2 The PAC Learning Framework

Several fundamental questions arise when designing and analyzing algorithms that learn from examples: What can be learned efficiently? What is inherently hard to learn? How many examples are needed to learn successfully? Is there a general model of learning? In this chapter, we begin to formalize and address these questions by introducing the *Probably Approximately Correct* (PAC) learning framework. The PAC framework helps define the class of learnable concepts in terms of the number of sample points needed to achieve an approximate solution, *sample complexity*, and the time and space complexity of the learning algorithm, which depends on the cost of the computational representation of the concepts.

We first describe the PAC framework and illustrate it, then present some general learning guarantees within this framework when the hypothesis set used is finite, both for the *consistent* case where the hypothesis set used contains the concept to learn and for the opposite *inconsistent* case.

2.1 The PAC learning model

We first introduce several definitions and the notation needed to present the PAC model, which will also be used throughout much of this book.

We denote by $\mathcal{X}$ the set of all possible *examples* or *instances*. $\mathcal{X}$ is also sometimes referred to as the *input space*. The set of all possible *labels* or *target values* is denoted by $\mathcal{Y}$. For the purpose of this introductory chapter, we will limit ourselves to the case where $\mathcal{Y}$ is reduced to two labels, $\mathcal{Y} = \{0, 1\}$, which corresponds to the so-called *binary classification*. Later chapters will extend these results to more general settings.

A *concept* $c\colon \mathcal{X} \to \mathcal{Y}$ is a mapping from $\mathcal{X}$ to $\mathcal{Y}$. Since $\mathcal{Y} = \{0, 1\}$, we can identify c with the subset of $\mathcal{X}$ over which it takes the value 1. Thus, in the following, we equivalently refer to a concept to learn as a mapping from $\mathcal{X}$ to $\{0, 1\}$, or as a subset of $\mathcal{X}$. As an example, a concept may be the set of points inside a triangle

or the indicator function of these points. In such cases, we will say in short that the concept to learn is a triangle. A *concept class* is a set of concepts we may wish to learn and is denoted by $\mathcal{C}$. This could, for example, be the set of all triangles in the plane.

We assume that examples are independently and identically distributed (i.i.d.) according to some fixed but unknown distribution $\mathcal{D}$. The learning problem is then formulated as follows. The learner considers a fixed set of possible concepts $\mathcal{H}$, called a *hypothesis set*, which might not necessarily coincide with $\mathcal{C}$. It receives a sample $S = (x_1, \ldots, x_m)$ drawn i.i.d. according to $\mathcal{D}$ as well as the labels $(c(x_1), \ldots, c(x_m))$, which are based on a specific target concept $c \in \mathcal{C}$ to learn. The task is then to use the labeled sample S to select a hypothesis $h_S \in \mathcal{H}$ that has a small *generalization error* with respect to the concept c. The generalization error of a hypothesis $h \in \mathcal{H}$, also referred to as the *risk* or *true error* (or simply *error*) of h is denoted by $R(h)$ and defined as follows.[1]

Definition 2.1 (Generalization error) *Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and an underlying distribution $\mathcal{D}$, the* generalization error *or* risk *of h is defined by*

$$R(h) = \mathop{\mathbb{P}}_{x \sim \mathcal{D}}[h(x) \neq c(x)] = \mathop{\mathbb{E}}_{x \sim \mathcal{D}}\left[1_{h(x) \neq c(x)}\right], \tag{2.1}$$

where 1_ω is the indicator function of the event ω.[2]

The generalization error of a hypothesis is not directly accessible to the learner since both the distribution $\mathcal{D}$ and the target concept c are unknown. However, the learner can measure the *empirical error* of a hypothesis on the labeled sample S.

Definition 2.2 (Empirical error) *Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and a sample $S = (x_1, \ldots, x_m)$, the* empirical error *or* empirical risk *of h is defined by*

$$\widehat{R}_S(h) = \frac{1}{m} \sum_{i=1}^{m} 1_{h(x_i) \neq c(x_i)}. \tag{2.2}$$

Thus, the empirical error of $h \in \mathcal{H}$ is its average error over the sample S, while the generalization error is its expected error based on the distribution $\mathcal{D}$. We will see in this chapter and the following chapters a number of guarantees relating these two quantities with high probability, under some general assumptions. We can already note that for a fixed $h \in \mathcal{H}$, the expectation of the empirical error based on an i.i.d.

[1] The choice of R instead of E to denote an error avoids possible confusions with the notation for expectations and is further justified by the fact that the term *risk* is also used in machine learning and statistics to refer to an error.

[2] For this and other related definitions, the family of functions $\mathcal{H}$ and the target concept c must be measurable. The function classes we consider in this book all have this property.

sample S is equal to the generalization error:

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[\widehat{R}_S(h)] = R(h). \tag{2.3}$$

Indeed, by the linearity of the expectation and the fact that the sample is drawn i.i.d., we can write

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[\widehat{R}_S(h)] = \frac{1}{m}\sum_{i=1}^{m}\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[1_{h(x_i)\neq c(x_i)}] = \frac{1}{m}\sum_{i=1}^{m}\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[1_{h(x)\neq c(x)}],$$

for any x in sample S. Thus,

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[\widehat{R}_S(h)] = \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[1_{h(x)\neq c(x)}] = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[1_{h(x)\neq c(x)}] = R(h).$$

The following introduces the *Probably Approximately Correct* (PAC) learning framework. Let n be a number such that the computational cost of representing any element $x \in \mathcal{X}$ is at most $O(n)$ and denote by size(c) the maximal cost of the computational representation of $c \in \mathcal{C}$. For example, x may be a vector in $\mathbb{R}^n$, for which the cost of an array-based representation would be in $O(n)$. In addition, let h_S denote the hypothesis returned by algorithm $\mathcal{A}$ after receiving a labeled sample S. To keep notation simple, the dependency of h_S on $\mathcal{A}$ is not explicitly indicated.

Definition 2.3 (PAC-learning) *A concept class $\mathcal{C}$ is said to be* PAC-learnable *if there exists an algorithm $\mathcal{A}$ and a polynomial function poly$(\cdot,\cdot,\cdot,\cdot)$ such that for any $\epsilon > 0$ and $\delta > 0$, for all distributions $\mathcal{D}$ on $\mathcal{X}$ and for any target concept $c \in \mathcal{C}$, the following holds for any sample size $m \geq$ poly$(1/\epsilon, 1/\delta, n,$ size$(c))$:*

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[R(h_S) \leq \epsilon] \geq 1-\delta. \tag{2.4}$$

If $\mathcal{A}$ further runs in poly$(1/\epsilon, 1/\delta, n,$ size$(c))$, then $\mathcal{C}$ is said to be efficiently PAC-learnable. *When such an algorithm $\mathcal{A}$ exists, it is called a* PAC-learning algorithm *for $\mathcal{C}$.*

A concept class $\mathcal{C}$ is thus PAC-learnable if the hypothesis returned by the algorithm after observing a number of points polynomial in $1/\epsilon$ and $1/\delta$ is *approximately correct* (error at most ϵ) with high *probability* (at least $1-\delta$), which justifies the PAC terminology. The parameter $\delta > 0$ is used to define the *confidence* $1-\delta$ and $\epsilon > 0$ the *accuracy* $1-\epsilon$. Note that if the running time of the algorithm is polynomial in $1/\epsilon$ and $1/\delta$, then the sample size m must also be polynomial if the full sample is received by the algorithm.

