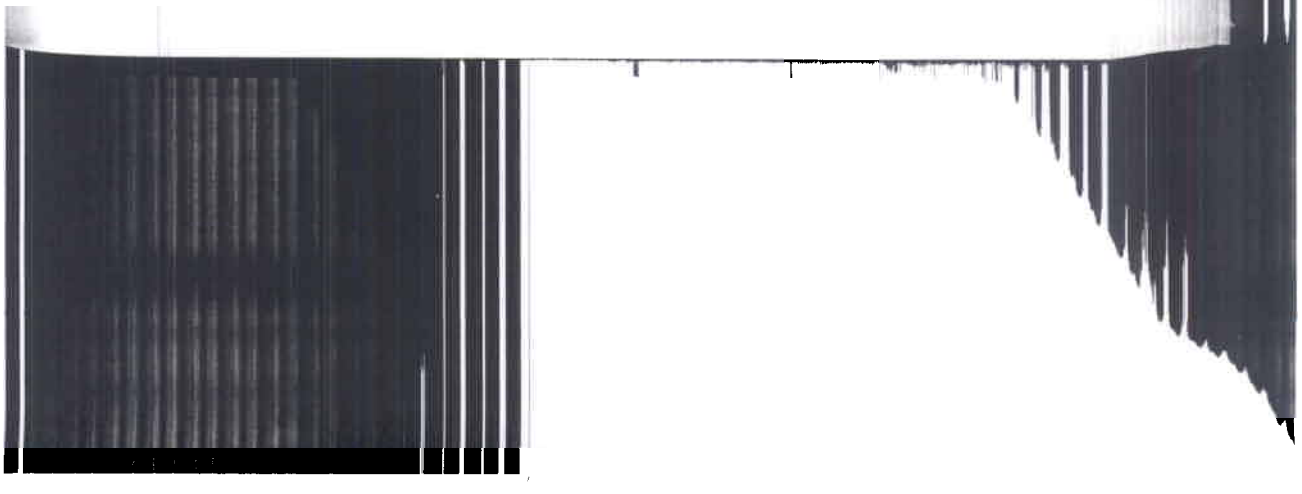
CHAPTER 7

COMPUTATIONAL LEARNING THEORY

This chapter presents a theoretical characterization of the difficulty of several types of machine learning problems and the capabilities of several types of machine learning algorithms. This theory seeks to answer questions such as "Under what conditions is successful learning possible and impossible?" and "Under what conditions is a particular learning algorithm assured of learning successfully?" Two specific frameworks for analyzing learning algorithms are considered. Within the probably approximately correct (PAC) framework, we identify classes of hypotheses that can and cannot be learned from a polynomial number of training examples and we define a natural measure of complexity for hypothesis spaces that allows bounding the number of training examples required for inductive learning. Within the mistake bound framework, we examine the number of training errors that will be made by a learner before it determines the correct hypothesis.

7.1 INTRODUCTION

When studying machine learning it is natural to wonder what general laws may govern machine (and nonmachine) learners. Is it possible to identify classes of learning problems that are inherently difficult or easy, independent of the learning algorithm? Can one characterize the number of training examples necessary or sufficient to assure successful learning? How is this number affected if the learner is allowed to pose queries to the trainer, versus observing a random sample of training examples? Can one characterize the number of mistakes that a learner



will make before learning the target function? Can one characterize the inherent computational complexity of classes of learning problems?

Although general answers to all these questions are not yet known, fragments of a computational theory of learning have begun to emerge. This chapter presents key results from this theory, providing answers to these questions within particular problem settings. We focus here on the problem of inductively learning an unknown target function, given only training examples of this target function and a space of candidate hypotheses. Within this setting, we will be chiefly concerned with questions such as how many training examples are sufficient to successfully learn the target function, and how many mistakes will the learner make before succeeding. As we shall see, it is possible to set quantitative bounds on these measures, depending on attributes of the learning problem such as:

- the size or complexity of the hypothesis space considered by the learner
- the accuracy to which the target concept must be approximated
- the probability that the learner will output a successful hypothesis
- the manner in which training examples are presented to the learner

For the most part, we will focus not on individual learning algorithms, but rather on broad classes of learning algorithms characterized by the hypothesis spaces they consider, the presentation of training examples, etc. Our goal is to answer questions such as:

- *Sample complexity.* How many training examples are needed for a learner to converge (with high probability) to a successful hypothesis?
- *Computational complexity.* How much computational effort is needed for a learner to converge (with high probability) to a successful hypothesis?
- *Mistake bound.* How many training examples will the learner misclassify before converging to a successful hypothesis?

