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, $\mathcal{Y} = \{0, 1\}$, 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 to 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 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 D. The learning problem is then formulated as follows. The learner considers a fixed set of possible concepts H, called a *hypothesis set*, which may not coincide with C. He receives a sample $S = (x_1, \ldots, x_m)$ drawn i.i.d. according to D as well as the labels $(c(x_1), \ldots, c(x_m))$, which are based on a specific target concept $c \in C$ to learn. His task is to use the labeled sample S to select a hypothesis $h_S \in H$ that has a small *generalization error* with respect to the concept c. The generalization error of a hypothesis $h \in H$, also referred to as the *true error* or just *error* of h is denoted by $R(h)$ and defined as follows.[1]

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

$$R(h) = \Pr_{x \sim D}[h(x) \neq c(x)] = \operatorname*{E}_{x \sim D}\left[1_{h(x) \neq c(x)}\right], \tag{2.1}$$

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 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 $h \in H$, a target concept $c \in C$, and a sample $S = (x_1, \ldots, x_m)$, the empirical error *or* empirical risk *of h is defined by*

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

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

$$\mathrm{E}[\widehat{R}(h)] = R(h). \tag{2.3}$$

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

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

for any x in sample S. Thus,

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

The following introduces the *Probably Approximately Correct* (PAC) learning framework. We denote by $O(n)$ an upper bound on the cost of the computational representation of any element $x \in \mathcal{X}$ and by size(c) the maximal cost of the computational representation of $c \in C$. For example, x may be a vector in $\mathbb{R}^n$, for which the cost of an array-based representation would be in $O(n)$.

Definition 2.3 **PAC-learning**
A concept class 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 D on $\mathcal{X}$ and for any target concept $c \in C$, the following holds for any sample size $m \geq poly(1/\epsilon, 1/\delta, n, size(c))$:*

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

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

A concept class 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. $\delta > 0$ 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 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 D. This is a necessary assumption for generalization to be possible in most cases.

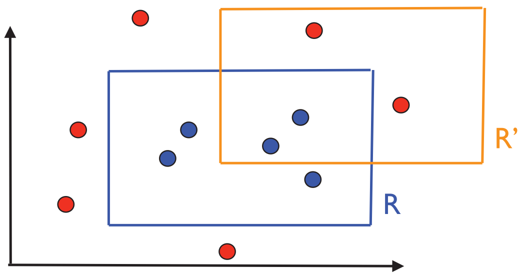


Figure 2.1 Target concept R and possible hypothesis R′. Circles represent training instances. A blue circle is a point labeled with 1, since it falls within the rectangle R. Others are red and labeled with 0.

Finally, the PAC framework deals with the question of learnability for a concept class C and not a particular concept. Note that the concept class C is known to the algorithm, but of course target concept $c \in C$ 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 2.1 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 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 positive, since its points must be included in the target concept R. Thus, the error region of R_S is included in R.

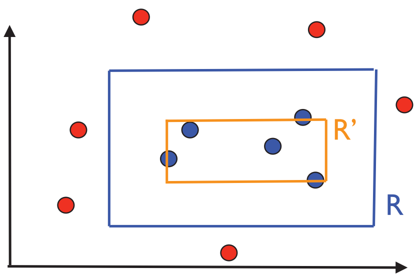


Figure 2.2 Illustration of the hypothesis $\mathsf{R}' = \mathsf{R}_S$ returned by the algorithm.

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

Now, since $\Pr[\mathsf{R}_S] > \epsilon$, we can define four rectangular regions r_1, r_2, r_3, and r_4 along the sides of R_S, each with probability at least $\epsilon/4$. These regions can be constructed by starting with the empty rectangle along a side and increasing its size until its distribution mass is at least $\epsilon/4$. Figure 2.3 illustrates the definition of these regions.

