Shai Shalev-Shwartz and Shai Ben-David

UNDERSTANDING MACHINE LEARNING

FROM THEORY TO ALGORITHMS

UNDERSTANDING MACHINE LEARNING

From Theory to Algorithms

Shai Shalev-Shwartz
The Hebrew University, Jerusalem

Shai Ben-David
University of Waterloo, Canada

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

32 Avenue of the Americas, New York, NY 10013-2473, USA

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning, and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781107057135

First published 2014

Printed in the United States of America

A catalog record for this publication is available from the British Library.

Library of Congress Cataloging in Publication data
Shalev-Shwartz, Shai.
Understanding machine learning : from theory to algorithms /
Shai Shalev-Shwartz, The Hebrew University, Jerusalem,
Shai Ben-David, University of Waterloo, Canada.
pages cm
Includes bibliographical references and index.
ISBN 978-1-107-05713-5 (hardback)
1. Machine learning. 2. Algorithms. I. Ben-David, Shai. II. Title.
Q325.5.S475 2014
006.3′1–dc23 2014001779

ISBN 978-1-107-05713-5 Hardback

Contents

Preface

The term *machine learning* refers to the automated detection of meaningful patterns in data. In the past couple of decades it has become a common tool in almost any task that requires information extraction from large data sets. We are surrounded by a machine learning–based technology: Search engines learn how to bring us the best results (while placing profitable ads), antispam software learns to filter our e-mail messages, and credit card transactions are secured by a software that learns how to detect frauds. Digital cameras learn to detect faces and intelligent personal assistance applications on smart-phones learn to recognize voice commands. Cars are equipped with accident-prevention systems that are built using machine learning algorithms. Machine learning is also widely used in scientific applications such as bioinformatics, medicine, and astronomy.

One common feature of all of these applications is that, in contrast to more traditional uses of computers, in these cases, due to the complexity of the patterns that need to be detected, a human programmer cannot provide an explicit, fine-detailed specification of how such tasks should be executed. Taking examples from intelligent beings, many of our skills are acquired or refined through *learning* from our experience (rather than following explicit instructions given to us). Machine learning tools are concerned with endowing programs with the ability to "learn" and adapt.

The first goal of this book is to provide a rigorous, yet easy-to-follow, introduction to the main concepts underlying machine learning: What is learning? How can a machine learn? How do we quantify the resources needed to learn a given concept? Is learning always possible? Can we know whether the learning process succeeded or failed?

The second goal of this book is to present several key machine learning algorithms. We chose to present algorithms that on one hand are successfully used in practice and on the other hand give a wide spectrum of different learning techniques. Additionally, we pay specific attention to algorithms appropriate for large-scale learning (a.k.a. "Big Data"), since in recent years, our world has become increasingly "digitized" and the amount of data available for learning is dramatically increasing. As a result, in many applications data is plentiful and computation

time is the main bottleneck. We therefore explicitly quantify both the amount of data and the amount of computation time needed to learn a given concept.

The book is divided into four parts. The first part aims at giving an initial rigorous answer to the fundamental questions of learning. We describe a generalization of Valiant's Probably Approximately Correct (PAC) learning model, which is a first solid answer to the question "What is learning?" We describe the Empirical Risk Minimization (ERM), Structural Risk Minimization (SRM), and Minimum Description Length (MDL) learning rules, which show "how a machine can learn." We quantify the amount of data needed for learning using the ERM, SRM, and MDL rules and show how learning might fail by deriving a "no-free-lunch" theorem. We also discuss how much computation time is required for learning. In the second part of the book we describe various learning algorithms. For some of the algorithms, we first present a more general learning principle and then show how the algorithm follows the principle. While the first two parts of the book focus on the PAC model, the third part extends the scope by presenting a wider variety of learning models. Finally, the last part of the book is devoted to advanced theory.

We made an attempt to keep the book as self-contained as possible. However, the reader is assumed to be comfortable with basic notions of probability, linear algebra, analysis, and algorithms. The first three parts of the book are intended for first-year graduate students in computer science, engineering, mathematics, or statistics. It can also be accessible to undergraduate students with the adequate background. The more advanced chapters can be used by researchers intending to gather a deeper theoretical understanding.

ACKNOWLEDGMENTS

The book is based on Introduction to Machine Learning courses taught by Shai Shalev-Shwartz at Hebrew University and by Shai Ben-David at the University of Waterloo. The first draft of the book grew out of the lecture notes for the course that was taught at Hebrew University by Shai Shalev-Shwartz during 2010–2013. We greatly appreciate the help of Ohad Shamir, who served as a teaching assistant for the course in 2010, and of Alon Gonen, who served as TA for the course in 2011–2013. Ohad and Alon prepared a few lecture notes and many of the exercises. Alon, to whom we are indebted for his help throughout the entire making of the book, has also prepared a solution manual.

We are deeply grateful for the most valuable work of Dana Rubinstein. Dana has scientifically proofread and edited the manuscript, transforming it from lecture-based chapters into fluent and coherent text.

Special thanks to Amit Daniely, who helped us with a careful read of the advanced part of the book and wrote the advanced chapter on multiclass learnability. We are also grateful for the members of a book reading club in Jerusalem who have carefully read and constructively criticized every line of the manuscript. The members of the reading club are Maya Alroy, Yossi Arjevani, Aharon Birnbaum, Alon Cohen, Alon Gonen, Roi Livni, Ofer Meshi, Dan Rosenbaum, Dana Rubinstein, Shahar Somin, Alon Vinnikov, and Yoav Wald. We would also like to thank Gal Elidan, Amir Globerson, Nika Haghtalab, Shie Mannor, Amnon Shashua, Nati Srebro, and Ruth Urner for helpful discussions.

1

Introduction

The subject of this book is automated learning, or, as we will more often call it, Machine Learning (ML). That is, we wish to program computers so that they can "learn" from input available to them. Roughly speaking, learning is the process of converting experience into expertise or knowledge. The input to a learning algorithm is training data, representing experience, and the output is some expertise, which usually takes the form of another computer program that can perform some task. Seeking a formal-mathematical understanding of this concept, we'll have to be more explicit about what we mean by each of the involved terms: What is the training data our programs will access? How can the process of learning be automated? How can we evaluate the success of such a process (namely, the quality of the output of a learning program)?

1.1 WHAT IS LEARNING?

Let us begin by considering a couple of examples from naturally occurring animal learning. Some of the most fundamental issues in ML arise already in that context, which we are all familiar with.

Bait Shyness – Rats Learning to Avoid Poisonous Baits: When rats encounter food items with novel look or smell, they will first eat very small amounts, and subsequent feeding will depend on the flavor of the food and its physiological effect. If the food produces an ill effect, the novel food will often be associated with the illness, and subsequently, the rats will not eat it. Clearly, there is a learning mechanism in play here – the animal used past experience with some food to acquire expertise in detecting the safety of this food. If past experience with the food was negatively labeled, the animal predicts that it will also have a negative effect when encountered in the future.

Inspired by the preceding example of successful learning, let us demonstrate a typical machine learning task. Suppose we would like to program a machine that learns how to filter spam e-mails. A naive solution would be seemingly similar to the way rats learn how to avoid poisonous baits. The machine will simply *memorize* all previous e-mails that had been labeled as spam e-mails by the human user. When a

new e-mail arrives, the machine will search for it in the set of previous spam e-mails. If it matches one of them, it will be trashed. Otherwise, it will be moved to the user's inbox folder.

While the preceding "learning by memorization" approach is sometimes useful, it lacks an important aspect of learning systems – the ability to label unseen e-mail messages. A successful learner should be able to progress from individual examples to broader *generalization*. This is also referred to as *inductive reasoning* or *inductive inference*. In the bait shyness example presented previously, after the rats encounter an example of a certain type of food, they apply their attitude toward it on new, unseen examples of food of similar smell and taste. To achieve generalization in the spam filtering task, the learner can scan the previously seen e-mails, and extract a set of words whose appearance in an e-mail message is indicative of spam. Then, when a new e-mail arrives, the machine can check whether one of the suspicious words appears in it, and predict its label accordingly. Such a system would potentially be able correctly to predict the label of unseen e-mails.

However, inductive reasoning might lead us to false conclusions. To illustrate this, let us consider again an example from animal learning.

Pigeon Superstition: In an experiment performed by the psychologist B. F. Skinner, he placed a bunch of hungry pigeons in a cage. An automatic mechanism had been attached to the cage, delivering food to the pigeons at regular intervals with no reference whatsoever to the birds' behavior. The hungry pigeons went around the cage, and when food was first delivered, it found each pigeon engaged in some activity (pecking, turning the head, etc.). The arrival of food reinforced each bird's specific action, and consequently, each bird tended to spend some more time doing that very same action. That, in turn, increased the chance that the next random food delivery would find each bird engaged in that activity again. What results is a chain of events that reinforces the pigeons' association of the delivery of the food with whatever chance actions they had been performing when it was first delivered. They subsequently continue to perform these same actions diligently.[1]

What distinguishes learning mechanisms that result in superstition from useful learning? This question is crucial to the development of automated learners. While human learners can rely on common sense to filter out random meaningless learning conclusions, once we export the task of learning to a machine, we must provide well defined crisp principles that will protect the program from reaching senseless or useless conclusions. The development of such principles is a central goal of the theory of machine learning.

What, then, made the rats' learning more successful than that of the pigeons? As a first step toward answering this question, let us have a closer look at the bait shyness phenomenon in rats.

Bait Shyness revisited – rats fail to acquire conditioning between food and electric shock or between sound and nausea: The bait shyness mechanism in rats turns out to be more complex than what one may expect. In experiments carried out by Garcia (Garcia & Koelling 1996), it was demonstrated that if the unpleasant stimulus that follows food consumption is replaced by, say, electrical shock (rather than nausea), then no conditioning occurs. Even after repeated trials in which the consumption

[1] See: http://psychclassics.yorku.ca/Skinner/Pigeon

of some food is followed by the administration of unpleasant electrical shock, the rats do not tend to avoid that food. Similar failure of conditioning occurs when the characteristic of the food that implies nausea (such as taste or smell) is replaced by a vocal signal. The rats seem to have some "built in" prior knowledge telling them that, while temporal correlation between food and nausea can be causal, it is unlikely that there would be a causal relationship between food consumption and electrical shocks or between sounds and nausea.

We conclude that one distinguishing feature between the bait shyness learning and the pigeon superstition is the incorporation of *prior knowledge* that biases the learning mechanism. This is also referred to as *inductive bias*. The pigeons in the experiment are willing to adopt *any* explanation for the occurrence of food. However, the rats "know" that food cannot cause an electric shock and that the co-occurrence of noise with some food is not likely to affect the nutritional value of that food. The rats' learning process is biased toward detecting some kind of patterns while ignoring other temporal correlations between events.

It turns out that the incorporation of prior knowledge, biasing the learning process, is inevitable for the success of learning algorithms (this is formally stated and proved as the "No-Free-Lunch theorem" in Chapter 5). The development of tools for expressing domain expertise, translating it into a learning bias, and quantifying the effect of such a bias on the success of learning is a central theme of the theory of machine learning. Roughly speaking, the stronger the prior knowledge (or prior assumptions) that one starts the learning process with, the easier it is to learn from further examples. However, the stronger these prior assumptions are, the less flexible the learning is – it is bound, a priori, by the commitment to these assumptions. We shall discuss these issues explicitly in Chapter 5.

1.2 WHEN DO WE NEED MACHINE LEARNING?

When do we need machine learning rather than directly program our computers to carry out the task at hand? Two aspects of a given problem may call for the use of programs that learn and improve on the basis of their "experience": the problem's complexity and the need for adaptivity.

Tasks That Are Too Complex to Program.

- *Tasks Performed by Animals/Humans:* There are numerous tasks that we human beings perform routinely, yet our introspection concerning how we do them is not sufficiently elaborate to extract a well defined program. Examples of such tasks include driving, speech recognition, and image understanding. In all of these tasks, state of the art machine learning programs, programs that "learn from their experience," achieve quite satisfactory results, once exposed to sufficiently many training examples.
- *Tasks beyond Human Capabilities:* Another wide family of tasks that benefit from machine learning techniques are related to the analysis of very large and complex data sets: astronomical data, turning medical archives into medical knowledge, weather prediction, analysis of genomic data, Web search engines, and electronic commerce. With more and more available

In this book we shall discuss only a subset of the possible learning paradigms. Our main focus is on supervised statistical batch learning with a passive learner (for example, trying to learn how to generate patients' prognoses, based on large archives of records of patients that were independently collected and are already labeled by the fate of the recorded patients). We shall also briefly discuss online learning and batch unsupervised learning (in particular, clustering).

