at least $1 - \delta$:*

$$R(h_S) \leq \widehat{R}_S(h_S) + \frac{4Mr^2}{\lambda m} + \Big(\frac{8Mr^2}{\lambda} + M\Big)\sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$

Proof: For any $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and $h, h' \in \mathcal{H}$,

$$\begin{aligned}
|L_2(h'(x), y) - L_2(h(x), y)| &= \left|(h'(x) - y)^2 - (h(x) - y)^2\right| \\
&= \left|\big[h'(x) - h(x)\big][(h'(x) - y) + (h(x) - y)]\right| \\
&\leq (|h'(x) - y| + |h(x) - y|)|h(x) - h'(x)| \\
&\leq 2\sqrt{M}|h(x) - h'(x)|,
\end{aligned}$$

where we used the M-boundedness of the loss. Thus, L_2 is σ-admissible with $\sigma = 2\sqrt{M}$. Therefore, by proposition 14.4, KRR is β-stable with $\beta \leq \frac{4r^2M}{m\lambda}$. Plugging this expression into the bound of theorem 14.2 yields the result. □

The previous two corollaries assumed bounded loss functions. We now present a lemma that implies in particular that the loss functions used by SVR and KRR are bounded when the label set is bounded.

Lemma 14.7 *Assume that $K(x, x) \leq r^2$ for all $x \in \mathcal{X}$ for some $r \geq 0$ and that for all $y \in \mathcal{Y}$, $L(0, y) \leq B$ for some $B \geq 0$. Then, the hypothesis h_S returned by a kernel-based regularization algorithm trained on a sample S is bounded as follows:*

$$\forall x \in \mathcal{X}, |h_S(x)| \leq r\sqrt{B/\lambda}.$$

Proof: By the reproducing kernel property and the Cauchy-Schwarz inequality, we can write

$$\forall x \in \mathcal{X}, |h_S(x)| = \langle h_S, K(x, \cdot)\rangle \leq \|h_S\|_K \sqrt{K(x, x)} \leq r\|h_S\|_K. \tag{14.12}$$

The minimization (14.6) is over $\mathbb{H}$, which includes 0. Thus, by definition of F_S and h_S, the following inequality holds:

$$F_S(h_S) \leq F_S(0) = \frac{1}{m}\sum_{i=1}^{m} L(0, y_i) \leq B.$$

Since the loss L is non-negative, we have $\lambda\|h_S\|_K^2 \leq F_S(h_S)$ and thus $\lambda\|h_S\|_K^2 \leq B$. Combining this inequality with (14.12) yields the result. □

14.3.2 Application to classification algorithms: SVMs

This section presents a generalization bound for SVMs, when using the standard hinge loss defined for all $y \in \mathcal{Y} = \{-1, +1\}$ and $y' \in \mathbb{R}$ by

$$L_{\text{hinge}}(y', y) = \begin{cases} 0 & \text{if } 1 - yy' \leq 0; \\ 1 - yy' & \text{otherwise.} \end{cases} \tag{14.13}$$

Corollary 14.8 (Stability-based learning bound for SVMs) *Assume that $K(x, x) \leq r^2$ for all $x \in \mathcal{X}$ for some $r \geq 0$. Let h_S denote the hypothesis returned by SVMs when* *trained on an i.i.d. sample S of size m. Then, for any $\delta > 0$, the following inequality holds with probability at least $1 - \delta$:*

$$R(h_S) \leq \widehat{R}_S(h_S) + \frac{r^2}{m\lambda} + \Big(\frac{2r^2}{\lambda} + \frac{r}{\sqrt{\lambda}} + 1\Big)\sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$

Proof: It is straightforward to verify that $L_{\text{hinge}}(\cdot, y)$ is 1-Lipschitz for any $y \in \mathcal{Y}$ and therefore that it is σ-admissible with $\sigma = 1$. Therefore, by proposition 14.4, SVMs is β-stable with $\beta \leq \frac{r^2}{m\lambda}$. Since $|L_{\text{hinge}}(0, y)| \leq 1$ for any $y \in \mathcal{Y}$, by lemma 14.7, $\forall x \in \mathcal{X}, |h_S(x)| \leq r/\sqrt{\lambda}$. Thus, for any sample S and any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the loss is bounded as follows: $L_{\text{hinge}}(h_S(x), y) \leq r/\sqrt{\lambda} + 1$. Plugging this value of M and the one found for β into the bound of theorem 14.2 yields the result. □

Since the hinge loss upper bounds the binary loss, the bound of the corollary 14.8 also applies to the generalization error of h_S measured in terms of the standard binary loss used in classification.

14.3.3 Discussion

Note that the learning bounds presented for kernel-based regularization algorithms are of the form $R(h_S) - \widehat{R}_S(h_S) \leq O\big(\frac{1}{\lambda\sqrt{m}}\big)$. Thus, these bounds are informative only when $\lambda \gg 1/\sqrt{m}$. The regularization parameter λ is a function of the sample size m: for larger values of m, it is expected to be smaller, decreasing the emphasis on regularization. The magnitude of λ affects the norm of the linear hypotheses used for prediction, with a larger value of λ implying a smaller hypothesis norm. In this sense, λ is a measure of the complexity of the hypothesis set and the condition required for λ can be interpreted as stating that a less complex hypothesis set guarantees better generalization.

Note also that our analysis of stability in this chapter assumed a fixed λ: the regularization parameter is assumed to be invariant to the change of one point of the training sample. While this is a mild assumption, it may not hold in general.

14.4 Chapter notes

The notion of algorithmic stability was first used by Devroye, Rogers and Wagner [Rogers and Wagner, 1978, Devroye and Wagner, 1979a,b] for the k-nearest neighbor algorithm and other k-local rules. Kearns and Ron [1999] later gave a formal definition of stability and used it to provide an analysis of the leave-one-out error. Much of the material presented in this chapter is based on Bousquet and Elisseeff [2002]. Our proof of proposition 14.4 is novel and generalizes the results of Bousquet and Elisseeff [2002] to the case of non-differentiable convex losses. Moreover, stability-based generalization bounds have been extended to ranking algorithms [Agarwal and Niyogi, 2005, Cortes et al., 2007b], as well as to the non-i.i.d.

scenario of stationary Φ- and β-mixing processes [Mohri and Rostamizadeh, 2010], and to the transductive setting [Cortes et al., 2008a]. Additionally, exercise 14.5 is based on Cortes et al. [2010b], which introduces and analyzes stability with respect to the choice of the kernel function or kernel matrix.

Note that while, as shown in this chapter, uniform stability is sufficient for deriving generalization bounds, it is not a necessary condition. Some algorithms may generalize well in the supervised learning scenario but may not be uniformly stable, for example, the Lasso algorithm [Xu et al., 2008]. Shalev-Shwartz et al. [2009] have used the notion of stability to provide necessary and sufficient conditions for a technical condition of learnability related to PAC-learning, even in general scenarios where learning is possible only by using non-ERM rules.

14.5 Exercises

14.1 Tighter stability bounds

(a) Assuming the conditions of theorem 14.2 hold, can one hope to guarantee a generalization with slack better than $O(1/\sqrt{m})$ even if the algorithm is very stable, i.e. $\beta \to 0$?

(b) Can you show an $O(1/m)$ generalization guarantee if L is bounded by $C/\sqrt{m}$ (a very strong condition)? If so, how stable does the learning algorithm need to be?

14.2 Quadratic hinge loss stability. Let L denote the quadratic hinge loss function defined for all $y \in \{+1, -1\}$ and $y' \in \mathbb{R}$ by

$$L(y', y) = \begin{cases} 0 & \text{if } 1 - y'y \leq 0; \\ (1 - y'y)^2 & \text{otherwise.} \end{cases}$$

Assume that $L(h(x), y)$ is bounded by M, $1 \leq M < \infty$, for all $h \in \mathcal{H}$, $x \in \mathcal{X}$, and $y \in \{+1, -1\}$, which also implies a bound on $|h(x)|$ for all $h \in \mathcal{H}$ and $x \in \mathcal{X}$. Derive a stability-based generalization bound for SVMs with the quadratic hinge loss.

14.3 Stability of linear regression.

(a) How does the stability bound in corollary 14.6 for ridge regression (i.e. kernel ridge regression with a linear kernel) behave as $\lambda \to 0$?

(b) Can you show a stability bound for linear regression (i.e. ridge regression with $\lambda = 0$)? If not, show a counter-example.

14.4 Kernel stability. Suppose an approximation of the kernel matrix $\mathbf{K}$, denoted $\mathbf{K}'$, is used to train the hypothesis h' (and let h denote the non-approximate hypothesis). At test time, no approximation is made, so if we let $\mathbf{k}_x = \big[K(x, x_1), \ldots, K(x, x_m)\big]^\top$ we can write $h(x) = \boldsymbol{\alpha}^\top \mathbf{k}_x$ and $h'(x) = \boldsymbol{\alpha}'^\top \mathbf{k}_x$. Show that if $\forall x, x' \in \mathcal{X}, K(x, x') \leq r$ then

$$|h'(x) - h(x)| \leq \frac{rmM}{\lambda^2} \|\mathbf{K}' - \mathbf{K}\|_2 \,.$$

(*Hint*: Use exercise 10.3)

14.5 Stability of relative-entropy regularization.

(a) Consider an algorithm that selects a distribution g over a hypothesis class which is parameterized by $\theta \in \Theta$. Given a point $z = (x, y)$ the expected loss is defined as

$$H(g, z) = \int_\Theta L(h_\theta(x), y) g(\theta) \, d\theta \,,$$

with respect to a base loss function L. Assuming the loss function L is bounded by M, show that the expected loss H is M-admissible, i.e. show $|H(g, z) - H(g', z)| \leq M \int_\Theta |g(\theta) - g'(\theta)| \, d\theta$.

(b) Consider an algorithm that minimizes the *entropy regularized* objective over the choice of distribution g:

$$F_S(g) = \underbrace{\frac{1}{m} \sum_{i=1}^m H(g, z_i)}_{\widehat{R}_S(g)} + \lambda K(g, f_0) \,.$$

Here, K is the Kullback-Leibler divergence (or relative entropy) between two distributions,

$$K(g, f_0) = \int_\Theta g(\theta) \log \frac{g(\theta)}{f_0(\theta)} \, d\theta \,, \tag{14.14}$$

and f_0 is some fixed distribution. Show that such an algorithm is stable by performing the following steps:

i. First use the fact $\frac{1}{2}(\int_\Theta |g(\theta) - g'(\theta)| \, d\theta)^2 \leq K(g, g')$ (Pinsker's inequality), to show

$$\Big(\int_\Theta |g_S(\theta) - g_{S'}(\theta)| \, d\theta \Big)^2 \leq B_{K(.,f_0)}(g \| g') + B_{K(.,f_0)}(g' \| g) \,.$$

ii. Next, let g be the minimizer of F_S and g' the minimizer of $F_{S'}$, where S and S' differ only at the index m. Show that

$$\begin{aligned} &B_{K(.,f_0)}(g\|g') + B_{K(.,f_0)}(g'\|g) \\ &\qquad \le \frac{1}{m\lambda}\big|H(g', z_m) - H(g, z_m) + H(g, z'_m) - H(g', z'_m)\big| \\ &\qquad\qquad \le \frac{2M}{m\lambda}\int_\Theta |g(\theta) - g'(\theta)|\, d\theta\,. \end{aligned}$$

iii. Finally, combine the results above to show that the entropy regularized algorithm is $\frac{2M^2}{m\lambda}$-stable.

15 Dimensionality Reduction

In settings where the data has a large number of features, it is often desirable to reduce its dimension, or to find a lower-dimensional representation preserving some of its properties. The key arguments for dimensionality reduction (or manifold learning) techniques are:

- *Computational*: to compress the initial data as a preprocessing step to speed up subsequent operations on the data.
- *Visualization*: to visualize the data for exploratory analysis by mapping the input data into two- or three-dimensional spaces.
- *Feature extraction*: to hopefully generate a smaller and more effective or useful set of features.

The benefits of dimensionality reduction are often illustrated via simulated data, such as the Swiss roll dataset. In this example, the input data, depicted in figure 15.1a, is three-dimensional, but it lies on a two-dimensional manifold that is "unfolded" in two-dimensional space as shown in figure 15.1b. It is important to note, however, that exact low-dimensional manifolds are rarely encountered in practice. Hence, this idealized example is more useful to illustrate the concept of dimensionality reduction than to verify the effectiveness of dimensionality reduction algorithms.

Dimensionality reduction can be formalized as follows. Consider a sample $S = (x_1, \ldots, x_m)$, a feature mapping $\mathbf{\Phi}\colon \mathcal{X} \to \mathbb{R}^N$ and the data matrix $\mathbf{X} \in \mathbb{R}^{N \times m}$ defined as $(\mathbf{\Phi}(x_1), \ldots, \mathbf{\Phi}(x_m))$. The ith data point is represented by $\mathbf{x}_i = \mathbf{\Phi}(x_i)$, or the ith column of $\mathbf{X}$, which is an N-dimensional vector. Dimensionality reduction techniques broadly aim to find, for $k \ll N$, a k-dimensional representation of the data, $\mathbf{Y} \in \mathbb{R}^{k \times m}$, that is in some way faithful to the original representation $\mathbf{X}$.

In this chapter we will discuss various techniques that address this problem. We first present the most commonly used dimensionality reduction technique called *principal component analysis* (PCA). We then introduce a kernelized version of PCA (KPCA) and show the connection between KPCA and manifold learning

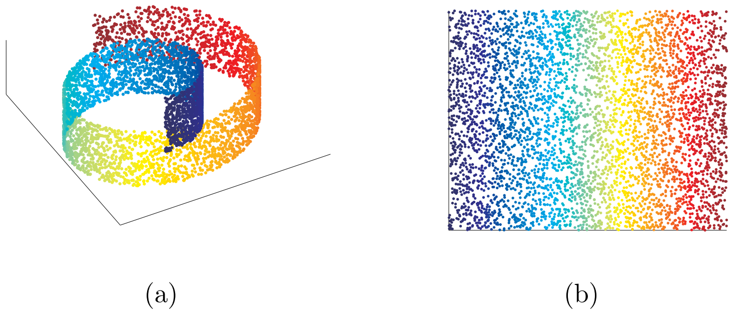


Figure 15.1
The "Swiss roll" dataset. (a) high-dimensional representation. (b) lower-dimensional representation.

algorithms. We conclude with a presentation of the Johnson-Lindenstrauss lemma, a classical theoretical result that has inspired a variety of dimensionality reduction methods based on the concept of random projections. The discussion in this chapter relies on basic matrix properties that are reviewed in appendix A.

15.1 Principal component analysis

Fix $k \in [N]$ and let $\mathbf{X}$ be a mean-centered data matrix, that is, $\sum_{i=1}^m \mathbf{x}_i = \mathbf{0}$. Define $\mathcal{P}_k$ as the set of N-dimensional rank-k orthogonal projection matrices. PCA consists of projecting the N-dimensional input data onto the k-dimensional linear subspace that minimizes *reconstruction error*, that is the sum of the squared L_2-distances between the original data and the projected data. Thus, the PCA algorithm is completely defined by the orthogonal projection matrix solution $\mathbf{P}^*$ of the following minimization problem:

$$\min_{\mathbf{P} \in \mathcal{P}_k} \quad \|\mathbf{P}\mathbf{X} - \mathbf{X}\|_F^2 \,. \tag{15.1}$$

The following theorem shows that PCA coincides with the projection of each data point onto the k top singular vectors of the sample covariance matrix, i.e., $\mathbf{C} = \frac{1}{m}\mathbf{X}\mathbf{X}^\top$ for the mean-centered data matrix $\mathbf{X}$. Figure 15.2 illustrates the basic intuition behind PCA, showing how two-dimensional data points with highly correlated features can be more succinctly represented with a one-dimensional representation that captures most of the variance in the data.

Theorem 15.1 *Let $\mathbf{P}^* \in \mathcal{P}_k$ be the PCA solution, i.e., the orthogonal projection matrix solution of* (15.1)*. Then, $\mathbf{P}^* = \mathbf{U}_k\mathbf{U}_k^\top$, where $\mathbf{U}_k \in \mathbb{R}^{N\times k}$ is the matrix* *formed by the top k singular vectors of $\mathbf{C} = \frac{1}{m}\mathbf{X}\mathbf{X}^\top$, the sample covariance matrix corresponding to $\mathbf{X}$. Moreover, the associated k-dimensional representation of $\mathbf{X}$ is given by $\mathbf{Y} = \mathbf{U}_k^\top \mathbf{X}$.*

Proof: Let $\mathbf{P} = \mathbf{P}^\top$ be an orthogonal projection matrix. By the definition of the Frobenius norm, the linearity of the trace operator and the fact that $\mathbf{P}$ is idempotent, i.e., $\mathbf{P}^2 = \mathbf{P}$, we observe that

$$\begin{aligned}\|\mathbf{PX} - \mathbf{X}\|_F^2 &= \mathrm{Tr}[(\mathbf{PX} - \mathbf{X})^\top(\mathbf{PX} - \mathbf{X})] = \mathrm{Tr}[\mathbf{X}^\top\mathbf{P}^2\mathbf{X} - 2\mathbf{X}^\top\mathbf{PX} + \mathbf{X}^\top\mathbf{X}]\\ &= -\,\mathrm{Tr}[\mathbf{X}^\top\mathbf{PX}] + \mathrm{Tr}[\mathbf{X}^\top\mathbf{X}]\,.\end{aligned}$$

Since $\mathrm{Tr}[\mathbf{X}^\top\mathbf{X}]$ is a constant with respect to $\mathbf{P}$, we have

$$\operatorname*{argmin}_{\mathbf{P}\in\mathcal{P}_k} \|\mathbf{PX} - \mathbf{X}\|_F^2 = \operatorname*{argmax}_{\mathbf{P}\in\mathcal{P}_k} \mathrm{Tr}[\mathbf{X}^\top\mathbf{PX}]\,. \tag{15.2}$$

By definition of orthogonal projections in $\mathcal{P}_k$, $\mathbf{P} = \mathbf{U}\mathbf{U}^\top$ for some $\mathbf{U} \in \mathbb{R}^{N\times k}$ containing orthogonal columns. Using the invariance of the trace operator under cyclic permutations and the orthogonality of the columns of $\mathbf{U}$, we have

$$\mathrm{Tr}[\mathbf{X}^\top\mathbf{PX}] = \mathrm{Tr}[\mathbf{U}^\top\mathbf{X}\mathbf{X}^\top\mathbf{U}] = \sum_{i=1}^{k} \mathbf{u}_i^\top\mathbf{X}\mathbf{X}^\top\mathbf{u}_i\,,$$

where $\mathbf{u}_i$ is the ith column of $\mathbf{U}$. By the Rayleigh quotient (section A.2.3), it is clear that the largest k singular vectors of $\mathbf{X}\mathbf{X}^\top$ maximize the rightmost sum above. Since $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{C}$ differ only by a scaling factor, they have the same singular vectors, and thus $\mathbf{U}_k$ maximizes this sum, which proves the first statement of the theorem. Finally, since $\mathbf{PX} = \mathbf{U}_k\mathbf{U}_k^\top\mathbf{X}$, $\mathbf{Y} = \mathbf{U}_k^\top\mathbf{X}$ is a k-dimensional representation of $\mathbf{X}$ with $\mathbf{U}_k$ as the basis vectors. □

By definition of the covariance matrix, the top singular vectors of $\mathbf{C}$ are the directions of maximal variance in the data, and the associated singular values are equal to these variances. Hence, PCA can also be viewed as projecting onto the subspace of maximal variance. Under this interpretation, the first principal component is derived from projection onto the direction of maximal variance, given by the top singular vector of $\mathbf{C}$. Similarly, the ith principal component, for $1 \le i \le k$, is derived from projection onto the ith direction of maximal variance, subject to orthogonality constraints to the previous $i-1$ directions of maximal variance (see exercise 15.1 for more details).

15.2 Kernel principal component analysis (KPCA)

In the previous section, we presented the PCA algorithm, which involved projecting onto the singular vectors of the sample covariance matrix $\mathbf{C}$. In this section, we

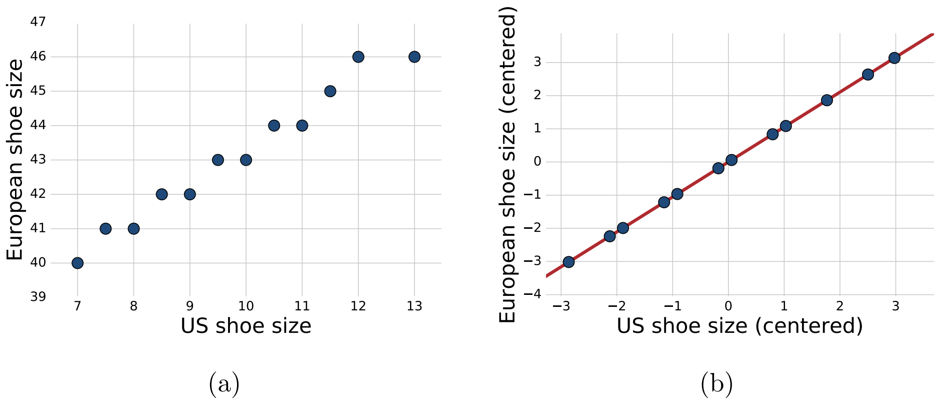


Figure 15.2
Example of PCA. (a) Two-dimensional data points with features capturing shoe size measured with different units. (b) One-dimensional representation that captures the most variance in the data, generated by projecting onto largest principal component (red line) of the mean-centered data points.

present a kernelized version of PCA, called KPCA. In the KPCA setting, $\mathbf{\Phi}$ is a feature mapping to an arbitrary RKHS (not necessarily to $\mathbb{R}^N$) and we work exclusively with a kernel function K corresponding to the inner product in this RKHS. The KPCA algorithm can thus be defined as a generalization of PCA in which the input data is projected onto the top principle components in this RKHS. We will show the relationship between PCA and KPCA by drawing upon the deep connections among the SVDs of $\mathbf{X}$, $\mathbf{C}$ and $\mathbf{K}$. We then illustrate how various manifold learning algorithms can be interpreted as special instances of KPCA.

Let K be a PDS kernel defined over $\mathcal{X} \times \mathcal{X}$ and define the kernel matrix as $\mathbf{K} = \mathbf{X}^\top \mathbf{X}$. Since $\mathbf{X}$ admits the following singular value decomposition: $\mathbf{X} = \mathbf{U\Sigma V}^\top$, $\mathbf{C}$ and $\mathbf{K}$ can be rewritten as follows:

$$\mathbf{C} = \frac{1}{m}\mathbf{U\Lambda U}^\top \qquad \mathbf{K} = \mathbf{V\Lambda V}^\top\,, \tag{15.3}$$

where $\mathbf{\Lambda} = \mathbf{\Sigma}^2$ is the diagonal matrix of the singular values (equivalently eigenvalues) of $m\mathbf{C}$ and $\mathbf{U}$ is the matrix of the singular vectors (equivalently eigenvectors) of $\mathbf{C}$ (and $m\mathbf{C}$).

Starting with the SVD of $\mathbf{X}$, note that right multiplying by $\mathbf{V\Sigma}^{-1}$ and using the relationship between $\mathbf{\Lambda}$ and $\mathbf{\Sigma}$ yields $\mathbf{U} = \mathbf{XV\Lambda}^{-1/2}$. Thus, the singular vector $\mathbf{u}$ of $\mathbf{C}$ associated to the singular value λ/m coincides with $\frac{\mathbf{Xv}}{\sqrt{\lambda}}$, where $\mathbf{v}$ is the singular vector of $\mathbf{K}$ associated to λ. Now fix an arbitrary feature vector $\mathbf{x} = \mathbf{\Phi}(x)$ for $x \in \mathcal{X}$. Then, following the expression for $\mathbf{Y}$ in theorem 15.1, the one-dimensional representation of $\mathbf{x}$ derived by projection onto $\mathbf{P}_u = \mathbf{u}\mathbf{u}^\top$ is defined by

$$\mathbf{x}^\top \mathbf{u} = \mathbf{x}^\top \frac{\mathbf{X}\mathbf{v}}{\sqrt{\lambda}} = \frac{\mathbf{k}_x^\top \mathbf{v}}{\sqrt{\lambda}}\,, \tag{15.4}$$

where $\mathbf{k}_x = (K(x_1, x), \ldots, K(x_m, x))^\top$. If $\mathbf{x}$ is one of the data points, i.e., $\mathbf{x} = \mathbf{x}_i$ for $1 \leq i \leq m$, then $\mathbf{k}_x$ is the ith column of $\mathbf{K}$ and (15.4) can be simplified as follows:

$$\mathbf{x}^\top \mathbf{u} = \frac{\mathbf{k}_x^\top \mathbf{v}}{\sqrt{\lambda}} = \frac{\lambda v_i}{\sqrt{\lambda}} = \sqrt{\lambda} v_i\,, \tag{15.5}$$

where v_i is the ith component of $\mathbf{v}$. More generally, the PCA solution of theorem 15.1 can be fully defined by the top k singular vectors (or eigenvectors) of $\mathbf{K}$, $\mathbf{v}_1, \ldots, \mathbf{v}_k$, and the corresponding singular values (or eigenvalues). This alternative derivation of the PCA solution in terms of $\mathbf{K}$ precisely defines the KPCA solution, providing a generalization of PCA via the use of PDS kernels (see chapter 6 for more details on kernel methods).

15.3 KPCA and manifold learning

Several manifold learning techniques have been proposed as non-linear methods for dimensionality reduction. These algorithms implicitly assume that high-dimensional data lie on or near a low-dimensional non-linear manifold embedded in the input space. They aim to learn this manifold structure by finding a low-dimensional space that in some way preserves the local structure of high-dimensional input data. For instance, the Isomap algorithm aims to preserve approximate geodesic distances, or distances along the manifold, between all pairs of data points. Other algorithms, such as Laplacian eigenmaps and locally linear embedding, focus only on preserving local neighborhood relationships in the high-dimensional space. We will next describe these classical manifold learning algorithms and then interpret them as specific instances of KPCA.

15.3.1 Isomap

Isomap aims to extract a low-dimensional data representation that best preserves all pairwise distances between input points, as measured by their geodesic distances along the underlying manifold. It approximates geodesic distance assuming that L_2 distance provides good approximations for nearby points, and for faraway points it estimates distance as a series of hops between neighboring points. The Isomap algorithm works as follows:

1. Find the t nearest neighbors for each data point based on L_2 distance and construct an undirected neighborhood graph, denoted by $\mathcal{G}$, with points as nodes and links between neighbors as edges.
2. Compute the approximate geodesic distances, Δ_{ij}, between all pairs of nodes (i, j) by computing all-pairs shortest distances in $\mathcal{G}$ using, for instance, the Floyd-Warshall algorithm.
3. Convert the squared distance matrix into a $m \times m$ similarity matrix by performing double centering, i.e., compute $\mathbf{K}_{\text{Iso}} = -\frac{1}{2}\mathbf{H}\Delta\mathbf{H}$, where Δ is the squared distance matrix, $\mathbf{H} = \mathbf{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$ is the centering matrix, $\mathbf{I}_m$ is the $m \times m$ identity matrix and $\mathbf{1}$ is a column vector of all ones (for more details on double centering see exercise 15.2).
4. Find the optimal k-dimensional representation, $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^n$, such that $\mathbf{Y} = \operatorname{argmin}_{\mathbf{Y}'} \sum_{i,j} \left(\|\mathbf{y}'_i - \mathbf{y}'_j\|_2^2 - \Delta_{ij}^2\right)$. The solution is given by,
$$\mathbf{Y} = (\mathbf{\Sigma}_{\text{Iso},k})^{1/2}\mathbf{U}_{\text{Iso},k}^\top \tag{15.6}$$
where $\mathbf{\Sigma}_{\text{Iso},k}$ is the diagonal matrix of the top k singular values of $\mathbf{K}_{\text{Iso}}$ and $\mathbf{U}_{\text{Iso},k}$ are the associated singular vectors.

$\mathbf{K}_{\text{Iso}}$ can naturally be viewed as a kernel matrix, thus providing a simple connection between Isomap and KPCA. Note, however, that this interpretation is valid only when $\mathbf{K}_{\text{Iso}}$ is in fact positive semidefinite, which is indeed the case in the continuum limit for a smooth manifold.

15.3.2 Laplacian eigenmaps

The *Laplacian eigenmaps* algorithm aims to find a low-dimensional representation that best preserves neighborhood relations as measured by a weight matrix $\mathbf{W}$. The algorithm works as follows:

1. Find t nearest neighbors for each point.
2. Construct $\mathbf{W}$, a sparse, symmetric $m \times m$ matrix, where $\mathbf{W}_{ij} = \exp\big(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\sigma^2\big)$ if $(\mathbf{x}_i, \mathbf{x}_j)$ are neighbors, 0 otherwise, and σ is a scaling parameter.
3. Construct the diagonal matrix $\mathbf{D}$, such that $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$.
4. Find the k-dimensional representation by minimizing the weighted distance between neighbors as,
$$\mathbf{Y} = \operatorname*{argmin}_{\mathbf{Y}'} \sum_{i,j} \mathbf{W}_{ij} \|\mathbf{y}'_i - \mathbf{y}'_j\|_2^2. \tag{15.7}$$

   This objective function penalizes nearby inputs for being mapped to faraway outputs, with "nearness" measured by the weight matrix $\mathbf{W}$. The solution to the minimization in (15.7) is $\mathbf{Y} = \mathbf{U}_{\mathbf{L},k}^\top$, where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian and $\mathbf{U}_{\mathbf{L},k}^{\top}$ are the bottom k singular vectors of $\mathbf{L}$, excluding the last singular vector corresponding to the singular value 0 (assuming that the underlying neighborhood graph is connected).

The solution to (15.7) can also be interpreted as finding the largest singular vectors of $\mathbf{L}^{\dagger}$, the pseudo-inverse of $\mathbf{L}$. Defining $\mathbf{K}_{\mathbf{L}} = \mathbf{L}^{\dagger}$ we can thus view Laplacian Eigenmaps as an instance of KPCA in which the output dimensions are normalized to have unit variance, which corresponds to setting $\lambda = 1$ in (15.5). Moreover, it can be shown that $\mathbf{K}_{\mathbf{L}}$ is the kernel matrix associated with the commute times of diffusion on the underlying neighborhood graph, where the commute time between nodes i and j in a graph is the expected time taken for a random walk to start at node i, reach node j and then return to i.

15.3.3 Locally linear embedding (LLE)

The *locally linear embedding* (LLE) algorithm also aims to find a low-dimensional representation that preserves neighborhood relations as measured by a weight matrix $\mathbf{W}$. The algorithm works as follows:

1. Find t nearest neighbors for each point.
2. Construct $\mathbf{W}$, a sparse, symmetric $m \times m$ matrix, whose ith row sums to one and contains the linear coefficients that optimally reconstruct $\mathbf{x}_i$ from its t neighbors. More specifically, if we assume that the ith row of $\mathbf{W}$ sums to one, then the reconstruction error is

$$\Big(\mathbf{x}_i - \sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij}\mathbf{x}_j\Big)^2 = \Big(\sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij}(\mathbf{x}_i - \mathbf{x}_j)\Big)^2 = \sum_{j,k \in \mathcal{N}_i} \mathbf{W}_{ij}\mathbf{W}_{ik}\mathbf{C}'_{jk} \tag{15.8}$$

   where $\mathcal{N}_i$ is the set of indices of the neighbors of point $\mathbf{x}_i$ and $\mathbf{C}'_{jk} = (\mathbf{x}_i - \mathbf{x}_j)^{\top}(\mathbf{x}_i - \mathbf{x}_k)$ the local covariance matrix. Minimizing this expression with the constraint $\sum_j \mathbf{W}_{ij} = 1$ gives the solution

$$\mathbf{W}_{ij} = \frac{\sum_k (\mathbf{C}'^{-1})_{jk}}{\sum_{st} (\mathbf{C}'^{-1})_{st}}\,. \tag{15.9}$$

   Note that the solution can be equivalently obtained by first solving the system of linear equations $\sum_j \mathbf{C}'_{kj}\mathbf{W}_{ij} = 1$, for $k \in \mathcal{N}_i$, and then normalizing so that the weights sum to one.
3. Find the k-dimensional representation that best obeys neighborhood relations as specified by $\mathbf{W}$, i.e.,

$$\mathbf{Y} = \operatorname*{argmin}_{\mathbf{Y}'} \sum_i \Big(\mathbf{y}'_i - \sum_j \mathbf{W}_{ij}\mathbf{y}'_j\Big)^2. \tag{15.10}$$

The solution to the minimization in (15.10) is $\mathbf{Y} = \mathbf{U}_{\mathbf{M},k}^\top$, where $\mathbf{M} = (\mathbf{I} - \mathbf{W}^\top)(\mathbf{I} - \mathbf{W}^\top)$ and $\mathbf{U}_{\mathbf{M},k}^\top$ are the bottom k singular vectors of $\mathbf{M}$, excluding the last singular vector corresponding to the singular value 0.

As discussed in exercise 15.5, LLE coincides with KPCA used with a particular kernel matrix $\mathbf{K}_{LLE}$ whereby the output dimensions are normalized to have unit variance (as in the case of Laplacian Eigenmaps).

15.4 Johnson-Lindenstrauss lemma

The Johnson-Lindenstrauss lemma is a fundamental result in dimensionality reduction that states that any m points in high-dimensional space can be mapped to a much lower dimension, $k \geq O(\frac{\log m}{\epsilon^2})$, without distorting pairwise distance between any two points by more than a factor of $(1 \pm \epsilon)$. In fact, such a mapping can be found in randomized polynomial time by projecting the high-dimensional points onto randomly chosen k-dimensional linear subspaces. The Johnson-Lindenstrauss lemma is formally presented in lemma 15.4. The proof of this lemma hinges on lemma 15.2 and lemma 15.3, and it is an example of the "probabilistic method", in which probabilistic arguments lead to a deterministic statement. Moreover, as we will see, the Johnson-Lindenstrauss lemma follows by showing that the squared norm of a random vector is sharply concentrated around its mean when the vector is projected onto a k-dimensional random subspace.

First, we prove the following property of the χ^2 distribution (see definition C.7 in appendix), which will be used in lemma 15.3.

Lemma 15.2 *Let Q be a random variable following a χ^2 distribution with k degrees of freedom. Then, for any $0 < \epsilon < 1/2$, the following inequality holds:*

$$\mathbb{P}[(1-\epsilon)k \leq Q \leq (1+\epsilon)k] \geq 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}. \tag{15.11}$$

Proof: By Markov's inequality, we can write

$$\begin{aligned}\mathbb{P}[Q \geq (1+\epsilon)k] = \mathbb{P}[\exp(\lambda Q) \geq \exp(\lambda(1+\epsilon)k)] &\leq \frac{\mathbb{E}[\exp(\lambda Q)]}{\exp(\lambda(1+\epsilon)k)} \\ &= \frac{(1-2\lambda)^{-k/2}}{\exp(\lambda(1+\epsilon)k)},\end{aligned}$$

where we used for the final equality the expression of the moment-generating function of a χ^2 distribution, $\mathbb{E}[\exp(\lambda Q)]$, for $\lambda < 1/2$ (equation (C.25)). Choosing $\lambda = \frac{\epsilon}{2(1+\epsilon)} < 1/2$, which minimizes the right-hand side of the final equality, and using the inequality $1 + \epsilon \leq \exp(\epsilon - (\epsilon^2 - \epsilon^3)/2)$ yield

$$\mathbb{P}[Q \geq (1+\epsilon)k] \leq \left(\frac{1+\epsilon}{\exp(\epsilon)}\right)^{k/2} \leq \left(\frac{\exp\left(\epsilon - \frac{\epsilon^2-\epsilon^3}{2}\right)}{\exp(\epsilon)}\right)^{k/2} = \exp\Big(-\frac{k}{4}(\epsilon^2 - \epsilon^3)\Big).$$

The statement of the lemma follows by using similar techniques to bound $\mathbb{P}[Q \leq (1-\epsilon)k]$ and by applying the union bound. □

Lemma 15.3 *Let $\mathbf{x} \in \mathbb{R}^N$, define $k < N$ and assume that entries in $\mathbf{A} \in \mathbb{R}^{k \times N}$ are sampled independently from the standard normal distribution, $N(0,1)$. Then, for any $0 < \epsilon < 1/2$,*

$$\mathbb{P}\left[(1-\epsilon)\|\mathbf{x}\|^2 \leq \|\frac{1}{\sqrt{k}}\mathbf{A}\mathbf{x}\|^2 \leq (1+\epsilon)\|\mathbf{x}\|^2\right] \geq 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}. \tag{15.12}$$

Proof: Let $\widehat{\mathbf{x}} = \mathbf{A}\mathbf{x}$ and observe that

$$\mathbb{E}[\widehat{x}_j^2] = \mathbb{E}\left[\Big(\sum_{i=1}^N A_{ji}x_i\Big)^2\right] = \mathbb{E}\left[\sum_{i=1}^N A_{ji}^2 x_i^2\right] = \sum_{i=1}^N x_i^2 = \|\mathbf{x}\|^2.$$

The second and third equalities follow from the independence and unit variance, respectively, of the A_{ij}. Now, define $T_j = \widehat{x}_j/\|\mathbf{x}\|$ and note that the T_js are independent standard normal random variables since the A_{ij} are i.i.d. standard normal random variables and $\mathbb{E}[\widehat{x}_j^2] = \|\mathbf{x}\|^2$. Thus, the variable Q defined by $Q = \sum_{j=1}^k T_j^2$ follows a χ^2 distribution with k degrees of freedom and we have

$$\begin{aligned}\mathbb{P}\left[(1-\epsilon)\|\mathbf{x}\|^2 \leq \frac{\|\widehat{\mathbf{x}}\|^2}{k} \leq (1+\epsilon)\|\mathbf{x}\|^2\right] &= \mathbb{P}\left[(1-\epsilon)k \leq \sum_{j=1}^k T_j^2 \leq (1+\epsilon)k\right] \\ &= \mathbb{P}\left[(1-\epsilon)k \leq Q \leq (1+\epsilon)k\right] \\ &\geq 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4},\end{aligned}$$

where the final inequality holds by lemma 15.2, thus proving the statement of the lemma. □

Lemma 15.4 (Johnson-Lindenstrauss) *For any $0 < \epsilon < 1/2$ and any integer $m > 4$, let $k = \frac{20 \log m}{\epsilon^2}$. Then for any set V of m points in $\mathbb{R}^N$, there exists a map $f \colon \mathbb{R}^N \to \mathbb{R}^k$ such that for all $\mathbf{u}, \mathbf{v} \in V$,*

$$(1-\epsilon)\|\mathbf{u}-\mathbf{v}\|^2 \leq \|f(\mathbf{u}) - f(\mathbf{v})\|^2 \leq (1+\epsilon)\|\mathbf{u}-\mathbf{v}\|^2. \tag{15.13}$$

Proof: Let $f = \frac{1}{\sqrt{k}}\mathbf{A}$ where $k < N$ and entries in $\mathbf{A} \in \mathbb{R}^{k \times N}$ are sampled independently from the standard normal distribution, $N(0,1)$. For fixed $\mathbf{u}, \mathbf{v} \in V$, we can apply lemma 15.3, with $\mathbf{x} = \mathbf{u} - \mathbf{v}$, to lower bound the success probability by $1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}$. Applying the union bound over the $O(m^2)$ pairs in V, setting $k = \frac{20}{\epsilon^2}\log m$ and upper bounding ϵ by $1/2$, we have

$$\mathbb{P}[success] \geq 1 - 2m^2 e^{-(\epsilon^2-\epsilon^3)k/4} = 1 - 2m^{5\epsilon-3} > 1 - 2m^{-1/2} > 0.$$

Since the success probability is strictly greater than zero, a map that satisfies the desired conditions must exist, thus proving the statement of the lemma. □

15.5 Chapter notes

PCA was introduced in the early 1900s by Pearson [1901]. KPCA was introduced roughly a century later, and our presentation of KPCA is a more concise derivation of results given by Mika et al. [1999]. Isomap and LLE were pioneering works on non-linear dimensionality reduction introduced by Tenenbaum et al. [2000], Roweis and Saul [2000]. Isomap itself is a generalization of a standard linear dimensionality reduction technique called Multidimensional Scaling [Cox and Cox, 2000]. Isomap and LLE led to the development of several related algorithms for manifold learning, e.g., Laplacian Eigenmaps and Maximum Variance Unfolding [Belkin and Niyogi, 2001, Weinberger and Saul, 2006]. As shown in this chapter, classical manifold learning algorithms are special instances of KPCA [Ham et al., 2004]. The Johnson-Lindenstrauss lemma was introduced by Johnson and Lindenstrauss [1984], though our proof of the lemma follows Vempala [2004]. Other simplified proofs of this lemma have also been presented, including Dasgupta and Gupta [2003].

15.6 Exercises

15.1 PCA and maximal variance. Let $\mathbf{X}$ be an *uncentered* data matrix and let $\bar{\mathbf{x}} = \frac{1}{m}\sum_i \mathbf{x}_i$ be the sample mean of the columns of $\mathbf{X}$.

(a) Show that the variance of one-dimensional projections of the data onto an arbitrary vector $\mathbf{u}$ equals $\mathbf{u}^\top \mathbf{C}\mathbf{u}$, where $\mathbf{C} = \frac{1}{m}\sum_i (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$ is the sample covariance matrix.

(b) Show that PCA with $k = 1$ projects the data onto the direction (i.e., $\mathbf{u}^\top\mathbf{u} = 1$) of maximal variance.

15.2 Double centering. In this problem we will prove the correctness of the double centering step in Isomap when working with Euclidean distances. Define $\mathbf{X}$ and $\bar{\mathbf{x}}$ as in exercise 15.1, and define $\mathbf{X}^*$ as the centered version of $\mathbf{X}$, that is, let $\mathbf{x}_i^* = \mathbf{x}_i - \bar{\mathbf{x}}$ be the ith column of $\mathbf{X}^*$. Let $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$, and let $\mathbf{D}$ denote the Euclidean distance matrix, i.e., $\mathbf{D}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$.

(a) Show that $\mathbf{K}_{ij} = \frac{1}{2}(\mathbf{K}_{ii} + \mathbf{K}_{jj} + \mathbf{D}_{ij}^2)$.

(b) Show that $\mathbf{K}^* = \mathbf{X}^{*\top}\mathbf{X}^* = \mathbf{K} - \frac{1}{m}\mathbf{K}\mathbf{1}\mathbf{1}^\top - \frac{1}{m}\mathbf{1}\mathbf{1}^\top\mathbf{K} + \frac{1}{m^2}\mathbf{1}\mathbf{1}^\top\mathbf{K}\mathbf{1}\mathbf{1}^\top$.

(c) Using the results from (a) and (b) show that

$$\mathbf{K}_{ij}^* = -\frac{1}{2}\left[\mathbf{D}_{ij}^2 - \frac{1}{m}\sum_{k=1}^{m}\mathbf{D}_{ik}^2 - \frac{1}{m}\sum_{k=1}^{m}\mathbf{D}_{kj}^2 + \bar{\mathbf{D}}\right],$$

where $\bar{\mathbf{D}} = \frac{1}{m^2}\sum_u \sum_v \mathbf{D}^2_{u,v}$ is the mean of the m^2 entries in $\mathbf{D}$.

(d) Show that $\mathbf{K}^* = -\frac{1}{2}\mathbf{HDH}$, where $\mathbf{H} = \mathbf{I}_m - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$.

15.3 Laplacian eigenmaps. Assume $k = 1$ and we seek a one-dimensional representation $\mathbf{y}$. Show that (15.7) is equivalent to $\mathbf{y} = \operatorname{argmin}_{\mathbf{y}'} \mathbf{y}'^\top \mathbf{L}\mathbf{y}'$, where $\mathbf{L}$ is the graph Laplacian.

15.4 Nyström method. Define the following block representation of a kernel matrix:

$$\mathbf{K} = \begin{bmatrix} \mathbf{W} & \mathbf{K}_{21}^\top \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} \mathbf{W} \\ \mathbf{K}_{21} \end{bmatrix}.$$

The Nyström method uses $\mathbf{W} \in \mathbb{R}^{l\times l}$ and $\mathbf{C} \in \mathbb{R}^{m\times l}$ to generate the approximation $\widetilde{\mathbf{K}} = \mathbf{C}\mathbf{W}^\dagger\mathbf{C}^\top \approx \mathbf{K}$.

(a) Show that $\mathbf{W}$ is SPSD and that $\|\mathbf{K} - \widetilde{\mathbf{K}}\|_F = \|\mathbf{K}_{22} - \mathbf{K}_{21}\mathbf{W}^\dagger\mathbf{K}_{21}^\top\|_F$.

(b) Let $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$ for some $\mathbf{X} \in \mathbb{R}^{N\times m}$, and let $\mathbf{X}' \in \mathbb{R}^{N\times l}$ be the first l columns of $\mathbf{X}$. Show that $\widetilde{\mathbf{K}} = \mathbf{X}^\top\mathbf{P}_{U_{X'}}\mathbf{X}$, where $\mathbf{P}_{U_{X'}}$ is the orthogonal projection onto the span of the left singular vectors of $\mathbf{X}'$.

(c) Is $\widetilde{\mathbf{K}}$ SPSD?

(d) If $\operatorname{rank}(\mathbf{K}) = \operatorname{rank}(\mathbf{W}) = r \ll m$, show that $\widetilde{\mathbf{K}} = \mathbf{K}$. Note: this statement holds whenever $\operatorname{rank}(\mathbf{K}) = \operatorname{rank}(\mathbf{W})$, but is of interest mainly in the low-rank setting.

(e) If $m = 20\text{M}$ and $\mathbf{K}$ is a dense matrix, how much space is required to store $\mathbf{K}$ if each entry is stored as a double? How much space is required by the Nyström method if $l = 10\text{K}$?

15.5 Expression for $\mathbf{K}_{LLE}$. Show the connection between LLE and KPCA by deriving the expression for $\mathbf{K}_{LLE}$.

15.6 Random projection, PCA, and nearest neighbors.

(a) Download the MNIST test set of handwritten digits at:

`http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz.`

Create a data matrix $\mathbf{X} \in \mathbb{R}^{N\times m}$ from the first $m = 2{,}000$ instances of this dataset (the dimension of each instance should be $N = 784$).

(b) Find the ten nearest neighbors for each point in $\mathbf{X}$, that is, compute $\mathcal{N}_{i,10}$ for $1 \le i \le m$, where $\mathcal{N}_{i,t}$ denotes the set of the t nearest neighbors for the

ith datapoint and nearest neighbors are defined with respect to the L_2 norm. Also compute $\mathcal{N}_{i,50}$ for all i.

(c) Generate $\tilde{\mathbf{X}} = \mathbf{AX}$, where $\mathbf{A} \in \mathbb{R}^{k \times N}$, $k = 100$ and entries of $\mathbf{A}$ are sampled independently from the standard normal distribution. Find the ten nearest neighbors for each point in $\tilde{\mathbf{X}}$, that is, compute $\tilde{\mathcal{N}}_{i,10}$ for $1 \leq i \leq m$.

(d) Report the quality of approximation by computing $\text{score}_{10} = \frac{1}{m} \sum_{i=1}^{m} |\mathcal{N}_{i,10} \cap \tilde{\mathcal{N}}_{i,10}|$. Similarly, compute $\text{score}_{50} = \frac{1}{m} \sum_{i=1}^{m} |\mathcal{N}_{i,50} \cap \tilde{\mathcal{N}}_{i,10}|$.

(e) Generate two plots that show score_{10} and score_{50} as functions of k (i.e., perform steps (c) and (d) for $k = \{1, 10, 50, 100, 250, 500\}$). Provide a one- or two-sentence explanation of these plots.

(f) Generate similar plots as in (e) using PCA (with various values of k) to generate $\tilde{\mathbf{X}}$ and subsequently compute nearest neighbors. Are the nearest neighbor approximations generated via PCA better or worse than those generated via random projections? Explain why.

16 Learning Automata and Languages

This chapter presents an introduction to the problem of learning languages. This is a classical problem explored since the early days of formal language theory and computer science, and there is a very large body of literature dealing with related mathematical questions. In this chapter, we present a brief introduction to this problem and concentrate specifically on the question of learning finite automata, which, by itself, has been a topic investigated in multiple forms by thousands of technical papers. We will examine two broad frameworks for learning automata, and for each, we will present an algorithm. In particular, we describe an algorithm for learning automata in which the learner has access to several types of query, and we discuss an algorithm for identifying a sub-class of the family of automata in the limit.

16.1 Introduction

Learning languages is one of the earliest problems discussed in linguistics and computer science. It has been prompted by the remarkable faculty of humans to learn natural languages. Humans are capable of uttering well-formed new sentences at an early age, after having been exposed only to finitely many sentences. Moreover, even at an early age, they can make accurate judgments of grammaticality for new sentences.

In computer science, the problem of learning languages is directly related to that of learning the representation of the computational device generating a language. Thus, for example, learning regular languages is equivalent to learning finite automata, or learning context-free languages or context-free grammars is equivalent to learning pushdown automata.

There are several reasons for examining specifically the problem of learning finite automata. Automata provide natural modeling representations in a variety of different domains including systems, networking, image processing, text and speech

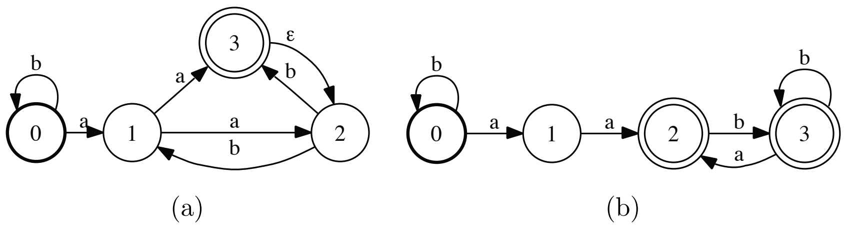


Figure 16.1
(a) A graphical representation of a finite automaton. (b) Equivalent (minimal) deterministic automaton.

processing, logic and many others. Automata can also serve as simple or efficient approximations for more complex devices. For example, in natural language processing, they can be used to approximate context-free languages. When it is possible, learning automata is often efficient, though, as we shall see, the problem is hard in a number of natural scenarios. Thus, learning more complex devices or languages is even harder.

We consider two general learning frameworks: the model of *efficient exact learning* and the model of *identification in the limit*. For each of these models, we briefly discuss the problem of learning automata and describe an algorithm.

We first give a brief review of some basic automata definitions and algorithms, then discuss the problem of efficient exact learning of automata and that of the identification in the limit.

16.2 Finite automata

We will denote by Σ a finite alphabet. The length of a string $x \in \Sigma^*$ over that alphabet is denoted by $|x|$. The *empty string* is denoted by ϵ, thus $|\epsilon| = 0$. For any string $x = x_1 \cdots x_k \in \Sigma^*$ of length $k \geq 0$, we denote by $x[j] = x_1 \cdots x_j$ its prefix of length $j \leq k$ and define $x[0]$ as ϵ.

Finite automata are labeled directed graphs equipped with initial and final states. The following gives a formal definition of these devices.

Definition 16.1 (Finite automata) *A* finite automaton *A is a 5-tuple (Σ, Q, I, F, E) where Σ is a finite alphabet, Q a finite set of states, $I \subseteq Q$ a set of initial states, $F \subseteq Q$ a set of final states, and $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$ a finite set of* transitions.