Observe that if R_S meets all of these four regions, then, because it is a rectangle, it will have one side in each of these four regions (geometric argument). Its error area, which is the part of R that it does not cover, is thus included in these regions 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, $i \in [1, 4]$. As a result, we can write

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

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

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

Thus, for any $\epsilon > 0$ and $\delta > 0$, if the sample size m is greater than $\frac{4}{\epsilon}\log\frac{4}{\delta}$, then $\Pr_{S\sim D^m}[R(\mathsf{R}_S) > \epsilon] \leq 1 - \delta$. Furthermore, the computational cost of the

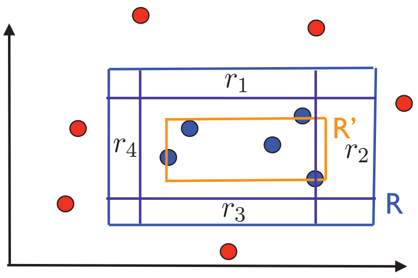


Figure 2.3 Illustration of the regions $r_1, \ldots, r_4$.

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*. It 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, if 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:

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

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 H we considered in this example coincided with the concept class 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 $|H|$ of the hypothesis set is finite. Since we consider consistent hypotheses, we will assume that the target concept c is in H.

Theorem 2.1 **Learning bounds — finite H, consistent case**
Let 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 $c \in H$ and i.i.d. sample S returns a consistent hypothesis h_S: $\widehat{R}(h_S) = 0$. Then, for any $\epsilon, \delta > 0$, the inequality $\Pr_{S \sim D^m}[R(h_S) \leq \epsilon] \geq 1 - \delta$ holds if

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

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

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

Proof Fix $\epsilon > 0$. We do not know which consistent hypothesis $h_S \in H$ 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 $h \in H$ would be consistent and have error more than ϵ:

$$\begin{aligned}
&\Pr[\exists h \in H \colon \widehat{R}(h) = 0 \wedge R(h) > \epsilon] \\
&= \Pr[(h_1 \in H, \widehat{R}(h_1) = 0 \wedge R(h_1) > \epsilon) \vee (h_2 \in H, \widehat{R}(h_2) = 0 \wedge R(h_2) > \epsilon) \vee \cdots] \\
&\leq \sum_{h \in H} \Pr[\widehat{R}(h) = 0 \wedge R(h) > \epsilon] \qquad \text{(union bound)} \\
&\leq \sum_{h \in H} \Pr[\widehat{R}(h) = 0 \mid R(h) > \epsilon]. \qquad \text{(definition of conditional probability)}
\end{aligned}$$

Now, consider any hypothesis $h \in H$ with $R(h) > \epsilon$. Then, the probability that h would be 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:

$$\Pr[\widehat{R}(h) = 0 \mid R(h) > \epsilon] \leq (1 - \epsilon)^m.$$

The previous inequality implies

$$\Pr[\exists h \in H \colon \widehat{R}(h) = 0 \wedge R(h) > \epsilon] \leq |H|(1-\epsilon)^m.$$

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 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.

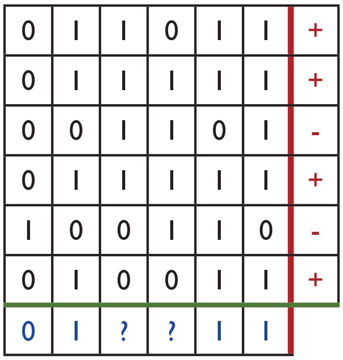
The price to pay for coming up with a consistent algorithm is the use of a larger hypothesis set H containing target concepts. Of course, the upper bound (2.9) increases with $|H|$. However, that dependency is only logarithmic. Note that the term $\log |H|$, or the related term $\log_2 |H|$ from which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

We now use theorem 2.1 to analyze PAC-learning with various concept classes.

Example 2.2 Conjunction of Boolean literals
Consider learning the concept class C_n of conjunctions of at most n Boolean literals $x_1, \ldots, x_n$. A Boolean literal is either a variable x_i, $i \in [1, n]$, 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 $i \in [1, n]$, 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 $|H| = |C_n| = 3^n$, 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 $\epsilon > 0$



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 $i \in [1, 6]$ 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 $\overline{x}_1 \wedge x_2 \wedge x_5 \wedge x_6$.

and $\delta > 0$:

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

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 $m \geq 149$. Thus, for a labeled sample of at least 149 examples, the bound guarantees 99% accuracy with a confidence of at least 98%.