Several key points of the PAC definition are worth emphasizing. First, the PAC framework is a *distribution-free model*: no particular assumption is made about the distribution $\mathcal{D}$ from which examples are drawn. Second, the training sample and the test examples used to define the error are drawn according to the same distribution $\mathcal{D}$. This is a natural and necessary assumption for generalization to

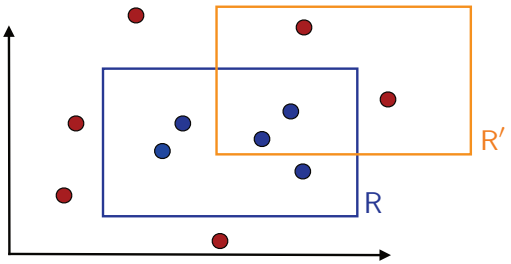

Figure 2.1
Target concept R and possible hypothesis R′. Circles represent training instances. A blue circle is a point labeled with 1, since it falls within the rectangle R. Others are red and labeled with 0.

be possible in general. It can be relaxed to include favorable *domain adaptation* problems. Finally, the PAC framework deals with the question of learnability for a concept class $\mathcal{C}$ and not a particular concept. Note that the concept class $\mathcal{C}$ is known to the algorithm, but of course the target concept $c \in \mathcal{C}$ is unknown.

In many cases, in particular when the computational representation of the concepts is not explicitly discussed or is straightforward, we may omit the polynomial dependency on n and size(c) in the PAC definition and focus only on the sample complexity.

We now illustrate PAC-learning with a specific learning problem.

Example 2.4 (Learning axis-aligned rectangles) Consider the case where the set of instances are points in the plane, $\mathcal{X} = \mathbb{R}^2$, and the concept class $\mathcal{C}$ is the set of all axis-aligned rectangles lying in $\mathbb{R}^2$. Thus, each concept c is the set of points inside a particular axis-aligned rectangle. The learning problem consists of determining with small error a target axis-aligned rectangle using the labeled training sample. We will show that the concept class of axis-aligned rectangles is PAC-learnable.

Figure 2.1 illustrates the problem. R represents a target axis-aligned rectangle and R′ a hypothesis. As can be seen from the figure, the error regions of R′ are formed by the area within the rectangle R but outside the rectangle R′ and the area within R′ but outside the rectangle R. The first area corresponds to *false negatives*, that is, points that are labeled as 0 or *negatively* by R′, which are in fact *positive* or labeled with 1. The second area corresponds to *false positives*, that is, points labeled positively by R′ which are in fact negatively labeled.

To show that the concept class is PAC-learnable, we describe a simple PAC-learning algorithm $\mathcal{A}$. Given a labeled sample S, the algorithm consists of returning the tightest axis-aligned rectangle $\mathsf{R}' = \mathsf{R}_S$ containing the points labeled with 1. Figure 2.2 illustrates the hypothesis returned by the algorithm. By definition, R_S does not produce any false positives, since its points must be included in the target concept R. Thus, the error region of R_S is included in R.

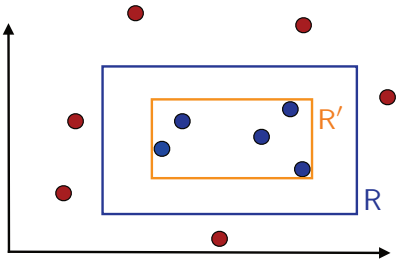

Figure 2.2
Illustration of the hypothesis $\mathsf{R}' = \mathsf{R}_S$ returned by the algorithm.

Let $\mathsf{R} \in \mathcal{C}$ be a target concept. Fix $\epsilon > 0$. Let $\mathbb{P}[\mathsf{R}]$ denote the probability mass of the region defined by R, that is the probability that a point randomly drawn according to $\mathcal{D}$ falls within R. Since errors made by our algorithm can be due only to points falling inside R, we can assume that $\mathbb{P}[\mathsf{R}] > \epsilon$; otherwise, the error of R_S is less than or equal to ϵ regardless of the training sample S received.

Now, since $\mathbb{P}[\mathsf{R}] > \epsilon$, we can define four rectangular regions r_1, r_2, r_3, and r_4 along the sides of R, each with probability at least $\epsilon/4$. These regions can be constructed by starting with the full rectangle R and then decreasing the size by moving one side as much as possible while keeping a distribution mass of at least $\epsilon/4$. Figure 2.3 illustrates the definition of these regions.

Let l, r, b, and t be the four real values defining R: $\mathsf{R} = [l, r] \times [b, t]$. Then, for example, the left rectangle r_4 is defined by $r_4 = [l, s_4] \times [b, t]$, with $s_4 = \inf\{s\colon \mathbb{P}[[l, s] \times [b, t]] \geq \epsilon/4\}$. It is not hard to see that the probability of the region $\overline{r}_4 = [l, s_4[\times[b, t]$ obtained from r_4 by excluding the rightmost side is at most $\epsilon/4$. r_1, r_2, r_3 and $\overline{r}_1$, $\overline{r}_2$, $\overline{r}_3$ are defined in a similar way.

Observe that if R_S meets all of these four regions r_i, $i \in [4]$, then, because it is a rectangle, it will have one side in each of these regions (geometric argument). Its error area, which is the part of R that it does not cover, is thus included in the union of the regions $\overline{r}_i$, $i \in [4]$, and cannot have probability mass more than ϵ. By contraposition, if $R(\mathsf{R}_S) > \epsilon$, then R_S must miss at least one of the regions r_i, $i \in [4]$. As a result, we can write

$$\begin{aligned}
\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[R(\mathsf{R}_S) > \epsilon] &\leq \mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[\cup_{i=1}^4 \{\mathsf{R}_S \cap r_i = \emptyset\}] && (2.5)\\
&\leq \sum_{i=1}^4 \mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[\{\mathsf{R}_S \cap r_i = \emptyset\}] && \text{(by the union bound)}\\
&\leq 4(1-\epsilon/4)^m && \text{(since } \mathbb{P}[r_i] \geq \epsilon/4)\\
&\leq 4\exp(-m\epsilon/4),
\end{aligned}$$

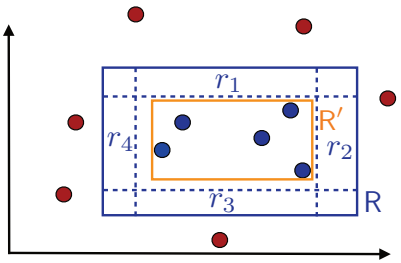

Figure 2.3
Illustration of the regions $r_1, \ldots, r_4$.

where for the last step we used the general inequality $1 - x \leq e^{-x}$ valid for all $x \in \mathbb{R}$. For any $\delta > 0$, to ensure that $\mathbb{P}_{S \sim \mathcal{D}^m}[R(\mathsf{R_S}) > \epsilon] \leq \delta$, we can impose

$$4 \exp(-\epsilon m/4) \leq \delta \Leftrightarrow m \geq \frac{4}{\epsilon} \log \frac{4}{\delta}. \tag{2.6}$$

Thus, for any $\epsilon > 0$ and $\delta > 0$, if the sample size m is greater than $\frac{4}{\epsilon} \log \frac{4}{\delta}$, then $\mathbb{P}_{S \sim \mathcal{D}^m}[R(\mathsf{R_S}) > \epsilon] \leq \delta$. Furthermore, the computational cost of the representation of points in $\mathbb{R}^2$ and axis-aligned rectangles, which can be defined by their four corners, is constant. This proves that the concept class of axis-aligned rectangles is PAC-learnable and that the sample complexity of PAC-learning axis-aligned rectangles is in $O(\frac{1}{\epsilon} \log \frac{1}{\delta})$.

