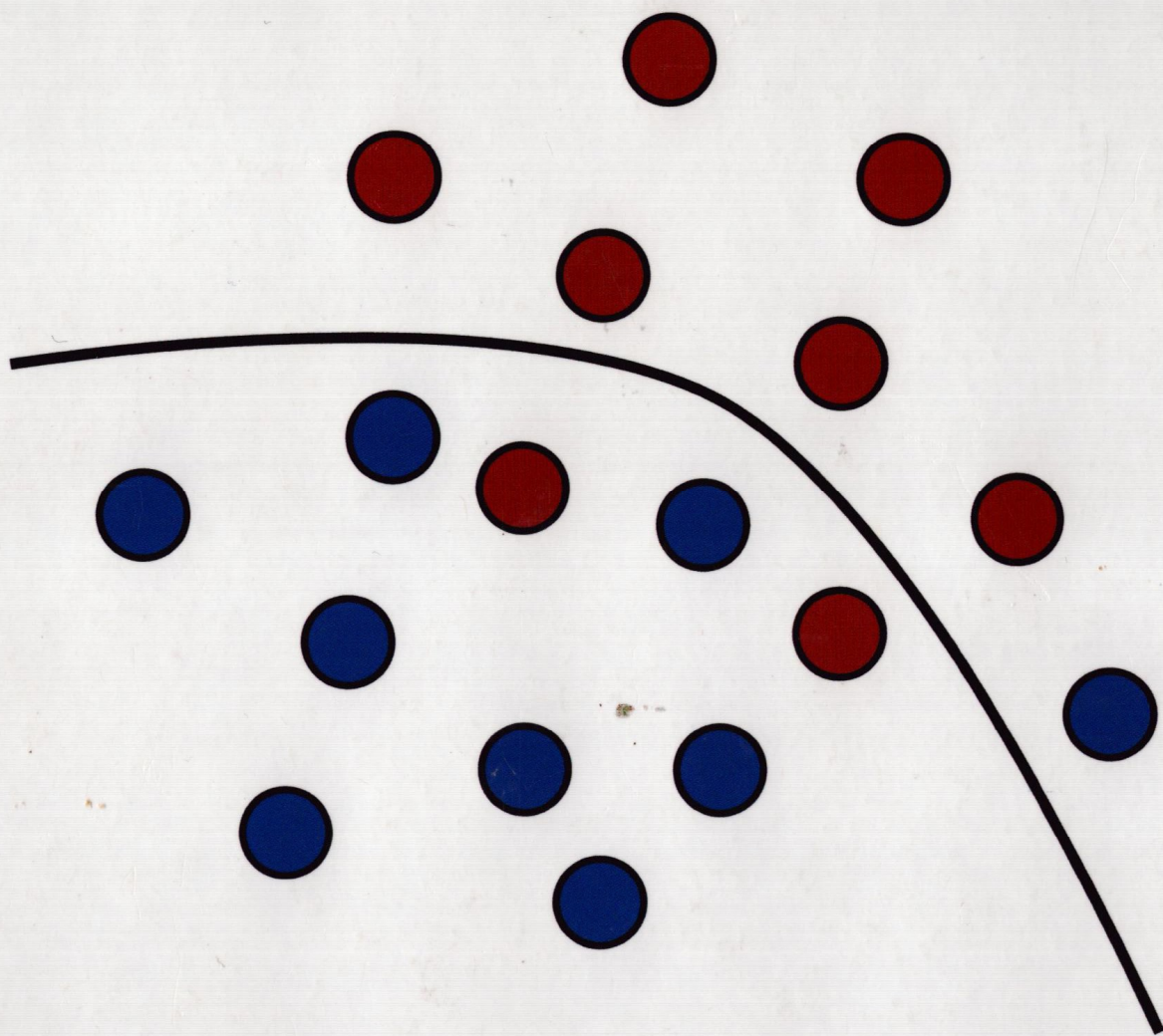


Foundations of Machine Learning



Mehryar Mohri,
Afshin Rostamizadeh,
and Ameet Talwalkar

Foundations of Machine Learning

Mohri, Rostamizadeh, and Talwalkar





Foundations of Machine Learning

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar

This graduate-level textbook introduces fundamental concepts and methods in machine learning. It describes several important modern algorithms, provides the theoretical underpinnings of these algorithms, and illustrates key aspects for their application. The authors aim to present novel theoretical tools and concepts while giving concise proofs even for relatively advanced topics.

