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

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

ISBN 978-1-107-05713-5 Hardback

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

[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.

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 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.

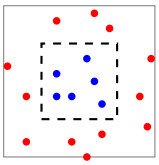
Since the training sample is the snapshot of the world that is available to the learner, it makes sense to search for a solution that works well on that data. This learning paradigm – coming up with a predictor h that minimizes $L_S(h)$ – is called *Empirical Risk Minimization* or ERM for short.

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 gray square and the labeling function, f, determines the label to be 1 if the instance is within the inner blue square, and 0 otherwise. The area of the gray square in the picture is 2 and the area of the blue 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 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,

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

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 $\mathrm{ERM}_{\mathcal{H}}$ 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 $\mathrm{ERM}_{\mathcal{H}}$ will not overfit.

A fundamental question in learning theory is, over which hypothesis classes $\mathrm{ERM}_{\mathcal{H}}$ 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 $\mathrm{ERM}_{\mathcal{H}}$ 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 $\mathrm{ERM}_{\mathcal{H}}$ 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).