Figure 16.1a shows a simple example of a finite automaton. States are represented by circles. A bold circle indicates an initial state, a double circle a final state. Each transition is represented by an arrow from its origin state to its destination state with its label in $\Sigma \cup \{\epsilon\}$.

A path from an initial state to a final state is said to be an *accepting path*. An automaton is said to be *trim* if all of its states are accessible from an initial state and admit a path to a final state, that is, if all of its states lie on an accepting path. A string $x \in \Sigma^*$ is *accepted* by an automaton A iff x labels an accepting path. For convenience, we will say that $x \in \Sigma^*$ is *rejected* by A when it is not accepted. The set of all strings accepted by A defines the *language accepted by A* denoted by $L(A)$. The class of languages accepted by finite automata coincides with the family of *regular languages*, that is, languages that can be described by *regular expressions*.

Any finite automaton admits an equivalent automaton with no *ϵ-transition*, that is, no transition labeled with the empty string: there exists a general ϵ-removal algorithm that takes as input an automaton and returns an equivalent automaton with no ϵ-transition.

An automaton with no *ϵ-transition* is said to be *deterministic* if it admits a unique initial state and if no two transitions sharing the same label leave any given state. A deterministic finite automaton is often referred to by the acronym *DFA*, while the acronym *NFA* is used for arbitrary automata, that is, non-deterministic finite automata. Any NFA admits an equivalent DFA: there exists a general (exponential-time) *determinization* algorithm that takes as input an NFA with no ϵ-transition and returns an equivalent DFA. Thus, the class of languages accepted by DFAs coincides with that of the languages accepted by NFAs, that is regular languages. For any string $x \in \Sigma^*$ and DFA A, we denote by $A(x)$ the state reached in A when reading x from its unique initial state.

A DFA is said to be *minimal* if it admits no equivalent deterministic automaton with a smaller number of states. There exists a general *minimization* algorithm taking as input a deterministic automaton and returning a minimal one that runs in $O(|E| \log |Q|)$. When the input DFA is *acyclic*, that is when it admits no path forming a cycle, it can be minimized in linear time $O(|Q|+|E|)$. Figure 16.1b shows the minimal DFA equivalent to the NFA of figure 16.1a.

16.3 Efficient exact learning

In the *efficient exact learning* framework, the problem consists of identifying a target concept c from a finite set of examples in time polynomial in the size of the representation of the concept and in an upper bound on the size of the representation of an example. Unlike the PAC-learning framework, in this model, there is no stochastic assumption, instances are not assumed to be drawn according to some unknown distribution. Furthermore, the objective is to identify the target concept *exactly*, without any approximation. A concept class $\mathcal{C}$ is said to be efficiently exactly learnable if there is an algorithm for efficient exact learning of any $c \in \mathcal{C}$.

We will consider two different scenarios within the framework of efficiently exact learning: a *passive* and an *active learning* scenario. The passive learning scenario is similar to the standard supervised learning scenario discussed in previous chapters but without any stochastic assumption: the learning algorithm *passively* receives data instances as in the PAC model and returns a hypothesis, but here, instances are not assumed to be drawn from any distribution. In the active learning scenario, the learner *actively* participates in the selection of the training samples by using various types of queries that we will describe. In both cases, we will focus more specifically on the problem of learning automata.

16.3.1 Passive learning

The problem of learning finite automata in this scenario is known as the *minimum consistent DFA learning problem* . It can be formulated as follows: the learner receives a finite sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$ with $x_i \in \Sigma^*$ and $y_i \in \{-1, +1\}$ for any $i \in [m]$. If $y_i = +1$, then x_i is an accepted string, otherwise it is rejected. The problem consists of using this sample to learn the smallest DFA A *consistent* with S, that is the automaton with the smallest number of states that accepts the strings of S with label $+1$ and rejects those with label -1. Note that seeking the smallest DFA consistent with S can be viewed as following Occam's razor principle.

The problem just described is distinct from the standard minimization of DFAs. A minimal DFA accepting exactly the strings of S labeled positively may not have the smallest number of states: in general there may be DFAs with fewer states accepting a superset of these strings and rejecting the negatively labeled sample strings. For example, in the simple case $S = ((a, +1), (b, -1))$, a minimal deterministic automaton accepting the unique positively labeled string a or the unique negatively labeled string b admits two states. However, the deterministic automaton accepting the language a^* accepts a and rejects b and has only one state.

Passive learning of finite automata turns out to be a computationally hard problem. The following theorems present several negative results known for this problem.

Theorem 16.2 *The problem of finding the smallest deterministic automaton consistent with a set of accepted or rejected strings is NP-complete.*

Hardness results are known even for a polynomial approximation, as stated by the following theorem.

Theorem 16.3 *If $P \neq NP$, then, no polynomial-time algorithm can be guaranteed to find a DFA consistent with a set of accepted or rejected strings of size smaller than* *a polynomial function of the smallest consistent DFA, even when the alphabet is reduced to just two elements.*

Other strong negative results are known for passive learning of finite automata under various cryptographic assumptions.

These negative results for passive learning invite us to consider alternative learning scenarios for finite automata. The next section describes a scenario leading to more positive results where the learner can actively participate in the data selection process using various types of queries.

16.3.2 Learning with queries

The model of *learning with queries* corresponds to that of a (minimal) teacher or oracle and an active learner. In this model, the learner can make the following two types of queries to which an oracle responds:

- *membership queries*: the learner requests the target label $f(x) \in \{-1, +1\}$ of an instance x and receives that label;
- *equivalence queries*: the learner conjectures hypothesis h; it receives the response yes if $h = f$, a counter-example otherwise.

We will say that a concept class $\mathcal{C}$ is *efficiently exactly learnable with membership and equivalence queries* when it is efficiently exactly learnable within this model.

This model is not realistic, since no such oracle is typically available in practice. Nevertheless, it provides a natural framework, which, as we shall see, leads to positive results. Note also that for this model to be significant, equivalence must be computationally testable. This would not be the case for some concept classes such as that of *context-free grammars*, for example, for which the equivalence problem is undecidable. In fact, equivalence must be further efficiently testable, otherwise the response to the learner cannot be supplied in a reasonable amount of time.[21]

Efficient exact learning within this model of learning with queries implies the following variant of PAC-learning: we will say that a concept class $\mathcal{C}$ is *PAC-learnable with membership queries* if it is PAC-learnable by an algorithm that has access to a polynomial number of membership queries.

Theorem 16.4 *Let $\mathcal{C}$ be a concept class that is efficiently exactly learnable with membership and equivalence queries, then $\mathcal{C}$ is PAC-learnable using membership queries.*

Proof: Let $\mathcal{A}$ be an algorithm for efficiently exactly learning $\mathcal{C}$ using membership and equivalence queries. Fix $\epsilon, \delta > 0$. We replace in the execution of $\mathcal{A}$ for learning

[21] For a human oracle, answering membership queries may also become very hard in some cases when the queries are near the class boundaries. This may also make the model difficult to adopt in practice.

target $c \in \mathcal{C}$, each equivalence query by a test of the current hypothesis on a polynomial number of labeled examples. Let $\mathcal{D}$ be the distribution according to which points are drawn. To simulate the tth equivalence query, we draw $m_t = \frac{1}{\epsilon}(\log \frac{1}{\delta} + t \log 2)$ points i.i.d. according to $\mathcal{D}$ to test the current hypothesis h_t. If h_t is consistent with all of these points, then the algorithm stops and returns h_t. Otherwise, one of the points drawn does not belong to h_t, which provides a counter-example.

Since $\mathcal{A}$ learns c exactly, it makes at most T equivalence queries, where T is polynomial in the size of the representation of the target concept and in an upper bound on the size of the representation of an example. Thus, if no equivalence query is positively responded by the simulation, the algorithm will terminate after T equivalence queries and return the correct concept c. Otherwise, the algorithm stops at the first equivalence query positively responded by the simulation. The hypothesis it returns is not an ϵ-approximation only if the equivalence query stopping the algorithm is incorrectly responded positively. By the union bound, since for any fixed $t \in [T]$, $\mathbb{P}[R(h_t) > \epsilon] \leq (1-\epsilon)^{m_t}$, the probability that for some $t \in [T]$, $R(h_t) > \epsilon$ can be bounded as follows:

$$\begin{aligned}\mathbb{P}[\exists t \in [T] \colon R(h_t) > \epsilon] &\leq \sum_{t=1}^{T} \mathbb{P}[R(h_t) > \epsilon] \\ &\leq \sum_{t=1}^{T} (1-\epsilon)^{m_t} \leq \sum_{t=1}^{T} e^{-m_t \epsilon} \leq \sum_{t=1}^{T} \frac{\delta}{2^t} \leq \sum_{t=1}^{+\infty} \frac{\delta}{2^t} = \delta.\end{aligned}$$

Thus, with probability at least $1-\delta$, the hypothesis returned by the algorithm is an ϵ-approximation. Finally, the maximum number of points drawn is $\sum_{t=1}^{T} m_t = \frac{1}{\epsilon}(T \log \frac{1}{\delta} + \frac{T(T+1)}{2} \log 2)$, which is polynomial in $1/\epsilon$, $1/\delta$, and T. Since the rest of the computational cost of $\mathcal{A}$ is also polynomial by assumption, this proves the PAC-learning of $\mathcal{C}$. $\square$

16.3.3 Learning automata with queries

In this section, we describe an algorithm for efficient exact learning of DFAs with membership and equivalence queries. We will denote by A the target DFA and by $\widehat{A}$ the DFA that is the current hypothesis of the algorithm. For the discussion of the algorithm, we assume without loss of generality that A is a minimal DFA.

The algorithm uses two sets of strings, U and V. U is a set of *access strings*: reading an access string $u \in U$ from the initial state of A leads to a state $A(u)$. The algorithm ensures that the states $A(u)$, $u \in U$, are all distinct. To do so, it uses a set V of *distinguishing strings*. Since A is minimal, for two distinct states q and q' of A, there must exist at least one string that leads to a final state from q and not from q', or vice versa. That string helps *distinguish* q and q'. The set of strings V

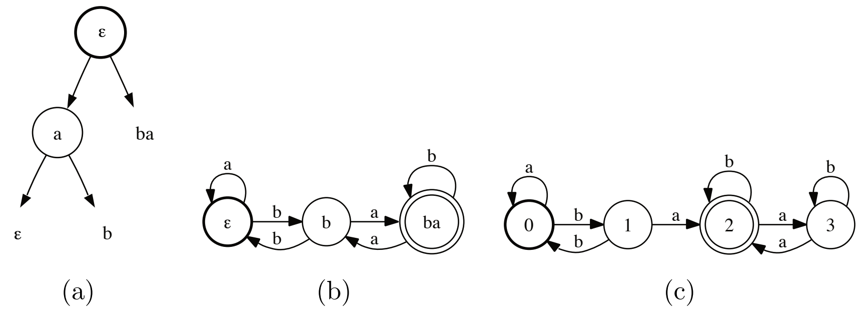


Figure 16.2
(a) Classification tree T, with $U = \{\epsilon, b, ba\}$ and $V = \{\epsilon, a\}$. (b) Current automaton $\widehat{A}$ constructed using T. (c) Target automaton A.

help distinguish any pair of access strings in U. They define in fact a partition of all strings of Σ^*.

The objective of the algorithm is to find at each iteration a new access string distinguished from all previous ones, ultimately obtaining a number of access strings equal to the number of states of A. It can then identify each state $A(u)$ of A with its access string u. To find the destination state of the transition labeled with $a \in \Sigma$ leaving state u, it suffices to determine, using the partition induced by V the access string u' that belongs to the same equivalence class as ua. The finality of each state can be determined in a similar way.

Both sets U and V are maintained by the algorithm via a binary decision tree T similar to those presented in chapter 9. Figure 16.2a shows an example. T defines the partition of all strings induced by the distinguishing strings V. The leaves of T are each labeled with a distinct $u \in U$ and its internal nodes with a string $v \in V$. The decision tree question defined by $v \in V$, given a string $x \in \Sigma^*$, is whether xv is accepted by A, which is determined via a membership query. If accepted, x is assigned to right sub-tree, otherwise to the left sub-tree, and the same is applied recursively with the sub-trees until a leaf is reached. We denote by $T(x)$ the label of the leaf reached. For example, for the tree T of figure 16.2a and target automaton A of figure 16.2c, $T(baa) = b$ since baa is not accepted by A (root question) and $baaa$ is (question at node a). At its initialization step, the algorithm ensures that the root node is labeled with ϵ, which is convenient to check the finality of the strings.

The tentative hypothesis DFA $\widehat{A}$ can be constructed from T as follows. We denote by ConstructAutomaton() the corresponding function. A distinct state $\widehat{A}(u)$ is created for each leaf $u \in U$. The finality of a state $\widehat{A}(u)$ is determined based on the sub-tree of the root node that u belongs to: $\widehat{A}(u)$ is made final iff u belongs

```
QueryLearnAutomata()
 1  $t \leftarrow$ MembershipQuery($\epsilon$)
 2  $T \leftarrow T_0$
 3  $\widehat{A} \leftarrow A_0$
 4  while (EquivalenceQuery($\widehat{A}$) $\neq$ true) do
 5      $x \leftarrow$ CounterExample()
 6      if ($T = T_0$) then
 7          $T \leftarrow T_1$ $\triangleright$ nil replaced with $x$.
 8      else $j \leftarrow \operatorname{argmin}_k A(x[k]) \not\equiv_T \widehat{A}(x[k])$
 9          Split($\widehat{A}(x[j-1])$)
10      $\widehat{A} \leftarrow$ ConstructAutomaton($T$)
11  return $\widehat{A}$
```

Figure 16.3
Algorithm for learning automata with membership and equivalence queries. A_0 is a single-state automaton with self-loops labeled with all $a \in \Sigma$. That state is initial. It is final iff $t =$ TRUE. T_0 is a tree with root node labeled with ϵ and two leaves, one labeled with ϵ, the other with NIL. the right leaf is labeled with ϵ labels iff $t =$ TRUE. T_1 is the tree obtained from T_0 by replacing NIL with x.

to the right sub-tree that is iff $u = \epsilon u$ is accepted by A. The destination of the transition labeled with $a \in \Sigma$ leaving state $\widehat{A}(u)$ is the state $\widehat{A}(v)$ where $v = T(ua)$. Figure 16.2b shows the DFA $\widehat{A}$ constructed from the decision tree of figure 16.2a. For convenience, for any $x \in \Sigma^*$, we denote by $U(\widehat{A}(x))$ the access string identifying state $\widehat{A}(x)$.

Figure 16.3 shows the pseudocode of the algorithm. The initialization steps at lines 1–3 construct a tree T with a single internal node labeled with ϵ and one leaf string labeled with ϵ, the other left undetermined and labeled with NIL. They also define a tentative DFA $\widehat{A}$ with a single state with self-loops labeled with all elements of the alphabet. That single state is an initial state. It is made a final state only if ϵ is accepted by the target DFA A, which is determined via the membership query of line 1.

At each iteration of the loop of lines 4–11, an equivalence query is used. If $\widehat{A}$ is not equivalent to A, then a counter-example string x is received (line 5). If T is the tree constructed in the initialization step, then the leaf labeled with NIL is replaced with x (lines 6–7). Otherwise, since x is a counter-example, states $A(x)$

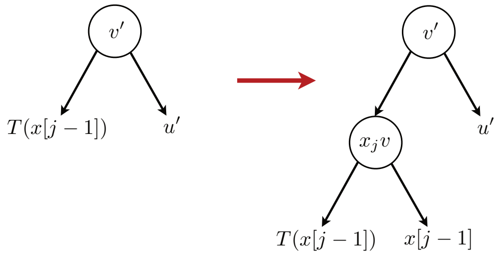


Figure 16.4
Illustration of the splitting procedure $\textsc{Split}(\widehat{A}(x[j-1]))$.

and $\widehat{A}(x)$ have a different finality; thus, the string x defining $A(x)$ and the access string $U(\widehat{A}(x))$ are assigned to different equivalence classes by T. Thus, there exists a smallest j such that $A(x[j])$ and $\widehat{A}(x[j])$ are not equivalent, that is, such that the prefix $x[j]$ of x and the access string $U(\widehat{A}(x[j]))$ are assigned to different leaves by T. j cannot be 0 since the initialization ensures that $\widehat{A}(\epsilon)$ is an initial state and has the same finality as the initial state $A(\epsilon)$ of A. The equivalence of $A(x[j])$ and $\widehat{A}(x[j])$ is tested by checking the equality of $T(x[j])$ and $T(U(\widehat{A}(x[j])))$, which can be both determined using the tree T and membership queries (line 8).

Now, by definition, $A(x[j-1])$ and $\widehat{A}(x[j-1])$ are equivalent, that is T assigns $x[j-1]$ to the leaf labeled with $U(\widehat{A}(x[j-1]))$. But, $x[j-1]$ and $U(\widehat{A}(x[j-1]))$ must be distinguished since $A(x[j-1])$ and $\widehat{A}(x[j-1])$ admit transitions labeled with the same label x_j to two non-equivalent states. Let v be a distinguishing string for $A(x[j])$ and $\widehat{A}(x[j])$. v can be obtained as the least common ancestor of the leaves labeled with $x[j]$ and $U(\widehat{A}(x[j]))$. To distinguish $x[j-1]$ and $U(\widehat{A}(x[j-1]))$, it suffices to split the leaf of T labeled with $T(x[j-1])$ to create an internal node $x_j v$ dominating a leaf labeled with $x[j-1]$ and another one labeled with $T(x[j-1])$ (line 9). Figure 16.4 illustrates this construction. Thus, this provides a new access string $x[j-1]$ which, by construction, is distinguished from $U(\widehat{A}(x[j-1]))$ and all other access strings.

Thus, the number of access strings (or states of $\widehat{A}$) increases by one at each iteration of the loop. When it reaches the number of states of A, all states of A are of the form $A(u)$ for a distinct $u \in U$. A and $\widehat{A}$ have then the same number of states and in fact $A = \widehat{A}$. Indeed, let $(A(u), a, A(u'))$ be a transition in A, then by definition the equality $A(ua) = A(u')$ holds. The tree T defines a partition of all strings in terms of their distinguishing strings in A. Since in A, ua and u' lead to the same state, they are assigned to the same leaf by T, that is, the leaf labeled with u'. The destination of the transition from $\widehat{A}(u)$ with label a is found by $\textsc{ConstructAutomaton}()$ by determining the leaf in T assigned to ua, that is, u'. Thus, by construction, the same transition $(\widehat{A}(u), a, \widehat{A}(u'))$ is created in $\widehat{A}$.

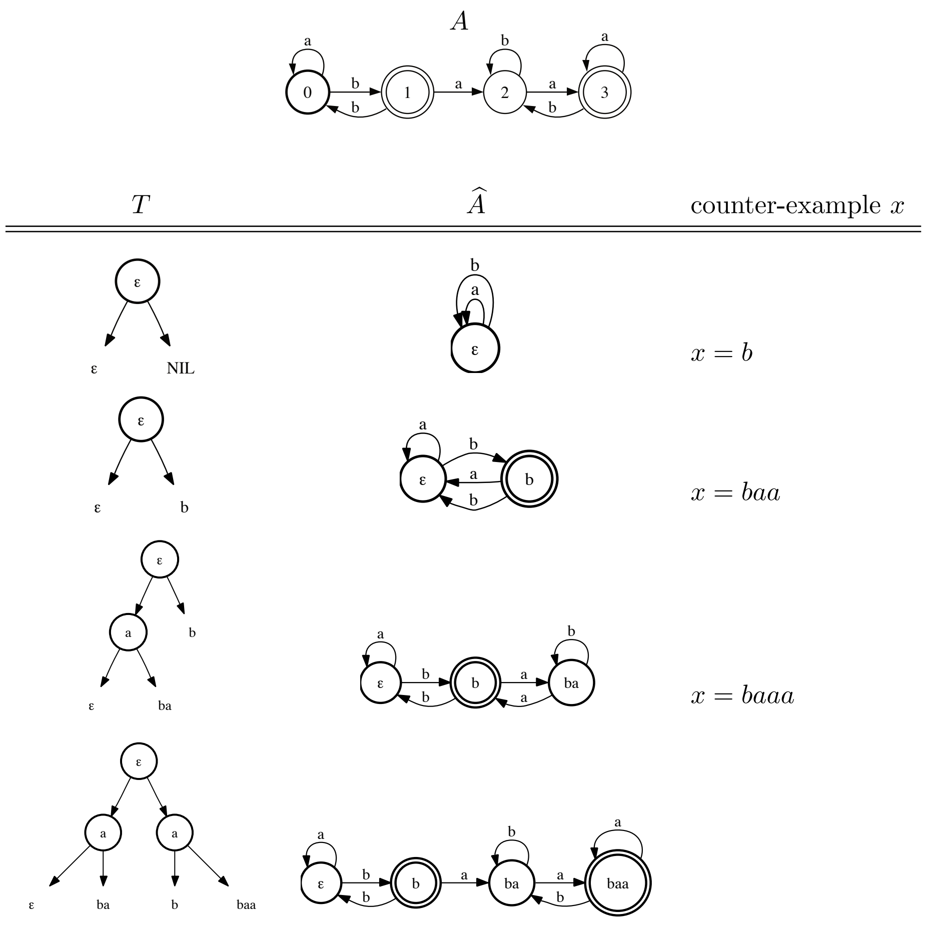


Figure 16.5
Illustration of the execution of Algorithm QueryLearnAutomata() for the target automaton A. Each line shows the current decision tree T and the tentative DFA $\widehat{A}$ constructed using T. When $\widehat{A}$ is not equivalent to A, the learner receives a counter-example x indicated in the third column.

Also, a state $A(u)$ of A is final iff u accepted by A that is iff u is assigned to the right sub-tree of the root node by T, which is the criterion determining the finality of $\widehat{A}(u)$. Thus, the automata A and $\widehat{A}$ coincide.

The following is the analysis of the running-time complexity of the algorithm. At each iteration, one new distinguished access string is found associated to a distinct state of A, thus, at most $|A|$ states are created. For each counter-example x, at most $|x|$ tree operations are performed. Constructing $\widehat{A}$ requires $O(|\Sigma||A|)$ tree operations. The cost of a tree operation is $O(|A|)$ since it consists of at most $|A|$

membership queries. Thus, the overall complexity of the algorithm is in $O(|\Sigma||A|^2 + n|A|)$, where n is the maximum length of a counter-example. Note that this analysis assumes that equivalence and membership queries are made in constant time.

Our analysis shows the following result.

Theorem 16.5 (Learning DFAs with queries) *The class of all DFAs is efficiently exactly learnable using membership and equivalence queries.*

Figure 16.5 illustrates a full execution of the algorithm in a specific case. In the next section, we examine a different learning scenario for automata.

16.4 Identification in the limit

In the *identification in the limit framework*, the problem consists of identifying a target concept c exactly after receiving a finite set of examples. A class of languages is said to be *identifiable in the limit* if there exists an algorithm that identifies any language L in that class after examining a finite number of examples and its hypothesis remains unchanged thereafter.

This framework is perhaps less realistic from a computational point of view since it requires no upper bound on the number of instances or the efficiency of the algorithm. Nevertheless, it has been argued by some to be similar to the scenario of humans learning languages. In this framework as well, negative results hold for the general problem of learning DFAs.

Theorem 16.6 *Deterministic automata are not identifiable in the limit from positive examples.*

Some sub-classes of finite automata can however be successfully identified in the limit. Most algorithms for inference of automata are based on a *state-partitioning paradigm*. They start with an initial DFA, typically a tree accepting the finite set of sample strings available and the trivial partition: each block is reduced to one state of the tree. At each iteration, they merge partition blocks while preserving some congruence property. The iteration ends when no other merging is possible and the final partition defines the automaton inferred. Thus, the choice of the congruence fully determines the algorithm and a variety of different algorithms can be defined by varying that choice. A *state-splitting paradigm* can be similarly defined starting from the single-state automaton accepting Σ^*. In this section, we present an algorithm for learning reversible automata, which is a special instance of the general state-partitioning algorithmic paradigm just described.

Let $A = (\Sigma, Q, I, F, E)$ be a DFA and let π be a partition of Q. The DFA defined by the partition π is called the *automaton quotient of A and π*. It is denoted by

A/π and defined as follows: $A/\pi = (\Sigma, \pi, I_\pi, F_\pi, E_\pi)$ with

$$\begin{aligned}
I_\pi &= \{B \in \pi \colon I \cap B \neq \emptyset\}\\
F_\pi &= \{B \in \pi \colon F \cap B \neq \emptyset\}\\
E_\pi &= \{(B, a, B') \colon \exists (q, a, q') \in E \mid q \in B, q' \in B', B \in \pi, B' \in \pi\}.
\end{aligned}$$

Let S be a finite set of strings and let $\mathrm{Pref}(S)$ denote the set of prefixes of all strings of S. A *prefix-tree automaton* accepting exactly the set of strings S is a particular DFA denoted by $PT(S) = (\Sigma, \mathrm{Pref}(S), \{\epsilon\}, S, E_S)$ where Σ is the set of alphabet symbols used in S and E_S defined as follows:

$$E_S = \{(x, a, xa) \colon x \in \mathrm{Pref}(S), xa \in \mathrm{Pref}(S)\}.$$

Figure 16.7a shows the prefix-tree automaton of a particular set of strings S.

16.4.1 Learning reversible automata

In this section, we show that the sub-class of *reversible automata* or *reversible languages* can be identified in the limit. In particular, we show that the language can be identified given a *positive presentation.*

A positive presentation of a language L is an infinite sequence $(x_n)_{n \in \mathbb{N}}$ such that $\{x_n \colon n \in \mathbb{N}\} = L$. Thus, in particular, for any $x \in L$ there exists $n \in \mathbb{N}$ such that $x = x_n$. An algorithm identifies L in the limit from a positive presentation if there exists $N \in \mathbb{N}$ such that for $n \geq N$ the hypothesis it returns is L.

Given a DFA A, we define its *reverse* A^R as the automaton derived from A by making the initial state final, the final states initial, and by reversing the direction of every transition. The language accepted by the reverse of A is precisely the language of the reverse (or mirror image) of the strings accepted by A.

Definition 16.7 (Reversible automata) *A finite automaton A is said to be* reversible *iff both A and A^R are deterministic. A language L is said to be* reversible *if it is the language accepted by some reversible automaton.*

Some direct consequences of this definition are that a reversible automaton A has a unique final state and that its reverse A^R is also reversible. Note also that a trim reversible automaton A is minimal. Indeed, if states q and q' in A are equivalent, then, they admit a common string x leading both from q and from q' to a final state. But, by the reverse determinism of A, reading the reverse of x from the final state must lead to a unique state, which implies that $q = q'$.

For any $u \in \Sigma^*$ and any language $L \subseteq \Sigma^*$, let $\mathrm{Suff}_L(u)$ denote the set of all possible suffixes in L for u:

$$\mathrm{Suff}_L(u) = \{v \in \Sigma^* \colon uv \in L\}. \tag{16.1}$$

$\mathrm{Suff}_L(u)$ is also often denoted by $u^{-1}L$. Observe that if L is a reversible language, then the following implication holds for any two strings $u, u' \in \Sigma^*$:

$$\mathrm{Suff}_L(u) \cap \mathrm{Suff}_L(u') \neq \emptyset \implies \mathrm{Suff}_L(u) = \mathrm{Suff}_L(u'). \tag{16.2}$$

Indeed, let A be a reversible automaton accepting L. Let q be the state of A reached from the initial state when reading u and q' the one reached reading u'. If $v \in \mathrm{Suff}_L(u) \cap \mathrm{Suff}_L(u')$, then v can be read both from q and q' to reach the final state. Since A^R is deterministic, reading back the reverse of v from the final state must lead to a unique state, therefore $q = q'$, that is $\mathrm{Suff}_L(u) = \mathrm{Suff}_L(u')$.

Let $A = (\Sigma, Q, \{i_0\}, \{f_0\}, E)$ be a reversible automaton accepting a reversible language L. We define a set of strings S_L as follows:

$$S_L = \{d[q]f[q] : q \in Q\} \cup \{d[q], a, f[q'] : q, q' \in Q, a \in \Sigma\},$$

where $d[q]$ is a string of minimum length from i_0 to q, and $f[q]$ a string of minimum length from q to f_0. As shown by the following proposition, S_L characterizes the language L in the sense that any reversible language containing S_L must contain L.

Proposition 16.8 *Let L be a reversible language. Then, L is the smallest reversible language containing S_L.*

Proof: Let L' be a reversible language containing S_L and let $x = x_1 \cdots x_n$ be a string accepted by L, with $x_k \in \Sigma$ for $k \in [n]$ and $n \geq 1$. For convenience, we also define x_0 as ϵ. Let $(q_0, x_1, q_1) \cdots (q_{n-1}, x_n, q_n)$ be the accepting path in A labeled with x. We show by recurrence that $\mathrm{Suff}_{L'}(x_0 \cdots x_k) = \mathrm{Suff}_{L'}(d[q_k])$ for all $k \in \{0, \ldots, n\}$. Since $d[q_0] = d[i_0] = \epsilon$, this clearly holds for $k = 0$. Now assume that $\mathrm{Suff}_{L'}(x_0 \cdots x_k) = \mathrm{Suff}_{L'}(d[q_k])$ for some $k \in \{0, \ldots, n-1\}$. This implies immediately that $\mathrm{Suff}_{L'}(x_0 \cdots x_k x_{k+1}) = \mathrm{Suff}_{L'}(d[q_k]x_{k+1})$. By definition, S_L contains both $d[q_{k+1}]f[q_{k+1}]$ and $d[q_k]x_{k+1}f[q_{k+1}]$. Since L' includes S_L, the same holds for L'. Thus, $f[q_{k+1}]$ belongs to $\mathrm{Suff}_{L'}(d[q_{k+1}) \cap \mathrm{Suff}_{L'}(d[q_k]x_{k+1})$. In view of (16.2), this implies that $\mathrm{Suff}_{L'}(d[q_k]x_{k+1}) = \mathrm{Suff}_{L'}(d[q_{k+1}])$. Thus, we have $\mathrm{Suff}_{L'}(x_0 \cdots x_k x_{k+1}) = \mathrm{Suff}_{L'}(d[q_{k+1}])$. This shows that $\mathrm{Suff}_{L'}(x_0 \cdots x_k) = \mathrm{Suff}_{L'}(d[q_k])$ holds for all $k \in \{0, \ldots, n\}$, in particular, for $k = n$. Note that since $q_n = f_0$, we have $f[q_n] = \epsilon$, therefore $d[q_n] = d[q_n]f[q_n]$ is in $S_L \subseteq L'$, which implies that $\mathrm{Suff}_{L'}(d[q_n])$ contains ϵ and thus that $\mathrm{Suff}_{L'}(x_0 \cdots x_n)$ contains ϵ. This is equivalent to $x = x_0 \cdots x_n \in L'$. □

Figure 16.6 shows the pseudocode of an algorithm for inferring a reversible automaton from a sample S of m strings $x_1, \ldots, x_m$. The algorithm starts by creating a prefix-tree automaton A for S (line 1) and then iteratively defines a partition π of the states of A, starting with the trivial partition π_0 with one block per state (line 2). The automaton returned is the quotient of A and the final partition π defined.

```
LearnReversibleAutomata($S = (x_1, \ldots, x_m)$)
 1  $A = (\Sigma, Q, \{i_0\}, F, E) \leftarrow PT(S)$
 2  $\pi \leftarrow \pi_0$ ▷ trivial partition.
 3  list $\leftarrow \{(f, f') \colon f' \in F\}$ ▷ $f$ arbitrarily chosen in $F$.
 4  while list $\neq \emptyset$ do
 5      Remove(list, $(q_1, q_2)$)
 6      if $B(q_1, \pi) \neq B(q_2, \pi)$ then
 7          $B_1 \leftarrow B(q_1, \pi)$
 8          $B_2 \leftarrow B(q_2, \pi)$
 9          for all $a \in \Sigma$ do
10              if $(succ(B_1, a) \neq \emptyset) \wedge (succ(B_2, a) \neq \emptyset)$ then
11                  Add(list, $(succ(B_1, a), succ(B_2, a))$)
12              if $(pred(B_1, a) \neq \emptyset \wedge (pred(B_2, a) \neq \emptyset))$ then
13                  Add(list, $(pred(B_1, a), pred(B_2, a))$)
14              Update($succ, pred, B_1, B_2$)
15              $\pi \leftarrow$ Merge($\pi, B_1, B_2$)
16  return $A/\pi$
```

Figure 16.6
Algorithm for learning reversible automata from a set of positive strings S.

The algorithm maintains a list list of pairs of states whose corresponding blocks are to be merged, starting with all pairs of final states (f, f') for an arbitrarily chosen final state $f \in F$ (line 3). We denote by $B(q, \pi)$ the block containing q based on the partition π.

For each block B and alphabet symbol $a \in \Sigma$, the algorithm also maintains a successor $succ(B, a)$, that is, a state that can be reached by reading a from a state of B; $succ(B, a) = \emptyset$ if no such state exists. It maintains similarly the predecessor $pred(B, a)$, which is a state that admits a transition labeled with a leading to a state in B; $pred(B, a) = \emptyset$ if no such state exists.

Then, while list is not empty, a pair is removed from list and processed as follows. If the pair (q_1, q_1') has not been already merged, the pairs formed by the successors and predecessors of $B_1 = B(q_1, \pi)$ and $B_2 = B(q_2, \pi)$ are added to list

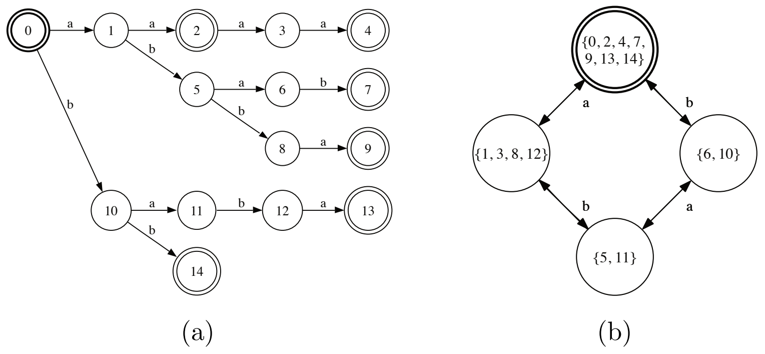


Figure 16.7
Example of inference of a reversible automaton. (a) Prefix-tree $PT(S)$ representing $S = (\epsilon, aa, bb, aaaa, abab, abba, baba)$. (b) Automaton $\widehat{A}$ returned by LearnReversibleAutomata() for the input S. A double-direction arrow represents two transitions with the same label with opposite directions. The language accepted by $\widehat{A}$ is that of strings with an even number of as and bs.

(lines 10–13). Before merging blocks B_1 and B_2 into a new block B' that defines a new partition π (line 15), the successor and predecessor values for the new block B' are defined as follows (line 14). For each symbol $a \in \Sigma$, $succ(B', a) = \emptyset$ if $succ(B_1, a) = succ(B_2, a) = \emptyset$, otherwise $succ(B', a)$ is set to one of $succ(B_1, a)$ if it is non-empty, $succ(B_2, a)$ otherwise. The predecessor values are defined in a similar way. Figure 16.7 illustrates the application of the algorithm in the case of a sample with $m = 7$ strings.

Proposition 16.9 *Let S be a finite set of strings and let $A = PT(S)$ be the prefix-tree automaton defined from S. Then, the final partition defined by* LearnReversibleAutomata() *used with input S is the finest partition π for which A/π is reversible.*

Proof: Let T be the number of iterations of the algorithm for the input sample S. We denote by π_t the partition defined by the algorithm after $t \geq 1$ iterations of the loop, with π_T the final partition.

A/π_T is a reversible automaton since all final states are guaranteed to be merged into the same block as a consequence of the initialization step of line 3 and, for any block B, by definition of the algorithm, states reachable by $a \in \Sigma$ from B are contained in the same block, and similarly for those admitting a transition labeled with a to a state of B.

Let π' be a partition of the states of A for which A/π' is reversible. We show by recurrence that π_T refines π'. Clearly, the trivial partition π_0 refines π'. Assume that π_s refines π' for all $s \leq t$. π_{t+1} is obtained from π by merging two blocks $B(q_1, \pi_t)$ and $B(q_2, \pi_t)$. Since π_t refines π', we must have $B(q_1, \pi_t) \subseteq B(q_1, \pi')$

and $B(q_2, \pi_t) \subseteq B(q_2, \pi')$. To show that π_{t+1} refines π', it suffices to prove that $B(q_1, \pi') = B(q_2, \pi')$.

A reversible automaton has only one final state, therefore, for the partition π', all final states of A must be placed in the same block. Thus, if the pair (q_1, q_2) processed at the $(t+1)$th iteration is a pair of final states placed in LIST at the initialization step (line 3), then we must have $B(q_1, \pi') = B(q_2, \pi')$. Otherwise, (q_1, q_2) was placed in LIST as a pair of successor or predecessor states of two states q_1' and q_2' merged at a previous iteration $s \leq t$. Since π_s refines π', q_1' and q_2' are in the same block of π' and since A/π' is reversible, q_1 and q_2 must also be in the same block as successors or predecessors of the same block for the same label $a \in \Sigma$, thus $B(q_1, \pi') = B(q_2, \pi')$. □

Theorem 16.10 *Let S be a finite set of strings and let A be the automaton returned by* LearnReversibleAutomata() *when used with input S. Then, $L(A)$ is the smallest reversible language containing S.*

Proof: Let L be a reversible language containing S, and let A' be a reversible automaton with $L(A') = L$. Since every string of S is accepted by A', any $u \in \text{Pref}(S)$ can be read from the initial state of A' to reach some state $q(u)$ of A'. Consider the automaton A'' derived from A' by keeping only states of the form $q(u)$ and transitions between such states. A'' has the unique final state of A' since $q(u)$ is final for $u \in S$, and it has the initial state of A', since ϵ is a prefix of strings of S. Furthermore, A'' directly inherits from A' the property of being deterministic and reverse deterministic. Thus, A'' is reversible.

The states of A'' define a partition of $\text{Pref}(S)$: $u, v \in \text{Pref}(S)$ are in the same block iff $q(u) = q(v)$. Since by definition of the prefix-tree $PT(S)$, its states can be identified with $\text{Pref}(S)$, the states of A'' also define a partition π' of the states of $PT(S)$ and thus $A'' = PT(S)/\pi'$. By proposition 16.9, the partition π defined by algorithm LearnReversibleAutomata() run with input S is the finest such that $PT(S)/\pi$ is reversible. Therefore, we must have $L(PT(S)/\pi) \subseteq L(PT(S)/\pi') = L(A'')$. Since A'' is a sub-automaton of A', L contains $L(A'')$ and therefore $L(PT(S)/\pi) = L(A)$, which concludes the proof. □

Theorem 16.11 (Identification in the limit of reversible languages) *Let L be a reversible language, then algorithm* LearnReversibleAutomata() *identifies L in the limit from a positive presentation.*

Proof: Let L be a reversible language. By proposition 16.8, L admits a finite characteristic sample S_L. Let $(x_n)_{n \in \mathbb{N}}$ be a positive presentation of L and let $\mathcal{X}_n$ denote the union of the first n elements of the sequence. Since S_L is finite, there exists $N \geq 1$ such that $S_L \subseteq \mathcal{X}_N$. By theorem 16.10, for any $n \geq N$, LearnReversibleAutomata() run on the finite sample $\mathcal{X}_n$ returns the smallest reversible language L' containing $\mathcal{X}_n$ a fortiori S_L, which, by definition of S_L, implies that $L' = L$. $\square$

The main operations needed for the implementation of the algorithm for learning reversible automata are the standard FIND and UNION to determine the block a state belongs to and to merge two blocks into a single one. Using a disjoint-set data structure for these operations, the time complexity of the algorithm can be shown to be in $O(n\alpha(n))$, where n denotes the sum of the lengths of all strings in the input sample S and $\alpha(n)$ the inverse of the Ackermann function, which is essentially constant ($\alpha(n) \leq 4$ for $n \leq 10^{80}$).

16.5 Chapter notes

For an overview of finite automata and some related results, see Hopcroft and Ullman [1979] or the more recent Handbook chapter by Perrin [1990], as well as the series of books by M. Lothaire [Lothaire, 1982, 1990, 2005] and the even more recent book by De la Higuera [2010].

Theorem 16.2, stating that the problem of finding a minimum consistent DFA is NP-hard, is due to Gold [1978]. This result was later extended by Angluin [1978]. Pitt and Warmuth [1993] further strengthened these results by showing that even an approximation within a polynomial function of the size of the smallest automaton is NP-hard (theorem 16.3). Their hardness results apply also to the case where prediction is made using NFAs. Kearns and Valiant [1994] presented hardness results of a different nature relying on cryptographic assumptions. Their results imply that no polynomial-time algorithm can learn consistent NFAs polynomial in the size of the smallest DFA from a finite sample of accepted and rejected strings if any of the generally accepted cryptographic assumptions holds: if factoring Blum integers is hard; or if the RSA public key cryptosystem is secure; or if deciding quadratic residuosity is hard. Most recently, Chalermsook et al. [2014] improved the non-approximation guarantee of Pitt and Warmuth [1993] to a tight bound.

On the positive side, Trakhtenbrot and Barzdin [1973] showed that the smallest finite automaton consistent with the input data can be learned exactly from a uniform complete sample, whose size is exponential in the size of the automaton. The worst-case complexity of their algorithm is exponential, but a better average-case complexity can be obtained assuming that the topology and the labeling are selected randomly [Trakhtenbrot and Barzdin, 1973] or even that the topology is selected adversarially [Freund et al., 1993].

Cortes, Kontorovich, and Mohri [2007a] study an approach to the problem of learning automata based on linear separation in some appropriate high-dimensional feature space; see also Kontorovich et al. [2006, 2008]. The mapping of strings to that feature space can be defined implicitly using the rational kernels presented in chapter 6, which are themselves defined via weighted automata and transducers.

The model of learning with queries was introduced by Angluin [1978], who also proved that finite automata can be learned in time polynomial in the size of the minimal automaton and that of the longest counter-example. Bergadano and Varricchio [1995] further extended this result to the problem of learning weighted automata defined over any field (see also an optimal algorithm by Bisht et al. [2006]). Using the relationship between the size of a minimal weighted automaton over a field and the rank of the corresponding Hankel matrix, the learnability of many other concepts classes such as disjoint DNF can be shown [Beimel et al., 2000]. Our description of an efficient implementation of the algorithm of Angluin [1982] using decision trees is adapted from Kearns and Vazirani [1994].

The model of identification in the limit of automata was introduced and analyzed by Gold [1967]. Deterministic finite automata were shown not to be identifiable in the limit from positive examples [Gold, 1967]. But, positive results were given for the identification in the limit of a number of sub-classes, such as the family of k-reversible languages Angluin [1982] considered in this chapter. Positive results also hold for learning subsequential transducers Oncina et al. [1993]. Some restricted classes of probabilistic automata such as acyclic probabilistic automata were also shown by Ron et al. [1995] to be efficiently learnable.

There is a vast literature dealing with the problem of learning automata. In particular, positive results have been shown for a variety of sub-families of finite automata in the scenario of learning with queries and learning scenarios of different kinds have been introduced and analyzed for this problem. The results presented in this chapter should therefore be viewed only as an introduction to that material.

16.6 Exercises

16.1 Minimal DFA. Show that a minimal DFA A also has the minimal number of transitions among all other DFAs equivalent to A. Prove that a language L is regular iff $Q = \{\mathrm{Suff}_L(u) \colon u \in \Sigma^*\}$ is finite. Show that the number of states of a minimal DFA A with $L(A) = L$ is precisely the cardinality of Q.

16.2 VC-dimension of finite automata.

(a) What is the VC-dimension of the family of all finite automata? What does that imply for PAC-learning of finite automata? Does this result change if we restrict ourselves to learning acyclic automata (automata with no cycles)?

(b) Show that the VC-dimension of the family of DFAs with at most n states is bounded by $O(|\Sigma| n \log n)$.

16.3 PAC learning with membership queries. Give an example of a concept class $\mathcal{C}$ that is efficiently PAC-learnable with membership queries but that is not efficiently exactly learnable.

16.4 Learning monotone DNF formulae with queries. Show that the class of monotone DNF formulae over n variables is efficiently exactly learnable using membership and equivalence queries. (*Hint*: a *prime implicant* t of a formula f is a product of literals such that t implies f but no proper sub-term of t implies f. Use the fact that for monotone DNF, the number of prime implicants is at the most the number of terms of the formula.)

16.5 Learning with unreliable query responses. Consider the problem where the learner must find an integer x selected by the oracle within $[n]$, where $n \geq 1$ is given. To do so, the learner can ask questions of the form $(x \leq m?)$ or $(x > m?)$ for $m \in [n]$. The oracle responds to these questions but may give an incorrect response to k questions. How many questions should the learner ask to determine x? (*Hint*: observe that the learner can repeat each question $2k+1$ times and use the majority vote.)

16.6 Algorithm for learning reversible languages. What is the DFA A returned by the algorithm for learning reversible languages when applied to the sample $S = \{ab, aaabb, aabbb, aabbbb\}$? Suppose we add a new string to the sample, say $x = abab$. How should A be updated to compute the result of the algorithm for $S \cup \{x\}$? More generally, describe a method for updating the result of the algorithm incrementally.

16.7 k-reversible languages. A finite automaton A' is said to be *k-deterministic* if it is deterministic modulo a lookahead k: if two distinct states p and q are both initial, or are both reached from another state r by reading $a \in \Sigma$, then no string u of length k can be read in A' both from p and q. A finite automaton A is said to be *k-reversible* if it is deterministic and if A^R is k-deterministic. A language L is *k-reversible* if it is accepted by some k-reversible automaton.

(a) Prove that L is k-reversible iff for any strings $u, u', v \in \Sigma^*$ with $|v| = k$,

$$\mathrm{Suff}_L(uv) \cap \mathrm{Suff}_L(u'v) \neq \emptyset \implies \mathrm{Suff}_L(uv) = \mathrm{Suff}_L(u'v).$$

(b) Show that a k-reversible language admits a characteristic language.

(c) Show that the following defines an algorithm for learning k-reversible automata. Proceed as in the algorithm for learning reversible automata but with the following merging rule instead: merge blocks B_1 and B_2 if they can be reached by the same string u of length k from some other block and if B_1 and B_2 are both final or have a common successor.

17 Reinforcement Learning

This chapter presents an introduction to reinforcement learning, a rich area of machine learning with connections to control theory, optimization, and cognitive sciences. Reinforcement learning is the study of planning and learning in a scenario where a learner actively interacts with the environment to achieve a certain goal. This active interaction justifies the terminology of *agent* used to refer to the learner. The achievement of the agent's goal is typically measured by the reward it receives from the environment and which it seeks to maximize.

We first introduce the general scenario of reinforcement learning and then introduce the model of Markov decision processes (MDPs), which is widely adopted in this area, as well as essential concepts such as that of *policy* or *policy value* related to this model. The rest of the chapter presents several algorithms for the *planning* problem, which corresponds to the case where the environment model is known to the agent, and then a series of *learning* algorithms for the more general case of an unknown model.

17.1 Learning scenario

The general scenario of reinforcement learning is illustrated by figure 17.1. Unlike the supervised learning scenario considered in previous chapters, here, the learner does not passively receive a labeled data set. Instead, it collects information through a course of *actions* by interacting with the *environment*. In response to an action, the learner or *agent*, receives two types of information: its current *state* in the environment, and a real-valued *reward*, which is specific to the task and its corresponding goal.

The objective of the agent is to maximize its reward and thus to determine the best course of actions, or *policy*, to achieve that objective. However, the information he receives from the environment is only the immediate reward related to the action just taken. No future or long-term reward feedback is provided by the environment. An important aspect of reinforcement learning is to consider delayed rewards or

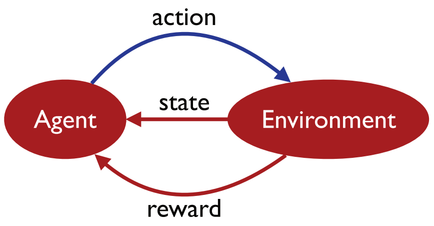


Figure 17.1
Representation of the general scenario of reinforcement learning.

penalties. The agent is faced with the dilemma between exploring unknown states and actions to gain more information about the environment and the rewards, and exploiting the information already collected to optimize its reward. This is known as the *exploration versus exploitation trade-off* inherent to reinforcement learning.

Note that there are several differences between the learning scenario of reinforcement learning and that of supervised learning examined in most of the previous chapters. Unlike supervised learning, in reinforcement learning there is no fixed distribution according to which instances are drawn; it is the choice of a policy that defines the distribution over observations. In fact, slight changes to the policy may have dramatic effects on the rewards received. Furthermore, in general, the environment may not be fixed and could vary as a result of the actions selected by the agent. This may be a more realistic model for some learning problems than the standard supervised learning. Finally, note that, unlike supervised learning, in reinforcement learning, training and testing phases are intermixed.

Two main settings can be distinguished here: the one where the environment model is known to the agent, in which case its objective of maximizing the reward received is reduced to a *planning problem*; and the one where the environment model is unknown, in which case the agent faces a *learning problem*. In the latter setting, the agent must learn from the state and reward information gathered to both gain information about the environment and determine the best action policy. This chapter presents algorithmic solutions for both of these settings.

17.2 Markov decision process model

We first introduce the model of Markov decision processes (MDPs), a model of the environment and interactions with the environment widely adopted in reinforcement learning. An MDP is a Markovian process defined as follows.

Definition 17.1 (MDPs) *A Markov decision process (MDP) is defined by:*

- *a set of* states S*, possibly infinite.*

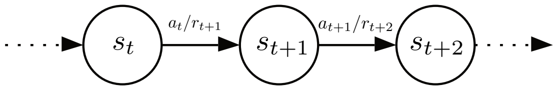


Figure 17.2
Illustration of the states and transitions of an MDP at different times.

- *a* start state *or* initial state $s_0 \in S$.
- *a set of* actions A, *possibly infinite.*
- *a* transition probability $\mathbb{P}[s'|s, a]$*: distribution over destination states* $s' = \delta(s, a)$.
- *a* reward probability $\mathbb{P}[r'|s, a]$*: distribution over rewards returned* $r' = r(s, a)$.

The model is Markovian because the transition and reward probabilities depend only on the current state s and not the entire history of states and actions taken. This definition of MDP can be further generalized to the case of non-discrete state and action sets.

In a discrete-time model, actions are taken at a set of *decision epochs* $\{0, \ldots, T\}$, and this is the model we will adopt in what follows. This model can also be straightforwardly generalized to a continuous-time one where actions are taken at arbitrary points in time.

When T is finite, the MDP is said to have a *finite horizon*. Independently of the finiteness of the time horizon, an MDP is said to be *finite* when both S and A are finite sets. Here, we are considering the general case where the reward $r(s, a)$ at state s when taking action a is a random variable. However, in many cases, the reward is assumed to be a deterministic function the state and action pair (s, a).

Figure 17.2 illustrates the model corresponding to an MDP. At time $t \in \{0, \ldots, T\}$ the state observed by the agent is s_t and it takes action $a_t \in A$. The state reached is s_{t+1} (with probability $\mathbb{P}[s_{t+1}|s_t, a_t]$) and the reward received $r_{t+1} \in \mathbb{R}$ (with probability $\mathbb{P}[r_{t+1}|s_t, a_t]$).

Many real-world tasks can be represented by MDPs. Figure 17.3 gives the example of a simple MDP for a robot picking up balls on a tennis court.

17.3 Policy

The main problem for an agent in an MDP environment is to determine the action to take at each state, that is, an action *policy*.

