Machine Learning: Foundations and Algorithms

Shai Ben-David and Shai Shalev-Shwartz

DRAFT

Preface

The term *machine learning* refers to the automated detection of meaningful patterns in data. In the past couple of decades it has become a common tool in almost any task that requires information extraction from large data sets. We are surrounded by a machine learning based technology: search engines learn how to bring us the best results (while placing profitable ads), anti-spam software learns to filter our email messages, and credit card transactions are secured by a software that learns how to detect frauds. Digital cameras learn to detect faces and intelligent personal assistance applications on smart-phones learn to recognize voice commands. Cars are equipped with accident prevention systems that are built using machine learning algorithms. Machine learning is also widely used in scientific applications such as bioinformatics and astronomy.

One common feature of all of these applications is that, in contrast to more traditional uses of computers, in these cases, due to the complexity of the patterns that need to be detected, a human programmer cannot provide an explicit, fine-detailed, specification of how such tasks should be executed. Taking example from intelligent beings, many of our skills are acquired or refined through *learning* from our experience (rather than following explicit instructions given to us). Machine learning tools are concerned with endowing programs with the ability to "learn" and adapt.

The first goal of this book is to provide a rigorous, yet easy to follow, introduction to the main concepts underlying machine learning: What is learning? How can a machine learn? How do we quantify the resources needed to learn a given concept? Is learning always possible? Can we know if the learning process succeeded or failed?

The second goal of this book is to present several key machine learning algorithms. We chose to present algorithms that on one hand are successfully used in practice and on the other hand give a wide spectrum of different learning techniques. Additionally, we pay specific attention to algorithms appropriate for large scale learning, since in recent years, our world has become increasingly "digitized" and the amount of data available for learning is dramatically increasing. As a result, in many applications data is plentiful and computation time is the main bottleneck.

The book is divided into four parts. The first part aims at giving an initial rigorous answer to the fundamental questions of learning. We describe a generalization of Valiant's Probably Approximately Correct (PAC) learning model, which is a first solid answer to the question "what is learning?". We describe the

Empirical Risk Minimization (ERM) learning rule, which shows "how can a machine learn". We also quantify the amount of data needed for learning using the ERM rule and show how learning might fail by deriving a "no-free-lunch" theorem. In the second part of the book we describe various learning algorithms. For many of the algorithms, we first present a more general learning principle, and then show how the algorithm follows the principle. While the first two parts of the book focus on the PAC model, the third part extends the scope by presenting a wider variety of learning models. Finally, the last part of the book is devoted to advanced theory.

We made an attempt to keep the book as self-contained as possible. However, the reader is assumed to be comfortable with basic notions of probability, linear algebra, and algorithms. The first three parts of the book are intended for first year graduate students in computer science, engineering, mathematics, or statistics. It can also be accessible to undergraduate students with the adequate background. The more advanced chapters can be used by researchers intending to gather a deeper theoretical understanding.

Contents

Chapter 1

Introduction

The subject of this book is automated learning, or, as we will more often call it, Machine Learning (ML). That is, we wish to program computers so that they can "learn" from input available to them. Roughly speaking, learning is the process of converting experience into expertise or knowledge. The input to a learning algorithm is training data, representing experience, and the output is some expertise, which usually takes the form of another computer program that can perform some task. Seeking a formal-mathematical understanding of this concept, we'll have to be more explicit about what we mean by each of the involved terms; What is the training data our programs will access? How can the process of learning be automated? How can we evaluate the success of such a process (namely the quality of the output of a learning program)?

1.1 What is learning?

Let us begin by considering a couple of examples from naturally occurring animal learning. Some of the most fundamental issues in ML arise already in that context, that we are all familiar with.

Bait Shyness—rats learning to avoid poisonous baits: When rats encounter food items with novel look or smell, they will first eat very small amounts, and subsequent feeding will depend on the flavor of the food and its physiological effect. If the food produces an ill effect, the novel food will often be associated with the illness, and subsequently, the rats will not eat it. Clearly, there is a learning mechanism in play here – the animal used past experience with some food to acquire expertise in detecting the safety of this food. If past experience with the

food was negatively labeled, the animal predicts that it will also have a negative effect when encountered in the future.

Inspired by the above example of successful learning, let us demonstrate a typical machine learning task. Suppose we would like to program a machine that learns how to filter spam emails. A naive solution would be seemingly similar to the way rats learn how to avoid poisonous baits. The machine would simply *memorize* all previous emails, that had been labeled as spam emails by the human user. When a new email arrives, the machine would search for it in the set of previous spam emails. If it matches one of them it will be trashed. Otherwise, it will be moved to the user's inbox folder.

While the above "learning by memorization" approach is sometimes useful, as we will not get the very same spam email twice, it lacks an important aspect of learning systems—the ability to label unseen email messages. A successful learner should be able to progress from individual examples to broader *generalization*. This is also referred to as *inductive reasoning* or *inductive inference*. In the bait shyness example presented above, after the rats encounter an example of a certain type of food, they apply their attitude towards it on new, unseen examples of food of similar smell and taste. To achieve generalization in the spam filtering task, the learner can scan the previously seen emails, and extract a set of words whose appearance in an email message is indicative of spam. Then, when a new email arrives, the machine can check if one of the suspicious words appear in it, and predict its label accordingly. Such a system would potentially be able to correctly predict the label of unseen emails.

Inductive reasoning might lead us to false conclusions. To illustrate this, let us consider again an example from animal learning.

