5. Learning Problems

We now aim to combine the concepts which we have thus far encountered, and construct a formal definition for a learning problem. We will begin by defining the necessary inputs, and the desired outputs, before working towards the fundamental definition of PAC learnability, which will allow us to have some notion of the possibility of learning a certain problem.

5.1 Introduction

We begin the chapter by laying out an example which we will reference and build around throughout.

Suppose that you are a loaner, aiming to determine if an arbitrary loanee will or will not default on the loan you give them. We phrase the problem deliberately in this manner, to emphasize the simplistic scenario in which we are working – the value of the loan is predetermined, and the goal is to provide a binary classification of loanees.

What information will be useful in this task? How will we determine if our solution is adequate? How will be provide a useful notation to describe the problem?

5.2 A formal learning model

5.2.1 Learning input

The learner of a statistical learning problem has access to the following,

- Domain set: an arbitrary set which we typically label by $\mathcal{X}$. This set is nothing more than the objects to which we aim to assign labels. In the example outlined in the previous section, this set $\mathcal{X}$ is the set of potential loanees.
- Label set: A set which we typically label by $\mathcal{Y}$. This is the set of possible labels, e.g., { default , no default }. Of course, it is more common to denote the possible labels simply by integers, and in our binary classification example, we will take $\mathcal{Y} = \{\pm 1\}$.
- Training data: a finite sequence of pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, i.e., a sequence of labelled objects. In our case, this will be a sequence of past loaners together with information relating to their repayment. We will tend to denote the training data by $\mathcal{S} = ((x_1, y_1), ..., (x_m, y_m))$, where m is the size of the training data.

> Remark. It is important to note that the training data is described as a sequence rather than a set. This is a rather nuanced point, and not one which we will dedicate much thought to. The reason for such pedanticity is that there exist learning algorithms which are dependent on the order of the training points, and there is no necessity for $\mathcal{S}$ to be duplicate free.
>
> It is however still common to refer to $\mathcal{S}$ as the training set.

5.2.2 Learning output

- Prediction rule: the only output from a statistical learning problem is a function $h : \mathcal{X} \to \mathcal{Y}$. We also refer to the function as a hypothesis, and is a rule which the learner can use to label new elements of the domain space.

If we are considering an algorithm $\mathcal{A}$, then we will denote by $\mathcal{A}(\mathcal{S})$ or $h_{\mathcal{S}}$ the output hypothesis of the algorithm.

5.2.3 A data generating model

It is important to consider how exactly the training data is generated, as this is integral to how we consider the learning to occur. We assume that there is some probability distribution $\mathcal{D}$ over $\mathcal{X}$ which, importantly, the learner does not know about. For the time being, we also make the assumption that there is some 'correct' labelling function $f : \mathcal{X} \to \mathcal{Y} : x_i \mapsto y_i$, which maps the object x_i to it's correct label in every case. Explicitly,

$$f(x_i) = y_i \quad \forall i.$$

In every learning scenario, it is exactly this function f which the learner aims to obtain. In our example, we can suppose that we have past data, $\mathcal{S}$ which provides a source of truth in the way that each of the training examples is an individual of whom we *know* the loaning outcome.

5.2.4 Measure of success

The final element of our formal learning model is the means to quantify the success of our learning, or more strictly our hypothesis, h. In our classification problem, the error of the hypothesis h is defined to be the probability that for a given data point $x \in \mathcal{X}$, the predicted classification is not the same as the 'correct' classification. Notatively,

$$L_{\mathcal{D},f}(h) := \mathbb{P}_{x\sim\mathcal{D}}\left[h(x) \neq f(x)\right]$$

> Remark. The notation of $L_{\mathcal{D},f}$ is well chosen since it makes clear the dependence of the error of the classifier on the underlying distribution $\mathcal{D}$ and correct classification function f, although if the context is clear we will omit this for brevity.
>
> It is also common to refer to the probability $L_{\mathcal{D},f}(h)$ as the 'risk', and we will often refer to it as such.

5.3 Empirical Risk Minimisation

We now have a good idea about the inputs, outputs and measures of success of a formal learning problem. The natural next step is to explore the strategy by which we can solve such problems.

It maybe initially seems obvious that we would like to minimise the risk, explicitly reducing the probability of incorrect classification, implicitly bringing our hypothesis output h closer to the true classifier f. Although this is the right idea, we are the learner is limited by their knowledge of the problem. In particular, the learning knows nothing of the distribution $\mathcal{D}$, nothing of the true classifier f, and hence nothing of the risk $L_{\mathcal{D},f}$. As a result, we must construct a different quantity which we would like to minimise.

> Definition 5.1. Given a training sequence $\mathcal{S} = ((x_1, y_1), ..., (x_m, y_m))$, the *empirical risk* with respect to $\mathcal{S}$ is defined and denoted by,
>
> $$L_{\mathcal{S}}(h) := \frac{|\{i \in \{1, ..., m\} : h(x_i) \neq f(x_i)\}|}{m}$$

This definition is basic in spite of it's verbosity. The numerator is expressing the process of counting misclassifications on the training set, and we are dividing by the size of the training set to ensure that $L_{\mathcal{S}}$ is bounded between 0 and 1.