An equivalent way to present sample complexity results like (2.6), which we will often see throughout this book, is to give a *generalization bound*. A generalization bound states that with probability at least $1 - \delta$, $R(\mathsf{R_S})$ is upper bounded by some quantity that depends on the sample size m and δ. To obtain this, it suffices to set δ to be equal to the upper bound derived in (2.5), that is $\delta = 4 \exp(-m\epsilon/4)$ and solve for ϵ. This yields that with probability at least $1 - \delta$, the error of the algorithm is bounded as follows:

$$R(\mathsf{R_S}) \leq \frac{4}{m} \log \frac{4}{\delta}. \tag{2.7}$$

Other PAC-learning algorithms could be considered for this example. One alternative is to return the largest axis-aligned rectangle not containing the negative points, for example. The proof of PAC-learning just presented for the tightest axis-aligned rectangle can be easily adapted to the analysis of other such algorithms.

Note that the hypothesis set $\mathcal{H}$ we considered in this example coincided with the concept class $\mathcal{C}$ and that its cardinality was infinite. Nevertheless, the problem admitted a simple proof of PAC-learning. We may then ask if a similar proof can readily apply to other similar concept classes. This is not as straightforward because the specific geometric argument used in the proof is key. It is non-trivial to extend the proof to other concept classes such as that of non-concentric circles

(see exercise 2.4). Thus, we need a more general proof technique and more general results. The next two sections provide us with such tools in the case of a finite hypothesis set.

2.2 Guarantees for finite hypothesis sets — consistent case

In the example of axis-aligned rectangles that we examined, the hypothesis h_S returned by the algorithm was always *consistent*, that is, it admitted no error on the training sample S. In this section, we present a general sample complexity bound, or equivalently, a generalization bound, for consistent hypotheses, in the case where the cardinality $|\mathcal{H}|$ of the hypothesis set is finite. Since we consider consistent hypotheses, we will assume that the target concept c is in $\mathcal{H}$.

Theorem 2.5 (Learning bound — finite $\mathcal{H}$, consistent case) *Let $\mathcal{H}$ be a finite set of functions mapping from $\mathcal{X}$ to $\mathcal{Y}$. Let $\mathcal{A}$ be an algorithm that for any target concept $c \in \mathcal{H}$ and i.i.d. sample S returns a consistent hypothesis h_S: $\widehat{R}_S(h_S) = 0$. Then, for any $\epsilon, \delta > 0$, the inequality $\mathbb{P}_{S\sim\mathcal{D}^m}[R(h_S) \leq \epsilon] \geq 1 - \delta$ holds if*

$$m \geq \frac{1}{\epsilon}\Big(\log|\mathcal{H}| + \log\frac{1}{\delta}\Big). \tag{2.8}$$

This sample complexity result admits the following equivalent statement as a generalization bound: for any $\epsilon, \delta > 0$, with probability at least $1 - \delta$,

$$R(h_S) \leq \frac{1}{m}\Big(\log|\mathcal{H}| + \log\frac{1}{\delta}\Big). \tag{2.9}$$

Proof: Fix $\epsilon > 0$. We do not know which consistent hypothesis $h_S \in \mathcal{H}$ is selected by the algorithm $\mathcal{A}$. This hypothesis further depends on the training sample S. Therefore, we need to give a *uniform convergence bound*, that is, a bound that holds for the set of all consistent hypotheses, which a fortiori includes h_S. Thus, we will bound the probability that some $h \in \mathcal{H}$ would be consistent and have error more than ϵ. For any $\epsilon > 0$, define $\mathcal{H}_\epsilon$ by $\mathcal{H}_\epsilon = \{h \in \mathcal{H} \colon R(h) > \epsilon\}$. The probability that a hypothesis h in $\mathcal{H}_\epsilon$ is consistent on a training sample S drawn i.i.d., that is, that it would have no error on any point in S, can be bounded as follows:

$$\mathbb{P}[\widehat{R}_S(h) = 0] \leq (1-\epsilon)^m.$$

Thus, by the union bound, the following holds:

$$\begin{aligned}
\mathbb{P}\Big[\exists h \in \mathcal{H}_\epsilon \colon \widehat{R}_S(h) = 0\Big] &= \mathbb{P}\Big[\widehat{R}_S(h_1) = 0 \vee \cdots \vee \widehat{R}_S(h_{|\mathcal{H}_\epsilon|}) = 0\Big] \\
&\leq \sum_{h\in\mathcal{H}_\epsilon} \mathbb{P}\Big[\widehat{R}_S(h) = 0\Big] \qquad \text{(union bound)} \\
&\leq \sum_{h\in\mathcal{H}_\epsilon} (1-\epsilon)^m \leq |\mathcal{H}|(1-\epsilon)^m \leq |\mathcal{H}|e^{-m\epsilon}.
\end{aligned}$$

Setting the right-hand side to be equal to δ and solving for ϵ concludes the proof. □

The theorem shows that when the hypothesis set $\mathcal{H}$ is finite, a consistent algorithm $\mathcal{A}$ is a PAC-learning algorithm, since the sample complexity given by (2.8) is dominated by a polynomial in $1/\epsilon$ and $1/\delta$. As shown by (2.9), the generalization error of consistent hypotheses is upper bounded by a term that decreases as a function of the sample size m. This is a general fact: as expected, learning algorithms benefit from larger labeled training samples. The decrease rate of $O(1/m)$ guaranteed by this theorem, however, is particularly favorable.

The price to pay for coming up with a consistent algorithm is the use of a larger hypothesis set $\mathcal{H}$ containing target concepts. Of course, the upper bound (2.9) increases with $|\mathcal{H}|$. However, that dependency is only logarithmic. Note that the term $\log|\mathcal{H}|$, or the related term $\log_2|\mathcal{H}|$ from which it differs by a constant factor, can be interpreted as the number of bits needed to represent $\mathcal{H}$. Thus, the generalization guarantee of the theorem is controlled by the ratio of this number of bits, $\log_2|\mathcal{H}|$, and the sample size m.

We now use theorem 2.5 to analyze PAC-learning with various concept classes.

Example 2.6 (Conjunction of Boolean literals) Consider learning the concept class $\mathcal{C}_n$ of conjunctions of at most n Boolean literals $x_1, \ldots, x_n$. A Boolean literal is either a variable x_i, $i \in [n]$, or its negation $\overline{x}_i$. For $n = 4$, an example is the conjunction: $x_1 \wedge \overline{x}_2 \wedge x_4$, where $\overline{x}_2$ denotes the negation of the Boolean literal x_2. $(1, 0, 0, 1)$ is a positive example for this concept while $(1, 0, 0, 0)$ is a negative example.