Pigeon superstition: In an experiment performed by the psychologist B.F. Skinner, he placed a bunch of hungry pigeons in a cage. An automatic mechanism has been attached to the cage, delivering food to the pigeons at regular intervals with no reference whatsoever to the birds' behavior. The hungry pigeons go around the cage, and when food is first delivered, it finds each pigeon engaged in some activity (pecking, turning the head, etc.). The arrival of food reinforces each bird's specific action, and consequently, each bird tends to spend some more time doing that very same action. That, in turn, increases the chance that the next random food delivery will find each bird engaged in that activity again. What results is a chain of events that reinforces the pigeons' association of the delivery of the food with whatever chance actions they had been performing when it was first

delivered. They subsequently continue to perform these same actions diligently.[1]

What distinguishes learning mechanisms that result in superstition from useful learning? This question is crucial to the development of automated learners. While human learners can rely on common sense to filter out random meaningless learning conclusions, once we export the task of learning to a machine, we must provide well defined crisp principles that will protect the program from reaching senseless/useless conclusions. The development of such principles is a central goal of the theory of machine learning.

What, then, made the rats' learning more successful than that of the pigeons? As a first step towards answering this question, let us have a closer look at the bait shyness phenomenon in rats.

Bait Shyness revisited—rats fail to acquire conditioning between food and electric shock or between sound and nausea: The bait shyness mechanism in rats turns out to be more complex than what one may expect. In experiments carried out by Garcia ([**?**]), it was demonstrated that if the unpleasant stimulus that follows food consumption is replaced by, say, electrical shock (rather than nausea), then no conditioning occurs. Even after repeated trials in which the consumption of some food is followed by the administration of unpleasant electrical shock, the rats do not tend to avoid that food. Similar failure of conditioning occurs when the characteristic of the food that implies nausea (such as taste or smell) is replaced by a vocal signal. The rats seem to have some "built in" prior knowledge telling them that, while temporal correlation between food and nausea can be causal, it is unlikely that there will be a causal relationship between food consumption and electrical shocks or between sounds and nausea.

We conclude that one distinguishing feature between the bait shyness learning and the pigeon superstition is the incorporation of *prior knowledge* that biases the learning mechanism. This is also referred to as *inductive bias*. The pigeons in the experiment are willing to adopt *any* explanation to the occurrence of food. However, the rats "know" that food cannot cause an electric shock and that the co-occurrence of noise with some food is not likely to effect the nutritional value of that food. The rats' learning process is biased towards detecting some kind of patterns while ignoring other temporal correlations between events.

It turns out that the incorporation of prior knowledge, biasing the learning process, is inevitable for the success of learning algorithms (this is formally stated and proved as the "No Free Lunch theorem" in Chapter **??**). The development of tools for expressing domain expertise, translating it into a learning bias, and quan-

[1] See: http://psychclassics.yorku.ca/Skinner/Pigeon

tifying the effect of such a bias on the success of learning, is a central theme of the theory of machine learning. Roughly speaking, the stronger the prior knowledge (or prior assumptions) that one starts the learning process with, the easier it is to learn from further examples. However, the stronger these prior assumptions are, the less flexible the learning is - it is bound, a priory, by the commitment to these assumptions. We shall discuss these issues explicitly in Chapter **??**.

1.2 When do we need machine learning?

When do we need machine learning rather than directly program our computers to carry out the task at hand? Two aspects of a given problem may call for the use of programs that learn and improve based on their "experience": the problem's complexity and the need for adaptivity.

Tasks that are too complex to program.

- *Tasks performed by animals/humans:* there are numerous tasks that we, human beings, perform routinely, yet our introspection concerning how we do them is not sufficiently elaborate to extract a well defined program. Examples of such tasks include driving, speech recognition, and image understanding. In all of these tasks, state of the art machine learning programs, programs that "learns from their experience", achieve quite satisfactory results, once exposed to sufficiently many training examples.
- *Tasks beyond human capabilities:* another wide family of tasks that benefit from machine learning techniques are related to the analysis of very large and complex data sets: Astronomical data, turning medical archives into medical knowledge, weather prediction, analysis of genomic data, web search engines, and electronic commerce. With more and more available digitally recorded data, it becomes obvious that there are treasures of meaningful information buried in data archives that are way too large and too complex for humans to make sense of. Learning to detect meaningful patterns in large and complex data sets is a promising domain in which the combination of programs that learn with the almost unlimited memory capacity and ever increasing processing speed of computers open up new horizons.

Adaptivity. One limiting feature of programmed tools is their rigidity - once the program has been written down and installed, it stays unchanged. However, many tasks change over time or from one user to another. Machine learning tools - programs whose behavior adapts to their input data - offer a solution to such issues; they are, by nature, adaptive to changes in the environment they interact with. Typical successful applications of machine learning to such problems include programs that decode hand written text, where a fixed program can adapt to variations between the handwriting of different users, spam detection programs, adapting automatically to changes in the nature of spam emails, and speech recognition programs.

1.3 Types of learning

Learning is, of course, a very wide domain. Consequently, the field of machine learning has branched into several subfields dealing with different types of learning tasks. We give a rough taxonomy of learning paradigms, aiming to provide some perspective of where the content of this book sits within the wide field of machine learning.

We describe four parameters along which learning paradigms can be classified.