17.3.1 Definition

Definition 17.2 (Policy) *A* policy *is a mapping* $\pi \colon S \to \Delta(A)$*, where* $\Delta(A)$ *is the set of probability distributions over* A*. A policy* π *is* deterministic *if for any* s*, there exists a unique* $a \in A$ *such that* $\pi(s)(a) = 1$*. In that case, we can identify* π *with a mapping from* S *to* A *and use* $\pi(s)$ *to denote that action.*

More precisely, this is the definition of a *stationary policy* since the choice of the distribution of actions does not depend on time. More generally, we could define a *non-stationary policy* as a sequence of mappings $\pi_t \colon S \to \Delta(A)$ indexed by t. In particular, in the finite horizon case, a non-stationary policy is typically necessary for optimizing rewards.

The agent's objective is to find a policy that maximizes its expected (reward) *return*. The return it receives following a deterministic policy π along a specific sequence of states $s_0, \ldots, s_T$ is defined as follows:

- for a finite horizon ($T < \infty$): $\sum_{t=0}^{T} r\big(s_t, \pi(s_t)\big)$.
- for an infinite horizon ($T = \infty$): $\sum_{t=0}^{+\infty} \gamma^t r\big(s_t, \pi(s_t)\big)$, where $\gamma \in [0, 1)$ is a constant factor less than one used to discount future rewards.

Note that the return is a single scalar summarizing a possibly infinite sequence of immediate rewards. In the discounted case, early rewards are viewed as more valuable than later ones.

17.3.2 Policy value

This leads to the following definition of the value of a policy at each state.

Definition 17.3 (Policy value) *The* value $V_\pi(s)$ *of a policy* π *at state* $s \in S$ *is defined as the expected reward returned when starting at* s *and following policy* π*:*

- *finite horizon:* $V_\pi(s) = \mathbb{E}_{a_t \sim \pi(s_t)} \Big[\sum_{t=0}^{T} r\big(s_t, a_t\big) \,\Big|\, s_0 = s\Big]$;
- *infinite discounted horizon:* $V_\pi(s) = \mathbb{E}_{a_t \sim \pi(s_t)} \Big[\sum_{t=0}^{+\infty} \gamma^t r\big(s_t, a_t\big) \,\Big|\, s_0 = s\Big]$,

where the expectations are over the random selection of an action a_t *according to the distribution* $\pi(s_t)$*, which is explicitly indicated, and over the random states* s_t *reached and the reward values* $r\big(s_t, a_t\big)$*.*[22] *An infinite undiscounted horizon is also often considered based on the limit of the average reward, when it exists.*

17.3.3 Optimal policies

Starting from a state $s \in S$, to maximize its reward, an agent naturally seeks a policy π with the largest value $V_\pi(s)$. In this section, we will show that, remarkably, for any finite MDP in the infinite horizon setting, there exists a policy that is *optimal* for *any* start state, that is one with the following definition.

Definition 17.4 (Optimal policy) *A policy* π^* *is* optimal *if its value is maximal for every state* $s \in S$*, that is, for any policy* π *and any state* $s \in S$*,* $V_{\pi^*}(s) \geq V_\pi(s)$*.*

[22] More generally, in all that follows, the randomization with respect to the reward function and the next state will not be explicitly indicated to simplify the notation.

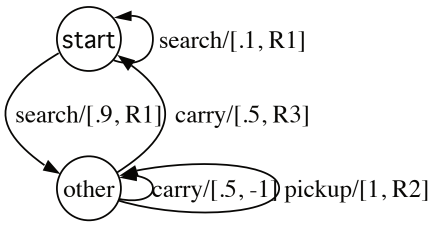


Figure 17.3
Example of a simple MDP for a robot picking up balls on a tennis court. The set of actions is $A = \{search, carry, pickup\}$ and the set of states reduced to $S = \{start, other\}$. Each transition is labeled with the action followed by the probability of the transition probability and the reward received after taking that action. R_1, R_2, and R_3 are real numbers indicating the reward associated to each transition (case of deterministic reward).

Moreover, we will show that for any MDP there exists a deterministic optimal policy. To do so, it is convenient to introduce the notion of *state-action value function.*

Definition 17.5 (State-action value function) *The* state-action value function *Q associated to a policy π is defined for all $(s, a) \in S \times A$ as the expected return for taking action $a \in A$ at state $s \in S$ and then following policy π:*

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}[r(s, a)] + \mathop{\mathbb{E}}_{a_t \sim \pi(s_t)} \left[\sum_{t=1}^{+\infty} \gamma^t r\big(s_t, a_t\big) \;\Big|\; s_0 = s, a_0 = a \right] \qquad (17.1) \\ &= \mathbb{E}\left[r(s, a) + \gamma V_\pi(s_1) \;\Big|\; s_0 = s, a_0 = a \right]. \end{aligned}$$

Observe that $\mathbb{E}_{a \sim \pi(s)} \big[Q_\pi(s, a)\big] = V_\pi(s)$ (see also proposition 17.9)

Theorem 17.6 (Policy improvement theorem) *For any two policies π and π' the following holds:*

$$\Big(\forall s \in S, \mathop{\mathbb{E}}_{a \sim \pi'(s)} \big[Q_\pi(s, a)\big] \geq \mathop{\mathbb{E}}_{a \sim \pi(s)} \big[Q_\pi(s, a)\big]\Big) \Rightarrow \Big(\forall s \in S, V_{\pi'}(s) \geq V_\pi(s)\Big).$$

Furthermore, a strict inequality for at least one state s in the left-hand side implies a strict inequality for at least one s in the right-hand side.

Proof: Assume that π and π' verify the left-hand side. For any $s \in S$, we have

$$\begin{aligned}
V_\pi(s) &= \mathop{\mathbb{E}}_{a \sim \pi(s)} \big[Q_\pi(s,a)\big] \\
&\leq \mathop{\mathbb{E}}_{a \sim \pi'(s)} \big[Q_\pi(s,a)\big] \\
&= \mathop{\mathbb{E}}_{a \sim \pi'(s)} \Big[r(s,a) + \gamma V_\pi(s_1) \,\Big|\, s_0 = s\Big] \\
&= \mathop{\mathbb{E}}_{a \sim \pi'(s)} \Big[r(s,a) + \gamma \mathop{\mathbb{E}}_{a_1 \sim \pi(s_1)} \big[Q_\pi(s_1,a_1)\big] \,\Big|\, s_0 = s\Big] \\
&\leq \mathop{\mathbb{E}}_{a \sim \pi'(s)} \Big[r(s,a) + \gamma \mathop{\mathbb{E}}_{a_1 \sim \pi'(s_1)} \big[Q_\pi(s_1,a_1)\big] \,\Big|\, s_0 = s\Big] \\
&= \mathop{\mathbb{E}}_{\substack{a \sim \pi'(s) \\ a_1 \sim \pi'(s_1)}} \Big[r(s,a) + \gamma r(s_1,a_1) + \gamma^2 V_\pi(s_2) \,\Big|\, s_0 = s\Big].
\end{aligned}$$

Proceeding in this way shows that for any $T \geq 1$:

$$V_\pi(s) \leq \mathop{\mathbb{E}}_{a_t \sim \pi'(s_t)} \left[\sum_{t=0}^{T} \gamma^t \, \mathbb{E}[r(s_t,a_t)] + \gamma^{T+1} V_\pi(s_{T+1}) \,\Big|\, s_0 = s\right].$$

Since $V_\pi(s_{T+1})$ is bounded, taking the limit $T \to +\infty$ gives

$$V_\pi(s) \leq \mathop{\mathbb{E}}_{a_t \sim \pi'(s_t)} \left[\sum_{t=0}^{+\infty} \gamma^t \, \mathbb{E}[r(s_t,a_t)] \,\Big|\, s_0 = s\right] = V_{\pi'}(s).$$

Finally, any strict inequality in the left-hand side property results in a strict inequality in the chain of inequalities above. □

Theorem 17.7 (Bellman's optimality condition) *A policy π is optimal iff for any pair $(s,a) \in S \times A$ with $\pi(s)(a) > 0$ the following holds:*

$$a \in \operatorname*{argmax}_{a' \in A} Q_\pi(s,a'). \tag{17.2}$$

Proof: By Theorem 17.6, if the condition (17.2) does not hold for some (s,a) with $\pi(s)(a) > 0$, then the policy π is not optimal. This is because π can then be improved by defining π' such that $\pi'(s') = \pi(s)$ for $s' \neq s$ and $\pi'(s)$ concentrated on any element of $\operatorname{argmax}_{a' \in A} Q_\pi(s,a')$. π' verifies $\mathbb{E}_{a \sim \pi'(s)}\big[Q_\pi(s',a)\big] = \mathbb{E}_{a \sim \pi(s)}\big[Q_\pi(s',a)\big]$ for $s' \neq s$ and $\mathbb{E}_{a \sim \pi'(s)}\big[Q_\pi(s,a)\big] > \mathbb{E}_{a \sim \pi(s)}\big[Q_\pi(s,a)\big]$. Thus, by Theorem 17.6, $V_{\pi'}(s) > V_\pi(s)$ for at least one s and π is not optimal.

Conversely, let π' be a non-optimal policy. Then there exists a policy π and at least one state s for which $V_{\pi'}(s) < V_\pi(s)$. By Theorem 17.6, this implies that there exists some state $s \in S$ with $\mathbb{E}_{a \sim \pi'(s)}\big[Q_\pi(s,a)\big] < \mathbb{E}_{a \sim \pi(s)}\big[Q_\pi(s,a)\big]$. Thus, π' cannot satisfy the condition (17.2). □

Theorem 17.8 (Existence of an optimal deterministic policy) *Any finite MDP admits an optimal deterministic policy.*

Proof: Let π^* be a deterministic policy maximizing $\sum_{s\in S} V_\pi(s)$. π^* exists since there are only finitely many deterministic policies. If π^* were not optimal, by Theorem 17.7, there would exist a state s with $\pi(s) \not\in \operatorname{argmax}_{a'\in A} Q_\pi(s,a')$. By theorem 17.6, π^* could then be improved by choosing a policy π with $\pi(s) \in \operatorname{argmax}_{a'\in A} Q_\pi(s,a')$ and π coinciding with π^* for all other states. But then π would verify $V_{\pi*}(s) \leq V_\pi(s)$ with a strict inequality at least for one state. This would contradict the fact that π^* maximizes $\sum_{s\in S} V_\pi(s)$. $\square$

In view of the existence of a deterministic optimal policy, in what follows, to simplify the discussion, we will consider only deterministic policies. Let π^* denote a (deterministic) optimal policy, and let Q^* and V^* denote its corresponding state-action value function and value function. By Theorem 17.7, we can write

$$\forall s \in S,\ \pi^*(s) = \operatorname*{argmax}_{a\in A} Q^*(s,a). \tag{17.3}$$

Thus, the knowledge of the state-action value function Q^* is sufficient for the agent to determine the optimal policy, without any direct knowledge of the reward or transition probabilities. Replacing Q^* by its definition gives the following system of equations for the optimal policy values $V^*(s) = Q^*(s, \pi^*(s))$:

$$\forall s \in S,\ V^*(s) = \max_{a\in A} \Big\{ \mathbb{E}[r(s,a)] + \gamma \sum_{s'\in S} \mathbb{P}[s'|s,a]V^*(s') \Big\}, \tag{17.4}$$

also known as *Bellman equations*. Note that this system of equations is not linear due to the presence of the max operator.

17.3.4 Policy evaluation

The value of a policy at state s can be expressed in terms of its values at other states, forming a system of linear equations.

Proposition 17.9 (*Bellman equations*) *The values $V_\pi(s)$ of policy π at states $s \in S$ for an infinite horizon MDP obey the following system of linear equations:*

$$\forall s \in S,\ V_\pi(s) = \mathop{\mathbb{E}}_{a_1\sim\pi(s)}[r(s,a_1)] + \gamma \sum_{s'} \mathbb{P}[s'|s,\pi(s)]V_\pi(s'). \tag{17.5}$$

Proof: We can decompose the expression of the policy value as a sum of the first term and the rest of the terms, which admit γ as a multiplier:

$$\begin{aligned}
V_\pi(s) &= \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t r\big(s_t, \pi(s_t)\big) \,\middle|\, s_0 = s\right].\\
&= \mathbb{E}[r(s, \pi(s))] + \gamma\, \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t r\big(s_{t+1}, \pi(s_{t+1})\big) \,\middle|\, s_0 = s\right]\\
&= \mathbb{E}[r(s, \pi(s))] + \gamma\, \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t r\big(s_{t+1}, \pi(s_{t+1})\big) \,\middle|\, s_1 = \delta(s, \pi(s))\right]\\
&= \mathbb{E}[r(s, \pi(s)] + \gamma\, \mathbb{E}[V_\pi(\delta(s, \pi(s)))].
\end{aligned}$$

This completes the proof. □

This a linear system of equations, also known as Bellman equations, that is distinct from the non-linear system (17.4). The system can be rewritten as

$$\mathbf{V} = \mathbf{R} + \gamma \mathbf{P} \mathbf{V}, \tag{17.6}$$

using the following notation: $\mathbf{P}$ denotes the transition probability matrix defined by $\mathbf{P}_{s,s'} = \mathbb{P}[s'|s, \pi(s)]$ for all $s, s' \in S$; $\mathbf{V}$ is the value column matrix whose sth component is $\mathbf{V}_s = V_\pi(s)$; and $\mathbf{R}$ the reward column matrix whose sth component is $\mathbf{R}_s = \mathbb{E}[r(s, \pi(s)]$. $\mathbf{V}$ is typically the unknown variable in the Bellman equations and is determined by solving for it.

The following theorem shows that, for a finite MDP, this system of linear equations admits a unique solution.

Theorem 17.10 *For a finite MDP, Bellman's equations admit a unique solution given by*

$$\mathbf{V}_0 = (\mathbf{I} - \gamma \mathbf{P})^{-1} \mathbf{R}. \tag{17.7}$$

Proof: The Bellman equations (17.6) can be equivalently written as

$$(\mathbf{I} - \gamma \mathbf{P})\mathbf{V} = \mathbf{R}.$$

Thus, to prove the theorem it suffices to show that $(\mathbf{I} - \gamma \mathbf{P})$ is invertible. To do so, note that the infinity of $\mathbf{P}$ can be computed using its stochasticity properties:

$$\|\mathbf{P}\|_\infty = \max_s \sum_{s'} |\mathbf{P}_{ss'}| = \max_s \sum_{s'} \mathbb{P}[s'|s, \pi(s)] = 1.$$

This implies that $\|\gamma \mathbf{P}\|_\infty = \gamma < 1$. The eigenvalues of $\gamma \mathbf{P}$ are thus all less than one, and $(\mathbf{I} - \gamma \mathbf{P})$ is invertible. □

Thus, for a finite MDP, when the transition probability matrix $\mathbf{P}$ and the reward expectations $\mathbf{R}$ are known, the value of policy π at all states can be determined by inverting a matrix.

17.4 Planning algorithms

In this section, we assume that the environment model is known. That is, the transition probability $\mathbb{P}[s'|s,a]$ and the expected reward $\mathbb{E}[r(s,a)]$ for all $s, s' \in S$ and $a \in A$ are assumed to be given. The problem of finding the optimal policy then does not require learning the parameters of the environment model or estimating other quantities helpful in determining the best course of actions, it is purely a *planning* problem.

This section discusses three algorithms for this planning problem: the value iteration algorithm, the policy iteration algorithm, and a linear programming formulation of the problem.

17.4.1 Value iteration

The *value iteration algorithm* seeks to determine the optimal policy values $V^*(s)$ at each state $s \in S$, and thereby the optimal policy. The algorithm is based on the Bellman equations (17.4). As already indicated, these equations do not form a system of linear equations and require a different technique to determine the solution. The main idea behind the design of the algorithm is to use an iterative method to solve them: the new values of $V(s)$ are determined using the Bellman equations and the current values. This process is repeated until a convergence condition is met.

For a vector $\mathbf{V}$ in $\mathbb{R}^{|S|}$, we denote by $V(s)$ its sth coordinate, for any $s \in S$. Let $\mathbf{\Phi}\colon \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ be the mapping defined based on Bellman's equations (17.4):

$$\forall s \in S, [\mathbf{\Phi}(\mathbf{V})](s) = \max_{a \in A} \Big\{ \mathbb{E}[r(s,a)] + \gamma \sum_{s' \in S} \mathbb{P}[s'|s,a] V(s') \Big\}. \tag{17.8}$$

The maximizing actions $a \in A$ in these equations define an action to take at each state $s \in S$, that is a policy π. We can thus rewrite these equations in matrix terms as follows:

$$\mathbf{\Phi}(\mathbf{V}) = \max_{\pi} \{\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V}\}, \tag{17.9}$$

where $\mathbf{P}_\pi$ is the transition probability matrix defined by $(\mathbf{P}_\pi)_{ss'} = \mathbb{P}[s'|s,\pi(s)]$ for all $s, s' \in S$, and $\mathbf{R}_\pi$ the reward vector defined by $(\mathbf{R}_\pi)_s = \mathbb{E}[r(s,\pi(s)]$, for all $s \in S$.

The algorithm is directly based on (17.9). The pseudocode is given above. Starting from an arbitrary policy value vector $\mathbf{V}_0 \in \mathbb{R}^{|S|}$, the algorithm iteratively applies

```
ValueIteration($\mathbf{V}_0$)
1  $\mathbf{V} \leftarrow \mathbf{V}_0$    $\triangleright$ $\mathbf{V}_0$ arbitrary value
2  while $\|\mathbf{V} - \mathbf{\Phi}(\mathbf{V})\| \geq \frac{(1-\gamma)\epsilon}{\gamma}$ do
3          $\mathbf{V} \leftarrow \mathbf{\Phi}(\mathbf{V})$
4  return $\mathbf{\Phi}(\mathbf{V})$
```

Figure 17.4
Value iteration algorithm.

$\mathbf{\Phi}$ to the current $\mathbf{V}$ to obtain a new policy value vector until $\|\mathbf{V} - \mathbf{\Phi}(\mathbf{V})\| < \frac{(1-\gamma)\epsilon}{\gamma}$, where $\epsilon > 0$ is a desired approximation. The following theorem proves the convergence of the algorithm to the optimal policy values.

Theorem 17.11 *For any initial value $\mathbf{V}_0$, the sequence defined by $\mathbf{V}_{n+1} = \mathbf{\Phi}(\mathbf{V}_n)$ converges to $\mathbf{V}^*$.*

Proof: We first show that $\mathbf{\Phi}$ is γ-Lipschitz for the $\|\cdot\|_\infty$.[23] For any $s \in S$ and $\mathbf{V} \in \mathbb{R}^{|S|}$, let $a^*(s)$ be the maximizing action defining $\mathbf{\Phi}(\mathbf{V})(s)$ in (17.8). Then, for any $s \in S$ and any $\mathbf{U} \in \mathbb{R}^{|S|}$,

$$\begin{aligned}
\mathbf{\Phi}(\mathbf{V})(s) - \mathbf{\Phi}(\mathbf{U})(s) &\leq \mathbf{\Phi}(\mathbf{V})(s) - \Big(\mathbb{E}[r(s, a^*(s))] + \gamma \sum_{s' \in S} \mathbb{P}[s' \mid s, a^*(s)]\mathbf{U}(s')\Big) \\
&= \gamma \sum_{s' \in S} \mathbb{P}[s'|s, a^*(s)][\mathbf{V}(s') - \mathbf{U}(s')] \\
&\leq \gamma \sum_{s' \in S} \mathbb{P}[s'|s, a^*(s)]\|\mathbf{V} - \mathbf{U}\|_\infty = \gamma\|\mathbf{V} - \mathbf{U}\|_\infty.
\end{aligned}$$

Proceeding similarly with $\mathbf{\Phi}(\mathbf{U})(s) - \mathbf{\Phi}(\mathbf{V})(s)$, we obtain $\mathbf{\Phi}(\mathbf{U})(s) - \mathbf{\Phi}(\mathbf{V})(s) \leq \gamma\|\mathbf{V} - \mathbf{U}\|_\infty$. Thus, $|\mathbf{\Phi}(\mathbf{V})(s) - \mathbf{\Phi}(\mathbf{U})(s)| \leq \gamma\|\mathbf{V} - \mathbf{U}\|_\infty$ for all s, which implies

$$\|\mathbf{\Phi}(\mathbf{V}) - \mathbf{\Phi}(\mathbf{U})\|_\infty \leq \gamma\|\mathbf{V} - \mathbf{U}\|_\infty,$$

[23] A β-Lipschitz function with $\beta < 1$ is also called *β-contracting*. In a *complete metric space*, that is a metric space where any Cauchy sequence converges to a point of that space, a β-contracting function f admits a *fixed point*: any sequence $(f(x_n))_{n \in \mathbb{N}}$ converges to some x with $f(x) = x$. $\mathbb{R}^N$, $N \geq 1$, or, more generally, any finite-dimensional vector space, is a complete metric space.

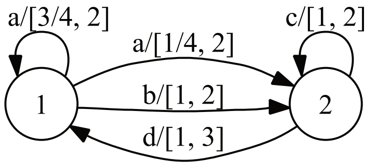


Figure 17.5
Example of MDP with two states. The state set is reduced to $S = \{1, 2\}$ and the action set to $A = \{a, b, c, d\}$. Only transitions with non-zero probabilities are represented. Each transition is labeled with the action taken followed by a pair $[p, r]$ after a slash separator, where p is the probability of the transition and r the expected reward for taking that transition.

that is the γ-Lipschitz property of $\mathbf{\Phi}$. Now, by Bellman equations (17.4), $\mathbf{V}^* = \mathbf{\Phi}(\mathbf{V}^*)$, thus for any $n \in \mathbb{N}$,

$$\|\mathbf{V}^* - \mathbf{V}_{n+1}\|_\infty = \|\mathbf{\Phi}(\mathbf{V}^*) - \mathbf{\Phi}(\mathbf{V}_n)\|_\infty \leq \gamma\|\mathbf{V}^* - \mathbf{V}_n\|_\infty \leq \gamma^{n+1}\|\mathbf{V}^* - \mathbf{V}_0\|_\infty,$$

which proves the convergence of the sequence to $\mathbf{V}^*$ since $\gamma \in (0, 1)$. □

The ϵ-optimality of the value returned by the algorithm can be shown as follows. By the triangle inequality and the γ-Lipschitz property of $\mathbf{\Phi}$, for any $n \in \mathbb{N}$,

$$\begin{aligned}\|\mathbf{V}^* - \mathbf{V}_{n+1}\|_\infty &\leq \|\mathbf{V}^* - \mathbf{\Phi}(\mathbf{V}_{n+1})\|_\infty + \|\mathbf{\Phi}(\mathbf{V}_{n+1}) - \mathbf{V}_{n+1}\|_\infty \\ &= \|\mathbf{\Phi}(\mathbf{V}^*) - \mathbf{\Phi}(\mathbf{V}_{n+1})\|_\infty + \|\mathbf{\Phi}(\mathbf{V}_{n+1}) - \mathbf{\Phi}(\mathbf{V}_n)\|_\infty \\ &\leq \gamma\|\mathbf{V}^* - \mathbf{V}_{n+1}\|_\infty + \gamma\|\mathbf{V}_{n+1} - \mathbf{V}_n\|_\infty.\end{aligned}$$

Thus, if $\mathbf{V}_{n+1}$ is the policy value returned by the algorithm, we have

$$\|\mathbf{V}^* - \mathbf{V}_{n+1}\|_\infty \leq \frac{\gamma}{1-\gamma}\|\mathbf{V}_{n+1} - \mathbf{V}_n\|_\infty \leq \epsilon.$$

The convergence of the algorithm is in $O(\log \frac{1}{\epsilon})$ number of iterations. Indeed, observe that

$$\|\mathbf{V}_{n+1} - \mathbf{V}_n\|_\infty = \|\mathbf{\Phi}(\mathbf{V}_n) - \mathbf{\Phi}(\mathbf{V}_{n-1})\|_\infty \leq \gamma\|\mathbf{V}_n - \mathbf{V}_{n-1}\|_\infty \leq \gamma^n\|\mathbf{\Phi}(\mathbf{V}_0) - \mathbf{V}_0\|_\infty.$$

Thus, if n is the largest integer such that $\frac{(1-\gamma)\epsilon}{\gamma} \leq \|\mathbf{V}_{n+1} - \mathbf{V}_n\|_\infty$, it must verify $\frac{(1-\gamma)\epsilon}{\gamma} \leq \gamma^n\|\mathbf{\Phi}(\mathbf{V}_0) - \mathbf{V}_0\|_\infty$ and therefore $n \leq O\big(\log \frac{1}{\epsilon}\big)$.[24]

[24] Here, the O-notation hides the dependency on the discount factor γ. As a function of γ, the running time is not polynomial.

```
PolicyIteration($\pi_0$)
1  $\pi \leftarrow \pi_0$    ▷ $\pi_0$ arbitrary policy
2  $\pi' \leftarrow$ NIL
3  while ($\pi \neq \pi'$) do
4      $\mathbf{V} \leftarrow \mathbf{V}_\pi$    ▷ policy evaluation: solve $(\mathbf{I} - \gamma \mathbf{P}_\pi)\mathbf{V} = \mathbf{R}_\pi$.
5      $\pi' \leftarrow \pi$
6      $\pi \leftarrow \text{argmax}_\pi\{\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V}\}$    ▷ greedy policy improvement.
7  return $\pi$
```

Figure 17.6
Policy iteration algorithm.

Figure 17.5 shows a simple example of MDP with two states. The iterated values of these states calculated by the algorithm for that MDP are given by

$$\mathbf{V}_{n+1}(1) = \max\Big\{2 + \gamma\Big(\frac{3}{4}\mathbf{V}_n(1) + \frac{1}{4}\mathbf{V}_n(2)\Big), 2 + \gamma \mathbf{V}_n(2)\Big\}$$
$$\mathbf{V}_{n+1}(2) = \max\Big\{3 + \gamma \mathbf{V}_n(1), 2 + \gamma \mathbf{V}_n(2)\Big\}.$$

For $\mathbf{V}_0(1) = -1$, $\mathbf{V}_0(2) = 1$, and $\gamma = 1/2$, we obtain $\mathbf{V}_1(1) = \mathbf{V}_1(2) = 5/2$. Thus, both states seem to have the same policy value initially. However, by the fifth iteration, $\mathbf{V}_5(1) = 4.53125$, $\mathbf{V}_5(2) = 5.15625$ and the algorithm quickly converges to the optimal values $\mathbf{V}^*(1) = 14/3$ and $\mathbf{V}^*(2) = 16/3$ showing that state 2 has a higher optimal value.

17.4.2 Policy iteration

An alternative algorithm for determining the best policy consists of using policy evaluations, which can be achieved via a matrix inversion, as shown by theorem 17.10. The pseudocode of the algorithm known as *policy iteration algorithm* is given in figure 17.6. Starting with an arbitrary action policy π_0, the algorithm repeatedly computes the value of the current policy π via that matrix inversion and greedily selects the new policy as the one maximizing the right-hand side of the Bellman equations (17.9).

The following theorem proves the convergence of the policy iteration algorithm.

Theorem 17.12 *Let $(\mathbf{V}_n)_{n\in\mathbb{N}}$ be the sequence of policy values computed by the algorithm, then, for any $n \in \mathbb{N}$, the following inequalities hold:*

$$\mathbf{V}_n \le \mathbf{V}_{n+1} \le \mathbf{V}^*. \tag{17.10}$$

Proof: Let π_{n+1} be the policy improvement at the nth iteration of the algorithm. We first show that $(\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1}$ preserves ordering, that is, for any column matrices $\mathbf{X}$ and $\mathbf{Y}$ in $\mathbb{R}^{|S|}$, if $(\mathbf{Y} - \mathbf{X}) \ge \mathbf{0}$, then $(\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1}(\mathbf{Y} - \mathbf{X}) \ge \mathbf{0}$. As shown in the proof of theorem 17.10, $\|\gamma\mathbf{P}\|_\infty = \gamma < 1$. Since the radius of convergence of the power series $(1-x)^{-1}$ is one, we can use its expansion and write

$$(\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1} = \sum_{k=0}^{\infty}(\gamma\mathbf{P}_{\pi_{n+1}})^k.$$

Thus, if $\mathbf{Z} = (\mathbf{Y} - \mathbf{X}) \ge \mathbf{0}$, then $(\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1}\mathbf{Z} = \sum_{k=0}^{\infty}(\gamma\mathbf{P}_{\pi_{n+1}})^k\mathbf{Z} \ge \mathbf{0}$, since the entries of matrix $\mathbf{P}_{\pi_{n+1}}$ and its powers are all non-negative as well as those of $\mathbf{Z}$.

Now, by definition of π_{n+1}, we have

$$\mathbf{R}_{\pi_{n+1}} + \gamma\mathbf{P}_{\pi_{n+1}}\mathbf{V}_n \ge \mathbf{R}_{\pi_n} + \gamma\mathbf{P}_{\pi_n}\mathbf{V}_n = \mathbf{V}_n,$$

which shows that $\mathbf{R}_{\pi_{n+1}} \ge (\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})\mathbf{V}_n$. Since $(\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1}$ preserves ordering, this implies that $\mathbf{V}_{n+1} = (\mathbf{I} - \gamma\mathbf{P}_{\pi_{n+1}})^{-1}\mathbf{R}_{\pi_{n+1}} \ge \mathbf{V}_n$, which concludes the proof of the theorem. □

Note that two consecutive policy values can be equal only at the last iteration of the algorithm. The total number of possible policies is $|A|^{|S|}$, thus this constitutes a straightforward upper bound on the maximal number of iterations. Better upper bounds of the form $O\big(\frac{|A|^{|S|}}{|S|}\big)$ are known for this algorithm.

For the simple MDP shown by figure 17.5, let the initial policy π_0 be defined by $\pi_0(1) = b, \pi_0(2) = c$. Then, the system of linear equations for evaluating this policy is

$$\begin{cases} V_{\pi_0}(1) = 1 + \gamma V_{\pi_0}(2) \\ V_{\pi_0}(2) = 2 + \gamma V_{\pi_0}(2), \end{cases}$$

which gives $V_{\pi_0}(1) = \frac{1+\gamma}{1-\gamma}$ and $V_{\pi_0}(2) = \frac{2}{1-\gamma}$.

Theorem 17.13 *Let $(\mathbf{U}_n)_{n\in\mathbb{N}}$ be the sequence of policy values generated by the value iteration algorithm, and $(\mathbf{V}_n)_{n\in\mathbb{N}}$ the one generated by the policy iteration algorithm. If $\mathbf{U}_0 = \mathbf{V}_0$, then,*

$$\forall n \in \mathbb{N},\ \mathbf{U}_n \le \mathbf{V}_n \le \mathbf{V}^*. \tag{17.11}$$

Proof: We first show that the function $\mathbf{\Phi}$ previously introduced is monotonic. Let $\mathbf{U}$ and $\mathbf{V}$ be such that $\mathbf{U} \le \mathbf{V}$ and let π be the policy such that $\mathbf{\Phi}(\mathbf{U}) = \mathbf{R}_\pi + \gamma\mathbf{P}_\pi\mathbf{U}$.

Then,

$$\mathbf{\Phi}(\mathbf{U}) \leq \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V} \leq \max_{\pi'} \{\mathbf{R}_{\pi'} + \gamma \mathbf{P}_{\pi'} \mathbf{V}\} = \mathbf{\Phi}(\mathbf{V}).$$

The proof is by induction on n. Assume that $\mathbf{U}_n \leq \mathbf{V}_n$, then by the monotonicity of $\mathbf{\Phi}$, we have

$$\mathbf{U}_{n+1} = \mathbf{\Phi}(\mathbf{U}_n) \leq \mathbf{\Phi}(\mathbf{V}_n) = \max_{\pi} \{\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V}_n\}.$$

Let π_{n+1} be the maximizing policy, that is, $\pi_{n+1} = \operatorname{argmax}_\pi \{\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V}_n\}$. Then,

$$\mathbf{\Phi}(\mathbf{V}_n) = \mathbf{R}_{\pi_{n+1}} + \gamma \mathbf{P}_{\pi_{n+1}} \mathbf{V}_n \leq \mathbf{R}_{\pi_{n+1}} + \gamma \mathbf{P}_{\pi_{n+1}} \mathbf{V}_{n+1} = \mathbf{V}_{n+1},$$

and thus $\mathbf{U}_{n+1} \leq \mathbf{V}_{n+1}$. □

The theorem shows that the policy iteration algorithm converges in a smaller number of iterations than the value iteration algorithm due to the optimal policy. But, each iteration of the policy iteration algorithm requires computing a policy value, that is, solving a system of linear equations, which is more expensive to compute than an iteration of the value iteration algorithm.

17.4.3 Linear programming

An alternative formulation of the optimization problem defined by the Bellman equations (17.4) or the proof of Theorem 17.8 is via linear programming (LP), that is an optimization problem with a linear objective function and linear constraints. LPs admit (weakly) polynomial-time algorithmic solutions. There exist a variety of different methods for solving relatively large LPs in practice, using the simplex method, interior-point methods, or a variety of special-purpose solutions. All of these methods could be applied in this context.

By definition, the equations (17.4) are each based on a maximization. These maximizations are equivalent to seeking to minimize all elements of $\{V(s) \colon s \in S\}$ under the constraints $V(s) \geq \mathbb{E}[r(s,a)] + \gamma \sum_{s' \in S} \mathbb{P}[s'|s,a] V(s')$, $(s \in S)$. Thus, this can be written as the following LP for any set of fixed positive weights $\alpha(s) > 0$, $(s \in S)$:

$$\begin{aligned} &\min_{\mathbf{V}} \quad \sum_{s \in S} \alpha(s) V(s) \qquad (17.12)\\ &\text{subject to} \quad \forall s \in S, \forall a \in A, V(s) \geq \mathbb{E}[r(s,a)] + \gamma \sum_{s' \in S} \mathbb{P}[s'|s,a] V(s'), \end{aligned}$$

where $\boldsymbol{\alpha} > \mathbf{0}$ is the vector with the sth component equal to $\alpha(s)$.[25] To make each coefficient $\alpha(s)$ interpretable as a probability, we can further add the constraints that $\sum_{s \in S} \alpha(s) = 1$. The number of rows of this LP is $|S||A|$ and its number of columns $|S|$. The complexity of the solution techniques for LPs is typically more favorable in terms of the number of rows than the number of columns. This motivates a solution based on the equivalent dual formulation of this LP which can be written as

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{s \in S, a \in A} \mathbb{E}[r(s,a)]\, x(s,a) \\ \text{subject to} \quad & \forall s \in S, \sum_{a \in A} x(s',a) = \alpha(s') + \gamma \sum_{s \in S, a \in A} \mathbb{P}[s'|s,a]\, x(s',a) \\ & \forall s \in S, \forall a \in A, x(s,a) \geq 0, \end{aligned} \tag{17.13}$$

and for which the number of rows is only $|S|$ and the number of columns $|S||A|$. Here $x(s,a)$ can be interpreted as the probability of being in state s and taking action a.

17.5 Learning algorithms

This section considers the more general scenario where the environment model of an MDP, that is the transition and reward probabilities, is unknown. This matches many realistic applications of reinforcement learning where, for example, a robot is placed in an environment that it needs to explore in order to reach a specific goal.

How can an agent determine the best policy in this context? Since the environment models are not known, it may seek to learn them by estimating transition or reward probabilities. To do so, as in the standard case of supervised learning, the agent needs some amount of training information. In the context of reinforcement learning with MDPs, the training information is the sequence of immediate rewards the agent receives based on the actions it has taken.

There are two main learning approaches that can be adopted. One known as the *model-free approach* consists of learning an action policy directly. Another one, a *model-based* approach, consists of first learning the environment model, and then of using that to learn a policy. The Q-learning algorithm we present for this problem is widely adopted in reinforcement learning and belongs to the family of model-free approaches.

[25] Let us emphasize that the LP is only in terms of the variables $V(s)$, as indicated by the subscript of the minimization operator, and not in terms of $V(s)$ and $\alpha(s)$.

The estimation and algorithmic methods adopted for learning in reinforcement learning are closely related to the concepts and techniques in *stochastic approximation*. Thus, we start by introducing several useful results of this field that will be needed for the proofs of convergence of the reinforcement learning algorithms presented.

17.5.1 Stochastic approximation

Stochastic approximation methods are iterative algorithms for solving optimization problems whose objective function is defined as the expectation of some random variable, or to find the fixed point of a function H that is accessible only through noisy observations. These are precisely the type of optimization problems found in reinforcement learning. For example, for the Q-learning algorithm we will describe, the optimal state-action value function Q^* is the fixed point of some function H that is defined as an expectation and thus not directly accessible.

We start with a basic result whose proof and related algorithm show the flavor of more complex ones found in stochastic approximation. The theorem is a generalization of a result known as the *strong law of large numbers*. It shows that under some conditions on the coefficients, an iterative sequence of estimates μ_m converges almost surely (a.s.) to the mean of a bounded random variable.

Theorem 17.14 (Mean estimation) *Let X be a random variable taking values in $[0,1]$ and let $x_0, \ldots, x_m$ be i.i.d. values of X. Define the sequence $(\mu_m)_{m\in\mathbb{N}}$ by*

$$\mu_{m+1} = (1-\alpha_m)\mu_m + \alpha_m x_m, \tag{17.14}$$

with $\mu_0 = x_0$, $\alpha_m \in [0,1]$, $\sum_{m\geq 0}\alpha_m = +\infty$ and $\sum_{m\geq 0}\alpha_m^2 < +\infty$. Then,

$$\mu_m \xrightarrow{a.s.} \mathbb{E}[X]. \tag{17.15}$$

Proof: We give the proof of the L_2 convergence. The a.s. convergence is shown later for a more general theorem. By the independence assumption, for $m \geq 0$,

$$\mathrm{Var}[\mu_{m+1}] = (1-\alpha_m)^2\,\mathrm{Var}[\mu_m] + \alpha_m^2\,\mathrm{Var}[x_m] \leq (1-\alpha_m)\,\mathrm{Var}[\mu_m] + \alpha_m^2. \tag{17.16}$$

Let $\epsilon > 0$ and suppose that there exists $N \in \mathbb{N}$ such that for all $m \geq N$, $\mathrm{Var}[\mu_m] \geq \epsilon$. Then, for $m \geq N$,

$$\mathrm{Var}[\mu_{m+1}] \leq \mathrm{Var}[\mu_m] - \alpha_m\,\mathrm{Var}[\mu_m] + \alpha_m^2 \leq \mathrm{Var}[\mu_m] - \alpha_m\epsilon + \alpha_m^2,$$

which implies, by reapplying this inequality, that

$$\mathrm{Var}[\mu_{m+N}] \leq \underbrace{\mathrm{Var}[\mu_N] - \epsilon\sum_{n=N}^{m+N}\alpha_n + \sum_{n=N}^{m+N}\alpha_n^2}_{\to -\infty \text{ when } m\to\infty},$$

contradicting $\operatorname{Var}[\mu_{m+N}] \geq 0$. Thus, this contradicts the existence of such an integer N. Therefore, for all $N \in \mathbb{N}$, there exists $m_0 \geq N$ such that $\operatorname{Var}[\mu_{m_0}] \leq \epsilon$.

Choose N large enough so that for all $m \geq N$, the inequality $\alpha_m \leq \epsilon$ holds. This is possible since the sequence $(\alpha_m^2)_{m\in\mathbb{N}}$ and thus $(\alpha_m)_{m\in\mathbb{N}}$ converges to zero in view of $\sum_{m\geq 0} \alpha_m^2 < +\infty$. We will show by induction that for any $m \geq m_0$, $\operatorname{Var}[\mu_m] \leq \epsilon$, which implies the statement of the theorem.

Assume that $\operatorname{Var}[\mu_m] \leq \epsilon$ for some $m \geq m_0$. Then, using this assumption, inequality 17.16, and the fact that $\alpha_m \leq \epsilon$, the following inequality holds:

$$\operatorname{Var}[\mu_{m+1}] \leq (1-\alpha_m)\epsilon + \epsilon\alpha_m = \epsilon.$$

Thus, this proves that $\lim_{m\to+\infty} \operatorname{Var}[\mu_m] = 0$, that is the L_2 convergence of μ_m to $\mathbb{E}[X]$. □

Note that the hypotheses of the theorem related to the sequence $(\alpha_m)_{m\in\mathbb{N}}$ hold in particular when $\alpha_m = \frac{1}{m}$. The special case of the theorem with this choice of α_m coincides with the strong law of large numbers. This result has tight connections with the general problem of stochastic optimization.

Stochastic optimization is the general problem of finding the solution to the equation

$$\mathbf{x} = H(\mathbf{x}),$$

where $\mathbf{x} \in \mathbb{R}^N$, when

- $H(x)$ cannot be computed, for example, because H is not accessible or because the cost of its computation is prohibitive;
- but an i.i.d. sample of m noisy observations $H(\mathbf{x}_i) + \mathbf{w}_i$ are available, $i \in [m]$, where the noise random variable $\mathbf{w}$ has expectation zero: $\mathbb{E}[\mathbf{w}] = \mathbf{0}$.

This problem arises in a variety of different contexts and applications. As we shall see, it is directly related to the learning problem for MDPs.

One general idea for solving this problem is to use an iterative method and define a sequence $(\mathbf{x}_t)_{t\in\mathbb{N}}$ in a way similar to what is suggested by theorem 17.14:

$$\mathbf{x}_{t+1} = (1-\alpha_t)\mathbf{x}_t + \alpha_t[H(\mathbf{x}_t) + \mathbf{w}_t] \tag{17.17}$$

$$= \mathbf{x}_t + \alpha_t[H(\mathbf{x}_t) + \mathbf{w}_t - \mathbf{x}_t], \tag{17.18}$$

where $(\alpha_t)_{t\in\mathbb{N}}$ follow conditions similar to those assumed in theorem 17.14. More generally, we consider sequences defined via

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_t D(\mathbf{x}_t, \mathbf{w}_t), \tag{17.19}$$

where D is a function mapping $\mathbb{R}^N \times \mathbb{R}^N$ to $\mathbb{R}^N$. There are many different theorems guaranteeing the convergence of this sequence under various assumptions. We will present one of the most general forms of such theorems, which relies on the following result.

Theorem 17.15 (Supermartingale convergence) *Let $(X_t)_{t\in\mathbb{N}}$, $(Y_t)_{t\in\mathbb{N}}$, and $(Z_t)_{t\in\mathbb{N}}$ be sequences of non-negative random variables such that $\sum_{t=0}^{+\infty} Y_t < +\infty$. Let $\mathcal{F}_t$ denote all the information for $t' \leq t$: $\mathcal{F}_t = \{(X_{t'})_{t'\leq t}, (Y_{t'})_{t'\leq t}, (Z_{t'})_{t'\leq t}\}$. Then, if $\mathbb{E}\Big[X_{t+1}\big|\mathcal{F}_t\Big] \leq X_t + Y_t - Z_t$, the following holds:*

- *X_t converges to a limit (with probability one).*
- *$\sum_{t=0}^{+\infty} Z_t < +\infty$.*

The following is one of the most general forms of such theorems.

Theorem 17.16 *Let D be a function mapping $\mathbb{R}^N \times \mathbb{R}^N$ to $\mathbb{R}^N$, $(\mathbf{w}_t)_{t\in\mathbb{N}}$ a sequence of random variables in $\mathbb{R}^N$, $(\alpha_t)_{t\in\mathbb{N}}$ a sequence of real numbers, and $(\mathbf{x}_t)_{t\in\mathbb{N}}$ a sequence defined by $\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_t D(\mathbf{x}_t, \mathbf{w}_t)$ with $\mathbf{x}_0 \in \mathbb{R}^N$. Let $\mathcal{F}_t$ denote the entire history up to t, that is: $\mathcal{F}_t = \{(\mathbf{x}_{t'})_{t'\leq t}, (\mathbf{w}_{t'})_{t'\leq t-1}, (\alpha_{t'})_{t'\leq t}\}$, and let Ψ denote the function $\mathbf{x} \to \frac{1}{2}\|\mathbf{x}-\mathbf{x}^*\|_2^2$ for some $\mathbf{x}^* \in \mathbb{R}^N$. Assume that D and $(\alpha)_{t\in\mathbb{N}}$ verify the following conditions:*

- *$\exists K_1, K_2 \in \mathbb{R}$: $\mathbb{E}\Big[\|D(\mathbf{x}_t, \mathbf{w}_t)\|_2^2 \,\Big|\, \mathcal{F}_t\Big] \leq K_1 + K_2\,\Psi(\mathbf{x}_t)$;*
- *$\exists c \geq 0$: $\nabla\Psi(\mathbf{x}_t)^\top\,\mathbb{E}\Big[D(\mathbf{x}_t, \mathbf{w}_t)\,\Big|\,\mathcal{F}_t\Big] \leq -c\,\Psi(\mathbf{x}_t)$;*
- *$\alpha_t > 0, \sum_{t=0}^{+\infty}\alpha_t = +\infty, \sum_{t=0}^{+\infty}\alpha_t^2 < +\infty$.*

Then, the sequence $\mathbf{x}_t$ converges almost surely to $\mathbf{x}^*$:

$$\mathbf{x}_t \xrightarrow{a.s.} \mathbf{x}^*. \tag{17.20}$$

Proof: Since function Ψ is quadratic, a Taylor expansion gives

$$\Psi(\mathbf{x}_{t+1}) = \Psi(\mathbf{x}_t) + \nabla\Psi(\mathbf{x}_t)^\top(\mathbf{x}_{t+1}-\mathbf{x}_t) + \frac{1}{2}(\mathbf{x}_{t+1}-\mathbf{x}_t)^\top\nabla^2\Psi(\mathbf{x}_t)(\mathbf{x}_{t+1}-\mathbf{x}_t).$$

Thus,

$$\begin{aligned}\mathbb{E}\Big[\Psi(\mathbf{x}_{t+1})\big|\mathcal{F}_t\Big] &= \Psi(\mathbf{x}_t) + \alpha_t\nabla\Psi(\mathbf{x}_t)^\top\,\mathbb{E}\Big[D(\mathbf{x}_t,\mathbf{w}_t)\big|\mathcal{F}_t\Big] + \frac{\alpha_t^2}{2}\,\mathbb{E}\Big[\|D(\mathbf{x}_t,\mathbf{w}_t)\|^2\big|\mathcal{F}_t\Big]\\ &\leq \Psi(\mathbf{x}_t) - \alpha_t c\Psi(\mathbf{x}_t) + \frac{\alpha_t^2}{2}(K_1 + K_2\Psi(\mathbf{x}_t))\\ &= \Psi(\mathbf{x}_t) + \frac{\alpha_t^2 K_1}{2} - \Big(\alpha_t c - \frac{\alpha_t^2 K_2}{2}\Big)\Psi(\mathbf{x}_t).\end{aligned}$$

Since by assumption the series $\sum_{t=0}^{+\infty}\alpha_t^2$ is convergent, $(\alpha_t^2)_t$ and thus $(\alpha_t)_t$ converges to zero. Therefore, for t sufficiently large, the term $\big(\alpha_t c - \frac{\alpha_t^2 K_2}{2}\big)\Psi(\mathbf{x}_t)$ has the sign of $\alpha_t c\Psi(\mathbf{x}_t)$ and is non-negative, since $\alpha_t > 0$, $\Psi(\mathbf{x}_t) \geq 0$, and $c > 0$. Thus, by the supermartingale convergence theorem 17.15, $\Psi(\mathbf{x}_t)$ converges and $\sum_{t=0}^{+\infty}\big(\alpha_t c - \frac{\alpha_t^2 K_2}{2}\big)\Psi(\mathbf{x}_t) < +\infty$. Since $\Psi(\mathbf{x}_t)$ converges and $\sum_{t=0}^{+\infty}\alpha_t^2 < +\infty$, we have $\sum_{t=0}^{+\infty}\frac{\alpha_t^2 K_2}{2}\Psi(\mathbf{x}_t) < +\infty$. But, since $\sum_{t=0}^{+\infty}\alpha_t = +\infty$, if the limit of $\Psi(\mathbf{x}_t)$ were non-zero, we would have $\sum_{t=0}^{+\infty}\alpha_t c\Psi(\mathbf{x}_t) = +\infty$. This implies that the limit of $\Psi(\mathbf{x}_t)$ is zero, that is $\lim_{t\to\infty} \|\mathbf{x}_t - \mathbf{x}^*\|_2 \to 0$, which implies $\mathbf{x}_t \xrightarrow{\text{a.s.}} \mathbf{x}^*$. □

The following is another related result for which we do not present the full proof.

Theorem 17.17 *Let $\mathbf{H}$ be a function mapping $\mathbb{R}^N$ to $\mathbb{R}^N$, $(\mathbf{w}_t)_{t\in\mathbb{N}}$ a sequence of random variables in $\mathbb{R}^N$, $(\alpha_t)_{t\in\mathbb{N}}$ a sequence of real numbers, and $(\mathbf{x}_t)_{t\in\mathbb{N}}$ a sequence defined by*

$$\forall s \in [N], \quad \mathbf{x}_{t+1}(s) = \mathbf{x}_t(s) + \alpha_t(s)\big[\mathbf{H}(\mathbf{x}_t)(s) - \mathbf{x}_t(s) + \mathbf{w}_t(s)\big],$$

for some $\mathbf{x}_0 \in \mathbb{R}^N$. Define $\mathcal{F}_t$ by $\mathcal{F}_t = \{(\mathbf{x}_{t'})_{t'\le t}, (\mathbf{w}_{t'})_{t'\le t-1}(\alpha_{t'})_{t'\le t}\}$, that is the entire history up to t, and assume that the following conditions are met:

- $\exists K_1, K_2 \in \mathbb{R}$: $\mathbb{E}\Big[\|\mathbf{w}_t\|^2(s)\,\Big|\,\mathcal{F}_t\Big] \le K_1 + K_2\,\|\mathbf{x}_t\|^2$ *for some norm* $\|\cdot\|$;
- $\mathbb{E}\big[\mathbf{w}_t\,\big|\,\mathcal{F}_t\big] = 0$;
- $\forall s \in [N], \sum_{t=0}^{+\infty} \alpha_t(s) = +\infty, \sum_{t=0}^{+\infty} \alpha_t^2(s) < +\infty$; *and*
- $\mathbf{H}$ *is a* $\|\cdot\|_\infty$*-contraction with fixed point* $\mathbf{x}^*$.

Then, the sequence $\mathbf{x}_t$ converges almost surely to $\mathbf{x}^*$:

$$\mathbf{x}_t \xrightarrow{a.s.} \mathbf{x}^*. \tag{17.21}$$

The next sections present several learning algorithms for MDPs with an unknown model.

