2 Foundations of Machine Learning

Soon we will embark on a theoretical study of AdaBoost in order to understand its properties, particularly its ability as a learning algorithm to generalize, that is, to make accurate predictions on data not seen during training. Before this will be possible, however, it will be necessary to take a step back to outline our approach to the more general problem of machine learning, including some fundamental general-purpose tools that will be invaluable in our analysis of AdaBoost.

We study the basic problem of inferring from a set of training examples a classification rule whose predictions are highly accurate on freshly observed test data. On first encounter, it may seem questionable whether this kind of learning should even be possible. After all, why should there be any connection between the training and test examples, and why should it be possible to generalize from a relatively small number of training examples to a potentially vast universe of test examples? Although such objections have indeed often been the subject of philosophical debate, in this chapter we will identify an idealized but realistic model of the inference problem in which this kind of learning can be proved to be entirely feasible when certain conditions are satisfied. In particular, we will see that if we can find a *simple* rule that fits the training data well, and if the training set is not too small, then this rule will in fact generalize well, providing accurate predictions on previously unseen test examples. This is the basis of the approach presented in this chapter, and we will often use the general analysis on which it is founded to guide us in understanding how, why, and when learning is possible.

We also outline in this chapter a mathematical framework for studying machine learning, one in which a precise formulation of the boosting problem can be clearly and naturally expressed.

Note that, unlike the rest of the book, this chapter omits nearly all of the proofs of the main results since these have largely all appeared in various texts and articles. See the bibliographic notes at the end of the chapter for references.

Table 2.1
A sample dataset

Instances	1.2	2.8	8.0	3.3	5.0	4.5	7.4	5.6	3.8	6.6	6.1	1.7
Labels	−	−	+	−	−	−	+	+	−	+	+	−

Each column is a labeled example. For instance, the third example is the instance $x = 8.0$ with corresponding label $y = +1$.

2.1 A Direct Approach to Machine Learning

We start with an introduction to our approach to machine learning. This will lead to the identification of criteria that are sufficient to assure generalization, laying the intuitive groundwork for the formal treatment that follows.

2.1.1 Conditions Sufficient for Learning

As described in chapter 1, a learning algorithm takes as input a labeled sequence of training examples $(x_1, y_1), \ldots, (x_m, y_m)$, and must use these to formulate a hypothesis for classifying new instances. As before, and for most of the book, we assume that there are only two possible labels, so that each $y_i \in \{-1, +1\}$. As is customary, we routinely refer to the training examples as comprising a training *set*, or a data*set*, even though they actually form a tuple (in which the same example may appear more than once) rather than a set. Likewise for the test set.

Let us begin with an example in which the instances x_i are simply real numbers, and in which the training set looks something like the one in table 2.1. Given such a dataset, how might we formulate a prediction rule? After examining these examples, most of us would eventually notice that the larger examples, above some cutoff, are positive, and the smaller examples are negative. This would lead most people to choose a rule based on the observed cutoff behavior exhibited by this data; in other words, they would eventually choose a threshold rule that predicts all instances x above some threshold ν are $+1$, and all instances below ν are -1; that is,

$$h(x) = \begin{cases} +1 & \text{if } x \geq \nu \\ -1 & \text{otherwise} \end{cases} \tag{2.1}$$

for some choice of ν, such as 5.3, or any other value between 5.0 (the largest negative example) and 5.6 (the smallest positive example). Such a rule—which is essentially the same in form as the decision stumps used in section 1.2.3—seems so obvious and irresistibly attractive because it has the two properties that people instinctively seem to prefer: First of all, it is *consistent* with the given data, meaning that it predicts the correct labels on all of the given training examples. And second, the rule is *simple*.

This preference for simple explanations, as noted in chapter 1, is often referred to as *Occam's razor*, a central tenet, for instance, of mathematics and most areas of scientific

Table 2.2
A slightly different sample dataset

Instances	1.2	2.8	8.0	3.3	5.0	4.5	7.4	5.6	3.8	6.6	6.1	1.7
Labels	−	−	+	−	−	−	+	+	+	−	+	−

inquiry. However, unlike consistency, simplicity is not at all simple to define, and seems vague, even mystical, although most of us feel that we recognize it when we see it. The notion of simplicity is closely related to our prior expectations: We expect the data to be explainable by a simple rule, and conversely, we usually consider a rule to be simple if it matches our expectations. One of the triumphs of modern research on learning has been the development of quantitative measures of simplicity and its role in generalization, as we will see shortly.

When faced with a harder dataset, such as the slightly modified version in table 2.2, it is not so immediate how to find a simple and consistent rule. On the one hand, we might choose a simple threshold rule as before, but since none are consistent with this dataset, we will inevitably be forced to accept a small number of mistakes on the training set itself. Alternatively, we might choose a considerably more complex rule that *is* consistent, such as this one:

$$h(x) = \begin{cases} -1 & \text{if } x < 3.4 \\ +1 & \text{if } 3.4 \leq x < 4.0 \\ -1 & \text{if } 4.0 \leq x < 5.2 \\ +1 & \text{if } 5.2 \leq x < 6.3 \\ -1 & \text{if } 6.3 \leq x < 7.1 \\ +1 & \text{if } x \geq 7.1. \end{cases} \tag{2.2}$$

Thus, we face a trade-off between simplicity and fit to the data, one that will require balance and compromise.

Generally, then, we see that a natural approach to learning is to seek a hypothesis satisfying two basic criteria:

1. It fits the data well.
2. It is simple.

As indicated in the example above, however, these two criteria are often in conflict: We can typically fit the data better at the cost of choosing a more complex hypothesis, and conversely, simpler hypotheses are prone to give a poorer fit to the data. This trade-off is quite general, and is at the heart of the most central issue in the study of machine learning.

This issue must be faced at some point in the design of every learning algorithm, since we must eventually make an explicit or implicit decision about the form of the hypotheses that will be used. The form that is chosen reflects an assumption about what we expect in the data, and it is the *form* that determines simplicity or complexity. For instance, we might choose to use threshold rules of the form given in equation (2.1), or we might instead use

rules of a much freer form, as in equation (2.2). In principle, the more we know about a particular learning problem, the simpler (and more restrictive) the form of the hypotheses that can be used.

We measure how well a particular hypothesis h fits the training data by its *training* (or *empirical*) *error*, that is, the fraction of the m training examples that it misclassifies, denoted

$$\widehat{\mathrm{err}}(h) \doteq \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}\{h(x_i) \neq y_i\},$$

where $\mathbf{1}\{\cdot\}$ is an indicator function that is 1 if its argument is true, and 0 otherwise. Although we seek a simple classifier h with low training error, the ultimate goal of learning is to find a rule that is highly accurate as measured on a separate test set. We generally assume that both the training examples and the test examples are generated from the same distribution $\mathcal{D}$ on labeled pairs (x, y). With respect to this true distribution, the expected test error of a hypothesis h is called the *true* or *generalization error*; it is the same as the probability of misclassifying a single example (x, y) chosen at random from $\mathcal{D}$, and is denoted

$$\mathrm{err}(h) \doteq \mathbf{Pr}_{(x,y)\sim\mathcal{D}}[h(x) \neq y]. \tag{2.3}$$

Of course, a learning algorithm does not have the means to directly measure the generalization error $\mathrm{err}(h)$ that it aims to minimize. Instead, it must use the training error $\widehat{\mathrm{err}}(h)$ as an estimate of or proxy for the generalization error. If working with a single hypothesis h, then the training error will be a reasonable estimate of the generalization error, which it equals in expectation; in this sense, it is an *unbiased* estimator. Generally, however, a learning algorithm will work with a large space of hypotheses, and will choose the hypothesis from this space with (approximately) minimum training error. Unfortunately, the training error of a hypothesis selected in this fashion will not be unbiased, but instead will almost certainly underestimate its true error. This is because, in selecting the hypothesis with minimum training error, the algorithm favors hypotheses whose training errors are, by chance, much lower than their true errors.

To get an intuition for this effect, imagine an experiment in which each student in a class is asked to predict the outcomes of ten coin flips. Clearly, any individual student will, in expectation, correctly predict just half of the flips. However, in a class of perhaps 50 students, it is highly likely that one student will be very lucky and will correctly predict eight or even nine of the coin flips. This student will appear to possess remarkable powers of clairvoyance, when in fact it was only chance that made the student's true prediction accuracy of 50% appear to be much higher. The larger the class, the greater this effect will be: At an extreme, for a very large class of over a thousand students, we will expect there to be one student who perfectly predicts all ten coin flips.

In the learning setting, for the same reasons, it is quite likely that the apparently good performance of the best hypothesis on the training set will in part be the result of having

fitted spurious patterns that appeared in the training data entirely by chance. Inevitably, this will cause the training error to be lower than the true error for the chosen hypothesis. Moreover, the amount of this bias depends directly on the simplicity or complexity of the hypotheses considered—the more complex, and thus the less restrictive, the form of the hypotheses, the larger the space from which they are selected and the greater the bias, just as in the example above. On the other hand, we will see that the bias can be controlled when a sufficiently large training set is available.

Finding the hypothesis with minimal or nearly minimal training error can often be accomplished through an exhaustive or heuristic search, even when working with an infinitely large space of hypotheses. As an example, consider our sample dataset in table 2.2 when using threshold rules as in equation (2.1). Figure 2.1 shows a plot of the training error $\widehat{\text{err}}(h)$ of such rules as a function of the threshold ν. As can be seen from the figure, even though the set of potential classifiers of this form is uncountably infinite, a training set of m examples partitions the classifiers into $m+1$ subsets such that all of the rules in any single subset make the same predictions on the training examples, and thus have the same empirical error. (For instance, the threshold rule defined by setting the threshold ν to be 3.9 makes exactly the same predictions on all m of the examples as if we instead set ν to be 4.2.) As a result, we can find a classifier that minimizes the training error by first sorting the data according to the instance values x_i, and then computing the training error for all possible values of the threshold ν in a single scan through the sorted examples. In this case, we would find that the minimum training error is attained when ν is between 5.0 and 5.6. (More details of this method will be given in section 3.4.2.)