Foundations of Machine Learning fills the need for a general textbook that also offers theoretical details and an emphasis on proofs. Certain topics that are often treated with insufficient attention are discussed in more detail here; for example, entire chapters are devoted to regression, multi-class classification, and ranking. The first three chapters lay the theoretical foundation for what follows, but each remaining chapter is mostly self-contained. The appendix offers a concise probability review, a short introduction to convex optimization, tools for concentration bounds, and several basic properties of matrices and norms used in the book.

The book is intended for graduate students and researchers in machine learning, statistics, and related areas; it can be used either as a textbook or as a reference text for a research seminar.

Mehryar Mohri is Professor of Computer Science at New York University's Courant Institute of Mathematical Sciences and a Research Consultant at Google Research. Afshin Rostamizadeh is a Research Scientist at Google Research. Ameet Talwalkar is a National Science Foundation Postdoctoral Fellow in the Department of Electrical Engineering and Computer Science at the University of California, Berkeley.

Adaptive Computation and Machine Learning series

"A solid, comprehensive, and self-contained book providing a uniform treatment of a very broad collection of machine learning algorithms and problems. *Foundations of Machine Learning* is an essential reference book for corporate and academic researchers, engineers, and students."—Corinna Cortes, Head of Google Research, NY

"Finally, a book that is both broad enough to cover many algorithmic topics of machine learning and mathematically deep enough to introduce the required theory for a graduate-level course. *Foundations of Machine Learning* is a great achievement and a significant contribution to the machine learning community."—Yishay Mansour, School of Computer Science, Tel Aviv University

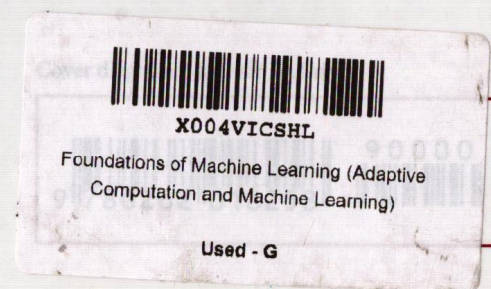
The MIT Press

Massachusetts Institute of Technology

Cambridge, Massachusetts 02142

<http://mitpress.mit.edu>

978-0-262-01825-8



Foundations of Machine Learning

Foundations of Machine Learning

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar

The MIT Press
Cambridge, Massachusetts
London, England

© 2012 Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

MIT Press books may be purchased at special quantity discounts for business or sales promotional use. For information, please email special_sales@mitpress.mit.edu or write to Special Sales Department, The MIT Press, 55 Hayward Street, Cambridge, MA 02142.

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

Mohri, Mehryar.

Foundations of machine learning / Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar.

p. cm. - (Adaptive computation and machine learning series)

Includes bibliographical references and index.

ISBN 978-0-262-01825-8 (hardcover : alk. paper) 1. Machine learning. 2. Computer algorithms. I. Rostamizadeh, Afshin. II. Talwalkar, Ameet. III. Title.