17.5.2 TD(0) algorithm

This section presents an algorithm, TD(0) algorithm, for evaluating a policy in the case where the environment model is unknown. The algorithm is based on Bellman's linear equations giving the value of a policy π (see proposition 17.9):

$$\begin{aligned} V_\pi(s) &= \mathbb{E}[r(s,\pi(s)] + \gamma \sum_{s'} \mathbb{P}[s'|s,\pi(s)] V_\pi(s') \\ &= \mathop{\mathbb{E}}_{s'}\big[r(s,\pi(s)) + \gamma V_\pi(s')|s\big]. \end{aligned}$$

However, here the probability distribution according to which this last expectation is defined is not known. Instead, the TD(0) algorithm consists of

- sampling a new state s'; and
- updating the policy values according to the following, which justifies the name of the algorithm:

$$\begin{aligned} V(s) &\leftarrow (1-\alpha)V(s) + \alpha[r(s,\pi(s)) + \gamma V(s')] \\ &= V(s) + \alpha[\underbrace{r(s,\pi(s)) + \gamma V(s') - V(s)}_{\text{temporal difference of } V \text{ values}}]. \end{aligned} \tag{17.22}$$

Here, the parameter α is a function of the number of visits to the state s.

```
TD(0)()
1  $\mathbf{V} \leftarrow \mathbf{V}_0$ ▷ initialization.
2  for $t \leftarrow 0$ to $T$ do
3      $s \leftarrow$ SelectState()
4      for each step of epoch $t$ do
5          $r' \leftarrow$ Reward$(s, \pi(s))$
6          $s' \leftarrow$ NextState$(\pi, s)$
7          $V(s) \leftarrow (1-\alpha)V(s) + \alpha[r' + \gamma V(s')]$
8          $s \leftarrow s'$
9  return $\mathbf{V}$
```

The pseudocode of the algorithm is given above. The algorithm starts with an arbitrary policy value vector $\mathbf{V}_0$. An initial state is returned by SelectState at the beginning of each epoch. Within each epoch, the iteration continues until a final state is found. Within each iteration, action $\pi(s)$ is taken from the current state s following policy π. The new state s' reached and the reward r' received are observed. The policy value of state s is then updated according to the rule (17.22) and current state set to be s'.

The convergence of the algorithm can be proven using theorem 17.17. We will give instead the full proof of the convergence of the Q-learning algorithm, for which that of TD(0) can be viewed as a special case.

17.5.3 Q-learning algorithm

This section presents an algorithm for estimating the optimal state-action value function Q^* in the case of an unknown model. Note that the optimal policy or policy value can be straightforwardly derived from Q^* via: $\pi^*(s) = \operatorname{argmax}_{a \in A} Q^*(s, a)$ and $V^*(s) = \max_{a \in A} Q^*(s, a)$. To simplify the presentation, we will assume a deterministic reward function.

The Q-learning algorithm is based on the equations giving the optimal state-action value function Q^* (17.1):

$$\begin{aligned} Q^*(s, a) &= \mathbb{E}[r(s, a)] + \gamma \sum_{s' \in S} \mathbb{P}[s' \mid s, a] V^*(s') \\ &= \mathop{\mathbb{E}}_{s'}[r(s, a) + \gamma \max_{a \in A} Q^*(s', a)]. \end{aligned}$$

```
Q-LEARNING($\pi$)
 1  $Q \leftarrow Q_0$    ▷ initialization, e.g., $Q_0 = 0$.
 2  for $t \leftarrow 0$ to $T$ do
 3      $s \leftarrow$ SELECTSTATE()
 4      for each step of epoch $t$ do
 5          $a \leftarrow$ SELECTACTION($\pi, s$) ▷   policy $\pi$ derived from $Q$, e.g., $\epsilon$-greedy.
 6          $r' \leftarrow$ REWARD($s, a$)
 7          $s' \leftarrow$ NEXTSTATE($s, a$)
 8          $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r' + \gamma \max_{a'} Q(s',a') - Q(s,a)\big]$
 9          $s \leftarrow s'$
10  return $Q$
```

As for the policy values in the previous section, the distribution model is not known. Thus, the Q-learning algorithm consists of the following main steps:

- sampling a new state s'; and
- updating the policy values according to the following:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r(s,a) + \gamma \max_{a' \in A} Q(s',a')]. \tag{17.23}$$

where the parameter α is a function of the number of visits to the state s.

The algorithm can be viewed as a stochastic formulation of the value iteration algorithm presented in the previous section. The pseudocode is given above. Within each epoch, an action is selected from the current state s using a policy π derived from Q. The choice of the policy π is arbitrary so long as it guarantees that every pair (s,a) is visited infinitely many times. The reward received and the state s' observed are then used to update Q following (17.23).

Theorem 17.18 *Consider a finite MDP. Assume that for all $s \in S$ and $a \in A$, $\sum_{t=0}^{+\infty} \alpha_t(s,a) = +\infty$, and $\sum_{t=0}^{+\infty} \alpha_t^2(s,a) < +\infty$ with $\alpha_t(s,a) \in [0,1]$. Then, the Q-learning algorithm converges to the optimal value Q^* (with probability one).*

Note that the conditions on $\alpha_t(s,a)$ impose that each state-action pair is visited infinitely many times.

Proof: Let $(Q_t(s,a))_{t\geq 0}$ denote the sequence of state-action value functions at $(s,a) \in S \times A$ generated by the algorithm. By definition of the Q-learning updates,

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha\big[r(s_t,a_t) + \gamma \max_{a'} Q_t(s_{t+1},a') - Q_t(s_t,a_t)\big].$$

This can be rewritten as the following for all $s \in S$ and $a \in A$:

$$\begin{aligned} Q_{t+1}(s,a) = Q_t(s,a) + \alpha_t(s,a) \left[r(s,a) + \gamma \operatorname*{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]} \left[\max_{a'} Q_t(u,a')\right] - Q_t(s,a)\right] \\ + \gamma\alpha_t(s,a)\left[\max_{a'} Q_t(s',a') - \operatorname*{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]}\left[\max_{a'} Q_t(u,a')\right]\right], \end{aligned} \tag{17.24}$$

if we define $s' = \textsc{NextState}(s,a)$ and $\alpha_t(s,a)$ as 0 if $(s,a) \neq (s_t,a_t)$ and $\alpha_t(s_t,a_t)$ otherwise. Now, let $\mathbf{Q}_t$ denote the vector with components $Q_t(s,a)$, $\mathbf{w}_t$ the vector whose s'th entry is

$$w_t(s) = \max_{a'} Q_t(s',a') - \operatorname*{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]}\left[\max_{a'} Q_t(u,a')\right],$$

and $\mathbf{H}(\mathbf{Q}_t)$ the vector with components $\mathbf{H}(\mathbf{Q}_t)(s,a)$ defined by

$$\mathbf{H}(\mathbf{Q}_t)(s,a) = r(s,a) + \gamma \operatorname*{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]}\left[\max_{a'} Q_t(u,a')\right].$$

Then, in view of (17.24),

$$\forall (s,a) \in S\times A, \quad \mathbf{Q}_{t+1}(s,a) = \mathbf{Q}_t(s,a) + \alpha_t(s,a)\big[\mathbf{H}(\mathbf{Q}_t)(s,a) - \mathbf{Q}_t(s,a) + \gamma \mathbf{w}_t(s)\big].$$

We now show that the hypotheses of theorem 17.17 hold for $\mathbf{Q}_t$ and $\mathbf{w}_t$, which will imply the convergence of $\mathbf{Q}_t$ to $\mathbf{Q}^*$. The conditions on α_t hold by assumption. By definition of $\mathbf{w}_t$, $\mathbb{E}[\mathbf{w}_t \mid \mathcal{F}_t] = 0$. Also, for any $s' \in S$,

$$\begin{aligned} |\mathbf{w}_t(s)| &\leq \max_{a'} |Q_t(s',a')| + \left|\operatorname*{\mathbb{E}}_{u\sim\mathbb{P}[\cdot|s,a]}\left[\max_{a'} Q_t(u,a')\right]\right| \\ &\leq 2\max_{s'} |\max_{a'} Q_t(s',a')| = 2\|\mathbf{Q}_t\|_\infty. \end{aligned}$$

Thus, $\mathbb{E}\left[\mathbf{w}_t^2(s) \,\middle|\, \mathcal{F}_t\right] \leq 4\|\mathbf{Q}_t\|_\infty^2$. Finally, $\mathbf{H}$ is a γ-contraction for $\|\cdot\|_\infty$ since for any $\mathbf{Q}_1, \mathbf{Q}_2 \in \mathbb{R}^{|S|\times|A|}$, and $(s,a) \in S \times A$, we can write

$$\begin{aligned}
|\mathbf{H}(\mathbf{Q}_2)(x,a) - \mathbf{H}(\mathbf{Q}_1)(x,a)| &= \left|\gamma \underset{u\sim\mathbb{P}[\cdot|s,a]}{\mathbb{E}}\left[\max_{a'} Q_2(u,a') - \max_{a'} Q_1(u,a')\right]\right| \\
&\leq \gamma \underset{u\sim\mathbb{P}[\cdot|s,a]}{\mathbb{E}}\left[\left|\max_{a'} Q_2(u,a') - \max_{a'} Q_1(u,a')\right|\right] \\
&\leq \gamma \underset{u\sim\mathbb{P}[\cdot|s,a]}{\mathbb{E}} \max_{a'} \left[|Q_2(u,a') - Q_1(u,a')|\right] \\
&\leq \gamma \max_{u} \max_{a'} \left[|Q_2(u,a') - Q_1(u,a')|\right] \\
&= \gamma \|\mathbf{Q}_2 - \mathbf{Q}_1\|_\infty.
\end{aligned}$$

Since $\mathbf{H}$ is a contraction, it admits a fixed point $\mathbf{Q}^*$: $\mathbf{H}(\mathbf{Q}^*) = \mathbf{Q}^*$. □

The choice of the policy π according to which an action a is selected (line 5) is not specified by the algorithm and, as already indicated, the theorem guarantees the convergence of the algorithm for an arbitrary policy so long as it ensures that every pair (s,a) is visited infinitely many times. In practice, several natural choices are considered for π. One possible choice is the policy determined by the state-action value at time t, Q_t. Thus, the action selected from state s is $\operatorname{argmax}_{a\in A} Q_t(s,a)$. But this choice typically does not guarantee that all actions are taken or that all states are visited. Instead, a standard choice in reinforcement learning is the so-called *ϵ-greedy policy*, which consists of selecting with probability $(1-\epsilon)$ the greedy action from state s, that is, $\operatorname{argmax}_{a\in A} Q_t(s,a)$, and with probability ϵ a random action from s, for some $\epsilon \in (0,1)$. Another possible choice is the so-called *Boltzmann exploration*, which, given the current state-action value Q, epoch $t \in \{0,\ldots,T\}$, and current state s, consists of selecting action a with the following probability:

$$p_t(a|s,Q) = \frac{e^{\frac{Q(s,a)}{\tau_t}}}{\sum_{a'\in A} e^{\frac{Q(s,a')}{\tau_t}}},$$

where τ_t is the *temperature.* τ_t must be defined so that $\tau_t \to 0$ as $t \to +\infty$, which ensures that for large values of t, the greedy action based on Q is selected. This is natural, since as t increases, we can expect Q to be close to the optimal function. On the other hand, τ_t must be chosen so that it does not tend to 0 too fast to ensure that all actions are visited infinitely often. It can be chosen, for instance, as $1/\log(n_t(s))$, where $n_t(s)$ is the number of times s has been visited up to epoch t.

Reinforcement learning algorithms include two components: a *learning policy*, which determines the action to take, and an *update rule*, which defines the new estimate of the optimal value function. For an *off-policy algorithm*, the update rule does not necessarily depend on the learning policy. Q-learning is an off-policy algorithm since its update rule (line 8 of the pseudocode) is based on the max operator and the comparison of all possible actions a', that is the greedy action, which may not coincide with the action recommended by the current the policy π. More generally, an off-policy algorithm evaluates or improves one policy, while acting based on another policy.

In contrast, the algorithm presented in the next section, SARSA, is an *on-policy algorithm*. An on-policy algorithm evaluates and improves the current policy used for control. It evaluates the return based on the algorithm's policy.

17.5.4 SARSA

SARSA is also an algorithm for estimating the optimal state-action value function in the case of an unknown model. The pseudocode is given in figure 17.7. The algorithm is in fact very similar to Q-learning, except that its update rule (line 9 of the pseudocode) is based on the action a' selected by the learning policy. Thus, SARSA is an on-policy algorithm, and its convergence therefore crucially depends on the learning policy. In particular, the convergence of the algorithm requires, in addition to all actions being selected infinitely often, that the learning policy becomes greedy in the limit. The proof of the convergence of the algorithm is nevertheless close to that of Q-learning.

The name of the algorithm derives from the sequence of instructions defining successively s, a, r', s', and a', and the fact that the update to the function Q depends on the quintuple (s, a, r', s', a).

17.5.5 TD(λ) algorithm

Both TD(0) and Q-learning algorithms are only based on immediate rewards. The idea of TD(λ) consists instead of using multiple steps ahead. Thus, for $n > 1$ steps, we would have the update

$$V(s) \leftarrow V(s) + \alpha\,(R_t^n - V(s)),$$

where R_t^n is defined by

$$R_t^n = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n}).$$

How should n be chosen? Instead of selecting a specific n, TD(λ) is based on a geometric distribution over all rewards R_t^n, that is, it uses $R_t^\lambda = (1-\lambda)\sum_{n=0}^{+\infty} \lambda^n R_t^n$ instead of R_t^n where $\lambda \in [0, 1]$. Thus, the main update becomes

$$V(s) \leftarrow V(s) + \alpha\,(R_t^\lambda - V(s)).$$

The pseudocode of the algorithm is given above. For $\lambda = 0$, the algorithm coincides with TD(0). $\lambda = 1$ corresponds to the total future reward.

```
SARSA($\pi$)
 1  $Q \leftarrow Q_0$    ▷ initialization, e.g., $Q_0 = 0$.
 2  for $t \leftarrow 0$ to $T$ do
 3      $s \leftarrow$ SelectState()
 4      $a \leftarrow$ SelectAction($\pi(Q), s$) ▷   policy $\pi$ derived from $Q$, e.g., $\epsilon$-greedy.
 5      for each step of epoch $t$ do
 6          $r' \leftarrow$ Reward($s, a$)
 7          $s' \leftarrow$ NextState($s, a$)
 8          $a' \leftarrow$ SelectAction($\pi(Q), s'$) ▷   policy $\pi$ derived from $Q$, e.g., $\epsilon$-greedy.
 9          $Q(s, a) \leftarrow Q(s, a) + \alpha_t(s, a)\big[r' + \gamma Q(s', a') - Q(s, a)\big]$
10          $s \leftarrow s'$
11          $a \leftarrow a'$
12  return $Q$
```

Figure 17.7
The SARSA algorithm.

In the previous sections, we presented learning algorithms for an agent navigating in an unknown environment. The scenario faced in many practical applications is more challenging; often, the information the agent receives about the environment is uncertain or unreliable. Such problems can be modeled as partially observable Markov decision processes (POMDPs). POMDPs are defined by augmenting the definition of MDPs with an observation probability distribution depending on the action taken, the state reached, and the observation. The presentation of their model and solution techniques are beyond the scope of this material.

17.5.6 Large state space

In some cases in practice, the number of states or actions to consider for the environment may be very large. For example, the number of states in the game of backgammon is estimated to be over 10^{20}. Thus, the algorithms presented in the previous section can become computationally impractical for such applications. More importantly, generalization becomes extremely difficult.

Suppose we wish to estimate the policy value $V_\pi(s)$ at each state s using experience obtained using policy π. To cope with the case of large state spaces, we

```
TD($\lambda$)()
 1  $\mathbf{V} \leftarrow \mathbf{V}_0$ ▷ initialization.
 2  $\mathbf{e} \leftarrow \mathbf{0}$
 3  for $t \leftarrow 0$ to $T$ do
 4      $s \leftarrow$ SelectState()
 5      for each step of epoch $t$ do
 6          $s' \leftarrow$ NextState($\pi, s$)
 7          $\delta \leftarrow r(s, \pi(s)) + \lambda V(s') - V(s)$
 8          $e(s) \leftarrow \lambda e(s) + 1$
 9          for $u \in S$ do
10              if $u \neq s$ then
11                  $e(u) \leftarrow \gamma\lambda e(u)$
12              $V(u) \leftarrow V(u) + \alpha\delta e(u)$
13          $s \leftarrow s'$
14  return $\mathbf{V}$
```

can map each state of the environment to $\mathbb{R}^N$ via a mapping $\mathbf{\Phi}\colon S \to \mathbb{R}^N$, with N relatively small ($N \approx 200$ has been used for backgammon) and approximate $V_\pi(s)$ by a function $f_{\mathbf{w}}(s)$ parameterized by some vector $\mathbf{w}$. For example, $f_{\mathbf{w}}$ could be a linear function defined by $f_{\mathbf{w}}(s) = \mathbf{w} \cdot \mathbf{\Phi}(s)$ for all $s \in S$, or some more complex non-linear function of $\mathbf{w}$. The problem then consists of approximating V_π with $f_{\mathbf{w}}$ and can be formulated as a regression problem. Note, however, that the empirical data available is not i.i.d.

Suppose that at each time step t the agent receives the exact policy value $V_\pi(s_t)$. Then, if the family of functions $f_{\mathbf{w}}$ is differentiable, a gradient descent method applied to the empirical squared loss can be used to sequentially update the weight vector $\mathbf{w}$ via:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla_{\mathbf{w}_t} \frac{1}{2}[V_\pi(s_t) - f_{\mathbf{w}_t}(s_t)]^2 = \mathbf{w}_t + \alpha[V_\pi(s_t) - f_{\mathbf{w}_t}(s_t)] \nabla_{\mathbf{w}_t} f_{\mathbf{w}_t}(s_t).$$

It is worth mentioning, however, that for large action spaces, there are simple cases where the methods used do not converge and instead cycle.

17.6 Chapter notes

Reinforcement learning is an important area of machine learning with a large body of literature. This chapter presents only a brief introduction to this area. For a more detailed study, the reader could consult the book of Sutton and Barto [1998], whose mathematical content is short, or those of Puterman [1994] and Bertsekas [1987], which discuss in more depth several aspects, as well as the more recent book of Szepesvári [2010]. The Ph.D. theses of Singh [1993] and Littman [1996] are also excellent sources.

Some foundational work on MDPs and the introduction of the temporal difference (TD) methods are due to Sutton [1984]. Q-learning was introduced and analyzed by Watkins [1989], though it can be viewed as a special instance of TD methods. The first proof of the convergence of Q-learning was given by Watkins and Dayan [1992].

Many of the techniques used in reinforcement learning are closely related to those of stochastic approximation which originated with the work of Robbins and Monro [1951], followed by a series of results including Dvoretzky [1956], Schmetterer [1960], Kiefer and Wolfowitz [1952], and Kushner and Clark [1978]. For a recent survey of stochastic approximation, including a discussion of powerful proof techniques based on ODE (ordinary differential equations), see Kushner [2010] and the references therein. The connection with stochastic approximation was emphasized by Tsitsiklis [1994] and Jaakkola et al. [1994], who gave a related proof of the convergence of Q-learning. For the convergence rate of Q-learning, consult Even-Dar and Mansour [2003]. For recent results on the convergence of the policy iteration algorithm, see Ye [2011], which shows that the algorithm is strongly polynomial for a fixed discount factor.

Reinforcement learning has been successfully applied to a variety of problems including robot control, board games such as backgammon in which Tesauro's TD-Gammon reached the level of a strong master [Tesauro, 1995] (see also chapter 11 of Sutton and Barto [1998]), chess, elevator scheduling problems [Crites and Barto, 1996], telecommunications, inventory management, dynamic radio channel assignment [Singh and Bertsekas, 1997], and a number of other problems (see chapter 1 of Puterman [1994]).

Conclusion

We described a large variety of machine learning algorithms and techniques and discussed their theoretical foundations as well as their use and applications. While this is not a fully comprehensive presentation, it should nevertheless offer the reader some idea of the breadth of the field and its multiple connections with a variety of other domains, including statistics, information theory, optimization, game theory, and automata and formal language theory.

The fundamental concepts, algorithms, and proof techniques we presented should supply the reader with the necessary tools for analyzing other learning algorithms, including variants of the algorithms analyzed in this book. They are also likely to be helpful for devising new algorithms or for studying new learning schemes. We strongly encourage the reader to explore both and more generally to seek enhanced solutions for all theoretical, algorithmic, and applied learning problems.

The exercises included at the end of each chapter, as well as the full solutions we provide separately, should help the reader become more familiar with the techniques and concepts described. Some of them could also serve as a starting point for research work and the investigation of new questions.

Many of the algorithms we presented as well as their variants can be directly used in applications to derive effective solutions to real-world learning problems. Our detailed description of the algorithms and discussion should help with their implementation or their adaptation to other learning scenarios.

Machine learning is a relatively recent field and yet probably one of the most active ones in computer science. Given the wide accessibility of digitized data and its many applications, we can expect it to continue to grow at a very fast pace over the next few decades. Learning problems of different nature, some arising due to the substantial increase of the scale of the data, which already requires processing billions of records in some applications, others related to the introduction of completely new learning frameworks, are likely to pose new research challenges and require novel algorithmic solutions. In all cases, learning theory, algorithms, and applications form an exciting area of computer science and mathematics, which we hope this book could at least partly communicate.

A Linear Algebra Review

In this appendix, we introduce some basic notions of linear algebra relevant to the material presented in this book. This appendix does not represent an exhaustive tutorial, and it is assumed that the reader has some prior knowledge of the subject.

A.1 Vectors and norms

We will denote by $\mathbb{H}$ a vector space whose dimension may be infinite.

A.1.1 Norms

Definition A.1 *A mapping* $\Phi\colon \mathbb{H} \to \mathbb{R}_+$ *is said to define a* norm *on* $\mathbb{H}$ *if it verifies the following axioms:*

- *definiteness:* $\forall \mathbf{x} \in \mathbb{H}, \Phi(\mathbf{x}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$*;*
- *homogeneity:* $\forall \mathbf{x} \in \mathbb{H}, \forall \alpha \in \mathbb{R}, \Phi(\alpha\mathbf{x}) = |\alpha|\Phi(\mathbf{x})$*;*
- *triangle inequality:* $\forall \mathbf{x}, \mathbf{y} \in \mathbb{H}, \Phi(\mathbf{x}+\mathbf{y}) \leq \Phi(\mathbf{x}) + \Phi(\mathbf{y})$*.*

A norm is typically denoted by $\|\cdot\|$. Examples of vector norms are the absolute value on $\mathbb{R}$ and the Euclidean (or L_2) norm on $\mathbb{R}^N$. More generally, for any $p \geq 1$ the L_p norm is defined on $\mathbb{R}^N$ as

$$\forall \mathbf{x} \in \mathbb{R}^N, \quad \|\mathbf{x}\|_p = \Big(\sum_{j=1}^N |x_j|^p\Big)^{1/p}. \tag{A.1}$$

The L_1, L_2, and L_∞ norms are some of the most commonly used norms, where $\|\mathbf{x}\|_\infty = \max_{j\in[N]} |x_j|$. Two norms $\|\cdot\|$ and $\|\cdot\|'$ are said to be *equivalent* iff there exists $\alpha, \beta > 0$ such that for all $\mathbf{x} \in \mathbb{H}$,

$$\alpha\|\mathbf{x}\| \leq \|\mathbf{x}\|' \leq \beta\|\mathbf{x}\|. \tag{A.2}$$

The following general inequalities relating these norms can be proven straightforwardly:

$$\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{N}\|\mathbf{x}\|_2 \tag{A.3}$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{N}\|\mathbf{x}\|_\infty \tag{A.4}$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_1 \leq N\|\mathbf{x}\|_\infty. \tag{A.5}$$

The second inequality of the first line can be shown using the *Cauchy-Schwarz inequality* presented later while the other inequalities are clear. These inequalities show the equivalence of these three norms. More generally, all norms on a finite-dimensional space are equivalent. The following additional properties hold for the L_∞ norm: for all $\mathbf{x} \in \mathbb{H}$,

$$\forall p \geq 1,\ \|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_p \leq N^{1/p}\|\mathbf{x}\|_\infty \tag{A.6}$$

$$\lim_{p\to+\infty} \|\mathbf{x}\|_p = \|\mathbf{x}\|_\infty. \tag{A.7}$$

The inequalities of the first line are straightforward and imply the limit property of the second line.

Definition A.2 (Hilbert space) *A* Hilbert space *is a vector space equipped with an inner product $\langle \cdot, \cdot \rangle$ and that is* complete *(all Cauchy sequences are convergent). The inner product induces a norm defined as follows:*

$$\forall \mathbf{x} \in \mathbb{H}, \quad \|\mathbf{x}\|_{\mathbb{H}} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}. \tag{A.8}$$

A.1.2 Dual norms

Definition A.3 *Let $\|\cdot\|$ be a norm on $\mathbb{R}^N$. Then, the* dual norm $\|\cdot\|_*$ *associated to $\|\cdot\|$ is the norm defined by*

$$\forall \mathbf{y} \in \mathbb{R}^N, \quad \|\mathbf{y}\|_* = \sup_{\|\mathbf{x}\|=1} |\langle \mathbf{y}, \mathbf{x} \rangle|. \tag{A.9}$$

For any $p, q \geq 1$ that are *conjugate* that is such that $\frac{1}{p} + \frac{1}{q} = 1$, the L_p and L_q norms are dual norms of each other. In particular, the dual norm of L_2 is the L_2 norm, and the dual norm of the L_1 norm is the L_∞ norm.

Proposition A.4 (Hölder's inequality) *Let $p, q \geq 1$ be conjugate: $\frac{1}{p} + \frac{1}{q} = 1$. Then, for all $x, y \in \mathbb{R}^N$,*

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\|_p \|\mathbf{y}\|_q, \tag{A.10}$$

with equality when $|y_i| = |x_i|^{p-1}$ for all $i \in [N]$.

Proof: The statement holds trivially for $\mathbf{x} = \mathbf{0}$ or $\mathbf{y} = \mathbf{0}$; thus, we can assume $\mathbf{x} \neq \mathbf{0}$ and $\mathbf{y} \neq \mathbf{0}$. Let $a, b > 0$. By the concavity of log (see definition B.7), we can write

$$\log\left(\frac{1}{p}a^p + \frac{1}{q}b^q\right) \geq \frac{1}{p}\log(a^p) + \frac{1}{q}\log(b^q) = \log(a) + \log(b) = \log(ab).$$

Taking the exponential of the left- and right-hand sides gives

$$\frac{1}{p}a^p + \frac{1}{q}b^q \geq ab,$$

which is known as *Young's inequality*. Using this inequality with $a = |x_j|/\|\mathbf{x}\|_p$ and $b = |y_j|/\|\mathbf{y}\|_q$ for $j \in [N]$ and summing up gives

$$\frac{\sum_{j=1}^N |x_j y_j|}{\|\mathbf{x}\|_p \|\mathbf{y}\|_q} \leq \frac{1}{p}\frac{\|\mathbf{x}\|^p}{\|\mathbf{x}\|^p} + \frac{1}{q}\frac{\|\mathbf{y}\|^q}{\|\mathbf{y}\|^q} = \frac{1}{p} + \frac{1}{q} = 1.$$

Since $|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \sum_{j=1}^N |x_j y_j|$, the inequality claim follows. The equality case can be verified straightforwardly. □

Taking $p = q = 2$ immediately yields the following result known as the *Cauchy-Schwarz inequality*.

Corollary A.5 (Cauchy-Schwarz inequality) *For all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$,*

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2, \tag{A.11}$$

with equality iff $\mathbf{x}$ and $\mathbf{y}$ are collinear.

Let $\mathcal{H}$ be the hyperplane in $\mathbb{R}^N$ whose equation is given by

$$\mathbf{w} \cdot \mathbf{x} + b = 0,$$

for some normal vector $\mathbf{w} \in \mathbb{R}^N$ and offset $b \in \mathbb{R}$. Let $d_p(\mathbf{x}, \mathcal{H})$ denote the distance of $\mathbf{x}$ to the hyperplane $\mathcal{H}$, that is,

$$d_p(\mathbf{x}, \mathcal{H}) = \inf_{\mathbf{x}' \in \mathcal{H}} \|\mathbf{x}' - \mathbf{x}\|_p. \tag{A.12}$$

Then, the following identity holds for all $p \geq 1$:

$$d_p(\mathbf{x}, \mathcal{H}) = \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|_q}, \tag{A.13}$$

where q is the conjugate of p: $\frac{1}{p} + \frac{1}{q} = 1$. (A.13) can be shown by a straightforward application of the results of appendix B to the constrained optimization problem (A.12).

A.1.3 Relationship between norms

A general form for the inequalities seen in equations (A.3), (A.4) and (A.5), which holds for all L_p norms, is shown in the following proposition.

Proposition A.6 *Let $1 \leq p \leq q$. Then the following inequalities hold for all $\mathbf{x} \in \mathbb{R}^N$:*

$$\|x\|_q \leq \|x\|_p \leq N^{\frac{1}{p}-\frac{1}{q}} \|x\|_q \,. \tag{A.14}$$

Proof: First, assume $\mathbf{x} \neq \mathbf{0}$, otherwise the inequalities hold trivially. Then the first inequality holds using $1 \leq p \leq q$ as follows:

$$\left[\frac{\|\mathbf{x}\|_p}{\|\mathbf{x}\|_q}\right]^p = \sum_{i=1}^{N} \left[\underbrace{\frac{x_i}{\|\mathbf{x}\|_q}}_{\leq 1}\right]^p \geq \sum_{i=1}^{N} \left[\frac{x_i}{\|\mathbf{x}\|_q}\right]^q = 1.$$

Finally, the second inequality follows by using Hölder's inequality (proposition A.4)

$$\|\mathbf{x}\|_p = \left[\sum_{i=1}^{N} |x_i|^p\right]^{\frac{1}{p}} \leq \left[\left(\sum_{i=1}^{N} (|x_i|^p)^{\frac{q}{p}}\right)^{\frac{p}{q}} \left(\sum_{i=1}^{N} (1)^{\frac{q}{q-p}}\right)^{1-\frac{p}{q}}\right]^{\frac{1}{p}} = \|\mathbf{x}\|_q N^{\frac{1}{p}-\frac{1}{q}} \,,$$

which completes the proof. □

A.2 Matrices

For a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ with m rows and n columns, we denote by $\mathbf{M}_{ij}$ its ijth entry, for all $i \in [m]$ and $j \in [n]$. For any $m \geq 1$, we denote by $\mathbf{I}_m$ the m-dimensional identity matrix, and refer to it as $\mathbf{I}$ when the dimension is clear from the context.

The *transpose* of $\mathbf{M}$ is denoted by $\mathbf{M}^\top$ and defined by $(\mathbf{M}^\top)_{ij} = \mathbf{M}_{ji}$ for all (i,j). For any two matrices $\mathbf{M} \in \mathbb{R}^{m \times n}$ and $\mathbf{N} \in \mathbb{R}^{n \times p}$, $(\mathbf{MN})^\top = \mathbf{N}^\top \mathbf{M}^\top$. $\mathbf{M}$ is said to be *symmetric* iff $\mathbf{M}_{ij} = \mathbf{M}_{ji}$ for all (i,j), that is, iff $\mathbf{M} = \mathbf{M}^\top$.

The *trace* of a square matrix $\mathbf{M}$ is denoted by $\mathrm{Tr}[\mathbf{M}]$ and defined as $\mathrm{Tr}[\mathbf{M}] = \sum_{i=1}^N \mathbf{M}_{ii}$. For any two matrices $\mathbf{M} \in \mathbb{R}^{m \times n}$ and $\mathbf{N} \in \mathbb{R}^{n \times m}$, the following identity holds: $\mathrm{Tr}[\mathbf{MN}] = \mathrm{Tr}[\mathbf{NM}]$. More generally, the following cyclic property holds with the appropriate dimensions for the matrices $\mathbf{M}$, $\mathbf{N}$, and $\mathbf{P}$:

$$\mathrm{Tr}[\mathbf{MNP}] = \mathrm{Tr}[\mathbf{PMN}] = \mathrm{Tr}[\mathbf{NPM}]. \tag{A.15}$$

The inverse of a square matrix $\mathbf{M}$, which exists when $\mathbf{M}$ has full rank, is denoted by $\mathbf{M}^{-1}$ and is the unique matrix satisfying $\mathbf{M}\mathbf{M}^{-1} = \mathbf{M}^{-1}\mathbf{M} = \mathbf{I}$.

A.2.1 Matrix norms

A *matrix norm* is a norm defined over $\mathbb{R}^{m \times n}$ where m and n are the dimensions of the matrices considered. Many matrix norms, including those discussed below, satisfy the following *submultiplicative property*:

$$\|\mathbf{MN}\| \leq \|\mathbf{M}\|\|\mathbf{N}\|. \tag{A.16}$$

The *matrix norm induced* by the vector norm $\|\cdot\|_p$ or the *operator norm induced* by that norm is also denoted by $\|\cdot\|_p$ and defined by

$$\|\mathbf{M}\|_p = \sup_{\|\mathbf{x}\|_p \leq 1} \|\mathbf{Mx}\|_p \,. \tag{A.17}$$

The norm induced for $p = 2$ is known as the *spectral norm*, which equals the largest singular value of $\mathbf{M}$ (see section A.2.2), or the square-root of the largest eigenvalue of $\mathbf{M}^\top\mathbf{M}$:

$$\|\mathbf{M}\|_2 = \sigma_1(\mathbf{M}) = \sqrt{\lambda_{\max}(\mathbf{M}^\top\mathbf{M})}. \tag{A.18}$$

Not all matrix norms are induced by vector norms. The *Frobenius norm* denoted by $\|\cdot\|_F$ is the most notable of such norms and is defined by:

$$\|\mathbf{M}\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbf{M}_{ij}^2\right)^{1/2}.$$

The Frobenius norm can be interpreted as the L_2 norm of a vector when treating $\mathbf{M}$ as a vector of size mn. It also coincides with the norm induced by the *Frobenius product*, which is the inner product defined for all $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{m\times n}$ by

$$\langle \mathbf{M}, \mathbf{N}\rangle_F = \mathrm{Tr}[\mathbf{M}^\top \mathbf{N}]. \tag{A.19}$$

This relates the Frobenius norm to the singular values of $\mathbf{M}$:

$$\|\mathbf{M}\|_F^2 = \mathrm{Tr}[\mathbf{M}^\top\mathbf{M}] = \sum_{i=1}^{r}\sigma_i(\mathbf{M})^2,$$

where $r = \mathrm{rank}(\mathbf{M})$. The second equality follows from properties of SPSD matrices (see section A.2.3).

For any $j \in [n]$, let $\mathbf{M}_j$ denote the jth column of $\mathbf{M}$, that is $\mathbf{M} = [\mathbf{M}_1 \cdots \mathbf{M}_n]$. Then, for any $p, r \geq 1$, the $L_{p,r}$ *group norm* of $\mathbf{M}$ is defined by

$$\|\mathbf{M}\|_{p,r} = \left(\sum_{j=1}^{n}\|\mathbf{M}_i\|_p^r\right)^{1/r}.$$

One of the most commonly used group norms is the $L_{2,1}$ norm defined by

$$\|\mathbf{M}\|_{2,1} = \sum_{i=1}^{n}\|\mathbf{M}_i\|_2.$$

A.2.2 Singular value decomposition

The compact *singular value decomposition (SVD)* of $\mathbf{M}$, with $r = \mathrm{rank}(\mathbf{M}) \leq \min(m, n)$, can be written as follows:

$$\mathbf{M} = \mathbf{U}_M\mathbf{\Sigma}_M\mathbf{V}_M^\top.$$

The $r\times r$ matrix $\mathbf{\Sigma}_M = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ is diagonal and contains the non-zero *singular values* of $\mathbf{M}$ sorted in decreasing order, that is $\sigma_1 \geq \ldots \geq \sigma_r > 0$. The matrices $\mathbf{U}_M \in \mathbb{R}^{m\times r}$ and $\mathbf{V}_M \in \mathbb{R}^{n\times r}$ have orthonormal columns that contain the *left* and *right singular vectors* of $\mathbf{M}$ corresponding to the sorted singular values. We denote by $\mathbf{U}_k \in \mathbb{R}^{m\times k}$ the top $k \leq r$ left singular vectors of $\mathbf{M}$.

The *orthogonal projection* onto the span of $\mathbf{U}_k$ can be written as $\mathbf{P}_{U_k} = \mathbf{U}_k\mathbf{U}_k^\top$, where $\mathbf{P}_{U_k}$ is SPSD and idempotent, i.e., $\mathbf{P}_{U_k}^2 = \mathbf{P}_{U_k}$. Moreover, the orthogonal projection onto the subspace orthogonal to $\mathbf{U}_k$ is defined as $\mathbf{P}_{U_k,\perp}$. Similar definitions, i.e., $\mathbf{V}_k, \mathbf{P}_{V_k}, \mathbf{P}_{V_k,\perp}$, hold for the right singular vectors.

The *generalized inverse*, or *Moore-Penrose pseudo-inverse* of a matrix $\mathbf{M}$ is denoted by $\mathbf{M}^\dagger$ and defined by

$$\mathbf{M}^\dagger = \mathbf{U}_M\mathbf{\Sigma}_M^\dagger\mathbf{V}_M^\top, \tag{A.20}$$

where $\mathbf{\Sigma}_M^\dagger = \mathrm{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1})$. For any square $m\times m$ matrix $\mathbf{M}$ with full rank, i.e., $r = m$, the pseudo-inverse coincides with the matrix inverse: $\mathbf{M}^\dagger = \mathbf{M}^{-1}$.

A.2.3 Symmetric positive semidefinite (SPSD) matrices

Definition A.7 *A symmetric matrix* $\mathbf{M} \in \mathbb{R}^{m\times m}$ *is said to be* positive semidefinite *iff*

$$\mathbf{x}^\top\mathbf{M}\mathbf{x} \geq 0 \tag{A.21}$$

for all $\mathbf{x} \in \mathbb{R}^m$. $\mathbf{M}$ *is said to be* positive definite *if the inequality is strict.*

Kernel matrices (see chapter 6) and orthogonal projection matrices are two examples of SPSD matrices. It is straightforward to show that a matrix $\mathbf{M}$ is SPSD iff its eigenvalues are all non-negative. Furthermore, the following properties hold for any SPSD matrix $\mathbf{M}$:

- $\mathbf{M}$ admits a decomposition $\mathbf{M} = \mathbf{X}^\top\mathbf{X}$ for some matrix $\mathbf{X}$ and the *Cholesky decomposition* provides one such decomposition in which $\mathbf{X}$ is an upper triangular matrix.
- The left and right singular vectors of $\mathbf{M}$ are the same and the SVD of $\mathbf{M}$ is also its eigenvalue decomposition.
- The SVD of an arbitrary matrix $\mathbf{X} = \mathbf{U}_X\mathbf{\Sigma}_X\mathbf{V}_X^\top$ defines the SVD of two related SPSD matrices: the left singular vectors ($\mathbf{U}_X$) are the eigenvectors of $\mathbf{X}\mathbf{X}^\top$, the right singular vectors ($\mathbf{V}_X$) are the eigenvectors of $\mathbf{X}^\top\mathbf{X}$ and the non-zero singular values of $\mathbf{X}$ are the square roots of the non-zero eigenvalues of $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{X}^\top\mathbf{X}$.
- The trace of $\mathbf{M}$ is the sum of its singular values, i.e., $\text{Tr}[\mathbf{M}] = \sum_{i=1}^r \sigma_i(\mathbf{M})$, where $\text{rank}(\mathbf{M}) = r$.
- The top singular vector of $\mathbf{M}$, $\mathbf{u}_1$, maximizes the *Rayleigh quotient*, which is defined as
$$r(\mathbf{x}, \mathbf{M}) = \frac{\mathbf{x}^\top\mathbf{M}\mathbf{x}}{\mathbf{x}^\top\mathbf{x}}\,.$$
In other words, $\mathbf{u}_1 = \text{argmax}_{\mathbf{x}}\, r(\mathbf{x}, \mathbf{M})$ and $r(\mathbf{u}, \mathbf{M}) = \sigma_1(\mathbf{M})$. Similarly, if $\mathbf{M}' = \mathbf{P}_{U_i,\perp}\mathbf{M}$, that is, the projection of $\mathbf{M}$ onto the subspace orthogonal to $\mathbf{U}_i$, then $\mathbf{u}_{i+1} = \text{argmax}_{\mathbf{x}}\, r(\mathbf{x}, \mathbf{M}')$, where $\mathbf{u}_{i+1}$ is the $(i+1)$st singular vector of $\mathbf{M}$.

B Convex Optimization

In this appendix, we introduce the main definitions and results of convex optimization needed for the analysis of the learning algorithms presented in this book.

B.1 Differentiation and unconstrained optimization

We start with some basic definitions for differentiation needed to present Fermat's theorem and to describe some properties of convex functions.

Definition B.1 (Gradient) *Let $f\colon \mathcal{X} \subseteq \mathbb{R}^N \to \mathbb{R}$ be a differentiable function. Then, the* gradient *of f at $\mathbf{x} \in \mathcal{X}$ is the vector in $\mathbb{R}^N$ denoted by $\nabla f(\mathbf{x})$ and defined by*

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial \mathbf{x}_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial \mathbf{x}_N}(\mathbf{x}) \end{bmatrix}.$$

Definition B.2 (Hessian) *Let $f\colon \mathcal{X} \subseteq \mathbb{R}^N \to \mathbb{R}$ be a twice differentiable function. Then, the* Hessian *of f at $\mathbf{x} \in \mathcal{X}$ is the matrix in $\mathbb{R}^{N \times N}$ denoted by $\nabla^2 f(\mathbf{x})$ and defined by*

$$\nabla^2 f(\mathbf{x}) = \Big[\frac{\partial^2 f}{\partial \mathbf{x}_i, \mathbf{x}_j}(\mathbf{x})\Big]_{1 \leq i,j \leq N}.$$

Next, we present a classic result for unconstrained optimization.

Theorem B.3 (Fermat's theorem) *Let $f\colon \mathcal{X} \subseteq \mathbb{R}^N \to \mathbb{R}$ be a differentiable function. If f admits a local extremum at $\mathbf{x}^* \in \mathcal{X}$, then $\nabla f(\mathbf{x}^*) = 0$, that is, $\mathbf{x}^*$ is a* stationary point.

B.2 Convexity

This section introduces the notions of *convex sets* and *convex functions*. Convex functions play an important role in the design and analysis of learning algorithms, in part because a local minimum of a convex function is necessarily also a global minimum. Thus, the properties of a hypothesis that is learned by finding a local minimum of a convex optimization are often well understood, while for some non-convex optimization problems there may be a very large number of local minima for which no clear characterization of the learned hypothesis can be given.

Definition B.4 (Convex set) *A set $\mathcal{X} \subseteq \mathbb{R}^N$ is said to be* convex *if for any two points $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ the segment $[\mathbf{x}, \mathbf{y}]$ lies in $\mathcal{X}$, that is*

$$\{\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \colon 0 \leq \alpha \leq 1\} \subseteq \mathcal{X}.$$

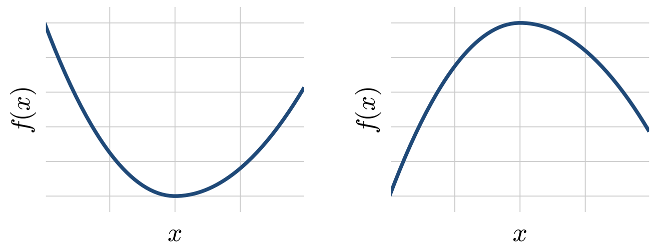


Figure B.1
Examples of a convex (left) and a concave (right) functions. Note that any line segment drawn between two points on the convex function lies entirely above the graph of the function while any line segment drawn between two points on the concave function lies entirely below the graph of the function.

The following lemma illustrates several operations on convex sets that preserve convexity. These will be useful for proving several subsequent results of this section.

Lemma B.5 (Operations that preserve convexity of sets) *The following operations on convex sets preserve convexity:*

- *Let $\{\mathcal{C}_i\}_{i\in I}$ be any family of sets where for all $i \in I$ the set $\mathcal{C}_i$ is convex. Then the intersection of these sets $\bigcap_{i\in I}\mathcal{C}_i$ is also convex.*
- *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be convex sets, then their sum $\mathcal{C}_1 + \mathcal{C}_2 = \{x_1 + x_2 : x_1 \in \mathcal{C}_1, x_2 \in \mathcal{C}_2\}$, when defined, is convex.*
- *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be convex sets, then their cross-product $(\mathcal{C}_1 \times \mathcal{C}_2)$ is also convex.*
- *Any projection of a convex set $\mathcal{C}$ is also convex.*

Proof: The first property holds since for any $x, y \in \bigcap_{i\in I}\mathcal{C}_i$ and any $\alpha \in [0,1]$, we have $\alpha x + (1-\alpha)y \in \mathcal{C}_i$ for any $i \in I$ by the convexity of $\mathcal{C}_i$.

The second property holds since for any $(x_1 + x_2), (y_1 + y_2) \in (\mathcal{C}_1 + \mathcal{C}_2)$ we have $\alpha(x_1 + x_2) + (1-\alpha)(y_1 + y_2) = (\alpha x_1 + (1-\alpha)y_1 + \alpha x_2 + (1-\alpha)y_2) \in (\mathcal{C}_1 + \mathcal{C}_2)$, which follows since $\alpha x_1 + (1-\alpha)y_1 \in \mathcal{C}_1$ and $\alpha x_2 + (1-\alpha)y_2 \in \mathcal{C}_2$.

The third property holds since for $(x_1, x_2), (y_1, y_2) \in (\mathcal{C}_1 \times \mathcal{C}_2)$ we have $\alpha(x_1, x_2) + (1-\alpha)(y_1, y_2) = (\alpha x_1 + (1-\alpha)y_1, \alpha x_2 + (1-\alpha)y_2) \in (\mathcal{C}_1 \times \mathcal{C}_2)$, where the membership holds due to the assumption that $\mathcal{C}_1$ and $\mathcal{C}_2$ are convex.

Finally, the fourth property holds by noting that for any decomposition of the convex set $\mathcal{C}$ into projections $\mathcal{C}_1$ and $\mathcal{C}_2$, such that $\mathcal{C} = (\mathcal{C}_1 \times \mathcal{C}_2)$, it must be the case that $\mathcal{C}_1$ is convex. If $\mathcal{C}_2$ is empty, then the result is trivially true. Otherwise, fix an element $x_2 \in \mathcal{C}_2$, then for any $x, y \in \mathcal{C}_1$ and any $\alpha \in [0,1]$ we have $\alpha(x, x_2) + (1-\alpha)(y, x_2) \in \mathcal{C}$, which implies $\alpha x + (1-\alpha)y \in \mathcal{C}_1$. Since $\mathcal{C}_1$ was chosen arbitrarily, this fact holds for any projection of $\mathcal{C}$. □

Note that many set operations may not preserve convexity. Consider the union of disjoint intervals on $\mathbb{R}$: $[a,b] \cup [c,d]$ where $a < b < c < d$. Clearly $[a,b]$ and $[c,d]$ are convex, however we have $\frac{1}{2}b + (1 - \frac{1}{2})c \notin ([a,b] \cup [c,d])$.

Definition B.6 (Convex hull) *The* convex hull $\text{conv}(\mathcal{X})$ *of a set of points $\mathcal{X} \subseteq \mathbb{R}^N$ is the minimal convex set containing $\mathcal{X}$ and can be equivalently defined as follows:*

$$\text{conv}(\mathcal{X}) = \Big\{\sum_{i=1}^{m} \alpha_i \mathbf{x}_i \colon m \geq 1, \forall i \in [m], \mathbf{x}_i \in \mathcal{X}, \alpha_i \geq 0, \sum_{i=1}^{m} \alpha_i = 1\Big\}. \tag{B.1}$$

Let $\text{Epi}\, f$ denote the *epigraph* of function $f \colon \mathcal{X} \to \mathbb{R}$, that is the set of points lying above its graph: $\{(x,y) \colon x \in \mathcal{X}, y \geq f(x)\}$.

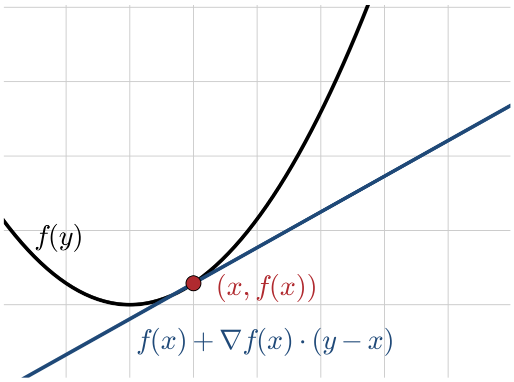


Figure B.2
Illustration of the first-order property satisfied by all convex functions.

Definition B.7 (Convex function) *Let $\mathcal{X}$ be a convex set. A function $f\colon \mathcal{X} \to \mathbb{R}$ is said to be* convex *iff* $\operatorname{Epi} f$ *is a convex set, or, equivalently, if for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and $\alpha \in [0, 1]$,*

$$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) \le \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y})\,. \tag{B.2}$$

f is said to be *strictly convex* if inequality (B.2) is strict for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ where $\mathbf{x} \neq \mathbf{y}$ and $\alpha \in (0, 1)$. f is said to be (strictly) *concave* when $-f$ is (strictly) convex. Figure B.1 shows simple examples of convex and concave functions. Convex functions can also be characterized in terms of their first- or second-order differential.

Theorem B.8 *Let f be a differentiable function, then f is convex if and only if* $\operatorname{dom}(f)$ *is convex and the following inequalities hold:*

$$\forall \mathbf{x}, \mathbf{y} \in \operatorname{dom}(f),\;\; f(\mathbf{y}) - f(\mathbf{x}) \ge \nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x})\,. \tag{B.3}$$

The property (B.3) is illustrated by figure B.2: for a convex function, the hyperplane tangent at $\mathbf{x}$ is always below the graph.

Theorem B.9 *Let f be a twice differentiable function, then f is convex iff* $\operatorname{dom}(f)$ *is convex and its Hessian is positive semidefinite:*

$$\forall \mathbf{x} \in \operatorname{dom}(f),\;\; \nabla^2 f(\mathbf{x}) \succeq 0\,.$$

Recall that a symmetric matrix is positive semidefinite if all of its eigenvalues are non-negative. Further, note that when f is scalar, this theorem states that f is convex if and only if its second derivative is always non-negative, that is, for all $x \in \operatorname{dom}(f)$, $f''(x) \ge 0$.

Example B.10 (Linear functions) Any linear function f is both convex and concave, since equation (B.2) holds with equality for both f and $-f$ by the definition of linearity.

Example B.11 (Quadratic function) The function $f\colon x \mapsto x^2$ defined over $\mathbb{R}$ is convex since it is twice differentiable and for all $x \in \mathbb{R}$, $f''(x) = 2 > 0$.

Example B.12 (Norms) Any norm $\|\cdot\|$ defined over a convex set $\mathcal{X}$ is convex since by the triangle inequality and the homogeneity property of the norm, for all $\alpha \in [0, 1], \mathbf{x}, \mathbf{y} \in \mathcal{X}$, we can write

$$\|\alpha\mathbf{x} + (1-\alpha)\mathbf{y}\| \le \|\alpha\mathbf{x}\| + \|(1-\alpha)\mathbf{y}\| = \alpha\|\mathbf{x}\| + (1-\alpha)\|\mathbf{y}\|\,.$$

Example B.13 (Maximum function) The max function defined for all $\mathbf{x} \in \mathbb{R}^N$, by $\mathbf{x} \mapsto \max_{j \in [N]} \mathbf{x}_j$ is convex. For all $\alpha \in [0,1], \mathbf{x}, \mathbf{y} \in \mathbb{R}^N$, by the sub-additivity of max, we can write

$$\max_j(\alpha \mathbf{x}_j + (1-\alpha)\mathbf{y}_j) \leq \max_j(\alpha \mathbf{x}_j) + \max_j((1-\alpha)\mathbf{y}_j) = \alpha \max_j(\mathbf{x}_j) + (1-\alpha)\max_j(\mathbf{y}_j)\,.$$

One useful approach for proving convexity or concavity of functions is to make use of composition rules. For simplicity of presentation, we will assume twice differentiability, although the results can also be proven without this assumption.