Example 2.3 Universal concept class
Consider the set $\mathcal{X} = \{0, 1\}^n$ of all Boolean vectors with n components, and let 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 $|H| \geq |U_n| = 2^{(2^n)}$. Theorem 2.1 gives the following sample complexity bound:

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

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 2.4 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 C of k-term DNF formulae is 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 H must contain C for consistency to be possible, thus $|H| \geq 3^{nk}$. Theorem 2.1 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 that the problem of learning k-term DNF is in RP, the complexity class of problems that admit a randomized polynomial-time decision solution. The problem is therefore computationally intractable unless 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 unless RP = NP.

Example 2.5 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. To do so, it suffices to associate to each term T_i a new variable. Then, this can be done with the following bijection:

$$a_i(x_1) \vee \cdots \vee a_i(x_n) \rightarrow Y_{a_i(x_1),\ldots,a_i(x_n)}, \tag{2.13}$$

where $a_i(x_j)$ denotes the assignment to x_j in term T_i. This reduction to PAC-learning of conjunctions of Boolean literals may affect the original distribution, but this is not an issue since in the PAC framework no assumption is made about the distribution. Thus, 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 can be rewritten as a k-CNF formula via

$$\bigvee_{i=1}^{k} a_i(x_1) \wedge \cdots \wedge a_i(x_n) = \bigwedge_{i_1,\ldots,i_k=1}^{n} a_1(x_{i_1}) \vee \cdots \vee a_k(x_{i_k}).$$

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

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

But, as we previously saw, k-term DNF formulae are not efficiently PAC-learnable! What can explain this apparent inconsistency? Observe that the number of new variables needed to write a k-term DNF as a k-CNF formula via the transformation just described is exponential in k, it is in $O(n^k)$. The discrepancy comes from the size of the representation of a concept. A k-term DNF formula can be an exponentially more compact representation, and efficient PAC-learning is intractable if a time-complexity polynomial in that size is required. Thus, this apparent paradox deals with key aspects of PAC-learning, which include the cost of the representation of a concept and the choice of the hypothesis set.

2.3 Guarantees for finite hypothesis sets — inconsistent case

In the most general case, there may be no hypothesis in 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.1) or the following corollary, which relates the generalization error and empirical error of a single hypothesis.

Corollary 2.1
Fix $\epsilon > 0$ and let S denote an i.i.d. sample of size m. Then, for any hypothesis $h\colon X \to \{0,1\}$, the following inequalities hold:

$$\Pr_{S\sim D^m}[\widehat{R}(h) - R(h) \geq \epsilon] \leq \exp(-2m\epsilon^2) \tag{2.14}$$

$$\Pr_{S\sim D^m}[\widehat{R}(h) - R(h) \leq -\epsilon] \leq \exp(-2m\epsilon^2). \tag{2.15}$$

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

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

Proof The result follows immediately theorem D.1. ■

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 2.2* 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 $1-\delta$:

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

The following example illustrates this corollary in a simple case.

Example 2.6 Tossing a coin
Imagine tossing a biased coin that lands heads with probability p, and let our hypothesis be the one that always guesses heads. Then the true error rate is $R(h) = p$ and the empirical error rate $\widehat{R}(h) = \widehat{p}$, where $\widehat{p}$ is the empirical probability of heads based on the training sample drawn i.i.d. Thus, corollary 2.2 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 2.2 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 $\mathrm{E}[\widehat{R}(h_S)]$, 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 $h \in H$.

Theorem 2.2 **Learning bound — finite H, inconsistent case**
Let H be a finite hypothesis set. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following inequality holds:

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

Proof Let $h_1, \ldots, h_{|H|}$ be the elements of H. Using the union bound and applying corollary 2.2 to each hypothesis yield:

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

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

Thus, for a finite hypothesis set H,

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

As already pointed out, $\log_2 |H|$ can be interpreted as the number of bits needed to represent 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 $|H|$, but only logarithmically. But, here, the bound is a less favorable function of $\frac{\log_2 |H|}{m}$; it varies as the square root of this term. This is not a minor price to pay: for a fixed $|H|$, 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 consider several important questions related to the learning scenario, which we left out of the discussion of the earlier sections for simplicity.