Supervised vs. Unsupervised Since learning involves an interaction between the learner and the environment, one can divide learning tasks according to the nature of that interaction. The first distinction to note is the difference between supervised and unsupervised learning. As an illustrative example, consider the task of learning to detect spam email versus the task of anomaly detection. For the spam detection task, we consider a setting in which the learner receives training emails for which the label `spam/not-spam` is provided. Based on such training the learner should figure out a rule for labeling a newly arriving email message. In contrast, for the task of anomaly detection, all the learner gets as training is a large body of email messages (with no labels) and the learner's task is to detect "unusual" messages.

More abstractly, viewing learning as a process of "using experience to gain expertise", supervised learning describes a scenario in which the "experience", a training example contains significant information (say, the `spam/not-spam` labels) that is missing in the unseen "test examples" to which the learned expertise is to be applied. In this setting, the acquired expertise is aimed to predict that missing information for the test data. In such

cases, we can think of the environment as a teacher that "supervises" the learner by providing the extra information (labels). In unsupervised learning, however, there is no distinction between training and test data. The learner processes input data with the goal of coming up with some summary, or compressed version of that data. Clustering a data set into subsets of similar objets is a typical example of such a task.

There is also an intermediate learning setting in which, while the training examples contain more information than the test examples, the learner is required to predict even more information for the test examples. For example, one may try to learn a value function, that describes for each setting of a chess board the degree by which White's position is better than the Black's. Yet, the only information available to the learner at training time is positions that occurred throughout actual chess games, labeled by who eventually won that game. Such learning framework are mainly investigated under the title of *reinforcement learning*.

Active vs. Passive learners Learning paradigms can vary by the role played by the learner. We distinguish between 'active' and 'passive' learners. An active learner interacts with the environment at training time, say by posing queries or performing experiments, while a passive learner only observes the information provided by the environment (or the teacher) without influencing or directing it. Note that, the learner of a spam filter is usually passive - waiting for users to mark the emails arriving to them. In an active setting, one could imagine asking users to label specific emails chosen by the learner, or even composed by the learner to enhance its understanding of what spam is.

Helpfulness of the teacher When one thinks about human learning, of a baby at home, or a student at school, the process often involves a helpful teacher. A teacher trying to feed the learner with the information most useful for achieving the learning goal. In contrast, when a scientist learns about nature, the environment, playing the role of the teacher, can be best thought of as passive - apples drop, stars shine and the rain falls without regards to the needs of the learner. We model such learning scenarios by postulating that the training data (or the learner's experience) is generated by some random process. This is the basic building block in the branch of 'statistical learning'. Finally, learning also occurs when the learner's input is generated by an adversarial "teacher". This may be the case in the spam filtering exam-

ple (if the spammer makes an effort to mislead the spam filtering designer) or in learning to detect fraud. One also uses an adversarial teacher model as a worst-case-scenario, when no milder setup can be safely assumed. If you can learn against an adversarial teacher, you are guaranteed to succeed interacting any odd teacher.

Online vs. Batch learning protocol The last parameter we mention is the distinction between situations in which the learner has to respond online, throughout the learning process, to settings in which the learner has to engage the acquired expertise only after having a chance to process large amounts of data. For example, a stock broker has to make daily decisions, based on the experience collected so far. He may become an expert over time, but might have made costly mistakes in the process. In contrast, in many data mining settings, the learner - the data miner - has large amounts of training data to play with before having to output conclusions.

In this book we shall discuss only a subset of the possible learning paradigms. Our main focus is on supervised statistical batch learning with a passive learner (like for example, trying to learn how to generate patients' prognosis, based on large archives of records of patients that were independently collected and are already labeled by the fate of the recorded patients). We shall also briefly discuss online learning and batch unsupervised learning (in particular, clustering).

1.4 Relations to other fields

As an interdisciplinary field, machine learning share common threads with the mathematical fields of statistics, information theory, game theory, and optimization. It is naturally a sub-field of computer science, as our goal is to program machines so that they will learn. In a sense, machine learning can be viewed as a branch of AI (Artificial Intelligence), since after all, the ability to turn experience into expertise or to detect meaningful patterns in complex sensory data is a corner stone of human (and animal) intelligence. However, one should note that, in contrast with traditional AI, machine learning is not trying to build automated imitation of intelligent behavior, but rather to use the strengths and special abilities of computers to complement human intelligence, often performing tasks that fall way beyond human capabilities. For example, the ability to scan and process huge databases allows machine learning programs to detect patterns that are outside the scope of human perception.

The component of experience, or training, in machine learning often refers to data that is randomly generated. The task of the learner is to process such randomly generated examples towards drawing conclusions that hold for the environment from which these examples are picked. This description of machine learning highlights its close relationship with statistics. Indeed there is a lot in common between the two disciplines, in terms of both the goals and techniques used. There are, however, a few significant differences in emphasis; If a doctor comes up with the hypothesis that there is a correlation between smoking and heart disease, its the statistician's role to view samples of patients and check the validity of that hypothesis (this is the common statistical task of hypothesis testing). In contrast, machine learning aims to use the data gathered from samples of patients to come up with a description of the causes of heart disease. The hope is that automated techniques may be able to figure out meaningful patterns (or hypotheses) that may have been missed by the human observer.

In contrast with traditional statistics, in machine learning in general, and in this book in particular, algorithmic considerations play a major role. Machine learning is about the execution of learning by computers, hence algorithmic issues are pivotal. We develop algorithms to perform the learning tasks and are concerned with their computational efficiency. Another difference is that while statistics is often interested in asymptotic behavior (like the convergence of sample-based statistical estimates as the sample sizes grow to infinity), the theory of machine learning focuses on finite sample bounds. Namely, given the size of available samples, the machine learning theory will aim to figure out the degree of accuracy that a learner can expect based on such samples.

