5 Elements of Statistical Learning Theory

We now give a more complete exposition of the ideas of statistical learning theory, which we briefly touched on in Chapter 1. We mentioned previously that in order to learn from a small training set, we should try to *explain* the data with a model of *small* capacity; we have not yet justified *why* this is the case, however. This is the main goal of the present chapter.

Overview

We start by revisiting the difference between risk minimization and empirical risk minimization, and illustrating some common pitfalls in machine learning, such as overfitting and training on the test set (Section 5.1). We explain that the motivation for empirical risk minimization is the law of large numbers, but that the classical version of this law is not sufficient for our purposes (Section 5.2). Thus, we need to introduce the statistical notion of *consistency* (Section 5.3). It turns out that consistency of learning algorithms amounts to a law of large numbers, which holds uniformly over all functions that the learning machine can implement (Section 5.4). This crucial insight, due to Vapnik and Chervonenkis, focuses our attention on the set of attainable functions; this set must be restricted in order to have any hope of succeeding. Section 5.5 states probabilistic bounds on the risk of learning machines, and summarizes different ways of characterizing precisely how the set of functions can be restricted. This leads to the notion of *capacity concepts*, which gives us the main ingredients of the typical generalization error bound of statistical learning theory. We do not indulge in a complete treatment; rather, we try to give the main insights to provide the reader with some intuition as to how the different pieces of the puzzle fit together. We end with a section showing an example application of risk bounds for model selection (Section 5.6).

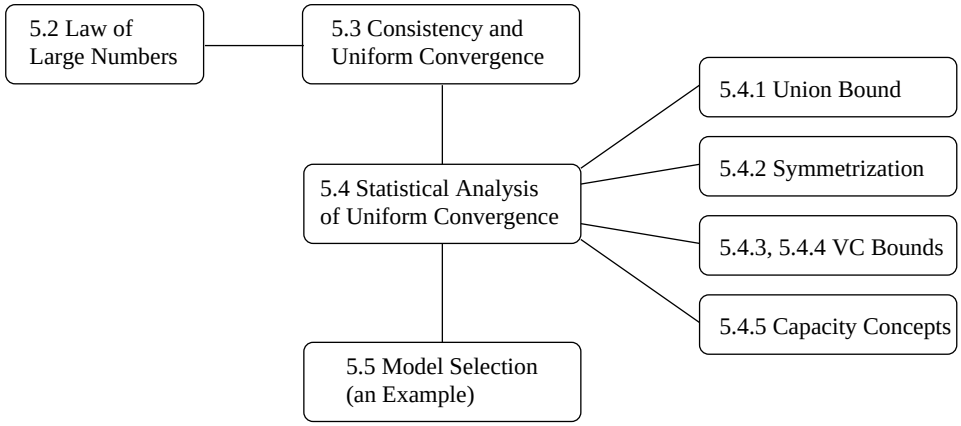
Prerequisites

The chapter attempts to present the material in a fairly non-technical manner, providing intuition wherever possible. Given the nature of the subject matter, however, a limited amount of mathematical background is required. The reader who is not familiar with basic probability theory should first read Section B.1.

5.1 Introduction

Let us start with an example. We consider a regression estimation problem. Suppose we are given empirical observations,

$$(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \mathcal{Y}, \tag{5.1}$$



Regression Example

where for simplicity we take $\mathcal{X} = \mathcal{Y} = \mathbb{R}$. Figure 5.1 shows a plot of such a dataset, along with two possible functional dependencies that could underlie the data. The dashed line represents a fairly complex model, and fits the training data perfectly. The straight line, on the other hand, does not completely "explain" the data, in the sense that there are some residual errors; it is much "simpler," however. A physicist measuring these data points would argue that it cannot be by chance that the measurements almost lie on a straight line, and would much prefer to attribute the residuals to measurement error than to an erroneous model. But is it possible to *characterize* the way in which the straight line is simpler, and why this should imply that it is, in some sense, closer to an underlying true dependency?

Bias-Variance Dilemma

In one form or another, this issue has long occupied the minds of researchers studying the problem of learning. In classical statistics, it has been studied as the *bias-variance dilemma*. If we computed a linear fit for every data set that we ever encountered, then every functional dependency we would ever "discover" would be linear. But this would not come from the data; it would be a *bias* imposed by us. If, on the other hand, we fitted a polynomial of sufficiently high degree to any given data set, we would always be able to fit the data perfectly, but the exact model we came up with would be subject to large fluctuations, depending on

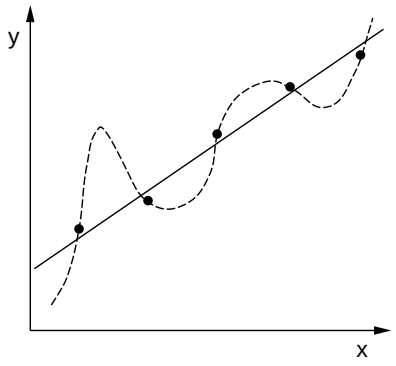


Figure 5.1 Suppose we want to estimate a functional dependence from a set of examples (black dots). Which model is preferable? The complex model perfectly fits all data points, whereas the straight line exhibits residual errors. Statistical learning theory formalizes the role of the *complexity* of the model class, and gives probabilistic guarantees for the validity of the inferred model.

how accurate our measurements were in the first place — the model would suffer from a large *variance*. A related dichotomy is the one between *estimation error* and *approximation error*. If we use a small class of functions, then even the best possible solution will poorly approximate the "true" dependency, while a large class of functions will lead to a large statistical estimation error.