Note there are many specific settings in which we could pursue such questions. For example, there are various ways to specify what it means for the learner to be "successful." We might specify that to succeed, the learner must output a hypothesis identical to the target concept. Alternatively, we might simply require that it output a hypothesis that agrees with the target concept most of the time, or that it usually output such a hypothesis. Similarly, we must specify how training examples are to be obtained by the learner. We might specify that training examples are presented by a helpful teacher, or obtained by the learner performing experiments, or simply generated at random according to some process outside the learner's control. As we might expect, the answers to the above questions depend on the particular setting, or learning model, we have in mind.

The remainder of this chapter is organized as follows. Section 7.2 introduces the probably approximately correct (PAC) learning setting. Section 7.3 then analyzes the sample complexity and computational complexity for several learning

problems within this PAC setting. Section 7.4 introduces an important measure of hypothesis space complexity called the VC-dimension and extends our PAC analysis to problems in which the hypothesis space is infinite. Section 7.5 introduces the mistake-bound model and provides a bound on the number of mistakes made by several learning algorithms discussed in earlier chapters. Finally, we introduce the WEIGHTED-MAJORITY algorithm, a practical algorithm for combining the predictions of multiple competing learning algorithms, along with a theoretical mistake bound for this algorithm.

7.2 PROBABLY LEARNING AN APPROXIMATELY CORRECT HYPOTHESIS

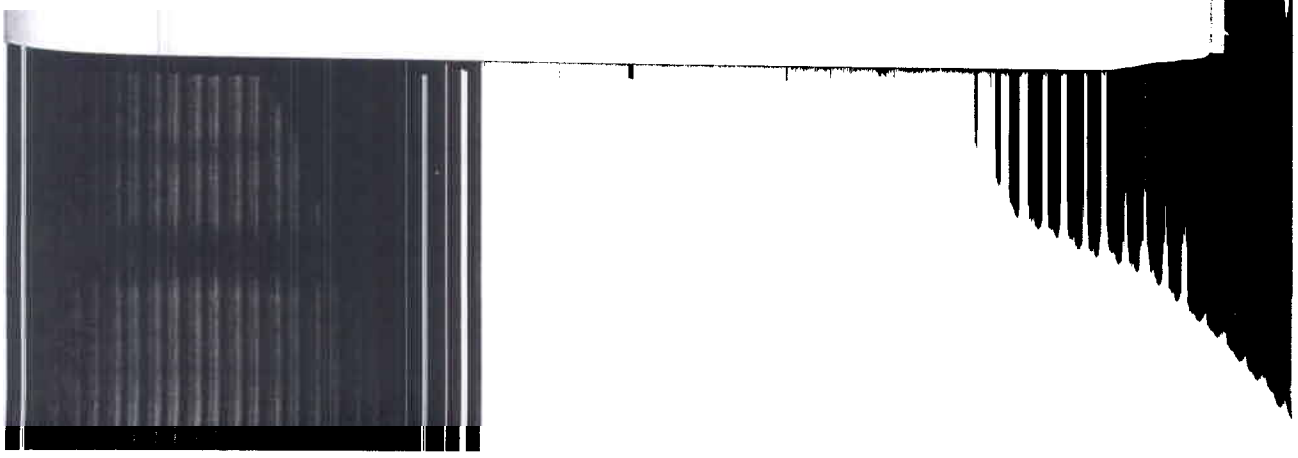
In this section we consider a particular setting for the learning problem, called the *probably approximately correct* (PAC) learning model. We begin by specifying the problem setting that defines the PAC learning model, then consider the questions of how many training examples and how much computation are required in order to learn various classes of target functions within this PAC model. For the sake of simplicity, we restrict the discussion to the case of learning boolean-valued concepts from noise-free training data. However, many of the results can be extended to the more general scenario of learning real-valued target functions (see, for example, Natarajan 1991), and some can be extended to learning from certain types of noisy data (see, for example, Laird 1988; Kearns and Vazirani 1994).