Lemma B.14 (Composition of convex/concave functions) *Assume $h\colon \mathbb{R} \to \mathbb{R}$ and $g\colon \mathbb{R}^N \to \mathbb{R}$ are twice differentiable functions and for all $\mathbf{x} \in \mathbb{R}^N$, define $f(\mathbf{x}) = h(g(\mathbf{x}))$. Then the following implications are valid:*

- *h is convex and non-decreasing, and g is convex $\implies$ f is convex.*
- *h is convex and non-increasing, and g is concave $\implies$ f is convex.*
- *h is concave and non-decreasing, and g is concave $\implies$ f is concave.*
- *h is concave and non-increasing, and g is convex $\implies$ f is concave.*

Proof: We restrict ourselves to $N = 1$, since it suffices to prove convexity (concavity) along all arbitrary lines that intersect the domain. Now, consider the second derivative of f:

$$f''(x) = h''(g(x))g'(x)^2 + h'(g(x))g''(x)\,. \tag{B.4}$$

Note that if h is convex and non-decreasing, we have $h'' \geq 0$ and $h' \geq 0$. Furthermore, if g is convex we also have $g'' \geq 0$, and it follows that $f''(x) \geq 0$, which proves the first statement. The remainder of the statements are proven in a similar manner. □

Example B.15 (Composition of functions) The previous lemma shows the convexity or concavity of the following composed functions:

- If $f\colon \mathbb{R}^N \to \mathbb{R}$ is convex, then $\exp(f)$ is convex.
- Any squared norm $\|\cdot\|^2$ is convex.
- For all $\mathbf{x} \in \mathbb{R}^N$ the function $\mathbf{x} \mapsto \log(\sum_{j=1}^N x_j)$ is concave.

The following two lemmas give examples of two other operations preserving convexity.

Lemma B.16 (Pointwise supremum or maximum of convex functions) *Let $(f_i)_{i \in \mathcal{I}}$ be a family of convex functions defined over a convex set $\mathcal{C}$. Then, their* pointwise supremum *f defined for all $x \in \mathcal{C}$ by $f(x) = \sup_{i \in \mathcal{I}} f_i(x)$ (resp. their* pointwise maximum *if $|\mathcal{I}| < +\infty$) is a convex function.*

Proof: Observe that $\operatorname{Epi} f = \cap_{i \in \mathcal{I}} \operatorname{Epi} f_i$ and is therefore convex as an intersection of convex sets. □

Example B.17 (Pointwise supremum of convex functions) The lemma shows in particular the convexity of the following functions:

- A piecewise linear function f defined for all $x \in \mathbb{R}^N$ by $f(x) = \max_{i \in [m]} \mathbf{w}_i^\top \mathbf{x} + b_i$ is convex as a pointwise maximum of affine (and thus convex) functions.
- The maximum eigenvalue $\lambda_{\max}(\mathbf{M})$ is a convex function over the set of symmetric matrices $\mathbf{M}$ since the set of symmetric matrices is convex and since $\lambda_{\max}(\mathbf{M}) = \sup_{\|\mathbf{x}\|_2 \leq 1} \mathbf{x}^\top \mathbf{M} \mathbf{x}$ is defined as the supremum of the linear (and thus convex) functions $\mathbf{M} \mapsto \mathbf{x}^\top \mathbf{M} \mathbf{x}$.
- More generally, let $\lambda_1(\mathbf{M}), \ldots, \lambda_k(\mathbf{M})$ denote the top $k \leq n$ eigenvalues of a symmetric $n \times n$ matrix $\mathbf{M}$. Then, by a similar argument, $\mathbf{M} \mapsto \sum_{i=1}^k \lambda_i(\mathbf{M})$ is a convex function since $\sum_{i=1}^k \lambda_i(\mathbf{M}) = \sup_{\dim(\mathbf{V})=k} \sum_{i=1}^k \mathbf{u}_i^\top \mathbf{M} \mathbf{u}_i$, where $\mathbf{u}_1, \ldots, \mathbf{u}_k$ is an orthonormal basis of $\mathbf{V}$.
- Using the previous property, along with the fact that $\operatorname{Tr}(\mathbf{M})$ is linear in $\mathbf{M}$, also shows that $\mathbf{M} \mapsto \sum_{i=k+1}^n \lambda_i(\mathbf{M}) = \operatorname{Tr}(\mathbf{M}) - \sum_{i=1}^k \lambda_i(\mathbf{M})$ or $\mathbf{M} \mapsto \sum_{i=n-k+1}^n \lambda_i(\mathbf{M}) = -\sum_{i=1}^k \lambda_i(-\mathbf{M})$ are concave functions.

Lemma B.18 (Partial infimum) *Let f be a convex function defined over a convex set $\mathcal{C} \subseteq \mathcal{X} \times \mathcal{Y}$ and let $\mathcal{B} \subseteq \mathcal{Y}$ be a convex set such that $\mathcal{A} = \{x \in \mathcal{X} \colon \exists y \in \mathcal{B} \mid (x, y) \in \mathcal{C}\}$ is non-empty. Then, $\mathcal{A}$ is a convex set and the function g defined for all $x \in \mathcal{A}$ by $g(x) = \inf_{y \in \mathcal{B}} f(x, y)$ is convex.*

Proof: First note that the intersection of the convex sets $\mathcal{C}$ and $(\mathcal{X} \times \mathcal{B})$ is convex. Thus, $\mathcal{A}$ is convex since it is the projection of the convex set $\mathcal{C} \cap (\mathcal{X} \times \mathcal{B})$ onto $\mathcal{X}$.

Let x_1 and x_2 be in $\mathcal{A}$. By definition of g, for any $\epsilon > 0$, there exist $y_1, y_2 \in \mathcal{B}$ with $(x_1, y_1), (x_2, y_2) \in \mathcal{C}$ such that $f(x_1, y_1) \leq g(x_1) + \epsilon$ and $f(x_2, y_2) \leq g(x_2) + \epsilon$. Then, for any $\alpha \in [0, 1]$,

$$\begin{aligned} g(\alpha x_1 + (1-\alpha)x_2) &= \inf_{y \in \mathcal{B}} f(\alpha x_1 + (1-\alpha)x_2, y) \\ &\leq f(\alpha x_1 + (1-\alpha)x_2, \alpha y_1 + (1-\alpha)y_2) \\ &\leq \alpha f(x_1, y_1) + (1-\alpha) f(x_2, y_2) \\ &\leq \alpha g(x_1) + (1-\alpha) g(x_2) + \epsilon. \end{aligned}$$

Since the inequality holds for all $\epsilon > 0$, it implies

$$g(\alpha x_1 + (1-\alpha)x_2) \leq \alpha g(x_1) + (1-\alpha) g(x_2),$$

which completes the proof. □

Example B.19 The lemma shows in particular that the distance to a convex set $\mathcal{B}$, $d(x, \mathcal{B}) = \inf_{y \in \mathcal{B}} \|x - y\|$, is a convex function of x in any normed vector space, since $(x, y) \mapsto \|x - y\|$ is jointly convex in x and y for any norm $\|\cdot\|$.

The following is a useful inequality applied in a variety of contexts. It is in fact a quasi-direct consequence of the definition of convexity.

Theorem B.20 (Jensen's inequality) *Let X be a random variable taking values in a non-empty convex set $\mathcal{C} \subseteq \mathbb{R}^N$ with a finite expectation $\mathbb{E}[X]$, and f a measurable convex function defined over $\mathcal{C}$. Then, $\mathbb{E}[X]$ is in $\mathcal{C}$, $\mathbb{E}[f(X)]$ is finite, and the following inequality holds:*

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)].$$

Proof: We give a sketch of the proof, which essentially follows from the definition of convexity. Note that for any finite set of elements $x_1, \ldots, x_n$ in $\mathcal{C}$ and any positive reals $\alpha_1, \ldots, \alpha_n$ such that $\sum_{i=1}^n \alpha_i = 1$, we have

$$f\Big(\sum_{i=1}^n \alpha_i x_i\Big) \leq \sum_{i=1}^n \alpha_i f(x_i)\,.$$

This follows straightforwardly by induction from the definition of convexity. Since the α_is can be interpreted as probabilities, this immediately proves the inequality for any distribution with a finite support defined by $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$:

$$f(\mathop{\mathbb{E}}_{\boldsymbol{\alpha}}[X]) \leq \mathop{\mathbb{E}}_{\boldsymbol{\alpha}}[f(X)]\,.$$

Extending this to arbitrary distributions can be shown via the continuity of f on any open set, which is guaranteed by the convexity of f, and the weak density of distributions with finite support in the family of all probability measures. □

B.3 Constrained optimization

We now define a general constrained optimization problem and the specific properties associated to convex constrained optimization problems.

Definition B.21 (Constrained optimization problem) *Let $\mathcal{X} \subseteq \mathbb{R}^N$ and $f, g_i \colon \mathcal{X} \to \mathbb{R}$, for all $i \in [m]$. Then, a* constrained optimization problem *has the form:*

$$\begin{aligned} &\min_{\mathbf{x} \in \mathcal{X}} \quad f(\mathbf{x}) \\ &\textit{subject to:} \quad g_i(\mathbf{x}) \leq 0, \ \forall i \in \{1, \ldots, m\}. \end{aligned}$$

This general formulation does not make any convexity assumptions and can be augmented with equality constraints. It is referred to as the *primal problem* in contrast with a related problem introduced later. We will denote by p^* the optimal value of the objective.

For any $\mathbf{x} \in \mathcal{X}$, we will denote by $g(\mathbf{x})$ the vector $(g_1(\mathbf{x}), \ldots, g_m(\mathbf{x}))^\top$. Thus, the constraints can be written as $g(\mathbf{x}) \leq \mathbf{0}$. To any constrained optimization problem, we can associate a *Lagrange function* that plays an important role in the analysis of the problem and its relationship with another related optimization problem.

Definition B.22 (Lagrangian) *The* Lagrange function *or the* Lagrangian *associated to the general constrained optimization problem defined in (B.21) is the function defined over $\mathcal{X} \times \mathbb{R}_+$ by:*

$$\forall \mathbf{x} \in \mathcal{X}, \forall \boldsymbol{\alpha} \geq 0, \quad \mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = f(\mathbf{x}) + \sum_{i=1}^{m} \alpha_i g_i(\mathbf{x}),$$

where the variables α_i are known as the Lagrange *or* dual *variables with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^\top$.*

Any equality constraint of the form $g(\mathbf{x}) = 0$ for a function g can be equivalently expressed by two inequalities: $-g(\mathbf{x}) \leq 0$ and $+g(\mathbf{x}) \leq 0$. Let $\alpha_- \geq 0$ be the Lagrange variable associated to the first constraint and $\alpha_+ \geq 0$ the one associated to the second constraint. The sum of the terms corresponding to these constraints in the definition of the Lagrange function can therefore be written as $\alpha g(\mathbf{x})$ with $\alpha = (\alpha_+ - \alpha_-)$. Thus, in general, for an equality constraint $g(\mathbf{x}) = 0$ the Lagrangian is augmented with a term $\alpha g(\mathbf{x})$ but with $\alpha \in \mathbb{R}$ not constrained to be non-negative. Note that in the case of a convex optimization problem, equality constraints $g(\mathbf{x})$ are required to be affine since both $g(\mathbf{x})$ and $-g(\mathbf{x})$ are required to be convex.

Definition B.23 (Dual function) *The* (Lagrange) dual function *associated to the constrained optimization problem is defined by*

$$\forall \boldsymbol{\alpha} \geq 0, F(\boldsymbol{\alpha}) = \inf_{\mathbf{x} \in \mathcal{X}} \mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = \inf_{\mathbf{x} \in \mathcal{X}} \big(f(\mathbf{x}) + \sum_{i=1}^{m} \alpha_i g_i(\mathbf{x})\big). \tag{B.5}$$

Note that F is always concave, since the Lagrangian is linear with respect to $\boldsymbol{\alpha}$ and since the infimum preserves concavity. We further observe that

$$\forall \boldsymbol{\alpha} \geq 0, \quad F(\boldsymbol{\alpha}) \leq p^*, \tag{B.6}$$

since for any feasible $\mathbf{x}$, $f(\mathbf{x}) + \sum_{i=1}^{m} \alpha_i g_i(\mathbf{x}) \leq f(\mathbf{x})$. The dual function naturally leads to the following optimization problem.

Definition B.24 (Dual problem) *The* dual (optimization) problem *associated to the constrained optimization problem is*

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \quad F(\boldsymbol{\alpha}) \\ &\textit{subject to:} \quad \boldsymbol{\alpha} \geq 0. \end{aligned}$$

The dual problem is always a convex optimization problem (as a maximization of a concave problem). Let d^* denote an optimal value. By (B.6), the following inequality always holds:

$$d^* \leq p^* \qquad \qquad (\textit{weak duality}).$$

The difference $(p^* - d^*)$ is known as the *duality gap*. The equality case

$$d^* = p^* \qquad \qquad (\textit{strong duality})$$

does not hold in general. However, strong duality does hold when convex problems satisfy a *constraint qualification*. We will denote by $\text{int}(\mathcal{X})$ the interior of the set $\mathcal{X}$.

Definition B.25 (Strong constraint qualification) *Assume that* $\operatorname{int}(\mathcal{X}) \neq \emptyset$. *Then, the* strong constraint qualification *or* Slater's condition *is defined as*

$$\exists\, \overline{\mathbf{x}} \in \operatorname{int}(\mathcal{X})\colon g(\overline{\mathbf{x}}) < 0. \tag{B.7}$$

A function $h\colon \mathcal{X} \to \mathbb{R}$ is said to be *affine* if it can be defined for all $\mathbf{x} \in \mathcal{X}$ by $h(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, for some $\mathbf{w} \in \mathbb{R}^N$ and $b \in \mathbb{R}$.

Definition B.26 (Weak constraint qualification) *Assume that* $\operatorname{int}(\mathcal{X}) \neq \emptyset$. *Then, the* weak constraint qualification *or* weak Slater's condition *is defined as*

$$\exists\, \overline{\mathbf{x}} \in \operatorname{int}(\mathcal{X})\colon \forall i \in [m], \big(g_i(\overline{\mathbf{x}}) < 0\big) \vee \big(g_i(\overline{\mathbf{x}}) = 0 \wedge g_i \text{ affine}\big). \tag{B.8}$$

We next present sufficient and necessary conditions for solutions to constrained optimization problems, based on the saddle point of the Lagrangian and Slater's condition.

Theorem B.27 (Saddle point — sufficient condition) *Let P be a constrained optimization problem over $\mathcal{X} = \mathbb{R}^N$. If $(\mathbf{x}^*, \boldsymbol{\alpha}^*)$ is a saddle point of the associated Lagrangian, that is,*

$$\forall \mathbf{x} \in \mathbb{R}^N, \forall \boldsymbol{\alpha} \geq 0, \quad \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}) \leq \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}^*) \leq \mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}^*), \tag{B.9}$$

then $\mathbf{x}^*$ is a solution of the problem P.

Proof: By the first inequality, the following holds:

$$\begin{aligned}\forall \boldsymbol{\alpha} \geq 0, \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}) \leq \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}^*) &\Rightarrow \forall \boldsymbol{\alpha} \geq 0, \boldsymbol{\alpha} \cdot g(\mathbf{x}^*) \leq \boldsymbol{\alpha}^* \cdot g(\mathbf{x}^*)\\ &\Rightarrow g(\mathbf{x}^*) \leq 0 \wedge \boldsymbol{\alpha}^* \cdot g(\mathbf{x}^*) = 0\,,\end{aligned} \tag{B.10}$$

where $g(\mathbf{x}^*) \leq 0$ in (B.10) follows by letting $\boldsymbol{\alpha} \to +\infty$ and $\boldsymbol{\alpha}^* \cdot g(\mathbf{x}^*) = 0$ follows by letting $\boldsymbol{\alpha} \to 0$. In view of (B.10), the second inequality in (B.9) gives,

$$\forall \mathbf{x}, \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}^*) \leq \mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}^*) \Rightarrow \forall \mathbf{x}, f(\mathbf{x}^*) \leq f(\mathbf{x}) + \boldsymbol{\alpha}^* \cdot g(\mathbf{x}).$$

Thus, for all $\mathbf{x}$ satisfying the constraints, that is $g(\mathbf{x}) \leq 0$, we have

$$f(\mathbf{x}^*) \leq f(\mathbf{x}),$$

which completes the proof. □

Theorem B.28 (Saddle point — necessary condition) *Assume that f and g_i, $i \in [m]$, are* convex functions *and that Slater's condition holds. Then, if $\mathbf{x}$ is a solution of the constrained optimization problem, there exists $\boldsymbol{\alpha} \geq 0$ such that $(\mathbf{x}, \boldsymbol{\alpha})$ is a saddle point of the Lagrangian.*

Theorem B.29 (Saddle point — necessary condition) *Assume that f and g_i, $i \in [m]$, are* convex differentiable functions *and that the weak Slater's condition holds. If $\mathbf{x}$ is a solution of the constrained optimization problem, then there exists $\boldsymbol{\alpha} \geq 0$ such that $(\mathbf{x}, \boldsymbol{\alpha})$ is a saddle point of the Lagrangian.*

We conclude with a theorem providing necessary and sufficient optimality conditions when the problem is convex, the objective function differentiable, and the constraints qualified.

Theorem B.30 (Karush-Kuhn-Tucker's theorem) *Assume that $f, g_i\colon \mathcal{X} \to \mathbb{R}, \forall i \in [m]$ are convex and differentiable and that the constraints are qualified. Then $\overline{\mathbf{x}}$ is a solution of the constrained program if and if only there exists $\overline{\boldsymbol{\alpha}} \geq 0$ such that,*

$$\nabla_{\mathbf{x}} \mathcal{L}(\overline{\mathbf{x}}, \overline{\boldsymbol{\alpha}}) = \nabla_{\mathbf{x}} f(\overline{\mathbf{x}}) + \overline{\boldsymbol{\alpha}} \cdot \nabla_{\mathbf{x}} g(\overline{\mathbf{x}}) = 0 \tag{B.11}$$

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}(\overline{\mathbf{x}}, \overline{\boldsymbol{\alpha}}) = g(\overline{\mathbf{x}}) \leq 0 \tag{B.12}$$

$$\overline{\boldsymbol{\alpha}} \cdot g(\overline{\mathbf{x}}) = \sum_{i=1}^{m} \overline{\boldsymbol{\alpha}}_i g_i(\overline{\mathbf{x}}) = 0\,. \tag{B.13}$$

The conditions B.11–B.13 are known as the KKT conditions. *Note that the last two KKT conditions are equivalent to*

$$g(\overline{\mathbf{x}}) \leq 0 \wedge (\forall i \in \{1, \ldots, m\}, \bar{\alpha}_i g_i(\overline{\mathbf{x}}) = 0). \tag{B.14}$$

These equalities are known as complementarity conditions.

Proof: For the forward direction, since the constraints are qualified, if $\overline{\mathbf{x}}$ is a solution, then there exists $\overline{\boldsymbol{\alpha}}$ such that the $(\overline{\mathbf{x}}, \overline{\boldsymbol{\alpha}})$ is a saddle point of the Lagrangian and all three conditions are satisfied (the first condition follows by definition of a saddle point, and the second two conditions follow from (B.10)).

In the opposite direction, if the conditions are met, then for any $\mathbf{x}$ such that $g(\mathbf{x}) \leq 0$, we can write

$$\begin{aligned}
f(\mathbf{x}) - f(\overline{\mathbf{x}}) &\geq \nabla_{\mathbf{x}} f(\overline{\mathbf{x}}) \cdot (\mathbf{x} - \overline{\mathbf{x}}) && \text{(convexity of } f\text{)}\\
&= -\sum_{i=1}^{m} \overline{\boldsymbol{\alpha}}_i \nabla_{\mathbf{x}} g_i(\overline{\mathbf{x}}) \cdot (\mathbf{x} - \overline{\mathbf{x}}) && \text{(first condition)}\\
&\geq -\sum_{i=1}^{m} \overline{\boldsymbol{\alpha}}_i [g_i(\mathbf{x}) - g_i(\overline{\mathbf{x}})] && \text{(convexity of } g_i\text{s)}\\
&= -\sum_{i=1}^{m} \overline{\boldsymbol{\alpha}}_i g_i(\mathbf{x}) \geq 0, && \text{(third condition)}
\end{aligned}$$

which shows that $f(\overline{\mathbf{x}})$ is the minimum of f over the set of points satisfying the constraints. □

B.4 Fenchel duality

In this section, we present an alternative theory of convex optimization or convex analysis where the functions f considered may be non-differentiable and take infinite values.

Throughout, this section, the set $\mathcal{X}$ denotes a Hilbert space with the inner product denoted by $\langle \cdot, \cdot \rangle$. However, the results presented can be straightforwardly extended to the case of a Banach space. We consider functions taking values in $[-\infty, +\infty]$. The *domain of a function* $f\colon \mathcal{X} \to [-\infty, +\infty]$ is defined as the set

$$\text{dom}(f) = \{x \in \mathcal{X}\colon f(x) < +\infty\}. \tag{B.15}$$

We extend the definition of convexity and say that $f\colon \mathcal{X} \to [-\infty, +\infty]$ is convex if it is convex over dom(f), that is if for all $x, x' \in \text{dom}(f)$ and all $t \in [0, 1]$,

$$f(tx + (1-t)x') \leq tu + (1-t)v, \tag{B.16}$$

for all $(u, v) \in \mathbb{R}^2$ with $u \geq f(x)$ and $v \geq f(x')$. A convex function is said to be *proper* if it takes values in $(-\infty, +\infty]$ and if it is not uniformly equal to $+\infty$. It is said to be *closed* when its epigraph is closed.

B.4.1 Subgradients

Definition B.31 *Let $f\colon \mathcal{X} \to (-\infty, +\infty]$ be a convex function. Then, a vector $g \in \mathcal{X}$ is a* subgradient *of f at a point $x \in \text{dom}(f)$ if the following inequality holds for all $z \in \mathcal{X}$:*

$$f(z) \geq f(x) + \langle z - x, g \rangle. \tag{B.17}$$

The set of all subgradients at x is called the subdifferential of f at x *and is denoted by $\partial f(x)$ with $\partial f(x) = \emptyset$ for $x \notin \text{dom}(f)$.*

Thus, g is a subgradient at x iff the hyperplane with normal vector g passing through the point $(x, f(x))$ is below the graph of f, that is iff it is supporting the graph of f. Figure 14.1 illustrates these definitions.

The following lemma shows that if f is differentiable at $x \in \text{dom}(f)$, then its subdifferential is reduced to its gradient at x.

Lemma B.32 *If f is differentiable at $x \in \text{dom}(f)$, then $\partial f(x) = \{\nabla f(x)\}$.*

Proof: Clearly, the gradient $\nabla f(x)$ is always a subgradient at x. Now, let g be in $\partial f(x)$. Then, by definition of a subgradient, for any $\epsilon \in \mathbb{R}$,

$$f(x + \epsilon(\nabla f(x) - g)) \geq f(x) + \epsilon\langle \nabla f(x) - g, g\rangle.$$

A first-order Taylor series expansion gives

$$f(x + \epsilon(\nabla f(x) - g)) - f(x) = \epsilon\langle \nabla f(x), \nabla f(x) - g\rangle + o(\epsilon\|\nabla f(x) - g\|).$$

In view of that, the first inequality can be rewritten as

$$\epsilon\|\nabla f(x) - g\|^2 \leq o(\epsilon\|\nabla f(x) - g\|),$$

which implies $\|\nabla f(x) - g\| = o(1)$ and $\nabla f(x) = g$. □

Proposition B.33 *Let $f \colon \mathcal{X} \to (-\infty, +\infty]$ be a proper function. Then, x^* is a global minimizer of f iff $\partial f(x^*)$ contains* 0.

Proof: Since f is proper, if x^* is a minimizer, $f(x^*)$ cannot be $+\infty$. Thus x^* must be in $\text{dom}(f)$ (and thus $\partial f(x)$ is not defined to be empty). Now, x^* is a global minimizer iff for all $z \in \mathcal{X}$, $f(z) \geq f(x^*)$, that is iff 0 is a subgradient of f at x^*. □

B.4.2 Core

The *core* of a set $\mathcal{C} \subseteq \mathcal{X}$ is denoted by $\text{core}(\mathcal{C})$ and defined as follows:

$$\text{core}(\mathcal{C}) = \{x \in \mathcal{C} \colon \forall u \in \mathcal{X}, \exists \epsilon > 0 \mid \forall t \in [0, \epsilon], (x + tu) \in \mathcal{C}\}. \tag{B.18}$$

Thus, for $x \in \text{core}(\mathcal{C})$, for any direction u, $(x + tu)$ is in $\mathcal{C}$ for t sufficiently small. In view of this definition, $\text{core}(\mathcal{C})$ clearly contains the interior of $\mathcal{C}$, $\text{int}(\mathcal{C})$.

Proposition B.34 *Let $h \colon \mathcal{X} \to [-\infty, +\infty]$ be a convex function. If there exists $x_0 \in \text{core}(\text{dom}(h))$ such that $h(x_0) > -\infty$, then $h(x) > -\infty$ for all $x \in \mathcal{X}$.*

Proof: Let x be in $\mathcal{X}$. Since x_0 is in $\text{core}(\text{dom}(h))$, there exists $t > 0$ such that $x'_0 = x_0 + t(x_0 - x)$ is in $\text{dom}(h)$, that is such that $h(x'_0) < +\infty$. Since $x_0 = \frac{1}{1+t}x'_0 + \frac{t}{1+t}x$, by convexity, the following holds:

$$h(x_0) \leq \frac{1}{1+t}h(x'_0) + \frac{t}{1+t}v \iff v \geq \frac{1+t}{t}\Big[h(x_0) - \frac{1}{1+t}h(x'_0)\Big] \tag{B.19}$$

for all $v \geq h(x)$. This implies $h(x) \geq \frac{1+t}{t}[h(x_0) - \frac{1}{1+t}h(x'_0)] > -\infty$, which concludes the proof. □

The proof of the following result is left as an exercise (Exercise B.3).

Proposition B.35 *Let $h \colon \mathcal{X} \to (-\infty, +\infty]$ be a convex function. Then, h admits a subdifferential at any $x \in \text{core}(\text{dom}(h))$.*

B.4.3 Conjugate functions

Definition B.36 (Conjugate functions) *Let $\mathcal{X}$ be a Hilbert space. The* conjugate function *or* Fenchel conjugate *of a function $f \colon \mathcal{X} \mapsto [-\infty, +\infty]$ is the function $f^* \colon \mathcal{X} \mapsto [-\infty, +\infty]$ defined by*

$$f^*(u) = \sup_{x \in \mathcal{X}} \big\{\langle u, x\rangle - f(x)\big\}. \tag{B.20}$$

Note that f^* is convex as the pointwise supremum of the set of affine and thus convex functions $u \mapsto \langle x, u\rangle - f(x)$. Also, if there exists x such that $f(x) < +\infty$, then $f > -\infty$. Conjugation is order-reversing: for any f and g, if $f \leq g$, then $g^* \leq f^*$. Also, it is straightforward to see that if f is closed proper convex, then $f^{**} = f$.

Figure B.3 illustrates the definition of conjugate functions. As shown by the figure, conjugate functions correspond to a dual description of the epigraph of a function in terms of supporting hyperplanes and their crossing points.

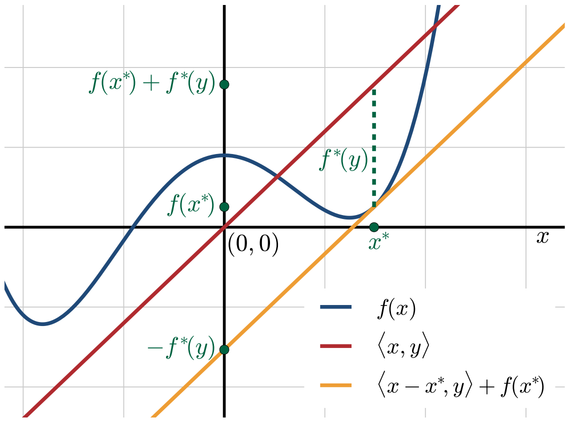


Figure B.3
Illustration of the conjugate f^* of a function f. Given y, x^* is the point at which the distance between the hyperplane of equation $z = \langle x, y\rangle$ with normal y (slope y in dimension one) and the plot of $f(x)$ is the largest. This largest distance is equal to $f^*(y)$. The parallel hyperplane $z = \langle x - x^*, y\rangle + f(x^*)$ with normal y and passing through the point $(x^*, f(x^*)$ is shown. This is a supporting hyperplane of the plot of $f(x)$. The point at which it intercepts the y-axis (crossing point) has y-coordinate $-f^*(y)$.

Lemma B.37 (Conjugate of extended relative entropy) *Let $p_0 \in \Delta$ be a distribution over $\mathcal{X}$ such that $p_0(x) > 0$ for all $x \in \mathcal{X}$. Define $f\colon \mathbb{R}^{\mathcal{X}} \to \mathbb{R}$ by*

$$f(p) = \begin{cases} \mathsf{D}(p\|p_0) & \text{if } p \in \Delta \\ +\infty & \text{otherwise.} \end{cases}$$

Then, the conjugate function $f^*\colon \mathbb{R}^{\mathcal{X}} \to \mathbb{R}$ of f is defined by

$$\forall q \in \mathbb{R}^{\mathcal{X}}, \;\; f^*(q) = \log\Bigg(\sum_{x\in\mathcal{X}} p_0(x) e^{q(x)}\Bigg).$$

Proof: By definition of f, for any $q \in \mathbb{R}^{\mathcal{X}}$, we can write

$$\sup_{p\in\mathbb{R}^{\mathcal{X}}} \big(\langle p, q\rangle - \mathsf{D}(p\|p_0)\big) = \sup_{p\in\Delta} \big(\langle p, q\rangle - \mathsf{D}(p\|p_0)\big). \tag{B.21}$$

Fix $q \in \mathbb{R}^{\mathcal{X}}$ and let $\bar{q} \in \Delta$ be defined for all $x \in \mathcal{X}$ by

$$\bar{q}(x) = \frac{p_0(x)e^{q(x)}}{\sum_{x\in\mathcal{X}} p_0(x)e^{q(x)}} = \frac{p_0(x)e^{q(x)}}{\mathbb{E}_{p_0}[e^q]}. \tag{B.22}$$

Then, the following holds for all $p \in \Delta$:

$$\langle p, q\rangle - \mathsf{D}(p\|p_0) = \mathop{\mathbb{E}}_{p}[\log(e^q)] - \mathop{\mathbb{E}}_{p}\left[\log\frac{p}{p_0}\right] = \mathop{\mathbb{E}}_{p}\left[\log\frac{p_0 e^q}{p}\right] = -\mathsf{D}(p\|\bar{q}) + \log\mathop{\mathbb{E}}_{p_0}[e^q].$$

Since $\mathsf{D}(p\|\bar{q}) \geq 0$ and $\mathsf{D}(p\|\bar{q}) = 0$ for $p = \bar{q}$, this shows that $\sup_{p\in\Delta}\big(p\cdot q - \mathsf{D}(p\|p_0)\big) = \log\big(\mathbb{E}_{p_0}[e^q]\big)$ and concludes the proof. □

Table B.1 gives a series of other examples of functions and their conjugates. The following is an immediate consequence of the definition of the conjugate functions.

Table B.1
Examples of functions g and their conjugates g^*.

$g(x)$	$\mathrm{dom}(g)$	$g^*(y)$	$\mathrm{dom}(g^*)$
$f(ax)\ (a \neq 0)$	$\mathcal{X}$	$f^*\left(\frac{y}{a}\right)$	$\mathcal{X}^*$
$f(x+b)$	$\mathcal{X}$	$f^*(y) - \langle b, y \rangle$	$\mathcal{X}^*$
$af(x)\ (a > 0)$	$\mathcal{X}$	$af^*\left(\frac{y}{a}\right)$	$\mathcal{X}^*$
$\alpha x + \beta$	$\mathbb{R}$	$\begin{cases} -\beta & \text{if } y = \alpha \\ +\infty & \text{otherwise} \end{cases}$	$\mathbb{R}$
$\frac{\lvert x \rvert^p}{p}\ (p > 1)$	$\mathbb{R}$	$\frac{\lvert y \rvert^q}{q}\ (\frac{1}{p} + \frac{1}{q} = 1)$	$\mathbb{R}$
$\frac{-x^p}{p}\ (0 < p < 1)$	$\mathbb{R}_+$	$\frac{-(-y)^q}{q}\ (\frac{1}{p} + \frac{1}{q} = 1)$	$\mathbb{R}_-$
$\sqrt{1+x^2}$	$\mathbb{R}$	$-\sqrt{1-(y)^2}$	$[-1, 1]$
$-\log(x)$	$\mathbb{R}_+ \setminus \{0\}$	$-(1 + \log(-y))$	$\mathbb{R}_- \setminus \{0\}$
e^x	$\mathbb{R}$	$\begin{cases} y\log(y) - y\,, & \text{if } y > 0 \\ 0\,, & \text{if } y = 0 \end{cases}$	$\mathbb{R}_+$
$\log(1 + e^x)$	$\mathbb{R}$	$\begin{cases} y\log(y) + (1-y)\log(1-y)\,, & \text{if } 0 < y < 1 \\ 0\,, & \text{if } y = 0, 1 \end{cases}$	$[0, 1]$
$-\log(1 - e^x)$	$\mathbb{R}$	$\begin{cases} y\log(y) - (1+y)\log(1+y)\,, & \text{if } y > 0 \\ 0\,, & \text{if } y = 0 \end{cases}$	$\mathbb{R}_+$

Proposition B.38 (Fenchel's inequality) *Let $\mathcal{X}$ be a Hilbert space. For any function $f\colon \mathcal{X} \mapsto [-\infty, +\infty]$ and any $x \in \mathrm{dom}(f)$ and $u \in \mathcal{X}$, the following inequality holds:*

$$f(x) + f^*(u) \geq \langle u, x \rangle\,. \tag{B.23}$$

Equality holds iff u is a subgradient of f at x.

We will denote by A^* the adjoint operator of a bounded (or continuous) linear map $A\colon \mathcal{X} \to \mathcal{Y}$. Also, we denote by $\mathrm{cont}(f)$ the set of points $x \in \mathcal{X}$ at which $f\colon \mathcal{X} \to [-\infty, +\infty]$ is finite and continuous.

Theorem B.39 (Fenchel duality theorem) *Let $\mathcal{X}$ and $\mathcal{Y}$ be two Hilbert spaces, $f\colon \mathcal{X} \to (-\infty, +\infty]$ and $g\colon \mathcal{Y} \to (-\infty, +\infty]$ two convex functions and $A\colon \mathcal{X} \to \mathcal{Y}$ a bounded linear map. Then, the following two optimization problems (*Fenchel problems*)*

$$p^* = \inf_{x \in \mathcal{X}} \{f(x) + g(Ax)\}$$
$$d^* = \sup_{y \in \mathcal{Y}} \{-f^*(A^*y) - g^*(-y)\}$$

satisfy the weak duality *$p^* \geq d^*$. If further f and g satisfy the condition*

$$0 \in \mathrm{core}\big(\mathrm{dom}(g) - A(\mathrm{dom}(f))\big),$$

or the stronger condition

$$A(\mathrm{dom}(f)) \cap \mathrm{cont}(g) \neq \emptyset,$$

then strong duality *holds, that is $p^* = d^*$ and the supremum in the dual problem is attained if $d^* \in \mathbb{R}$.*

Proof: By Fenchel's inequality (proposition B.38) applied to both f and g, for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the following inequalities hold:

$$f(x) + f^*(A^*y) \geq \langle A^*y, x\rangle = \langle y, Ax\rangle = -\langle -y, Ax\rangle \geq -g(Ax) - g^*(-y).$$

Comparing the leftmost and the rightmost terms gives

$$f(x) + f^*(A^*y) \geq -g(Ax) - g^*(-y) \iff f(x) + g(Ax) \geq -f^*(A^*y) - g^*(-y).$$

Taking the infimum over $x \in \mathcal{X}$ of the left-hand side and the supremum over $y \in \mathcal{Y}$ of the right-hand side of the last inequality yields $p^* \geq d^*$.

Consider now the function $h\colon \mathcal{Y} \to [-\infty, +\infty]$ defined for all $u \in \mathcal{Y}$ by

$$h(u) = \inf_{x\in\mathcal{X}} \{f(x) + g(Ax + u)\}. \tag{B.24}$$

Since $(x, u) \mapsto f(x) + g(Ax + u)$ is convex, h is convex as the infimum over one argument of that function. u is in $\operatorname{dom}(h)$ iff there exists $x \in \mathcal{X}$ such that $f(x) + g(Ax + u) < +\infty$, that is iff there exists $x \in \mathcal{X}$ such that $f(x) < +\infty$ and $g(Ax + u) < +\infty$, that is iff there exists $x \in \operatorname{dom}(f)$ such that $(Ax + u) \in \operatorname{dom}(g)$. Thus, we have $\operatorname{dom}(h) = \operatorname{dom}(g) - A \operatorname{dom}(f)$.

If $p^* = -\infty$, then strong duality clearly holds. Otherwise, $p^* > -\infty$. If $0 \in \operatorname{core}\big(\operatorname{dom}(g) - A(\operatorname{dom}(f))\big) = \operatorname{core}(\operatorname{dom}(h))$, then 0 is in $\operatorname{dom}(h)$ and $p^* < +\infty$. Thus, $p^* = h(0)$ is in $\mathbb{R}$. By proposition B.34, since $h(0) > -\infty$ and $0 \in \operatorname{core}(\operatorname{dom}(h))$, h takes values in $(-\infty, +\infty]$. Thus, by proposition B.35, h admits a subgradient $-y$ at 0. By definition of y, for all $x \in \mathcal{X}$ and $u \in \mathcal{Y}$,

$$\begin{aligned}
h(0) &\leq h(u) + \langle y, u\rangle \\
&\leq f(x) + g(Ax + u) + \langle y, u\rangle \\
&= \{f(x) - \langle A^*y, x\rangle\} + \{g(Ax + u) + \langle y, u\rangle + \langle A^*y, x\rangle\} \\
&= \{f(x) - \langle A^*y, x\rangle\} + \{g(Ax + u) + \langle y, Ax + u\rangle\}.
\end{aligned}$$

Taking the infimum over u and the supremum over x yields

$$h(0) \leq -f^*(A^*y) - g^*(-y) \leq d^* \leq p^* = h(0),$$

which proves $d^* = p^*$ and that the supremum defining d^* is reached at y.

Finally, assume that $A(\operatorname{dom}(f)) \cap \operatorname{cont}(g) \neq \emptyset$ and let $u \in A(\operatorname{dom}(f)) \cap \operatorname{cont}(g)$. Then, $u = Ax$ with $x \in \operatorname{dom}(f)$ and $u \in \operatorname{cont}(g) \subseteq \operatorname{dom}(g)$. Thus, we have $0 = u - Ax \in \operatorname{dom}(g) - A \operatorname{dom}(f)$. Since g is continuous at u and $g(u)$ is finite, for any $v \in \mathcal{X}$, there exists $\epsilon > 0$ such that $g(u + tv)$ is finite for all $t \in [0, \epsilon]$, thus $w_t = (u + tv) \in \operatorname{dom}(g)$. Therefore, for any $t \in [0, \epsilon]$, $tv = w_t - u = w_t - Ax \in \operatorname{dom}(g) - A \operatorname{dom}(f)$, which shows that $0 \in \operatorname{core}\big(\operatorname{dom}(g) - A(\operatorname{dom}(f))\big)$. □

To illustrate the theorem, consider the case where A is the identity operator. The primal optimization problem is then $\min_x \big\{f(x) + g(x)\big\}$. Figure B.4 illustrates the Fenchel duality theorem in that case. The primal problem consists of finding the point x^* at which the distance between the plots of $f(x)$ and $-g(x)$ is minimal since $f(x) + g(x) = f(x) - (-g(x))$. As shown by the figure, under the conditions of the theorem, this coincides with seeking y^* at which the difference of the conjugate values of $f(x)$ and $-g(x)$, that is the difference $-f^*(y) - g^*(-y)$ is maximal.

B.5 Chapter notes

The results presented in this appendix are based on several key theorems: theorem B.3 due to Fermat (1629); theorem B.27 due to Lagrange (1797), theorem B.30 due to Karush [1939] and Kuhn and Tucker [1951], and theorem B.39 due to Werner Fenchel, based on the notion of conjugate functions or Legendre transformations. For a more extensive material on convex optimization, we strongly recommend the books of Boyd and Vandenberghe [2004], Bertsekas, Nedić, and Ozdaglar [2003], Rockafellar [1997], Borwein and Lewis [2000] and Borwein and Zhu [2005] which have formed the basis for part of the material presented in this appendix. In particular, our table of conjugate functions is extracted from [Borwein and Lewis, 2000].

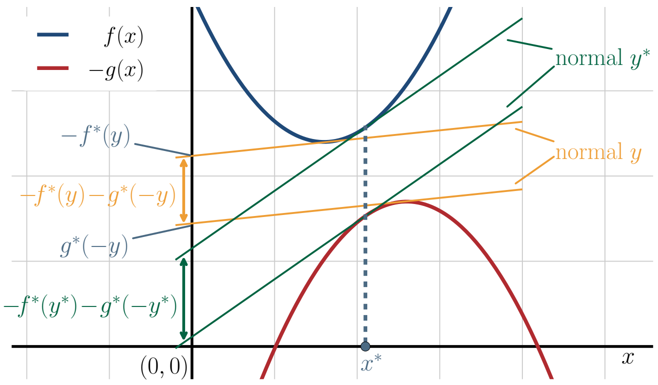


Figure B.4
Illustration of Fenchel Duality; $\min_x \{f(x) + g(x)\} = \max_y \{-f^*(y) - g^*(-y)\}$.

B.6 Exercises

B.1 Give the conjugate of the function f defined by $f(x) = |x|$ for all $x \in \mathbb{R}$.

B.2 Prove the correctness of the conjugate function g^* for each function g of Table B.1.

B.3 Give the proof of Proposition B.35.

C Probability Review

In this appendix, we give a brief review of some basic notions of probability and will also define the notation that is used throughout the textbook.

C.1 Probability

A *probability space* is a tuple consisting of three components: a *sample space*, an *events set*, and a *probability distribution*:

- *sample space* Ω: Ω is the set of all elementary events or outcomes possible in a trial, for example, each of the six outcomes in $\{1, \ldots, 6\}$ when casting a die.

- *events set* $\mathcal{F}$: $\mathcal{F}$ is a σ-*algebra*, that is a set of subsets of Ω containing Ω that is closed under complementation and countable union (therefore also countable intersection). An example of an event may be "the die lands on an odd number".

- *probability distribution*: $\mathbb{P}$ is a mapping from the set of all events $\mathcal{F}$ to $[0, 1]$ such that $\mathbb{P}[\Omega] = 1$, $\mathbb{P}[\emptyset] = 1$, and, for all mutually exclusive events $A_1, \ldots, A_n$,
$$\mathbb{P}[A_1 \cup \ldots \cup A_n] = \sum_{i=1}^{n} \mathbb{P}[A_i].$$
The discrete probability distribution associated with a fair die can be defined by $\mathbb{P}[A_i] = 1/6$ for $i \in \{1 \ldots 6\}$, where A_i is the event that the die lands on value i.

C.2 Random variables

Definition C.1 (Random variables) *A* random variable X *is a function* $X\colon \Omega \to \mathbb{R}$ *that is* measurable*, that is such that for any interval I, the subset of the sample space $\{\omega \in \Omega\colon X(\omega) \in I\}$ is an event.*

The *probability mass function* of a discrete random variable X is defined as the function $x \mapsto \mathbb{P}[X = x]$. The *joint probability mass function* of discrete random variables X and Y is defined as the function $(x, y) \mapsto \mathbb{P}[X = x \wedge Y = y]$.

A probability distribution is said to be *absolutely continuous* when it admits a *probability density function*, that is a function f associated to a real-valued random variable X that satisfies for all $a, b \in \mathbb{R}$
$$\mathbb{P}[a \leq X \leq b] = \int_a^b f(x)dx\,. \tag{C.1}$$

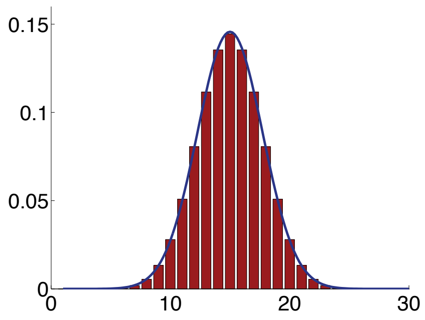


Figure C.1
Approximation of the binomial distribution (in red) by a normal distribution (in blue).

Definition C.2 (Binomial distribution) *A random variable X is said to follow a* binomial distribution $B(n, p)$ *with $n \in \mathbb{N}$ and $p \in [0, 1]$ if for any $k \in \{0, 1, \ldots, n\}$,*

$$\mathbb{P}[X = k] = \binom{n}{k} p^k (1 - p)^{n-k} .$$

Definition C.3 (Normal distribution) *A random variable X is said to follow a* normal *(or* Gaussian*) distribution $N(\mu, \sigma^2)$ with $\mu \in \mathbb{R}$ and $\sigma > 0$ if its probability density function is given by,*

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right).$$

The standard normal *distribution $N(0, 1)$ is the normal distribution with zero mean and unit variance.*

The normal distribution is often used to approximate a binomial distribution. Figure C.1 illustrates that approximation.

Definition C.4 (Laplace distribution) *A random variable X is said to follow a* Laplace distribution *with location parameter $\mu \in \mathbb{R}$ and scale parameter $b > 0$ if its probability density function is given by,*

$$f(x) = \frac{1}{2b} \exp\left(-\frac{|x - \mu|}{b}\right).$$

Definition C.5 (Gibbs distributions) *Given a set $\mathcal{X}$ and* feature function $\Phi \colon \mathcal{X} \to \mathbb{R}^N$, *a random variable X is said to follow a* Gibbs distribution *with parameter $\mathbf{w} \in \mathbb{R}^N$ if for any $x \in \mathcal{X}$,*

$$\mathbb{P}[X = x] = \frac{\exp(\mathbf{w} \cdot \Phi(x))}{\sum_{x \in \mathcal{X}} \exp(\mathbf{w} \cdot \Phi(x))} .$$

The normalizing quantity in the denominator $Z = \sum_{x \in \mathcal{X}} \exp(\mathbf{w} \cdot \Phi(x))$ is also called the partition function.

Definition C.6 (Poisson distribution) *A random variable X is said to follow a* Poisson distribution *with $\lambda > 0$ if for any $k \in \mathbb{N}$,*

$$\mathbb{P}[X = k] = \frac{\lambda^k e^{-\lambda}}{k!} .$$

The definition of the following family of distributions uses the notion of independence of random variables defined in the next section.

Definition C.7 (χ^2 distribution) *The* χ^2 distribution *(or* chi-squared distribution*) with k* degrees of freedom *is the distribution of the sum of the squares of k independent random variables, each following a standard normal distribution.*

C.3 Conditional probability and independence

Definition C.8 (Conditional probability) *The* conditional probability *of event A given event B is defined by*

$$\mathbb{P}[A \mid B] = \frac{\mathbb{P}[A \cap B]}{\mathbb{P}[B]}, \tag{C.2}$$

when $\mathbb{P}[B] \neq 0$.

Definition C.9 (Independence) *Two events A and B are said to be* independent *if*

$$\mathbb{P}[A \cap B] = \mathbb{P}[A]\,\mathbb{P}[B]. \tag{C.3}$$

Equivalently, A and B are independent iff $\mathbb{P}[A \mid B] = \mathbb{P}[A]$ when $\mathbb{P}[B] \neq 0$.

A sequence of random variables is said to be *independent and identically distributed (i.i.d.)* when the random variables are mutually independent and follow the same distribution.

The following are basic probability formulae related to the notion of conditional probability. They hold for any events A, B, and $A_1, \ldots, A_n$, with the additional constraint $\mathbb{P}[B] \neq 0$ needed for the Bayes formula to be well defined:

$$\mathbb{P}[A \cup B] = \mathbb{P}[A] + \mathbb{P}[B] - \mathbb{P}[A \cap B] \qquad (\textit{sum rule}) \tag{C.4}$$

$$\mathbb{P}[\bigcup_{i=1}^{n} A_i] \leq \sum_{i=1}^{n} \mathbb{P}[A_i] \qquad (\textit{union bound}) \tag{C.5}$$

$$\mathbb{P}[A \mid B] = \frac{\mathbb{P}[B \mid A]\,\mathbb{P}[A]}{\mathbb{P}[B]} \qquad (\textit{Bayes formula}) \tag{C.6}$$

$$\mathbb{P}[\bigcap_{i=1}^{n} A_i] = \mathbb{P}[A_1]\,\mathbb{P}[A_2 \mid A_1] \cdots \mathbb{P}[A_n \mid \bigcap_{i=1}^{n-1} A_i] \qquad (\textit{chain rule}). \tag{C.7}$$

The sum rule follows immediately from the decomposition of $A \cup B$ as the union of the disjoint sets A and $(B - A \cap B)$. The union bound is a direct consequence of the sum rule. The Bayes formula follows immediately from the definition of conditional probability and the observation that: $\mathbb{P}[A|B]\,\mathbb{P}[B] = \mathbb{P}[B|A]\,\mathbb{P}[A] = \mathbb{P}[A \cap B]$. Similarly, the chain rule follows the observation that $\mathbb{P}[A_1]\,\mathbb{P}[A_2|A_1] = \mathbb{P}[A_1 \cap A_2]$; using the same argument shows recursively that the product of the first k terms of the right-hand side equals $\mathbb{P}[\bigcap_{i=1}^{k} A_i]$.

Finally, assume that $\Omega = A_1 \cup A_2 \cup \ldots \cup A_n$ with $A_i \cap A_j = \emptyset$ for $i \neq j$, i.e., the A_is are mutually disjoint. Then, the following formula is valid for any event B:

$$\mathbb{P}[B] = \sum_{i=1}^{n} \mathbb{P}[B \mid A_i]\,\mathbb{P}[A_i] \qquad (\textit{theorem of total probability}). \tag{C.8}$$

This follows the observation that $\mathbb{P}[B \mid A_i]\,\mathbb{P}[A_i] = \mathbb{P}[B \cap A_i]$ by definition of the conditional probability and the fact that the events $B \cap A_i$ are mutually disjoint.

C.4 Expectation and Markov's inequality

Definition C.10 (Expectation) *The* expectation *or* mean *of a random variable X is denoted by $\mathbb{E}[X]$ and defined by*

$$\mathbb{E}[X] = \sum_{x} x\,\mathbb{P}[X = x]. \tag{C.9}$$

When X follows a probability distribution $\mathcal{D}$, we will also write $\mathbb{E}_{x \sim \mathcal{D}}[x]$ instead of $\mathbb{E}[X]$ to explicitly indicate the distribution. A fundamental property of expectation, which is straightforward to verify using its definition, is that it is linear, that is, for any two random variables X and Y and any $a, b \in \mathbb{R}$, the following holds:

$$\mathbb{E}[aX + bY] = a\,\mathbb{E}[X] + b\,\mathbb{E}[Y]. \tag{C.10}$$

Furthermore, when X and Y are independent random variables, then the following identity holds:

$$\mathbb{E}[XY] = \mathbb{E}[X]\,\mathbb{E}[Y]. \tag{C.11}$$

Indeed, by definition of expectation and of independence, we can write

$$\begin{aligned}\mathbb{E}[XY] = \sum_{x,y} xy\,\mathbb{P}[X = x \wedge Y = y] &= \sum_{x,y} xy\,\mathbb{P}[X = x]\,\mathbb{P}[Y = y]\\ &= \Big(\sum_x x\,\mathbb{P}[X = x]\Big)\Big(\sum_y y\,\mathbb{P}[Y = y]\Big),\end{aligned}$$

where in the last step we used Fubini's theorem . The following provides a simple bound for a non-negative random variable in terms of its expectation, known as *Markov's inequality*.