Overfitting

In the terminology of applied machine learning and the design of neural networks, the complex explanation shows *overfitting*, while an overly simple explanation imposed by the learning machine design would lead to *underfitting*. A great deal of research has gone into clever engineering tricks and heuristics; these are used, for instance, to aid in the design of neural networks which will not overfit on a given data set [397]. In neural networks, overfitting can be avoided in a number of ways, such as by choosing a number of hidden units that is not too large, by *stopping* the training procedure early in order not to enforce a perfect explanation of the training set, or by using *weight decay* to limit the size of the weights, and thus of the function class implemented by the network.

Statistical learning theory provides a solid mathematical framework for studying these questions in depth. As mentioned in Chapters 1 and 3, it makes the assumption that the data are generated by sampling from an unknown underlying distribution $P(x, y)$. The learning problem then consists in minimizing the *risk* (or *expected loss* on the test data, see Definition 3.3),

Risk

$$R[f] = \int_{\mathcal{X}\times\mathcal{Y}} c(x, y, f(x))\, dP(x, y). \tag{5.2}$$

Here, c is a loss function. In the case of pattern recognition, where $\mathcal{Y} = \{\pm 1\}$, a common choice is the misclassification error, $c(x, y, f(x)) = \frac{1}{2}|f(x) - y|$.

The difficulty of the task stems from the fact that we are trying to minimize a quantity that we cannot actually evaluate: since we do not know P, we cannot compute the integral (5.2). What we *do* know, however, are the training data (5.1), which are sampled from P. We can thus try to infer a function f from the training sample that is, in some sense, *close* to the one minimizing (5.2). To this end, we need what is called an *induction principle*.

Empirical Risk

One way to proceed is to use the training sample to approximate the integral in (5.2) by a finite sum (see (B.18)). This leads to the empirical risk (Definition 3.4),

$$R_{\text{emp}}[f] = \frac{1}{m}\sum_{i=1}^{m} c(x_i, y_i, f(x_i)), \tag{5.3}$$

and the *empirical risk minimization (ERM) induction principle*, which recommends that we choose an f that minimizes (5.3).

Cast in these terms, the fundamental trade-off in learning can be stated as follows: if we allow f to be taken from a very large class of functions $\mathcal{F}$, we can always find an f that leads to a rather small value of (5.3). For instance, if we allow the use of *all* functions f mapping $\mathcal{X} \to \mathcal{Y}$ (in compact notation, $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$), then we can minimize (5.3) yet still be distant from the minimizer of (5.2). Considering a

pattern recognition problem, we could set

$$f(x) = \begin{cases} y_i & \text{if } x = x_i \text{ for some } i = 1, \ldots, m \\ 1 & \text{otherwise.} \end{cases} \tag{5.4}$$

This does not amount to any form of learning, however: suppose we are now given a test point drawn from the same distribution, $(x, y) \sim P(x, y)$. If $\mathcal{X}$ is a continuous domain, and we are not in a degenerate situation, the new pattern x will almost never be exactly equal to any of the training inputs x_i. Therefore, the learning machine will almost always predict that $y = 1$. *If we allow* all *functions from $\mathcal{X}$ to $\mathcal{Y}$, then the values of the function at points $x_1, \ldots, x_m$ carry no information about the values at other points.* In this situation, a learning machine cannot do better than chance. This insight lies at the core of the so-called *No-Free-Lunch Theorem* popularized in [608]; see also [254, 48].

The message is clear: if we make no restrictions on the class of functions from which we choose our estimate f, we cannot hope to learn anything. Consequently, machine learning research has studied various ways to implement such restrictions. In statistical learning theory, these restrictions are enforced by taking into account the *complexity* or *capacity* (measured by VC dimension, covering numbers, entropy numbers, or other concepts) of the class of functions that the learning machine can implement.[1]

In the Bayesian approach, a similar effect is achieved by placing *prior distributions $P(f)$* over the class of functions (Chapter 16). This may sound fundamentally different, but it leads to algorithms which are closely related; and on the theoretical side, recent progress has highlighted intriguing connections [92, 91, 353, 238].

5.2 The Law of Large Numbers

Let us step back and try to look at the problem from a slightly different angle. Consider the case of pattern recognition using the misclassification loss function. Given a fixed function f, then for each example, the loss $\xi_i := \frac{1}{2}|f(x_i) - y_i|$ is either

1. As an aside, note that the same problem applies to *training on the test set* (sometimes called *data snooping*): sometimes, people optimize tuning parameters of a learning machine by looking at how they change the results on an independent test set. Unfortunately, once one has adjusted the parameter in this way, the test set is not independent anymore. This is identical to the corresponding problem in training on the *training* set: once we have chosen the function to minimize the training error, the latter no longer provides an unbiased estimate of the test error. Overfitting occurs much faster on the training set, however, than it does on the test set. This is usually due to the fact that the number of tuning parameters of a learning machine is much smaller than the total number of parameters, and thus the capacity tends to be smaller. For instance, an SVM for pattern recognition typically has two tuning parameters, and optimizes m weight parameters (for a training set size of m). See also Problem 5.3 and [461].