7.2.1 The Problem Setting

As in earlier chapters, let X refer to the set of all possible instances over which target functions may be defined. For example, X might represent the set of all people, each described by the attributes *age* (e.g., *young* or *old*) and *height* (*short* or *tall*). Let C refer to some set of target concepts that our learner might be called upon to learn. Each target concept c in C corresponds to some subset of X, or equivalently to some boolean-valued function $c : X \rightarrow \{0, 1\}$. For example, one target concept c in C might be the concept "people who are skiers." If x is a positive example of c, then we will write $c(x) = 1$; if x is a negative example, $c(x) = 0$.

We assume instances are generated at random from X according to some probability distribution $\mathcal{D}$. For example, $\mathcal{D}$ might be the distribution of instances generated by observing people who walk out of the largest sports store in Switzerland. In general, $\mathcal{D}$ may be any distribution, and it will not generally be known to the learner. All that we require of $\mathcal{D}$ is that it be stationary; that is, that the distribution not change over time. Training examples are generated by drawing an instance x at random according to $\mathcal{D}$, then presenting x along with its target value, $c(x)$, to the learner.

The learner L considers some set H of possible hypotheses when attempting to learn the target concept. For example, H might be the set of all hypotheses



describable by conjunctions of the attributes *age* and *height*. After observing a sequence of training examples of the target concept c, L must output some hypothesis h from H, which is its estimate of c. To be fair, we evaluate the success of L by the performance of h over new instances drawn randomly from X according to $\mathcal{D}$, the same probability distribution used to generate the training data.

Within this setting, we are interested in characterizing the performance of various learners L using various hypothesis spaces H, when learning individual target concepts drawn from various classes C. Because we demand that L be general enough to learn any target concept from C regardless of the distribution of training examples, we will often be interested in worst-case analyses over all possible target concepts from C and all possible instance distributions $\mathcal{D}$.

7.2.2 Error of a Hypothesis

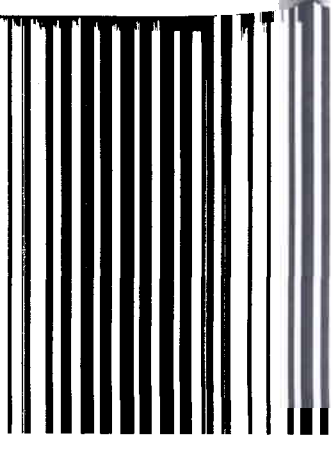
Because we are interested in how closely the learner's output hypothesis h approximates the actual target concept c, let us begin by defining the *true error* of a hypothesis h with respect to target concept c and instance distribution $\mathcal{D}$. Informally, the true error of h is just the error rate we expect when applying h to future instances drawn according to the probability distribution $\mathcal{D}$. In fact, we already defined the true error of h in Chapter 5. For convenience, we restate the definition here using c to represent the boolean target function.

> ***Definition:*** The **true error** (denoted $error_{\mathcal{D}}(h)$) of hypothesis h with respect to target concept c and distribution $\mathcal{D}$ is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$.
>
> $$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$

Here the notation $\Pr_{x \in \mathcal{D}}$ indicates that the probability is taken over the instance distribution $\mathcal{D}$.

Figure 7.1 shows this definition of error in graphical form. The concepts c and h are depicted by the sets of instances within X that they label as positive. The error of h with respect to c is the probability that a randomly drawn instance will fall into the region where h and c disagree (i.e., their set difference). Note we have chosen to define error over the *entire distribution* of instances—not simply over the training examples—because this is the true error we expect to encounter when actually using the learned hypothesis h on subsequent instances drawn from $\mathcal{D}$.