Theorem C.11 (Markov's inequality) *Let X be a non-negative random variable with $\mathbb{E}[X] < \infty$. Then for all $t > 0$,*

$$\mathbb{P}\big[X \geq t\,\mathbb{E}[X]\big] \leq \frac{1}{t}. \tag{C.12}$$

Proof: The proof steps are as follows:

$$\begin{aligned}\mathbb{P}[X \geq t\,\mathbb{E}[X]] &= \sum_{x \geq t\,\mathbb{E}[X]} \mathbb{P}[X = x] && \text{(by definition)}\\ &\leq \sum_{x \geq t\,\mathbb{E}[X]} \mathbb{P}[X = x]\frac{x}{t\,\mathbb{E}[X]} && \Big(\text{using } \frac{x}{t\,\mathbb{E}[X]} \geq 1\Big)\\ &\leq \sum_x \mathbb{P}[X = x]\frac{x}{t\,\mathbb{E}[X]} && \text{(extending non-negative sum)}\\ &= \mathbb{E}\left[\frac{X}{t\,\mathbb{E}[X]}\right] = \frac{1}{t} && \text{(linearity of expectation).}\end{aligned}$$

This concludes the proof. □

C.5 Variance and Chebyshev's inequality

Definition C.12 (Variance — Standard deviation) *The* variance *of a random variable X is denoted by* Var$[X]$ *and defined by*

$$\mathrm{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2]. \tag{C.13}$$

The standard deviation *of a random variable X is denoted by σ_X and defined by*

$$\sigma_X = \sqrt{\mathrm{Var}[X]}. \tag{C.14}$$

For any random variable X and any $a \in \mathbb{R}$, the following basic properties hold for the variance, which can be proven straightforwardly:

$$\mathrm{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2 \tag{C.15}$$

$$\mathrm{Var}[aX] = a^2\,\mathrm{Var}[X]. \tag{C.16}$$

Furthermore, when X and Y are independent, then

$$\mathrm{Var}[X + Y] = \mathrm{Var}[X] + \mathrm{Var}[Y]. \tag{C.17}$$

Indeed, using the linearity of expectation and the identity $\mathbb{E}[X]\,\mathbb{E}[Y] - \mathbb{E}[XY] = 0$ which holds by the independence of X and Y, we can write

$$\begin{aligned}\mathrm{Var}[X + Y] &= \mathbb{E}[(X + Y)^2] - \mathbb{E}[X + Y]^2\\ &= \mathbb{E}[X^2 + Y^2 + 2XY] - (\mathbb{E}[X]^2 + \mathbb{E}[Y]^2 + 2\,\mathbb{E}[XY])\\ &= (\mathbb{E}[X^2] - \mathbb{E}[X]^2) + (\mathbb{E}[Y^2] - \mathbb{E}[Y]^2) + 2(\mathbb{E}[X]\,\mathbb{E}[Y] - \mathbb{E}[XY])\\ &= \mathrm{Var}[X] + \mathrm{Var}[Y].\end{aligned}$$

The following inequality known as *Chebyshev's inequality* bounds the deviation of a random variable from its expectation in terms of its standard deviation.

Theorem C.13 (Chebyshev's inequality) *Let X be a random variable with $\mathrm{Var}[X] < +\infty$. Then, for all $t > 0$, the following inequality holds:*

$$\mathbb{P}\left[|X - \mathbb{E}[X]| \geq t\sigma_X\right] \leq \frac{1}{t^2}. \tag{C.18}$$

Proof: Observe that:

$$\mathbb{P}\left[|X - \mathbb{E}[X]| \geq t\sigma_X\right] = \mathbb{P}[(X - \mathbb{E}[X])^2 \geq t^2\sigma_X^2].$$

The result follows by application of Markov's inequality to $(X - \mathbb{E}[X])^2$. □

We will use Chebyshev's inequality to prove the following theorem.

Theorem C.14 (Weak law of large numbers) *Let $(X_n)_{n\in\mathbb{N}}$ be a sequence of independent random variables with the same mean μ and variance $\sigma^2 < \infty$. Let $\overline{X}_n = \frac{1}{n}\sum_{i=1}^n X_i$, then, for any $\epsilon > 0$,*

$$\lim_{n\to\infty} \mathbb{P}[|\overline{X}_n - \mu| \geq \epsilon] = 0. \tag{C.19}$$

Proof: Since the variables are independent, we can write

$$\mathrm{Var}[\overline{X}_n] = \sum_{i=1}^n \mathrm{Var}\left[\frac{X_i}{n}\right] = \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n}.$$

Thus, by Chebyshev's inequality (with $t = \epsilon/(\mathrm{Var}[\overline{X}_n])^{1/2}$), the following holds:

$$\mathbb{P}[|\overline{X}_n - \mu| \geq \epsilon] \leq \frac{\sigma^2}{n\epsilon^2},$$

which implies (C.19). □

Example C.15 (Applying Chebyshev's inequality) Suppose we roll a pair of fair dice n times. Can we give a good estimate of the total value of the n rolls? If we compute the mean and variance, we find $\mu = 7n$ and $\sigma^2 = 35/6n$ (we leave it to the reader to verify these expressions). Thus, applying Chebyshev's inequality, we see that the final sum will lie within $7n \pm 10\sqrt{\frac{35}{6}n}$ in at least 99 percent of all experiments. Therefore, the odds are better than 99 to 1 that the sum will be between 6.975M and 7.025M after 1M rolls.

Definition C.16 (Covariance) *The* covariance *of two random variables X and Y is denoted by $\mathrm{Cov}(X, Y)$ and defined by*

$$\mathrm{Cov}(X, Y) = \mathbb{E}\left[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\right]. \tag{C.20}$$

Two random variables X and Y are said to be *uncorrelated* when $\mathrm{Cov}(X, Y) = 0$. It is straightforward to see that if two random variables X and Y are independent then they are uncorrelated, but the converse does not hold in general. The covariance defines a positive semidefinite and symmetric bilinear form:

- symmetry: $\mathrm{Cov}(X, Y) = \mathrm{Cov}(Y, X)$ for any two random variables X and Y;
- bilinearity: $\mathrm{Cov}(X + X', Y) = \mathrm{Cov}(X, Y) + \mathrm{Cov}(X', Y)$ and $\mathrm{Cov}(aX, Y) = a\,\mathrm{Cov}(X, Y)$ for any random variables X, X', and Y and $a \in \mathbb{R}$;
- positive semidefiniteness: $\mathrm{Cov}(X, X) = \mathrm{Var}[X] \geq 0$ for any random variable X.

The following Cauchy-Schwarz inequality holds for random variables X and Y with $\mathrm{Var}[X] < +\infty$ and $\mathrm{Var}[Y] < +\infty$:

$$|\mathrm{Cov}(X, Y)| \leq \sqrt{\mathrm{Var}[X]\,\mathrm{Var}[Y]}. \tag{C.21}$$

The following definition extends the notion of covariance to a vector of random variables.

Definition C.17 *The* covariance matrix *of a vector of random variables* $\mathbf{X} = (X_1, \ldots, X_N)$ *is the matrix in* $\mathbb{R}^{N \times N}$ *denoted by* $\mathbf{C}(\mathbf{X})$ *and defined by*

$$\mathbf{C}(\mathbf{X}) = \mathbb{E}\left[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^\top\right]. \tag{C.22}$$

Thus, $\mathbf{C}(\mathbf{X}) = (\mathrm{Cov}(X_i, X_j))_{ij}$. It is straightforward to show that

$$\mathbf{C}(\mathbf{X}) = \mathbb{E}[\mathbf{X}\mathbf{X}^\top] - \mathbb{E}[\mathbf{X}]\,\mathbb{E}[\mathbf{X}]^\top. \tag{C.23}$$

We close this appendix with the following well-known theorem of probability.

Theorem C.18 (Central limit theorem) *Let* $X_1, \ldots, X_n$ *be a sequence of i.i.d. random variables with mean* μ *and standard deviation* σ. *Let* $\overline{X}_n = \frac{1}{n}\sum_{i=1}^n X_i$ *and* $\overline{\sigma}_n^2 = \sigma^2/n$. *Then,* $(\overline{X}_n - \mu)/\overline{\sigma}_n$ converges to the $N(0,1)$ in distribution, *that is for any* $t \in \mathbb{R}$,

$$\lim_{n\to\infty} \mathbb{P}[(\overline{X}_n - \mu)/\overline{\sigma}_n \le t] = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}\, dx\,.$$

C.6 Moment-generating functions

The expectation $\mathbb{E}[X^p]$ is called the *pth-moment* of the random variable X. The moment-generating function of a random variable X is a key function from which its different moments can be straightforwardly computed via differentiation at zero. It can therefore be crucial for specifying the distribution of X or analyzing its properties.

Definition C.19 (Moment-generating function) *The* moment-generating function *of a random variable* X *is the function* $M_X \colon t \mapsto \mathbb{E}[e^{tX}]$ *defined over the set of* $t \in \mathbb{R}$ *for which the expectation is finite.*

If M_X is differentiable at zero, then the pth-moment of X is given by $\mathbb{E}[X^p] = M_X^{(p)}(0)$. We will present in the next chapter a general bound on the moment-generating function of a zero-mean bounded random variable (Lemma D.1). Here, we illustrate its computation in two special cases.

Example C.20 (Standard normal distribution) Let X be a random variable following a normal distribution with mean 0 and variance 1. Then, M_X is defined for all $t \in \mathbb{R}$ by

$$M_X(t) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} e^{tx} dx = e^{\frac{t^2}{2}} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-t)^2} dx = e^{\frac{t^2}{2}}\,, \tag{C.24}$$

by recognizing that the last integrand is the probability density function of a normal distribution with mean t and variance 1.

Example C.21 (χ^2 distribution) Let X be a random variable following a χ^2 distribution with k degrees of freedom. We can write $X = \sum_{i=1}^k X_i^2$ where the X_is are independent and follow a standard normal distribution.

Let $t < 1/2$. By the i.i.d. assumption about the variables X_i, we can write

$$\mathbb{E}[e^{tX}] = \mathbb{E}\Big[\prod_{i=1}^k e^{tX_i^2}\Big] = \prod_{i=1}^k \mathbb{E}\left[e^{tX_i^2}\right] = \mathbb{E}\left[e^{tX_1^2}\right]^k.$$

By definition of the standard normal distribution, we have

$$\begin{aligned}\mathbb{E}[e^{tX_1^2}] &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{tx^2} e^{\frac{-x^2}{2}} dx = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{(1-2t)\frac{-x^2}{2}} dx \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \frac{e^{\frac{-u^2}{2}}}{\sqrt{1-2t}} du = (1-2t)^{-\frac{1}{2}},\end{aligned}$$

where we used the change of variable $u = \sqrt{1-2t}\,x$. In view of that, the moment-generating function of the χ^2 distribution is given by

$$\forall t < 1/2,\ M_X(t) = \mathbb{E}[e^{tX}] = (1-2t)^{-\frac{k}{2}}. \tag{C.25}$$

C.7 Exercises

C.1 Let $f\colon (0, +\infty) \to \mathbb{R}_+$ be a function admitting an inverse f^{-1} and let X be a random variable. Show that if for any $t > 0$, $\mathbb{P}[X > t] \leq f(t)$, then, for any $\delta > 0$, with probability at least $1 - \delta$, $X \leq f^{-1}(\delta)$.

C.2 Let X be a discrete random variable taking non-negative integer values. Show that $\mathbb{E}[X] = \sum_{n \geq 1} \mathbb{P}[X \geq n]$ (*Hint*: rewrite $\mathbb{P}[X = n]$ as $\mathbb{P}[X \geq n] - \mathbb{P}[X \geq n + 1]$).

D Concentration Inequalities

In this appendix, we present several *concentration inequalities* used in the proofs given in this book. Concentration inequalities give probability bounds for a random variable to be concentrated around its mean, or for it to deviate from its mean or some other value.

D.1 Hoeffding's inequality

We first present Hoeffding's inequality, whose proof makes use of the general *Chernoff bounding technique*. Given a random variable X and $\epsilon > 0$, this technique consists of proceeding as follows to bound $\mathbb{P}[X \geq \epsilon]$. For any $t > 0$, first Markov's inequality is used to bound $\mathbb{P}[X \geq \epsilon]$:

$$\mathbb{P}[X \geq \epsilon] = \mathbb{P}[e^{tX} \geq e^{t\epsilon}] \leq e^{-t\epsilon}\, \mathbb{E}[e^{tX}]\,. \tag{D.1}$$

Then, an upper bound $g(t)$ is found for $\mathbb{E}[e^{tX}]$ and t is selected to minimize $e^{-t\epsilon}g(t)$. For Hoeffding's inequality, the following lemma provides an upper bound on $\mathbb{E}[e^{tX}]$.

Lemma D.1 (Hoeffding's lemma) *Let X be a random variable with $E[X] = 0$ and $a \leq X \leq b$ with $b > a$. Then, for any $t > 0$, the following inequality holds:*

$$\mathbb{E}[e^{tX}] \leq e^{\frac{t^2(b-a)^2}{8}}\,. \tag{D.2}$$

Proof: By the convexity of $x \mapsto e^x$, for all $x \in [a, b]$, the following holds:

$$e^{tx} \leq \frac{b-x}{b-a}e^{ta} + \frac{x-a}{b-a}e^{tb}\,.$$

Thus, using $\mathbb{E}[X] = 0$,

$$\mathbb{E}[e^{tX}] \leq \mathbb{E}\left[\frac{b-X}{b-a}e^{ta} + \frac{X-a}{b-a}e^{tb}\right] = \frac{b}{b-a}e^{ta} + \frac{-a}{b-a}e^{tb} = e^{\phi(t)}\,,$$

where,

$$\phi(t) = \log\left(\frac{b}{b-a}e^{ta} + \frac{-a}{b-a}e^{tb}\right) = ta + \log\left(\frac{b}{b-a} + \frac{-a}{b-a}e^{t(b-a)}\right)\,.$$

For any $t > 0$, the first and second derivative of ϕ are given below:

$$\begin{aligned}
\phi'(t) &= a - \frac{ae^{t(b-a)}}{\frac{b}{b-a} - \frac{a}{b-a}e^{t(b-a)}} = a - \frac{a}{\frac{b}{b-a}e^{-t(b-a)} - \frac{a}{b-a}}\,,\\
\phi''(t) &= \frac{-abe^{-t(b-a)}}{[\frac{b}{b-a}e^{-t(b-a)} - \frac{a}{b-a}]^2}\\
&= \frac{\alpha(1-\alpha)e^{-t(b-a)}(b-a)^2}{[(1-\alpha)e^{-t(b-a)} + \alpha]^2}\\
&= \frac{\alpha}{[(1-\alpha)e^{-t(b-a)} + \alpha]}\,\frac{(1-\alpha)e^{-t(b-a)}}{[(1-\alpha)e^{-t(b-a)} + \alpha]}(b-a)^2\,.
\end{aligned}$$

where α denotes $\frac{-a}{b-a}$. Note that $\phi(0) = \phi'(0) = 0$ and that $\phi''(t) = u(1-u)(b-a)^2$ where $u = \frac{\alpha}{[(1-\alpha)e^{-t(b-a)}+\alpha]}$. Since u is in $[0,1]$, $u(1-u)$ is upper bounded by $1/4$ and $\phi''(t) \leq \frac{(b-a)^2}{4}$. Thus, by the second order expansion of function ϕ, there exists $\theta \in [0,t]$ such that:

$$\phi(t) = \phi(0) + t\phi'(0) + \frac{t^2}{2}\phi''(\theta) \leq t^2 \frac{(b-a)^2}{8}, \tag{D.3}$$

which completes the proof. □

The lemma can be used to prove the following result known as *Hoeffding's inequality*.

Theorem D.2 (Hoeffding's inequality) *Let $X_1, \ldots, X_m$ be independent random variables with X_i taking values in $[a_i, b_i]$ for all $i \in [m]$. Then, for any $\epsilon > 0$, the following inequalities hold for $S_m = \sum_{i=1}^m X_i$:*

$$\mathbb{P}[S_m - \mathbb{E}[S_m] \geq \epsilon] \leq e^{-2\epsilon^2/\sum_{i=1}^m (b_i - a_i)^2} \tag{D.4}$$

$$\mathbb{P}[S_m - \mathbb{E}[S_m] \leq -\epsilon] \leq e^{-2\epsilon^2/\sum_{i=1}^m (b_i - a_i)^2}. \tag{D.5}$$

Proof: Using the Chernoff bounding technique and lemma D.1, we can write:

$$\begin{aligned}
\mathbb{P}[S_m - \mathbb{E}[S_m] \geq \epsilon] &\leq e^{-t\epsilon}\, \mathbb{E}[e^{t(S_m - \mathbb{E}[S_m])}] \\
&= e^{-t\epsilon} \Pi_{i=1}^m \mathbb{E}[e^{t(X_i - \mathbb{E}[X_i])}] && \text{(independence of } X_i\text{s)} \\
&\leq e^{-t\epsilon} \Pi_{i=1}^m e^{t^2(b_i - a_i)^2/8} && \text{(lemma D.1)} \\
&= e^{-t\epsilon} e^{t^2 \sum_{i=1}^m (b_i - a_i)^2/8} \\
&\leq e^{-2\epsilon^2/\sum_{i=1}^m (b_i - a_i)^2},
\end{aligned}$$

where we chose $t = 4\epsilon/\sum_{i=1}^m (b_i - a_i)^2$ to minimize the upper bound. This proves the first statement of the theorem, and the second statement is shown in a similar way. □

When the variance $\sigma_{X_i}^2$ of each random variable X_i is known and the $\sigma_{X_i}^2$s are relatively small, better concentration bounds can be derived (see *Bennett's* and *Bernstein's inequalities* proven in exercise D.6).

D.2 Sanov's theorem

Here, we present a finer upper bound than Hoeffding's inequality expressed in terms of the binary relative entropy.

Theorem D.3 (Sanov's theorem) *Let $X_1, \ldots, X_m$ be independent random variables drawn according to some distribution $\mathcal{D}$ with mean p and support included in $[0,1]$. Then, for any $q \in [0,1]$, the following inequality holds for $\widehat{p} = \frac{1}{m}\sum_{i=1}^m X_i$:*

$$\mathbb{P}[\widehat{p} \geq q] \leq e^{-m\mathsf{D}(q\|p)},$$

where $\mathsf{D}(q\|p) = q\log\frac{q}{p} + (1-q)\log\frac{1-q}{1-p}$ is the binary relative entropy of p and q.

Proof: For any $t > 0$, by convexity of the function $x \mapsto e^{tx}$, the following inequality holds for all $x \in [0,1]$: $e^{tx} = e^{t[(1-x)\cdot 0 + x \cdot 1]} \leq 1 - x + e^t x$. In view of that, for any $t > 0$, we can write

$$\begin{aligned}
\mathbb{P}[\widehat{p} \geq q] &= \mathbb{P}[e^{tm\widehat{p}} \geq e^{tmq}] \\
&= \mathbb{P}[e^{tm\widehat{p}} \geq e^{tmq}] \\
&\leq e^{-tmq}\, \mathbb{E}[e^{tm\widehat{p}}] && \text{(by Markov's inequality)} \\
&= e^{-tmq}\, \mathbb{E}[e^{t\sum_{i=1}^m X_i}] \\
&= e^{-tmq} \prod_{i=1}^m \mathbb{E}[e^{tX_i}] \\
&\leq e^{-tmq} \prod_{i=1}^m \mathbb{E}[1 - X_i + e^t X_i] && (\forall x \in [0,1], e^{tx} \leq 1 - x + e^t x) \\
&= [e^{-tq}(1 - p + e^t p)]^m.
\end{aligned}$$

Now, the function $f\colon t \mapsto e^{-tq}(1 - p + e^t p) = (1-p)e^{-tq} + pe^{t(1-q)}$ reaches its minimum at $t = \log \frac{q(1-p)}{p(1-q)}$. Plugging in this value of t in the inequality above yields $\mathbb{P}[\widehat{p} \geq q] \leq e^{-m\mathsf{D}(q\|p)}$. □

Note that for any $\epsilon > 0$, $\epsilon \leq 1 - p$, with the choice $q = p + \epsilon$, the theorem implies

$$\mathbb{P}[\widehat{p} \geq p + \epsilon] \leq e^{-m\mathsf{D}(p+\epsilon\|p)}. \tag{D.6}$$

This is a finer bound than Hoeffding's inequality (Theorem D.2) since, by Pinsker's inequality (Proposition E.7), $\mathsf{D}(p + \epsilon\|p) \geq \frac{1}{2}(2\epsilon)^2 = 2\epsilon^2$. Similarly, we can derive a symmetric bound by applying the theorem to the random variables $Y_i = 1 - X_i$. Then, for any $\epsilon > 0$, $\epsilon \leq p$, with the choice $q = p - \epsilon$, the theorem implies

$$\mathbb{P}[\widehat{p} \leq p - \epsilon] \leq e^{-m\mathsf{D}(p-\epsilon\|p)}. \tag{D.7}$$

D.3 Multiplicative Chernoff bounds

Sanov's theorem can be used to prove the following *multiplicative Chernoff bounds*.

Theorem D.4 (Multiplicative Chernoff bounds) *Let $X_1, \ldots, X_m$ be independent random variables drawn according to some distribution $\mathcal{D}$ with mean p and support included in $[0,1]$. Then, for any $\gamma \in \left[0, \frac{1}{p} - 1\right]$, the following inequality holds for $\widehat{p} = \frac{1}{m}\sum_{i=1}^m X_i$:*

$$\begin{aligned}
\mathbb{P}[\widehat{p} \geq (1+\gamma)p] &\leq e^{-\frac{mp\gamma^2}{3}} \\
\mathbb{P}[\widehat{p} \leq (1-\gamma)p] &\leq e^{-\frac{mp\gamma^2}{2}}.
\end{aligned}$$

Proof: The proof consists of deriving in each case a finer lower bound for the binary relative entropy than Pinsker's inequality. Using the inequalities $\log(1 + x) \geq \frac{x}{1+\frac{x}{2}}$ and $\log(1+x) < x$, we can write

$$\begin{aligned}
-\mathsf{D}((1+\gamma)p\|p) &= (1+\gamma)p \log \frac{p}{(1+\gamma)p} + (1 - (1+\gamma)p) \log \left[\frac{1-p}{1-(1+\gamma)p}\right] \\
&= (1+\gamma)p \log \frac{1}{1+\gamma} + (1 - p - \gamma p) \log \left[1 + \frac{\gamma p}{1 - p - \gamma p}\right] \\
&\leq (1+\gamma)p \frac{-\gamma}{1+\frac{\gamma}{2}} + (1 - p - \gamma p) \frac{\gamma p}{1 - p - \gamma p} \\
&= \gamma p \left[1 - \frac{1+\gamma}{1+\frac{\gamma}{2}}\right] = \frac{-\frac{\gamma^2 p}{2}}{1+\frac{\gamma}{2}} = \frac{-\gamma^2 p}{2+\gamma} \\
&\leq \frac{-\gamma^2 p}{2+1} = \frac{-\gamma^2 p}{3}.
\end{aligned}$$

Similarly, using the inequalities $(1-x)\log(1-x) \geq -x+\frac{x^2}{2}$ valid for $x \in (0,1)$ and $\log(1-x) < -x$, we can write

$$\begin{aligned}
-\mathsf{D}((1-\gamma)p\|p) &= (1-\gamma)p\log\frac{p}{(1-\gamma)p} + (1-(1-\gamma)p)\log\left[\frac{1-p}{1-(1-\gamma)p}\right]\\
&= (1-\gamma)p\log\frac{1}{1-\gamma} + (1-p+\gamma p)\log\left[1-\frac{\gamma p}{1-p+\gamma p}\right]\\
&\leq \Big(\gamma-\frac{\gamma^2}{2}\Big)p + (1-p+\gamma p)\frac{-\gamma p}{1-p+\gamma p} = \frac{-\gamma^2 p}{2}.
\end{aligned}$$

This completes the proof. □

D.4 Binomial distribution tails: Upper bounds

Let $X_1, \ldots, X_m$ be independent random variables taking values in $\{0,1\}$ with $\mathbb{P}[X_i = 1] = p \in [0,1]$ for $i = 1, \ldots, m$. Then, $\sum_{i=1}^m X_i$ follows the binomial distribution $B(m,p)$. We will denote by $\overline{X}$ the average $\overline{X} = \frac{1}{m}\sum_{i=1}^m X_i$. Then, the following equality and inequalities hold:

$$\mathbb{P}\Big[\overline{X} - p > \epsilon\Big] = \sum_{k=\lceil (p+\epsilon)m\rceil}^{m} \binom{m}{k} p^k (1-p)^{m-k} \qquad \text{(Binomial formula)}$$

$$\mathbb{P}\Big[\overline{X} - p > \epsilon\Big] \leq e^{-2m\epsilon^2} \qquad \text{(Hoeffding's inequality)}$$

$$\mathbb{P}\Big[\overline{X} - p > \epsilon\Big] \leq e^{-\frac{m\epsilon^2}{2\sigma^2+\frac{2\epsilon}{3}}} \qquad \text{(Bernstein's inequality)}$$

$$\mathbb{P}\Big[\overline{X} - p > \epsilon\Big] \leq e^{-m\sigma^2\theta\left(\frac{\epsilon}{\sigma^2}\right)} \qquad \text{(Bennett's inequality)}$$

$$\mathbb{P}\Big[\overline{X} - p > \epsilon\Big] \leq e^{-m\mathsf{D}(p+\epsilon\|p)} \qquad \text{(Sanov's inequality)},$$

where $\sigma^2 = p(1-p) = \mathrm{Var}[X_i]$ and $\theta(x) = (1+x)\log(1+x) - x$. The last three inequalities are shown in exercises D.6 and D.7. Using Bernstein's inequality, for example, we can see that for ϵ relatively small, that is $\epsilon \ll 2\sigma^2$, the upper bound is approximately of the form $e^{-\frac{m\epsilon^2}{2\sigma^2}}$ and thus admits a Gaussian behavior. For $\epsilon \gg 2\sigma^2$, $e^{-\frac{3m\epsilon}{2}}$, the upper bound admits a Poisson behavior.

Figure D.1 shows a comparison of these bounds for different values of the variance $\sigma^2 = p(1-p)$: small variance ($p = .05$), large variance ($p = .5$).

D.5 Binomial distribution tails: Lower bound

Let X be a random variable following the binomial distribution $B(m,p)$ and let k be an integer such that $p \leq \frac{1}{4}$ and $k \geq mp$ or $p \leq \frac{1}{2}$ and $mp \leq k \leq m(1-p)$. Then, the following inequality known as *Slud's inequality* holds:

$$\mathbb{P}[X \geq k] \geq \mathbb{P}\left[N \geq \frac{k-mp}{\sqrt{mp(1-p)}}\right], \tag{D.8}$$

where N is in standard normal form.

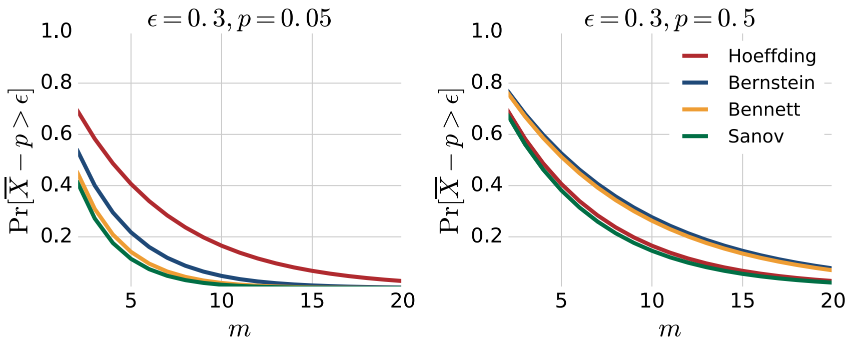


Figure D.1
Comparison of tail bounds for a binomial random variable for $\epsilon = .3$ and $p = .05$ (small variance) or $p = .5$ (maximal variance) as a function of the sample size m.

D.6 Azuma's inequality

This section presents a concentration inequality that is more general than Hoeffding's inequality. Its proof makes use of a Hoeffding's inequality for *martingale differences*.

Definition D.5 (Martingale difference) *A sequence of random variables $V_1, V_2, \ldots$ is a martingale difference sequence with respect to $X_1, X_2, \ldots$ if for all $i > 0$, V_i is a function of $X_1, \ldots, X_i$ and*

$$\mathbb{E}[V_{i+1}|X_1, \ldots, X_i] = 0\,. \tag{D.9}$$

The following result is similar to Hoeffding's lemma.

Lemma D.6 *Let V and Z be random variables satisfying $\mathbb{E}[V|Z] = 0$ and, for some function f and constant $c \geq 0$, the inequalities:*

$$f(Z) \leq V \leq f(Z) + c\,. \tag{D.10}$$

Then, for all $t > 0$, the following upper bound holds:

$$\mathbb{E}[e^{tV}|Z] \leq e^{t^2c^2/8}\,. \tag{D.11}$$

Proof: The proof follows using the same steps as in that of lemma D.1 with conditional expectations used instead of expectations: conditioned on Z, V takes values in $[a, b]$ with $a = f(Z)$ and $b = f(Z) + c$ and its expectation vanishes. □

The lemma is used to prove the following theorem, which is one of the main results of this section.

Theorem D.7 (Azuma's inequality) *Let $V_1, V_2, \ldots$ be a martingale difference sequence with respect to the random variables $X_1, X_2, \ldots$, and assume that for all $i > 0$ there is a constant $c_i \geq 0$ and random variable Z_i, which is a function of $X_1, \ldots, X_{i-1}$, that satisfy*

$$Z_i \leq V_i \leq Z_i + c_i\,. \tag{D.12}$$

Then, for all $\epsilon > 0$ and m, the following inequalities hold:

$$\mathbb{P}\left[\sum_{i=1}^m V_i \geq \epsilon\right] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2}\right) \tag{D.13}$$

$$\mathbb{P}\left[\sum_{i=1}^m V_i \leq -\epsilon\right] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2}\right)\,. \tag{D.14}$$

Proof: For any $k \in [m]$, let $S_k = \sum_{i=1}^k V_i$. Then, using Chernoff's bounding technique, for any $t > 0$, we can write

$$\begin{aligned}
\mathbb{P}\left[S_m \geq \epsilon\right] &\leq e^{-t\epsilon}\,\mathbb{E}[e^{tS_m}] \\
&= e^{-t\epsilon}\,\mathbb{E}\left[e^{tS_{m-1}}\,\mathbb{E}[e^{tV_m}|X_1,\ldots,X_{m-1}]\right] \\
&\leq e^{-t\epsilon}\,\mathbb{E}[e^{tS_{m-1}}]e^{t^2c_m^2/8} && \text{(lemma D.6)} \\
&\leq e^{-t\epsilon}e^{t^2\sum_{i=1}^m c_i^2/8} && \text{(iterating previous argument)} \\
&= e^{-2\epsilon^2/\sum_{i=1}^m c_i^2},
\end{aligned}$$

where we chose $t = 4\epsilon/\sum_{i=1}^m c_i^2$ to minimize the upper bound. This proves the first statement of the theorem, and the second statement is shown in a similar way. □

D.7 McDiarmid's inequality

The following is the main result of this section. Its proof makes use of Azuma's inequality.

Theorem D.8 (McDiarmid's inequality) *Let $X_1, \ldots, X_m \in \mathcal{X}^m$ be a set of $m \geq 1$ independent random variables and assume that there exist $c_1, \ldots, c_m > 0$ such that $f\colon \mathcal{X}^m \to \mathbb{R}$ satisfies the following conditions:*

$$\left|f(x_1,\ldots,x_i,\ldots,x_m) - f(x_1,\ldots,x_i',\ldots x_m)\right| \leq c_i\,, \tag{D.15}$$

for all $i \in [m]$ and any points $x_1, \ldots, x_m, x_i' \in \mathcal{X}$. Let $f(S)$ denote $f(X_1, \ldots, X_m)$, then, for all $\epsilon > 0$, the following inequalities hold:

$$\mathbb{P}[f(S) - \mathbb{E}[f(S)] \geq \epsilon] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2}\right) \tag{D.16}$$

$$\mathbb{P}[f(S) - \mathbb{E}[f(S)] \leq -\epsilon] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2}\right). \tag{D.17}$$

Proof: Define a sequence of random variables V_k, $k \in [m]$, as follows: $V = f(S) - \mathbb{E}[f(S)]$, $V_1 = \mathbb{E}[V|X_1] - \mathbb{E}[V]$, and for $k > 1$,

$$V_k = \mathbb{E}[V|X_1,\ldots,X_k] - \mathbb{E}[V|X_1,\ldots,X_{k-1}]\,.$$

Note that $V = \sum_{k=1}^m V_k$. Furthermore, the random variable $\mathbb{E}[V|X_1,\ldots,X_k]$ is a function of $X_1, \ldots, X_k$. Conditioning on $X_1, \ldots, X_{k-1}$ and taking its expectation is therefore:

$$\mathbb{E}\left[\,\mathbb{E}[V|X_1,\ldots,X_k]|X_1,\ldots,X_{k-1}\right] = \mathbb{E}[V|X_1,\ldots,X_{k-1}],$$

which implies $\mathbb{E}[V_k|X_1,\ldots,X_{k-1}] = 0$. Thus, the sequence $(V_k)_{k\in[m]}$ is a martingale difference sequence. Next, observe that, since $\mathbb{E}[f(S)]$ is a scalar, V_k can be expressed as follows:

$$V_k = \mathbb{E}[f(S)|X_1,\ldots,X_k] - \mathbb{E}[f(S)|X_1,\ldots,X_{k-1}]\,.$$

Thus, we can define an upper bound W_k and lower bound U_k for V_k by:

$$\begin{aligned}
W_k &= \sup_x \mathbb{E}[f(S)|X_1,\ldots,X_{k-1},x] - \mathbb{E}[f(S)|X_1,\ldots,X_{k-1}] \\
U_k &= \inf_x \mathbb{E}[f(S)|X_1,\ldots,X_{k-1},x] - \mathbb{E}[f(S)|X_1,\ldots,X_{k-1}].
\end{aligned}$$

Now, by (D.15), for any $k \in [m]$, the following holds:

$$W_k - U_k = \sup_{x,x'} \mathbb{E}[f(S)|X_1,\ldots,X_{k-1},x] - \mathbb{E}[f(S)|X_1,\ldots,X_{k-1},x'] \leq c_k\,, \tag{D.18}$$

thus, $U_k \leq V_k \leq U_k + c_k$. In view of these inequalities, we can apply Azuma's inequality to $V = \sum_{k=1}^m V_k$, which yields exactly (D.16) and (D.17). □

McDiarmid's inequality is used in several of the proofs in this book. It can be understood in terms of stability: if changing any of its argument affects f only in a limited way, then, its deviations from its mean can be exponentially bounded. Note also that Hoeffding's inequality (theorem D.2) is a special instance of McDiarmid's inequality where f is defined by $f\colon (x_1, \ldots, x_m) \mapsto \frac{1}{m} \sum_{i=1}^m x_i$.

D.8 Normal distribution tails: Lower bound

If N is a random variable following the standard normal distribution, then for $u > 0$,

$$\mathbb{P}[N \geq u] \geq \frac{1}{2}\Big(1 - \sqrt{1 - e^{-u^2}}\Big). \tag{D.19}$$

D.9 Khintchine-Kahane inequality

The following inequality is useful in a variety of different contexts, including in the proof of a lower bound for the empirical Rademacher complexity of linear hypotheses (chapter 6).

Theorem D.9 (Khintchine-Kahane inequality) *Let $(\mathbb{H}, \|\cdot\|)$ be a normed vector space and let $\mathbf{x}_1, \ldots, \mathbf{x}_m$ be $m \geq 1$ elements of $\mathbb{H}$. Let $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)^\top$ with σ_is independent uniform random variables taking values in $\{-1, +1\}$ (Rademacher variables). Then, the following inequalities hold:*

$$\frac{1}{2} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \Big[\Big\| \sum_{i=1}^m \sigma_i \mathbf{x}_i \Big\|^2\Big] \leq \Big(\mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \Big[\Big\| \sum_{i=1}^m \sigma_i \mathbf{x}_i \Big\|\Big]\Big)^2 \leq \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \Big[\Big\| \sum_{i=1}^m \sigma_i \mathbf{x}_i \Big\|^2\Big]. \tag{D.20}$$

Proof: The second inequality is a direct consequence of the convexity of $x \mapsto x^2$ and Jensen's inequality (theorem B.20).

To prove the left-hand side inequality, first note that for any $\beta_1, \ldots, \beta_m \in \mathbb{R}$, expanding the product $\prod_{i=1}^m (1 + \beta_i)$ leads exactly to the sum of all monomials $\beta_1^{\delta_1} \cdots \beta_m^{\delta_m}$, with exponents $\delta_1, \ldots, \delta_m$ in $\{0, 1\}$. We will use the notation $\beta_1^{\delta_1} \cdots \beta_m^{\delta_m} = \beta^{\boldsymbol{\delta}}$ and $|\boldsymbol{\delta}| = \sum_{i=1}^m \delta_m$ for any $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_m) \in \{0, 1\}^m$. In view of that, for any $(\alpha_1, \ldots, \alpha_m) \in \mathbb{R}^m$ and $t > 0$, the following equality holds:

$$t^2 \prod_{i=1}^m (1 + \alpha_i / t) = t^2 \sum_{\boldsymbol{\delta} \in \{0,1\}^m} \alpha^{\boldsymbol{\delta}} / t^{|\boldsymbol{\delta}|} = \sum_{\boldsymbol{\delta} \in \{0,1\}^m} t^{2 - |\boldsymbol{\delta}|} \alpha^{\boldsymbol{\delta}}.$$

Differentiating both sides with respect to t and setting $t = 1$ yields

$$2 \prod_{i=1}^m (1 + \alpha_i) - \sum_{j=1}^m \alpha_j \prod_{i \neq j} (1 + \alpha_i) = \sum_{\boldsymbol{\delta} \in \{0,1\}^m} (2 - |\boldsymbol{\delta}|) \alpha^{\boldsymbol{\delta}}. \tag{D.21}$$

For any $\boldsymbol{\sigma} \in \{-1, +1\}^m$, let $S_{\boldsymbol{\sigma}}$ be defined by $S_{\boldsymbol{\sigma}} = \|s_{\boldsymbol{\sigma}}\|$ with $s_{\boldsymbol{\sigma}} = \sum_{i=1}^m \sigma_i \mathbf{x}_i$. Then, setting $\alpha_i = \sigma_i \sigma'_i$, multiplying both sides of (D.21) by $S_{\boldsymbol{\sigma}} S_{\boldsymbol{\sigma}'}$, and taking the sum over all $\boldsymbol{\sigma}, \boldsymbol{\sigma}' \in \{-1, +1\}^m$ yields

$$\begin{aligned}
\sum_{\boldsymbol{\sigma}, \boldsymbol{\sigma}' \in \{-1,+1\}^m} \Big(2 \prod_{i=1}^m (1 + \sigma_i \sigma'_i) - \sum_{j=1}^m \sigma_j \sigma'_j \prod_{i \neq j} (1 + \sigma_i \sigma'_i)\Big) S_{\boldsymbol{\sigma}} S_{\boldsymbol{\sigma}'} \\
&= \sum_{\boldsymbol{\sigma}, \boldsymbol{\sigma}' \in \{-1,+1\}^m} \sum_{\boldsymbol{\delta} \in \{0,1\}^m} (2 - |\boldsymbol{\delta}|) \sigma^{\boldsymbol{\delta}} \sigma'^{\boldsymbol{\delta}} S_{\boldsymbol{\sigma}} S_{\boldsymbol{\sigma}'} \\
&= \sum_{\boldsymbol{\delta} \in \{0,1\}^m} (2 - |\boldsymbol{\delta}|) \sum_{\boldsymbol{\sigma}, \boldsymbol{\sigma}' \in \{-1,+1\}^m} \sigma^{\boldsymbol{\delta}} \sigma'^{\boldsymbol{\delta}} S_{\boldsymbol{\sigma}} S_{\boldsymbol{\sigma}'} \\
&= \sum_{\boldsymbol{\delta} \in \{0,1\}^m} (2 - |\boldsymbol{\delta}|) \Big[\sum_{\boldsymbol{\sigma} \in \{-1,+1\}^m} \sigma^{\boldsymbol{\delta}} S_{\boldsymbol{\sigma}}\Big]^2.
\end{aligned} \tag{D.22}$$

Note that the terms of the right-hand sum with $|\boldsymbol{\delta}| \geq 2$ are non-positive. The terms with $|\boldsymbol{\delta}| = 1$ are null: since $S_{\boldsymbol{\sigma}} = S_{-\boldsymbol{\sigma}}$, we have $\sum_{\boldsymbol{\sigma} \in \{-1,+1\}^m} \sigma^{\boldsymbol{\delta}} S_{\boldsymbol{\sigma}} = 0$ in that case. Thus, the right-hand side can be upper bounded by the term with $\boldsymbol{\delta} = 0$, that is, $2\Big(\sum_{\boldsymbol{\sigma} \in \{-1,+1\}^m} S_{\boldsymbol{\sigma}}\Big)^2$. The left-hand side of (D.22) can be rewritten as follows:

$$\begin{aligned}
&\sum_{\boldsymbol{\sigma}\in\{-1,+1\}^m} (2^{m+1} - m2^{m-1}) S_{\boldsymbol{\sigma}}^2 + 2^{m-1} \sum_{\substack{\boldsymbol{\sigma}\in\{-1,+1\}^m \\ \boldsymbol{\sigma}'\in B(\boldsymbol{\sigma},1)}} S_{\boldsymbol{\sigma}} S_{\boldsymbol{\sigma}'} \\
&\quad = 2^m \sum_{\boldsymbol{\sigma}\in\{-1,+1\}^m} S_{\boldsymbol{\sigma}}^2 + 2^{m-1} \sum_{\boldsymbol{\sigma}\in\{-1,+1\}^m} S_{\boldsymbol{\sigma}} \bigg(\sum_{\boldsymbol{\sigma}'\in B(\boldsymbol{\sigma},1)} S_{\boldsymbol{\sigma}'} - (m-2) S_{\boldsymbol{\sigma}} \bigg), \qquad \text{(D.23)}
\end{aligned}$$

where $B(\boldsymbol{\sigma}, 1)$ denotes the set of $\boldsymbol{\sigma}'$ that differ from $\boldsymbol{\sigma}$ in exactly one coordinate $j \in [m]$, that is the set of $\boldsymbol{\sigma}'$ with Hamming distance one from $\boldsymbol{\sigma}$. Note that for any such $\boldsymbol{\sigma}'$, $s_{\boldsymbol{\sigma}} - s_{\boldsymbol{\sigma}'} = 2\sigma_j \mathbf{x}_j$ for one coordinate $j \in [m]$, thus, $\sum_{\boldsymbol{\sigma}'\in B(\boldsymbol{\sigma},1)} s_{\boldsymbol{\sigma}} - s_{\boldsymbol{\sigma}'} = 2 s_{\boldsymbol{\sigma}}$. In light of that and using the triangle inequality, we can write

$$\begin{aligned}
(m-2) S_{\boldsymbol{\sigma}} = \|m s_{\boldsymbol{\sigma}}\| - \|2 s_{\boldsymbol{\sigma}}\| &= \Big\| \sum_{\boldsymbol{\sigma}'\in B(\boldsymbol{\sigma},1)} s_{\boldsymbol{\sigma}} \Big\| - \Big\| \sum_{\boldsymbol{\sigma}'\in B(\boldsymbol{\sigma},1)} s_{\boldsymbol{\sigma}} - s_{\boldsymbol{\sigma}'} \Big\| \\
&\le \Big\| \sum_{\boldsymbol{\sigma}'\in B(\boldsymbol{\sigma},1)} s_{\boldsymbol{\sigma}'} \Big\| \le \sum_{\boldsymbol{\sigma}'\in B(\boldsymbol{\sigma},1)} S_{\boldsymbol{\sigma}'}.
\end{aligned}$$

Thus, the second sum of (D.23) is non-negative and the left-hand side of (D.22) can be lower bounded by the first sum $2^m \sum_{\boldsymbol{\sigma}\in\{-1,+1\}^m} S_{\boldsymbol{\sigma}}^2$. Combining this with the upper bound found for (D.22) gives

$$2^m \sum_{\boldsymbol{\sigma}\in\{-1,+1\}^m} S_{\boldsymbol{\sigma}}^2 \le 2 \Big[\sum_{\boldsymbol{\sigma}\in\{-1,+1\}^m} S_{\boldsymbol{\sigma}} \Big]^2.$$

Dividing both sides by 2^{2m} and using $\mathbb{P}[\boldsymbol{\sigma}] = 1/2^m$ gives $\mathbb{E}_{\boldsymbol{\sigma}}[S_{\boldsymbol{\sigma}}^2] \le 2(\mathbb{E}_{\boldsymbol{\sigma}}[S_{\boldsymbol{\sigma}}])^2$ and completes the proof. □

The constant $1/2$ appearing in the first inequality of (D.20) is optimal. To see this, consider the case where $m = 2$ and $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{x}$ for some non-zero vector $\mathbf{x} \in \mathbb{H}$. Then, the left-hand side of the first inequality is $\frac{1}{2}\sum_{i=1}^m \|\mathbf{x}_i\|^2 = \|\mathbf{x}\|^2$ and the right-hand side $\big(\mathbb{E}_{\boldsymbol{\sigma}}\big[\|(\sigma_1 + \sigma_2)\mathbf{x}\|\big]\big)^2 = \|\mathbf{x}\|^2 (\mathbb{E}_{\boldsymbol{\sigma}}[|\sigma_1 + \sigma_2|])^2 = \|\mathbf{x}\|^2$.

Note that when the norm $\|\cdot\|$ corresponds to an inner product, as in the case of a Hilbert space $\mathbb{H}$, we can write

$$\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\Big\|\sum_{i=1}^m \sigma_i \mathbf{x}_i\Big\|^2\Big] = \sum_{i,j=1}^m \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\sigma_i\sigma_j(\mathbf{x}_i\cdot\mathbf{x}_j)\Big] = \sum_{i,j=1}^m \mathop{\mathbb{E}}_{\boldsymbol{\sigma}}[\sigma_i\sigma_j](\mathbf{x}_i\cdot\mathbf{x}_j) = \sum_{i=1}^m \|\mathbf{x}_i\|^2,$$

since by the independence of the random variables σ_i, for $i \ne j$, $\mathbb{E}_{\boldsymbol{\sigma}}[\sigma_i\sigma_j] = \mathbb{E}_{\boldsymbol{\sigma}}[\sigma_i]\,\mathbb{E}_{\boldsymbol{\sigma}}[\sigma_j] = 0$. Thus, (D.20) can then be rewritten as follows:

$$\frac{1}{2}\sum_{i=1}^m \|\mathbf{x}_i\|^2 \le \bigg(\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\Big[\Big\|\sum_{i=1}^m \sigma_i\mathbf{x}_i\Big\|\Big]\bigg)^2 \le \sum_{i=1}^m \|\mathbf{x}_i\|^2. \qquad \text{(D.24)}$$

D.10 Maximal inequality

The following gives an upper bound on the expectation of the maximum of a finite set of random variables that is useful in several contexts.

Theorem D.10 (Maximal inequality) *Let $X_1 \ldots X_n$ be $n \ge 1$ real-valued random variables such that for all $j \in [n]$ and $t > 0$, $\mathbb{E}[e^{tX_j}] \le e^{\frac{t^2r^2}{2}}$ for some $r > 0$. Then, the following inequality holds:*

$$\mathbb{E}\Big[\max_{j\in[n]} X_j\Big] \le r\sqrt{2\log n}.$$

Proof: For any $t > 0$, by the convexity of exp and Jensen's inequality, the following holds:

$$e^{t\,\mathbb{E}[\max_{j\in[n]} X_j]} \le \mathbb{E}[e^{t\max_{j\in[n]} X_j}] = \mathbb{E}\Big[\max_{j\in[n]} e^{tX_j}\Big] \le \mathbb{E}\Big[\sum_{j\in[n]} e^{tX_j}\Big] \le n e^{\frac{t^2r^2}{2}}.$$

Taking the log of both sides yields

$$\mathbb{E}\Big[\max_{j\in[n]} X_j\Big] \le \frac{\log n}{t} + \frac{tr^2}{2}. \tag{D.25}$$

Choosing $t = \frac{\sqrt{2\log n}}{r}$, which minimizes the right-hand side, gives the upper bound $r\sqrt{2\log n}$. □

Note that, in view of the expression of their moment-generating function (equation (C.24)), for standard Gaussian random variables X_j, the assumptions of the theorem hold as equalities: $\mathbb{E}[e^{tX_j}] = e^{\frac{t^2}{2}}$.

Corollary D.11 (Maximal inequality) *Let $X_1 \ldots X_n$ be $n \ge 1$ real-valued random variables such that for all $j \in [n]$, $X_j = \sum_{i=1}^m Y_{ij}$ where, for each fixed $j \in [n]$, Y_{ij} are independent zero mean random variables taking values in $[-r_i, +r_i]$, for some $r_i > 0$. Then, the following inequality holds:*

$$\mathbb{E}\Big[\max_{j\in[n]} X_j\Big] \le r\sqrt{2\log n},$$

with $r = \sqrt{\sum_{i=1}^m r_i^2}$.

Proof: By the independence of the Y_{ij}s for fixed j and Hoeffding's lemma (Lemma D.1), the following inequality holds for all $j \in [n]$:

$$\mathbb{E}[e^{tX_j}] = \mathbb{E}\Big[\prod_{i=1}^m e^{tY_{ij}}\Big] = \prod_{i=1}^m \mathbb{E}[e^{tY_{ij}}] \le \prod_{i=1}^m e^{\frac{t^2 r_j^2}{2}} = e^{\frac{t^2 r^2}{2}}, \tag{D.26}$$

with $r^2 = \sum_{i=1}^m r_i^2$. The result then follows immediately by Theorem D.10. □

D.11 Chapter notes

Several of the concentration inequalities presented in this chapter are based on a bounding technique due to Chernoff [1952]. Theorem D.3 is due to Sanov [1957]. For the exponential inequality of exercise D.7, which is an alternative form of Sanov's inequality, see [Hagerup and Rüb, 1990] and the references therein. The multiplicative Chernoff bounds presented in this chapter (Theorem D.4) were given by Angluin and Valiant [1979]. Hoeffding's inequality and lemma (Lemma D.1 and Theorem D.2) are due to Hoeffding [1963]. The improved version of Azuma's inequality [Hoeffding, 1963, Azuma, 1967] presented in this chapter is due to McDiarmid [1989]. The improvement is a reduction of the exponent by a factor of 4. This also appears in McDiarmid's inequality, which is derived from the inequality for bounded martingale sequences. The inequalities presented in exercise D.6 are due to Bernstein [1927] and Bennett [1962]; the exercise is from Devroye and Lugosi [1995].

