3 A Formal Learning Model

In this chapter we define our main formal learning model – the PAC learning model and its extensions. We will consider other notions of learnability in Chapter 7.

3.1 PAC Learning

In the previous chapter we have shown that for a finite hypothesis class, if the ERM rule with respect to that class is applied on a sufficiently large training sample (whose size is independent of the underlying distribution or labeling function) then the output hypothesis will be probably approximately correct. More generally, we now define *Probably Approximately Correct* (PAC) learning.

DEFINITION 3.1 (PAC Learnability) A hypothesis class $\mathcal{H}$ is PAC learnable if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$, for every distribution $\mathcal{D}$ over $\mathcal{X}$, and for every labeling function $f : \mathcal{X} \to \{0,1\}$, if the realizable assumption holds with respect to $\mathcal{H}, \mathcal{D}, f$, then when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$ and labeled by f, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the examples), $L_{(\mathcal{D},f)}(h) \leq \epsilon$.

The definition of Probably Approximately Correct learnability contains two approximation parameters. The accuracy parameter ϵ determines how far the output classifier can be from the optimal one (this corresponds to the "approximately correct"), and a confidence parameter δ indicating how likely the classifier is to meet that accuracy requirement (corresponds to the "probably" part of "PAC"). Under the data access model that we are investigating, these approximations are inevitable. Since the training set is randomly generated, there may always be a small chance that it will happen to be noninformative (for example, there is always some chance that the training set will contain only one domain point, sampled over and over again). Furthermore, even when we are lucky enough to get a training sample that does faithfully represent $\mathcal{D}$, because it is just a finite sample, there may always be some fine details of $\mathcal{D}$ that it fails

to reflect. Our accuracy parameter, ϵ, allows "forgiving" the learner's classifier for making minor errors.

Sample Complexity

The function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ determines the *sample complexity* of learning $\mathcal{H}$: that is, how many examples are required to guarantee a probably approximately correct solution. The sample complexity is a function of the accuracy (ϵ) and confidence (δ) parameters. It also depends on properties of the hypothesis class $\mathcal{H}$ – for example, for a finite class we showed that the sample complexity depends on log the size of $\mathcal{H}$.

Note that if $\mathcal{H}$ is PAC learnable, there are many functions $m_{\mathcal{H}}$ that satisfy the requirements given in the definition of PAC learnability. Therefore, to be precise, we will define the sample complexity of learning $\mathcal{H}$ to be the "minimal function," in the sense that for any ϵ, δ, $m_{\mathcal{H}}(\epsilon, \delta)$ is the minimal integer that satisfies the requirements of PAC learning with accuracy ϵ and confidence δ.

Let us now recall the conclusion of the analysis of finite hypothesis classes from the previous chapter. It can be rephrased as stating:

COROLLARY 3.2 *Every finite hypothesis class is PAC learnable with sample complexity*

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{\log(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil .$$

There are infinite classes that are learnable as well (see, for example, Exercise 3). Later on we will show that what determines the PAC learnability of a class is not its finiteness but rather a combinatorial measure called the *VC dimension*.

3.2 A More General Learning Model

The model we have just described can be readily generalized, so that it can be made relevant to a wider scope of learning tasks. We consider generalizations in two aspects:

Removing the Realizability Assumption

We have required that the learning algorithm succeeds on a pair of data distribution $\mathcal{D}$ and labeling function f provided that the realizability assumption is met. For practical learning tasks, this assumption may be too strong (can we really guarantee that there is a rectangle in the color-hardness space that *fully determines* which papayas are tasty?). In the next subsection, we will describe the *agnostic PAC* model in which this realizability assumption is waived.

Learning Problems beyond Binary Classification
The learning task that we have been discussing so far has to do with predicting a binary label to a given example (like being tasty or not). However, many learning tasks take a different form. For example, one may wish to predict a real valued number (say, the temperature at 9:00 p.m. tomorrow) or a label picked from a finite set of labels (like the topic of the main story in tomorrow's paper). It turns out that our analysis of learning can be readily extended to such and many other scenarios by allowing a variety of loss functions. We shall discuss that in Section 3.2.2 later.

3.2.1 Releasing the Realizability Assumption – Agnostic PAC Learning

A More Realistic Model for the Data-Generating Distribution
Recall that the realizability assumption requires that there exists $h^\star \in \mathcal{H}$ such that $\mathbb{P}_{x\sim\mathcal{D}}[h^\star(x) = f(x)] = 1$. In many practical problems this assumption does not hold. Furthermore, it is maybe more realistic not to assume that the labels are fully determined by the features we measure on input elements (in the case of the papayas, it is plausible that two papayas of the same color and softness will have different taste). In the following, we relax the realizability assumption by replacing the "target labeling function" with a more flexible notion, a data-labels generating distribution.