Note that error depends strongly on the unknown probability distribution $\mathcal{D}$. For example, if $\mathcal{D}$ is a uniform probability distribution that assigns the same probability to every instance in X, then the error for the hypothesis in Figure 7.1 will be the fraction of the total instance space that falls into the region where h and c disagree. However, the same h and c will have a much higher error if $\mathcal{D}$ happens to assign very high probability to instances for which h and c disagree. In the extreme, if $\mathcal{D}$ happens to assign zero probability to the instances for which



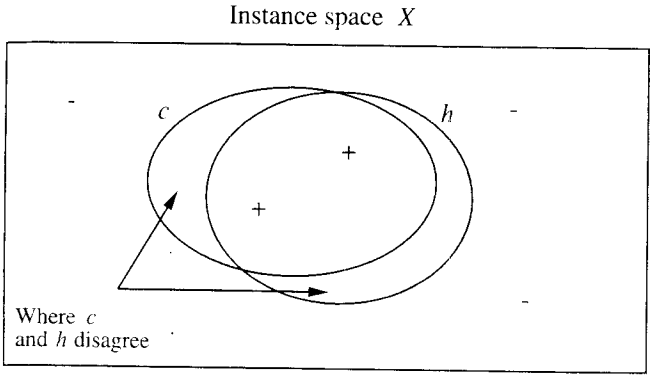


FIGURE 7.1
The error of hypothesis h with respect to target concept c. The error of h with respect to c is the probability that a randomly drawn instance will fall into the region where h and c disagree on its classification. The + and − points indicate positive and negative training examples. Note h has a nonzero error with respect to c despite the fact that h and c agree on all five training examples observed thus far.

$h(x) = c(x)$, then the error for the h in Figure 7.1 will be 1, despite the fact the h and c agree on a very large number of (zero probability) instances.

Finally, note that the error of h with respect to c is not directly observable to the learner. L can only observe the performance of h over the *training examples*, and it must choose its output hypothesis on this basis only. We will use the term *training error* to refer to the fraction of training examples misclassified by h, in contrast to the *true error* defined above. Much of our analysis of the complexity of learning centers around the question "how probable is it that the observed *training error* for h gives a misleading estimate of the *true* $error_{\mathcal{D}}(h)$?"

Notice the close relationship between this question and the questions considered in Chapter 5. Recall that in Chapter 5 we defined the *sample error* of h with respect to a set S of examples to be the fraction of S misclassified by h. The training error defined above is just the sample error when S is the set of training examples. In Chapter 5 we determined the probability that the sample error will provide a misleading estimate of the true error, under the assumption that the data sample S is drawn independent of h. However, when S is the set of training data, the learned hypothesis h depends very much on S! Therefore, in this chapter we provide an analysis that addresses this important special case.

7.2.3 PAC Learnability

Our aim is to characterize classes of target concepts that can be reliably learned from a reasonable number of randomly drawn training examples and a reasonable amount of computation.

What kinds of statements about learnability should we guess hold true? We might try to characterize the number of training examples needed to learn

a hypothesis h for which $error_{\mathcal{D}}(h) = 0$. Unfortunately, it turns out this is futile in the setting we are considering, for two reasons. First, unless we provide training examples corresponding to every possible instance in X (an unrealistic assumption), there may be multiple hypotheses consistent with the provided training examples, and the learner cannot be certain to pick the one corresponding to the target concept. Second, given that the training examples are drawn randomly, there will always be some nonzero probability that the training examples encountered by the learner will be misleading. (For example, although we might frequently see skiers of different heights, on any given day there is some small chance that all observed training examples will happen to be 2 meters tall.)

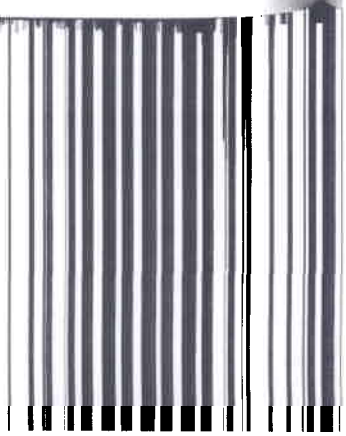
To accommodate these two difficulties, we weaken our demands on the learner in two ways. First, we will not require that the learner output a zero error hypothesis—we will require only that its error be bounded by some constant, ϵ, that can be made arbitrarily small. Second, we will not require that the learner succeed for *every* sequence of randomly drawn training examples—we will require only that its probability of failure be bounded by some constant, δ, that can be made arbitrarily small. In short, we require only that the learner *probably* learn a hypothesis that is *approximately correct*—hence the term probably approximately correct learning, or PAC learning for short.