Observe that for $n = 4$, a positive example $(1, 0, 1, 0)$ implies that the target concept cannot contain the literals $\overline{x}_1$ and $\overline{x}_3$ and that it cannot contain the literals x_2 and x_4. In contrast, a negative example is not as informative since it is not known which of its n bits are incorrect. A simple algorithm for finding a consistent hypothesis is thus based on positive examples and consists of the following: for each positive example $(b_1, \ldots, b_n)$ and $i \in [n]$, if $b_i = 1$ then $\overline{x}_i$ is ruled out as a possible literal in the concept class and if $b_i = 0$ then x_i is ruled out. The conjunction of all the literals not ruled out is thus a hypothesis consistent with the target. Figure 2.4 shows an example training sample as well as a consistent hypothesis for the case $n = 6$.

We have $|\mathcal{H}| = |\mathcal{C}_n| = 3^n$, since each literal can be included positively, with negation, or not included. Plugging this into the sample complexity bound for consistent hypotheses yields the following sample complexity bound for any $\epsilon > 0$ and $\delta > 0$:

$$m \geq \frac{1}{\epsilon}\left((\log 3)n + \log\frac{1}{\delta}\right). \tag{2.10}$$

Thus, the class of conjunctions of at most n Boolean literals is PAC-learnable. Note that the computational complexity is also polynomial, since the training cost per example is in $O(n)$. For $\delta = 0.02$, $\epsilon = 0.1$, and $n = 10$, the bound becomes

0	1	1	0	1	1	+
0	1	1	1	1	1	+
0	0	1	1	0	1	-
0	1	1	1	1	1	+
1	0	0	1	1	0	-
0	1	0	0	1	1	+
0	1	?	?	1	1	

Figure 2.4
Each of the first six rows of the table represents a training example with its label, + or −, indicated in the last column. The last row contains 0 (respectively 1) in column $i \in [6]$ if the ith entry is 0 (respectively 1) for all the positive examples. It contains "?" if both 0 and 1 appear as an ith entry for some positive example. Thus, for this training sample, the hypothesis returned by the consistent algorithm described in the text is $\overline{x}_1 \wedge x_2 \wedge x_5 \wedge x_6$.

$m \geq 149$. Thus, for a labeled sample of at least 149 examples, the bound guarantees 90% accuracy with a confidence of at least 98%.

Example 2.7 (Universal concept class) Consider the set $\mathcal{X} = \{0,1\}^n$ of all Boolean vectors with n components, and let $\mathcal{U}_n$ be the concept class formed by all subsets of $\mathcal{X}$. Is this concept class PAC-learnable? To guarantee a consistent hypothesis the hypothesis class must include the concept class, thus $|\mathcal{H}| \geq |\mathcal{U}_n| = 2^{(2^n)}$. Theorem 2.5 gives the following sample complexity bound:

$$m \geq \frac{1}{\epsilon}\left((\log 2)2^n + \log\frac{1}{\delta}\right). \tag{2.11}$$

Here, the number of training samples required is exponential in n, which is the cost of the representation of a point in $\mathcal{X}$. Thus, PAC-learning is not guaranteed by the theorem. In fact, it is not hard to show that this universal concept class is not PAC-learnable.

Example 2.8 (k-term DNF formulae) A disjunctive normal form (DNF) formula is a formula written as the disjunction of several terms, each term being a conjunction of Boolean literals. A k-term DNF is a DNF formula defined by the disjunction of k terms, each term being a conjunction of at most n Boolean literals. Thus, for $k = 2$ and $n = 3$, an example of a k-term DNF is $(x_1 \wedge \overline{x}_2 \wedge x_3) \vee (\overline{x}_1 \wedge x_3)$.

Is the class $\mathcal{C}$ of k-term DNF formulae PAC-learnable? The cardinality of the class is 3^{nk}, since each term is a conjunction of at most n variables and there are 3^n such conjunctions, as seen previously. The hypothesis set $\mathcal{H}$ must contain $\mathcal{C}$ for

consistency to be possible, thus $|\mathcal{H}| \geq 3^{nk}$. Theorem 2.5 gives the following sample complexity bound:

$$m \geq \frac{1}{\epsilon}\Big((\log 3)nk + \log\frac{1}{\delta}\Big), \tag{2.12}$$

which is polynomial. However, it can be shown by a reduction from the graph 3-coloring problem that the problem of learning k-term DNF, even for $k = 3$, is not efficiently PAC-learnable, unless RP, the complexity class of problems that admit a randomized polynomial-time decision solution, coincides with NP(RP = NP), which is commonly conjectured not to be the case. Thus, while the sample size needed for learning k-term DNF formulae is only polynomial, efficient PAC-learning of this class is not possible if RP $\neq$ NP.

Example 2.9 (k-CNF formulae) A conjunctive normal form (CNF) formula is a conjunction of disjunctions. A k-CNF formula is an expression of the form $T_1 \wedge \ldots \wedge T_j$ with arbitrary length $j \in \mathbb{N}$ and with each term T_i being a disjunction of at most k Boolean attributes.

The problem of learning k-CNF formulae can be reduced to that of learning conjunctions of Boolean literals, which, as seen previously, is a PAC-learnable concept class. This can be done at the cost of introducing $(2n)^k$ new variables $Y_{u_1,\ldots,u_k}$ using the following bijection:

$$(u_1, \ldots, u_k) \to Y_{u_1,\ldots,u_k}, \tag{2.13}$$

where $u_1, \ldots, u_k$ are Boolean literals over the original variables $x_1, \ldots, x_n$. The value of $Y_{u_1,\ldots,u_k}$ is determined by $Y_{u_1,\ldots,u_k} = u_1 \vee \cdots \vee u_k$. Using this mapping, the original training sample can be transformed into one defined in terms of the new variables and any k-CNF formula over the original variables can be written as a conjunction over the variables $Y_{u_1,\ldots,u_k}$. This reduction to PAC-learning of conjunctions of Boolean literals can affect the original distribution of examples, but this is not an issue since in the PAC framework no assumption is made about the distribution. Thus, using this transformation, the PAC-learnability of conjunctions of Boolean literals implies that of k-CNF formulae.

This is a surprising result, however, since any k-term DNF formula can be written as a k-CNF formula. Indeed, using associativity, a k-term DNF $T_1 \vee \cdots \vee T_k$ with $T_i = u_{i,1} \wedge \cdots \wedge u_{i,n_i}$ for $i \in [k]$ can be rewritten as a k-CNF formula via

$$\bigvee_{i=1}^{k} u_{i,1} \wedge \cdots \wedge u_{i,n_i} = \bigwedge_{j_1 \in [n_1], \ldots, j_k \in [n_k]} u_{1,j_1} \vee \cdots \vee u_{k,j_k},$$

To illustrate this rewriting in a specific case, observe, for example, that

$$(u_1 \wedge u_2 \wedge u_3) \vee (v_1 \wedge v_2 \wedge v_3) = \bigwedge_{i,j=1}^{3} (u_i \vee v_j).$$

But, as we previously saw, k-term DNF formulae are not efficiently PAC-learnable if RP $\neq$ NP! What can explain this apparent inconsistency? The issue is that converting into a k-term DNF a k-CNF formula we have learned (which is equivalent to a k-term DNF) is in general intractable if RP $\neq$ NP.