1.4 RELATIONS TO OTHER FIELDS

As an interdisciplinary field, machine learning shares common threads with the mathematical fields of statistics, information theory, game theory, and optimization. It is naturally a subfield of computer science, as our goal is to program machines so that they will learn. In a sense, machine learning can be viewed as a branch of AI (Artificial Intelligence), since, after all, the ability to turn experience into expertise or to detect meaningful patterns in complex sensory data is a cornerstone of human (and animal) intelligence. However, one should note that, in contrast with traditional AI, machine learning is not trying to build automated imitation of intelligent behavior, but rather to use the strengths and special abilities of computers to complement human intelligence, often performing tasks that fall way beyond human capabilities. For example, the ability to scan and process huge databases allows machine learning programs to detect patterns that are outside the scope of human perception.

The component of experience, or training, in machine learning often refers to data that is randomly generated. The task of the learner is to process such randomly generated examples toward drawing conclusions that hold for the environment from which these examples are picked. This description of machine learning highlights its close relationship with statistics. Indeed there is a lot in common between the two disciplines, in terms of both the goals and techniques used. There are, however, a few significant differences of emphasis; if a doctor comes up with the hypothesis that there is a correlation between smoking and heart disease, it is the statistician's role to view samples of patients and check the validity of that hypothesis (this is the common statistical task of hypothesis testing). In contrast, machine learning aims to use the data gathered from samples of patients to come up with a description of the causes of heart disease. The hope is that automated techniques may be able to figure out meaningful patterns (or hypotheses) that may have been missed by the human observer.

In contrast with traditional statistics, in machine learning in general, and in this book in particular, algorithmic considerations play a major role. Machine learning is about the execution of learning by computers; hence algorithmic issues are pivotal. We develop algorithms to perform the learning tasks and are concerned with their computational efficiency. Another difference is that while statistics is often interested in asymptotic behavior (like the convergence of sample-based statistical estimates as the sample sizes grow to infinity), the theory of machine learning focuses on finite sample bounds. Namely, given the size of available samples, machine learning theory aims to figure out the degree of accuracy that a learner can expect on the basis of such samples.

There are further differences between these two disciplines, of which we shall mention only one more here. While in statistics it is common to work under the assumption of certain presubscribed data models (such as assuming the normality of data-generating distributions, or the linearity of functional dependencies), in machine learning the emphasis is on working under a "distribution-free" setting, where the learner assumes as little as possible about the nature of the data distribution and allows the learning algorithm to figure out which models best approximate the data-generating process. A precise discussion of this issue requires some technical preliminaries, and we will come back to it later in the book, and in particular in Chapter 5.

1.5 HOW TO READ THIS BOOK

The first part of the book provides the basic theoretical principles that underlie machine learning (ML). In a sense, this is the foundation upon which the rest of the book is built. This part could serve as a basis for a minicourse on the theoretical foundations of ML.

The second part of the book introduces the most commonly used algorithmic approaches to supervised machine learning. A subset of these chapters may also be used for introducing machine learning in a general AI course to computer science, Math, or engineering students.

The third part of the book extends the scope of discussion from statistical classification to other learning models. It covers online learning, unsupervised learning, dimensionality reduction, generative models, and feature learning.

The fourth part of the book, Advanced Theory, is geared toward readers who have interest in research and provides the more technical mathematical techniques that serve to analyze and drive forward the field of theoretical machine learning.

The Appendixes provide some technical tools used in the book. In particular, we list basic results from measure concentration and linear algebra.

A few sections are marked by an asterisk, which means they are addressed to more advanced students. Each chapter is concluded with a list of exercises. A solution manual is provided in the course Web site.

1.5.1 Possible Course Plans Based on This Book

A 14 Week Introduction Course for Graduate Students:

1. Chapters 2–4.
2. Chapter 9 (without the VC calculation).
3. Chapters 5–6 (without proofs).
4. Chapter 10.
5. Chapters 7, 11 (without proofs).
6. Chapters 12, 13 (with some of the easier proofs).
7. Chapter 14 (with some of the easier proofs).
8. Chapter 15.
9. Chapter 16.
10. Chapter 18.

11. Chapter 22.
12. Chapter 23 (without proofs for compressed sensing).
13. Chapter 24.
14. Chapter 25.

A 14 Week Advanced Course for Graduate Students:

1. Chapters 26, 27.
2. (continued)
3. Chapters 6, 28.
4. Chapter 7.
5. Chapter 31.
6. Chapter 30.
7. Chapters 12, 13.
8. Chapter 14.
9. Chapter 8.
10. Chapter 17.
11. Chapter 29.
12. Chapter 19.
13. Chapter 20.
14. Chapter 21.

1.6 NOTATION

Most of the notation we use throughout the book is either standard or defined on the spot. In this section we describe our main conventions and provide a table summarizing our notation (Table 1.1). The reader is encouraged to skip this section and return to it if during the reading of the book some notation is unclear.

We denote scalars and abstract objects with lowercase letters (e.g. x and λ). Often, we would like to emphasize that some object is a vector and then we use boldface letters (e.g. $\mathbf{x}$ and $\boldsymbol{\lambda}$). The ith element of a vector $\mathbf{x}$ is denoted by x_i. We use uppercase letters to denote matrices, sets, and sequences. The meaning should be clear from the context. As we will see momentarily, the input of a learning algorithm is a sequence of training examples. We denote by z an abstract example and by $S = z_1, \ldots, z_m$ a sequence of m examples. Historically, S is often referred to as a training *set*; however, we will always assume that S is a *sequence* rather than a set. A sequence of m vectors is denoted by $\mathbf{x}_1, \ldots, \mathbf{x}_m$. The ith element of $\mathbf{x}_t$ is denoted by $x_{t,i}$.

Throughout the book, we make use of basic notions from probability. We denote by $\mathcal{D}$ a distribution over some set,[2] for example, Z. We use the notation $z \sim \mathcal{D}$ to denote that z is sampled according to $\mathcal{D}$. Given a random variable $f : Z \to \mathbb{R}$, its expected value is denoted by $\mathbb{E}_{z\sim\mathcal{D}}[f(z)]$. We sometimes use the shorthand $\mathbb{E}[f]$ when the dependence on z is clear from the context. For $f : Z \to \{\text{true}, \text{false}\}$ we also use $\mathbb{P}_{z\sim\mathcal{D}}[f(z)]$ to denote $\mathcal{D}(\{z : f(z) = \text{true}\})$. In the next chapter we will also

[2] To be mathematically precise, $\mathcal{D}$ should be defined over some σ-algebra of subsets of Z. The user who is not familiar with measure theory can skip the few footnotes and remarks regarding more formal measurability definitions and assumptions.

Table 1.1. Summary of notation

symbol	meaning
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^d$	the set of d-dimensional vectors over $\mathbb{R}$
$\mathbb{R}_+$	the set of non-negative real numbers
$\mathbb{N}$	the set of natural numbers
$O, o, \Theta, \omega, \Omega, \tilde{O}$	asymptotic notation (see text)
$\mathbb{1}_{[\text{Boolean expression}]}$	indicator function (equals 1 if expression is true and 0 o.w.)
$[a]_+$	$= \max\{0, a\}$
$[n]$	the set $\{1, \ldots, n\}$ (for $n \in \mathbb{N}$)
$\mathbf{x}, \mathbf{v}, \mathbf{w}$	(column) vectors
x_i, v_i, w_i	the ith element of a vector
$\langle \mathbf{x}, \mathbf{v} \rangle$	$= \sum_{i=1}^d x_i v_i$ (inner product)
$\|\mathbf{x}\|_2$ or $\|\mathbf{x}\|$	$= \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ (the ℓ_2 norm of $\mathbf{x}$)
$\|\mathbf{x}\|_1$	$= \sum_{i=1}^d \|x_i\|$ (the ℓ_1 norm of $\mathbf{x}$)
$\|\mathbf{x}\|_\infty$	$= \max_i \|x_i\|$ (the ℓ_∞ norm of $\mathbf{x}$)
$\|\mathbf{x}\|_0$	the number of nonzero elements of $\mathbf{x}$
$A \in \mathbb{R}^{d,k}$	a $d \times k$ matrix over $\mathbb{R}$
$A^\top$	the transpose of A
$A_{i,j}$	the (i, j) element of A
$\mathbf{x}\mathbf{x}^\top$	the $d \times d$ matrix A s.t. $A_{i,j} = x_i x_j$ (where $\mathbf{x} \in \mathbb{R}^d$)
$\mathbf{x}_1, \ldots, \mathbf{x}_m$	a sequence of m vectors
$x_{i,j}$	the jth element of the ith vector in the sequence
$\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(T)}$	the values of a vector $\mathbf{w}$ during an iterative algorithm
$w_i^{(t)}$	the ith element of the vector $\mathbf{w}^{(t)}$
$\mathcal{X}$	instances domain (a set)
$\mathcal{Y}$	labels domain (a set)
Z	examples domain (a set)
$\mathcal{H}$	hypothesis class (a set)
$\ell : \mathcal{H} \times Z \to \mathbb{R}_+$	loss function
$\mathcal{D}$	a distribution over some set (usually over Z or over $\mathcal{X}$)
$\mathcal{D}(A)$	the probability of a set $A \subseteq Z$ according to $\mathcal{D}$
$z \sim \mathcal{D}$	sampling z according to $\mathcal{D}$
$S = z_1, \ldots, z_m$	a sequence of m examples
$S \sim \mathcal{D}^m$	sampling $S = z_1, \ldots, z_m$ i.i.d. according to $\mathcal{D}$
$\mathbb{P}, \mathbb{E}$	probability and expectation of a random variable
$\mathbb{P}_{z \sim \mathcal{D}}[f(z)]$	$= \mathcal{D}(\{z : f(z) = \text{true}\})$ for $f : Z \to \{\text{true}, \text{false}\}$
$\mathbb{E}_{z \sim \mathcal{D}}[f(z)]$	expectation of the random variable $f : Z \to \mathbb{R}$
$N(\boldsymbol{\mu}, C)$	Gaussian distribution with expectation $\boldsymbol{\mu}$ and covariance C
$f'(x)$	the derivative of a function $f : \mathbb{R} \to \mathbb{R}$ at x
$f''(x)$	the second derivative of a function $f : \mathbb{R} \to \mathbb{R}$ at x
$\frac{\partial f(\mathbf{w})}{\partial w_i}$	the partial derivative of a function $f : \mathbb{R}^d \to \mathbb{R}$ at $\mathbf{w}$ w.r.t. w_i
$\nabla f(\mathbf{w})$	the gradient of a function $f : \mathbb{R}^d \to \mathbb{R}$ at $\mathbf{w}$
$\partial f(\mathbf{w})$	the differential set of a function $f : \mathbb{R}^d \to \mathbb{R}$ at $\mathbf{w}$
$\min_{x \in C} f(x)$	$= \min\{f(x) : x \in C\}$ (minimal value of f over C)
$\max_{x \in C} f(x)$	$= \max\{f(x) : x \in C\}$ (maximal value of f over C)
$\operatorname{argmin}_{x \in C} f(x)$	the set $\{x \in C : f(x) = \min_{z \in C} f(z)\}$
$\operatorname{argmax}_{x \in C} f(x)$	the set $\{x \in C : f(x) = \max_{z \in C} f(z)\}$
log	the natural logarithm

introduce the notation $\mathcal{D}^m$ to denote the probability over Z^m induced by sampling $(z_1, \ldots, z_m)$ where each point z_i is sampled from $\mathcal{D}$ independently of the other points.