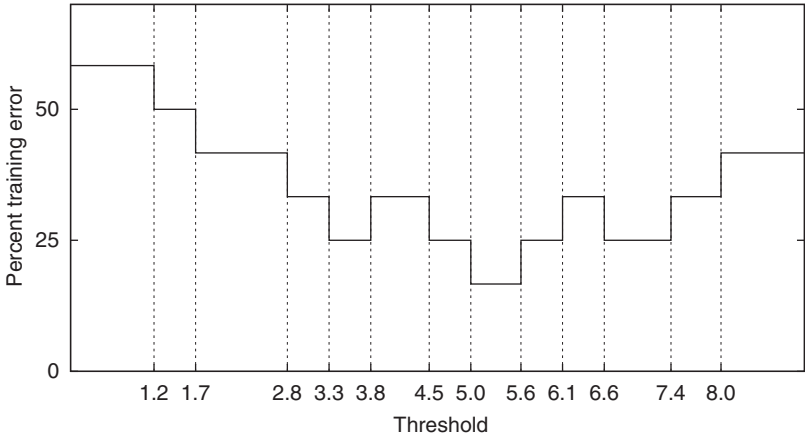

Figure 2.1
The empirical error, on the data in table 2.2, of a threshold rule of the form given in equation (2.1), plotted as a function of the threshold ν.

In general, balancing simplicity against fit to the training data is key to many practical learning algorithms. For instance, decision-tree algorithms (see section 1.3) typically grow a large tree that fits the data very well (usually too well), and then heavily prune the tree so as to limit its overall complexity as measured by its gross size.

2.1.2 Comparison to an Alternative Approach

As a brief aside, we take a moment to contrast the approach we are following with a well-studied alternative. When faced with data as in tables 2.1 and 2.2, we may in some cases have additional information available to us. For instance, maybe we know that each example i corresponds to a person where x_i represents the person's height and y_i represents the person's gender (say $+1$ means male and -1 means female); in other words, the problem is to learn to predict a person's gender from his or her height. This information leads to some natural assumptions. Specifically, we might assume that height among men is normally distributed, and likewise among women, where naturally the means μ_+ and μ_- will be different, with $\mu_+ > \mu_-$. We might further assume equal variance σ^2 for these two normal distributions, and that the two classes (genders) occur with equal probability.

These assumptions suggest a different way of proceeding. Our goal is still to classify instances whose label is unknown, but now, if we know the values of the means μ_+ and μ_-, we can easily calculate the best possible classifier, which in this case simply predicts, for a given instance (height) x, with the mean that is closest; in other words, $+1$ if $x \geq (\mu_+ + \mu_-)/2$, and -1 otherwise. (See figure 2.2.) Note that this classifier has the form of a threshold rule as in equation (2.1), although our rationale for using such a rule here is rather different. If the distribution parameters μ_+ and μ_- are unknown, they can be estimated from training data, and then used to classify test examples in the same way.

This is sometimes called a *generative* approach since we attempt to model how the data is being generated. In contrast, the approach we outlined in section 2.1.1 is often said to be *discriminative*, since the emphasis is on directly discriminating between the two classes.

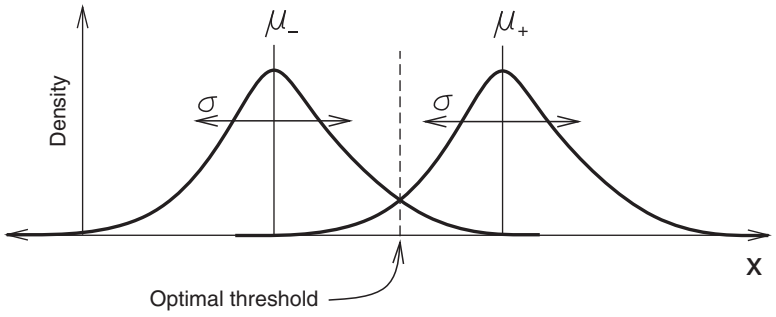

Figure 2.2
A classification problem in which both classes are normally distributed with equal variances but different means.

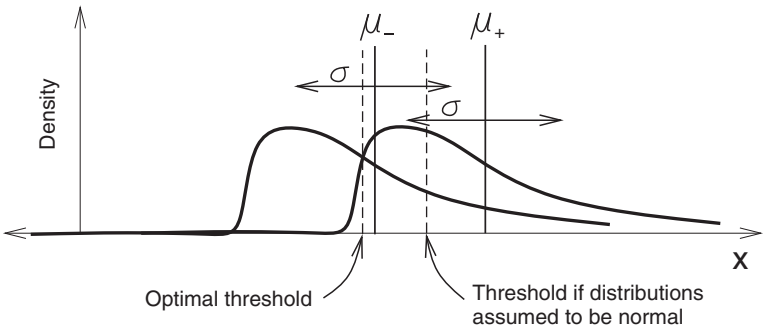

Figure 2.3
A classification problem in which both classes have the same skewed, nonnormal distribution but with different means.

When the assumptions we have made above about the data are valid, good generalization is assured in this simple case. Most importantly, such a guarantee of good performance is dependent on the data actually being generated by two normal distributions. If this assumption does not hold, then the performance of the resulting classifier can become arbitrarily poor, as can be seen in figure 2.3. There, because the two distributions are far from normal, the threshold between positives and negatives that is found by incorrectly assuming normality ends up being well away from optimal, regardless of how much training data is provided.

Normality assumptions lead naturally to the use of threshold rules as in equation (2.1), but we see in this example that an over-dependence on this assumption can yield poor performance. On the other hand, a discriminative approach to this problem in which the best threshold rule is selected based on its training error would be very likely to perform well, since the optimal classifier is a simple threshold here as well. So because the discriminative approach is not based on distributional assumptions, it also can be more robust.

Moreover, if our only goal is to produce a good classifier, then estimating the distribution parameters of each class is entirely irrelevant to the task at hand. We do not care what the means of the distributions are, or even whether or not the distributions are normal. The only thing we care about in classification is which label y is more likely to correspond to any given instance x.

The generative approach can provide a powerful and convenient framework for incorporating into the learning process information that may be available about how the data is generated. However, its performance may be sensitive to the validity of those assumptions. The discriminative approach, on the other hand, attempts more directly to find a good hypothesis by searching for a simple rule that makes accurate predictions on the training data without regard to underlying distributional assumptions. In this latter approach, preconceptions about the data are used to guide how we choose the form of the rules considered, but actual performance may be relatively robust.

2.2 General Methods of Analysis

We return now to the development of general methods for analyzing the generalization error of classifiers generated by learning algorithms. This is the essence of proving an algorithm's effectiveness. As we have discussed, success in learning depends intuitively on finding a classifier (1) that fits the training data well, that is, has low training error, and (2) that is simple. A third prerequisite, of course, is that the learner be provided with a sufficiently large training set. We will see that the generalization error depends on these same three interacting factors in a way that can be formalized in precise terms. On the other hand, the analysis we present does not in any way depend on the form of the distribution of data; for instance, we make no assumptions of normality. This reflects the general robustness of this approach which is largely immune to changes in the underlying distribution.

We begin by analyzing the generalization error of a single hypothesis and then move on to analyze families of hypotheses. Along the way, we will develop several different techniques for measuring the central notion of simplicity and complexity.

2.2.1 A Single Hypothesis

To start, consider a single, fixed hypothesis h. Earlier, we discussed how the training error is used as a proxy for the true error. This motivates us to ask how much the training error $\widehat{\mathrm{err}}(h)$ can differ from the true error $\mathrm{err}(h)$ as a function of the number of training examples m. Note first that there is always a chance that the selected training set will be highly unrepresentative, so that the training error will be a very poor estimate of the true error. This means that it is impossible to give a guarantee that holds with absolute certainty. Instead, we seek bounds that hold true *with high probability* over the choice of the random training set.

In fact, the problem can be seen to be equivalent to one involving coin flipping. When a training example (x_i, y_i) is selected at random, the probability that $h(x_i) \neq y_i$ is exactly $p = \mathrm{err}(h)$, an event that we can identify with a flipped, biased coin coming up heads. In this way, the training set can be viewed as a sequence of m coin flips, each of which is heads with probability p. The problem then is to determine the probability that the fraction $\hat{p}$ of heads in the actual observed sequence of coin flips—that is, the training error—will be significantly different from p. We can explicitly write down the probability, say, of getting at most $(p - \varepsilon)m$ heads, which is exactly

$$\sum_{i=0}^{\lfloor (p-\varepsilon)m \rfloor} \binom{m}{i} p^i (1-p)^{m-i}. \tag{2.4}$$

This is a rather unwieldy expression, but fortunately there exist a number of tools for bounding it. Foremost among these is the family of *Chernoff bounds*, including *Hoeffding's inequality*, one of the simplest and most widely used, which can be stated as follows:

Theorem 2.1 Let $X_1, \ldots, X_m$ be independent random variables such that $X_i \in [0, 1]$. Denote their average value by $A_m = \frac{1}{m} \sum_{i=1}^{m} X_i$. Then for any $\varepsilon > 0$ we have

$$\mathbf{Pr}[A_m \geq \mathbf{E}[A_m] + \varepsilon] \leq e^{-2m\varepsilon^2} \tag{2.5}$$

and

$$\mathbf{Pr}[A_m \leq \mathbf{E}[A_m] - \varepsilon] \leq e^{-2m\varepsilon^2}. \tag{2.6}$$

Hoeffding's inequality applies to averages of arbitrary bounded and independent random variables. In the coin flipping example, we can take $X_i = 1$ (heads) with probability p and $X_i = 0$ (tails) with probability $1 - p$. Then A_m is equal to $\hat{p}$, the fraction of heads observed in a sequence of m flips; its expected value $\mathbf{E}[A_m]$ is p; and equation (2.6) tells us that the chance of at most $(p - \varepsilon)m$ heads, written explicitly in equation (2.4), is at most $e^{-2m\varepsilon^2}$.