There are further differences between these two disciplines, of which we shall mention only one more here. While in statistics it is common to work under the assumption of certain pre-subscribed data models (such as assuming the normality of data-generating distributions, or the linearity of functional dependencies), in machine learning the emphasis is on working under "distribution-free" setting, where the learner assumes as little as possible about the nature of the data distribution and allows the learning algorithm to figure out which models best approximate the data generating process. A precise discussion of this issue requires some technical preliminaries, and we will come back to it along the book, and in particular in Chapter **??**.

1.5 How to read this book

The first part of the book provides the basic theoretical principles that underlie machine learning. In a sense, this is the foundation upon which the rest of the book builds, and, with the possible exception of Chapter **??**, is less technical than the later sections of the book. This part could serve as a basis for a mini-course on the theoretical foundations of ML for general science students.

The first 5 chapters of the second part of the book introduce the most basic and "traditional" algorithmic approaches to machine learning. These chapters may also be used for introducing machine learning in a general AI course to CS or Math students. The later chapters of the second part of the book cover the most commonly used algorithmic paradigms of machine learning in the past 5-10 years. This part is suitable for students that have a particular interest in machine learning (either applied or theoretical). The third part of the book extends the scope of discussion from statistical classification prediction to other learning models. Finally, the last, part of the book, Advanced Theory, is geared towards readers who have interest in research and provides the more technical mathematical techniques that serve to analyze and drive forward the field of theoretical machine learning.

Part I

Foundations

Chapter 2

A gentle start

Let us begin our mathematical analysis by showing how successful learning can be achieved in a relatively simplistic setting. Imagine you have just arrived in some small Pacific island. You soon find out that papayas are a significant ingredient in the local diet. However, you have never before tasted papayas. You have to learn how to predict whether a papaya you see in the market is tasty or not. First, you need to decide which features of a papaya should your prediction be based on. Based on your previous experience with other fruits, you decide to use two features; the papaya's color, ranging from dark green, through orange and red to dark brown, and the papaya's softness, ranging from rock hard to mushy. Your input for figuring out your prediction rule is a sample of papayas that you have examined for color and softness and then tasted and found out if they were tasty or not. Let us analyze this task as a demonstration of the considerations involved in learning problems.

Our first step is to describe a formal model aimed to capture such learning tasks.

2.1 A Formal Model - the statistical learning framework

The Learner's Input: In the basic statistical learning setting, the learner has access to the following:

Domain Set: An arbitrary set, $\mathcal{X}$. This is the set of objects that we may wish to label. For example, these could be papayas that we wish to

classify as tasty or not-tasty, or email messages that we wish to classify as spam or not-spam. Usually, these domain points will be represented by a vector of *features* (like the papaya's color and softness). We also refer to domain points as *instances*.

Label Set: For our current discussion, we will restrict the label set to be a two-element set, usually, $\{0, 1\}$ or $\{-1, +1\}$. Let $\mathcal{Y}$ denote our set of possible labels. For our papayas example, let $\mathcal{Y}$ be $\{0, 1\}$, where 1 represents being tasty and 0 stands for being not-tasty.

Training data: $S = ((\mathbf{x}_1, y_1) \ldots (\mathbf{x}_m, y_m))$ is a finite sequence of pairs in $\mathcal{X} \times \mathcal{Y}$. That is, a sequence of labeled domain points. This is the input that the learner has access to (like a set of papayas that have been tasted and their color, softness and tastiness). Such labeled examples are often called *training examples*.

The Learner's Output: The learner is requested to output a *prediction rule*, $h : \mathcal{X} \to \mathcal{Y}$. This function is also called a *predictor*, a *hypothesis*, or a *classifier*. The predictor can be used to predict the label of new domain points. In our papayas example, it is a rule that our learner will employ to predict whether future papayas he examines in the farmers market are going to be tasty or not.

A simple data-generation model We now explain how the training data is generated. First, we assume that the instances (the papayas we encounter) are generated by some probability distribution (in this case, representing the environment). Let us denote that probability distribution over $\mathcal{X}$ by $\mathcal{D}$. It is important to note that we do not assume that the learner knows anything about this distribution. For the type of learning tasks we discuss, this could be any arbitrary probability distribution. As to the labels, in the current discussion we assume that there is some "correct" labeling function, $f : \mathcal{X} \to \mathcal{Y}$, and that $y_i = f(\mathbf{x}_i)$ for all i. This assumption will be relaxed in the next chapter. The labeling function is unknown to the learner. In fact, this is just what the learner is trying to figure out. In summary, each pair in the training data S is generated by first sampling a point $\mathbf{x}_i$ according to $\mathcal{D}$ and then labeling it by f.

Measures of success: We define the *error of a classifier* to be the probability that it does not predict the correct label on a random data point generated by the aforementioned underlying distribution. That is, the error of h is the

probability to draw a random instance $\mathbf{x}$, according to the distribution $\mathcal{D}$, such that $h(\mathbf{x})$ does not equal to $f(\mathbf{x})$.

Formally, given a domain subset[1], $A \subset \mathcal{X}$, the probability distribution, $\mathcal{D}$, assigns a number, $\mathcal{D}(A)$, which determines how likely it is to observe a point $\mathbf{x} \in A$. In many cases, we refer to A as an event and express it using a function $\pi : \mathcal{X} \to \{0, 1\}$, namely, $A = \{\mathbf{x} \in \mathcal{X} : \pi(\mathbf{x}) = 1\}$. In that case, we also use the notation $\mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[\pi(\mathbf{x})]$ to express $\mathcal{D}(A)$.