In general, we have made an effort to avoid asymptotic notation. However, we occasionally use it to clarify the main results. In particular, given $f : \mathbb{R} \to \mathbb{R}_+$ and $g : \mathbb{R} \to \mathbb{R}_+$ we write $f = O(g)$ if there exist $x_0, \alpha \in \mathbb{R}_+$ such that for all $x > x_0$ we have $f(x) \leq \alpha g(x)$. We write $f = o(g)$ if for every $\alpha > 0$ there exists x_0 such that for all $x > x_0$ we have $f(x) \leq \alpha g(x)$. We write $f = \Omega(g)$ if there exist $x_0, \alpha \in \mathbb{R}_+$ such that for all $x > x_0$ we have $f(x) \geq \alpha g(x)$. The notation $f = \omega(g)$ is defined analogously. The notation $f = \Theta(g)$ means that $f = O(g)$ and $g = O(f)$. Finally, the notation $f = \tilde{O}(g)$ means that there exists $k \in \mathbb{N}$ such that $f(x) = O(g(x) \log^k (g(x)))$.

The inner product between vectors $\mathbf{x}$ and $\mathbf{w}$ is denoted by $\langle \mathbf{x}, \mathbf{w} \rangle$. Whenever we do not specify the vector space we assume that it is the d-dimensional Euclidean space and then $\langle \mathbf{x}, \mathbf{w} \rangle = \sum_{i=1}^{d} x_i w_i$. The Euclidean (or ℓ_2) norm of a vector $\mathbf{w}$ is $\|\mathbf{w}\|_2 = \sqrt{\langle \mathbf{w}, \mathbf{w} \rangle}$. We omit the subscript from the ℓ_2 norm when it is clear from the context. We also use other ℓ_p norms, $\|\mathbf{w}\|_p = \left(\sum_i |w_i|^p\right)^{1/p}$, and in particular $\|\mathbf{w}\|_1 = \sum_i |w_i|$ and $\|\mathbf{w}\|_\infty = \max_i |w_i|$.

We use the notation $\min_{x \in C} f(x)$ to denote the minimum value of the set $\{f(x) : x \in C\}$. To be mathematically more precise, we should use $\inf_{x \in C} f(x)$ whenever the minimum is not achievable. However, in the context of this book the distinction between infimum and minimum is often of little interest. Hence, to simplify the presentation, we sometimes use the min notation even when inf is more adequate. An analogous remark applies to max versus sup.

2

A Gentle Start

Let us begin our mathematical analysis by showing how successful learning can be achieved in a relatively simplified setting. Imagine you have just arrived in some small Pacific island. You soon find out that papayas are a significant ingredient in the local diet. However, you have never before tasted papayas. You have to learn how to predict whether a papaya you see in the market is tasty or not. First, you need to decide which features of a papaya your prediction should be based on. On the basis of your previous experience with other fruits, you decide to use two features: the papaya's color, ranging from dark green, through orange and red to dark brown, and the papaya's softness, ranging from rock hard to mushy. Your input for figuring out your prediction rule is a sample of papayas that you have examined for color and softness and then tasted and found out whether they were tasty or not. Let us analyze this task as a demonstration of the considerations involved in learning problems.

Our first step is to describe a formal model aimed to capture such learning tasks.

2.1 A FORMAL MODEL – THE STATISTICAL LEARNING FRAMEWORK

The learner's input: In the basic statistical learning setting, the learner has access to the following:

Domain set: An arbitrary set, $\mathcal{X}$. This is the set of objects that we may wish to label. For example, in the papaya learning problem mentioned before, the domain set will be the set of all papayas. Usually, these domain points will be represented by a vector of *features* (like the papaya's color and softness). We also refer to domain points as *instances* and to $\mathcal{X}$ as instance space.

Label set: For our current discussion, we will restrict the label set to be a two-element set, usually $\{0, 1\}$ or $\{-1, +1\}$. Let $\mathcal{Y}$ denote our set of possible labels. For our papayas example, let $\mathcal{Y}$ be $\{0, 1\}$, where 1 represents being tasty and 0 stands for being not-tasty.

Training data: $S = ((x_1, y_1) \ldots (x_m, y_m))$ is a finite sequence of pairs in $\mathcal{X} \times \mathcal{Y}$: that is, a sequence of labeled domain points. This is the input that the

learner has access to (like a set of papayas that have been tasted and their color, softness, and tastiness). Such labeled examples are often called *training examples*. We sometimes also refer to S as a *training set*.[1]

The learner's output: The learner is requested to output a *prediction rule*, $h : \mathcal{X} \to \mathcal{Y}$. This function is also called a *predictor*, a *hypothesis*, or a *classifier*. The predictor can be used to predict the label of new domain points. In our papayas example, it is a rule that our learner will employ to predict whether future papayas he examines in the farmers' market are going to be tasty or not. We use the notation $A(S)$ to denote the hypothesis that a learning algorithm, A, returns upon receiving the training sequence S.

A simple data-generation model We now explain how the training data is generated. First, we assume that the instances (the papayas we encounter) are generated by some probability distribution (in this case, representing the environment). Let us denote that probability distribution over $\mathcal{X}$ by $\mathcal{D}$. It is important to note that we do not assume that the learner knows anything about this distribution. For the type of learning tasks we discuss, this could be any arbitrary probability distribution. As to the labels, in the current discussion we assume that there is some "correct" labeling function, $f : \mathcal{X} \to \mathcal{Y}$, and that $y_i = f(x_i)$ for all i. This assumption will be relaxed in the next chapter. The labeling function is unknown to the learner. In fact, this is just what the learner is trying to figure out. In summary, each pair in the training data S is generated by first sampling a point x_i according to $\mathcal{D}$ and then labeling it by f.

Measures of success: We define the *error of a classifier* to be the probability that it does not predict the correct label on a random data point generated by the aforementioned underlying distribution. That is, the error of h is the probability to draw a random instance x, according to the distribution $\mathcal{D}$, such that $h(x)$ does not equal $f(x)$.

Formally, given a domain subset,[2] $A \subset \mathcal{X}$, the probability distribution, $\mathcal{D}$, assigns a number, $\mathcal{D}(A)$, which determines how likely it is to observe a point $x \in A$. In many cases, we refer to A as an event and express it using a function $\pi : \mathcal{X} \to \{0, 1\}$, namely, $A = \{x \in \mathcal{X} : \pi(x) = 1\}$. In that case, we also use the notation $\mathbb{P}_{x \sim \mathcal{D}}[\pi(x)]$ to express $\mathcal{D}(A)$.

We define the error of a prediction rule, $h : \mathcal{X} \to \mathcal{Y}$, to be

$$L_{\mathcal{D},f}(h) \stackrel{\text{def}}{=} \mathop{\mathbb{P}}_{x \sim \mathcal{D}}[h(x) \neq f(x)] \stackrel{\text{def}}{=} \mathcal{D}(\{x : h(x) \neq f(x)\}). \tag{2.1}$$

That is, the error of such h is the probability of randomly choosing an example x for which $h(x) \neq f(x)$. The subscript $(\mathcal{D}, f)$ indicates that the error is measured with respect to the probability distribution $\mathcal{D}$ and the correct labeling function f. We omit this subscript when it is clear from the context. $L_{(\mathcal{D},f)}(h)$ has several synonymous names such as the *generalization error*, the *risk*, or the *true error* of h, and we will use these names interchangeably throughout

[1] Despite the "set" notation, S is a sequence. In particular, the same example may appear twice in S and some algorithms can take into account the order of examples in S.

[2] Strictly speaking, we should be more careful and require that A is a member of some σ-algebra of subsets of $\mathcal{X}$, over which $\mathcal{D}$ is defined. We will formally define our measurability assumptions in the next chapter.

the book. We use the letter L for the error, since we view this error as the *loss* of the learner. We will later also discuss other possible formulations of such loss.

A note about the information available to the learner The learner is blind to the underlying distribution $\mathcal{D}$ over the world and to the labeling function f. In our papayas example, we have just arrived in a new island and we have no clue as to how papayas are distributed and how to predict their tastiness. The only way the learner can interact with the environment is through observing the training set.

In the next section we describe a simple learning paradigm for the preceding setup and analyze its performance.

2.2 EMPIRICAL RISK MINIMIZATION

As mentioned earlier, a learning algorithm receives as input a training set S, sampled from an unknown distribution $\mathcal{D}$ and labeled by some target function f, and should output a predictor $h_S : \mathcal{X} \to \mathcal{Y}$ (the subscript S emphasizes the fact that the output predictor depends on S). The goal of the algorithm is to find h_S that minimizes the error with respect to the unknown $\mathcal{D}$ and f.

Since the learner does not know what $\mathcal{D}$ and f are, the true error is not directly available to the learner. A useful notion of error that can be calculated by the learner is the *training error* – the error the classifier incurs over the training sample:

$$L_S(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m}, \tag{2.2}$$

where $[m] = \{1, \ldots, m\}$.

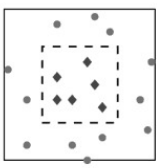
The terms *empirical error* and *empirical risk* are often used interchangeably for this error.

Since the training sample is the snapshot of the world that is available to the learner, it makes sense to search for a solution that works well on that data. This learning paradigm – coming up with a predictor h that minimizes $L_S(h)$ – is called *Empirical Risk Minimization* or ERM for short.

2.2.1 Something May Go Wrong – Overfitting

Although the ERM rule seems very natural, without being careful, this approach may fail miserably.

To demonstrate such a failure, let us go back to the problem of learning to predict the taste of a papaya on the basis of its softness and color. Consider a sample as depicted in the following:



Assume that the probability distribution $\mathcal{D}$ is such that instances are distributed uniformly within the larger square and the labeling function, f, determines the label to be 1 if the instance is within the inner square, and 0 otherwise. The area of the larger square in the picture is 2 and the area of the inner square is 1. Consider the following predictor:

$$h_S(x) = \begin{cases} y_i & \text{if } \exists i \in [m] \text{ s.t. } x_i = x \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

While this predictor might seem rather artificial, in Exercise 2.1 we show a natural representation of it using polynomials. Clearly, no matter what the sample is, $L_S(h_S) = 0$, and therefore this predictor may be chosen by an ERM algorithm (it is one of the empirical-minimum-cost hypotheses; no classifier can have smaller error). On the other hand, the true error of any classifier that predicts the label 1 only on a finite number of instances is, in this case, $1/2$. Thus, $L_{\mathcal{D}}(h_S) = 1/2$. We have found a predictor whose performance on the training set is excellent, yet its performance on the true "world" is very poor. This phenomenon is called *overfitting*. Intuitively, overfitting occurs when our hypothesis fits the training data "too well" (perhaps like the everyday experience that a person who provides a perfect detailed explanation for each of his single actions may raise suspicion).

2.3 EMPIRICAL RISK MINIMIZATION WITH INDUCTIVE BIAS

We have just demonstrated that the ERM rule might lead to overfitting. Rather than giving up on the ERM paradigm, we will look for ways to rectify it. We will search for conditions under which there is a guarantee that ERM does not overfit, namely, conditions under which when the ERM predictor has good performance with respect to the training data, it is also highly likely to perform well over the underlying data distribution.

A common solution is to apply the ERM learning rule over a restricted search space. Formally, the learner should choose in advance (before seeing the data) a set of predictors. This set is called a *hypothesis class* and is denoted by $\mathcal{H}$. Each $h \in \mathcal{H}$ is a function mapping from $\mathcal{X}$ to $\mathcal{Y}$. For a given class $\mathcal{H}$, and a training sample, S, the $\text{ERM}_{\mathcal{H}}$ learner uses the ERM rule to choose a predictor $h \in \mathcal{H}$, with the lowest possible error over S. Formally,

$$\text{ERM}_{\mathcal{H}}(S) \in \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h),$$

where argmin stands for the set of hypotheses in $\mathcal{H}$ that achieve the minimum value of $L_S(h)$ over $\mathcal{H}$. By restricting the learner to choosing a predictor from $\mathcal{H}$, we *bias* it toward a particular set of predictors. Such restrictions are often called an *inductive bias*. Since the choice of such a restriction is determined before the learner sees the training data, it should ideally be based on some prior knowledge about the problem to be learned. For example, for the papaya taste prediction problem we may choose the class $\mathcal{H}$ to be the set of predictors that are determined by axis aligned rectangles (in the space determined by the color and softness coordinates). We will later show that $\text{ERM}_{\mathcal{H}}$ over this class is guaranteed not to overfit. On the other hand, the example of overfitting that we have seen previously, demonstrates that choosing $\mathcal{H}$

to be a class of predictors that includes all functions that assign the value 1 to a finite set of domain points does not suffice to guarantee that $\text{ERM}_{\mathcal{H}}$ will not overfit.