In the learning setting, we can define the random variable X_i to be 1 if $h(x_i) \neq y_i$ and 0 otherwise. This means that the average value A_m is exactly $\widehat{\mathrm{err}}(h)$, the training error of h, and that $\mathbf{E}[A_m]$ is exactly the generalization error $\mathrm{err}(h)$. Thus, theorem 2.1, using equation (2.6), implies that the probability of a training sample of size m for which

$$\mathrm{err}(h) \geq \widehat{\mathrm{err}}(h) + \varepsilon$$

is at most $e^{-2m\varepsilon^2}$. Said differently, given m random examples, and for any $\delta > 0$, we can deduce that with probability at least $1 - \delta$, the following upper bound holds on the generalization error of h:

$$\mathrm{err}(h) \leq \widehat{\mathrm{err}}(h) + \sqrt{\frac{\ln(1/\delta)}{2m}}.$$

This follows by setting $\delta = e^{-2m\varepsilon^2}$ and solving for ε. This means, in quantifiable terms, that if h has low training error on a good-size training set, then we can be quite confident that h's true error is also low.

Using equation (2.5) gives a corresponding lower bound on $\mathrm{err}(h)$. We can combine these using the *union bound*, which states simply that

$$\mathbf{Pr}[a \vee b] \leq \mathbf{Pr}[a] + \mathbf{Pr}[b]$$

for any two events a and b. Together, the two bounds imply that the chance that

$$|\mathrm{err}(h) - \widehat{\mathrm{err}}(h)| \geq \varepsilon$$

is at most $2e^{-2m\varepsilon^2}$, or that

$$|\mathrm{err}(h) - \widehat{\mathrm{err}}(h)| \leq \sqrt{\frac{\ln(2/\delta)}{2m}}$$

with probability at least $1 - \delta$.

As with most of the results in this chapter, we do not prove theorem 2.1. However, we note as an aside that the standard technique for proving Chernoff bounds is closely related to the method we use in chapter 3 to analyze AdaBoost's training error. Indeed, as will be seen in section 3.3, a special case of Hoeffding's inequality follows as a direct corollary of our analysis of AdaBoost.

2.2.2 Finite Hypothesis Spaces

Thus, we can bound the difference between the training error and the true error of a *single*, fixed classifier. To apply this to a learning algorithm, it might seem tempting to use the same argument to estimate the error of the single hypothesis that is chosen by the algorithm by minimizing the training error—after all, this is still only one hypothesis that we care about. However, such reasoning would be entirely fallacious. Informally, the problem is that in such an argument, the training errors are used twice—first to choose the seemingly best hypothesis, and then to estimate its true error. Said differently, the reasoning in section 2.2.1 requires that we select the single hypothesis h *before* the training set is randomly chosen. The argument is invalid if h is itself a random variable that depends on the training set as it would be if selected to minimize the training error. Moreover, we have already argued informally in section 2.1.1 that the hypothesis that appears best on the training set is very likely to have a true error that is significantly higher, whereas, for a single hypothesis, the training error is an unbiased estimate of the true error; this is another indication of the fallacy of such an argument.

Despite these difficulties, we will see now how we can analyze the error of the classifier produced by a learning algorithm, even if it is found by minimizing the training error. The intuitive arguments outlined in section 2.1.1 indicate that such a bound will depend on the form of the hypotheses being used, since this form defines how simple or complex they are. Saying that a hypothesis has a particular form is abstractly equivalent to saying that it belongs to some set of hypotheses $\mathcal{H}$, since we can define $\mathcal{H}$ tautologically to be the set of all hypotheses of the chosen form. For instance, $\mathcal{H}$ might be the set of all threshold rules, as in equation (2.1). We call $\mathcal{H}$ the *hypothesis class* or *hypothesis space*.

Generally, our approach will be to show that the training error of *every* hypothesis $h \in \mathcal{H}$ is close to its true error with high probability, leading to so-called *uniform error* or *uniform convergence* bounds. This condition will ensure that the hypothesis with minimum training error also has nearly minimal true error among all hypotheses in the class. For if

$$|\mathrm{err}(h) - \widehat{\mathrm{err}}(h)| \leq \varepsilon \tag{2.7}$$

holds for all $h \in \mathcal{H}$, and if $\hat{h}$ minimizes the training error $\widehat{\mathrm{err}}(h)$, and h^* minimizes the true error $\mathrm{err}(h)$, then $\hat{h}$ also approximately minimizes the true error, since

$$\begin{aligned}\mathrm{err}(\hat{h}) &\leq \widehat{\mathrm{err}}(\hat{h}) + \varepsilon \\ &= \min_{h \in \mathcal{H}} \widehat{\mathrm{err}}(h) + \varepsilon.\end{aligned}$$

$$\begin{aligned} &\leq \widehat{\mathrm{err}}(h^*) + \varepsilon \\ &\leq (\mathrm{err}(h^*) + \varepsilon) + \varepsilon \\ &= \min_{h \in \mathcal{H}} \mathrm{err}(h) + 2\varepsilon. \end{aligned} \tag{2.8}$$

(We assumed that the minima above exist, as will be the case, for instance, if $\mathcal{H}$ is finite; it is straightforward to modify the argument if they do not.)

Although we assumed a two-sided bound as in equation (2.7), we will henceforth restrict our attention primarily to proving one-sided bounds of the form

$$\mathrm{err}(h) \leq \widehat{\mathrm{err}}(h) + \varepsilon \tag{2.9}$$

for all $h \in \mathcal{H}$. We do this for simplicity of presentation, but also for the reason that we typically are interested only in upper bounds on generalization error. This is because, first of all, there is no harm done if the learning algorithm manages to get lucky in picking a hypothesis whose generalization error is significantly *lower* than its training error. But more importantly, this case almost never occurs: Since the learning algorithm is biased toward choosing hypotheses that already have low training error, the generalization error is quite unlikely to be even lower. In fact, a closer look at the argument in equation (2.8) reveals that only one-sided uniform error bounds, as in equation (2.9), were actually used. (We did use the fact that $\widehat{\mathrm{err}}(h^*) \leq \mathrm{err}(h^*) + \varepsilon$, but this does not require the use of a *uniform* bound since it involves only the single, fixed hypothesis h^*.)

To prove that such uniform bounds hold with high probability, the simplest case is when $\mathcal{H}$ is finite, so that the hypothesis h is selected from a finite set of hypotheses that is fixed before observing the training set. In this case, we can use a simple argument based on the union bound: If we fix any *single* hypothesis $h \in \mathcal{H}$, then, as in section 2.2.1, we can use theorem 2.1 to bound the probability of choosing a training set for which $\mathrm{err}(h) - \widehat{\mathrm{err}}(h) \geq \varepsilon$; this will be at most $e^{-2m\varepsilon^2}$. By the union bound, the chance that this happens for *any* hypothesis in $\mathcal{H}$ can be upper bounded simply by summing this probability bound over all the hypotheses in $\mathcal{H}$, which gives $|\mathcal{H}|e^{-2m\varepsilon^2}$. Thus, we obtain the following:

Theorem 2.2 Let $\mathcal{H}$ be a finite space of hypotheses, and assume that a random training set of size m is chosen. Then for any $\varepsilon > 0$,

$$\mathbf{Pr}[\exists h \in \mathcal{H} : \mathrm{err}(h) \geq \widehat{\mathrm{err}}(h) + \varepsilon] \leq |\mathcal{H}|e^{-2m\varepsilon^2}.$$

Thus, with probability at least $1 - \delta$,

$$\mathrm{err}(h) \leq \widehat{\mathrm{err}}(h) + \sqrt{\frac{\ln|\mathcal{H}| + \ln(1/\delta)}{2m}} \tag{2.10}$$

for all $h \in \mathcal{H}$.

The second bound, equation (2.10), on the generalization performance of any hypothesis h captures in a single formula the three factors noted earlier which determine the success of

a learning algorithm. With regard to the training error $\widehat{\mathrm{err}}(h)$, the first of these factors, we see that the formula is consistent with the intuition that better fit to the training data implies better generalization. The second factor is the number of training examples m, where again the bound captures quantitatively the unarguable notion that having more data is better. Finally, the formula contains a term that depends on the form of the hypotheses being used, that is, on the class $\mathcal{H}$ from which they are chosen. Note that if hypotheses in $\mathcal{H}$ are written down in some way, then the number of bits needed to give each hypothesis $h \in \mathcal{H}$ a unique name is about $\lg |\mathcal{H}|$. Thus, we can think of $\ln |\mathcal{H}|$—the term appearing in the formula, which is off by only a constant from $\lg |\mathcal{H}|$—as roughly the "description length" of the hypotheses being used, rather a natural measure of $\mathcal{H}$'s complexity, the third factor affecting generalization performance. In this way, the formula is consistent with Occam's razor, the notion that, all else being equal, simpler hypotheses perform better than more complex ones.

2.2.3 Infinite Hypothesis Spaces

We next consider infinitely large hypothesis spaces. The results of section 2.2.2 are useless in such cases—for instance, when using threshold rules of the form given in equation (2.1), where there are infinitely many choices for the threshold ν, and thus infinitely many hypotheses of this form. However, this hypothesis space has an important property that was noted in section 2.1: Any training set of m (distinct) examples partitions the infinitely large space into just $m + 1$ equivalence classes such that the predictions of any two hypotheses in the same equivalence class are identical on all m points. Said differently, the number of distinct labelings of the m training points that can be produced by hypotheses in $\mathcal{H}$ is at most $m + 1$. See figure 2.4 for an example.