Consider some class C of possible target concepts and a learner L using hypothesis space H. Loosely speaking, we will say that the concept class C is PAC-learnable by L using H if, for any target concept c in C, L will with probability $(1 - \delta)$ output a hypothesis h with $error_{\mathcal{D}}(h) < \epsilon$, after observing a reasonable number of training examples and performing a reasonable amount of computation. More precisely,

> ***Definition:*** Consider a concept class C defined over a set of instances X of length n and a learner L using hypothesis space H. C is **PAC-learnable** by L using H if for all $c \in C$, distributions $\mathcal{D}$ over X, ϵ such that $0 < \epsilon < 1/2$, and δ such that $0 < \delta < 1/2$, learner L will with probability at least $(1 - \delta)$ output a hypothesis $h \in H$ such that $error_{\mathcal{D}}(h) \leq \epsilon$, in time that is polynomial in $1/\epsilon$, $1/\delta$, n, and $size(c)$.

Our definition requires two things from L. First, L must, with arbitrarily high probability $(1 - \delta)$, output a hypothesis having arbitrarily low error (ϵ). Second, it must do so efficiently—in time that grows at most polynomially with $1/\epsilon$ and $1/\delta$, which define the strength of our demands on the output hypothesis, and with n and $size(c)$ that define the inherent complexity of the underlying instance space X and concept class C. Here, n is the size of instances in X. For example, if instances in X are conjunctions of k boolean features, then $n = k$. The second space parameter, $size(c)$, is the encoding length of c in C, assuming some representation for C. For example, if concepts in C are conjunctions of up to k boolean features, each described by listing the indices of the features in the conjunction, then $size(c)$ is the number of boolean features actually used to describe c.

Our definition of PAC learning may at first appear to be concerned only with the computational resources required for learning, whereas in practice we are



usually more concerned with the number of training examples required. However, the two are very closely related: If L requires some minimum processing time per training example, then for C to be PAC-learnable by L, L *must learn from a polynomial number of training examples.* In fact, a typical approach to showing that some class C of target concepts is PAC-learnable, is to first show that each target concept in C can be learned from a polynomial number of training examples and then show that the processing time per example is also polynomially bounded.

Before moving on, we should point out a restrictive assumption implicit in our definition of PAC-learnable. This definition implicitly assumes that the learner's hypothesis space H contains a hypothesis with arbitrarily small error for every target concept in C. This follows from the requirement in the above definition that the learner succeed when the error bound ϵ is arbitrarily close to zero. Of course this is difficult to assure if one does not know C in advance (what is C for a program that must learn to recognize faces from images?), unless H is taken to be the power set of X. As pointed out in Chapter 2, such an unbiased H will not support accurate generalization from a reasonable number of training examples. Nevertheless, the results based on the PAC learning model provide useful insights regarding the relative complexity of different learning problems and regarding the rate at which generalization accuracy improves with additional training examples. Furthermore, in Section 7.3.1 we will lift this restrictive assumption, to consider the case in which the learner makes no prior assumption about the form of the target concept.

7.3 SAMPLE COMPLEXITY FOR FINITE HYPOTHESIS SPACES

As noted above, PAC-learnability is largely determined by the number of training examples required by the learner. The growth in the number of required training examples with problem size, called the *sample complexity* of the learning problem, is the characteristic that is usually of greatest interest. The reason is that in most practical settings the factor that most limits success of the learner is the limited availability of training data.

Here we present a general bound on the sample complexity for a very broad class of learners, called *consistent learners*. A learner is *consistent* if it outputs hypotheses that perfectly fit the training data, whenever possible. It is quite reasonable to ask that a learning algorithm be consistent, given that we typically prefer a hypothesis that fits the training data over one that does not. Note that many of the learning algorithms discussed in earlier chapters, including all the learning algorithms described in Chapter 2, are consistent learners.