Q325.5.M64 2012

006.3'1-dc23

2012007249

10 9 8 7 6 5 4 3 2 1

Contents

Preface	xi
1 Introduction	1
1.1 Applications and problems	1
1.2 Definitions and terminology	3
1.3 Cross-validation	5
1.4 Learning scenarios	7
1.5 Outline	8
2 The PAC Learning Framework	11
2.1 The PAC learning model	11
2.2 Guarantees for finite hypothesis sets — consistent case	17
2.3 Guarantees for finite hypothesis sets — inconsistent case	21
2.4 Generalities	24
2.4.1 Deterministic versus stochastic scenarios	24
2.4.2 Bayes error and noise	25
2.4.3 Estimation and approximation errors	26
2.4.4 Model selection	27
2.5 Chapter notes	28
2.6 Exercises	29
3 Rademacher Complexity and VC-Dimension	33
3.1 Rademacher complexity	34
3.2 Growth function	38
3.3 VC-dimension	41
3.4 Lower bounds	48
3.5 Chapter notes	54
3.6 Exercises	55
4 Support Vector Machines	63
4.1 Linear classification	63
4.2 SVMs — separable case	64

4.2.1	Primal optimization problem	64
4.2.2	Support vectors	66
4.2.3	Dual optimization problem	67
4.2.4	Leave-one-out analysis	69
4.3	SVMs — non-separable case	71
4.3.1	Primal optimization problem	72
4.3.2	Support vectors	73
4.3.3	Dual optimization problem	74
4.4	Margin theory	75
4.5	Chapter notes	83
4.6	Exercises	84
5	Kernel Methods	89
5.1	Introduction	89
5.2	Positive definite symmetric kernels	92
5.2.1	Definitions	92
5.2.2	Reproducing kernel Hilbert space	94
5.2.3	Properties	96
5.3	Kernel-based algorithms	100
5.3.1	SVMs with PDS kernels	100
5.3.2	Representer theorem	101
5.3.3	Learning guarantees	102
5.4	Negative definite symmetric kernels	103
5.5	Sequence kernels	106
5.5.1	Weighted transducers	106
5.5.2	Rational kernels	111
5.6	Chapter notes	115
5.7	Exercises	116
6	Boosting	121
6.1	Introduction	121
6.2	AdaBoost	122
6.2.1	Bound on the empirical error	124
6.2.2	Relationship with coordinate descent	126
6.2.3	Relationship with logistic regression	129
6.2.4	Standard use in practice	129
6.3	Theoretical results	130
6.3.1	VC-dimension-based analysis	131
6.3.2	Margin-based analysis	131
6.3.3	Margin maximization	136
6.3.4	Game-theoretic interpretation	137

6.4	Discussion	140
6.5	Chapter notes	141
6.6	Exercises	142
7	On-Line Learning	147
7.1	Introduction	147
7.2	Prediction with expert advice	148
7.2.1	Mistake bounds and Halving algorithm	148
7.2.2	Weighted majority algorithm	150
7.2.3	Randomized weighted majority algorithm	152
7.2.4	Exponential weighted average algorithm	156
7.3	Linear classification	159
7.3.1	Perceptron algorithm	160
7.3.2	Winnow algorithm	168
7.4	On-line to batch conversion	171
7.5	Game-theoretic connection	174
7.6	Chapter notes	175
7.7	Exercises	176
8	Multi-Class Classification	183
8.1	Multi-class classification problem	183
8.2	Generalization bounds	185
8.3	Uncombined multi-class algorithms	191
8.3.1	Multi-class SVMs	191
8.3.2	Multi-class boosting algorithms	192
8.3.3	Decision trees	194
8.4	Aggregated multi-class algorithms	198
8.4.1	One-versus-all	198
8.4.2	One-versus-one	199
8.4.3	Error-correction codes	201
8.5	Structured prediction algorithms	203
8.6	Chapter notes	206
8.7	Exercises	207
9	Ranking	209
9.1	The problem of ranking	209
9.2	Generalization bound	211
9.3	Ranking with SVMs	213
9.4	RankBoost	214
9.4.1	Bound on the empirical error	216
9.4.2	Relationship with coordinate descent	218