Thus, in this case, although there are infinitely many hypotheses in $\mathcal{H}$, there are, in a sense, effectively only $m + 1$ hypotheses of any relevance with respect to a fixed set of m examples. It would be tempting indeed to regard $m + 1$ as the "effective" size of the hypothesis space, and then to attempt to apply theorem 2.2 with $|\mathcal{H}|$ replaced by $m + 1$. This would give very reasonable bounds. Unfortunately, such an argument would be flawed because the finite set of effective hypotheses that is induced in this way *depends on the training data*, so that the argument suggested earlier for proving theorem 2.2 using the union bound cannot be applied. Nevertheless, using more sophisticated arguments, it turns out to be possible to prove that this alluring idea actually works so that, modulo some adjusting of the constants, $|\mathcal{H}|$ can be replaced in theorem 2.2 with the "effective" size of $\mathcal{H}$ as measured by the number of labelings of $\mathcal{H}$ on a finite sample.

To make these ideas formal, for any hypothesis class $\mathcal{H}$ over $\mathcal{X}$ and for any finite sample $S = \langle x_1, \ldots, x_m \rangle$, we define the set of *dichotomies* or *behaviors* $\Pi_{\mathcal{H}}(S)$ to be all possible labelings of S by functions in $\mathcal{H}$. That is,

$$\Pi_{\mathcal{H}}(S) \doteq \{\langle h(x_1), \ldots, h(x_m) \rangle : h \in \mathcal{H}\}.$$

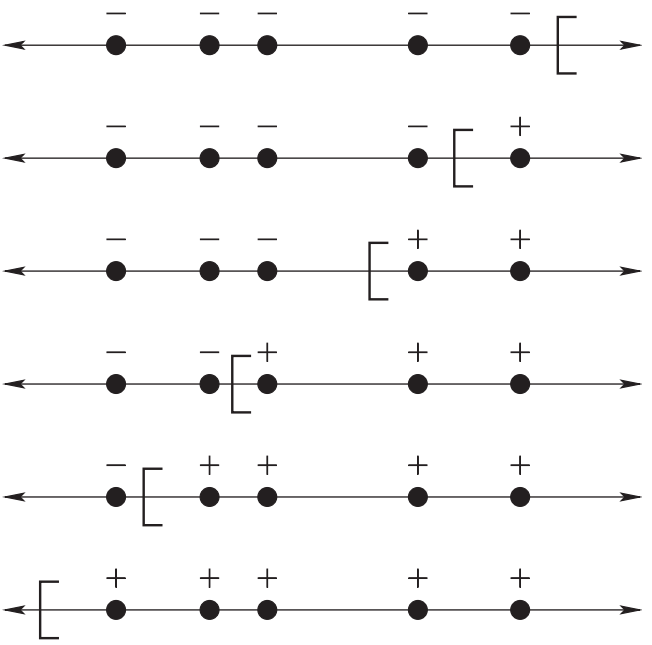

Figure 2.4
Five points on the real line, and a listing of all six possible labelings of the points using threshold functions of the form shown in equation (2.1). On each line, the left bracket [shows a sample threshold ν realizing the given labeling.

We also define the *growth function* $\Pi_{\mathcal{H}}(m)$ which measures the maximum number of dichotomies for any sample S of size m:

$$\Pi_{\mathcal{H}}(m) \doteq \max_{S \in \mathcal{X}^m} |\Pi_{\mathcal{H}}(S)|.$$

For instance, when $\mathcal{H}$ is the class of threshold functions, $\Pi_{\mathcal{H}}(m) = m + 1$, as we have already seen.

We can now state a more general result than theorem 2.2 that is applicable to both finite and infinite hypothesis spaces. Ignoring constants, this theorem has replaced $|\mathcal{H}|$ with the growth function $\Pi_{\mathcal{H}}(m)$.

Theorem 2.3 Let $\mathcal{H}$ be any[1] space of hypotheses, and assume that a random training set of size m is chosen. Then for any $\varepsilon > 0$,

$$\mathbf{Pr}[\exists h \in \mathcal{H} : \mathrm{err}(h) \geq \widehat{\mathrm{err}}(h) + \varepsilon] \leq 8\Pi_{\mathcal{H}}(m)e^{-m\varepsilon^2/32}.$$

Thus, with probability at least $1 - \delta$,

1. To be strictly formal, we have to restrict the class $\mathcal{H}$ to be measurable with respect to an appropriate probability space. Here, and throughout this book, we ignore this finer point, which we implicitly assume to hold.

$$\text{err}(h) \leq \widehat{\text{err}}(h) + \sqrt{\frac{32(\ln \Pi_{\mathcal{H}}(m) + \ln(8/\delta))}{m}} \tag{2.11}$$

for all $h \in \mathcal{H}$.

So our attention naturally turns to the growth function $\Pi_{\mathcal{H}}(m)$. In "nice" cases, such as for threshold functions, the growth function is only polynomial in m, that is, $O(m^d)$ for some constant d that depends on $\mathcal{H}$. In such a case, $\ln \Pi_{\mathcal{H}}(m)$ is roughly $d \ln m$ so that the term on the far right of equation (2.11), as a function of the training set size m, approaches zero at the favorable rate $O(\sqrt{(\ln m)/m})$.

However, the growth function need not always be polynomial. For instance, consider the class of all hypotheses that are defined to be $+1$ on some finite but unrestricted set of intervals of the real line and -1 on the complement (where instances here are points on the line). An example of a hypothesis in this class is given by the classifier in equation (2.2), which is $+1$ on the intervals $[3.4, 4.0)$, $[5.2, 6.3)$, and $[7.1, \infty)$, and -1 on all other points. This hypothesis space is so rich that for *any* labeling of any set of distinct training points, there always exists a consistent classifier which can be constructed simply by choosing sufficiently small intervals around each of the positively labeled instances. Thus, the number of dichotomies for any set of m distinct points is exactly 2^m, the worst possible value of the growth function. In such a case, $\ln \Pi_{\mathcal{H}}(m) = m \ln 2$, so the bound in theorem 2.3 is useless, being of order $\theta(1)$. On the other hand, the very richness that makes it so easy to find a hypothesis consistent with any training set also strongly suggests that the generalization capabilities of such hypotheses will be very weak indeed.

So we have seen one case in which $\Pi_{\mathcal{H}}(m)$ is polynomial, and another in which it is 2^m for all m. It is a remarkable fact of combinatorics that these are the *only* two behaviors that are possible for the growth function, no matter what the space $\mathcal{H}$ may be. Moreover, as we will soon see, statistically tractable learning turns out to correspond exactly to the former case, with the exponent of the polynomial acting as a natural measure of the complexity of the class $\mathcal{H}$.

To characterize this exponent, we now define some key combinatorial concepts. First, when all 2^m possible labelings of a sample S of size m can be realized by hypotheses in $\mathcal{H}$, we say that S is *shattered* by $\mathcal{H}$. Thus, S is shattered by $\mathcal{H}$ if $|\Pi_{\mathcal{H}}(S)| = 2^m$. Further, we define the *Vapnik-Chervonenkis (VC) dimension* of $\mathcal{H}$ to be the size of the largest sample S shattered by $\mathcal{H}$. If arbitrarily large finite samples can be shattered by $\mathcal{H}$, then the VC-dimension is ∞.

For instance, for threshold functions as in equation (2.1), the VC-dimension is 1 since a single point can be labeled $+1$ or -1 by such rules (which means the VC-dimension is at least 1), but no pair of points can be shattered, since if the leftmost point is labeled $+1$, the rightmost point must also be labeled $+1$ (thus, the VC-dimension is strictly less than 2). For the unions-of-intervals example above, we saw that any set of distinct points is shattered, so the VC-dimension is ∞ in this case.

Indeed, when the VC-dimension is infinite, $\Pi_{\mathcal{H}}(m) = 2^m$ by definition. On the other hand, when the VC-dimension is a finite number d, the growth function turns out to be polynomial, specifically, $O(m^d)$. This follows from a beautiful combinatorial fact known as Sauer's lemma:

Lemma 2.4 (Sauer's lemma) *If $\mathcal{H}$ is a hypothesis class of VC-dimension $d < \infty$, then for all m*

$$\Pi_{\mathcal{H}}(m) \leq \sum_{i=0}^{d} \binom{m}{i}.$$

(We follow the convention that $\binom{n}{k} = 0$ if $k < 0$ or $k > n$.) For $m \leq d$, this bound is equal to 2^m (and indeed, by d's definition, the bound matches $\Pi_{\mathcal{H}}(m)$ in this case). For $m \geq d \geq 1$, the following bound is often useful:

$$\sum_{i=0}^{d} \binom{m}{i} \leq \left(\frac{em}{d}\right)^d \tag{2.12}$$

(where, as usual, e is the base of the natural logarithm).

We can plug this bound immediately into theorem 2.3 to obtain the following:

Theorem 2.5 Let $\mathcal{H}$ be a hypothesis space of VC-dimension $d < \infty$, and assume that a random training set of size m is chosen where $m \geq d \geq 1$. Then for any $\varepsilon > 0$,

$$\mathbf{Pr}[\exists h \in \mathcal{H} : \mathrm{err}(h) \geq \widehat{\mathrm{err}}(h) + \varepsilon] \leq 8\left(\frac{em}{d}\right)^d e^{-m\varepsilon^2/32}.$$

Thus, with probability at least $1 - \delta$,

$$\mathrm{err}(h) \leq \widehat{\mathrm{err}}(h) + O\left(\sqrt{\frac{d\ln(m/d) + \ln(1/\delta)}{m}}\right) \tag{2.13}$$

for all $h \in \mathcal{H}$.

As before, this second bound on the generalization error captures the three factors discussed earlier. However, the complexity of $\mathcal{H}$, which was earlier measured by $\ln|\mathcal{H}|$, now is measured instead by the VC-dimension d. This is an important result because it shows that limiting the VC-dimension of a hypothesis class can be used as a general tool for avoiding overfitting.