2.4.1 Deterministic versus stochastic scenarios

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

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

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

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

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 2.4* Agnostic PAC-learning**
Let 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 D over $\mathcal{X} \times \mathcal{Y}$, the following holds for any sample size $m \geq poly(1/\epsilon, 1/\delta, n, size(c))$:*

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

If $\mathcal{A}$ further runs in poly$(1/\epsilon, 1/\delta, n, size(c))$, 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 $f : \mathcal{X} \to \mathcal{Y}$ (with probability one), then the scenario is said to be *deterministic*. In that case, it suffices to consider a distribution D over the input space. The training sample is obtained by drawing $(x_1, \ldots, x_m)$ according to D and the labels are obtained via f: $y_i = f(x_i)$ for all $i \in [1, m]$. 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 2.5 **Bayes error**
Given a distribution D over $\mathcal{X} \times \mathcal{Y}$, the Bayes error R^* *is defined as the infimum of the errors achieved by measurable functions $h : \mathcal{X} \to \mathcal{Y}$:*

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

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:

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

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

Definition 2.6 **Noise**
Given a distribution D over $\mathcal{X} \times \mathcal{Y}$, the noise *at point $x \in \mathcal{X}$ is defined by*

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

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

Thus, the average noise is precisely the Bayes error: $\text{noise} = \text{E}[\text{noise}(x)] = R^*$. 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.

2.4.3 Estimation and approximation errors

The difference between the error of a hypothesis $h \in H$ and the Bayes error can be decomposed as:

$$R(h) - R^* = \underbrace{(R(h) - R(h^*))}_{\text{estimation}} + \underbrace{(R(h^*) - R^*)}_{\text{approximation}}, \tag{2.25}$$

where h^* is a hypothesis in H with minimal error, or a *best-in-class hypothesis*.[3]

The second term is referred to as the *approximation error*, since it measures how well the Bayes error can be approximated using H. It is a property of the hypothesis set H, a measure of its richness. The approximation error is not accessible, since in general the underlying distribution D is not known. Even with various noise assumptions, estimating the approximation error is difficult.

The first term is the *estimation error*, and it depends on the hypothesis h selected. It measures the quality of the hypothesis h with respect to the best-in-class hypothesis. The definition of agnostic PAC-learning is also based on the estimation error. The *estimation error of an algorithm* $\mathcal{A}$, that is, the estimation error of the hypothesis h_S returned after training on a sample S, can sometimes be bounded in terms of the generalization error.

For example, let h_S^{ERM} denote the hypothesis returned by the empirical risk minimization algorithm, that is the algorithm that returns a hypothesis h_S^{ERM} with the smallest empirical error. Then, the generalization bound given by theorem 2.2, or any other bound on $\sup_{h \in H} |R(h) - \widehat{R}(h)|$, can be used to bound the estimation error of the empirical risk minimization algorithm. Indeed, rewriting the estimation error to make $\widehat{R}(h_S^{\text{ERM}})$ appear and using $\widehat{R}(h_S^{\text{ERM}}) \leq \widehat{R}(h^*)$, which holds by the definition of the algorithm, we can write

$$\begin{aligned} R(h_S^{\text{ERM}}) - R(h^*) &= R(h_S^{\text{ERM}}) - \widehat{R}(h_S^{\text{ERM}}) + \widehat{R}(h_S^{\text{ERM}}) - R(h^*) \\ &\leq R(h_S^{\text{ERM}}) - \widehat{R}(h_S^{\text{ERM}}) + \widehat{R}(h^*) - R(h^*) \\ &\leq 2 \sup_{h \in H} |R(h) - \widehat{R}(h)|. \end{aligned} \tag{2.26}$$

3. When H is a finite hypothesis set, h^* necessarily exists; otherwise, in this discussion $R(h^*)$ can be replaced by $\inf_{h \in H} R(h)$.