9.4.3	Margin bound for ensemble methods in ranking	220
9.5	Bipartite ranking	221
9.5.1	Boosting in bipartite ranking	222
9.5.2	Area under the ROC curve	224
9.6	Preference-based setting	226
9.6.1	Second-stage ranking problem	227
9.6.2	Deterministic algorithm	229
9.6.3	Randomized algorithm	230
9.6.4	Extension to other loss functions	231
9.7	Discussion	232
9.8	Chapter notes	233
9.9	Exercises	234
10	Regression	237
10.1	The problem of regression	237
10.2	Generalization bounds	238
10.2.1	Finite hypothesis sets	238
10.2.2	Rademacher complexity bounds	239
10.2.3	Pseudo-dimension bounds	241
10.3	Regression algorithms	245
10.3.1	Linear regression	245
10.3.2	Kernel ridge regression	247
10.3.3	Support vector regression	252
10.3.4	Lasso	257
10.3.5	Group norm regression algorithms	260
10.3.6	On-line regression algorithms	261
10.4	Chapter notes	262
10.5	Exercises	263
11	Algorithmic Stability	267
11.1	Definitions	267
11.2	Stability-based generalization guarantee	268
11.3	Stability of kernel-based regularization algorithms	270
11.3.1	Application to regression algorithms: SVR and KRR	274
11.3.2	Application to classification algorithms: SVMs	276
11.3.3	Discussion	276
11.4	Chapter notes	277
11.5	Exercises	277
12	Dimensionality Reduction	281
12.1	Principal Component Analysis	282

12.2	Kernel Principal Component Analysis (KPCA)	283
12.3	KPCA and manifold learning	285
12.3.1	Isomap	285
12.3.2	Laplacian eigenmaps	286
12.3.3	Locally linear embedding (LLE)	287
12.4	Johnson-Lindenstrauss lemma	288
12.5	Chapter notes	290
12.6	Exercises	290
13	Learning Automata and Languages	293
13.1	Introduction	293
13.2	Finite automata	294
13.3	Efficient exact learning	295
13.3.1	Passive learning	296
13.3.2	Learning with queries	297
13.3.3	Learning automata with queries	298
13.4	Identification in the limit	303
13.4.1	Learning reversible automata	304
13.5	Chapter notes	309
13.6	Exercises	310
14	Reinforcement Learning	313
14.1	Learning scenario	313
14.2	Markov decision process model	314
14.3	Policy	315
14.3.1	Definition	315
14.3.2	Policy value	316
14.3.3	Policy evaluation	316
14.3.4	Optimal policy	318
14.4	Planning algorithms	319
14.4.1	Value iteration	319
14.4.2	Policy iteration	322
14.4.3	Linear programming	324
14.5	Learning algorithms	325
14.5.1	Stochastic approximation	326
14.5.2	TD(0) algorithm	330
14.5.3	Q-learning algorithm	331
14.5.4	SARSA	334
14.5.5	TD(λ) algorithm	335
14.5.6	Large state space	336
14.6	Chapter notes	337

Conclusion	339
A Linear Algebra Review	341
A.1 Vectors and norms	341
A.1.1 Norms	341
A.1.2 Dual norms	342
A.2 Matrices	344
A.2.1 Matrix norms	344
A.2.2 Singular value decomposition	345
A.2.3 Symmetric positive semidefinite (SPSD) matrices	346
B Convex Optimization	349
B.1 Differentiation and unconstrained optimization	349
B.2 Convexity	350
B.3 Constrained optimization	353
B.4 Chapter notes	357
C Probability Review	359
C.1 Probability	359
C.2 Random variables	359
C.3 Conditional probability and independence	361
C.4 Expectation, Markov's inequality, and moment-generating function	363
C.5 Variance and Chebyshev's inequality	365
D Concentration inequalities	369
D.1 Hoeffding's inequality	369
D.2 McDiarmid's inequality	371
D.3 Other inequalities	373
D.3.1 Binomial distribution: Slud's inequality	374
D.3.2 Normal distribution: tail bound	374
D.3.3 Khintchine-Kahane inequality	374
D.4 Chapter notes	376
D.5 Exercises	377
E Notation	379
References	381
Index	397