We define the error of a prediction rule, $h : \mathcal{X} \to \mathcal{Y}$ to be:

$$L_{\mathcal{D},f}(h) \stackrel{\text{def}}{=} \mathbb{P}_{\mathbf{x}\sim\mathcal{D}}[h(\mathbf{x}) \neq f(\mathbf{x})] \stackrel{\text{def}}{=} \mathcal{D}(\{\mathbf{x} : h(\mathbf{x}) \neq f(\mathbf{x})\}) \,. \tag{2.1}$$

That is, the error of such h is the probability to randomly choose an example $\mathbf{x}$ for which $h(\mathbf{x}) \neq f(\mathbf{x})$. The subscript $(\mathcal{D}, f)$ indicates that the error is measured with respect to the probability distribution $\mathcal{D}$ and the correct labeling function f. We omit this subscript when it is clear from the context. $L_{(\mathcal{D},f)}(h)$ has several synonymous names such as the *generalization error*, the *risk*, or the *true error* of h. We use the letter L for the error, since we view this error as the *loss* of the learner. We will later also discuss other possible formulations of such loss.

A note about the information available to the learner The learner is blind to the underlying distribution $\mathcal{D}$ over the world and to the labeling function f. In our papayas example, we have just arrived to a new island and we have no clue as to how papayas are distributed and how to predict their tastiness. The only way the learner can interact with the environment is through observing the training set.

2.2 Empirical Risk Minimization

Next, we describe a simple learning paradigm for the above setup and analyze its performance.
Recall that a learning algorithm receives as input a training set S, sampled from an unknown distribution $\mathcal{D}$ and labeled by some target function f, and should output

[1] Strictly speaking, we should be more careful and require that A is a member of some σ-algebra of subsets of $\mathcal{X}$, over which $\mathcal{D}$ is defined. We will formally define our measurability assumptions in the next chapter.

a predictor $h_S : \mathcal{X} \to \mathcal{Y}$ (the subscript S emphasizes the fact that the output predictor depends on S). The goal of the algorithm is to find h_S that minimizes the error with respect to the unknown $\mathcal{D}$ and f.

Since the learner does not know what $\mathcal{D}$ and f are, the true error is not directly available to the learner. A useful notion of error that can be calculated by the learner is the *training error* - the error the classifier incurs over the training sample:

$$L_S(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(\mathbf{x}_i) \neq y_i\}|}{m}, \tag{2.2}$$

where $[m] = \{1, \ldots, m\}$.

The terms *empirical error*, or *empirical risk*, are often used interchangeably for this error.

Since the training sample is the snapshot of the world that is available to the learner, it makes sense to search for a solution that works well on that data. This learning paradigm (coming up with a predictor h that minimizes $L_S(h)$) is called *Empirical Risk Minimization* or ERM for short.

2.2.1 Something may go wrong - overfitting

Although the ERM rule seems very natural, without being careful, this approach may fail miserably.

To demonstrate such a failure, let us go back to the problem of learning to predict the taste of a papaya based on its shape and color. Consider a sample as depicted in Figure **??**. Assume that the probability distribution $\mathcal{D}$ is such that instances are distributed uniformly within the gray square and the labeling function determines the label to be 1 if the instance is within the inner blue square, and 0 otherwise. The area of the gray square in the picture is 2 and the area of the blue square is 1. Consider the following predictor:

$$h_S(\mathbf{x}) = \begin{cases} y_i & \text{if } \exists i \text{ s.t. } \mathbf{x}_i = \mathbf{x} \\ 0 & \text{otherwise} \end{cases}. \tag{2.3}$$

Clearly, no matter what the sample is, $L_S(h_S) = 0$, and therefore this predictor may be chosen by an ERM algorithm (it is one of the empirical-minimum-cost hypotheses, no classifier can have smaller error). On the other hand, the true error of any classifier that predicts the label 1 only on a finite number of instances is,

in this case, $1/2$. Thus, $L_{\mathcal{D}}(h_S) = 1/2$. We have found a predictor whose performance on the training set is excellent, yet its performance on the true "world" is very poor. This phenomenon is called *overfitting*. Intuitively, overfitting occurs when our hypothesis fits the training data "too well" (perhaps like the everyday experience that a person that provides a perfect detailed explanation for every single action of his may raise suspicion).

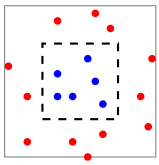

Figure 2.1: An illustration of a sample for the Papaya taste learning problem.

2.3 Empirical Risk Minimization with inductive bias

We have just demonstrated that the ERM rule might lead to overfitting. Rather than giving up on the ERM paradigm, we will look for ways to rectify it. We will search for conditions under which there is a guarantee that ERM does not overfit. Namely, conditions under which when the ERM predictor has good performance with respect to the training data, it is also highly likely to perform well over the underlying data distribution.