The empirical risk is a good quantity to consider in learning problems since it is related to the training data available to us, i.e., "the snapshot of the world available to the learner"[1]. So, our strategy of learning comes down to minimising the empirical risk – finding a hypothesis h, which agrees well with f on the training set. Learning by this strategy is called *empirical risk minimisation*, referred to by ERM.

5.3.1 Overfitting

We now explore a pitfall into which many learning algorithms fall, and one which we will illustrate using the framework so far developed in this chapter, working again with the example of loaning.

> Example 5.2. In order to illustrate overfitting in the ERM paradigm, we suppose that the underlying distribution $\mathcal{D}$ on loan applicants $\mathcal{X}$ is uniform, that is, the correct classification of the feature space is half-half between default (1) and non-default (-1).
>
> It may seem initially that the process of finding an ERM hypothesis is challenging, but this is not actually the case. We can find a scenario agnostic ERM hypothesis,
>
> $$h_{\mathcal{S}}(x) = \begin{cases} y_i & \text{if } \exists i \in \{1, ..., m\} \text{ s.t. } x_i = x \\ 1 & \text{otherwise} \end{cases}$$
>
> What exactly is this hypothesis doing? If the selected data point $x \in \mathcal{X}$ can be found in the training set $\mathcal{S}$, then the hypothesis returns the label which is assigned to x in the training set. Otherwise, the hypothesis returns 1, i.e., predicts that the loanee will be good for their loan and won't default. This hypothesis $h_{\mathcal{S}}$ is ERM since is agrees exactly with the labels provided in the dataset, and hence has $L_{\mathcal{S}}(h_{\mathcal{S}}) = 0$ – there is zero probability of the hypothesis incorrectly classifying an instance of the training set. However, if the hypothesis returns -1 on only a finite number of instances, then $L_{\mathcal{D},f}(h_{\mathcal{S}}) = 1/2$.
>
> We have found a hypothesis which according to our strategy provides perfect learning, but in reality, performs very badly. That is, the hypothesis cannot generalise. It is 'overfitted' to the training set.

The previous example outlines a clear failure of the ERM paradigm, but we shouldn't let this provide obstruction to it's use in total; instead we will find ways to reprimand it's issues.

5.4 Limiting the hypothesis space

One way to address the problem of overfitting in ERM is to limit the possible hypotheses which we will output from learning. In particular, the learner should explicitly define a set of possible predictors. We call this set the *hypothesis class*, and typically label it by $\mathcal{H}$, which contains hypotheses of the form we have seen previously; functions $h : \mathcal{X} \to \mathcal{Y}$.

In order to distinguish the ERM learning rule from the hypothesis restricted analogue, we use $\text{ERM}_{\mathcal{H}}$. Furthermore, we can understand the $\text{ERM}_{\mathcal{H}}$ learning rule as minimisation of the empirical risk over the set of hypotheses. That is,

$$\text{ERM}_{\mathcal{H}}(\mathcal{S}) \in \arg\min_{h \in \mathcal{H}} \{L_{\mathcal{S}}(h)\}.$$

> Notation. For those who haven't seen it before, the notation $\arg\min_{x \in \mathbb{R}}\{f(x)\}$ denotes the value (or set of values) of x in the set $\mathbb{R}$ which minimises the function f. So, in our case, we are looking for the hypothesis h in the hypothesis set $\mathcal{H}$ which minimises the empirical risk function $L_{\mathcal{S}}$.

It is natural to ask how exactly a learner should go about limiting/defining a hypothesis set. The restriction to a certain set of hypotheses should be based on some prior knowledge of the problem, the process of choosing this hypothesis set is referred to as *inductive bias*. It should also be fully considered whether or not the restricted hypothesis set is a good one; will the restriction actually prevent overfitting of the learner? We will explore some examples in this direction later on.

EXAMPLE 5.3. A natural way to limit the potential hypotheses is to limit the size of the hypothesis set. That is, we may impose a finiteness condition on $\mathcal{H}$. We could let $\mathcal{H}$ be the set of all predictors which can be implemented via python programs of at most 10^{32} characters. There are a number of other common examples.

5.5 PAC Learnability

DEFINITION 5.4. A hypothesis class $\mathcal{H}$ is said to be *Probably Approximately Correct (PAC) learnable* if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm $\mathcal{A}$ that satisfy the following condition.

For every $\epsilon, \delta \in (0,1)$, for every probability distribution $\mathcal{D}$ over $\mathcal{X}$, and for every labelling function $f : \mathcal{X} \to \mathcal{Y}$, if the assumption of realisability holds, then when the algorithm takes as input $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. data points generated by $\mathcal{D}$ and labelled by f, the hypothesis returned by $\mathcal{A}$ is, with probability at least $1 - \delta$, such that,

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

REMARK. If you haven't taken any higher level mathematics courses, it can be said, almost with certainty, that this is the most intimidating definition you have seen. The majority of this intimidation is due to the necessity to speak very carefully in definitions, not to assume or state things which may not necessarily be true. The idea hidden in the definition is however fairly intuitive.

NEEDS EXPLANATION.

PROPOSITION 5.5. Every finite hypothesis class $\mathcal{H}$ is PAC learnable.