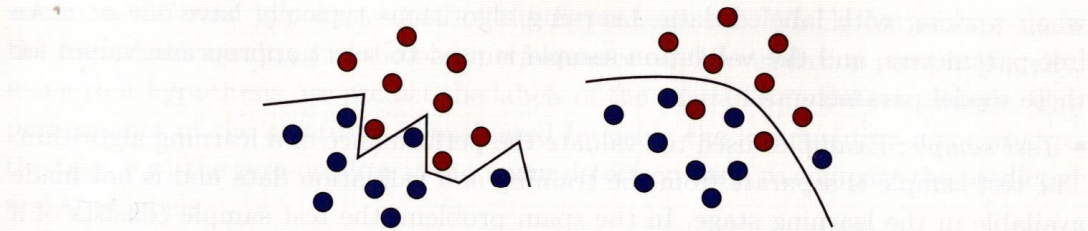


Figure 1.1 The zig-zag line on the left panel is consistent over the blue and red training sample, but it is a complex separation surface that is not likely to generalize well to unseen data. In contrast, the decision surface on the right panel is simpler and might generalize better in spite of its misclassification of a few points of the training sample.

Which concept families can actually be learned, and under what conditions? How well can these concepts be learned computationally?

1.2 Definitions and terminology

We will use the canonical problem of spam detection as a running example to illustrate some basic definitions and to describe the use and evaluation of machine learning algorithms in practice. Spam detection is the problem of learning to automatically classify email messages as either SPAM or non-SPAM.

- *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.
- *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 1.4.
- *Validation sample*: Examples used to tune the parameters of a learning algorithm

when working with labeled data. Learning algorithms typically have one or more free parameters, and the validation sample is used to select appropriate values for these model parameters.

- *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 misclassification) loss defined over $\{-1, +1\} \times \{-1, +1\}$ by $L(y, y') = 1_{y' \neq y}$ and the squared loss defined over $I \times I$ by $L(y, y') = (y' - y)^2$, where $I \subseteq \mathbb{R}$ 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. 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 free parameters of the algorithm. Also, when the labeled sample is relatively small, the amount of training data is often chosen to be larger than that of 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 by fixing different values of its free parameters. For each value of these parameters, the algorithm

selects a different hypothesis out of the hypothesis set. We choose among them the hypothesis resulting in the best performance on the validation sample. 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. A learning algorithm may be *consistent*, that is it may commit no error on the examples of the training data, and yet have a poor performance on the test data. This occurs for consistent learners defined by very complex decision surfaces, as illustrated in figure 1.1, which tend to memorize a relatively small training sample instead of seeking to generalize well. This highlights the key distinction between memorization and generalization, which is the fundamental property sought for an accurate learning algorithm. Theoretical guarantees for consistent learners will be discussed with great detail in chapter 2.

1.3 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* (selection of the free parameters of the algorithm) and for training.

Let θ denote the vector of free parameters of the algorithm. For a fixed value of θ , the method consists of first randomly partitioning a given sample S of m labeled examples into n subsamples, or folds. The i th fold is thus a labeled sample $((x_{i1}, y_{i1}), \dots, (x_{im_i}, y_{im_i}))$ of size m_i . Then, for any $i \in [1, n]$, the learning algorithm is trained on all but the i th fold to generate a hypothesis h_i , and the performance of h_i is tested on the i th fold, as illustrated in figure 1.2a. The parameter value θ is evaluated based on the average error of the hypotheses h_i , which is called the *cross-validation error*. This quantity is denoted by $\hat{R}_{CV}(\theta)$ and defined by

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

The folds are generally chosen to have equal size, that is $m_i = m/n$ for all $i \in [1, n]$. How should n be chosen? The appropriate choice is subject to a trade-off and the topic of much learning theory research that we cannot address in this introductory