This example reveals some key aspects of PAC-learning, which include the cost of the representation of a concept and the choice of the hypothesis set. For a fixed concept class, learning can be intractable or not depending on the choice of the representation.

2.3 Guarantees for finite hypothesis sets — inconsistent case

In the most general case, there may be no hypothesis in $\mathcal{H}$ consistent with the labeled training sample. This, in fact, is the typical case in practice, where the learning problems may be somewhat difficult or the concept classes more complex than the hypothesis set used by the learning algorithm. However, inconsistent hypotheses with a small number of errors on the training sample can be useful and, as we shall see, can benefit from favorable guarantees under some assumptions. This section presents learning guarantees precisely for this inconsistent case and finite hypothesis sets.

To derive learning guarantees in this more general setting, we will use Hoeffding's inequality (theorem D.2) or the following corollary, which relates the generalization error and empirical error of a single hypothesis.

Corollary 2.10 *Fix $\epsilon > 0$. Then, for any hypothesis $h\colon X \to \{0,1\}$, the following inequalities hold:*

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left[\widehat{R}_S(h) - R(h) \geq \epsilon\right] \leq \exp(-2m\epsilon^2) \tag{2.14}$$

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left[\widehat{R}_S(h) - R(h) \leq -\epsilon\right] \leq \exp(-2m\epsilon^2). \tag{2.15}$$

By the union bound, this implies the following two-sided inequality:

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}\left[\left|\widehat{R}_S(h) - R(h)\right| \geq \epsilon\right] \leq 2\exp(-2m\epsilon^2). \tag{2.16}$$

Proof: The result follows immediately from theorem D.2. □

Setting the right-hand side of (2.16) to be equal to δ and solving for ϵ yields immediately the following bound for a single hypothesis.

Corollary 2.11 (Generalization bound — single hypothesis) *Fix a hypothesis $h\colon \mathcal{X} \to \{0,1\}$. Then, for any $\delta > 0$, the following inequality holds with probability at least $1-\delta$:*

$$R(h) \leq \widehat{R}_S(h) + \sqrt{\frac{\log\frac{2}{\delta}}{2m}}. \tag{2.17}$$

The following example illustrates this corollary in a simple case.

Example 2.12 (Tossing a coin) Imagine tossing a biased coin that lands heads with probability p, and let our hypothesis be the one that always guesses tails. Then the true error rate is $R(h) = p$ and the empirical error rate $\widehat{R}_S(h) = \widehat{p}$, where $\widehat{p}$ is the empirical probability of heads based on the training sample drawn i.i.d. Thus, corollary 2.11 guarantees with probability at least $1 - \delta$ that

$$|p - \widehat{p}| \leq \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{2.18}$$

Therefore, if we choose $\delta = 0.02$ and use a sample of size 500, with probability at least 98%, the following approximation quality is guaranteed for $\widehat{p}$:

$$|p - \widehat{p}| \leq \sqrt{\frac{\log(10)}{1000}} \approx 0.048. \tag{2.19}$$

Can we readily apply corollary 2.11 to bound the generalization error of the hypothesis h_S returned by a learning algorithm when training on a sample S? No, since h_S is not a fixed hypothesis, but a random variable depending on the training sample S drawn. Note also that unlike the case of a fixed hypothesis for which the expectation of the empirical error is the generalization error (equation (2.3)), the generalization error $R(h_S)$ is a random variable and in general distinct from the expectation $\mathbb{E}[\widehat{R}_S(h_S)]$, which is a constant.

Thus, as in the proof for the consistent case, we need to derive a uniform convergence bound, that is a bound that holds with high probability for all hypotheses $h \in \mathcal{H}$.

Theorem 2.13 (Learning bound — finite $\mathcal{H}$, inconsistent case) *Let $\mathcal{H}$ be a finite hypothesis set. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following inequality holds:*

$$\forall h \in \mathcal{H}, \quad R(h) \leq \widehat{R}_S(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{2}{\delta}}{2m}}. \tag{2.20}$$

Proof: Let $h_1, \ldots, h_{|\mathcal{H}|}$ be the elements of $\mathcal{H}$. Using the union bound and applying corollary 2.11 to each hypothesis yield:

$$\begin{aligned}
&\mathbb{P}\left[\exists h \in \mathcal{H} \middle| \widehat{R}_S(h) - R(h)\middle| > \epsilon\right] \\
&\qquad = \mathbb{P}\left[\left(\middle|\widehat{R}_S(h_1) - R(h_1)\middle| > \epsilon\right) \vee \ldots \vee \left(\middle|\widehat{R}_S(h_{|\mathcal{H}|}) - R(h_{|\mathcal{H}|})\middle| > \epsilon\right)\right] \\
&\qquad \leq \sum_{h \in \mathcal{H}} \mathbb{P}\left[\middle|\widehat{R}_S(h) - R(h)\middle| > \epsilon\right] \\
&\qquad \leq 2|\mathcal{H}| \exp(-2m\epsilon^2).
\end{aligned}$$

Setting the right-hand side to be equal to δ completes the proof. □

Thus, for a finite hypothesis set $\mathcal{H}$,

$$R(h) \leq \widehat{R}_S(h) + O\left(\sqrt{\frac{\log_2 |\mathcal{H}|}{m}}\right).$$

As already pointed out, $\log_2 |\mathcal{H}|$ can be interpreted as the number of bits needed to represent $\mathcal{H}$. Several other remarks similar to those made on the generalization bound in the consistent case can be made here: a larger sample size m guarantees better generalization, and the bound increases with $|\mathcal{H}|$, but only logarithmically. But, here, the bound is a less favorable function of $\frac{\log_2 |\mathcal{H}|}{m}$; it varies as the square root of this term. This is not a minor price to pay: for a fixed $|\mathcal{H}|$, to attain the same guarantee as in the consistent case, a quadratically larger labeled sample is needed.

Note that the bound suggests seeking a trade-off between reducing the empirical error versus controlling the size of the hypothesis set: a larger hypothesis set is penalized by the second term but could help reduce the empirical error, that is the first term. But, for a similar empirical error, it suggests using a smaller hypothesis set. This can be viewed as an instance of the so-called *Occam's Razor principle* named after the theologian William of Occam: *Plurality should not be posited without necessity*, also rephrased as, *the simplest explanation is best*. In this context, it could be expressed as follows: All other things being equal, a simpler (smaller) hypothesis set is better.

2.4 Generalities

In this section we will discuss some general aspects of the learning scenario, which, for simplicity, we left out of the discussion of the earlier sections.