The binomial inequality of section D.5 is due to Slud [1977]. The tail bound of section D.8 is due to Tate [1953] (see also Anthony and Bartlett [1999]). The Khintchine-Kahane inequality was first studied in the case of real-valued variables $x_1, \ldots, x_m$ by Khintchine [1923], with better constants and simpler proofs later provided by Szarek [1976], Haagerup [1982], and Tomaszewski [1982]. The inequality was extended to normed vector spaces by Kahane [1964]. The proof presented here is due to Latała and Oleszkiewicz [1994] and provides the best possible constants.

D.12 Exercises

D.1 Twins paradox. Professor Mamoru teaches at a university whose computer science and math building has $F = 30$ floors.

(1) Assume that the floors are independent and that they have the same probability to be selected by someone taking the elevator. How many people should take the elevator in order to make it likely (probability more than half) that two of them go to the same floor?

(*Hint*: use the Taylor series expansion of $e^{-x} = 1 - x + \ldots$ and give an approximate general expression of the solution.)

(2) Professor Mamoru is popular, and his floor is in fact more likely to be selected than others. Assuming that all other floors are equiprobable, derive the general expression of the probability that two people go to the same floor, using the same approximation as before. How many people should take the elevator in order to make it likely that two of them go to the same floor when the probability of Professor Mamoru's floor is .25, .35, or .5? When $q = .5$, would the answer change if the number of floors were instead $F = 1{,}000$?

(3) The probability models assumed in (1) and (2) are both naive. If you had access to the data collected by the elevator guard, how would you define a more faithful model?

D.2 Estimating label bias. Let $\mathcal{D}$ be a distribution over $\mathcal{X}$ and let $f\colon \mathcal{X} \times \{-1, +1\}$ be a labeling function. Suppose we wish to find a good approximation of the label bias of the distribution $\mathcal{D}$, that is of p_+ defined by:

$$p_+ = \mathop{\mathbb{P}}_{x \sim \mathcal{D}}[f(x) = +1]. \tag{D.27}$$

Let $\mathcal{S}$ be a finite labeled sample of size m drawn i.i.d. according to $\mathcal{D}$. Use $\mathcal{S}$ to derive an estimate $\widehat{p}_+$ of p_+. Show that for any $\delta > 0$, with probability at least $1 - \delta$, $|p_+ - \widehat{p}_+| \leq \sqrt{\frac{\log(2/\delta)}{2m}}$.

D.3 Biased coins. Professor Moent has two coins in his pocket, coin x_A and coin x_B. Both coins are slightly biased, i.e., $\mathbb{P}[x_A = 0] = 1/2 - \epsilon/2$ and $\mathbb{P}[x_B = 0] = 1/2 + \epsilon/2$, where $0 < \epsilon < 1$ is a small positive number, 0 denotes heads and 1 denotes tails. He likes to play the following game with his students. He picks a coin $x \in \{x_A, x_B\}$ from his pocket uniformly at random, tosses it m times, reveals the sequence of 0s and 1s he obtained and asks which coin was tossed. Determine how large m needs to be for a student's coin prediction error to be at most $\delta > 0$.

(a) Let S be a sample of size m. Professor Moent's best student, Oskar, plays according to the decision rule $f_o\colon \{0,1\}^m \to \{x_A, x_B\}$ defined by $f_o(S) = x_A$ iff $N(S) < m/2$, where $N(S)$ is the number of 0's in sample S.

Suppose m is even, then show that

$$error(f_o) \geq \frac{1}{2}\, \mathbb{P}\left[N(S) \geq \frac{m}{2} \middle| x = x_A\right]. \tag{D.28}$$

(b) Assuming m even, show that

$$error(f_o) > \frac{1}{4}\left[1 - \left[1 - e^{-\frac{m\epsilon^2}{1-\epsilon^2}}\right]^{\frac{1}{2}}\right]. \tag{D.29}$$

(c) Argue that if m is odd, the probability can be lower bounded by using $m+1$ in the bound in (a) and conclude that for both odd and even m,

$$error(f_o) > \frac{1}{4}\left[1 - \left[1 - e^{-\frac{2\lceil m/2\rceil\epsilon^2}{1-\epsilon^2}}\right]^{\frac{1}{2}}\right]. \tag{D.30}$$

(d) Using this bound, how large must m be if Oskar's error is at most δ, where $0 < \delta < 1/4$. What is the asymptotic behavior of this lower bound as a function of ϵ?

(e) Show that no decision rule $f\colon \{0,1\}^m \to \{x_A, x_B\}$ can do better than Oskar's rule f_o. Conclude that the lower bound of the previous question applies to all rules.

D.4 Concentration bounds. Let X be a non-negative random variable satisfying $\mathbb{P}[X > t] \leq ce^{-2mt^2}$ for all $t > 0$ and some $c > 0$. Show that $\mathbb{E}[X^2] \leq \frac{\log(ce)}{2m}$ (*Hint*: to do that, use the identity $\mathbb{E}[X^2] = \int_0^{+\infty} \mathbb{P}[X^2 > t]dt$, write $\int_0^{+\infty} = \int_0^u + \int_u^{+\infty}$, bound the first term by u and find the best u to minimize the upper bound).

D.5 Comparison of Hoeffding's and Chebyshev's inequalities. Let $X_1, \ldots, X_m$ be a sequence of random variables taking values in $[0, 1]$ with the same mean μ and variance $\sigma^2 < \infty$ and let $\overline{X} = \frac{1}{m}\sum_{i=1}^m X_i$.

(a) For any $\epsilon > 0$, give a bound on $\mathbb{P}[|\overline{X} - \mu| > \epsilon]$ using Chebyshev's inequality, then Hoeffding's inequality. For what values of σ is Chebyshev's inequality tighter?

(b) Assume that the random variables X_i take values in $\{0, 1\}$. Show that $\sigma^2 \leq \frac{1}{4}$. Use this to simplify Chebyshev's inequality. Choose $\epsilon = .05$ and plot Chebyshev's inequality thereby modified and Hoeffding's inequality as a function of m (you can use your preferred program for generating the plots).

D.6 Bennett's and Bernstein's inequalities. The objective of this problem is to prove these two inequalities.

(a) Show that for any $t > 0$, and any random variable X with $\mathbb{E}[X] = 0$, $\mathbb{E}[X^2] = \sigma^2$, and $X \leq c$,

$$\mathbb{E}[e^{tX}] \leq e^{f(\sigma^2/c^2)}, \tag{D.31}$$

where

$$f(x) = \log\Big(\frac{1}{1+x}e^{-ctx} + \frac{x}{1+x}e^{ct}\Big).$$

(b) Show that $f''(x) \leq 0$ for $x \geq 0$.

(c) Using Chernoff's bounding technique, show that

$$\mathbb{P}\left[\frac{1}{m}\sum_{i=1}^m X_i \geq \epsilon\right] \leq e^{-tm\epsilon + \sum_{i=1}^m f(\sigma^2_{X_i}/c^2)},$$

where $(\sigma^2_{X_i}$ is the variance of X_i.

(d) Show that $f(x) \leq f(0) + xf'(0) = (e^{ct} - 1 - ct)x$.

(e) Using the bound derived in (4), find the optimal value of t.

(f) *Bennett's inequality*. Let $X_1, \ldots, X_m$ be independent real-valued random variables with zero mean such that for $i = 1, \ldots, m$, $X_i \leq c$. Let $\sigma^2 = \frac{1}{m}\sum_{i=1}^m \sigma^2_{X_i}$. Show that

$$\mathbb{P}\left[\frac{1}{m}\sum_{i=1}^m X_i > \epsilon\right] \leq \exp\left(-\frac{m\sigma^2}{c^2}\theta\Big(\frac{\epsilon c}{\sigma^2}\Big)\right), \tag{D.32}$$

where $\theta(x) = (1 + x)\log(1 + x) - x$.

(g) *Bernstein's inequality*. Show that under the same conditions as Bennett's inequality

$$\mathbb{P}\left[\frac{1}{m}\sum_{i=1}^m X_i > \epsilon\right] \leq \exp\Big(-\frac{m\epsilon^2}{2\sigma^2 + 2c\epsilon/3}\Big). \tag{D.33}$$

(*Hint*: show that for all $x \geq 0$, $\theta(x) \geq h(x) = \frac{3}{2}\frac{x^2}{x+3}$.)

(h) Write Hoeffding's inequality assuming the same conditions. For what values of σ is Bernstein's inequality better than Hoeffding's inequality?

D.7 Exponential inequality. Let X be a random variable following a binomial distribution $B(m, p)$.

(a) Use Sanov's inequality to show that the following *exponential inequality* holds for any $\epsilon > 0$:

$$\mathbb{P}\left[\frac{X}{m} - p > \epsilon\right] \leq \left[\left(\tfrac{p}{p+\epsilon}\right)^{p+\epsilon}\left(\tfrac{1-p}{1-(p+\epsilon)}\right)^{1-(p+\epsilon)}\right]^m. \tag{D.34}$$

(b) Use that to show that the following holds:

$$\mathbb{P}\left[\frac{X}{m} - p > \epsilon\right] \leq \left(\tfrac{p}{p+\epsilon}\right)^{m(p+\epsilon)} e^{m\epsilon}. \tag{D.35}$$

(c) Prove that

$$\mathbb{P}\left[\frac{X}{m} - p > \epsilon\right] \leq e^{-mp\theta(\frac{\epsilon}{p})}, \tag{D.36}$$

where θ is defined as in exercise D.6.

E Notions of Information Theory

This chapter introduces some basic notions of information theory useful for the presentation of several learning algorithms and their properties. The definitions and theorems are given in the case of discrete random variables or distributions, but they can be straightforwardly extended to the continuous case.

We start with the notion of *entropy*, which can be viewed as a measure of the uncertainty of a random variable.

E.1 Entropy

Definition E.1 (Entropy) *The* entropy *of a discrete random variable X with probability mass function $p(x) = \mathbb{P}[X = x]$ is denoted by $\mathsf{H}(X)$ and defined by*

$$\mathsf{H}(X) = -\mathbb{E}\left[\log(p(X))\right] = -\sum_{x \in \mathcal{X}} p(x) \log(p(x)). \tag{E.1}$$

We define by the same expression the entropy of a distribution p *and abusively denote that by* $\mathsf{H}(p)$.

The base of the logarithm is not critical in this definition since it only affects the value by a multiplicative constant. Thus, unless otherwise specified, we will consider the natural logarithm (base e). If we use base 2, then $-\log_2(p(x))$ is the number of bits needed to represent $p(x)$. Thus, by definition, the entropy of X can be viewed as the average number of bits (or amount of information) needed for the description of the random variable X. By the same property, the entropy is always non-negative:

$$\mathsf{H}(X) \geq 0. \tag{E.2}$$

As an example, the entropy of a biased coin X_p taking value 1 with probability p and 0 with probability $1 - p$ is given by

$$\mathsf{H}(X_p) = -p \log p - (1-p)\log(1-p). \tag{E.3}$$

The corresponding function of p is often referred to as the *binary entropy function*. Figure E.1 shows a plot of that function when using base 2 for the logarithm. As can be seen from the figure, the entropy is a concave function.[26] It reaches its maximum at $p = \frac{1}{2}$, which corresponds to the most uncertain case, and its minima at $p = 0$ or $p = 1$ which correspond to the fully certain cases. More generally, assume that the input space $\mathcal{X}$ has a finite cardinality $N \geq 1$. Then, by Jensen's

[26] We will see later that the entropy function is always concave.

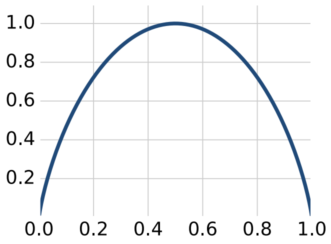


Figure E.1
A plot of the binary entropy as a function of the bias p.

inequality, in view of the concavity of logarithm, the following inequality holds:

$$\mathsf{H}(X) = \mathbb{E}\left[\log \frac{1}{p(X)}\right] \leq \log \mathbb{E}\left[\frac{1}{p(X)}\right] = \log\left(\sum_{x\in\mathcal{X}} \frac{p(x)}{p(x)}\right) = \log N. \tag{E.4}$$

Thus, more generally, the maximum value of the entropy is $\log N$, that is the entropy of the uniform distribution.

The entropy is a lower bound on lossless data compression and is therefore a critical quantity to consider in information theory. It is also closely related to the notions of entropy in thermodynamics and quantum physics.

E.2 Relative entropy

Here, we introduce a measure of divergence between two distributions p and q, *relative entropy*, which is related to the notion of entropy. The following is its definition in the discrete case.

Definition E.2 (Relative entropy) *The* relative entropy *(or Kullback-Leibler divergence) of two distributions p and q is denoted by $\mathsf{D}(p\|q)$ and defined by*

$$\mathsf{D}(p\|q) = \mathop{\mathbb{E}}_{p}\left[\log \frac{p(X)}{q(X)}\right] = \sum_{x\in\mathcal{X}} p(x) \log\left[\frac{p(x)}{q(x)}\right], \tag{E.5}$$

with the conventions $0 \log 0 = 0$, $0 \log \frac{0}{0} = 0$, and $a \log \frac{a}{0} = +\infty$ for $a > 0$.

Note that, in view of these conventions, whenever $q(x) = 0$ for some x in the support of p ($p(x) > 0$), the relative entropy is infinite: $\mathsf{D}(p\|q) = \infty$. Thus, the relative entropy does not provide an informative measure of the divergence of p and q in such cases.

As for the entropy, the base of the logarithm is not critical in the definition of the relative entropy and we will consider the natural logarithm unless otherwise specified. If we use base 2, the relative entropy can be interpreted in terms of coding length. Ideally, one could design for p an optimal code with average length the entropy $\mathsf{H}(p)$. The relative entropy is the average number of additional bits needed to encode p when using an optimal code for q instead of one for p since it can be expressed as the difference $\mathsf{D}(p\|q) = \mathbb{E}_p[\log \frac{1}{q(X)}] - \mathsf{H}(p)$, which, as shown by the following proposition, is always non-negative.

Proposition E.3 (Non-negativity of relative entropy) *For any two distributions p and q, the following inequality holds:*

$$\mathsf{D}(p\|q) \geq 0. \tag{E.6}$$

Furthermore, $\mathsf{D}(p\|q) = 0$ iff $p = q$.

Proof: By the concavity of logarithm and Jensen's inequality, the following holds:

$$-\mathsf{D}(p\|q) = \sum_{x\colon p(x)>0} p(x)\log\left(\frac{q(x)}{p(x)}\right) \leq \log\Bigg(\sum_{x\colon p(x)>0} p(x)\frac{q(x)}{p(x)}\Bigg)$$

$$= \log\Bigg(\sum_{x\colon p(x)>0} q(x)\Bigg) \leq \log(1) = 0.$$

Thus, the relative entropy is always non-negative for all distributions p and q. The equality $\mathsf{D}(p\|q) = 0$ can hold only if both of the inequalities above are equalities. The last one implies that $\sum_{x\colon p(x)>0} q(x) = 1$. Since the log function is strictly concave, the first inequality can be an equality only if $\frac{q(x)}{p(x)}$ is some constant α over $\{x\colon p(x) > 0\}$. Since $p(x)$ sums to one over that set, we must have $\sum_{x\colon p(x)>0} q(x) = \alpha$. Thus, $\alpha = 1$, which implies $q(x) = p(x)$ for all $x \in \{x\colon p(x) > 0\}$ and thus for all x. Finally, by definition, for any distribution p, $\mathsf{D}(p\|p) = 0$, which completes the proof. □

The relative entropy is not a distance. It is asymmetric: in general, $\mathsf{D}(p\|q) \neq \mathsf{D}(q\|p)$ for two distributions p and q. Furthermore, in general, the relative entropy does not verify the triangle inequality.

Corollary E.4 (Log-sum inequality) *For any set of non-negative real numbers $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, the following inequality holds:*

$$\sum_{i=1}^{n} a_i \log\left(\frac{a_i}{b_i}\right) \geq \Big(\sum_{i=1}^{n} a_i\Big)\log\left(\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}\right), \tag{E.7}$$

with the conventions $0\log 0 = 0$, $0\log\frac{0}{0} = 0$, and $a\log\frac{a}{0} = +\infty$ for $a > 0$.

Furthermore, equality holds in (E.7) iff $\frac{a_i}{b_i}$ is a constant (does not depend on i).

Proof: With the conventions adopted, it is clear that the equality holds if $\sum_{i=1}^{n} a_i = 0$, that is $a_i = 0$ for all $i \in [n]$, or $\sum_{i=1}^{n} b_i = 0$, that is $b_i = 0$ for all $i \in [n]$. Thus, we can assume that $\sum_{i=1}^{n} a_i \neq 0$ and $\sum_{i=1}^{n} b_i \neq 0$. Since the inequality is invariant by scaling of the a_is or b_is, we can multiply them by positive constants such that $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i = 1$. The inequality then coincides with the non-negativity of the relative entropy of the distributions thereby defined by a_is and b_is and the result holds by Proposition E.3. □

Corollary E.5 (Joint convexity of relative entropy) *The relative entropy function $(p, q) \mapsto \mathsf{D}(p\|q)$ is convex.*

Proof: For any $\alpha \in [0, 1]$ and any four probability distributions p_1, p_2, q_1, q_2, by the Log-sum inequality (Corollary E.4), the following holds for any fixed x:

$$(\alpha p_1(x) + (1-\alpha)p_2(x))\log\left[\frac{\alpha p_1(x) + (1-\alpha)p_2(x)}{\alpha q_1(x) + (1-\alpha)q_2(x)}\right]$$

$$\leq \alpha p_1(x)\log\left[\frac{\alpha p_1(x)}{\alpha q_1(x)}\right] + (1-\alpha)p_2(x)\log\left[\frac{(1-\alpha)p_1(x)}{(1-\alpha)q_2(x)}\right]. \tag{E.8}$$

Summing up these inequalities over all x yields:

$$\mathsf{D}\Big(\alpha p_1 + (1-\alpha)p_2 \| \alpha q_1 + (1-\alpha)q_2\Big) \leq \alpha\mathsf{D}(p_1\|q_1) + (1-\alpha)\mathsf{D}(p_2\|q_2), \tag{E.9}$$

which concludes the proof. □

Corollary E.6 (Concavity of the entropy) *The entropy function $p \mapsto \mathsf{H}(p)$ is concave.*

Proof: Observe that for any fixed distribution p_0 over $\mathcal{X}$, by definition of the relative entropy, we can write

$$\mathsf{D}(p\|p_0) = \sum_{x\in\mathcal{X}} p(x)\log(p(x)) - \sum_{x\in\mathcal{X}} p(x)\log(p_0(x)). \tag{E.10}$$

Thus, $\mathsf{H}(p) = -\mathsf{D}(p\|p_0) - \sum_{x\in\mathcal{X}} p(x)\log(p_0(x))$. By Corollary E.5, the first term is a concave function of p. The second term is linear in p and therefore is concave. Thus, H is concave as a sum of two concave functions. □

Proposition E.7 (Pinsker's inequality) *For any two distributions p and q, the following inequality holds:*

$$\mathsf{D}(p\|q) \geq \frac{1}{2}\|p-q\|_1^2. \tag{E.11}$$

Proof: We first show that the inequality holds for distributions over a set $\mathcal{A} = \{a_0, a_1\}$ of cardinality 2. Let $p_0 = p(a_0)$ and $q_0 = q(a_0)$. Fix $p_0 \in [0,1]$ and consider the function $f\colon q_0 \mapsto f(q_0)$ defined by

$$f(q_0) = p_0 \log\frac{p_0}{q_0} + (1-p_0)\log\frac{1-p_0}{1-q_0} - 2(p_0-q_0)^2. \tag{E.12}$$

Observe that $f(p_0) = 0$ and that for $q_0 \in (0,1)$,

$$f'(q_0) = -\frac{p_0}{q_0} + \frac{1-p_0}{1-q_0} + 4(p_0-q_0) = (q_0-p_0)\left[\frac{1}{(1-q_0)q_0} - 4\right]. \tag{E.13}$$

Since $(1-q_0)q_0 \leq \frac{1}{4}$, $[\frac{1}{(1-q_0)q_0} - 4]$ is non-negative. Thus, $f'(q_0) \leq 0$ for $q_0 \leq p_0$ and $f'(q_0) \geq 0$ for $q_0 \geq p_0$. Thus, f reaches its minimum at $q_0 = p_0$, which implies $f(q_0) \geq f(p_0) = 0$ for all q_0. Since $f(q_0)$ can be expressed as follows:

$$\begin{aligned} f(q_0) &= \mathsf{D}(p\|q) - 2(p_0-q_0)^2 && \text{(E.14)}\\ &= \mathsf{D}(p\|q) - \frac{1}{2}\Big[|p_0-q_0| + |(1-p_0)-(1-q_0)|\Big]^2 && \text{(E.15)}\\ &= \mathsf{D}(p\|q) - \frac{1}{2}\|p-q\|_1^2 \geq 0, && \text{(E.16)} \end{aligned}$$

this proves the inequality for a set $\mathcal{A} = \{a_0, a_1\}$ of cardinality 2.

Now, consider the distributions p' and q' defined over $\mathcal{A} = \{a_0, a_1\}$ with $p'(a_0) = \sum_{x\in a_0} p(x)$, and $q'(a_0) = \sum_{x\in a_0} q(x)$ where $a_0 = \{x \in \mathcal{X}\colon p(x) \geq q(x)\}$ and $a_1 = \{x \in \mathcal{X}\colon p(x) < q(x)\}$. By the Log-sum inequality (Corollary E.4),

$$\begin{aligned} \mathsf{D}(p\|q) &= \sum_{x\in a_0} p(x)\log\left[\frac{p(x)}{q(x)}\right] + \sum_{x\in a_1} p(x)\log\left[\frac{p(x)}{q(x)}\right] && \text{(E.17)}\\ &\geq p(a_0)\log\left[\frac{p(a_0)}{q(a_0)}\right] + p(a_1)\log\left[\frac{p(a_1)}{q(a_1)}\right] && \text{(E.18)}\\ &= \mathsf{D}(p'\|q'). && \text{(E.19)} \end{aligned}$$

Combining this inequality with the observation that

$$\begin{aligned} \|p'-q'\|_1 &= (p(a_0)-q(a_0)) - (p(a_1)-q(a_1)) && \text{(E.20)}\\ &= \sum_{x\in a_0}(p(x)-q(x)) - \sum_{x\in a_1}(p(x)-q(x)) && \text{(E.21)}\\ &= \sum_{x\in\mathcal{X}} |p(x)-q(x)| && \text{(E.22)}\\ &= \|p-q\|_1, && \text{(E.23)} \end{aligned}$$

shows that $\mathsf{D}(p\|q) \geq \mathsf{D}(p'\|q') \geq \frac{1}{2}\|p-q\|_1^2$ and concludes the proof. □

Definition E.8 (Conditional relative entropy) *Let p and q be two probability distributions defined over $\mathcal{X}\times\mathcal{Y}$ and r a distribution over $\mathcal{X}$. Then, the* conditional relative entropy *of p and q with respect to the marginal r is defined as the expectation of the relative entropy of $p(\cdot|X)$ and $q(\cdot|X)$ with respect to r:*

$$\mathop{\mathbb{E}}_{X\sim r}\Big[\mathsf{D}\big(p(\cdot|X)\|q(\cdot|X)\big)\Big] = \sum_{x\in\mathcal{X}} r(x)\sum_{y\in\mathcal{Y}} p(y|x)\log\frac{p(y|x)}{q(y|x)} = \mathsf{D}(\widetilde{p}\|\widetilde{q}), \tag{E.24}$$

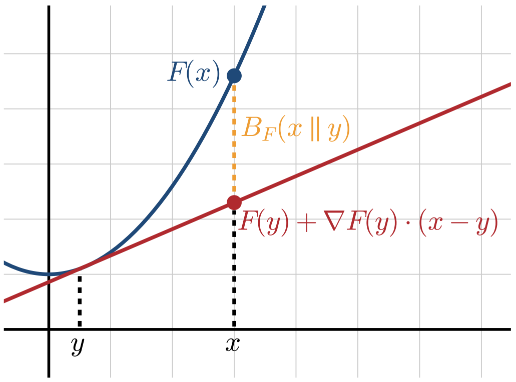


Figure E.2
Illustration of the quantity measured by the Bregman divergence defined based on a convex and differentiable function F. The divergence measures the distance between $F(x)$ and the hyperplane tangent to the curve at point y.

where $\widetilde{p}(x, y) = r(x)p(y|x)$ *and* $\widetilde{q}(x, y) = r(x)q(y|x)$*, with the conventions* $0 \log 0 = 0$*,* $0 \log \frac{0}{0} = 0$*, and* $a \log \frac{a}{0} = +\infty$ *for* $a > 0$*.*

E.3 Mutual information

Definition E.9 (Mutual information) *Let X and Y be two random variables with joint probability distribution function $p(\cdot, \cdot)$ and marginal probability distribution functions $p(x)$ and $p(y)$. Then, the* mutual information *of X and Y is denoted by $I(X, Y)$ and defined as follows:*

$$I(X, Y) = \mathsf{D}(p(x, y) \| p(x)p(y)) \tag{E.25}$$

$$= \mathop{\mathbb{E}}_{p(x,y)}\left[\log \frac{p(X, Y)}{p(X)p(Y)}\right] = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log \left[\frac{p(x, y)}{p(x)p(y)}\right], \tag{E.26}$$

with the conventions $0 \log 0 = 0$*,* $0 \log \frac{0}{0} = 0$*, and* $a \log \frac{a}{0} = +\infty$ *for* $a > 0$*.*

When the random variables X and Y are independent, their joint distributions is the product of the marginals $p(x)$ and $p(y)$. Thus, the mutual information is a measure of the closeness of the joint distribution $p(x, y)$ to its value when X and Y are independent, where closeness is measured via the relative entropy divergence. As such, it can be viewed as a measure of the amount of information that each random variable can provide about the other. Note that by Proposition E.3, the equality $I(X, Y) = 0$ holds iff $p(x, y) = p(x)p(y)$ for all x, y, that is iff X and Y are independent.

E.4 Bregman divergences

Here we introduce the so-called *unnormalized relative entropy* $\widetilde{\mathsf{D}}$ defined for all non-negative functions p, q in $\mathbb{R}^{\mathcal{X}}$ by

$$\widetilde{\mathsf{D}}(p \| q) = \sum_{x \in \mathcal{X}} p(x) \log \left[\frac{p(x)}{q(x)}\right] + \big(q(x) - p(x)\big), \tag{E.27}$$

Table E.1
Examples of Bregman divergences and corresponding convex functions.

	$\mathsf{B}_F(x \,\|\, y)$	$F(x)$
Squared L_2-distance	$\|x-y\|^2$	$\|x\|^2$
Mahalanobis distance	$(x-y)^\top Q(x-y)$	$x^\top Qx$
Unnormalized relative entropy	$\widetilde{\mathsf{D}}(x \,\|\, y)$	$\sum_{i\in I} x(i)\log(x(i)) - x(i)$

with the conventions $0\log 0 = 0$, $0\log\frac{0}{0} = 0$, and $a\log\frac{a}{0} = +\infty$ for $a > 0$. The relative entropy coincides with the unnormalized relative entropy when restricted to $\Delta\times\Delta$, where Δ is the family of distributions defined over $\mathcal{X}$. The relative entropy inherits several properties of the unnormalized relative entropy, in particular, it can be shown that $\widetilde{\mathsf{D}}(p\|q) \geq 0$. Many of these properties are in fact shared by a broader family of divergences known as *Bregman divergences*.

Definition E.10 (Bregman divergences) *Let F be a convex and differentiable function defined over a convex (open) set $\mathcal{C}$ in a Hilbert space $\mathbb{H}$. Then, the* Bregman divergence B_F *associated to F is defined for all $x, y \in \mathcal{C}$ by*

$$\mathsf{B}_F(x\|y) = F(x) - F(y) - \langle \nabla F(y), x - y\rangle\,. \tag{E.28}$$

Thus, $\mathsf{B}_F(x \,\|\, y)$ measures the difference of $F(x)$ and its linear approximation. Figure E.2 illustrates this definition. Table E.1 provides several examples of Bregman divergences along with their corresponding convex functions $F(x)$. Note that, although the unnormalized relative entropy is a Bregman divergence, the relative entropy is not a Bregman divergence since it is defined over the simplex which is not an open set and has an empty interior.

The following proposition presents several general properties of Bregman divergences.

Proposition E.11 *Let F be a convex and differentiable function defined over a convex set $\mathcal{C}$ in a Hilbert space $\mathbb{H}$. Then, the following properties hold:*

1. $\forall x, y \in \mathcal{C},\ \mathsf{B}_F(x \,\|\, y) \geq 0$.
2. $\forall x, y, z \in \mathcal{C}, \langle \nabla F(x) - \nabla F(y), x - z\rangle = \mathsf{B}_F(x \,\|\, y) + \mathsf{B}_F(z \,\|\, x) - \mathsf{B}_F(z \,\|\, y)$.
3. *B_F is convex in its first argument. If additionally F is strictly convex, then B_F is strictly convex in its first argument.*
4. *Linearity: let G be a convex and differentiable function over $\mathcal{C}$, then, for any $\alpha, \beta \in \mathbb{R}$, $\mathsf{B}_{\alpha F + \beta G} = \alpha \mathsf{B}_F + \beta \mathsf{B}_G$.*

   For the following properties, we will assume additionally that F is strictly convex.

5. *Projection: for any $y \in \mathcal{C}$ and any closed convex set $\mathcal{K} \subseteq \mathcal{C}$, the* B_F-projection of y over K, *$P_{\mathcal{K}}(y) = \operatorname{argmin}_{x\in\mathcal{K}} \mathsf{B}_F(x \,\|\, y)$, is unique.*
6. *Pythagorean theorem: for $y \in \mathcal{C}$ and any closed convex set $\mathcal{K} \subseteq \mathcal{C}$, the following holds for all $x \in \mathcal{K}$:*
$$\mathsf{B}_F(x \,\|\, y) \geq \mathsf{B}_F(x \,\|\, P_{\mathcal{K}}(y)) + \mathsf{B}_F(P_{\mathcal{K}}(y) \,\|\, y). \tag{E.29}$$
7. *Conjugate divergence: assume that F is closed proper strictly convex, and that the norm of its gradient tends to infinity near the boundary of $\mathcal{C}$: $\lim_{x\to\partial\mathcal{C}} \|\nabla F(x)\| = +\infty$. The pair $(\mathcal{C}, F)$ is then said to be a* convex function of Legendre type. *Then, the conjugate of F, F^*, is differentiable and the following holds for all $x, y \in \mathcal{C}$:*
$$\mathsf{B}_F(x \,\|\, y) = \mathsf{B}_{F^*}(\nabla F(y) \,\|\, \nabla F(x)). \tag{E.30}$$

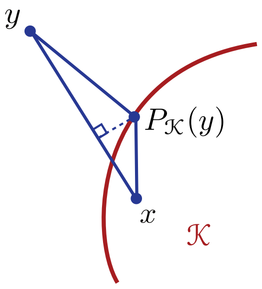


Figure E.3
A depiction of the Pythagorean theorem stated in proposition E.11, where the squared length of each line illustrates the magnitude of the Bregman divergence between the points it connects.

Proof: Property (1) holds by convexity of the function F (the graph of F is above its tangent, see equation (B.3)).

Property (2) follows directly from the definition of the Bregman divergence:

$$\begin{aligned}
&\mathsf{B}_F(x \,\|\, y) + \mathsf{B}_F(z \,\|\, x) - \mathsf{B}_F(z \,\|\, y) \\
&\quad = -\langle \nabla F(y), x - y\rangle - \langle \nabla F(x), z - x\rangle + \langle \nabla F(y), z - y\rangle \\
&\quad = \langle \nabla F(x) - \nabla F(y), x - z\rangle \ .
\end{aligned}$$

Property (3) holds since $x \mapsto F(x) - F(y) - \langle \nabla F(y), x - y\rangle$ is convex as a sum of the convex function $x \mapsto F(x)$ and the affine and thus convex function $x \mapsto -F(y) - \langle \nabla F(y), x - y\rangle$. Similarly, B_F is strictly convex with respect to its first argument if F is strictly convex, as a sum of a strictly convex function and an affine function.

Property (4) follows from a series of equalities:

$$\begin{aligned}
\mathsf{B}_{\alpha F + \beta G} &= \alpha F(x) + \beta G(x) - \alpha F(y) - \beta G(y) - \big\langle \nabla\big(\alpha F(y) + \beta G(y)\big), x - y\big\rangle \\
&= \alpha\big(F(x) - F(y) - \langle \nabla F(y), x - y\rangle\big) + \beta\big(G(x) - G(y) - \langle \nabla G(y), x - y\rangle\big) \\
&= \alpha \mathsf{B}_F + \beta \mathsf{B}_G,
\end{aligned}$$

where we have used the fact that both the gradient and inner-product are linear functions.

Property (5) holds since, by Property (3), $\min_{x \in \mathcal{K}} \mathsf{B}_F(x \,\|\, y)$ is a convex optimization problem with a strictly convex objective function.

For property (6), fix $y \in \mathcal{C}$ and let J be the function defined for all $\alpha \in [0,1]$ by

$$J(\alpha) = \mathsf{B}_F(\alpha x + (1-\alpha) P_{\mathcal{K}}(y) \,\|\, y).$$

Since $\mathcal{C}$ is convex, for any $\alpha \in [0,1]$, $\alpha x + (1-\alpha) P_{\mathcal{K}}(y)$ is in $\mathcal{C}$. F is differentiable over $\mathcal{C}$ therefore J is also differentiable as a composition of F with $\alpha \mapsto \alpha x + (1-\alpha) P_{\mathcal{K}}(y)$. By definition of $P_{\mathcal{K}}(y)$, for any $\alpha \in (0,1]$,

$$\frac{J(\alpha) - J(0)}{\alpha} = \frac{\mathsf{B}_F(\alpha x + (1-\alpha) P_{\mathcal{K}}(y) \,\|\, y) - \mathsf{B}_F(P_{\mathcal{K}}(y) \,\|\, y)}{\alpha} \geq 0. \tag{E.31}$$

This implies that $J'(0) \geq 0$. From the following expression of $J(\alpha)$:

$$J(\alpha) = F(\alpha x + (1-\alpha) P_{\mathcal{K}}(y)) - F(y) - \langle \nabla F(y), \alpha x + (1-\alpha) P_{\mathcal{K}}(y) - y\rangle\,, \tag{E.32}$$

we can compute its derivative at 0:

$$\begin{aligned}
J'(0) &= \langle x - P_{\mathcal{K}}(y), \nabla F(P_{\mathcal{K}}(y))\rangle - \langle \nabla F(y), x - P_{\mathcal{K}}(y)\rangle \\
&= -\mathsf{B}_F(x \,\|\, P_{\mathcal{K}}(y)) + F(x) - F(P_{\mathcal{K}}(y)) - \langle \nabla F(y), x - P_{\mathcal{K}}(y)\rangle \\
&= -\mathsf{B}_F(x \,\|\, P_{\mathcal{K}}(y)) + F(x) - F(P_{\mathcal{K}}(y)) - \langle \nabla F(y), x - y\rangle - \langle \nabla F(y), y - P_{\mathcal{K}}(y)\rangle \\
&= -\mathsf{B}_F(x \,\|\, P_{\mathcal{K}}(y)) + \mathsf{B}_F(x \,\|\, y) + F(y) - F(P_{\mathcal{K}}(y)) - \langle \nabla F(y), y - P_{\mathcal{K}}(y)\rangle \\
&= -\mathsf{B}_F(x \,\|\, P_{\mathcal{K}}(y)) + \mathsf{B}_F(x \,\|\, y) - \mathsf{B}_F(P_{\mathcal{K}}(y) \,\|\, y) \geq 0,
\end{aligned}$$

which concludes the proof of Property (6).

For property (7), note that, by definition, for any y, F^* is defined by

$$F^*(y) = \sup_{x \in \mathcal{C}} \big\{ \langle x, y \rangle - F(x) \big\}. \tag{E.33}$$

F^* is convex and admits a sub-differential at any y. By the strict convexity of F, the function $x \mapsto \langle x, y \rangle - F(x)$ is strictly concave and differentiable over $\mathcal{C}$ and the norm of its gradient, $y - \nabla F(x)$, tends to infinity near the boundary of $\mathcal{C}$ (by the corresponding property assumed for F). Thus, its supremum is reached at a unique point $x_y \in \mathcal{C}$ where its gradient is zero, that is at x_y with $\nabla F(x_y) = y$. This implies that for any y, $\partial F^*(y)$, the subdifferential of F^*, is reduced to a singleton. Thus, F^* is differentiable and its gradient at y is $\nabla F^*(y) = x_y = \nabla^{-1} F(y)$. Since F^* is convex and differentiable, its Bregman divergence is well defined. Furthermore, $F^*(y) = \big\langle \nabla F^{-1}(y), y \big\rangle - F(\nabla F^{-1}(y))$ since $x_y = \nabla^{-1} F(y)$. For any $x, y \in \mathcal{C}$, using the definition of B_{F^*} and the expression of $\nabla F^*(y)$ and $F^*(y)$ we can write

$$\begin{aligned}
&\mathsf{B}_{F^*}(\nabla F(y) \,\|\, \nabla F(x)) \\
&= F^*(\nabla F(y)) - F^*(\nabla F(x)) - \big\langle \nabla^{-1} F(\nabla F(x)), \nabla F(y) - \nabla F(x) \big\rangle \\
&= F^*(\nabla F(y)) - F^*(\nabla F(x)) - \langle x, \nabla F(y) - \nabla F(x) \rangle \\
&= \big\langle \nabla^{-1} F(\nabla F(y)), \nabla F(y) \big\rangle - F(\nabla^{-1} F(\nabla F(y))) \\
&\qquad - \big\langle \nabla^{-1} F(\nabla F(x)), \nabla F(x) \big\rangle + F(\nabla^{-1} F(\nabla F(x))) - \langle x, \nabla F(y) - \nabla F(x) \rangle \\
&= \langle y, \nabla F(y) \rangle - F(y) - \langle x, \nabla F(x) \rangle + F(x) - \langle x, \nabla F(y) - \nabla F(x) \rangle \\
&= \langle y, \nabla F(y) \rangle - F(y) + F(x) - \langle x, \nabla F(y) \rangle \\
&= F(x) - F(y) - \langle x - y, \nabla F(y) \rangle = \mathsf{B}_F(x \,\|\, y),
\end{aligned}$$

which completes the proof. □

Notice that while the unnormalized relative entropy (and thus the relative entropy) are convex functions of the pair of their arguments, this in general does not hold for all Bregman divergences, only convexity with respect to the first argument is guaranteed.

The notion of Bregman divergence can be extended to the case of non-differentiable functions (see section 14.3).

E.5 Chapter notes

The notion of entropy presented in this chapter is due to Shannon [1948] who, more generally, within the same article, set the foundation of information theory. More general definitions of entropy (*Rényi entropy*) and relative entropy (*Rényi divergence*) were later introduced by Rényi [1961]. The Kullback-Leibler divergence was introduced in [Kullback and Leibler, 1951].

Pinsker's inequality is due to Pinsker [1964]. Finer inequalities relating the relative entropy and the L_1-norm were later given by Csiszár [1967] and Kullback [1967]. See [Reid and Williamson, 2009] for a generalization of such inequalities to the case of f-divergences. The notion of Bregman divergence is due to Bregman [1967].

For a more extensive material on information theory, we strongly recommend the book of Cover and Thomas [2006].

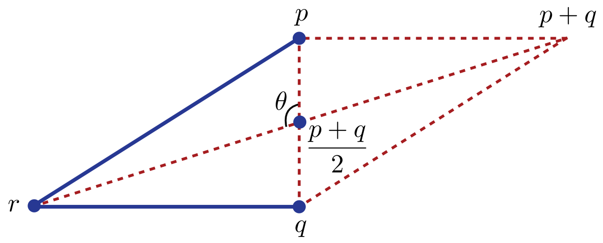


Figure E.4
An illustration of the parallelogram identity.

E.6 Exercises

E.1 Parallelogram identity. Prove the following *parallelogram identity* for any three distributions p, q, and r on $\mathcal{X}$:

$$\mathsf{D}(p \,\|\, r) + \mathsf{D}(q \,\|\, r) = 2\mathsf{D}\Big(\frac{p+q}{2} \,\|\, r\Big) + \mathsf{D}\Big(p \,\|\, \frac{p+q}{2}\Big) + \mathsf{D}\Big(q \,\|\, \frac{p+q}{2}\Big). \tag{E.34}$$

Does the equality hold if we replace the relative entropy by the norm-2 squared? Figure E.4 illustrates a particular example of this identity. Note, in the example we have

$$\begin{aligned}\|p-r\|^2 &= \big\|(p-\frac{p+q}{2}) + (\frac{p+q}{2}-r)\big\|^2\\ &= \big\|p-\frac{p+q}{2}\big\|^2 + \big\|\frac{p+q}{2}-r\big\|^2 - 2\cos(\pi-\theta)\big\|p-\frac{p+q}{2}\big\|\big\|\frac{p+q}{2}-r\big\|\end{aligned}$$

and

$$\begin{aligned}\|q-r\|^2 &= \big\|(q-\frac{p+q}{2}) + (\frac{p+q}{2}-r)\big\|^2\\ &= \big\|q-\frac{p+q}{2}\big\|^2 + \big\|\frac{p+q}{2}-r\big\|^2 - 2\cos(\theta)\big\|q-\frac{p+q}{2}\big\|\big\|\frac{p+q}{2}-r\big\|\,.\end{aligned}$$

Summing these two quantities shows the identity holds for the example.

F Notation

Table F.1
Summary of notation.

$\mathbb{R}$	Set of real numbers
$\mathbb{R}_+$	Set of non-negative real numbers
$\mathbb{R}^n$	Set of n-dimensional real-valued vectors
$\mathbb{R}^{n\times m}$	Set of $n \times m$ real-valued matrices
$[a,b]$	Closed interval between a and b
(a,b)	Open interval between a and b
$\{a,b,c\}$	Set containing elements a, b and c
$[n]$	The set $\{1,2,\ldots,n\}$
$\mathbb{N}$	Set of natural numbers, i.e., $\{0,1,\ldots\}$
$\log$	Logarithm with base e
$\log_a$	Logarithm with base a
$\mathcal{S}$	An arbitrary set
$\lvert\mathcal{S}\rvert$	Number of elements in $\mathcal{S}$
$s \in \mathcal{S}$	An element in set $\mathcal{S}$
$\mathcal{X}$	Input space
$\mathcal{Y}$	Target space
$\mathbb{H}$	Feature space
$\langle\cdot,\cdot\rangle$	Inner product in feature space
$\mathbf{v}$	An arbitrary vector
$\mathbf{1}$	Vector of all ones
v_i	ith component of $\mathbf{v}$
$\lVert\mathbf{v}\rVert$	L_2 norm of $\mathbf{v}$
$\lVert\mathbf{v}\rVert_p$	L_p norm of $\mathbf{v}$
$\mathbf{u}\circ\mathbf{v}$	Hadamard or entry-wise product of vectors $\mathbf{u}$ and $\mathbf{v}$

$f \circ g$	Composition of functions f and g
$T_1 \circ T_2$	Composition of weighted transducers T_1 and T_2
$\mathbf{M}$	An arbitrary matrix
$\|\mathbf{M}\|_2$	Spectral norm of $\mathbf{M}$
$\|\mathbf{M}\|_F$	Frobenius norm of $\mathbf{M}$
$\mathbf{M}^\top$	Transpose of $\mathbf{M}$
$\mathbf{M}^\dagger$	Pseudo-inverse of $\mathbf{M}$
$\mathrm{Tr}[\mathbf{M}]$	Trace of $\mathbf{M}$
$\mathbf{I}$	Identity matrix
$K \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$	Kernel function over $\mathcal{X}$
$\mathbf{K}$	Kernel matrix
$1_{\mathcal{A}}$	Indicator function indicating membership in subset $\mathcal{A}$
h_S	The hypothesis function returned when training with sample S
$R(\cdot)$	Generalization error or risk
$\widehat{R}_S(\cdot)$	Empirical error or risk with respect to sample S
$\widehat{R}_{S,\rho}(\cdot)$	Empirical margin error with margin ρ and with respect to sample S
$\mathfrak{R}_m(\cdot)$	Rademacher complexity over all samples of size m
$\widehat{\mathfrak{R}}_S(\cdot)$	Empirical Rademacher complexity with respect to sample S
$N(0,1)$	Standard normal distribution
$\mathop{\mathbb{E}}_{x \sim \mathcal{D}}[\cdot]$	Expectation over x drawn from distribution $\mathcal{D}$
Σ^*	Kleene closure over a set of characters Σ

Bibliography

Shivani Agarwal and Partha Niyogi. Stability and generalization of bipartite ranking algorithms. In *Conference On Learning Theory*, pages 32–47, 2005.

Shivani Agarwal, Thore Graepel, Ralf Herbrich, Sariel Har-Peled, and Dan Roth. Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research*, 6:393–425, 2005.

Nir Ailon and Mehryar Mohri. An efficient reduction of ranking to classification. In *Conference On Learning Theory*, pages 87–98, 2008.

Mark A. Aizerman, E. M. Braverman, and Lev I. Rozonoèr. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25: 821–837, 1964.

Cyril Allauzen and Mehryar Mohri. N-way composition of weighted finite-state transducers. *International Journal of Foundations of Computer Science*, 20(4):613–627, 2009.

Cyril Allauzen, Corinna Cortes, and Mehryar Mohri. Large-scale training of SVMs with automata kernels. In *International Conference on Implementation and Application of Automata*, pages 17–27, 2010.

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley, 1992.

Noga Alon, Shai Ben-David, Nicolò Cesa-Bianchi, and David Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of ACM*, 44:615–631, July 1997.

Yasemin Altun and Alexander J. Smola. Unifying divergence minimization and statistical inference via convex duality. In *Conference On Learning Theory*, pages 139–153, 2006.

Galen Andrew and Jianfeng Gao. Scalable training of l_1-regularized log-linear models. In *Proceedings of ICML*, pages 33–40, 2007.

Dana Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39(3):337–350, 1978.

Dana Angluin. Inference of reversible languages. *Journal of the ACM*, 29(3):741–765, 1982.

Dana Angluin and Leslie G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. *J. Comput. Syst. Sci.*, 18(2):155–193, 1979.

Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

Patrick Assouad. Densité et dimension. *Annales de l'institut Fourier*, 33(3):233–282, 1983.

Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 19(3):357–367, 1967.

Maria-Florina Balcan, Nikhil Bansal, Alina Beygelzimer, Don Coppersmith, John Langford, and Gregory B. Sorkin. Robust reductions from ranking to classification. *Machine Learning*, 72(1-2): 139–153, 2008.

Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3, 2002.

Peter L. Bartlett, Stéphane Boucheron, and Gábor Lugosi. Model selection and error estimation. *Machine Learning*, 48:85–113, September 2002a.

Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Localized Rademacher complexities. In *Conference on Computational Learning Theory*, volume 2375, pages 79–97. Springer-Verlag, 2002b.

Amos Beimel, Francesco Bergadano, Nader H. Bshouty, Eyal Kushilevitz, and Stefano Varricchio. Learning functions represented as multiplicity automata. *Journal of the ACM*, 47:2000, 2000.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Neural Information Processing Systems*, 2001.

George Bennett. Probability inequalities for the sum of independent random variables. *Journal of the American Statistical Association*, 57:33–45, 1962.

Christian Berg, Jens P.R. Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*, volume 100. Springer, 1984.

Francesco Bergadano and Stefano Varricchio. Learning behaviors of automata from shortest counterexamples. In *European Conference on Computational Learning Theory*, pages 380–391, 1995.

Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Comp. Linguistics*, 22(1), 1996.

Joseph Berkson. Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, 39:357—-365, 1944.

Sergei Natanovich Bernstein. Sur l'extension du théorème limite du calcul des probabilités aux sommes de quantités dépendantes. *Mathematische Annalen*, 97:1–59, 1927.

Dimitri P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, 1987.

Dmitri P. Bertsekas, Angelica Nedić, and Asuman E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.

Laurence Bisht, Nader H. Bshouty, and Hanna Mazzawi. On optimal learning algorithms for multiplicity automata. In *Conference On Learning Theory*, pages 184–198, 2006.

Avrim Blum and Yishay Mansour. From external to internal regret. In *Conference On Learning Theory*, pages 621–636, 2005.

Avrim Blum and Yishay Mansour. Learning, regret minimization, and equilibria. In Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay Vazirani, editors, *Algorithmic Game Theory*, chapter 4, pages 4–30. Cambridge University Press, 2007.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

Jonathan Borwein and Qiji Zhu. *Techniques of Variational Analysis*. Springer, New York, 2005.

Jonathan M. Borwein and Adrian S. Lewis. *Convex Analysis and Nonlinear Optimization, Theory and Examples*. Springer, 2000.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Conference On Learning Theory*, pages 144–152, 1992.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

Lev M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

Leo Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11:1493–1517, October 1999.

Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

Nicolò Cesa-Bianchi. Analysis of two gradient-based algorithms for on-line regression. *Journal of Computer System Sciences*, 59(3):392–411, 1999.

Nicolò Cesa-Bianchi and Gábor Lugosi. Potential-based algorithms in online prediction and game theory. In *Conference On Learning Theory*, pages 48–64, 2001.

Nicolò Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, 2006.

Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, 1997.

Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. In *Neural Information Processing Systems*, pages 359–366, 2001.

Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

Nicolò Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. In *Conference On Learning Theory*, pages 217–232, 2005.

Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. Pre-reduction graph products: Hardnesses of properly learning dfas and approximating edp on dags. In *Symposium on Foundations of Computer Science*, pages 444–453. IEEE, 2014.

Bernard Chazelle. *The Discrepancy Method: Randomness and Complexity.* Cambridge University Press, New York, NY, USA, 2000.

Stanley F. Chen and Ronald Rosenfeld. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1), 2000.

Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 12 1952.

Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, Adaboost and Bregman distances. *Machine Learning*, 48:253–285, September 2002.

Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In *Neural Information Processing Systems*, 2003.

Corinna Cortes and Mehryar Mohri. Confidence intervals for the area under the ROC curve. In *Neural Information Processing Systems*, volume 17, Vancouver, Canada, 2005. MIT Press.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research*, 5:1035–1062, 2004.

Corinna Cortes, Leonid Kontorovich, and Mehryar Mohri. Learning languages with rational kernels. In *Conference On Learning Theory*, volume 4539 of *Lecture Notes in Computer Science*, pages 349–364. Springer, Heidelberg, Germany, June 2007a.

Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. An alternative ranking problem for search engines. In *Workshop on Experimental Algorithms*, pages 1–22, 2007b.

Corinna Cortes, Mehryar Mohri, and Jason Weston. A general regression framework for learning string-to-string mappings. In *Predicted Structured Data*. MIT Press, 2007c.

Corinna Cortes, Mehryar Mohri, Dmitry Pechyony, and Ashish Rastogi. Stability of transductive regression algorithms. In *International Conference on Machine Learning*, Helsinki, Finland, July 2008a.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Learning sequence kernels. In *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing*, Cancún, Mexico, October 2008b.

Corinna Cortes, Yishay Mansour, and Mehryar Mohri. Learning bounds for importance weighting. In *Neural Information Processing Systems*, Vancouver, Canada, 2010a. MIT Press.

Corinna Cortes, Mehryar Mohri, and Ameet Talwalkar. On the impact of kernel approximation on learning accuracy. In *Conference on Artificial Intelligence and Statistics*, 2010b.