The VC-dimension may not at first seem intuitive as a measure of complexity. However, it can be shown that the VC-dimension can also be used to provide a general lower bound on the number of examples needed for learning. Thus, in this sense, VC-dimension fully characterizes the (statistical) complexity of learning. Moreover, VC-dimension is related to our earlier complexity measure in that it can never exceed $\lg|\mathcal{H}|$ (see exercise 2.2).

In addition, VC-dimension turns out often (but not always!) to be equal to the number of parameters defining hypotheses in $\mathcal{H}$. For instance, suppose that examples are points $\mathbf{x}$ in $\mathbb{R}^n$, and that hypotheses in $\mathcal{H}$ are *linear threshold functions* of the form

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \mathbf{x} > 0 \\ -1 & \text{else} \end{cases}$$

for some weight vector $\mathbf{w} \in \mathbb{R}^n$. Then it can be shown (see lemma 4.1) that the VC-dimension of this class is exactly n, which matches the number of parameters, that is, the number of dimensions in the vector $\mathbf{w}$ that defines each hypothesis in $\mathcal{H}$.

2.2.4 A More Abstract Formulation

The framework above, particularly theorem 2.3, can be stated in more abstract terms that we will sometimes find easier to work with. However, the reader may wish to skip this technical section and come back to it when needed later in the book.

Briefly, let $\mathcal{Z}$ be any set, let $\mathcal{A}$ be a family of subsets of $\mathcal{Z}$, and let $\mathcal{D}$ be a distribution over $\mathcal{Z}$. We consider the problem of estimating from a random sample $S = \langle z_1, \ldots, z_m \rangle$ the probability of each set $A \in \mathcal{A}$. As usual, $\mathbf{Pr}_{z \sim \mathcal{D}}[\cdot]$ denotes probability when z is chosen at random according to the distribution $\mathcal{D}$, and we let $\mathbf{Pr}_{z \sim S}[\cdot]$ denote empirical probability, that is, probability when z is chosen uniformly at random from among the m sample points $z_1, \ldots, z_m$. We wish to show that $\mathbf{Pr}_{z \sim S}[z \in A]$, the empirical probability of any set A, is likely to be close to its true probability $\mathbf{Pr}_{z \sim \mathcal{D}}[z \in A]$, and we want this to be true simultaneously for *every* set $A \in \mathcal{A}$. For a single, fixed set A, this can be shown using Hoeffding's inequality (theorem 2.1). Likewise, if $\mathcal{A}$ is finite, then it can be shown by an application of the union bound. But when $\mathcal{A}$ is infinite, we need to generalize the machinery developed in section 2.2.3.

For any finite sample S as above, we consider the restriction of $\mathcal{A}$ to S, denoted $\Pi_{\mathcal{A}}(S)$, that is, the intersection of S (treated as a set) with each set $A \in \mathcal{A}$:

$$\Pi_{\mathcal{A}}(S) \doteq \{\{z_1, \ldots, z_m\} \cap A : A \in \mathcal{A}\}.$$

Analogous to $\Pi_{\mathcal{H}}(S)$, this collection can be viewed as the set of all "in-out behaviors" of sets $A \in \mathcal{A}$ on the points in S. As before, the growth function is the maximum cardinality of this set over all samples S of size m:

$$\Pi_{\mathcal{A}}(m) \doteq \max_{S \in \mathcal{Z}^m} |\Pi_{\mathcal{A}}(S)|.$$

With these definitions, theorem 2.3 becomes:

Theorem 2.6 Let $\mathcal{A}$ be a family of subsets of $\mathcal{Z}$, and suppose that a random sample S of m points is chosen independently from $\mathcal{Z}$, each point selected according to the same distribution $\mathcal{D}$. Then for any $\varepsilon > 0$,

$$\mathbf{Pr}[\exists A \in \mathcal{A} : \mathbf{Pr}_{z \sim \mathcal{D}}[z \in A] \geq \mathbf{Pr}_{z \sim S}[z \in A] + \varepsilon] \leq 8\Pi_{\mathcal{A}}(m)e^{-m\varepsilon^2/32}.$$

Thus, with probability at least $1 - \delta$,

$$\mathbf{Pr}_{z\sim\mathcal{D}}[z \in A] \leq \mathbf{Pr}_{z\sim S}[z \in A] + \sqrt{\frac{32(\ln \Pi_{\mathcal{A}}(m) + \ln(8/\delta))}{m}}$$

for all $A \in \mathcal{A}$.

To obtain the formulation given in section 2.2.3 as a special case, we first let $\mathcal{Z} \doteq \mathcal{X} \times \{-1, +1\}$, the space of all possible labeled examples. Then, for a given hypothesis space $\mathcal{H}$, we define a family $\mathcal{A}$ of subsets A_h, one for each $h \in \mathcal{H}$, where A_h is the set of examples on which h makes a mistake. That is,

$$\mathcal{A} \doteq \{A_h : h \in \mathcal{H}\}, \tag{2.14}$$

and

$$A_h \doteq \{(x, y) \in \mathcal{Z} : h(x) \neq y\}.$$

Then it can be verified with these definitions that

$$\Pi_{\mathcal{H}}(m) = \Pi_{\mathcal{A}}(m) \tag{2.15}$$

and that theorem 2.6 yields exactly theorem 2.3. (See exercise 2.9.)

2.2.5 Consistent Hypotheses

We have seen that, when possible, it is sometimes desirable for a learning algorithm to produce a hypothesis that is *consistent* with the training data so that it makes no mistakes at all on the training set. Of course, our preceding analyses hold; this is just a special case in which $\widehat{\mathrm{err}}(h) = 0$. However, the bounds obtained in this fashion are particularly loose, giving bounds on the order of $1/\sqrt{m}$. In fact, for consistent hypotheses, it turns out that the square roots on all of our convergence bounds can generally be removed (with some adjusting of constants), giving the far faster convergence rate of just $1/m$ (ignoring log terms).

To get an intuitive feeling for why this is so, consider again the problem of estimating the bias p of a coin, as in section 2.2.1. It turns out that a coin with bias p close to 0 or 1 is much easier to estimate (in the sense of requiring fewer samples) than one with bias close to $\frac{1}{2}$. This is reflected in the bounds that can be proved. According to Hoeffding's inequality (theorem 2.1), if $\hat{p}$ is the observed fraction of heads in m flips, then

$$p \leq \hat{p} + \sqrt{\frac{\ln(1/\delta)}{2m}} \tag{2.16}$$

with probability at least $1 - \delta$. In other words, the true probability p is within $O(1/\sqrt{m})$ of its estimate $\hat{p}$.

Now let us consider what happens when $\hat{p} = 0$, that is, when there are no heads in the observed sequence, corresponding to the case of a hypothesis that is consistent with the entire

training set. The probability of getting *no* heads in m flips is exactly $(1-p)^m \leq e^{-pm}$. This means that if $p \geq \ln(1/\delta)/m$, then $\hat{p}$ will be zero with probability at most δ. Turning this statement around implies that when $\hat{p} = 0$, we can conclude that

$$p < \frac{\ln(1/\delta)}{m}$$

with probability at least $1-\delta$. Note that this estimate is $O(1/m)$ rather than $O(1/\sqrt{m})$ as in equation (2.16).

This style of argument can be applied in the learning setting as well, yielding results such as those summarized in the following theorem:

Theorem 2.7 Let $\mathcal{H}$ be a space of hypotheses, and assume that a random training set S of size m is chosen.

If $\mathcal{H}$ is finite, then with probability at least $1-\delta$,

$$\text{err}(h) \leq \frac{\ln|\mathcal{H}| + \ln(1/\delta)}{m} \tag{2.17}$$

for every $h \in \mathcal{H}$ that is consistent with S.

More generally, for any (finite or infinite) $\mathcal{H}$, with probability at least $1-\delta$,

$$\text{err}(h) \leq \frac{2\lg \Pi_{\mathcal{H}}(2m) + 2\lg(2/\delta)}{m} \tag{2.18}$$

for every $h \in \mathcal{H}$ that is consistent with S. If $\mathcal{H}$ has VC-dimension d, where $m \geq d \geq 1$, then with probability at least $1-\delta$,

$$\text{err}(h) \leq \frac{2d\lg(2em/d) + 2\lg(2/\delta)}{m} \tag{2.19}$$

for every $h \in \mathcal{H}$ that is consistent with S.

This theorem gives high-probability bounds on the true error of all consistent hypotheses. Each of the three bounds in equations (2.17), (2.18), and (2.19) states that, with probability at least $1-\delta$, $\text{err}(h) \leq \varepsilon$ for every $h \in \mathcal{H}$ that is consistent with S (for values of ε as given in the theorem). In other words, using slightly different phrasing, each bound says that with probability at least $1-\delta$, for every $h \in \mathcal{H}$, if h is consistent with S, then $\text{err}(h) \leq \varepsilon$. Or, formalizing these results in more precise, mathematical terms, the bounds state that with probability at least $1-\delta$, the random variable

$$\sup\{\text{err}(h) \mid h \in \mathcal{H} \text{ is consistent with } S\}$$

is at most ε.

2.2.6 Compression-Based Bounds

We have described two general techniques for analyzing a learning algorithm, one based simply on counting the number of hypotheses in the class $\mathcal{H}$, and the other based on $\mathcal{H}$'s VC-dimension. In this section, we briefly describe a third approach.

It has already been noted that $\lg |\mathcal{H}|$, our first complexity measure, is closely related to the number of bits needed to describe each hypothesis $h \in \mathcal{H}$. Thus, from this perspective, the learning algorithm must find a relatively short description that can be used to reconstruct labels for the training examples. This idea depends on the description being in bits, and thus on $\mathcal{H}$ being finite.