A fundamental question in learning theory is, over which hypothesis classes $\text{ERM}_{\mathcal{H}}$ learning will not result in overfitting. We will study this question later in the book.

Intuitively, choosing a more restricted hypothesis class better protects us against overfitting but at the same time might cause us a stronger inductive bias. We will get back to this fundamental tradeoff later.

2.3.1 Finite Hypothesis Classes

The simplest type of restriction on a class is imposing an upper bound on its size (that is, the number of predictors h in $\mathcal{H}$). In this section, we show that if $\mathcal{H}$ is a finite class then $\text{ERM}_{\mathcal{H}}$ will not overfit, provided it is based on a sufficiently large training sample (this size requirement will depend on the size of $\mathcal{H}$).

Limiting the learner to prediction rules within some finite hypothesis class may be considered as a reasonably mild restriction. For example, $\mathcal{H}$ can be the set of all predictors that can be implemented by a C++ program written in at most 10^9 bits of code. In our papayas example, we mentioned previously the class of axis aligned rectangles. While this is an infinite class, if we discretize the representation of real numbers, say, by using a 64 bits floating-point representation, the hypothesis class becomes a finite class.

Let us now analyze the performance of the $\text{ERM}_{\mathcal{H}}$ learning rule assuming that $\mathcal{H}$ is a finite class. For a training sample, S, labeled according to some $f : \mathcal{X} \to \mathcal{Y}$, let h_S denote a result of applying ERM_H to S, namely,

$$h_S \in \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h). \tag{2.4}$$

In this chapter, we make the following simplifying assumption (which will be relaxed in the next chapter).

Definition 2.1 (The Realizability Assumption). There exists $h^\star \in \mathcal{H}$ s.t. $L_{(\mathcal{D},f)}(h^\star) = 0$. Note that this assumption implies that with probability 1 over random samples, S, where the instances of S are sampled according to $\mathcal{D}$ and are labeled by f, we have $L_S(h^\star) = 0$.

The realizability assumption implies that for every ERM hypothesis we have that[3] $L_S(h_S) = 0$. However, we are interested in the *true* risk of h_S, $L_{(\mathcal{D},f)}(h_S)$, rather than its empirical risk.

Clearly, any guarantee on the error with respect to the underlying distribution, $\mathcal{D}$, for an algorithm that has access only to a sample S should depend on the relationship between $\mathcal{D}$ and S. The common assumption in statistical machine learning is that the training sample S is generated by sampling points from the distribution $\mathcal{D}$ independently of each other. Formally,

[3] Mathematically speaking, this holds with probability 1. To simplify the presentation, we sometimes omit the "with probability 1" specifier.


image
not
available

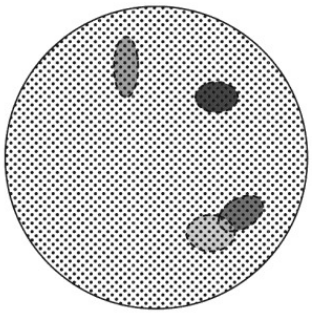


Figure 2.1. Each point in the large circle represents a possible m-tuple of instances. Each colored oval represents the set of "misleading" m-tuple of instances for some "bad" predictor $h \in \mathcal{H}_B$. The ERM can potentially overfit whenever it gets a misleading training set S. That is, for some $h \in \mathcal{H}_B$ we have $L_S(h) = 0$. Equation (2.9) guarantees that for each individual bad hypothesis, $h \in \mathcal{H}_B$, at most $(1-\epsilon)^m$-fraction of the training sets would be misleading. In particular, the larger m is, the smaller each of these colored ovals becomes. The union bound formalizes the fact that the area representing the training sets that are misleading with respect to some $h \in \mathcal{H}_B$ (that is, the training sets in M) is at most the sum of the areas of the colored ovals. Therefore, it is bounded by $|\mathcal{H}_B|$ times the maximum size of a colored oval. Any sample S outside the colored ovals cannot cause the ERM rule to overfit.

Then, for any labeling function, f, and for any distribution, $\mathcal{D}$, for which the realizability assumption holds (that is, for some $h \in \mathcal{H}$, $L_{(\mathcal{D},f)}(h) = 0$), with probability of at least $1-\delta$ over the choice of an i.i.d. sample S of size m, we have that for every ERM hypothesis, h_S, it holds that

$$L_{(\mathcal{D},f)}(h_S) \leq \epsilon .$$

The preceeding corollary tells us that for a sufficiently large m, the $\text{ERM}_{\mathcal{H}}$ rule over a finite hypothesis class will be *probably* (with confidence $1-\delta$) *approximately* (up to an error of ϵ) correct. In the next chapter we formally define the model of Probably Approximately Correct (PAC) learning.

2.4 EXERCISES

2.1 **Overfitting of polynomial matching:** We have shown that the predictor defined in Equation (2.3) leads to overfitting. While this predictor seems to be very unnatural, the goal of this exercise is to show that it can be described as a thresholded polynomial. That is, show that given a training set $S = \{(\mathbf{x}_i, f(\mathbf{x}_i))\}_{i=1}^m \subseteq (\mathbb{R}^d \times \{0,1\})^m$, there exists a polynomial p_S such that $h_S(\mathbf{x}) = 1$ if and only if $p_S(\mathbf{x}) \geq 0$, where h_S is as defined in Equation (2.3). It follows that learning the class of all thresholded polynomials using the ERM rule may lead to overfitting.

2.2 Let $\mathcal{H}$ be a class of binary classifiers over a domain $\mathcal{X}$. Let $\mathcal{D}$ be an unknown distribution over $\mathcal{X}$, and let f be the target hypothesis in $\mathcal{H}$. Fix some $h \in \mathcal{H}$. Show that the expected value of $L_S(h)$ over the choice of $S|_x$ equals $L_{(\mathcal{D},f)}(h)$, namely,

$$\mathop{\mathbb{E}}_{S|_x \sim \mathcal{D}^m} [L_S(h)] = L_{(\mathcal{D},f)}(h).$$

2.3 **Axis aligned rectangles:** An axis aligned rectangle classifier in the plane is a classifier that assigns the value 1 to a point if and only if it is inside a certain rectangle.

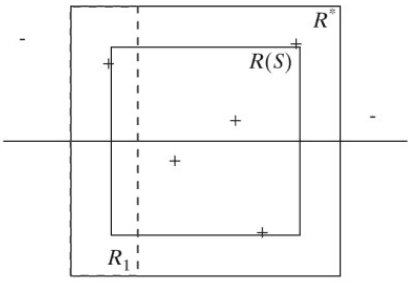


Figure 2.2. Axis aligned rectangles.

Formally, given real numbers $a_1 \leq b_1, a_2 \leq b_2$, define the classifier $h_{(a_1,b_1,a_2,b_2)}$ by

$$h_{(a_1,b_1,a_2,b_2)}(x_1,x_2) = \begin{cases} 1 & \text{if } a_1 \leq x_1 \leq b_1 \text{ and } a_2 \leq x_2 \leq b_2 \\ 0 & \text{otherwise} \end{cases}. \tag{2.10}$$

The class of all axis aligned rectangles in the plane is defined as

$$\mathcal{H}^2_{\text{rec}} = \{h_{(a_1,b_1,a_2,b_2)} : a_1 \leq b_1, \text{ and } a_2 \leq b_2\}.$$

Note that this is an infinite size hypothesis class. Throughout this exercise we rely on the realizability assumption.

1. Let A be the algorithm that returns the smallest rectangle enclosing all positive examples in the training set. Show that A is an ERM.
2. Show that if A receives a training set of size $\geq \frac{4\log(4/\delta)}{\epsilon}$ then, with probability of at least $1-\delta$ it returns a hypothesis with error of at most ϵ.
   Hint: Fix some distribution $\mathcal{D}$ over $\mathcal{X}$, let $R^* = R(a_1^*, b_1^*, a_2^*, b_2^*)$ be the rectangle that generates the labels, and let f be the corresponding hypothesis. Let $a_1 \geq a_1^*$ be a number such that the probability mass (with respect to $\mathcal{D}$) of the rectangle $R_1 = R(a_1^*, a_1, a_2^*, b_2^*)$ is exactly $\epsilon/4$. Similarly, let b_1, a_2, b_2 be numbers such that the probability masses of the rectangles $R_2 = R(b_1, b_1^*, a_2^*, b_2^*)$, $R_3 = R(a_1^*, b_1^*, a_2^*, a_2)$, $R_4 = R(a_1^*, b_1^*, b_2, b_2^*)$ are all exactly $\epsilon/4$. Let $R(S)$ be the rectangle returned by A. See illustration in Figure 2.2.
   - Show that $R(S) \subseteq R^*$.
   - Show that if S contains (positive) examples in all of the rectangles R_1, R_2, R_3, R_4, then the hypothesis returned by A has error of at most ϵ.
   - For each $i \in \{1, \ldots, 4\}$, upper bound the probability that S does not contain an example from R_i.
   - Use the union bound to conclude the argument.
3. Repeat the previous question for the class of axis aligned rectangles in $\mathbb{R}^d$.
4. Show that the runtime of applying the algorithm A mentioned earlier is polynomial in $d, 1/\epsilon$, and in $\log(1/\delta)$.

3

A Formal Learning Model

In this chapter we define our main formal learning model – the PAC learning model and its extensions. We will consider other notions of learnability in Chapter 7.

3.1 PAC LEARNING

In the previous chapter we have shown that for a finite hypothesis class, if the ERM rule with respect to that class is applied on a sufficiently large training sample (whose size is independent of the underlying distribution or labeling function) then the output hypothesis will be probably approximately correct. More generally, we now define*Probably Approximately Correct* (PAC) learning.

Definition 3.1 (PAC Learnability). A hypothesis class $\mathcal{H}$ is PAC learnable if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$, for every distribution $\mathcal{D}$ over $\mathcal{X}$, and for every labeling function $f : \mathcal{X} \to \{0,1\}$, if the realizable assumption holds with respect to $\mathcal{H}, \mathcal{D}, f$, then when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$ and labeled by f, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the examples), $L_{(\mathcal{D},f)}(h) \leq \epsilon$.

The definition of Probably Approximately Correct learnability contains two approximation parameters. The accuracy parameter ϵ determines how far the output classifier can be from the optimal one (this corresponds to the "approximately correct"), and a confidence parameter δ indicating how likely the classifier is to meet that accuracy requirement (corresponds to the "probably" part of "PAC"). Under the data access model that we are investigating, these approximations are inevitable. Since the training set is randomly generated, there may always be a small chance that it will happen to be noninformative (for example, there is always some chance that the training set will contain only one domain point, sampled over and over again). Furthermore, even when we are lucky enough to get a training sample that does faithfully represent $\mathcal{D}$, because it is just a finite sample, there may always be some fine details of $\mathcal{D}$ that it fails to reflect. Our accuracy parameter, ϵ, allows "forgiving" the learner's classifier for making minor errors.

Sample Complexity

The function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ determines the *sample complexity* of learning $\mathcal{H}$: that is, how many examples are required to guarantee a probably approximately correct solution. The sample complexity is a function of the accuracy (ϵ) and confidence (δ) parameters. It also depends on properties of the hypothesis class $\mathcal{H}$ – for example, for a finite class we showed that the sample complexity depends on log the size of $\mathcal{H}$.

Note that if $\mathcal{H}$ is PAC learnable, there are many functions $m_{\mathcal{H}}$ that satisfy the requirements given in the definition of PAC learnability. Therefore, to be precise, we will define the sample complexity of learning $\mathcal{H}$ to be the "minimal function," in the sense that for any ϵ, δ, $m_{\mathcal{H}}(\epsilon, \delta)$ is the minimal integer that satisfies the requirements of PAC learning with accuracy ϵ and confidence δ.

Let us now recall the conclusion of the analysis of finite hypothesis classes from the previous chapter. It can be rephrased as stating:

Corollary 3.2. *Every finite hypothesis class is PAC learnable with sample complexity*

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{\log(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil .$$

There are infinite classes that are learnable as well (see, for example, Exercise 3.3). Later on we will show that what determines the PAC learnability of a class is not its finiteness but rather a combinatorial measure called the *VC dimension*.