2.4.1 Deterministic versus stochastic scenarios

In the most general scenario of supervised learning, the distribution $\mathcal{D}$ is defined over $\mathcal{X} \times \mathcal{Y}$, and the training data is a labeled sample S drawn i.i.d. according to $\mathcal{D}$:

$$S = ((x_1, y_1), \ldots, (x_m, y_m)).$$

The learning problem is to find a hypothesis $h \in \mathcal{H}$ with small generalization error

$$R(h) = \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[1_{h(x)\neq y}].$$

This more general scenario is referred to as the *stochastic scenario*. Within this setting, the output label is a probabilistic function of the input. The stochastic scenario captures many real-world problems where the label of an input point is not unique. For example, if we seek to predict gender based on input pairs formed by the height and weight of a person, then the label will typically not be unique.

For most pairs, both male and female are possible genders. For each fixed pair, there would be a probability distribution of the label being male.

The natural extension of the PAC-learning framework to this setting is known as the *agnostic PAC-learning*.

Definition 2.14 (Agnostic PAC-learning) *Let $\mathcal{H}$ be a hypothesis set. $\mathcal{A}$ is an* agnostic PAC-learning *algorithm if there exists a polynomial function $poly(\cdot,\cdot,\cdot,\cdot)$ such that for any $\epsilon > 0$ and $\delta > 0$, for all distributions $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, the following holds for any sample size $m \geq poly(1/\epsilon, 1/\delta, n, size(c))$:*

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[R(h_S) - \min_{h\in\mathcal{H}} R(h) \leq \epsilon] \geq 1 - \delta. \tag{2.21}$$

If $\mathcal{A}$ further runs in $poly(1/\epsilon, 1/\delta, n)$, then it is said to be an efficient agnostic PAC-learning algorithm.

When the label of a point can be uniquely determined by some measurable function $f\colon \mathcal{X} \to \mathcal{Y}$ (with probability one), then the scenario is said to be *deterministic*. In that case, it suffices to consider a distribution $\mathcal{D}$ over the input space. The training sample is obtained by drawing $(x_1, \ldots, x_m)$ according to $\mathcal{D}$ and the labels are obtained via f: $y_i = f(x_i)$ for all $i \in [m]$. Many learning problems can be formulated within this deterministic scenario.

In the previous sections, as well as in most of the material presented in this book, we have restricted our presentation to the deterministic scenario in the interest of simplicity. However, for all of this material, the extension to the stochastic scenario should be straightforward for the reader.

2.4.2 Bayes error and noise

In the deterministic case, by definition, there exists a target function f with no generalization error: $R(h) = 0$. In the stochastic case, there is a minimal non-zero error for any hypothesis.

Definition 2.15 (Bayes error) *Given a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, the* Bayes error R^* *is defined as the infimum of the errors achieved by measurable functions $h\colon \mathcal{X} \to \mathcal{Y}$:*

$$R^\star = \inf_{\substack{h \\ h\ measurable}} R(h). \tag{2.22}$$

A hypothesis h with $R(h) = R^*$ is called a Bayes hypothesis *or* Bayes classifier.

By definition, in the deterministic case, we have $R^* = 0$, but, in the stochastic case, $R^* \neq 0$. Clearly, the Bayes classifier h_{Bayes} can be defined in terms of the conditional probabilities as:

$$\forall x \in \mathcal{X}, \quad h_{\text{Bayes}}(x) = \operatorname*{argmax}_{y\in\{0,1\}} \mathbb{P}[y|x]. \tag{2.23}$$

The average error made by h_{Bayes} on $x \in \mathcal{X}$ is thus $\min\{\mathbb{P}[0|x], \mathbb{P}[1|x]\}$, and this is the minimum possible error. This leads to the following definition of *noise*.

Definition 2.16 (Noise) *Given a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, the* noise *at point $x \in \mathcal{X}$ is defined by*

$$\mathit{noise}(x) = \min\{\mathbb{P}[1|x], \mathbb{P}[0|x]\}. \tag{2.24}$$

The average noise *or the* noise *associated to $\mathcal{D}$ is $\mathbb{E}[\mathit{noise}(x)]$.*

Thus, the average noise is precisely the Bayes error: $\text{noise} = \mathbb{E}[\text{noise}(x)] = R^*$. The noise is a characteristic of the learning task indicative of its level of difficulty. A point $x \in \mathcal{X}$, for which $\text{noise}(x)$ is close to $1/2$, is sometimes referred to as *noisy* and is of course a challenge for accurate prediction.

2.5 Chapter notes

The PAC learning framework was introduced by Valiant [1984]. The book of Kearns and Vazirani [1994] is an excellent reference dealing with most aspects of PAC-learning and several other foundational questions in machine learning. Our example of learning axis-aligned rectangles, also discussed in that reference, is originally due to Blumer et al. [1989].

The PAC learning framework is a computational framework since it takes into account the cost of the computational representations and the time complexity of the learning algorithm. If we omit the computational aspects, it is similar to the learning framework considered earlier by Vapnik and Chervonenkis [see Vapnik, 2000]. The definition of noise presented in this chapter can be generalized to arbitrary loss functions (see exercise 2.14).

Occam's razor principle is invoked in a variety of contexts, such as in linguistics to justify the superiority of a set of rules or syntax. The Kolmogorov complexity can be viewed as the corresponding framework in information theory. In the context of the learning guarantees presented in this chapter, the principle suggests selecting the most parsimonious explanation (the hypothesis set with the smallest cardinality). We will see in the next sections other applications of this principle with different notions of simplicity or complexity.

2.6 Exercises

2.1 Two-oracle variant of the PAC model. Assume that positive and negative examples are now drawn from two separate distributions $\mathcal{D}_+$ and $\mathcal{D}_-$. For an accuracy $(1-\epsilon)$, the learning algorithm must find a hypothesis h such that:

$$\mathop{\mathbb{P}}_{x \sim \mathcal{D}_+}[h(x) = 0] \leq \epsilon \text{ and } \mathop{\mathbb{P}}_{x \sim \mathcal{D}_-}[h(x) = 1] \leq \epsilon\,. \tag{2.25}$$

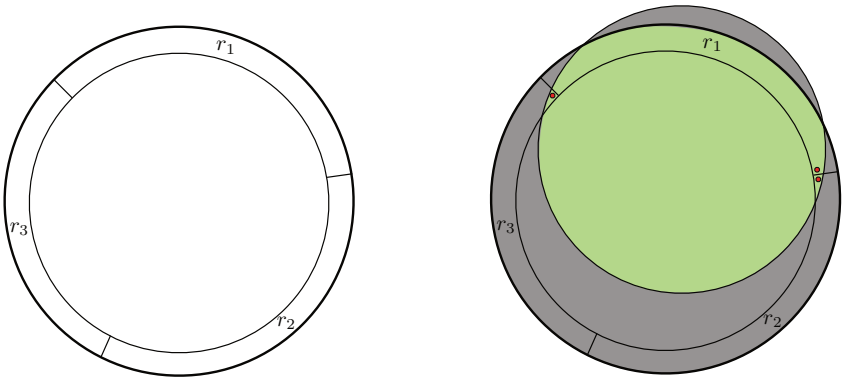

Figure 2.5
(a) Gertrude's regions r_1, r_2, r_3. (b) Hint for solution.