Nevertheless, even when $\mathcal{H}$ is infinite, it will often be the case that the hypotheses produced by a particular algorithm can be given a short description, not in bits but in terms of *training examples* themselves. For instance, for the class of threshold functions as in equation (2.1), each classifier is defined by the threshold ν. Moreover, as we have seen, the behavior of such classifiers is equivalent with respect to a particular training set for all thresholds ν lying between two adjacent data points. Thus, a learning algorithm can choose ν to be one of those data points, and, in so doing, produces a classifier that can be described by just one of the training examples. (Alternatively, if it seems more natural, the learner can take ν to be the midpoint between two adjacent data points. This hypothesis can be described by two of the training examples.)

We call such an algorithm whose hypotheses can be represented by κ of the training examples a *compression scheme of size* κ. Thus, formally, such an algorithm is associated with a function $\mathcal{K}$ that maps κ-tuples of labeled examples to hypotheses h in some space $\mathcal{H}$. Given training examples $(x_1, y_1), \ldots, (x_m, y_m)$, such an algorithm chooses some indices $i_1, \ldots, i_\kappa \in \{1, \ldots, m\}$, and outputs the hypothesis

$$h = \mathcal{K}((x_{i_1}, y_{i_1}), \ldots, (x_{i_\kappa}, y_{i_\kappa}))$$

determined by the corresponding examples.

For such algorithms, which arise quite often and quite naturally, bounds on the generalization error can be derived in much the same way as for the case of finite $\mathcal{H}$. In particular, the following can be proved:

Theorem 2.8 Suppose a learning algorithm based on a compression scheme of size κ is provided with a random training set of size m. Then with probability at least $1 - \delta$, the hypothesis h produced by this algorithm satisfies

$$\mathrm{err}(h) \leq \left(\frac{m}{m-\kappa}\right) \widehat{\mathrm{err}}(h) + \sqrt{\frac{\kappa \ln m + \ln(1/\delta)}{2(m-\kappa)}}.$$

Furthermore, with probability at least $1 - \delta$, any consistent hypothesis h produced by this algorithm satisfies

$$\text{err}(h) \leq \frac{\kappa \ln m + \ln(1/\delta)}{m - \kappa}.$$

Thus, for such algorithms it is the size κ of the compression scheme that acts as a complexity term.

2.2.7 Discussion

We have explored three general methods for analyzing learning algorithms. The techniques are closely related, differing primarily in the complexity measure employed. These bounds are extremely useful in understanding the *qualitative* behavior of learning algorithms. As we have discussed already, the bounds describe the dependence of the generalization error on the training error, the number of examples, and the complexity of the chosen hypothesis. Furthermore, these bounds are quite helpful in understanding the important and ubiquitous phenomenon of overfitting. Disregarding δ and log factors, each of the bounds in theorems 2.2, 2.5 and 2.8 have the form

$$\text{err}(h) \leq \widehat{\text{err}}(h) + \tilde{O}\left(\sqrt{\frac{C_{\mathcal{H}}}{m}}\right) \tag{2.20}$$

where $C_{\mathcal{H}}$ is some measure of the complexity of $\mathcal{H}$. As the complexity of the hypotheses used is permitted to increase, the training error tends to decrease, causing the first term in equation (2.20) to decrease. However, this also causes the second term to increase. The result is an idealized "learning curve" like the one in figure 2.5, which pretty well matches the kind of overfitting behavior often observed in practice.

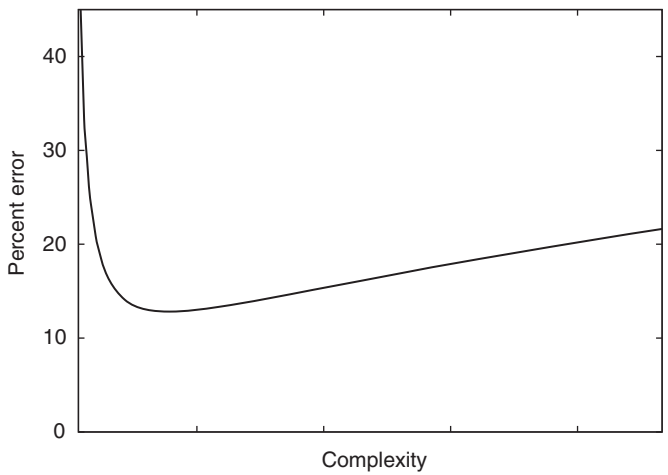

Figure 2.5
An idealized plot of the generalization error of a classifier with varying complexity, as predicted by a bound of the form in equation (2.20).

On the other hand, these bounds are usually too loose to be applied *quantitatively* on actual learning problems. In most cases, the bounds suggest that, with the amount of training data available, only very simple hypothesis classes can be used, while in actual practice, quite large hypothesis classes are used regularly with good results. The problem is that the bounds are overly pessimistic, holding as they do for *all* distributions, including those "worst-case" distributions which make learning as difficult as possible. Thus, while the uniform nature of the bounds is an unquestionable strength that lends generality and robustness to the results, this same uniformity can also be a weakness in the sense that the results may better characterize the theoretical worst case than the actual case encountered in practice.

One way to tighten the bounds may be to take into account additional quantities that can be measured on the training set. Bounds of the type given in the theorems above take into account only the training error $\widehat{\mathrm{err}}(h)$, but other quantities can be considered. For instance, chapter 5 describes bounds on the generalization error of boosting algorithms which depend on properties of the *margin distribution* of the training set.

2.3 A Foundation for the Study of Boosting Algorithms

We next introduce an idealized framework for the mathematical study of machine learning, one that admits absolute guarantees on performance. As we will see, this model of learning will allow us to define the concept of boosting in precise terms, and will thus provide a foundation for the development and analysis of boosting algorithms.

2.3.1 Absolute Guarantees on Performance

As just discussed, the mode of analysis studied in section 2.2 is very general and quite agnostic in the sense that we have made no assumptions about the form of the distribution $\mathcal{D}$ generating the labeled examples (x, y). On the one hand, this generality has made it possible to state results that are applicable in a very broad range of settings without the need for prior assumptions about the data distribution. On the other hand, this same generality has also precluded us from providing *absolute* guarantees on performance. Rather, the bounds that we have discussed tell us that the generalization error will be small *if* during training we can place our hands on a simple hypothesis with low training error on a sufficiently large training set. The bounds do not tell us when it will be possible to obtain a hypothesis that has, say, 99% generalization accuracy. This can only be deduced, according to the bounds, *after* the training error has been observed.

Even as the training set becomes extremely large, the bounds do not guarantee low generalization error. Indeed, such guarantees are impossible in such generality since the distribution $\mathcal{D}$ may be such that the label y is intrinsically unpredictable. For instance, since we assume nothing about $\mathcal{D}$, it is possible that y is equally likely to be $+1$ or -1, independent of x; in such an extreme case, no amount of training or computation can result

in a hypothesis with generalization error less than 50%. Thus, to develop a mathematical model of learning which admits absolute guarantees on the generalization error, we must accept additional assumptions about the form of the generating distribution $\mathcal{D}$.

As a first step, we suppose for now that the goal is to learn a classifier with nearly perfect accuracy. We have just seen that this is not always possible, and indeed, realistically, is almost never attainable in practice. Nevertheless, achieving the highest possible accuracy is the ultimate goal of learning, and as such, understanding theoretically when near perfect accuracy can be achieved is a fundamental question.

As we have seen, when there is intrinsic randomness in the labels y themselves, so that y is not strictly determined by x, there is no way to find a hypothesis h which can predict y perfectly for any x, since such a hypothesis does not exist. Thus, the first necessary assumption in our model is that there exists a functional relationship between the instances x and the labels y; in other words, we assume that there exists a *target function*

$$c : \mathcal{X} \to \{-1, +1\}$$

such that, for any x, the associated label y is equal to $c(x)$ with probability 1. That is,

$$\mathbf{Pr}_{(x,y)\sim\mathcal{D}}[y = c(x) \mid x] = 1.$$

This is equivalent to simply regarding $\mathcal{D}$ as a distribution over $\mathcal{X}$ and assuming that examples are of the form $(x, c(x))$, so that each example x is deterministically assigned the label $c(x)$.

Even with such deterministic labeling of the data, learning may be impossible. For instance, if the target function c is an entirely arbitrary function over $\mathcal{X}$, then no finite training set can provide any connection to examples not seen during training; therefore, we cannot hope to find a hypothesis that makes accurate predictions, other than on the examples seen during training. Thus, for generalization to be plausible, we also must make assumptions about the nature of the target function so that there can exist some relationship between the labels of the training data and those on test examples. We can summarize any such knowledge about c by assuming that c comes from some known class of functions $\mathcal{C}$, called the *target (function) class*.

So our problem of understanding the circumstances under which nearly perfect learning is possible can be rephrased as that of determining for which classes $\mathcal{C}$—embodying our assumptions about the target c—such learning is achievable. Here, again, our goal is to learn a hypothesis h with nearly perfect accuracy, that is, with generalization error below ϵ for any specified value of $\epsilon > 0$, though naturally more data will need to be allowed for smaller values of ϵ. Moreover, to avoid further assumptions, we ask that learning be possible for any distribution $\mathcal{D}$ over the instance space $\mathcal{X}$, and for any target function $c \in \mathcal{C}$. Finally, since there is always the possibility that a highly unrepresentative random training sample will be selected, we allow learning to fail entirely with probability $\delta > 0$, where δ should be controllable in a manner similar to ϵ.