3.2 A MORE GENERAL LEARNING MODEL

The model we have just described can be readily generalized, so that it can be made relevant to a wider scope of learning tasks. We consider generalizations in two aspects:

Removing the Realizability Assumption

We have required that the learning algorithm succeeds on a pair of data distribution $\mathcal{D}$ and labeling function f provided that the realizability assumption is met. For practical learning tasks, this assumption may be too strong (can we really guarantee that there is a rectangle in the color-hardness space that *fully determines* which papayas are tasty?). In the next subsection, we will describe the *agnostic PAC* model in which this realizability assumption is waived.

Learning Problems beyond Binary Classification

The learning task that we have been discussing so far has to do with predicting a binary label to a given example (like being tasty or not). However, many learning tasks take a different form. For example, one may wish to predict a real valued number (say, the temperature at 9:00 p.m. tomorrow) or a label picked from a finite set of labels (like the topic of the main story in tomorrow's paper). It turns out that our analysis of learning can be readily extended to such and many other scenarios by allowing a variety of loss functions. We shall discuss that in Section 3.2.2 later.

3.2.1 Releasing the Realizability Assumption – Agnostic PAC Learning

A More Realistic Model for the Data-Generating Distribution

Recall that the realizability assumption requires that there exists $h^\star \in \mathcal{H}$ such that $\mathbb{P}_{x\sim\mathcal{D}}[h^\star(x) = f(x)] = 1$. In many practical problems this assumption does not hold. Furthermore, it is maybe more realistic not to assume that the labels are fully determined by the features we measure on input elements (in the case of the papayas, it is plausible that two papayas of the same color and softness will have different taste). In the following, we relax the realizability assumption by replacing the "target labeling function" with a more flexible notion, a data-labels generating distribution.

Formally, from now on, let $\mathcal{D}$ be a probability distribution over $\mathcal{X} \times \mathcal{Y}$, where, as before, $\mathcal{X}$ is our domain set and $\mathcal{Y}$ is a set of labels (usually we will consider $\mathcal{Y} = \{0, 1\}$). That is, $\mathcal{D}$ is a *joint distribution* over domain points and labels. One can view such a distribution as being composed of two parts: a distribution $\mathcal{D}_x$ over unlabeled domain points (sometimes called the *marginal distribution*) and a *conditional* probability over labels for each domain point, $\mathcal{D}((x, y)|x)$. In the papaya example, $\mathcal{D}_x$ determines the probability of encountering a papaya whose color and hardness fall in some color-hardness values domain, and the conditional probability is the probability that a papaya with color and hardness represented by x is tasty. Indeed, such modeling allows for two papayas that share the same color and hardness to belong to different taste categories.

The empirical and the True Error Revised

For a probability distribution, $\mathcal{D}$, over $\mathcal{X} \times \mathcal{Y}$, one can measure how likely h is to make an error when labeled points are randomly drawn according to $\mathcal{D}$. We redefine the true error (or risk) of a prediction rule h to be

$$L_{\mathcal{D}}(h) \stackrel{\text{def}}{=} \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] \stackrel{\text{def}}{=} \mathcal{D}(\{(x, y) : h(x) \neq y\}). \tag{3.1}$$

We would like to find a predictor, h, for which that error will be minimized. However, the learner does not know the data generating $\mathcal{D}$. What the learner does have access to is the training data, S. The definition of the empirical risk remains the same as before, namely,

$$L_S(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m}.$$

Given S, a learner can compute $L_S(h)$ for any function $h : X \to \{0, 1\}$. Note that $L_S(h) = L_{D(\text{uniform over } S)}(h)$.

The Goal

We wish to find some hypothesis, $h : \mathcal{X} \to \mathcal{Y}$, that (probably approximately) minimizes the true risk, $L_D(h)$.

That is, we consider the expectation of the loss of h over objects z picked randomly according to $\mathcal{D}$. Similarly, we define the *empirical risk* to be the expected loss over a given sample $S = (z_1, \ldots, z_m) \in Z^m$, namely,

$$L_S(h) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} \ell(h, z_i). \tag{3.4}$$

The loss functions used in the preceding examples of classification and regression tasks are as follows:

- **0–1 loss:** Here, our random variable z ranges over the set of pairs $\mathcal{X} \times \mathcal{Y}$ and the loss function is

$$\ell_{0-1}(h, (x, y)) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{if } h(x) \neq y \end{cases}$$

  This loss function is used in binary or multiclass classification problems.
  One should note that, for a random variable, α, taking the values $\{0, 1\}$, $\mathbb{E}_{\alpha \sim D}[\alpha] = \mathbb{P}_{\alpha \sim D}[\alpha = 1]$. Consequently, for this loss function, the definitions of $L_{\mathcal{D}}(h)$ given in Equation (3.3) and Equation (3.1) coincide.
- **Square Loss:** Here, our random variable z ranges over the set of pairs $\mathcal{X} \times \mathcal{Y}$ and the loss function is

$$\ell_{\text{sq}}(h, (x, y)) \stackrel{\text{def}}{=} (h(x) - y)^2.$$

  This loss function is used in regression problems.

We will later see more examples of useful instantiations of loss functions.

To summarize, we formally define agnostic PAC learnability for general loss functions.

Definition 3.4 (Agnostic PAC Learnability for General Loss Functions). A hypothesis class $\mathcal{H}$ is agnostic PAC learnable with respect to a set Z and a loss function $\ell : \mathcal{H} \times Z \to \mathbb{R}_+$, if there exist a function $m_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0, 1)$ and for every distribution $\mathcal{D}$ over Z, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$, the algorithm returns $h \in \mathcal{H}$ such that, with probability of at least $1 - \delta$ (over the choice of the m training examples),

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon,$$

where $L_{\mathcal{D}}(h) = \mathbb{E}_{z \sim \mathcal{D}}[\ell(h, z)]$.

Remark 3.1 (A Note About Measurability*). In the aforementioned definition, for every $h \in \mathcal{H}$, we view the function $\ell(h, \cdot) : Z \to \mathbb{R}_+$ as a random variable and define $L_{\mathcal{D}}(h)$ to be the expected value of this random variable. For that, we need to require that the function $\ell(h, \cdot)$ is measurable. Formally, we assume that there is a σ-algebra of subsets of Z, over which the probability $\mathcal{D}$ is defined, and that the preimage of every initial segment in $\mathbb{R}_+$ is in this σ-algebra. In the specific case of binary classification with the 0–1 loss, the σ-algebra is over $\mathcal{X} \times \{0, 1\}$ and our assumption on ℓ is equivalent to the assumption that for every h, the set $\{(x, h(x)) : x \in \mathcal{X}\}$ is in the σ-algebra.

Remark 3.2 (Proper versus Representation-Independent Learning*). In the preceding definition, we required that the algorithm will return a hypothesis from $\mathcal{H}$. In some situations, $\mathcal{H}$ is a subset of a set $\mathcal{H}'$, and the loss function can be naturally extended to be a function from $\mathcal{H}' \times Z$ to the reals. In this case, we may allow the algorithm to return a hypothesis $h' \in \mathcal{H}'$, as long as it satisfies the requirement $L_{\mathcal{D}}(h') \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$. Allowing the algorithm to output a hypothesis from $\mathcal{H}'$ is called *representation independent* learning, while proper learning occurs when the algorithm must output a hypothesis from $\mathcal{H}$. Representation independent learning is sometimes called "improper learning," although there is nothing improper in representation independent learning.

3.3 SUMMARY

In this chapter we defined our main formal learning model – PAC learning. The basic model relies on the realizability assumption, while the agnostic variant does not impose any restrictions on the underlying distribution over the examples. We also generalized the PAC model to arbitrary loss functions. We will sometimes refer to the most general model simply as PAC learning, omitting the "agnostic" prefix and letting the reader infer what the underlying loss function is from the context. When we would like to emphasize that we are dealing with the original PAC setting we mention that the realizability assumption holds. In Chapter 7 we will discuss other notions of learnability.

3.4 BIBLIOGRAPHIC REMARKS

Our most general definition of agnostic PAC learning with general loss functions follows the works of Vladimir Vapnik and Alexey Chervonenkis (Vapnik and Chervonenkis 1971). In particular, we follow Vapnik's general setting of learning (Vapnik 1982, Vapnik 1992, Vapnik 1995, Vapnik 1998).

PAC learning was introduced by Valiant (1984). Valiant was named the winner of the 2010 Turing Award for the introduction of the PAC model. Valiant's definition requires that the sample complexity will be polynomial in $1/\epsilon$ and in $1/\delta$, as well as in the representation size of hypotheses in the class (see also Kearns and Vazirani (1994)). As we will see in Chapter 6, if a problem is at all PAC learnable then the sample complexity depends polynomially on $1/\epsilon$ and $\log(1/\delta)$. Valiant's definition also requires that the *runtime* of the learning algorithm will be polynomial in these quantities. In contrast, we chose to distinguish between the statistical aspect of learning and the computational aspect of learning. We will elaborate on the computational aspect later on in Chapter 8, where we introduce the full PAC learning model of Valiant. For expository reasons, we use the term PAC learning even when we ignore the runtime aspect of learning. Finally, the formalization of agnostic PAC learning is due to Haussler (1992).

3.5 EXERCISES

3.1 **Monotonicity of Sample Complexity:** Let $\mathcal{H}$ be a hypothesis class for a binary classification task. Suppose that $\mathcal{H}$ is PAC learnable and its sample complexity is given

by $m_{\mathcal{H}}(\cdot,\cdot)$. Show that $m_{\mathcal{H}}$ is monotonically nonincreasing in each of its parameters. That is, show that given $\delta \in (0,1)$, and given $0 < \epsilon_1 \leq \epsilon_2 < 1$, we have that $m_{\mathcal{H}}(\epsilon_1, \delta) \geq m_{\mathcal{H}}(\epsilon_2, \delta)$. Similarly, show that given $\epsilon \in (0,1)$, and given $0 < \delta_1 \leq \delta_2 < 1$, we have that $m_{\mathcal{H}}(\epsilon, \delta_1) \geq m_{\mathcal{H}}(\epsilon, \delta_2)$.

3.2 Let $\mathcal{X}$ be a discrete domain, and let $\mathcal{H}_{\text{Singleton}} = \{h_z : z \in \mathcal{X}\} \cup \{h^-\}$, where for each $z \in \mathcal{X}$, h_z is the function defined by $h_z(x) = 1$ if $x = z$ and $h_z(x) = 0$ if $x \neq z$. h^- is simply the all-negative hypothesis, namely, $\forall x \in X$, $h^-(x) = 0$. The realizability assumption here implies that the true hypothesis f labels negatively all examples in the domain, perhaps except one.

1. Describe an algorithm that implements the ERM rule for learning $\mathcal{H}_{\text{Singleton}}$ in the realizable setup.
2. Show that $\mathcal{H}_{\text{Singleton}}$ is PAC learnable. Provide an upper bound on the sample complexity.

3.3 Let $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{0,1\}$, and let $\mathcal{H}$ be the class of concentric circles in the plane, that is, $\mathcal{H} = \{h_r : r \in \mathbb{R}_+\}$, where $h_r(x) = \mathbb{1}_{[\|x\| \leq r]}$. Prove that $\mathcal{H}$ is PAC learnable (assume realizability), and its sample complexity is bounded by

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{\log(1/\delta)}{\epsilon} \right\rceil .$$

3.4 In this question, we study the hypothesis class of *Boolean conjunctions* defined as follows. The instance space is $\mathcal{X} = \{0,1\}^d$ and the label set is $\mathcal{Y} = \{0,1\}$. A literal over the variables $x_1, \ldots, x_d$ is a simple Boolean function that takes the form $f(\mathbf{x}) = x_i$, for some $i \in [d]$, or $f(\mathbf{x}) = 1 - x_i$ for some $i \in [d]$. We use the notation $\bar{x}_i$ as a shorthand for $1 - x_i$. A conjunction is any product of literals. In Boolean logic, the product is denoted using the $\wedge$ sign. For example, the function $h(\mathbf{x}) = x_1 \cdot (1 - x_2)$ is written as $x_1 \wedge \bar{x}_2$.