Can we derive a bound on the number of training examples required by *any* consistent learner, independent of the specific algorithm it uses to derive a consistent hypothesis? The answer is yes. To accomplish this, it is useful to recall the definition of version space from Chapter 2. There we defined the version space, $VS_{H,D}$, to be the set of all hypotheses $h \in H$ that correctly classify the training examples D.

$$VS_{H,D} = \{h \in H | (\forall \langle x, c(x) \rangle \in D)\ (h(x) = c(x))\}$$

The significance of the version space here is that *every consistent learner outputs a hypothesis belonging to the version space*, regardless of the instance space X, hypothesis space H, or training data D. The reason is simply that by definition the version space $VS_{H,D}$ contains every consistent hypothesis in H. Therefore, *to bound the number of examples needed by any consistent learner, we need only bound the number of examples needed to assure that the version space contains no unacceptable hypotheses.* The following definition, after Haussler (1988), states this condition precisely.

> ***Definition:*** Consider a hypothesis space H, target concept c, instance distribution $\mathcal{D}$, and set of training examples D of c. The version space $VS_{H,D}$ is said to be **ϵ-exhausted** with respect to c and $\mathcal{D}$, if every hypothesis h in $VS_{H,D}$ has error less than ϵ with respect to c and $\mathcal{D}$.
>
> $$(\forall h \in VS_{H,D})\ error_{\mathcal{D}}(h) < \epsilon$$

This definition is illustrated in Figure 7.2. The version space is ϵ-exhausted just in the case that all the hypotheses consistent with the observed training examples (i.e., those with zero training error) happen to have true error less than ϵ. Of course from the learner's viewpoint all that can be known is that these hypotheses fit the training data equally well—they all have zero training error. Only an observer who knew the identity of the target concept could determine with certainty whether the version space is ϵ-exhausted. Surprisingly, a probabilistic argument allows us to bound the probability that the version space will be ϵ-exhausted after a given number of training examples, even without knowing the identity of the target concept or the distribution from which training examples

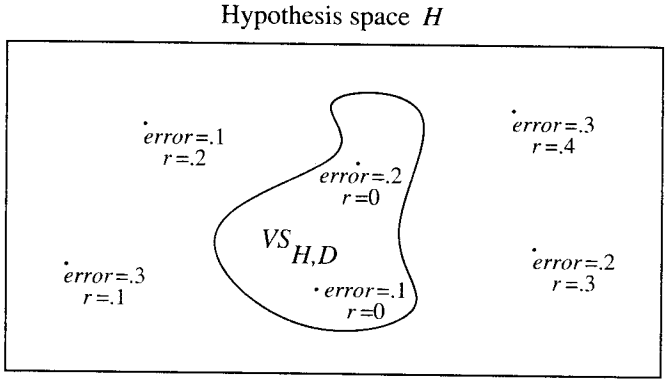


FIGURE 7.2
Exhausting the version space. The version space $VS_{H,D}$ is the subset of hypotheses $h \in H$, which have zero training error (denoted by $r = 0$ in the figure). Of course the true $error_{\mathcal{D}}(h)$ (denoted by *error* in the figure) may be nonzero, even for hypotheses that commit zero errors over the training data. The version space is said to be ϵ-exhausted when all hypotheses h remaining in $VS_{H,D}$ have $error_{\mathcal{D}}(h) < \epsilon$.

are drawn. Haussler (1988) provides such a bound, in the form of the following theorem.

Theorem 7.1. ϵ-exhausting the version space. If the hypothesis space H is finite, and D is a sequence of $m \geq 1$ independent randomly drawn examples of some target concept c, then for any $0 \leq \epsilon \leq 1$, the probability that the version space $VS_{H,D}$ is not ϵ-exhausted (with respect to c) is less than or equal to

$$|H|e^{-\epsilon m}$$

Proof. Let $h_1, h_2, \ldots h_k$ be all the hypotheses in H that have true error greater than ϵ with respect to c. We fail to ϵ-exhaust the version space if and only if at least one of these k hypotheses happens to be consistent with all m independent random training examples. The probability that any single hypothesis having true error greater than ϵ would be consistent with one randomly drawn example is at most $(1-\epsilon)$. Therefore the probability that this hypothesis will be consistent with m independently drawn examples is at most $(1-\epsilon)^m$. Given that we have k hypotheses with error greater than ϵ, the probability that at least one of these will be consistent with all m training examples is at most

$$k(1-\epsilon)^m$$

And since $k \leq |H|$, this is at most $|H|(1-\epsilon)^m$. Finally, we use a general inequality stating that if $0 \leq \epsilon \leq 1$ then $(1-\epsilon) \leq e^{-\epsilon}$. Thus,

$$k(1-\epsilon)^m \leq |H|(1-\epsilon)^m \leq |H|e^{-\epsilon m}$$

which proves the theorem. □

We have just proved an upper bound on the probability that the version space is not ϵ-exhausted, based on the number of training examples m, the allowed error ϵ, and the size of H. Put another way, this bounds the probability that m training examples will fail to eliminate all "bad" hypotheses (i.e., hypotheses with true error greater than ϵ), for any consistent learner using hypothesis space H.