A common solution is to apply the ERM learning rule over a restricted search space. Formally, the learner should choose in advance (before seeing the data) a set of predictors. This set is called a *hypothesis class* and is denoted by $\mathcal{H}$. Each $h \in \mathcal{H}$ is a function mapping from $\mathcal{X}$ to $\mathcal{Y}$. For a given class $\mathcal{H}$, and a training sample, S, the $\mathrm{ERM}_{\mathcal{H}}$ learner uses the ERM rule to choose a predictor $h \in \mathcal{H}$, with as low as possible error over S. Formally,

$$\mathrm{ERM}_{\mathcal{H}}(S) \in \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h) \ ,$$

where argmin stands for the set of hypotheses in $\mathcal{H}$ that achieves the minimum value of $L_S(h)$ over $\mathcal{H}$. By restricting the learner to choosing a predictor from $\mathcal{H}$, we *bias* it toward a particular set of predictors. Such restrictions are often called

an *inductive bias*. Since the choice of such a restriction is determined before the learner sees the training data, it should ideally be based on some prior knowledge about the problem to be learnt. For example, for the Papaya taste prediction problem we may choose the class $\mathcal{H}$ to be the set of predictors which are determined by axis aligned rectangles (in the space determined by the color and softness coordinates). We will later show that $\mathrm{ERM}_{\mathcal{H}}$ over this class is guaranteed not to overfit. On the other hand, the example of overfitting that we have seen above, demonstrates that choosing $\mathcal{H}$ to be a class of predictors that includes all functions that assign the value 1 to a finite set of domain points, does not suffice to guarantee that $\mathrm{ERM}_{\mathcal{H}}$ will not overfit.

A fundamental question in learning theory is, over which hypothesis classes $\mathrm{ERM}_{\mathcal{H}}$ learning will not result in overfitting. We will study this question later in the book.

Intuitively, choosing a more restricted hypothesis class better protects us against overfitting but at the same time might cause us a larger inductive bias. We will get back to this fundamental tradeoff later.

2.3.1 Finite hypothesis classes

The simplest type of restriction on a class is imposing an upper bound on its size (that is, the number of predictors h in $\mathcal{H}$). In this section, we show that if $\mathcal{H}$ is a finite class then $\mathrm{ERM}_{\mathcal{H}}$ will not overfit, provided it is based on a sufficiently large training sample (this size requirement will depend on the size of $\mathcal{H}$).

Limiting the learner to prediction rules within some finite hypothesis class may be considered as a reasonably mild restriction. For example, $\mathcal{H}$ can be the set of all predictors that can be implemented by a C++ program written in at most 1000000000 bits of code. In our papayas example, we mentioned previously the class of axis aligned rectangles. While this is an infinite class, if we discretize the representation of real numbers, say by using a 64 bits floating-point representation, the hypothesis class becomes a finite class.

Let us now analyze the performance of the $\mathrm{ERM}_{\mathcal{H}}$ learning rule assuming that $\mathcal{H}$ is a finite class. For a training sample, S, labeled according to some $f : \mathcal{X} \to \mathcal{Y}$, let h_S denote a result of applying ERM_H to S. Namely,

$$h_S \in \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h) \,, \tag{2.4}$$

In this chapter, we make the next simplifying assumption (that will be relaxed in the next Chapter).

The Realizability assumption: There exists $h^\star \in \mathcal{H}$ s.t. $L_{(\mathcal{D},f)}(h^\star) = 0$. Note that this assumption implies that with probability 1 over random samples, S, where the instances of S are sampled according to $\mathcal{D}$ and are labeled by f, we have $L_S(h^\star) = 0$.

For every sample, S, since the realizability assumption implies that there exists some $h \in \mathcal{H}$ for which $L_S(h) = 0$ and h_S is a minimizer of the sample error, $L_S(h_S) = 0$. However, we are interested in the *true* loss of h_S, $L_{(\mathcal{D},f)}(h_S)$, rather than its empirical loss.

Clearly, any guarantee on the error with respect to the underlying distribution, $\mathcal{D}$ for an algorithm that has access only to a sample S, should depend on the relationship between $\mathcal{D}$ and S. The common assumption in machine learning is that the training sample S is generated by sampling points from the distribution $\mathcal{D}$ independently of each other. Formally,

The i.i.d. assumption: The examples in the training set are independently and identically distributed (i.i.d.) according to the distribution $\mathcal{D}$. That is, every $\mathbf{x}_i$ in S is freshly sampled according to $\mathcal{D}$ and then labeled according to the labeling function, f. We denote this assumption by $S \sim \mathcal{D}^m$ where m is the size of S, and D^m denotes the probability over m-tuples induced by applying D to pick each element of the tuple independently of the other members of the tuple.

Intuitively, the training set S is a window through which the learner gets partial information about the distribution $\mathcal{D}$ over the world and the labeling function, f. The larger the sample gets, the more likely it is to reflect more accurately the distribution and labeling used to generate it.

Since $L_{(\mathcal{D},f)}(h_S)$ depends on the training set, S, and that training set is picked by a random process, there is randomness in the choice of the predictor h_S and, consequently, in the loss $L_{(\mathcal{D},f)}(h_S)$. Formally, we say that it is a random variable. It is not realistic to expect that with full certainty S will suffice to direct the learner towards a good classifier (from the point of view of $\mathcal{D}$), there is always some probability that the sampled training data happens to be very non-representative of the underlying $\mathcal{D}$. If we go back to the papaya-tasting example, there is always some (small) chance that all the papayas we have happened to taste were not tasty, in spite of the fact that, say 70% of the papayas in our island are tasty. In such a case, $\mathrm{ERM}_{\mathcal{H}}(S)$ may be the constant function that labels every papaya as 'not tasty' (and has 70% error on the true distribution of papapyas in the island). We

will therefore address the *probability* to sample a training set for which $L_{(\mathcal{D},f)}(h_S)$ is not too large. Usually, we denote the probability of getting a non-representative sample by δ, and call $(1-\delta)$ the *confidence parameter* of our prediction.