Formally, from now on, let $\mathcal{D}$ be a probability distribution over $\mathcal{X} \times \mathcal{Y}$, where, as before, $\mathcal{X}$ is our domain set and $\mathcal{Y}$ is a set of labels (usually we will consider $\mathcal{Y} = \{0, 1\}$). That is, $\mathcal{D}$ is a *joint distribution* over domain points and labels. One can view such a distribution as being composed of two parts: a distribution $\mathcal{D}_x$ over unlabeled domain points (sometimes called the *marginal distribution*) and a *conditional* probability over labels for each domain point, $\mathcal{D}((x, y)|x)$. In the papaya example, $\mathcal{D}_x$ determines the probability of encountering a papaya whose color and hardness fall in some color-hardness values domain, and the conditional probability is the probability that a papaya with color and hardness represented by x is tasty. Indeed, such modeling allows for two papayas that share the same color and hardness to belong to different taste categories.

The empirical and the True Error Revised
For a probability distribution, $\mathcal{D}$, over $\mathcal{X} \times \mathcal{Y}$, one can measure how likely h is to make an error when labeled points are randomly drawn according to $\mathcal{D}$. We redefine the true error (or risk) of a prediction rule h to be

$$L_{\mathcal{D}}(h) \stackrel{\text{def}}{=} \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] \stackrel{\text{def}}{=} \mathcal{D}(\{(x, y) : h(x) \neq y\}). \tag{3.1}$$

We would like to find a predictor, h, for which that error will be minimized. However, the learner does not know the data generating $\mathcal{D}$. What the learner does have access to is the training data, S. The definition of the empirical risk

remains the same as before, namely,

$$L_S(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m}.$$

Given S, a learner can compute $L_S(h)$ for any function $h : X \to \{0,1\}$. Note that $L_S(h) = L_{D(\text{uniform over } S)}(h)$.

The Goal

We wish to find some hypothesis, $h : \mathcal{X} \to \mathcal{Y}$, that (probably approximately) minimizes the true risk, $L_D(h)$.

The Bayes Optimal Predictor.

Given any probability distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$, the best label predicting function from $\mathcal{X}$ to $\{0,1\}$ will be

$$f_{\mathcal{D}}(x) = \begin{cases} 1 & \text{if } \mathbb{P}[y = 1|x] \geq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

It is easy to verify (see Exercise 7) that for every probability distribution $\mathcal{D}$, the Bayes optimal predictor $f_{\mathcal{D}}$ is optimal, in the sense that no other classifier, $g : \mathcal{X} \to \{0,1\}$ has a lower error. That is, for every classifier g, $L_{\mathcal{D}}(f_{\mathcal{D}}) \leq L_{\mathcal{D}}(g)$.

Unfortunately, since we do not know $\mathcal{D}$, we cannot utilize this optimal predictor $f_{\mathcal{D}}$. What the learner does have access to is the training sample. We can now present the formal definition of agnostic PAC learnability, which is a natural extension of the definition of PAC learnability to the more realistic, nonrealizable, learning setup we have just discussed.

Clearly, we cannot hope that the learning algorithm will find a hypothesis whose error is smaller than the minimal possible error, that of the Bayes predictor.

Furthermore, as we shall prove later, once we make no prior assumptions about the data-generating distribution, no algorithm can be guaranteed to find a predictor that is as good as the Bayes optimal one. Instead, we require that the learning algorithm will find a predictor whose error is not much larger than the best possible error of a predictor in some given benchmark hypothesis class. Of course, the strength of such a requirement depends on the choice of that hypothesis class.

DEFINITION 3.3 (Agnostic PAC Learnability) A hypothesis class $\mathcal{H}$ is agnostic PAC learnable if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the m training examples),

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon.$$

Clearly, if the realizability assumption holds, agnostic PAC learning provides the same guarantee as PAC learning. In that sense, agnostic PAC learning generalizes the definition of PAC learning. When the realizability assumption does not hold, no learner can guarantee an arbitrarily small error. Nevertheless, under the definition of agnostic PAC learning, a learner can still declare success if its error is not much larger than the best error achievable by a predictor from the class $\mathcal{H}$. This is in contrast to PAC learning, in which the learner is required to achieve a small error in absolute terms and not relative to the best error achievable by the hypothesis class.

3.2.2 The Scope of Learning Problems Modeled

We next extend our model so that it can be applied to a wide variety of learning tasks. Let us consider some examples of different learning tasks.