Thus, the hypothesis must have a small error on both distributions. Let $\mathcal{C}$ be any concept class and $\mathcal{H}$ be any hypothesis space. Let h_0 and h_1 represent the identically 0 and identically 1 functions, respectively. Prove that $\mathcal{C}$ is efficiently PAC-learnable using $\mathcal{H}$ in the standard (one-oracle) PAC model if and only if it is efficiently PAC-learnable using $\mathcal{H} \cup \{h_0, h_1\}$ in this two-oracle PAC model.

2.2 PAC learning of hyper-rectangles. An axis-aligned hyper-rectangle in $\mathbb{R}^n$ is a set of the form $[a_1, b_1] \times \ldots \times [a_n, b_n]$. Show that axis-aligned hyper-rectangles are PAC-learnable by extending the proof given in Example 2.4 for the case $n = 2$.

2.3 Concentric circles. Let $\mathcal{X} = \mathbb{R}^2$ and consider the set of concepts of the form $c = \{(x, y) \colon x^2 + y^2 \leq r^2\}$ for some real number r. Show that this class can be (ϵ, δ)-PAC-learned from training data of size $m \geq (1/\epsilon)\log(1/\delta)$.

2.4 Non-concentric circles. Let $\mathcal{X} = \mathbb{R}^2$ and consider the set of concepts of the form $c = \{x \in \mathbb{R}^2 \colon ||x - x_0|| \leq r\}$ for some point $x_0 \in \mathbb{R}^2$ and real number r. Gertrude, an aspiring machine learning researcher, attempts to show that this class of concepts may be (ϵ, δ)-PAC-learned with sample complexity $m \geq (3/\epsilon)\log(3/\delta)$, but she is having trouble with her proof. Her idea is that the learning algorithm would select the smallest circle consistent with the training data. She has drawn three regions r_1, r_2, r_3 around the edge of concept c, with each region having probability $\epsilon/3$ (see figure 2.5(a)). She wants to argue that if the generalization error is greater than or equal to ϵ, then one of these regions must have been missed by the training data, and hence this event will occur with probability at most δ. Can you tell Gertrude if her approach works? (*Hint*: You may wish to use figure 2.5(b) in your solution).

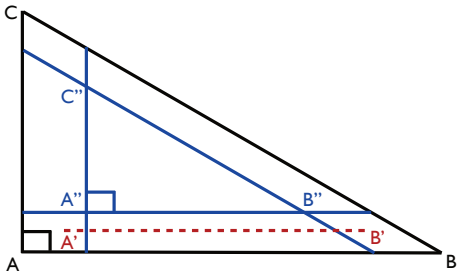

Figure 2.6
Axis-aligned right triangles.

2.5 Triangles. Let $\mathcal{X} = \mathbb{R}^2$ with orthonormal basis $(\mathbf{e}_1, \mathbf{e}_2)$, and consider the set of concepts defined by the area inside a right triangle ABC with two sides parallel to the axes, with $\overrightarrow{AB}/\|\overrightarrow{AB}\| = \mathbf{e}_1$ and $\overrightarrow{AC}/\|\overrightarrow{AC}\| = \mathbf{e}_2$, and $\|\overrightarrow{AB}\|/\|\overrightarrow{AC}\| = \alpha$ for some positive real $\alpha \in \mathbb{R}_+$. Show, using similar methods to those used in the chapter for the axis-aligned rectangles, that this class can be (ϵ, δ)-PAC-learned from training data of size $m \geq (3/\epsilon)\log(3/\delta)$. (*Hint*: You may consider using figure 2.6 in your solution).

2.6 Learning in the presence of noise — rectangles. In example 2.4, we showed that the concept class of axis-aligned rectangles is PAC-learnable. Consider now the case where the training points received by the learner are subject to the following noise: points negatively labeled are unaffected by noise but the label of a positive training point is randomly flipped to negative with probability $\eta \in (0, \frac{1}{2})$. The exact value of the noise rate η is not known to the learner but an upper bound η' is supplied to him with $\eta \leq \eta' < 1/2$. Show that the algorithm returning the tightest rectangle containing positive points can still PAC-learn axis-aligned rectangles in the presence of this noise. To do so, you can proceed using the following steps:

(a) Using the same notation as in example 2.4, assume that $\mathbb{P}[\mathsf{R}] > \epsilon$. Suppose that $R(\mathsf{R}') > \epsilon$. Give an upper bound on the probability that R' misses a region r_j, $j \in [4]$ in terms of ϵ and η'?

(b) Use that to give an upper bound on $\mathbb{P}[R(\mathsf{R}') > \epsilon]$ in terms of ϵ and η' and conclude by giving a sample complexity bound.

2.7 Learning in the presence of noise — general case. In this question, we will seek a result that is more general than in the previous question. We consider a finite hypothesis set $\mathcal{H}$, assume that the target concept is in $\mathcal{H}$, and adopt the following noise model: the label of a training point received by the learner is

randomly changed with probability $\eta \in (0, \frac{1}{2})$. The exact value of the noise rate η is not known to the learner but an upper bound η' is supplied to him with $\eta \leq \eta' < 1/2$.

(a) For any $h \in \mathcal{H}$, let $d(h)$ denote the probability that the label of a training point received by the learner disagrees with the one given by h. Let h^* be the target hypothesis, show that $d(h^*) = \eta$.

(b) More generally, show that for any $h \in \mathcal{H}$, $d(h) = \eta + (1 - 2\eta)\, R(h)$, where $R(h)$ denotes the generalization error of h.

(c) Fix $\epsilon > 0$ for this and all the following questions. Use the previous questions to show that if $R(h) > \epsilon$, then $d(h) - d(h^*) \geq \epsilon'$, where $\epsilon' = \epsilon(1 - 2\eta')$.

(d) For any hypothesis $h \in \mathcal{H}$ and sample S of size m, let $\widehat{d}(h)$ denote the fraction of the points in S whose labels disagree with those given by h. We will consider the algorithm L which, after receiving S, returns the hypothesis h_S with the smallest number of disagreements (thus $\widehat{d}(h_S)$ is minimal). To show PAC-learning for L, we will show that for any h, if $R(h) > \epsilon$, then with high probability $\widehat{d}(h) \geq \widehat{d}(h^*)$. First, show that for any $\delta > 0$, with probability at least $1 - \delta/2$, for $m \geq \frac{2}{\epsilon'^2} \log \frac{2}{\delta}$, the following holds:

$$\widehat{d}(h^*) - d(h^*) \leq \epsilon'/2$$

(e) Second, show that for any $\delta > 0$, with probability at least $1 - \delta/2$, for $m \geq \frac{2}{\epsilon'^2}(\log |\mathcal{H}| + \log \frac{2}{\delta})$, the following holds for all $h \in \mathcal{H}$:

$$d(h) - \widehat{d}(h) \leq \epsilon'/2$$

(f) Finally, show that for any $\delta > 0$, with probability at least $1 - \delta$, for $m \geq \frac{2}{\epsilon^2(1-2\eta')^2}(\log |\mathcal{H}| + \log \frac{2}{\delta})$, the following holds for all $h \in \mathcal{H}$ with $R(h) > \epsilon$:

$$\widehat{d}(h) - \widehat{d}(h^*) \geq 0.$$

(*Hint*: use $\widehat{d}(h) - \widehat{d}(h^*) = [\widehat{d}(h) - d(h)] + [d(h) - d(h^*)] + [d(h^*) - \widehat{d}(h^*)]$ and use previous questions to lower bound each of these three terms).