On top of that, since we cannot guarantee perfect label prediction, we introduce another parameter for the quality of prediction, the *accuracy parameter*, commonly denoted by ϵ. We interpret the event $L_{(\mathcal{D},f)}(h_S) > \epsilon$ as a failure of the learner, while if $L_{(\mathcal{D},f)}(h_S) \leq \epsilon$ we view the output of the algorithm as an approximately correct predictor. Therefore (fixing some labeling function $f : \mathcal{X} \to \mathcal{Y}$), we are interested in upper bounding the probability to sample m-tuple of instances that will lead to failure of the learner. Formally, let $S|_{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ be the instances of the training set, we would like to upper bound

$$\mathcal{D}^m(\{S|_{\mathbf{x}} : L_{(\mathcal{D},f)}(h_S) > \epsilon\}) \,.$$

Let $\mathcal{H}_B$ be the set of "bad" hypotheses, that is $\mathcal{H}_B = \{h \in \mathcal{H} : L_{(\mathcal{D},f)}(h) > \epsilon\}$. As mentioned previously, the realizability assumption implies that $L_S(h_S) = 0$ with probability 1. This also implies that the event $L_{(\mathcal{D},f)}(h_S) > \epsilon$ can only happen if for some $h \in \mathcal{H}_B$ we have $L_S(h) = 0$. Therefore, the set $\{S : L_{(\mathcal{D},f)}(h_S) > \epsilon\}$ is a subset of $\{S : \exists h \in \mathcal{H}_B, L_S(h) = 0\}$, which in turns can be rewritten as $\cup_{h\in\mathcal{H}_B}\{S|_{\mathbf{x}} : L_S(h) = 0\}$. Therefore,

$$\mathcal{D}^m(\{S|_{\mathbf{x}} : L_{(\mathcal{D},f)}(h_S) > \epsilon\}) \;=\; \mathcal{D}^m(\cup_{h\in\mathcal{H}_B}\{S|_{\mathbf{x}} : L_S(h) = 0\}) \,. \tag{2.5}$$

Next, we upper bound the right-hand side of the above using the *union bound* - a basic property of probabilities.

Lemma 1 (Union bound)**.** *For any two sets A, B and a distribution $\mathcal{D}$ we have*

$$\mathcal{D}(A \cup B) \leq \mathcal{D}(A) + \mathcal{D}(B) \,.$$

Applying the union bound to the right-hand side of Eq. (**??**) yields

$$\mathcal{D}^m(\{S|_{\mathbf{x}} : L_{(\mathcal{D},f)}(h_S) > \epsilon\}) \;\leq\; \sum_{h\in\mathcal{H}_B} \mathcal{D}^m(\{S|_{\mathbf{x}} : L_S(h) = 0\}) \,. \tag{2.6}$$

Next, let us bound each summand of the right-hand side of the above. Fix some "bad" hypothesis $h \in \mathcal{H}_B$. The event $L_S(h) = 0$ is equivalent to the event $\forall i, h(\mathbf{x}_i) = f(\mathbf{x}_i)$. Since the examples in the training set are sampled i.i.d. we get that

$$\begin{aligned}\mathcal{D}^m(\{S|_{\mathbf{x}} : L_S(h) = 0\}) &= \mathcal{D}^m(\{S|_{\mathbf{x}} : \forall i, h(\mathbf{x}_i) = f(\mathbf{x}_i)\}) \\ &= \prod_{i=1}^{m} \mathcal{D}(\{\mathbf{x}_i : h(\mathbf{x}_i) = f(\mathbf{x}_i)\}) \,. \end{aligned} \tag{2.7}$$

For each individual sampling of an element of the training set we have,

$$\mathcal{D}(\{\mathbf{x}_i : h(\mathbf{x}_i) = y_i\}) = 1 - L_{(\mathcal{D},f)}(h) \leq 1 - \epsilon \ .$$

Combining the above with Eq. (**??**) and using the inequality $1-\epsilon \leq e^{-\epsilon}$ we obtain that for $h \in \mathcal{H}_B$,

$$\mathcal{D}^m(\{S|_{\mathbf{x}} : L_S(h) = 0\}) \leq (1-\epsilon)^m \leq e^{-\epsilon m} \ . \tag{2.8}$$

Combining the above with Eq. (**??**) we conclude that

$$\mathcal{D}^m(\{S|_{\mathbf{x}} : L_{(\mathcal{D},f)}(h_S) > \epsilon\}) \ \leq \ |\mathcal{H}_B| \, e^{-\epsilon m} \ \leq \ |\mathcal{H}| \, e^{-\epsilon \, m} \ .$$

Corollary 1. *Let $\mathcal{H}$ be a finite hypothesis class. Let $\delta \in (0,1)$ and $\epsilon > 0$ and let m be an integer that satisfies*

$$m \geq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon} \ .$$

Then, for any labeling function, f, and for any distribution, $\mathcal{D}$, for which the realizability assumption holds (that is, for some $h \in \mathcal{H}$, $L_{(\mathcal{D},f)}(h) = 0$), with probability of at least $1-\delta$ over the choice of an i.i.d. sample S of size m we have

$$L_{(\mathcal{D},f)}(h_S) \leq \epsilon \ .$$

A graphical illustration which explains how we used the union bound is given in Figure **??**.