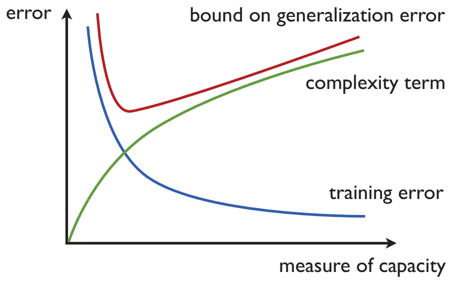


Figure 2.5 Illustration of structural risk minimization. The plots of three errors are shown as a function of a measure of capacity. Clearly, as the size or capacity of the hypothesis set increases, the training error decreases, while the complexity term increases. SRM selects the hypothesis minimizing a bound on the generalization error, which is a sum of the empirical error, and the complexity term is shown in red.

The right-hand side of (2.26) can be bounded by theorem 2.2 and increases with the size of the hypothesis set, while $R(h^*)$ decreases with $|H|$.

2.4.4 Model selection

Here, we discuss some broad model selection and algorithmic ideas based on the theoretical results presented in the previous sections. We assume an i.i.d. labeled training sample S of size m and denote the error of a hypothesis h on S by $\widehat{R}_S(h)$ to explicitly indicate its dependency on S.

While the guarantee of theorem 2.2 holds only for finite hypothesis sets, it already provides us with some useful insights for the design of algorithms and, as we will see in the next chapters, similar guarantees hold in the case of infinite hypothesis sets. Such results invite us to consider two terms: the empirical error and a complexity term, which here is a function of $|H|$ and the sample size m.

In view of that, the ERM algorithm , which only seeks to minimize the error on the training sample

$$h_S^{\mathrm{ERM}} = \operatorname*{argmin}_{h \in H} \widehat{R}_S(h), \tag{2.27}$$

might not be successful, since it disregards the complexity term. In fact, the performance of the ERM algorithm is typically very poor in practice. Additionally, in many cases, determining the ERM solution is computationally intractable. For example, finding a linear hypothesis with the smallest error on the training sample is NP-hard (as a function of the dimension of the space).

Another method known as *structural risk minimization* (SRM) consists of considering instead an infinite sequence of hypothesis sets with increasing sizes

$$H_0 \subset H_1 \subset \cdots \subset H_n \cdots \tag{2.28}$$

and to find the ERM solution h_n^{ERM} for each H_n. The hypothesis selected is the one among the h_n^{ERM} solutions with the smallest sum of the empirical error and a complexity term complexity(H_n, m) that depends on the size (or more generally the *capacity*, that is, another measure of the richness of H) of H_n, and the sample size m:

$$h_S^{\mathrm{SRM}} = \operatorname*{argmin}_{\substack{h \in H_n \\ n \in \mathbb{N}}} \widehat{R}_S(h) + \mathrm{complexity}(H_n, m). \tag{2.29}$$

Figure 2.5 illustrates the SRM method. While SRM benefits from strong theoretical guarantees, it is typically computationally very expensive, since it requires determining the solution of multiple ERM problems. Note that the number of ERM problems is not infinite if for some n the minimum empirical error is zero: The objective function can only be larger for $n' \geq n$.

An alternative family of algorithms is based on a more straightforward optimization that consists of minimizing the sum of the empirical error and a *regularization* term that penalizes more complex hypotheses. The regularization term is typically defined as $\|h\|^2$ for some norm $\|\cdot\|$ when H is a vector space:

$$h_S^{\mathrm{REG}} = \operatorname*{argmin}_{h \in H} \widehat{R}_S(h) + \lambda \|h\|^2. \tag{2.30}$$

$\lambda \geq 0$ is a *regularization parameter*, which can be used to determine the trade-off between empirical error minimization and control of the complexity. In practice, λ is typically selected using n-fold cross-validation. In the next chapters, we will see a number of different instances of such regularization-based algorithms.

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 is based on that reference.

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].