- **Multiclass Classification** Our classification does not have to be binary. Take, for example, the task of document classification: We wish to design a program that will be able to classify given documents according to topics (e.g., news, sports, biology, medicine). A learning algorithm for such a task will have access to examples of correctly classified documents and, on the basis of these examples, should output a program that can take as input a new document and output a topic classification for that document. Here, the *domain set* is the set of all potential documents. Once again, we would usually represent documents by a set of *features* that could include counts of different key words in the document, as well as other possibly relevant features like the size of the document or its origin. The *label set* in this task will be the set of possible document topics (so $\mathcal{Y}$ will be some large finite set). Once we determine our domain and label sets, the other components of our framework look exactly the same as in the papaya tasting example; Our *training sample* will be a finite sequence of (feature vector, label) pairs, the learner's output will be a function from the domain set to the label set, and, finally, for our measure of success, we can use the probability, over (document, topic) pairs, of the event that our predictor suggests a wrong label.
- **Regression** In this task, one wishes to find some simple *pattern* in the data – a functional relationship between the $\mathcal{X}$ and $\mathcal{Y}$ components of the data. For example, one wishes to find a linear function that best predicts a baby's birth weight on the basis of ultrasound measures of his head circumference, abdominal circumference, and femur length. Here, our domain set $\mathcal{X}$ is some subset of $\mathbb{R}^3$ (the three ultrasound measurements), and the set of "labels," $\mathcal{Y}$, is the the set of real numbers (the weight in grams). In this context, it is more adequate to call $\mathcal{Y}$ the *target* set. Our training data as well as the learner's output are as before (a finite sequence of (x, y) pairs, and a function from $\mathcal{X}$ to $\mathcal{Y}$ respectively). However, our measure of success is

different. We may evaluate the quality of a hypothesis function, $h : \mathcal{X} \to \mathcal{Y}$, by the *expected square difference* between the true labels and their predicted values, namely,

$$L_{\mathcal{D}}(h) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}(h(x) - y)^2. \tag{3.2}$$

To accommodate a wide range of learning tasks we generalize our formalism of the measure of success as follows:

Generalized Loss Functions

Given any set $\mathcal{H}$ (that plays the role of our hypotheses, or models) and some domain Z let ℓ be any function from $\mathcal{H} \times Z$ to the set of nonnegative real numbers, $\ell : \mathcal{H} \times Z \to \mathbb{R}_+$.

We call such functions *loss functions*.

Note that for prediction problems, we have that $Z = \mathcal{X} \times \mathcal{Y}$. However, our notion of the loss function is generalized beyond prediction tasks, and therefore it allows Z to be any domain of examples (for instance, in unsupervised learning tasks such as the one described in Chapter 22, Z is not a product of an instance domain and a label domain).

We now define the *risk function* to be the expected loss of a classifier, $h \in \mathcal{H}$, with respect to a probability distribution D over Z, namely,

$$L_{\mathcal{D}}(h) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{z\sim\mathcal{D}}[\ell(h, z)]. \tag{3.3}$$

That is, we consider the expectation of the loss of h over objects z picked randomly according to $\mathcal{D}$. Similarly, we define the *empirical risk* to be the expected loss over a given sample $S = (z_1, \ldots, z_m) \in Z^m$, namely,

$$L_S(h) \stackrel{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} \ell(h, z_i). \tag{3.4}$$

The loss functions used in the preceding examples of classification and regression tasks are as follows:

- **0–1 loss:** Here, our random variable z ranges over the set of pairs $\mathcal{X} \times \mathcal{Y}$ and the loss function is

$$\ell_{0-1}(h, (x, y)) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{if } h(x) \neq y \end{cases}$$

  This loss function is used in binary or multiclass classification problems.

  One should note that, for a random variable, α, taking the values $\{0, 1\}$, $\mathbb{E}_{\alpha\sim D}[\alpha] = \mathbb{P}_{\alpha\sim D}[\alpha = 1]$. Consequently, for this loss function, the definitions of $L_{\mathcal{D}}(h)$ given in Equation (3.3) and Equation (3.1) coincide.
- **Square Loss:** Here, our random variable z ranges over the set of pairs $\mathcal{X} \times \mathcal{Y}$ and the loss function is

$$\ell_{\text{sq}}(h, (x, y)) \stackrel{\text{def}}{=} (h(x) - y)^2.$$

This loss function is used in regression problems.

We will later see more examples of useful instantiations of loss functions.

To summarize, we formally define agnostic PAC learnability for general loss functions.

DEFINITION 3.4 (Agnostic PAC Learnability for General Loss Functions) A hypothesis class $\mathcal{H}$ is agnostic PAC learnable with respect to a set Z and a loss function $\ell : \mathcal{H} \times Z \to \mathbb{R}_+$, if there exist a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over Z, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$, the algorithm returns $h \in \mathcal{H}$ such that, with probability of at least $1 - \delta$ (over the choice of the m training examples),

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon,$$

where $L_{\mathcal{D}}(h) = \mathbb{E}_{z \sim \mathcal{D}}[\ell(h, z)]$.