The above corollary tells us that for a sufficiently large m, the $\mathrm{ERM}_{\mathcal{H}}$ rule over a finite hypothesis class will be *probably* (with confidence $1-\delta$) *approximately* (up to an error of ϵ) correct. In the next chapter we formally define the model of Probably Approximately Correct (PAC) learning.

Exercises

1. **Overfitting of polynomial matching:** Show that the rule given Eq. (**??**) can be described as a thresholded polynomial. That is, there exists a polynomial p such that $h_S(\mathbf{x}) = \mathbb{1}_{[p(x)\geq 0]}$.

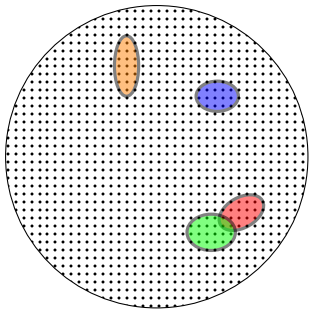

Figure 2.2: Each point in the large circle represents a possible m-tuple of instances. Each colored oval represents the set of 'bad' m-tuple of instances for some 'bad' predictor $h \in \mathcal{H}_B$, that is $\{S_{\mathbf{x}} : L_{\mathcal{D}}(h) > \epsilon \wedge L_S(h) = 0\}$. The ERM can potentially overfit whenever it gets a training set S which is bad for some $h \in \mathcal{H}_B$. Eq. (**??**) guarantees that for each individual $h \in \mathcal{H}_B$, at most $(1-\epsilon)^m$-fraction of the training sets will be bad. In particular, the larger m is, the smaller each of these colored ovals becomes. The union bound formalizes the fact that the area representing the training sets which are bad for some $h \in \mathcal{H}_B$ is at most the sum of the areas of the colored ovals. Therefore, it is bounded by $|\mathcal{H}|$ times the maximum size of a colored oval. Any sample S outside the colored ovals will not cause the ERM rule to overfit.

Chapter 3

A formal learning model

3.1 PAC learning

In the previous chapter we showed 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 1 (PAC learnability)**.** A hypothesis class $\mathcal{H}$ is PAC learnable if there exists 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 which satisfies the realizability assumption with respect to $\mathcal{H}$, when running the learning algorithm on $m_{\mathcal{H}}(\epsilon, \delta)$ 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 $1 - \delta$, $L_{(\mathcal{D},f)}(h) \leq \epsilon$.

The definition of Probably Approximately Correct learnability contains two approximation parameters. The accuracy parameter ϵ determines how far can the output classifier be from the optimal one (this corresponds to the "approximately correct"), and a confidence parameter δ indicating how likely is the classifier 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 sample S is randomly generated, there may always be a small chance that it will happen to be non-informative or highly biased (for example, there is always some chance that S 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 $\mathcal{D}$, due to being 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 $\mathcal{H}$.

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

Every finite hypothesis class is PAC learnable

There are infinite classes that are learnable as well. 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 could 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 sub-section, 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 9pm 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 **??** below.

3.2.1 Releasing the realizability assumption – Agnostic PAC learning

A more realistic model for data-generating distribution. Recall that the realizability assumption requires that there exists $h^\star \in \mathcal{H}$ such that $\mathbb{P}_{\mathbf{x}\in\mathcal{X}}[h^\star(\mathbf{x}) = f(\mathbf{x})] = 1$. 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 a 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 $\mathcal{D}_\mathcal{X}$ over unlabeled domain points (sometimes called the *marginal distribution*) and a *conditional* probability over labels for each domain point, $\mathcal{D}((\mathbf{x}, y)|\mathbf{x})$. In the papaya example, $\mathcal{D}_\mathcal{X}$ 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-hardness represented by $\mathbf{x}$ 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 with respect to such $\mathcal{D}$. For a probability distribution, $\mathcal{D}$, over $\mathcal{X} \times \mathcal{Y}$, one can measure how likely is h to make an error when labeled points are randomly drawn according to $\mathcal{D}$. We redefine the true

error of a prediction rule h to be

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

We would like to find a predictor, h, for which that error will be minimized. However, the learner does not know the data generating 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,

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

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

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

It is easy to verify (see Exercise **??**) 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, non-realizable, 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 which 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 2 (agnostic PAC learnability)**.** A hypothesis class $\mathcal{H}$ is agnostic PAC learnable if there exists 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 $m_{\mathcal{H}}(\epsilon, \delta)$ 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 term 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 could 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, etc.). A learning algorithm for such a task, will have access to examples of correctly classified documents, and, based on 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* which 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 would 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 based on 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 $(\mathbf{x}, y)$ 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,

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

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 non-negative 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 **??**, 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 above examples of classification and regression tasks are as follows:

- $0-1$ **loss:** Here, our random variable z ranges over the set of pairs $\mathcal{X} \times \{0, 1\}$ 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 Eq. (**??**) and Eq. (**??**) coincides.

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

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

  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 (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 exists a function $m_{\mathcal{H}} : \mathbb{R}_+ \times (0, 1) \to \mathbb{N}$ 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 $m_{\mathcal{H}}(\epsilon, \delta)$

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 1 (A note about measurability*). In the above 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 pre-image 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 $\{(\mathbf{x}, h(\mathbf{x})) : \mathbf{x} \in \mathcal{X}\}$ is in the σ-algebra.

Remark 2 (Proper vs. Improper learning*). In the above 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 *improper learning*, while proper learning is when the algorithm must output a hypothesis from $\mathcal{H}$.

Exercises

1. Show 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, $L_{\mathcal{D}}(f_{\mathcal{D}}) \leq L_{\mathcal{D}}(g)$.