So this notion of learning requires that, with high probability over the choice of the random sample, a hypothesis h be found that is nearly perfect, or approximately correct, hence the name *probably approximately correct* (*PAC*) *learnability*. To distinguish from what is to follow, we also call this *strong learnability*. Formally, then, a class $\mathcal{C}$ is *strongly PAC learnable* if there exists a learning algorithm A such that for any distribution $\mathcal{D}$ over the instance space $\mathcal{X}$, and for any $c \in \mathcal{C}$ and for any positive values of ϵ and δ, algorithm A takes as input $m = m(\epsilon, \delta)$ examples $(x_1, c(x_1)), \ldots, (x_m, c(x_m))$ where x_i is chosen independently at random according to $\mathcal{D}$, and produces a hypothesis h such that

$$\mathbf{Pr}[\mathrm{err}(h) > \epsilon] \leq \delta.$$

Here, $\mathrm{err}(h)$ is the generalization error of h with respect to distribution $\mathcal{D}$, and the probability is over the random selection of training examples $x_1, \ldots, x_m$. Note that the training set size m can depend on ϵ and δ, and we typically require these to be polynomial in $1/\epsilon$ and $1/\delta$. Further, the sample size will usually also depend on properties of the class $\mathcal{C}$.

When computability is not an issue, we can immediately apply the results of section 2.2.5 to obtain quite general PAC results. In particular, for any class $\mathcal{C}$, consider an algorithm A that, given any sample, selects h to be any function in $\mathcal{C}$ that is consistent with the observed data. By assumption, such a hypothesis h must exist (although the computational considerations of finding it might be prohibitive). Thus, in this case our hypothesis space $\mathcal{H}$ is identical to $\mathcal{C}$. When $\mathcal{C}$ is finite, applying theorem 2.7, we thus immediately get that with probability at least $1 - \delta$,

$$\mathrm{err}(h) \leq \frac{\ln |\mathcal{C}| + \ln(1/\delta)}{m}.$$

Setting the right-hand side equal to ϵ, this means that A is PAC (so that $\mathrm{err}(h) \leq \epsilon$ with probability at least $1 - \delta$) when given a sample of size

$$m = \left\lceil \frac{\ln |\mathcal{C}| + \ln(1/\delta)}{\epsilon} \right\rceil.$$

Similarly, for $\mathcal{C}$ infinite, the algorithm can be shown to be PAC for some sample size that depends on the VC-dimension of $\mathcal{C}$ (provided it is finite).

Thus, our generalization bounds indicate generally that only a moderate-size sample is statistically adequate for PAC learning. What remains is the problem of finding an efficient learning algorithm, an issue that cannot be ignored in the real world. Therefore, we often add an efficiency requirement that the learning algorithm A compute h in polynomial time. Characterizing *efficient* PAC learnability turns out to be much more difficult and involved. Indeed, computational tractability has very often been found to be far more limiting and constraining in the design of learning systems than any statistical considerations. In other words, it is often the case that a learning problem cannot be solved, even when more

than enough data has been provided to ensure statistical generalization, solely because the associated computational problem is intractable. (For instance, it is known that this is the case when $\mathcal{C}$ is the class of all polynomial-size formulas constructed using the usual operations (AND, OR, and NOT) over n Boolean variables.)

2.3.2 Weak Learnability and Boosting Algorithms

As indicated above, requiring nearly perfect generalization is usually unrealistic, sometimes for statistical reasons and often for purely computational reasons. What happens if we drop this overly stringent requirement? In other words, rather than requiring a generalization error below, say, 1%, what if we were content with error below 10%? or 25%? In the most extreme case, what if our goal is merely to find a hypothesis whose error is just slightly below the trivial baseline of 50%, which is attainable simply by guessing every label at random? Surely, learning must become easier when (far) less-than-perfect accuracy is deemed sufficient.

Learning with such a weak demand on accuracy is called *weak learning*. In terms of its definition, the problem is only a slight modification of PAC learning in which we drop the requirement that the learning algorithm achieve error at most ϵ for *every* $\epsilon > 0$. Rather, we only make this requirement for some *fixed* ϵ, say $\epsilon = \frac{1}{2} - \gamma$, for some fixed but small "edge" $\gamma > 0$. Formally, then, a target class $\mathcal{C}$ is *weakly PAC learnable* if for some $\gamma > 0$, there exists a learning algorithm A such that for any distribution $\mathcal{D}$ over the instance space $\mathcal{X}$, and for any $c \in \mathcal{C}$ and for any positive value of δ, algorithm A takes as input $m = m(\delta)$ examples $(x_1, c(x_1)), \ldots, (x_m, c(x_m))$, where x_i is chosen independently at random according to $\mathcal{D}$, and produces a hypothesis h such that

$$\mathbf{Pr}\left[\text{err}(h) > \tfrac{1}{2} - \gamma\right] \leq \delta.$$

Weak learnability arises as a natural relaxation of the overly demanding strong learning model. Indeed, on its face one might be concerned that the model is now too weak, accepting as it does any hypothesis that is even slightly better than random guessing. Surely, it would seem, there must be many examples of classes that are weakly learnable (with accuracy only 51%) but not strongly learnable (to accuracy 99%).

This intuition—which turns out to be entirely incorrect—points to a fundamental question: Are the strong and weak learning models equivalent? In other words, is it the case that there exist classes that are weakly but not strongly learnable? Or is it the case that any class that can be weakly learned can also be strongly learned? Boosting arose as an answer to exactly this theoretical question. The existence of boosting algorithms proves that the models are equivalent by showing constructively that any weak learning algorithm can be converted into a strong learning algorithm. Indeed, it is exactly this property that defines boosting in its true technical sense.

Formally, a *boosting algorithm* B is given access to a weak learning algorithm A for $\mathcal{C}$ which, when provided with $m_0(\delta)$ examples, is guaranteed to produce a weak hypothesis

h with $\text{err}(h) \leq \frac{1}{2} - \gamma$ with probability at least $1 - \delta$. In addition, like any PAC algorithm, B is provided with $\epsilon > 0$, $\delta > 0$, and m labeled examples $(x_1, c(x_1)), \ldots, (x_m, c(x_m))$ for some $c \in \mathcal{C}$ (where $\mathcal{C}$ need not be known to B). Using its access to A, the boosting algorithm must produce its own (strong) hypothesis H such that

$$\mathbf{Pr}[\text{err}(H) > \epsilon] \leq \delta. \tag{2.21}$$

Further, there should only be a polynomial blowup in complexity. In other words, B's sample size m should only be polynomial in $1/\epsilon$, $1/\delta$, $1/\gamma$ and m_0, and similarly, B's running time should be polynomially related to A's (as well as the other parameters). Clearly, applying such an algorithm to a weak learning algorithm for some class $\mathcal{C}$ demonstrates, by definition, that the class is strongly learnable as well. AdaBoost is indeed a boosting algorithm in this technical sense, as will be discussed in section 4.3.

That strong and weak learnability are equivalent implies that learning, as we have defined it, is an "all or nothing" phenomenon in the sense that for every class $\mathcal{C}$, either $\mathcal{C}$ is learnable with nearly perfect accuracy for every distribution, or $\mathcal{C}$ cannot be learned in even the most minimal way on every distribution. There is nothing in between these two extremes.

2.3.3 Approaches to the Analysis of Boosting Algorithms

In this chapter, we have explored two modes of analysis. In the first, the generalization error of a selected hypothesis is bounded in terms of measurable empirical statistics, most commonly its training error. No explicit assumptions are made about the data, and as a result, good generalization depends on an implicit assumption that a hypothesis with low training error can be found. In the second style of analysis, additional assumptions are made about the underlying data generation process, admitting absolute bounds on generalization. In the same way, boosting algorithms can be analyzed using either approach.

Every learning algorithm depends explicitly or implicitly upon assumptions, since learning is quite impossible otherwise. Boosting algorithms are built upon the assumption of weak learnability, the premise that a method already exists for finding poor though not entirely trivial classifiers. In its original form, the boosting question begins by assuming that a given class $\mathcal{C}$ is weakly PAC learnable as defined above. With such an assumption, as we have seen, it is possible to prove strong absolute guarantees on the PAC learnability of $\mathcal{C}$, ensuring near perfect generalization.

However, in practical settings, this assumption may be too onerous, requiring that the labels be deterministic according to a target function from a known class $\mathcal{C}$, that weak learning hold for every distribution, and that the edge γ be known ahead of time. Practically, these requirements can be very difficult to check or guarantee. As we now discuss, there are many ways in which these assumptions can be weakened, and most (but not all) of the book is founded on such relaxed versions of the weak learning assumption.

To begin, for the sake of generality we can usually drop any explicit assumptions about the data, returning to the more agnostic framework seen earlier in the chapter in which

labeled examples (x, y) are generated by an arbitrary distribution $\mathcal{D}$ with no functional dependence assumed for the labels y. In this case, as in chapter 1, the training set consists simply of labeled pairs $(x_1, y_1), \ldots, (x_m, y_m)$.

Furthermore, rather than the far-reaching assumption of weak PAC learnability described above, we can instead assume only that the weak hypotheses found by the given weak learning algorithm A have weighted *training* error bounded away from $\frac{1}{2}$. This leads to a different and weaker notion of weak learnability that is particular to the actual training set. Specifically, we say that the *empirical γ-weak learning assumption* holds if for any distribution D on the indices $\{1, \ldots, m\}$ of the training examples, the weak learning algorithm A is able to find a hypothesis h with weighted *training* error at most $\frac{1}{2} - \gamma$:

$$\mathbf{Pr}_{i \sim D}[h(x_i) \neq y_i] \leq \tfrac{1}{2} - \gamma.$$

Thus, empirical weak learnability is defined with respect to distributions defined on a particular training set with particular labels, while weak PAC learnability is defined with respect to any distribution on the entire domain $\mathcal{X}$, and any labeling consistent with one of the targets in the class $\mathcal{C}$. These two notions are clearly related, but are also quite distinct. Even so, when clear from context, we often use shortened terminology, such as "weak learning assumption" and "weakly learnable," omitting "PAC" or "empirical."

