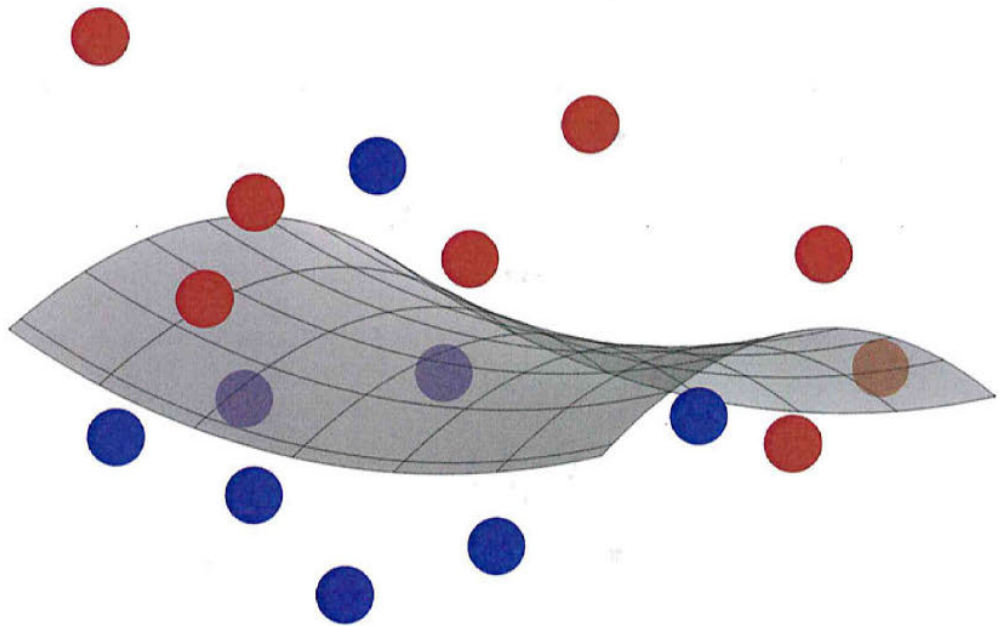


Foundations of Machine Learning

second edition



Mehryar Mohri,
Afshin Rostamizadeh,
and Ameet Talwalkar

Foundations of Machine Learning

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar

second edition

This book is a general introduction to machine learning that can serve as a textbook for graduate students and a reference for researchers. 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. The authors aim to present novel theoretical tools and concepts while giving concise proofs even for relatively advanced topics.

Foundations of Machine Learning is unique in its focus on the analysis and theory of algorithms. The first four chapters lay the theoretical foundation for what follows; subsequent chapters are mostly self-contained. Topics covered include the Probably Approximately Correct (PAC) learning framework; generalization bounds based on Rademacher complexity and VC-dimension; Support Vector Machines (SVMs); kernel methods; boosting; on-line learning; multi-class classification; ranking; regression; algorithmic stability; dimensionality reduction; learning automata and languages; and reinforcement learning. Each chapter ends with a set of exercises. Appendixes provide additional material including concise probability review.

This second edition offers three new chapters, on model selection, maximum entropy models, and conditional entropy models. New material in the appendixes includes a major section on Fenchel duality, expanded coverage of concentration inequalities, and an entirely new entry on information theory. More than half of the exercises are new to this edition.

Mehryar Mohri is Professor of Computer Science at New York University's Courant Institute of Mathematical Sciences and a Research Scientist at Google Research. Afshin Rostamizadeh is a Research Scientist at Google Research. Ameet Talwalkar is Assistant Professor in the Machine Learning Department in the School of Computer Science at Carnegie Mellon University.

Adaptive Computation and Machine Learning series

"A clear, rigorous treatment of machine learning that covers a broad range of problems and methods from a theoretical perspective. This edition includes many updates, including new chapters on model selection and maximum entropy methods. It will be a standard graduate-level reference."—Peter Bartlett, Professor of Computer Science, University of California, Berkeley

"*Foundations of Machine Learning* is a neat and mathematically rigorous book providing broad coverage of basic and advanced topics in machine learning, but is also a valuable textbook for graduate-level courses in the modern theory of machine learning. This book is unique in its content and style, a 'must-have' reference book for researchers and students."—Claudio Gentile, Inria Lille and Google Research, New York

"I've found the first edition of this book to be a valuable resource in five or so years of teaching—and look forward to using the much-improved and expanded second edition in future courses."—Aryeh Kontorovich, Associate Professor of Computer Science, Ben-Gurion University

The MIT Press Massachusetts Institute of Technology
Cambridge, Massachusetts 02142 <http://mitpress.mit.edu>

Cover diagram: Mehryar Mohri and Afshin Rostamizadeh
Cover design: Erin Hasley



Adaptive Computation and Machine Learning

Francis Bach, Editor

A complete list of books published in The Adaptive Computation and Machine Learning series appears at the back of this book.

Foundations of Machine Learning

second edition

Mehryar Mohri

Afshin Rostamizadeh

Ameet Talwalkar

The MIT Press
Cambridge, Massachusetts
London, England

© 2018 Massachusetts Institute of Technology

This work is subject to a Creative Commons CC BY-NC-ND license. Subject to such license, all rights are reserved.



This book was set in L^AT_EX 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

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

Contents

vii

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

Contents

ix

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

Contents

xi

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

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.

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 $L: \mathcal{Y} \times \mathcal{Y}' \rightarrow \mathbb{R}_+$. 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

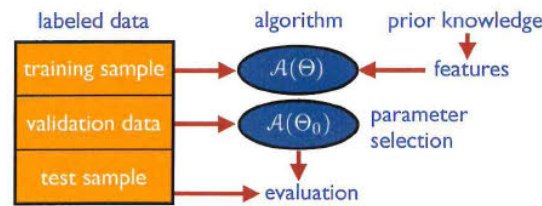


Figure 1.1

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.

tion task,
gorithm is
ag sample.

e scenarios
id method
ie learning

us training
nmon sce-
The spam
supervised

ning data,
d example
e the per-
example of

nsisting of
en points.
a is easily
; arising in
be framed
ibution of
rformance
ch this can
d machine

er receives
wever, the
particular
d matches
ever, as in
rformance
been fully

- *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,

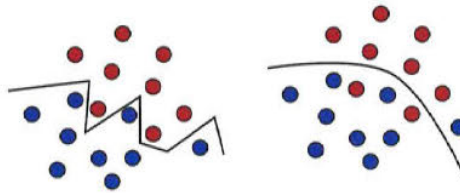


Figure 1.2

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.