2.8 Learning intervals. Give a PAC-learning algorithm for the concept class $\mathcal{C}$ formed by closed intervals $[a, b]$ with $a, b \in \mathbb{R}$.

2.9 Learning union of intervals. Give a PAC-learning algorithm for the concept class $\mathcal{C}_2$ formed by unions of two closed intervals, that is $[a, b] \cup [c, d]$, with $a, b, c, d \in \mathbb{R}$. Extend your result to derive a PAC-learning algorithm for the concept class $\mathcal{C}_p$ formed by unions of $p \geq 1$ closed intervals, thus $[a_1, b_1] \cup \cdots \cup [a_p, b_p]$, with $a_k, b_k \in \mathbb{R}$ for $k \in [p]$. What are the time and sample complexities of your algorithm as a function of p?

2.10 Consistent hypotheses. In this chapter, we showed that for a finite hypothesis set $\mathcal{H}$, a consistent learning algorithm $\mathcal{A}$ is a PAC-learning algorithm. Here, we consider a converse question. Let $\mathcal{Z}$ be a finite set of m labeled points. Suppose that you are given a PAC-learning algorithm $\mathcal{A}$. Show that you can use $\mathcal{A}$ and a finite training sample S to find in polynomial time a hypothesis $h \in \mathcal{H}$ that is consistent with $\mathcal{Z}$, with high probability. (*Hint*: you can select an appropriate distribution $\mathcal{D}$ over $\mathcal{Z}$ and give a condition on $R(h)$ for h to be consistent.)

2.11 Senate laws. For important questions, President Mouth relies on expert advice. He selects an appropriate advisor from a collection of $\mathcal{H} = 2{,}800$ experts.

(a) Assume that laws are proposed in a random fashion independently and identically according to some distribution $\mathcal{D}$ determined by an unknown group of senators. Assume that President Mouth can find and select an expert senator out of $\mathcal{H}$ who has consistently voted with the majority for the last $m = 200$ laws. Give a bound on the probability that such a senator incorrectly predicts the global vote for a future law. What is the value of the bound with 95% confidence?

(b) Assume now that President Mouth can find and select an expert senator out of $\mathcal{H}$ who has consistently voted with the majority for all but $m' = 20$ of the last $m = 200$ laws. What is the value of the new bound?

2.12 Bayesian bound. Let $\mathcal{H}$ be a countable hypothesis set of functions mapping $\mathcal{X}$ to $\{0, 1\}$ and let p be a probability measure over $\mathcal{H}$. This probability measure represents the *prior probability* over the hypothesis class, i.e. the probability that a particular hypothesis is selected by the learning algorithm. Use Hoeffding's inequality to show that for any $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds:

$$\forall h \in \mathcal{H}, R(h) \leq \widehat{R}_S(h) + \sqrt{\frac{\log \frac{1}{p(h)} + \log \frac{1}{\delta}}{2m}}. \tag{2.26}$$

Compare this result with the bound given in the inconsistent case for finite hypothesis sets (*Hint*: you could use $\delta' = p(h)\delta$ as confidence parameter in Hoeffding's inequality).

2.13 Learning with an unknown parameter. In example 2.9, we showed that the concept class of k-CNF is PAC-learnable. Note, however, that the learning algorithm is given k as input. Is PAC-learning possible even when k is not provided? More generally, consider a family of concept classes $\{\mathcal{C}_s\}_s$ where $\mathcal{C}_s$ is the set of concepts in $\mathcal{C}$ with size at most s. Suppose we have a PAC-learning algorithm $\mathcal{A}$ that can be used for learning any concept class $\mathcal{C}_s$ when s is given.

Can we convert $\mathcal{A}$ into a PAC-learning algorithm $\mathcal{B}$ that does not require the knowledge of s? This is the main objective of this problem.

To do this, we first introduce a method for testing a hypothesis h, with high probability. Fix $\epsilon > 0$, $\delta > 0$, and $i \geq 1$ and define the sample size n by $n = \frac{32}{\epsilon}[i \log 2 + \log \frac{2}{\delta}]$. Suppose we draw an i.i.d. sample S of size n according to some unknown distribution $\mathcal{D}$. We will say that a hypothesis h is *accepted* if it makes at most $3/4\epsilon$ errors on S and that it is *rejected* otherwise. Thus, h is accepted iff $\widehat{R}(h) \leq 3/4\epsilon$.

(a) Assume that $R(h) \geq \epsilon$. Use the (multiplicative) Chernoff bound to show that in that case $\mathbb{P}_{S \sim \mathcal{D}^n}[h \text{ is accepted}] \leq \frac{\delta}{2^{i+1}}$.

(b) Assume that $R(h) \leq \epsilon/2$. Use the (multiplicative) Chernoff bounds to show that in that case $\mathbb{P}_{S \sim \mathcal{D}^n}[h \text{ is rejected}] \leq \frac{\delta}{2^{i+1}}$.

(c) Algorithm $\mathcal{B}$ is defined as follows: we start with $i = 1$ and, at each round $i \geq 1$, we guess the parameter size s to be $\widetilde{s} = \lfloor 2^{(i-1)/\log \frac{2}{\delta}} \rfloor$. We draw a sample S of size n (which depends on i) to test the hypothesis h_i returned by $\mathcal{A}$ when it is trained with a sample of size $S_{\mathcal{A}}(\epsilon/2, 1/2, \widetilde{s})$, that is the sample complexity of $\mathcal{A}$ for a required precision $\epsilon/2$, confidence $1/2$, and size $\widetilde{s}$ (we ignore the size of the representation of each example here). If h_i is accepted, the algorithm stops and returns h_i, otherwise it proceeds to the next iteration. Show that if at iteration i, the estimate $\widetilde{s}$ is larger than or equal to s, then $\mathbb{P}[h_i \text{ is accepted}] \geq 3/8$.

(d) Show that the probability that $\mathcal{B}$ does not halt after $j = \lceil \log \frac{2}{\delta} / \log \frac{8}{5} \rceil$ iterations with $\widetilde{s} \geq s$ is at most $\delta/2$.

(e) Show that for $i \geq \lceil 1 + (\log_2 s) \log \frac{2}{\delta} \rceil$, the inequality $\widetilde{s} \geq s$ holds.

(f) Show that with probability at least $1 - \delta$, algorithm $\mathcal{B}$ halts after at most $j' = \lceil 1 + (\log_2 s) \log \frac{2}{\delta} \rceil + j$ iterations and returns a hypothesis with error at most ϵ.

2.14 In this exercise, we generalize the notion of noise to the case of an arbitrary loss function $L \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$.

(a) Justify the following definition of the noise at point $x \in \mathcal{X}$:

$$\text{noise}(x) = \min_{y' \in \mathcal{Y}} \mathop{\mathbb{E}}_{y}[L(y, y')|x].$$

What is the value of noise(x) in a deterministic scenario? Does the definition match the one given in this chapter for binary classification?

(b) Show that the average noise coincides with the Bayes error (minimum loss achieved by a measurable function).