This condition can be weakened still further. Rather than assuming that the weak learner A can achieve a training error of $\frac{1}{2} - \gamma$ on *every* distribution on the training set, we can assume only that this happens on the *particular* distributions on which it is actually trained during boosting. In the notation of algorithm 1.1 (p. 5), this means, for AdaBoost, simply that $\epsilon_t \leq \frac{1}{2} - \gamma$ for all t, for some $\gamma > 0$. This property clearly follows from the empirical γ-weak learning assumption, and also follows from the weak PAC learnability assumption, as will be seen in section 4.3. Furthermore, in comparison to what was earlier assumed regarding the weak PAC learnability of a known class $\mathcal{C}$, this assumption is quite benign, and can be verified immediately in an actual learning setting.

We can even go one step farther and drop the assumption that the edge γ is known; as discussed in chapter 1, this is the defining property of a boosting algorithm that is *adaptive*, such as AdaBoost.

In the resulting fully relaxed framework, no assumptions at all are made about the data, analogous to the agnostic approach taken earlier in the chapter. Rather, generalization performance is analyzed in terms of the weighted training errors ϵ_t, as well as other relevant parameters, such as the complexity of the weak hypotheses. Although the ϵ_t's are not explicitly assumed to be bounded below $\frac{1}{2}$, when this is the case, such bounds should imply high generalization accuracy. Since it is the most general and practical, we will mostly follow this mode of analysis, particularly in chapters 4 and 5, where bounds of just this form are proved.

Summary

This chapter has reviewed some fundamental results at the foundation of theoretical machine learning, tools we will use extensively in later chapters in the analysis of boosting algorithms. These general results formalize our intuition of the requirements for learning: sufficient data, low training error, and simplicity, the last of these being measurable in various ways, including description length, VC-dimension, and degree of compressibility. This understanding also captures the essential problem of learning, namely, the careful balancing of the trade-off between fit to training data and simplicity. Finally, we looked at the formal PAC learning framework and the notion of weak learnability from which arises the basic question of the existence of boosting algorithms.

With this foundation, we can begin our analysis of AdaBoost.

Bibliographic Notes

The overall approach to machine learning that we have adopted in this chapter, particularly in sections 2.1 and 2.2, was initially pioneered in the groundbreaking work of Vapnik and Chervonenkis [225, 226]. For a more complete treatment, see, for instance, the books by Vapnik [222, 223, 224], and by Devroye, Györfi, and Lugosi [67]. The approach and analysis described in sections 2.2.3 and 2.2.4, including theorems 2.3, 2.5, and 2.6, lemma 2.4, and the VC-dimension as applied in this setting, are all due to Vapnik and Chervonenkis [225]. However, the constants that appear in their versions of these theorems are slightly different from what we have given here, which are based instead on theorem 12.5 of Devroye, Györfi, and Lugosi [67]. This latter source includes an overview of some of the other versions that have been proved, some of which have better constants. Also, lemma 2.4 was proved independently by Sauer [198]. A short proof of equation (2.12) is given by Kearns and Vazirani [134]. Examples of lower bounds on learning in terms of the VC-dimension include the work of Ehrenfeucht et al. [80] and Gentile and Helmbold [107]. Hoeffding's inequality (theorem 2.1) is due to Hoeffding [123]. Regarding theorem 2.7, equation (2.17) was proved by Vapnik and Chervonenkis [226], and later by Blumer et al. [27]. Equation (2.18) was proved by Blumer et al. [28].

Further background on the generative (or Bayesian) approach described in section 2.1.2 can be found, for instance, in Duda, Hart, and Stork [73], and Bishop [22].

The approach given in section 2.2.6, including theorem 2.8, is due to Littlestone and Warmuth [154], and Floyd and Warmuth [85]. The minimum description length principle, though not discussed in this book, is the foundation for another, related approach to learning and statistics which is also based on compression—see, for instance, Grünwald's book [112].

The PAC learning model of section 2.3.1 was proposed by Valiant [221]. Further background can be found in Kearns and Vazirani's book [134]. The notion of weak PAC learning

in section 2.3.2 was first considered by Kearns and Valiant [133]. This latter work also contains examples of learning problems which are computationally intractable even when provided with a statistically adequate amount of data, as mentioned in section 2.3.1.

Some of the exercises in this chapter are based on material from [28, 154, 198, 221, 225].

Exercises

2.1 For any hypothesis space $\mathcal{H}$, let $\hat{h}$ minimize the training error, and let h^* minimize the generalization error:

$$\hat{h} \doteq \arg\min_{h \in \mathcal{H}} \widehat{\mathrm{err}}(h)$$

$$h^* \doteq \arg\min_{h \in \mathcal{H}} \mathrm{err}(h).$$

Note that $\hat{h}$ depends implicitly on the training set. Prove that

$$\mathbf{E}\left[\widehat{\mathrm{err}}(\hat{h})\right] \leq \mathrm{err}(h^*) \leq \mathbf{E}\left[\mathrm{err}(\hat{h})\right]$$

(where expectation is with respect to the choice of the random training examples).

2.2 Show that the VC-dimension of any finite hypothesis space $\mathcal{H}$ is at most $\lg |\mathcal{H}|$. Also, show that this bound is tight; that is, for every $d \geq 1$, give an example of a hypothesis space $\mathcal{H}$ with VC-dimension equal to d for which $d = \lg |\mathcal{H}|$.

2.3 Let $\mathcal{X} = \{0, 1\}^n$, and let $\mathcal{C}$ be the space of Boolean monomials, that is, functions of the form

$$c(\mathbf{x}) = \begin{cases} +1 & \text{if } \prod_{j \in R} x_j = 1 \\ -1 & \text{else} \end{cases}$$

for some $R \subseteq \{1, \ldots, n\}$. In other words, $c(\mathbf{x}) = +1$ if and only if all of the variables $x_j = 1$, for all $j \in R$. Show that $\mathcal{C}$ is efficiently PAC learnable. That is, describe an efficient (polynomial-time) algorithm for finding a monomial consistent with any dataset (assuming one exists), and show that the PAC criterion (equation (2.21)) holds for a sample of size polynomial in $1/\epsilon$, $1/\delta$, and n.

2.4 Let $\mathcal{X} = \mathbb{R}^n$, and let $\mathcal{H}$ be the space of hypotheses defined by *axis-aligned rectangles*, that is, functions of the form

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } a_j \leq x_j \leq b_j \text{ for all } j = 1, \ldots, n \\ -1 & \text{else} \end{cases}$$

for some $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{R}$. Compute exactly the VC-dimension of $\mathcal{H}$.

2.5 Let $\mathcal{X} = \mathbb{R}^n$, and let $\mathcal{C}$ be the space of functions defined by axis-aligned rectangles, as in exercise 2.4. Show that $\mathcal{C}$ is efficiently PAC learnable (as in exercise 2.3), using a compression scheme.

2.6 Let $\mathcal{X} = \mathbb{R}$, and let $\mathcal{C}$ be the space of functions defined by unions of at most n intervals, that is, functions of the form

$$c(x) = \begin{cases} +1 & \text{if } x \in [a_1, b_1] \cup \cdots \cup [a_n, b_n] \\ -1 & \text{else} \end{cases}$$

for some $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{R}$.

a. Compute the VC-dimension of $\mathcal{C}$ exactly.

b. Use the result from part (a) to show that $\mathcal{C}$ is efficiently PAC learnable (as in exercise 2.3).

2.7 Show that Sauer's lemma (lemma 2.4) is tight. That is, for every $d \geq 1$, give an example of a space $\mathcal{H}$ with VC-dimension equal to d such that for each m,

$$\Pi_{\mathcal{H}}(m) = \sum_{i=0}^{d} \binom{m}{i}.$$

2.8 Let $\mathcal{H}$ be a countably infinite space of hypotheses. Let $g : \mathcal{H} \to (0, 1]$ be any function such that

$$\sum_{h \in \mathcal{H}} g(h) \leq 1.$$

Although g may look a bit like a probability distribution, it is just a function—any function—whose positive values happen to add up to a number not bigger than 1. Assume a random training set of size m has been chosen.

a. Prove that, with probability at least $1 - \delta$,

$$\text{err}(h) \leq \widehat{\text{err}}(h) + \sqrt{\frac{\ln(1/g(h)) + \ln(1/\delta)}{2m}}$$

for all $h \in \mathcal{H}$.

b. Suppose hypotheses in $\mathcal{H}$ are represented by bit strings and that $|h|$ denotes the number of bits needed to represent h. Show how to choose g to prove that, with probability at least $1 - \delta$,

$$\text{err}(h) \leq \widehat{\text{err}}(h) + O\left(\sqrt{\frac{|h| + \ln(1/\delta)}{m}}\right)$$

for all $h \in \mathcal{H}$.

c. How does the bound in part (b) reflect the intuitive trade-off between fit to data and simplicity?

2.9 Show that theorems 2.3 and 2.6 are equivalent. That is:

a. For $\mathcal{A}$ constructed as in equation (2.14), verify equation (2.15), and show that theorem 2.6 yields exactly theorem 2.3.

b. For a general family $\mathcal{A}$ of subsets, show that theorem 2.3 can be used to prove theorem 2.6.

2.10 Let the domain be $\mathcal{X}_n = \mathbb{R}^n$, and let $\mathcal{H}_n$ be the space of all decision stumps of the (simplified) form

$$h(\mathbf{x}) = \begin{cases} c_0 & \text{if } x_k \leq v \\ c_1 & \text{if } x_k > v \end{cases}$$

for some $c_0, c_1 \in \{-1, +1\}$, $k \in \{1, \ldots, n\}$, and $v \in \mathbb{R}$. (In section 3.4.2, we will consider decision stumps in greater generality.)

a. Show that $\Pi_{\mathcal{H}_n}(m) \leq 2nm$.

b. Show that there exist positive constants a and b such that for all $n \geq 1$, the VC-dimension of $\mathcal{H}_n$ is at most $a + b \ln n$.