We consider the hypothesis class of all conjunctions of literals over the d variables. The empty conjunction is interpreted as the all-positive hypothesis (namely, the function that returns $h(\mathbf{x}) = 1$ for all $\mathbf{x}$). The conjunction $x_1 \wedge \bar{x}_1$ (and similarly any conjunction involving a literal and its negation) is allowed and interpreted as the all-negative hypothesis (namely, the conjunction that returns $h(\mathbf{x}) = 0$ for all $\mathbf{x}$). We assume realizability: Namely, we assume that there exists a Boolean conjunction that generates the labels. Thus, each example $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ consists of an assignment to the d Boolean variables $x_1, \ldots, x_d$, and its truth value (0 for false and 1 for true).

For instance, let $d = 3$ and suppose that the true conjunction is $x_1 \wedge \bar{x}_2$. Then, the training set S might contain the following instances:

$$((1,1,1),0), ((1,0,1),1), ((0,1,0),0)((1,0,0),1).$$

Prove that the hypothesis class of all conjunctions over d variables is PAC learnable and bound its sample complexity. Propose an algorithm that implements the ERM rule, whose runtime is polynomial in $d \cdot m$.

3.5 Let $\mathcal{X}$ be a domain and let $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m$ be a sequence of distributions over $\mathcal{X}$. Let $\mathcal{H}$ be a finite class of binary classifiers over $\mathcal{X}$ and let $f \in \mathcal{H}$. Suppose we are getting a sample S of m examples, such that the instances are independent but are not identically distributed; the ith instance is sampled from $\mathcal{D}_i$ and then y_i is set to be $f(\mathbf{x}_i)$. Let $\bar{D}_m$ denote the average, that is, $\bar{\mathcal{D}}_m = (\mathcal{D}_1 + \cdots + \mathcal{D}_m)/m$.

Fix an accuracy parameter $\epsilon \in (0,1)$. Show that

$$\mathbb{P}\left[\exists h \in \mathcal{H} \text{ s.t. } L_{(\tilde{\mathcal{D}}_m, f)}(h) > \epsilon \text{ and } L_{(S,f)}(h) = 0\right] \leq |\mathcal{H}| e^{-\epsilon m}.$$

Hint: Use the geometric-arithmetic mean inequality.

3.6 Let $\mathcal{H}$ be a hypothesis class of binary classifiers. Show that if $\mathcal{H}$ is agnostic PAC learnable, then $\mathcal{H}$ is PAC learnable as well. Furthermore, if A is a successful agnostic PAC learner for $\mathcal{H}$, then A is also a successful PAC learner for $\mathcal{H}$.

3.7 **(*) The Bayes optimal predictor:** Show that for every probability distribution $\mathcal{D}$, the Bayes optimal predictor $f_{\mathcal{D}}$ is optimal, in the sense that for every classifier g from $\mathcal{X}$ to $\{0,1\}$, $L_{\mathcal{D}}(f_{\mathcal{D}}) \leq L_{\mathcal{D}}(g)$.

3.8 **(*)** We say that a learning algorithm A *is better than* B *with respect to* some probability distribution, $\mathcal{D}$, if

$$L_{\mathcal{D}}(A(S)) \leq L_{\mathcal{D}}(B(S))$$

for all samples $S \in (\mathcal{X} \times \{0,1\})^m$. We say that a learning algorithm A *is better than* B, if it is better than B with respect to all probability distributions $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$.

1. A probabilistic label predictor is a function that assigns to every domain point x a probability value, $h(x) \in [0,1]$, that determines the probability of predicting the label 1. That is, given such an h and an input, x, the label for x is predicted by tossing a coin with bias $h(x)$ toward Heads and predicting 1 iff the coin comes up Heads. Formally, we define a probabilistic label predictor as a function, $h : \mathcal{X} \to [0,1]$. The loss of such h on an example (x,y) is defined to be $|h(x) - y|$, which is exactly the probability that the prediction of h will not be equal to y. Note that if h is deterministic, that is, returns values in $\{0,1\}$, then $|h(x) - y| = \mathbb{1}_{[h(x) \neq y]}$. Prove that for every data-generating distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$, the Bayes optimal predictor has the smallest risk (w.r.t. the loss function $\ell(h,(x,y)) = |h(x) - y|$, among all possible label predictors, including probabilistic ones).
2. Let $\mathcal{X}$ be a domain and $\{0,1\}$ be a set of labels. Prove that for every distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$, there exist a learning algorithm $A_{\mathcal{D}}$ that is better than any other learning algorithm with respect to $\mathcal{D}$.
3. Prove that for every learning algorithm A there exist a probability distribution, $\mathcal{D}$, and a learning algorithm B such that A is not better than B w.r.t. $\mathcal{D}$.

3.9 Consider a variant of the PAC model in which there are two example oracles: one that generates positive examples and one that generates negative examples, both according to the underlying distribution $\mathcal{D}$ on $\mathcal{X}$. Formally, given a target function $f : \mathcal{X} \to \{0,1\}$, let $\mathcal{D}^+$ be the distribution over $\mathcal{X}^+ = \{x \in \mathcal{X} : f(x) = 1\}$ defined by $\mathcal{D}^+(A) = \mathcal{D}(A)/\mathcal{D}(\mathcal{X}^+)$, for every $A \subset \mathcal{X}^+$. Similarly, $\mathcal{D}^-$ is the distribution over $\mathcal{X}^-$ induced by $\mathcal{D}$.

The definition of PAC learnability in the two-oracle model is the same as the standard definition of PAC learnability except that here the learner has access to $m^+_{\mathcal{H}}(\epsilon,\delta)$ i.i.d. examples from $\mathcal{D}^+$ and $m^-(\epsilon,\delta)$ i.i.d. examples from $\mathcal{D}^-$. The learner's goal is to output h s.t. with probability at least $1-\delta$ (over the choice of the two training sets, and possibly over the nondeterministic decisions made by the learning algorithm), both $L_{(D^+,f)}(h) \leq \epsilon$ and $L_{(D^-,f)}(h) \leq \epsilon$.

1. **(*)** Show that if $\mathcal{H}$ is PAC learnable (in the standard one-oracle model), then $\mathcal{H}$ is PAC learnable in the two-oracle model.
2. **(**)** Define h^+ to be the always-plus hypothesis and h^- to be the always-minus hypothesis. Assume that $h^+, h^- \in \mathcal{H}$. Show that if $\mathcal{H}$ is PAC learnable in the two-oracle model, then $\mathcal{H}$ is PAC learnable in the standard one-oracle model.

4

Learning via Uniform Convergence

The first formal learning model that we have discussed was the PAC model. In Chapter 2 we have shown that under the realizability assumption, any finite hypothesis class is PAC learnable. In this chapter we will develop a general tool, *uniform convergence*, and apply it to show that any finite class is learnable in the agnostic PAC model with general loss functions, as long as the range loss function is bounded.

4.1 UNIFORM CONVERGENCE IS SUFFICIENT FOR LEARNABILITY

The idea behind the learning condition discussed in this chapter is very simple. Recall that, given a hypothesis class, $\mathcal{H}$, the ERM learning paradigm works as follows: Upon receiving a training sample, S, the learner evaluates the risk (or error) of each h in $\mathcal{H}$ on the given sample and outputs a member of $\mathcal{H}$ that minimizes this empirical risk. The hope is that an h that minimizes the empirical risk with respect to S is a risk minimizer (or has risk close to the minimum) with respect to the true data probability distribution as well. For that, it suffices to ensure that the empirical risks of all members of $\mathcal{H}$ are good approximations of their true risk. Put another way, we need that uniformly over all hypotheses in the hypothesis class, the empirical risk will be close to the true risk, as formalized in the following.

Definition 4.1 (ϵ-representative sample). A training set S is called ϵ-representative (w.r.t. domain Z, hypothesis class $\mathcal{H}$, loss function ℓ, and distribution $\mathcal{D}$) if

$$\forall h \in \mathcal{H}, \quad |L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon.$$

The next simple lemma states that whenever the sample is $(\epsilon/2)$-representative, the ERM learning rule is guaranteed to return a good hypothesis.

Lemma 4.2. *Assume that a training set S is $\frac{\epsilon}{2}$-representative (w.r.t. domain Z, hypothesis class $\mathcal{H}$, loss function ℓ, and distribution $\mathcal{D}$). Then, any output of* $\mathrm{ERM}_{\mathcal{H}}(S)$, *namely, any $h_S \in \operatorname{argmin}_{h \in \mathcal{H}} L_S(h)$, satisfies*

$$L_{\mathcal{D}}(h_S) \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon.$$

Finally, if we choose

$$m \geq \frac{\log(2|\mathcal{H}|/\delta)}{2\epsilon^2}$$

then

$$\mathcal{D}^m(\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) \leq \delta.$$

Corollary 4.6. *Let $\mathcal{H}$ be a finite hypothesis class, let Z be a domain, and let $\ell : \mathcal{H} \times Z \to [0,1]$ be a loss function. Then, $\mathcal{H}$ enjoys the uniform convergence property with sample complexity*

$$m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq \left\lceil \frac{\log(2|\mathcal{H}|/\delta)}{2\epsilon^2} \right\rceil.$$

Furthermore, the class is agnostically PAC learnable using the ERM algorithm with sample complexity

$$m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{UC}(\epsilon/2, \delta) \leq \left\lceil \frac{2\log(2|\mathcal{H}|/\delta)}{\epsilon^2} \right\rceil.$$

Remark 4.1 (The "Discretization Trick"). While the preceding corollary only applies to finite hypothesis classes, there is a simple trick that allows us to get a very good estimate of the practical sample complexity of infinite hypothesis classes. Consider a hypothesis class that is parameterized by d parameters. For example, let $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \{\pm 1\}$, and the hypothesis class, $\mathcal{H}$, be all functions of the form $h_\theta(x) = \text{sign}(x - \theta)$. That is, each hypothesis is parameterized by one parameter, $\theta \in \mathbb{R}$, and the hypothesis outputs 1 for all instances larger than θ and outputs -1 for instances smaller than θ. This is a hypothesis class of an infinite size. However, if we are going to learn this hypothesis class in practice, using a computer, we will probably maintain real numbers using floating point representation, say, of 64 bits. It follows that in practice, our hypothesis class is parameterized by the set of scalars that can be represented using a 64 bits floating point number. There are at most 2^{64} such numbers; hence the actual size of our hypothesis class is at most 2^{64}. More generally, if our hypothesis class is parameterized by d numbers, in practice we learn a hypothesis class of size at most 2^{64d}. Applying Corollary 4.6 we obtain that the sample complexity of such classes is bounded by $\frac{128d + 2\log(2/\delta)}{\epsilon^2}$. This upper bound on the sample complexity has the deficiency of being dependent on the specific representation of real numbers used by our machine. In Chapter 6 we will introduce a rigorous way to analyze the sample complexity of infinite size hypothesis classes. Nevertheless, the discretization trick can be used to get a rough estimate of the sample complexity in many practical situations.

4.3 SUMMARY

If the uniform convergence property holds for a hypothesis class $\mathcal{H}$ then in most cases the empirical risks of hypotheses in $\mathcal{H}$ will faithfully represent their true risks. Uniform convergence suffices for agnostic PAC learnability using the ERM rule. We have shown that finite hypothesis classes enjoy the uniform convergence property and are hence agnostic PAC learnable.

4.4 BIBLIOGRAPHIC REMARKS

Classes of functions for which the uniform convergence property holds are also called Glivenko-Cantelli classes, named after Valery Ivanovich Glivenko and Francesco Paolo Cantelli, who proved the first uniform convergence result in the 1930s. See (Dudley, Gine & Zinn 1991). The relation between uniform convergence and learnability was thoroughly studied by Vapnik – see (Vapnik 1992, Vapnik 1995, Vapnik 1998). In fact, as we will see later in Chapter 6, the fundamental theorem of learning theory states that in binary classification problems, uniform convergence is not only a sufficient condition for learnability but is also a necessary condition. This is not the case for more general learning problems (see (Shalev-Shwartz, Shamir, Srebro & Sridharan 2010)).

4.5 EXERCISES

4.1 In this exercise, we show that the (ϵ, δ) requirement on the convergence of errors in our definitions of PAC learning, is, in fact, quite close to a simpler looking requirement about averages (or expectations). Prove that the following two statements are equivalent (for any learning algorithm A, any probability distribution $\mathcal{D}$, and any loss function whose range is $[0,1]$):

1. For every $\epsilon, \delta > 0$, there exists $m(\epsilon, \delta)$ such that $\forall m \geq m(\epsilon, \delta)$

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S)) > \epsilon] < \delta$$

2.

$$\lim_{m\to\infty} \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S))] = 0$$

(where $\mathbb{E}_{S\sim\mathcal{D}^m}$ denotes the expectation over samples S of size m).