Let us use this result to determine the number of training examples required to reduce this probability of failure below some desired level δ.

$$|H|e^{-\epsilon m} \leq \delta \tag{7.1}$$

Rearranging terms to solve for m, we find

$$m \geq \frac{1}{\epsilon}(\ln|H| + \ln(1/\delta)) \tag{7.2}$$

To summarize, the inequality shown in Equation (7.2) provides a general bound on the number of training examples sufficient for *any consistent learner* to successfully learn any target concept in H, for any desired values of δ and ϵ. This number m of training examples is sufficient to assure that any consistent hypothesis will be probably (with probability $(1-\delta)$) approximately (within error ϵ) correct. Notice m grows linearly in $1/\epsilon$ and logarithmically in $1/\delta$. It also grows logarithmically in the size of the hypothesis space H.

Note that the above bound can be a substantial overestimate. For example, although the probability of failing to exhaust the version space must lie in the interval $[0, 1]$, the bound given by the theorem grows linearly with $|H|$. For sufficiently large hypothesis spaces, this bound can easily be greater than one. As a result, the bound given by the inequality in Equation (7.2) can substantially overestimate the number of training examples required. The weakness of this bound is mainly due to the $|H|$ term, which arises in the proof when summing the probability that a single hypothesis could be unacceptable, over all possible hypotheses. In fact, a much tighter bound is possible in many cases, as well as a bound that covers infinitely large hypothesis spaces. This will be the subject of Section 7.4.

7.3.1 Agnostic Learning and Inconsistent Hypotheses

Equation (7.2) is important because it tells us how many training examples suffice to ensure (with probability $(1-\delta)$) that every hypothesis in H having zero training error will have a true error of at most ϵ. Unfortunately, if H does not contain the target concept c, then a zero-error hypothesis cannot always be found. In this case, the most we might ask of our learner is to output the hypothesis from H that has the *minimum* error over the training examples. A learner that makes no assumption that the target concept is representable by H and that simply finds the hypothesis with minimum training error, is often called an *agnostic* learner, because it makes no prior commitment about whether or not $C \subseteq H$.

Although Equation (7.2) is based on the assumption that the learner outputs a zero-error hypothesis, a similar bound can be found for this more general case in which the learner entertains hypotheses with nonzero training error. To state this precisely, let D denote the particular set of training examples available to the learner, in contrast to $\mathcal{D}$, which denotes the probability distribution over the entire set of instances. Let $error_D(h)$ denote the training error of hypothesis h. In particular, $error_D(h)$ is defined as the fraction of the training examples in D that are misclassified by h. Note the $error_D(h)$ over the particular sample of training data D may differ from the true error $error_{\mathcal{D}}(h)$ over the entire probability distribution $\mathcal{D}$. Now let h_{best} denote the hypothesis from H having lowest training error over the training examples. How many training examples suffice to ensure (with high probability) that its true error $error_{\mathcal{D}}(h_{best})$ will be no more than $\epsilon + error_D(h_{best})$? Notice the question considered in the previous section is just a special case of this question, when $error_D(h_{best})$ happens to be zero.

This question can be answered (see Exercise 7.3) using an argument analogous to the proof of Theorem 7.1. It is useful here to invoke the general Hoeffding bounds (sometimes called the additive Chernoff bounds). The Hoeffding bounds characterize the deviation between the true probability of some event and its observed frequency over m independent trials. More precisely, these bounds apply to experiments involving m distinct Bernoulli trials (e.g., m independent flips of a coin with some probability of turning up heads). This is exactly analogous to the setting we consider when estimating the error of a hypothesis in Chapter 5: The