Corinna Cortes, Spencer Greenberg, and Mehryar Mohri. Relative deviation learning bounds and generalization with unbounded loss functions. *ArXiv 1310.5796*, October 2013. URL `http://arxiv.org/pdf/1310.5796v4.pdf`.

Corinna Cortes, Mehryar Mohri, and Umar Syed. Deep boosting. In *International Conference on Machine Learning*, pages 1179–1187, 2014.

Corinna Cortes, Vitaly Kuznetsov, Mehryar Mohri, and Umar Syed. Structural Maxent models. In *International Conference on Machine Learning*, pages 391–399, 2015.

David Cossock and Tong Zhang. Statistical analysis of Bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54(11):5140–5154, 2008.

Thomas M. Cover and Joy M. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2006.

Trevor F. Cox and Michael A. A. Cox. *Multidimensional Scaling*. Chapman & Hall/CRC, 2nd edition, 2000.

Koby Crammer and Yoram Singer. Improved output coding for classification using continuous relaxation. In *Neural Information Processing Systems*, 2001.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2, 2002.

Robert Crites and Andrew Barto. Improving elevator performance using reinforcement learning. In *Neural Information Processing Systems*, pages 1017–1023. MIT Press, 1996.

Imre Csiszár. Information-type measures of difference of probability distributions and indirect observations. *Studia Scientiarum Mathematicarum Hungarica*, 2:299–318, 1967.

Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39(1):1–49, 2001.

J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, pages 1470–1480, 1972.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures and Algorithms*, 22(1):60–65, 2003.

Colin De la Higuera. *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010.

Giulia DeSalvo, Mehryar Mohri, and Umar Syed. Learning with deep cascades. In *Conference on Algorithmic Learning Theory*, pages 254–269, 2015.

Luc Devroye and Gábor Lugosi. Lower bounds in pattern recognition and learning. *Pattern Recognition*, 28(7):1011–1018, 1995.

Luc Devroye and T. J. Wagner. Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Transactions on Information Theory*, 25(2):202–207, 1979a.

Luc Devroye and T. J. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604, 1979b.

Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

Harris Drucker and Corinna Cortes. Boosting decision trees. In *Neural Information Processing Systems*, pages 479–485, 1995.

Harris Drucker, Robert E. Schapire, and Patrice Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):705–719, 1993.

Miroslav Dudík, Steven J. Phillips, and Robert E. Schapire. Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, 8, 2007.

Richard M. Dudley. The sizes of compact subsets of Hilbert space and continuity of Gaussian processes. *Journal of Functional Analysis*, 1(3):290–330, 1967.

Richard M. Dudley. A course on empirical processes. *Lecture Notes in Mathematics*, 1097:2 – 142, 1984.

Richard M. Dudley. Universal Donsker classes and metric entropy. *Annals of Probability*, 14(4): 1306–1326, 1987.

Richard M. Dudley. *Uniform Central Limit Theorems.* Cambridge University Press, 1999.

Nigel Duffy and David P. Helmbold. Potential boosters? In *Neural Information Processing Systems*, pages 258–264, 1999.

Aryeh Dvoretzky. On stochastic approximation. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*, pages 39–55, 1956.

Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *International World Wide Web Conference*, pages 613–622, 2001.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

James P. Egan. *Signal Detection Theory and ROC Analysis.* Academic Press, 1975.

Andrzej Ehrenfeucht, David Haussler, Michael J. Kearns, and Leslie G. Valiant. A general lower bound on the number of examples needed for learning. In *Conference On Learning Theory*, pages 139–154, 1988.

Jane Elith, Steven J. Phillips, Trevor Hastie, Miroslav Dudík, Yung En Chee, and Colin J. Yates. A statistical explanation of MaxEnt for ecologists. *Diversity and Distributions*, 1, 2011.

Eyal Even-Dar and Yishay Mansour. Learning rates for q-learning. *Machine Learning*, 5:1–25, 2003.

Dean P. Foster and Rakesh V. Vohra. Calibrated learning and correlated equilibrium. *Games and Economic Behavior*, 21:40–55, 1997.

Dean P. Foster and Rakesh V. Vohra. Asymptotic calibration. *Biometrika*, pages 379–390, 1998.

Dean P. Foster and Rakesh V. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1-2):7–35, 1999.

Yoav Freund. Boosting a weak learning algorithm by majority. In *Information and Computation*, pages 202–216. Morgan Kaufmann Publishers Inc., 1990.

Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121:256–285, September 1995.

Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *Conference On Learning Theory*, pages 325–332, 1996.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer System Sciences*, 55(1):119–139, 1997.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–296, 1999a.

Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, October 1999b.

Yoav Freund, Michael J. Kearns, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. Efficient learning of typical finite automata from random walks. In *Proceedings the ACM Symposium on Theory of Computing*, pages 315–324, 1993.

Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4, 2003.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

Jerome H. Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 38(2), 2000.

E. Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

E. Mark Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.

Joshua Goodman. Exponential priors for maximum entropy models. In *Proceedings of HLT-NAACL*, pages 305–312, 2004.

David M. Green and John A Swets. *Signal Detection Theory and Psychophysics*. Wiley, 1966.

Michelangelo Grigni, Vincent Mirelli, and Christos H Papadimitriou. On the difficulty of designing good classifiers. *SIAM Journal on Computing*, 30(1):318–323, 2000.

Adam J. Grove and Dale Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 692–699, 1998.

Uffe Haagerup. The best constants in the Khintchine inequality. *Studia Math*, 70(3):231–283, 1982.

Torben Hagerup and Christine Rüb. A guided tour of chernoff bounds. *Information Processing Letters*, 33(6):305–308, 1990.

Jihun Ham, Daniel D. Lee, Sebastian Mika, and Bernhard Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *International Conference on Machine Learning*, 2004.

James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982.

James Hannan. Approximation to Bayes risk in repeated plays. *Contributions to the Theory of Games*, 3:97–139, 1957.

Sergiu Hart and Andreu M. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

David Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

David Haussler. Sphere packing numbers for subsets of the boolean n-cube with bounded Vapnik-Chervonenkis dimension. *Journal of Combinatorial Theory, Series A*, 69(2):217 – 232, 1995.

David Haussler. Convolution Kernels on Discrete Structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, 1999.

David Haussler, Nick Littlestone, and Manfred K. Warmuth. Predicting {0,1}-functions on randomly drawn points (extended abstract). In *Symposium on Foundations of Computer Science*, pages 100–109, 1988.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, Cambridge, MA, 2000.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Klaus-Uwe Höffgen, Hans-Ulrich Simon, and Kevin S. Van Horn. Robust trainability of single neurons. *Journal of Computer and Systems Sciences*, 50(1):114–125, 1995.

John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *International Conference on Machine Learning*, pages 408–415, 2008.

Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. Convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201, 1994.

Kalervo Järvelin and Jaana Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *ACM Special Interest Group on Information Retrieval*, pages 41–48, 2000.

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, 1957.

E. T. Jaynes. *Papers on probability, statistics, and statistical physics*. Synthese library. D. Reidel Pub. Co., 1983.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Knowledge and Discovery and Data Mining*, pages 133–142, 2002.

William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189—206, 1984.

Jean-Pierre Kahane. Sur les sommes vectorielles $\sum \pm u_n$. *Comptes Rendus Hebdomadaires des S'eances de l'Académie des Sciences, Paris*, 259:2577–2580, 1964.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. In *Conference On Learning Theory*, pages 26–40, 2003.

William Karush. *Minima of Functions of Several Variables with Inequalities as Side Constraints*. Master's thesis, Department of Mathematics, University of Chicago, 1939.

Jun'ichi Kazama and Jun'ichi Tsujii. Evaluation and extension of maximum entropy models with inequality constraints. In *Proceedings of EMNLP*, pages 137–144, 2003.

Michael J. Kearns and Yishay Mansour. A fast, bottom-up decision tree pruning algorithm with near-optimal generalization. In *International Conference on Machine Learning*, pages 269–277, 1998.

Michael J. Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. *Journal of Computer and System Sciences*, 58(1):109–128, 1999.

Michael J. Kearns and Dana Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11(6):1427–1453, 1999.

Michael J. Kearns and Robert E. Schapire. Efficient distribution-free learning of probabilistic concepts (extended abstract). In *Symposium on Foundations of Computer Science*, pages 382–391, 1990.

Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. Technical Report 14, Harvard University, 1988.

Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of ACM*, 41(1):67–95, 1994.

Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

Aleksandr Khintchine. Über dyadische brüche. *Mathematische Zeitschrift*, 18(1):109–116, 1923.

Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23(1):462–466, 1952.

George Kimeldorf and Grace Wahba. Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971.

Jyrki Kivinen and Manfred K. Warmuth. Boosting as entropy projection. In *Conference On Learning Theory*, pages 134–144, 1999.

Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, 2001.

Vladimir Koltchinskii and Dmitry Panchenko. Rademacher processes and bounding the risk of function learning. In *High Dimensional Probability II*, pages 443–459. Birkhäuser, 2000.

Vladmir Koltchinskii and Dmitry Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30, 2002.

Leonid Kontorovich, Corinna Cortes, and Mehryar Mohri. Learning linearly separable languages. In *Algorithmic Learning Theory*, pages 288–303, 2006.

Leonid Kontorovich, Corinna Cortes, and Mehryar Mohri. Kernel methods for learning languages. *Theoretical Computer Science*, 405:223–236, 2008.

Harold W. Kuhn and Albert W. Tucker. Nonlinear programming. In *2nd Berkeley Symposium*, pages 481–492, Berkeley, 1951. University of California Press.

Solomon Kullback. A lower bound for discrimination information in terms of variation. *IEEE Transactions on Information Theory*, 13(1):126–127, 1967.

Solomon Kullback and Richard A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

Harold Kushner. Stochastic approximation: a survey. *Wiley Interdisciplinary Reviews Computational Statistics*, 2(1):87–96, 2010.

Harold J. Kushner and D. S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, volume 26 of *Applied Mathematical Sciences*. Springer-Verlag, 1978.

Vitaly Kuznetsov, Mehryar Mohri, and Umar Syed. Multi-class deep boosting. In *Neural Information Processing Systems*, 2014.

John Lafferty. Additive models, boosting, and inference for generalized divergences. In *Conference On Learning Theory*, pages 125–133, 1999.

John D. Lafferty, Stephen Della Pietra, and Vincent J. Della Pietra. Statistical learning algorithms based on bregman distances. In *Proceedings of the Canadian Workshop on Information Theory*, 1997.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pages 282–289, 2001.

Rafał Latała and Krzysztof Oleszkiewicz. On the best constant in the khintchine-kahane inequality. *Studia Math*, 109(1):101–104, 1994.

Guy Lebanon and John D. Lafferty. Boosting and maximum likelihood for exponential models. In *Neural Information Processing Systems*, pages 447–454, 2001.

Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes.* Springer, New York, 1991.

Ehud Lehrer. A wide range no-regret theorem. *Games and Economic Behavior*, 42(1):101–115, 2003.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1987.

Nick Littlestone. From on-line to batch learning. In *Conference On Learning Theory*, pages 269–284, 1989.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. In *Symposium on Foundations of Computer Science*, pages 256–261, 1989.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

Michael L. Littman. *Algorithms for Sequential Decision Making.* PhD thesis, Brown University, 1996.

Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. *Machine Learning*, 78:287–304, March 2010.

M. Lothaire. *Combinatorics on Words.* Cambridge University Press, 1982.

M. Lothaire. *Mots.* Hermès, 1990.

M. Lothaire. *Applied Combinatorics on Words.* Cambridge University Press, 2005.

Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL-2002*, pages 49–55, 2002.

Christopher D. Manning and Dan Klein. Optimization, maxent models, and conditional estimation without magic. In *Proceedings of HLT-NAACL*, 2003.

Yishay Mansour and David A. McAllester. Boosting with multi-way branching in decision trees. In *Neural Information Processing Systems*, pages 300–306, 1999.

Yishay Mansour and David A. McAllester. Generalization bounds for decision trees. In *Conference On Learning Theory*, pages 69–74, 2000.

Llew Mason, Jonathan Baxter, Peter L. Bartlett, and Marcus R. Frean. Boosting algorithms as gradient descent. In *Neural Information Processing Systems*, pages 512–518, 1999.

Pascal Massart. Some applications of concentration inequalities to statistics. *Annales de la Faculté des Sciences de Toulouse*, IX:245–303, 2000.

Peter McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society B*, 42(2), 1980.

Peter McCullagh and John A. Nelder. *Generalized Linear Models.* Chapman & Hall, 1983.

Colin McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, 141(1): 148–188, 1989.

Ron Meir and Gunnar Rätsch. Advanced lectures on machine learning, machine learning summer school, canberra, australia. In *Machine Learning Summer School*, pages 118–183, 2002.

Ron Meir and Gunnar Rätsch. *An Introduction to Boosting and Leveraging*, pages 118–183. Springer, 2003.

James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209(441-458):415, 1909.

Sebastian Mika, Bernhard Scholkopf, Alex J. Smola, Klaus-Robert Muller, Matthias Scholz, and Gunnar Ratsch. Kernel PCA and de-noising in feature spaces. In *Neural Information Processing Systems*, pages 536–542, 1999.

Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969.

Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

Mehryar Mohri. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, pages 213–254. Springer, 2009.

Mehryar Mohri and Afshin Rostamizadeh. Stability bounds for stationary φ-mixing and β-mixing processes. *Journal of Machine Learning Research*, 11:789–814, 2010.

Mehryar Mohri and Afshin Rostamizadeh. Perceptron mistake bounds. *ArXiv 1305.0208*, March 2013.

Mehryar Mohri, Fernando Pereira, and Michael D. Riley. Weighted automata in text and speech processing. *European Conference on Artificial Intelligence, Workshop on Extended Finite State Models of Language*, 2005.

Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

Albert B.J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, 1962.

José Oncina, Pedro García, and Enrique Vidal. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):448–458, 1993.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

Fernando C. N. Pereira and Michael D. Riley. Speech recognition by composition of weighted finite automata. In *Finite-State Language Processing*, pages 431–453. MIT Press, 1997.

Dominique Perrin. Finite automata. In J. Van Leuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 1–57. Elsevier, 1990.

Steven J. Phillips, Miroslav Dudík, and Robert E. Schapire. A maximum entropy approach to species distribution modeling. In *Proceedings of ICML*, 2004.

Steven J. Phillips, Robet P. Anderson, and Robert E. Schapire. Maximum entropy modeling of species geographic distributions. *Ecological Modelling*, 190:231–259, 2006.

Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. Inducing features of random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(4), 1997.

Mark Semenovich Pinsker. *Information and Information Stability of Random Variables and Processes.* Holden-Day, 1964.

Leonard Pitt and Manfred K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM*, 40(1):95–142, 1993.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods*, pages 185–208. MIT Press, 1999.

David Pollard. *Convergence of Stochastic Processess*. Springer, 1984.

David Pollard. Asymptotics via empirical processes. *Statistical Science*, 4(4):341 – 366, 1989.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Neural Information Processing Systems*, pages 1177–1184, 2007.

Adwait Ratnaparkhi. Maximum entropy models for natural language processing. In *Encyclopedia of Machine Learning*, pages 647–651. Springer, 2010.

Gunnar Rätsch and Manfred K. Warmuth. Maximizing the margin with boosting. In *Conference On Learning Theory*, pages 334–350, 2002.

Gunnar Rätsch, Sebastian Mika, and Manfred K. Warmuth. On the convergence of leveraging. In *NIPS*, pages 487–494, 2001.

Gunnar Rätsch, Takashi Onoda, and Klaus-Robert Müller. Soft margins for AdaBoost. *Machine Learning*, 42:287–320, March 2001.

Mark D. Reid and Robert C. Williamson. Generalised pinsker inequalities. In *22nd Conference on Learning Theory (COLT 2009)*, 2009.

Alfréd Rényi. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 547–561. University of California Press, 1961.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

Ryan M. Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, Massachusetts Institute of Technology, 2002.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

R. Tyrrell Rockafellar. *Convex analysis*. Princeton University Press, 1997.

W.H. Rogers and T. J. Wagner. A finite sample distribution-free performance bound for local discrimination rules. *Annals of Statistics*, 6(3):506–514, 1978.

Dana Ron, Yoram Singer, and Naftali Tishby. On the learnability and usage of acyclic probabilistic finite automata. In *Journal of Computer and System Sciences*, pages 31–40, 1995.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech & Language*, 10(3):187–228, 1996.

Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323, 2000.

Cynthia Rudin, Ingrid Daubechies, and Robert E. Schapire. The dynamics of AdaBoost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research*, 5:1557–1595, 2004.

Cynthia Rudin, Corinna Cortes, Mehryar Mohri, and Robert E. Schapire. Margin-based ranking meets boosting in the middle. In *Conference On Learning Theory*, 2005.

Walter Rudin. *Fourier analysis on groups.* Number 12 in Interscience tracts in pure and applied mathematics. John Wiley & Sons, 1990.

I. N. Sanov. On the probability of large deviations of random variables. *Matematicheskii Sbornik*, 42(84):11–44, 1957.

Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13 (1):145–147, 1972.

Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *International Conference on Machine Learning*, volume 521, 1998.

Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, July 1990.

Robert E. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*, pages 149–172. Springer, 2003.

Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms.* The MIT Press, 2012.

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168, 2000.

Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *International Conference on Machine Learning*, pages 322–330, 1997.

Leopold Schmetterer. Stochastic approximation. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, pages 587–609, 1960.

Isaac J. Schoenberg. Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536, 1938.

Bernhard Schölkopf and Alex Smola. *Learning with Kernels.* MIT Press, 2002.

Bernhard Schölkopf, Ralf Herbrich, Alex J. Smola, and Robert Williamson. A generalized representer theorem. Technical Report 2000-81, Neuro-COLT, 2000.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability and stability in the general learning setting. In *Conference On Learning Theory*, 2009.

Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.

Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1), 1972.

Satinder P. Singh. *Learning to Solve Markovian Decision Processes.* PhD thesis, University of Massachusetts, 1993.

Satinder P. Singh and Dimitri Bertsekas. Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *Neural Information Processing Systems*, pages 974–980. MIT Press, 1997.

Maurice Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8(1):171–176, 1958.

Eric V. Slud. Distribution inequalities for the binomial law. *Annals of Probability*, 5(3):404–412, 1977.

Bharath Sriperumbudur and Zoltán Szabó. Optimal rates for random fourier features. In *Neural Information Processing Systems*, pages 1144–1152, 2015.

Gilles Stoltz and Gábor Lugosi. Internal regret in on-line portfolio selection. In *Conference On Learning Theory*, pages 403–417, 2003.

Rich Sutton. *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts, 1984.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning : An Introduction.* MIT Press, 1998.

S.J. Szarek. On the best constants in the Khintchin inequality. *Studia Math*, 58(2):197–208, 1976.

Csaba Szepesvári. *Algorithms for Reinforcement Learning.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2010.

Eiji Takimoto and Manfred K. Warmuth. Path kernels and multiplicative updates. In *Conference On Learning Theory*, pages 74–89, 2002.

Benjamin Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Neural Information Processing Systems*, 2003.

Robert F. Tate. On a double inequality of the normal distribution. *The Annals of Mathematical Statistics*, 1:132–134, 1953.

Joshua Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Gerald Tesauro. Temporal difference learning and TD-gammon. *Communications of the ACM*, 38:58–68, March 1995.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, 58(1):267–288, 1996.

B. Tomaszewski. Two remarks on the Khintchine-Kahane inequality. In *Colloquium Mathematicum*, volume 46, 1982.

Boris Trakhtenbrot and Janis M. Barzdin. *Finite Automata: Behavior and Synthesis.* North-Holland, 1973.

John N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. In *Machine Learning*, volume 16, pages 185–202, 1994.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1998.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 2000.

Vladimir N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, 2006.

Vladimir N. Vapnik and Alexey Chervonenkis. A note on one class of perceptrons. *Automation and Remote Control*, 25, 1964.

Vladimir N. Vapnik and Alexey Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16:264, 1971.

Vladimir N. Vapnik and Alexey Chervonenkis. *Theory of Pattern Recognition.* Nauka, 1974.

Santosh S. Vempala. The random projection method. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 65. American Mathematical Society, 2004.

Pierre François Verhulst. Notice sur la loi que la population suit dans son accroissement. *Correspondance mathématique et physique*, 10:113—-121, 1838.

Pierre François Verhulst. Recherches mathématiques sur la loi d'accroissement de la population. *Nouveaux Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles*, 18:1—-42, 1845.

Mathukumalli Vidyasagar. *A Theory of Learning and Generalization: With Applications to Neural Networks and Control Systems.* Springer-Verlag, 1997.

Sethu Vijayakumar and Si Wu. Sequential support vector classifiers and regression. *International Conference on Soft Computing*, 1999.

John von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

Vladimir G. Vovk. Aggregating strategies. In *Conference On Learning Theory*, pages 371–386, 1990.

Grace Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics.* Society for Industrial and Applied Mathematics, 1990.

Christopher J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, Cambridge University, 1989.

Christopher J. C. H. Watkins. Dynamic alignment kernels. Technical Report CSD-TR-98-11, Royal Holloway, University of London, 1999.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

André Weil. *L'intégration dans les groupes topologiques et ses applications*, volume 1145. Hermann Paris, 1965.

Kilian Q. Weinberger and Lawrence K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *Conference on Artificial Intelligence*, 2006.

Jason Weston and Chris Watkins. Support vector machines for multi-class pattern recognition. *European Symposium on Artificial Neural Networks*, 4(6), 1999.

Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. *Neurocomputing: Foundations of Research*, 1988.

Peter M. Williams. Bayesian regularisation and pruning using a Laplace prior. *Neural Computation*, 7:117–143, 1994.

Huan Xu, Shie Mannor, and Constantine Caramanis. Sparse algorithms are not stable: A no-free-lunch theorem. In *Conference on Communication, Control, and Computing*, pages 1299–1303, 2008.

Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.

Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32:56–134, 2003a.

Tong Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Trans. Inf. Theor.*, 49(3):682–691, 2003b.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning*, pages 928–936, 2003.

Index

L_1-geometric margin, *see* margin
L_1-margin, *see* margin
L_1-regularized
  AdaBoost, 165
  logistic regression, 325
β-contracting, 388
β-stable, 334, 338, 340–342
  uniformly, 334
ϵ-greedy policy, 401
ϵ-insensitive loss, 282
ϵ-transition, 361
γ-fat-dimension, 274, *see* fat-shattering dimension
γ-shattered, 274, *see* fat-shattered
log-linear model, 321, 326
ρ-margin loss function, *see* margin
σ-admissible, 337, 338, 340–342
σ-algebra, 429
k-CNF formula, 18, 19
k-deterministic, 377
k-reversible, 377
k-term DNF formula, 18
n-fold cross-validation, 71
n-way composition, 128, 136
pth-moment, 434

absolutely continuous, 429
accepted, 28, 361
accepting path, 123, 361
access string, 364–368
accuracy, 8, 11, 17, 23, 46, 148–150, 154, 167, 169, 172, 244, 283
  pairwise ranking, 245, 255, 256
action, 7, 163, 164, 183, 205, 240, 379–383, 387–390, 393, 398, 399, 401–404
  greedy, 401, 402
  policy, *see* policy
  random, 401
active learning, 7, 362
acyclic, 361
AdaBoost, 145
  L_1-regularized, 165
AdaBoost's weak learning condition, 162
AdaBoost.MH, 222, 223, 236, 237
AdaBoost.MR, 222, 236, 238
adaptive boosting, 150
adversarial, 177, 178, 180, 204, 260
  argument, 180
  assumption, 178
  choice, 260
  scenario, 177
advice, 178
affine, 421
agent, 379
aggregated algorithms, 213, 221
algebraic transductions, 127
algorithm
  dependent, 333
  deterministic, 183, 258–260, 264
  learning, 1, 4–6, 9, 19, 20, 23, 24, 27, 43, 46, 47, 55, 57, 71, 80, 85, 98, 146, 148, 150, 168, 173, 179, 186, 202, 252, 257, 333, 334, 343, 362
  off-policy, 401, 402
  on-policy, 402
  randomized, 186, 209, 239, 259, 260, 264
algorithmic stability, *see* stability

approximately correct, 11
approximation error, 61–64
area under the curve, *see* AUC
AUC, 239, 255, 256, 264, 265
automaton
  k-reversible, 377
  deterministic, 360, *see also* DFA, 361, 362
  finite, 125, 129, 130, 360, 361, 370, 375, 377
  learning, 359
  prefix-tree, 370, 371, 373
  reverse deterministic, 374
  reversible, 370, 371, 373, 374
average
  noise, 23
  precision, 263
  regret, 186
Azuma's inequality, 202, 442, 445

base
  classifier set, 146
  classifiers, 146
  rankers, 244
Bayes
  classifier, 22, 47, 61, 74, 75, 78, 140
  error, 22, 23, 28, 61, 67, 259
  formula, 431
  hypothesis, 22
  scoring function, 74
Bellman
  equations, 385–387, 389, 390, 392
Bennett's inequality, 447
Bernstein's inequality, 438, 440, 447
bias, 46, 71, 296, 446, 450
bigram, 128, 129
  gappy, 128, 129
  kernel, 128, 129
binary
  classification, 9
  classifier, 79
  decision tree, 224
  entropy, 450
  entropy function, 449
  space partition (BSP) trees, 225
binomial distribution, 430, 440, 448
bipartite ranking, 251
Boltzmann exploration, 401
boosted, 168
boosting, 145–149, 152, 154, 155, 159, 160, 163, 165, 167–172, 174, 175, 221–224, 236, 237, 239, 244–246, 251, 291, 298, 320, 330
  by filtering, 168
  by majority, 168
  multi-class, 213, *see also* AdaBoost.MH, *see also* AdaBoost.MR, 237
  round, 147, 148, 171
  trees, 291
Bregman divergence, 169, 295, 307, 313, 331, 337, 453–456
  generalized, 337, 338

calibration problem, 229, 230
categorical question, 224
Cauchy-Schwarz inequality, 53, 97, 98, 112, 118, 197, 309, 310, 339, 341, 409, 410, 433
chain rule, 431
Chebyshev's inequality, 433, 447
Chernoff
  bound, 28, 45
  bounding technique, 437, 438
  multiplicative bounds, 439, 445
chi-squared distribution, 430
Cholesky decomposition, 115, 413
  partial, 280, 285
classification, 3, 259
  binary, 4, 33, 34, 61, 74, 79, 102, 159, 173, 213, 228–231, 239, 244, 252, 257, 264, 271, 281, 325, 330, 331
  document, 2, 3, 106, 215
  image, 3, 140
  linear, 79, 105, 177, 190, 198
  multi-class, xiii, 168, 213–215, 217, 219–222, 224, 225, 228, 229, 232, 233, 235–237, 264, 315, 321, 331
  text, 3
classifier, 38, 67, 146–149, 153, 159, 164, 169, 172, 173, 214, 223, 230–232, 239, 255, 273
  base, 146–148, 151, 160, 162, 163, 167, 169, 170, 175, 224

binary, 74, 229–232, 259, 265
linear, 80
clique, 234
closed, 422
clustering, 3, 117, 224
co-accessible, 125
code
binary, 232
word, 231, 232
complementarity conditions, 83, 90, 282, 421
complete metric space, 388
composition, 123
concave, 84, 91, 118, 226, 278, 304, 416–418, 420, 449, 451, 452, 456
function, 84, 118, 416, 449, 452
problem, 420
concentration inequality, xiv, 437, 445
concept, 1, 3, 9–19, 24–27, 29, 36, 41, 54–57, 145, 179, 180, 347, 348, 361–364, 369, 377
class, 10–12, 14, 16–19, 24–27, 29, 54, 55, 57, 145, 179, 362, 363, 377
class universal, 17
conditional
maximum entropy models, 315
probability, 431
relative entropy, 316, 319, 331, 452
Conditional Random Fields, *see* CRFs
confidence, 17, 27, 28, 57, 92, 93, 95, 97, 148, 155, 157, 159, 215, 229, 231, 232, 241, 264, 315
margin, 92, *see* margin, 95, 97, 155, 157, 159
conjugate, 157, 201, 300, 304, 307, 308, 313, 328, 329, 410, 423, 424, 426, 427, 454
function, 300, 304, 307, 308, 328, 329, 423, 424, 427
consistent, 7–9, 15–17, 19–21, 24, 27, 55, 57, 163, 172, 179, 362–364, 375
algorithm, 16, 17
case, 20, 21
DFA, 362, 363, 375
hypothesis, 15–17, 172
pairwise, 257
constrained optimization problem, 420
constraint
equality, 102, 420
qualification, 420
qualification strong, 421
qualification weak, 421
context-free grammars, 363
convex, 415, 417
combination, 157–159, 222
differentiable functions, 421
function, 73, 151, 152, 165, 235, 237, 255, 275, 307, 337, 415–419, 422, 423, 455
function of Legendre type, 454
functions, 415, 421
hull, 37, 38, 157, 191, 250, 416
loss, 77, 153, 250, 284
optimization, xiii, xiv, 75, 82, 84, 88, 99, 110, 151, 162, 166, 221, 278, 286, 299, 304, 307, 317, 320, 323, 325, 331, 415, 420, 422, 426, 455
set, 299, 300, 317, 318, 416–419, 454
sets, 415
strictly, 82, 299, 307, 317, 417, 454, 455
convexity, 47, 75, 77, 88, 107, 188, 191, 203, 210, 211, 237, 277, 310, 416, 418–420, 422, 423, 437, 439, 443, 444, 451, 455, 456
core, 423
covariance, 348, 349, 353, 356, 433
matrix, 348, 349, 353, 356, 434
covering, 49, 58, 59, 133, 134, 263
number, 49, 58, 134, 263
CRFs, 235, 237
cross-validation, 68–73, 99, 102, 103, 166, 167, 285, 323, 324
n-fold, 70–73, 88, 228
error, 71, 102

data
set, 102, 109, 139, 161, 255, 347, 348, 357, 379
test, 5, 6
training, 5, 6, 21, 24, 25, 71, 72, 87, 88, 93, 101, 102, 170, 176, 228, 229
unseen, 8

validation, 4
DCG, 263, 264
normalized, 263
decision
epochs, 381
stump, 154
tree, 154, 155, 168, 169, 213, 221, 224, 225, 227, 228, 236, 291, 298, 365, 366, 368, 376
tree binary, 238, 365
DeepBoost, 169
degrees of freedom, 430
deterministic, 22, 361, 381
determinization, 361
DFA, 361, 362, 364–366, 368–370, 375–377
consistent, 362, 363, 375
equivalent, 361
minimal, 361, 362, 364, 376
dichotomy, 34, 36, 37, 50
differentiable
function, 170, 337, 415, 417, 453, 454
dimensionality reduction, 3, 6, 117, 347, 348, 351, 354, 356
discounted cumulative gain, *see* DCG
distinguishing strings, 364
distribution, 430
-free model, 11
binomial, 430, 440, 448
chi-squared, 430
Gaussian, 303
Gibbs, 295, 299, 300, 306, 312, 317, 430
Laplace, 430
normal, 330, 355, 358, 430, 434, 443, 460
Poisson, 430
probability, 22, 131, 140, 397, 403, 429, 431, 453
divergence
Bregman, 169, 295, 307, 313, 331, 337, 453–456
Bregman generalized, 337, 338
Kullback-Leibler, 344, 450, 456
Rényi, 456
DNF formula, 17
doubling trick, 185, 189, 204, 205
dual, 420
Langrange function, 420
norm, 158, 287, 301, 308, 324, 327, 410
optimization, 83–85, 89–91, 99, 103, 116, 142, 194, 222, 237, 278, 279, 284, 292, 295, 302, 315, 318, 319
optimization problem, 420
problem, 84, 91, 278, 279, 282, 284, 299, 300, 302, 307, 313, 317, 318, 320, 331, 420, 425
variables, 83, 86, 90, 292, 293
duality
Fenchel, 300, 307, 328, 426
gap, 420
strong, 84, 420, 425, 426
weak, 425

early stopping, 165, 167, 170
edge, 149, 246
eigenvalue, 411, 413, 418
emphasis function, 262
empirical
error, 10
kernel map, 113, 114
kernel maps, 112
Rademacher complexity, 30
risk, 10
risk minimization, *see* risk minimization
empty string, 122, 360
ensemble
algorithms, 145
hypotheses, 155
methods, 145, 155, 165, 250, 251
entropy, 168, 169, 199, 201, 226, 227, 295, 296, 298, 299, 302, 306–308, 312, 317, 330, 331, 344, 345, 438, 439, 449–454, 456, 457, 477, 482
binary, 450
binary function, 449
conditional relative, 319, 331
maximum, 295
maximum conditional, 315
Rényi, 456
regularized, 344
relative, 450
relative conditional, 316, 452
unnormalized relative, 453
envelope, 290

environment, 1, 7, 379–381, 387, 393, 397, 403, 404
  model, 379, 380, 387, 393, 397
  unknown, 403
epigraph, 416
equivalence queries, 363
equivalent, 409
Erdös, 43
ERM, *see* risk minimization
error, 10
  empirical, 10, 19–21, 57, 59, 65, 67, 88, 145, 148–151, 154, 168, 171, 172, 175, 211, 214, 227, 236, 241, 246–248, 252, 269, 270, 273, 275, 276, 286, 294, 334, 336
  estimation, 61–64, 67, 73
  excess, 64, 65, 67, 74, 76–78
  generalization, 10, 11, 16, 19–22, 24, 26, 43, 59, 65, 66, 69, 70, 79, 86, 87, 95, 97–99, 140, 155, 161, 166, 172, 178, 201, 202, 204, 212, 214, 217, 230, 238, 241–243, 252, 268, 276, 323, 334, 336, 342
  leave-one-out, 72, 85, 86, 193, 194, 293, 294, 342
  mean squared, 228, 275, 277, 289
  reconstruction, 353
  test, 6, 155
  training, 65, 155, 174
  true, 20
error-correcting output codes (ECOC), 231
estimation error, 61
events, 140
  set, 429
examples, 4
  labeled, 5–7, 59, 71, 170, 223, 364
  misclassified, 171
  negative, 23
  positive, 16, 17, 227, 369, 376
excess error, 61
expectation, 431
  linearity, 86, 133, 336, 432
expected loss, 163
experience, 1
expert
  advice, 27, 177–179
  algorithm, 205
  best, 7, 178, 181–183, 205
exploration versus exploitation, 7
exponential inequality, 448

false
  negative, 12
  positive, 12, 103, 143, 256
  positive rate, 256
fat-shattered, 274
fat-shattering, 290
  dimension, 274
feature, 4
  extraction, 347
  function, 430
  mapping, 112–114, 117–119, 137, 139, 219, 220, 243, 244, 275, 276, 281, 283, 284, 297, 315, 347, 350
  missing, 228
  space, 92, 97–99, 104, 106, 107, 112, 130, 131, 139, 140, 167, 224, 243, 276, 280, 289, 376, 459
  vector, 156, 225, 234, 279, 280, 300, 320, 321, 327, 350
Fenchel
  conjugate, 423
  duality theorem, 300, 307, 328, 426
  problems, 425
Fermat's theorem, 415
final
  state, 123, 125, 360, 361, 364, 366, 370–372, 374, 398
  weight, 122–124
finite, 381
  horizon, 381
  query subset, 257
fixed point, 230, 388, 394, 397, 401
Frobenius
  norm, 412
  product, 412
Fubini's theorem, 44, 432
function
  affine, 151, 275, 455
  measurable, 22, 28, 273
  symmetric, 107, 262

game
  zero-sum, 163, 164, 204
gap penalty, 128
Gaussian, 430
  distribution, 303
  kernel, 110
generalization
  bound, 14, 15, 21, 35, 43, 58, 59, 93, 94, 97, 159, 217, 220, 235, 236, 243, 268, 272, 273, 280, 283, 284, 287, 288, 293, 331, 333, 341, 343
  error, 10
geometric margin
  L_1-, 157, 161
Gibbs distribution, 295, 299, 300, 306, 312, 317, 430
gradient, 415
  descent, 191, 192, 207, 289, 291, 294, 304, 313, 320, 327, 404
Gram matrix, 84, 108, 139
graph
  acyclic, 55
  Laplacian, 352, 357
  neighborhood, 352, 353
  structure, 235
graphical model, 234
group
  Lasso, *see* Lasso
  norm, 412
growth function, 29, 34–36, 40–42, 50, 56, 63, 333

Hölder's inequality, 166, 210, 301, 303, 322, 329, 411
Halving algorithm, 179, 181, 183
Hamming distance, 214, 231, 232, 234, 444
Hessian, 82, 84, 210, 313, 415, 417
Hilbert space, 73, 105, 107, 108, 110, 112, 113, 121, 138, 139, 141, 142, 410, 422, 423, 425, 444, 454
  pre-, 112
  reproducing kernel, 110–112, 117, 336, 350
hinge loss, 88, 89, 99, 174, 207, 341–343
  quadratic, 89, 343
Hoeffding's
  inequality, 19, 27, 59, 69, 134, 203, 268, 269, 437–439, 441, 443, 445, 447
  lemma, 188, 201, 306, 441, 445
horizon
  finite, 382
  infinite, 382, 385
Huber loss, 284
hyperparameters, 4, 5
hyperplane
  marginal, 81, 83, 87, 90, 91
  maximum-margin, 80, 81, 83, 197
  optimal, 100
hypothesis
  base, 151, 152, 155–157, 173, 174, 211, 255
  linear, 64, 97, 98, 155, 265, 277
  set, 5, 7, 10
  set finite, 15, 20, 21, 25, 27, 53, 180, 268
  single, 19

i.i.d., 10, 11, 15, 20, 32, 193, 194, 252, 296, 334–336, 404, 431, 480
impurity
  Gini index, 226
  misclassification, 226
inconsistent, 9
  case, 19, 27, 269
independent and identically distributed (i.i.d.), 431
inequality
  Azuma's, 202, 442, 445
  Bennett's, 447
  Bernstein's, 438, 440, 447
  Cauchy-Schwarz, 53, 97, 98, 112, 118, 197, 309, 310, 339, 341, 409, 410, 433
  Chebyshev's, 433, 447
  concentration, xiv, 437, 445
  exponential, 448
  Hölder's, 166, 210, 301, 303, 322, 329, 411
  Hoeffding's, 19, 27, 59, 69, 134, 203, 268, 269, 437–439, 441, 443, 445, 447
  Jensen's, 51–53, 77, 97, 104, 118, 133, 134, 188, 311, 318, 327, 443, 444, 450, 451

Khintchine-Kahane, 118, 186, 445
Log-sum, 451, 452
Markov's, 134, 354, 432, 433, 437
maximal, 324
McDiarmid's, 29, 31, 32, 139, 310, 335, 442, 443, 445
Pinsker's, 302, 318, 344, 439, 456
Slud's, 440
Young's, 410
information theory, xiv, 23, 407, 449, 450, 456
initial state, 381
input space, 9, 22, 30, 33, 52, 56, 79, 105–109, 112, 121, 130, 139, 173, 197, 213, 240, 267, 275, 285, 351, 449
instances, 9
inverse
generalized, 412
Isomap, 351, 352, 356
iterative scaling
generalized, 313

Jensen's inequality, 51–53, 77, 97, 104, 118, 133, 134, 188, 311, 318, 327, 443, 444, 450, 451
Johnson-Lindenstrauss lemma, 348, 354, 356
joint probability mass function, 429

kernel, 105, 106
approximate feature maps, 131
bigram, 128, 129
bigram sequence, 129
continuous, 131, 142
convolution, 136
difference, 138
empirical map, 112, 113
functions, 107, 130–132, 135, 137, 138, 222, 343, 350
gappy bigram, 129
Gaussian, 110, 113, 116
map empirical, 114
matrix, 108, 113–116, 118, 128, 143, 244, 270, 278, 280, 282, 284, 285, 293, 343, 344, 350, 352–354, 357
methods, xiv, 85, 105, 106, 130, 136, 351
negative definite symmetric, 105, 119, 121, 141
normalized, 112, 113, 116, 137, 285
PCA, 347, 349–354, 356, 357
polynomial, 108, 109, 131, 139
positive definite, 108
positive definite symmetric, 110–119, 121, 137–140, 197, 199, 219, 220, 233, 243, 276, 281–284, 289, 293, 336, 338, 350
positive semidefinite, 108
rational, 105, 122, 127, 142
ridge regression, 267, 275–277, 292, 294, 333, 343
sequence, 121, 129
shift-invariant, 131
sigmoid, 110
Khintchine-Kahane inequality, 118, 186, 445
KKT conditions, 83, 89, 221, 278, 282, 284, 421
KPCA, 347, 349–354, 356, 357
Kullback-Leibler divergence, 344, 450, 456

labels, 4, 9
Lagrange, 83, 89, 90, 99, 101, 102, 166, 304, 313, 324, 420
dual function, *see* dual
function, 420
multipliers, 101, 102
variables, 83, 89, 90
Lagrangian, 83, 89, 90, 221, 278, 282, 284, 420–422
Laplace distribution, 430
Laplacian eigenmaps, 351, 352, 357
Lasso, 267, 275, 285–288, 290, 291, 293, 294, 343
group, 289
on-line, 294
law of large numbers, 394
learner
strong, 146
weak, 145, 146
learning
active, 7, 362
algorithm, 1, 4–6, 9, 19, 20, 23, 24, 27, 43, 46, 47, 55, 57, 71, 80, 85, 98, 146,

148, 150, 168, 173, 179, 186, 202, 252, 257, 333, 334, 343, 362
algorithm PAC, 12, 16, 26–28, 146
algorithm weak, 146, 244
on-line, 7, 177
passive, 7
policy, 401
problem, 380
reinforcement, 7, 379
with queries, 363
leave-one-out
cross-validation, 71
error, 85
lemma
Hoeffding's, 188, 201, 306, 441, 445
Johnson-Lindenstrauss, 348, 354, 356
Massart's, 35, 51, 287
Sauer's, 40–43, 49, 50, 55
Talagrand's, 52, 216, 217, 242
linear
-ly separable labeling, 50
algebra, 409
classification problem, 79
classifiers, 79
Lipschitz
function, 52, 93
LLE, 353, 354, 356, 357
locally linear embedding, *see* LLE
Log-sum inequality, 451, 452
logistic, 330
form, 326
loss, 153
multinomial regression, 315
regression, 153, 315, 325
logistic regression
L_1-regularized, 325
loss
ϵ-insensitive, 282
convex, 77, 153, 250, 284
function, 268
hinge, 88, 89, 99, 174, 207, 341–343
Huber, 284
logistic, 153
margin, 92
matrix, 163
quadratic ϵ-insensitive, 283
quadratic hinge, 88, 89, 343
squared, 268
loss function, 4

manifold learning, 3
margin
L_1-, 156
L_1-geometric, 156
confidence, 92
geometric, 80
hard, 88
loss function, 92
soft, 88
Markov decision process, *see* MDP
Markov's inequality, 134, 354, 432, 433, 437
martingale differences, 441
Massart's lemma, 35, 51, 287
Maxent
conditional, 316
conditional models, 315
conditional principle, 316
conditional structural models, 330
models, 295, 298, 299, 306, 307, 312, 315–317, 319–321, 325–327, 330–332
principle, 298, 299, 302, 306, 312, 315, 316, 319, 330
structural models, 312
unregularized, 298
unregularized conditional, 316
maximal inequality, 324
maximum a posteriori, 297
maximum entropy models, 295
maximum likelihood
principle, 296
McDiarmid's inequality, 29, 31, 32, 139, 310, 335, 442, 443, 445
MDP, 380, 381, 383, 385, 389–391, 393
finite, 382, 385–387, 399
partially observable, 403
mean squared error, 268
measurable, 429
membership queries, 363
Mercer's condition, 107, 142
minimization, 361
mistake
bound, 179

bound model, 179
model, 178
mixed strategy, 163
model
-based, 393
-free approach, 393
selection, 61, 71
model selection, 61
moment-generating function, 354, 434, 445
mono-label case, 213
Moore-Penrose pseudo-inverse, 412
multi-class classification, xiii, 168, 213–215, 217, 219–222, 224, 225, 228, 229, 232, 233, 235–237, 264, 315, 321, 331
multi-label case, 213
mutual information, 453

NFA, 361
node impurity, 226
noise, 23
non-realizable, 29
non-stationary policy, 382
norm, 409
dual, 158, 287, 301, 308, 324, 327, 410
Frobenius, 412
matrix, 411
operator induced, 411
spectral, 411
normal, 430
distribution, 330, 355, 358, 430, 434, 443, 460
normalization, 110
normalized discounted cumulative gain, *see* DCG normalized

Occam's razor principle, 23, 43, 79, 269, 362
off-policy algorithm, 401
on-line learning, 7, 177
on-policy algorithm, 402
one-versus-all, 229–232, 236
one-versus-one, 229–232, 238
orthogonal projection, 412
outlier, 87, 168
OVA, *see* one-versus-all
OVO, *see* one-versus-one

PAC, 11
agnostic learning, 22, 48, 61
learnable, 11
learnable with membership queries, 363
learning, 12, 14, 16–19, 22, 23, 26–28, 45, 57, 145, 146, 343, 361, 363, 364, 376
learning algorithm, 11, 12, 16, 26–28, 146
learning framework, 9
weakly learnable, 145
packing numbers, 49
pairwise
consistent, 257
independence, 258
parallelogram identity, 457
part-of-speech tagging, 233
partition function, 300, 430
passive, 362
learning, 7
PCA, 347–351, 356–358
penalized risk estimate, 211
Perceptron
algorithm, 100, 190–193, 196–199, 201, 206–208, 265
dual, 197
kernel, 197, 206, 212
update, 207
voted, 193, 197
Pinsker's inequality, 302, 318, 344, 439, 456
pivot, 261
planning, 379, 387
pointwise
maximum, 418
supremum, 418
Poisson distribution, 430
policy, 379–381, 390, 393
iteration algorithm, 390
value, 379
POMDP, *see* MDP partially observable
positive, 12
definite, 105, 108, 412
definite symmetric, 107, 108
semidefinite, 108, 412

precision, 263
preference
  -based setting, 240, 257
  function, 240, 257
prefix-tree automaton, 370
primal problem, 420
principal component analysis, *see* PCA
prior
  knowledge, 5
  probability, 27
probabilistic method, 43
probability, 11
  density function, 429
  distribution, 429
  mass function, 429
  space, 429
probably approximately correct, *see* PAC
probit model, 330
proper, 422
pseudo-dimension, 267, 272
pure strategies, 163

Q-learning
  algorithm, 393, 394, 398, 399
QP, 82, 84, 100, 101, 230, 235, 282, 288
  convex, 82, 91, 282–284
quadratic
  ϵ-insensitive loss, 283
  hinge loss, 88
  programming, 82
  SVR, 283
QuickSort algorithm, 261
  randomized, 260

Rényi
  divergence, 456
  entropy, 456
Rademacher complexity, 29–36, 43, 48, 50–53, 63, 68, 79, 97, 100, 118, 157–159, 169, 213, 219, 220, 236, 239, 241, 243, 251, 263, 267, 269, 270, 274, 275, 287, 298, 316, 333, 460
  bound, 43, 287, 298, 316
  empirical, 29–31, 33, 34, 49, 51–53, 93, 97, 117, 118, 143, 157, 216, 250, 271, 277, 283, 287, 323, 324, 327, 443
  generalization bound, 158
  local, 48
Rademacher variables, 30, 32, 51, 118, 186, 219, 443
radial basis function, 110
Radon's theorem, 38
random variable, 429
Randomized-Weighted-Majority algorithm, 184, 186, 209
RankBoost, 236–239, 244–255, 264, 265
ranking, 3, 239, 259
RankPerceptron, 265
rational, 126
Rayleigh quotient, 349, 413
RBF, *see* radial basis function
realizable, 29
recall, 263
reconstruction error, 348
regression, 3, 228, 267
  kernel, 267, 275, 276
  linear, 267, 275
  ordinal, 264
  support vector, 267, 275, 281
regret, 7, 178, 258
  external, 178, 205
  internal, 205
  swap, 205
regular
  expressions, 361
  languages, 361
regularization
  -based algorithm, 72
  parameter, 73
  path, 288
  term, 73
reinforcement learning, 7, 379
relative entropy, 450
representer theorem, 117
reproducing
  property, 111
reversible, 370
  languages, 370
reward, 379
  probability, 381
risk, 10
risk minimization
  empirical, 34, 62–65, 67, 68, 70, 73, 236

structural, 64
voted, 78
RKHS, *see* Hilbert space
ROC curve, 239, *see also* AUC, 255, 256, 264
RWM algorithm, *see* Randomized-Weighted-Majority algorithm

sample
complexity, 1, 9
space, 429
Sanov's theorem, 439
SARSA algorithm, 402, 403
Sauer's lemma, 40–43, 49, 50, 55
scoring function, 240
semi-supervised learning, 6
shattering, 36, 271
shortest-distance, 123
algorithm, 136
singular value, 412
decomposition, 350, 412, 413
singular vector, 412
slack variables, 87
Slater's condition, 421
weak, 421
Slud's inequality, 440
SMO algorithm, 84, 100–102
society, 213
soft-max, 235
sphere trees, 225
SPSD, *see* symmetric positive semidefinite
squared loss, 268
SRM, 64–70, 72, 73, 77
stability, 334
algorithmic, 333
coefficient, 334
standard deviation, 432
start state, 381
state, 379, 380
state-action value function, 385, 394, 398, 402
stationary
point, 415
policy, 382
stochastic
approximation, 394
assumption, 361, 362
scenario, 21
subgradient descent, 191
structural risk minimization, *see* risk minimization
structured
output, 233
stumps, 154
subdifferential, 337, 422
subgradient, 337, 422
submultiplicative property, 411
supervised learning, 6
support vector, 83, 90, 193
support vector machines, *see* SVM
support vector networks, *see* SVM
SVD, *see* singular value decomposition
SVM, 79–84, 86, 90, 91, 100, 101, 103, 105, 116, 130, 143, 156, 162, 170, 194, 196, 207, 222, 243, 281
multi-class, 221, 236
SVMStruct, 235
SVR, 275, 281–285, 289, 291–293, 333, 337, 339–341
dual, 291
on-line, 291
quadratic, 284, 294
symmetric, 411
positive semidefinite, 108, 110, 114, 115, 128, 279, 357, 412, 413

Talagrand's lemma, 52, 216, 217, 242
tensor product, 114
test sample, 4
theorem
Fenchel duality, 300, 307, 328, 426
Fermat's, 415
Fubini's, 44, 432
Radon's, 38
representer, 117
Sanov's, 439
trace, 411
training sample, 4
transducer
weighted, 122
transductive inference, 6

transition, 360
  probability, 381
transpose, 411
trigrams, 106
true positive rate, 256

uncentered, 356
uncombined algorithms, 213, 221
uncorrelated, 433
uniform convergence bound, 15
unnormalized relative entropy, 453
unstable, 228
unsupervised learning, 6
update rule, 401

validation sample, 4
validation set, 68
value, 382
  iteration algorithm, 387
variance, 432
VC-dimension, 29, 36–40, 42, 43, 45, 46, 48–51, 53–57, 63, 68, 77, 79, 91, 98, 100, 103, 104, 145, 154, 155, 158, 167, 168, 170, 172, 179, 238, 243, 263, 267, 271–273, 290, 333, 376, 377
  generalization bound, 57
visualization, 347
voted risk minimization, *see* risk minimization

weight, 181
  function, 262
Weighted-Majority algorithm, 181, 183, 184, 186, 198, 205
Widrow-Hoff algorithm, 289, 290
  on-line, 291
Winnow
  algorithm, 198, 199, 206
  update, 198
witness, 271
WM algorithm, *see* Weighted-Majority algorithm

Young's inequality, 410

zero-sum game, 163