4.2 **Bounded loss functions:** In Corollary 4.6 we assumed that the range of the loss function is $[0,1]$. Prove that if the range of the loss function is $[a,b]$ then the sample complexity satisfies

$$m_{\mathcal{H}}(\epsilon, \delta) \leq m^{\mathrm{UC}}_{\mathcal{H}}(\epsilon/2, \delta) \leq \left\lceil \frac{2\log(2|\mathcal{H}|/\delta)(b-a)^2}{\epsilon^2} \right\rceil .$$

5

The Bias-Complexity Trade-off

In Chapter 2 we saw that unless one is careful, the training data can mislead the learner, and result in overfitting. To overcome this problem, we restricted the search space to some hypothesis class $\mathcal{H}$. Such a hypothesis class can be viewed as reflecting some prior knowledge that the learner has about the task – a belief that one of the members of the class $\mathcal{H}$ is a low-error model for the task. For example, in our papayas taste problem, on the basis of our previous experience with other fruits, we may assume that some rectangle in the color-hardness plane predicts (at least approximately) the papaya's tastiness.

Is such prior knowledge really necessary for the success of learning? Maybe there exists some kind of universal learner, that is, a learner who has no prior knowledge about a certain task and is ready to be challenged by any task? Let us elaborate on this point. A specific learning task is defined by an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, where the goal of the learner is to find a predictor $h : \mathcal{X} \to \mathcal{Y}$, whose risk, $L_{\mathcal{D}}(h)$, is small enough. The question is therefore whether there exist a learning algorithm A and a training set size m, such that for every distribution $\mathcal{D}$, if A receives m i.i.d. examples from $\mathcal{D}$, there is a high chance it outputs a predictor h that has a low risk.

The first part of this chapter addresses this question formally. The No-Free-Lunch theorem states that no such universal learner exists. To be more precise, the theorem states that for binary classification prediction tasks, for every learner there exists a distribution on which it fails. We say that the learner fails if, upon receiving i.i.d. examples from that distribution, its output hypothesis is likely to have a large risk, say, ≥ 0.3, whereas for the same distribution, there exists another learner that will output a hypothesis with a small risk. In other words, the theorem states that no learner can succeed on all learnable tasks – every learner has tasks on which it fails while other learners succeed.

Therefore, when approaching a particular learning problem, defined by some distribution $\mathcal{D}$, we should have some prior knowledge on $\mathcal{D}$. One type of such prior knowledge is that $\mathcal{D}$ comes from some specific parametric family of distributions. We will study learning under such assumptions later on in Chapter 24. Another type of prior knowledge on $\mathcal{D}$, which we assumed when defining the PAC learning model,

is that there exists h in some predefined hypothesis class $\mathcal{H}$, such that $L_{\mathcal{D}}(h) = 0$. A softer type of prior knowledge on $\mathcal{D}$ is assuming that $\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$ is small. In a sense, this weaker assumption on $\mathcal{D}$ is a prerequisite for using the agnostic PAC model, in which we require that the risk of the output hypothesis will not be much larger than $\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$.

In the second part of this chapter we study the benefits and pitfalls of using a hypothesis class as a means of formalizing prior knowledge. We decompose the error of an ERM algorithm over a class $\mathcal{H}$ into two components. The first component reflects the quality of our prior knowledge, measured by the minimal risk of a hypothesis in our hypothesis class, $\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$. This component is also called the *approximation error*, or the *bias* of the algorithm toward choosing a hypothesis from $\mathcal{H}$. The second component is the error due to overfitting, which depends on the size or the complexity of the class $\mathcal{H}$ and is called the *estimation error*. These two terms imply a tradeoff between choosing a more complex $\mathcal{H}$ (which can decrease the bias but increases the risk of overfitting) or a less complex $\mathcal{H}$ (which might increase the bias but decreases the potential overfitting).

5.1 THE NO-FREE-LUNCH THEOREM

In this part we prove that there is no universal learner. We do this by showing that no learner can succeed on all learning tasks, as formalized in the following theorem:

Theorem 5.1. (No-Free-Lunch) *Let A be any learning algorithm for the task of binary classification with respect to the 0−1 loss over a domain $\mathcal{X}$. Let m be any number smaller than $|\mathcal{X}|/2$, representing a training set size. Then, there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ such that:*

1. *There exists a function $f : \mathcal{X} \to \{0,1\}$ with $L_{\mathcal{D}}(f) = 0$.*
2. *With probability of at least $1/7$ over the choice of $S \sim \mathcal{D}^m$ we have that $L_{\mathcal{D}}(A(S)) \geq 1/8$.*

This theorem states that for every learner, there exists a task on which it fails, even though that task can be successfully learned by another learner. Indeed, a trivial successful learner in this case would be an ERM learner with the hypothesis class $\mathcal{H} = \{f\}$, or more generally, ERM with respect to any finite hypothesis class that contains f and whose size satisfies the equation $m \geq 8\log(7|\mathcal{H}|/6)$ (see Corollary 2.3).

Proof. Let C be a subset of $\mathcal{X}$ of size $2m$. The intuition of the proof is that any learning algorithm that observes only half of the instances in C has no information on what should be the labels of the rest of the instances in C. Therefore, there exists a "reality," that is, some target function f, that would contradict the labels that $A(S)$ predicts on the unobserved instances in C.

Note that there are $T = 2^{2m}$ possible functions from C to $\{0,1\}$. Denote these functions by $f_1, \ldots, f_T$. For each such function, let $\mathcal{D}_i$ be a distribution over $C \times \{0,1\}$ defined by

$$\mathcal{D}_i(\{(x,y)\}) = \begin{cases} 1/|C| & \text{if } y = f_i(x) \\ 0 & \text{otherwise.} \end{cases}$$

That is, the probability to choose a pair (x, y) is $1/|C|$ if the label y is indeed the true label according to f_i, and the probability is 0 if $y \neq f_i(x)$. Clearly, $L_{\mathcal{D}_i}(f_i) = 0$.

We will show that for every algorithm, A, that receives a training set of m examples from $C \times \{0,1\}$ and returns a function $A(S) : C \to \{0,1\}$, it holds that

$$\max_{i \in [T]} \mathop{\mathbb{E}}_{S \sim \mathcal{D}_i^m} [L_{\mathcal{D}_i}(A(S))] \geq 1/4. \tag{5.1}$$

Clearly, this means that for every algorithm, A', that receives a training set of m examples from $\mathcal{X} \times \{0,1\}$ there exist a function $f : \mathcal{X} \to \{0,1\}$ and a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$, such that $L_{\mathcal{D}}(f) = 0$ and

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m} [L_{\mathcal{D}}(A'(S))] \geq 1/4. \tag{5.2}$$

It is easy to verify that the preceding suffices for showing that $\mathbb{P}[L_{\mathcal{D}}(A'(S)) \geq 1/8] \geq 1/7$, which is what we need to prove (see Exercise 5.1).

We now turn to proving that Equation (5.1) holds. There are $k = (2m)^m$ possible sequences of m examples from C. Denote these sequences by $S_1, \ldots, S_k$. Also, if $S_j = (x_1, \ldots, x_m)$ we denote by S_j^i the sequence containing the instances in S_j labeled by the function f_i, namely, $S_j^i = ((x_1, f_i(x_1)), \ldots, (x_m, f_i(x_m)))$. If the distribution is $\mathcal{D}_i$ then the possible training sets A can receive are $S_1^i, \ldots, S_k^i$, and all these training sets have the same probability of being sampled. Therefore,

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}_i^m} [L_{\mathcal{D}_i}(A(S))] = \frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}(A(S_j^i)). \tag{5.3}$$

Using the facts that "maximum" is larger than "average" and that "average" is larger than "minimum," we have

$$\begin{aligned} \max_{i \in [T]} \frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}(A(S_j^i)) &\geq \frac{1}{T} \sum_{i=1}^{T} \frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}(A(S_j^i)) \\ &= \frac{1}{k} \sum_{j=1}^{k} \frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)) \\ &\geq \min_{j \in [k]} \frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)). \end{aligned} \tag{5.4}$$

Next, fix some $j \in [k]$. Denote $S_j = (x_1, \ldots, x_m)$ and let $v_1, \ldots, v_p$ be the examples in C that do not appear in S_j. Clearly, $p \geq m$. Therefore, for every function $h : C \to$

think of the size of $\mathcal{H}$ as a measure of its complexity. In future chapters we will define other complexity measures of hypothesis classes.

Since our goal is to minimize the total risk, we face a tradeoff, called the *bias-complexity tradeoff*. On one hand, choosing $\mathcal{H}$ to be a very rich class decreases the approximation error but at the same time might increase the estimation error, as a rich $\mathcal{H}$ might lead to *overfitting*. On the other hand, choosing $\mathcal{H}$ to be a very small set reduces the estimation error but might increase the approximation error or, in other words, might lead to *underfitting*. Of course, a great choice for $\mathcal{H}$ is the class that contains only one classifier – the Bayes optimal classifier. But the Bayes optimal classifier depends on the underlying distribution $\mathcal{D}$, which we do not know (indeed, learning would have been unnecessary had we known $\mathcal{D}$).

Learning theory studies how rich we can make $\mathcal{H}$ while still maintaining reasonable estimation error. In many cases, empirical research focuses on designing good hypothesis classes for a certain domain. Here, "good" means classes for which the approximation error would not be excessively high. The idea is that although we are not experts and do not know how to construct the optimal classifier, we still have some prior knowledge of the specific problem at hand, which enables us to design hypothesis classes for which both the approximation error and the estimation error are not too large. Getting back to our papayas example, we do not know how exactly the color and hardness of a papaya predict its taste, but we do know that papaya is a fruit and on the basis of previous experience with other fruit we conjecture that a rectangle in the color-hardness space may be a good predictor.

5.3 SUMMARY

The No-Free-Lunch theorem states that there is no universal learner. Every learner has to be specified to some task, and use some prior knowledge about that task, in order to succeed. So far we have modeled our prior knowledge by restricting our output hypothesis to be a member of a chosen hypothesis class. When choosing this hypothesis class, we face a tradeoff, between a larger, or more complex, class that is more likely to have a small approximation error, and a more restricted class that would guarantee that the estimation error will be small. In the next chapter we will study in more detail the behavior of the estimation error. In Chapter 7 we will discuss alternative ways to express prior knowledge.

5.4 BIBLIOGRAPHIC REMARKS

(Wolpert & Macready 1997) proved several no-free-lunch theorems for optimization, but these are rather different from the theorem we prove here. The theorem we prove here is closely related to lower bounds in VC theory, as we will study in the next chapter.

5.5 EXERCISES

5.1 Prove that Equation (5.2) suffices for showing that $\mathbb{P}[L_{\mathcal{D}}(A(S)) \geq 1/8] \geq 1/7$.
Hint: Let θ be a random variable that receives values in $[0, 1]$ and whose expectation satisfies $\mathbb{E}[\theta] \geq 1/4$. Use Lemma B.1 to show that $\mathbb{P}[\theta \geq 1/8] \geq 1/7$.

5.2 Assume you are asked to design a learning algorithm to predict whether patients are going to suffer a heart attack. Relevant patient features the algorithm may have access to include blood pressure (BP), body-mass index (BMI), age (A), level of physical activity (P), and income (I).

You have to choose between two algorithms; the first picks an axis aligned rectangle in the two dimensional space spanned by the features BP and BMI and the other picks an axis aligned rectangle in the five dimensional space spanned by all the preceding features.

1. Explain the pros and cons of each choice.
2. Explain how the number of available labeled training samples will affect your choice.

5.3 Prove that if $|\mathcal{X}| \geq km$ for a positive integer $k \geq 2$, then we can replace the lower bound of 1/4 in the No-Free-Lunch theorem with $\frac{k-1}{2k} = \frac{1}{2} - \frac{1}{2k}$. Namely, let A be a learning algorithm for the task of binary classification. Let m be any number smaller than $|\mathcal{X}|/k$, representing a training set size. Then, there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ such that:

- There exists a function $f : \mathcal{X} \to \{0,1\}$ with $L_{\mathcal{D}}(f) = 0$.
- $\mathbb{E}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(A(S))] \geq \frac{1}{2} - \frac{1}{2k}$.