Remark 3.1 (A Note About Measurability*) In the aforementioned definition, for every $h \in \mathcal{H}$, we view the function $\ell(h, \cdot) : Z \to \mathbb{R}_+$ as a random variable and define $L_{\mathcal{D}}(h)$ to be the expected value of this random variable. For that, we need to require that the function $\ell(h, \cdot)$ is measurable. Formally, we assume that there is a σ-algebra of subsets of Z, over which the probability $\mathcal{D}$ is defined, and that the preimage of every initial segment in $\mathbb{R}_+$ is in this σ-algebra. In the specific case of binary classification with the 0−1 loss, the σ-algebra is over $\mathcal{X} \times \{0,1\}$ and our assumption on ℓ is equivalent to the assumption that for every h, the set $\{(x, h(x)) : x \in \mathcal{X}\}$ is in the σ-algebra.

Remark 3.2 (Proper versus Representation-Independent Learning*) In the preceding definition, we required that the algorithm will return a hypothesis from $\mathcal{H}$. In some situations, $\mathcal{H}$ is a subset of a set $\mathcal{H}'$, and the loss function can be naturally extended to be a function from $\mathcal{H}' \times Z$ to the reals. In this case, we may allow the algorithm to return a hypothesis $h' \in \mathcal{H}'$, as long as it satisfies the requirement $L_{\mathcal{D}}(h') \leq \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$. Allowing the algorithm to output a hypothesis from $\mathcal{H}'$ is called *representation independent* learning, while proper learning occurs when the algorithm must output a hypothesis from $\mathcal{H}$. Representation independent learning is sometimes called "improper learning," although there is nothing improper in representation independent learning.

3.3 Summary

In this chapter we defined our main formal learning model – PAC learning. The basic model relies on the realizability assumption, while the agnostic variant does

not impose any restrictions on the underlying distribution over the examples. We also generalized the PAC model to arbitrary loss functions. We will sometimes refer to the most general model simply as PAC learning, omitting the "agnostic" prefix and letting the reader infer what the underlying loss function is from the context. When we would like to emphasize that we are dealing with the original PAC setting we mention that the realizability assumption holds. In Chapter 7 we will discuss other notions of learnability.

3.4 Bibliographic Remarks

Our most general definition of agnostic PAC learning with general loss functions follows the works of Vladimir Vapnik and Alexey Chervonenkis (Vapnik & Chervonenkis 1971). In particular, we follow Vapnik's general setting of learning (Vapnik 1982, Vapnik 1992, Vapnik 1995, Vapnik 1998).

PAC learning was introduced by Valiant (1984). Valiant was named the winner of the 2010 Turing Award for the introduction of the PAC model. Valiant's definition requires that the sample complexity will be polynomial in $1/\epsilon$ and in $1/\delta$, as well as in the representation size of hypotheses in the class (see also Kearns & Vazirani (1994)). As we will see in Chapter 6, if a problem is at all PAC learnable then the sample complexity depends polynomially on $1/\epsilon$ and $\log(1/\delta)$. Valiant's definition also requires that the *runtime* of the learning algorithm will be polynomial in these quantities. In contrast, we chose to distinguish between the statistical aspect of learning and the computational aspect of learning. We will elaborate on the computational aspect later on in Chapter 8, where we introduce the full PAC learning model of Valiant. For expository reasons, we use the term PAC learning even when we ignore the runtime aspect of learning. Finally, the formalization of agnostic PAC learning is due to Haussler (1992).

3.5 Exercises

1. **Monotonicity of Sample Complexity:** Let $\mathcal{H}$ be a hypothesis class for a binary classification task. Suppose that $\mathcal{H}$ is PAC learnable and its sample complexity is given by $m_{\mathcal{H}}(\cdot,\cdot)$. Show that $m_{\mathcal{H}}$ is monotonically nonincreasing in each of its parameters. That is, show that given $\delta \in (0,1)$, and given $0 < \epsilon_1 \leq \epsilon_2 < 1$, we have that $m_{\mathcal{H}}(\epsilon_1,\delta) \geq m_{\mathcal{H}}(\epsilon_2,\delta)$. Similarly, show that given $\epsilon \in (0,1)$, and given $0 < \delta_1 \leq \delta_2 < 1$, we have that $m_{\mathcal{H}}(\epsilon,\delta_1) \geq m_{\mathcal{H}}(\epsilon,\delta_2)$.
2. Let $\mathcal{X}$ be a discrete domain, and let $\mathcal{H}_{\text{Singleton}} = \{h_z : z \in \mathcal{X}\} \cup \{h^-\}$, where for each $z \in \mathcal{X}$, h_z is the function defined by $h_z(x) = 1$ if $x = z$ and $h_z(x) = 0$ if $x \neq z$. h^- is simply the all-negative hypothesis, namely, $\forall x \in X$, $h^-(x) = 0$. The realizability assumption here implies that the true hypothesis f labels negatively all examples in the domain, perhaps except one.