6

The VC-Dimension

In the previous chapter, we decomposed the error of the $\text{ERM}_{\mathcal{H}}$ rule into approximation error and estimation error. The approximation error depends on the fit of our prior knowledge (as reflected by the choice of the hypothesis class $\mathcal{H}$) to the underlying unknown distribution. In contrast, the definition of PAC learnability requires that the estimation error would be bounded uniformly over all distributions.

Our current goal is to figure out which classes $\mathcal{H}$ are PAC learnable, and to characterize exactly the sample complexity of learning a given hypothesis class. So far we have seen that finite classes are learnable, but that the class of all functions (over an infinite size domain) is not. What makes one class learnable and the other unlearnable? Can infinite-size classes be learnable, and, if so, what determines their sample complexity?

We begin the chapter by showing that infinite classes can indeed be learnable, and thus, finiteness of the hypothesis class is not a necessary condition for learnability. We then present a remarkably crisp characterization of the family of learnable classes in the setup of binary valued classification with the zero-one loss. This characterization was first discovered by Vladimir Vapnik and Alexey Chervonenkis in 1970 and relies on a combinatorial notion called the Vapnik-Chervonenkis dimension (VC-dimension). We formally define the VC-dimension, provide several examples, and then state the fundamental theorem of statistical learning theory, which integrates the concepts of learnability, VC-dimension, the ERM rule, and uniform convergence.

6.1 INFINITE-SIZE CLASSES CAN BE LEARNABLE

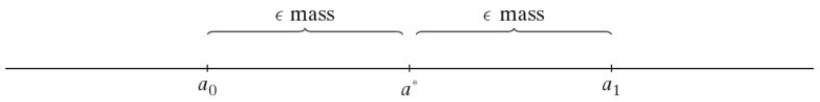
In Chapter 4 we saw that finite classes are learnable, and in fact the sample complexity of a hypothesis class is upper bounded by the log of its size. To show that the size of the hypothesis class is not the right characterization of its sample complexity, we first present a simple example of an infinite-size hypothesis class that is learnable.

Example 6.1. Let $\mathcal{H}$ be the set of threshold functions over the real line, namely, $\mathcal{H} = \{h_a : a \in \mathbb{R}\}$, where $h_a : \mathbb{R} \to \{0, 1\}$ is a function such that $h_a(x) = \mathbb{1}_{[x<a]}$. To remind the reader, $\mathbb{1}_{[x<a]}$ is 1 if $x < a$ and 0 otherwise. Clearly, $\mathcal{H}$ is of infinite size. Nevertheless, the following lemma shows that $\mathcal{H}$ is learnable in the PAC model using the ERM algorithm.

Lemma 6.1. *Let $\mathcal{H}$ be the class of thresholds as defined earlier. Then, $\mathcal{H}$ is PAC learnable, using the ERM rule, with sample complexity of $m_{\mathcal{H}}(\epsilon, \delta) \leq \lceil \log(2/\delta)/\epsilon \rceil$.*

Proof. Let $a^\star$ be a threshold such that the hypothesis $h^\star(x) = \mathbb{1}_{[x<a^\star]}$ achieves $L_{\mathcal{D}}(h^\star) = 0$. Let $\mathcal{D}_x$ be the marginal distribution over the domain $\mathcal{X}$ and let $a_0 < a^\star < a_1$ be such that

$$\mathbb{P}_{x\sim\mathcal{D}_x}[x \in (a_0, a^\star)] = \mathbb{P}_{x\sim\mathcal{D}_x}[x \in (a^\star, a_1)] = \epsilon.$$



(If $\mathcal{D}_x(-\infty, a^\star) \leq \epsilon$ we set $a_0 = -\infty$ and similarly for a_1). Given a training set S, let $b_0 = \max\{x : (x, 1) \in S\}$ and $b_1 = \min\{x : (x, 0) \in S\}$ (if no example in S is positive we set $b_0 = -\infty$ and if no example in S is negative we set $b_1 = \infty$). Let b_S be a threshold corresponding to an ERM hypothesis, h_S, which implies that $b_S \in (b_0, b_1)$. Therefore, a sufficient condition for $L_{\mathcal{D}}(h_S) \leq \epsilon$ is that both $b_0 \geq a_0$ and $b_1 \leq a_1$. In other words,

$$\mathbb{P}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(h_S) > \epsilon] \leq \mathbb{P}_{S\sim\mathcal{D}^m}[b_0 < a_0 \vee b_1 > a_1],$$

and using the union bound we can bound the preceding by

$$\mathbb{P}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(h_S) > \epsilon] \leq \mathbb{P}_{S\sim\mathcal{D}^m}[b_0 < a_0] + \mathbb{P}_{S\sim\mathcal{D}^m}[b_1 > a_1]. \tag{6.1}$$

The event $b_0 < a_0$ happens if and only if all examples in S are not in the interval (a_0, a^*), whose probability mass is defined to be ϵ, namely,

$$\mathbb{P}_{S\sim\mathcal{D}^m}[b_0 < a_0] = \mathbb{P}_{S\sim\mathcal{D}^m}[\forall (x, y) \in S,\ x \notin (a_0, a^\star)] = (1-\epsilon)^m \leq e^{-\epsilon m}.$$

Since we assume $m > \log(2/\delta)/\epsilon$ it follows that the equation is at most $\delta/2$. In the same way it is easy to see that $\mathbb{P}_{S\sim\mathcal{D}^m}[b_1 > a_1] \leq \delta/2$. Combining with Equation (6.1) we conclude our proof. □

6.2 THE VC-DIMENSION

We see, therefore, that while finiteness of $\mathcal{H}$ is a sufficient condition for learnability, it is not a necessary condition. As we will show, a property called the VC-dimension of a hypothesis class gives the correct characterization of its learnability. To motivate the definition of the VC-dimension, let us recall the No-Free-Lunch theorem (Theorem 5.1) and its proof. There, we have shown that without restricting the hypothesis class, for any learning algorithm, an adversary can construct a distribution for which the learning algorithm will perform poorly, while there is another

learning algorithm that will succeed on the same distribution. To do so, the adversary used a finite set $C \subset \mathcal{X}$ and considered a family of distributions that are concentrated on elements of C. Each distribution was derived from a "true" target function from C to $\{0,1\}$. To make any algorithm fail, the adversary used the power of choosing a target function from the set of *all* possible functions from C to $\{0,1\}$.

When considering PAC learnability of a hypothesis class $\mathcal{H}$, the adversary is restricted to constructing distributions for which some hypothesis $h \in \mathcal{H}$ achieves a zero risk. Since we are considering distributions that are concentrated on elements of C, we should study how $\mathcal{H}$ behaves on C, which leads to the following definition.

Definition 6.2 (Restriction of $\mathcal{H}$ to C). Let $\mathcal{H}$ be a class of functions from $\mathcal{X}$ to $\{0,1\}$ and let $C = \{c_1, \ldots, c_m\} \subset \mathcal{X}$. The restriction of $\mathcal{H}$ to C is the set of functions from C to $\{0,1\}$ that can be derived from $\mathcal{H}$. That is,

$$\mathcal{H}_C = \{(h(c_1), \ldots, h(c_m)) : h \in \mathcal{H}\},$$

where we represent each function from C to $\{0,1\}$ as a vector in $\{0,1\}^{|C|}$.

If the restriction of $\mathcal{H}$ to C is the set of all functions from C to $\{0,1\}$, then we say that $\mathcal{H}$ *shatters* the set C. Formally:

Definition 6.3 (Shattering). A hypothesis class $\mathcal{H}$ shatters a finite set $C \subset \mathcal{X}$ if the restriction of $\mathcal{H}$ to C is the set of all functions from C to $\{0,1\}$. That is, $|\mathcal{H}_C| = 2^{|C|}$.

Example 6.2. Let $\mathcal{H}$ be the class of threshold functions over $\mathbb{R}$. Take a set $C = \{c_1\}$. Now, if we take $a = c_1 + 1$, then we have $h_a(c_1) = 1$, and if we take $a = c_1 - 1$, then we have $h_a(c_1) = 0$. Therefore, $\mathcal{H}_C$ is the set of all functions from C to $\{0,1\}$, and $\mathcal{H}$ shatters C. Now take a set $C = \{c_1, c_2\}$, where $c_1 \leq c_2$. No $h \in \mathcal{H}$ can account for the labeling $(0,1)$, because any threshold that assigns the label 0 to c_1 must assign the label 0 to c_2 as well. Therefore not all functions from C to $\{0,1\}$ are included in $\mathcal{H}_C$; hence C is not shattered by $\mathcal{H}$.

Getting back to the construction of an adversarial distribution as in the proof of the No-Free-Lunch theorem (Theorem 5.1), we see that whenever some set C is shattered by $\mathcal{H}$, the adversary is not restricted by $\mathcal{H}$, as they can construct a distribution over C based on *any* target function from C to $\{0,1\}$, while still maintaining the realizability assumption. This immediately yields:

Corollary 6.4. *Let $\mathcal{H}$ be a hypothesis class of functions from $\mathcal{X}$ to $\{0,1\}$. Let m be a training set size. Assume that there exists a set $C \subset \mathcal{X}$ of size $2m$ that is shattered by $\mathcal{H}$. Then, for any learning algorithm, A, there exist a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ and a predictor $h \in \mathcal{H}$ such that $L_{\mathcal{D}}(h) = 0$ but with probability of at least $1/7$ over the choice of $S \sim \mathcal{D}^m$ we have that $L_{\mathcal{D}}(A(S)) \geq 1/8$.*

Corollary 6.4 tells us that if $\mathcal{H}$ shatters some set C of size $2m$ then we cannot learn $\mathcal{H}$ using m examples. Intuitively, if a set C is shattered by $\mathcal{H}$, and we receive a sample containing half the instances of C, the labels of these instances give us no information about the labels of the rest of the instances in C – every possible labeling of the rest of the instances can be explained by some hypothesis in $\mathcal{H}$. Philosophically,

6.4 THE FUNDAMENTAL THEOREM OF PAC LEARNING

We have already shown that a class of infinite VC-dimension is not learnable. The converse statement is also true, leading to the fundamental theorem of statistical learning theory:

Theorem 6.7 (The Fundamental Theorem of Statistical Learning). *Let $\mathcal{H}$ be a hypothesis class of functions from a domain $\mathcal{X}$ to $\{0,1\}$ and let the loss function be the $0-1$ loss. Then, the following are equivalent:*

1. *$\mathcal{H}$ has the uniform convergence property.*
2. *Any ERM rule is a successful agnostic PAC learner for $\mathcal{H}$.*
3. *$\mathcal{H}$ is agnostic PAC learnable.*
4. *$\mathcal{H}$ is PAC learnable.*
5. *Any ERM rule is a successful PAC learner for $\mathcal{H}$.*
6. *$\mathcal{H}$ has a finite VC-dimension.*

The proof of the theorem is given in the next section.

Not only does the VC-dimension characterize PAC learnability; it even determines the sample complexity.

Theorem 6.8 (The Fundamental Theorem of Statistical Learning – Quantitative Version). *Let $\mathcal{H}$ be a hypothesis class of functions from a domain $\mathcal{X}$ to $\{0,1\}$ and let the loss function be the $0-1$ loss. Assume that $\text{VCdim}(\mathcal{H}) = d < \infty$. Then, there are absolute constants C_1, C_2 such that*

1. *$\mathcal{H}$ has the uniform convergence property with sample complexity*

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

2. *$\mathcal{H}$ is agnostic PAC learnable with sample complexity*

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

3. *$\mathcal{H}$ is PAC learnable with sample complexity*

$$C_1 \frac{d + \log(1/\delta)}{\epsilon} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}$$

The proof of this theorem is given in Chapter 28.

Remark 6.3. We stated the fundamental theorem for binary classification tasks. A similar result holds for some other learning problems such as regression with the absolute loss or the squared loss. However, the theorem does not hold for all learning tasks. In particular, learnability is sometimes possible even though the uniform convergence property does not hold (we will see an example in Chapter 13, Exercise 6.2). Furthermore, in some situations, the ERM rule fails but learnability is possible with other learning rules.