Contents

Chapter 1

Introduction

Some kinds of learning include;

1.1 Supervised learning

Given a dataset of pair

$$\mathcal{D}_n = \left\{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\right\} \tag{1.1}$$

we wish to establish a relationship between $x^{(i)}$ and $y^{(i)}$. Typically, $x^{(i)} \in \mathbb{R}^d$ is a representation of input, called **feature representation**. Based on the format of the output we can have different types of supervised learning:

Classification when the set of possible values of $y^{(i)}$ is discrete (small finite set). If there two possible values then the classification problem is *binary* otherwise, it is called *multi-class*.

Regression when the set of possible values of $y^{(i)}$ is continuous (or a large finite set). That is, $y^{(i)} \in \mathbb{R}^k$.

1.2 Unsupervised learning

Given a dataset we wish to find some patterns or structures in it. There are several types of unsupervised learning:

Density estimation The data is i.i.d from some distribution $p_X(x)$. The goal is to predict the probability $p_X\left(x^{(n+1)}\right)$.

Clustering the goal is to find a partitioning of the sample data that groups together samples that are similiar. Clustering is sometimes used in density estimation.

Dimensionality reduction the goal is to re-represent the same data in $\mathbb{R}^l$ where $l < d$.

1.3 Reinforcement learning

The goal is to learn a mapping from input values to output values without a direct supervision signal. There is no training set specified *a priori*. Instead, the learning problem is framed as an agent interacting with an environment. Looking at input as our states and output as a transition between states, we can assign a reward value $r^{(i,j)}$ to each such transition. We aim to find a policy π that maximizes the long-term sum or average od rewards.

Chapter 2

Statistical Learning

A statistical learner needs to know the *domain set*, $\mathcal{X}$, *label set*, $\mathcal{Y}$, and a training data set (more like a sequence) $S \subset \mathcal{X} \times \mathcal{Y}$. Given these, learner outputs a *predictor* $h : \mathcal{X} \to \mathcal{Y}$ which is also called *hypothesis* or *classifier*. We can assume that $\mathcal{D} = \mathbb{P}_{\mathcal{X}}$ is the distribution on $\mathcal{X}$ and there exists a correct function f that for each sampled x output the corresponding label $y = f(x)$. Then the *error* of h is defined as

$$L_{\mathcal{D},f}(h) = \mathbb{P}(h(x) \neq f(x))$$

Since we know neither f nor $\mathcal{D}$ we can not find the exact error. To approximate this error, we can use the *empirical error*.

$$L_S(h) = \frac{|\{i \mid h(x_i) \neq y_i\}|}{|S|}$$

Since S is a representation of the real distribution it makes sense to minimize $L_S(h)$ and expect that $L_{\mathcal{D},f}(h)$ is minimized as well. This is called **empirical risk minimization** or ERM for short. *Overfitting* is one drawback of ERM which arises when S is not fully representitive of $\mathcal{D}$. In that case, predictor though working well on the training data, fails to generalize and mislabels the new data.

One way to avoid overfitting is to restrict possible hypotheses to a class of hypothese $\mathcal{H}$. Then

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

This way, we increase the bias toward $\mathcal{H}$ and possibly increasing the true error.

2.1 Finite hypothesis class

Suppose $\mathcal{D}$ is finite and assume that there exists a $h^* \in \mathcal{H}$ such that

$$L_{\mathcal{D},f}(h^*) = 0$$

This is called the *realizability assumption*. Furthermore, we can assume that training data are selected independent of each other.

We often assign a probabilityδ to getting a non-representitive training data. $1 - \delta$ is called the *confidence parameter*. We then assign an *accuracy parameter* ϵ where $L_{\mathcal{D},f}(h_S) > \epsilon$ is a failure. We wish the find an upperbound for the probability of getting a training data that results in a failure.

$$\mathbb{P}(S \text{ s.t.} L_{\mathcal{D},f}(h_S) > \epsilon)$$

Let $\mathcal{H}_B$ be the set of bad hypotheses

$$\mathcal{H}_B = \{h \mid L_{\mathcal{D},f}(h) > \epsilon\}$$

and M the set of misleading samples

$$M = \{S \mid \exists h \in \mathcal{H}_B,\ L_S(h) = 0\}$$

Chapter 3

Probably Approximately Correct Learning

Definition: 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 alogrithm such that:

- For every $\epsilon, \delta \in]0,1[$, distribution $\mathcal{D}$ over $\mathcal{X}$, and labeling function $f : \mathcal{X} \to \{0,1\}$
- If the realizable assumption hold with respect to $\mathcal{H}, \mathcal{D}, f$
- Then, when running the algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ of i.i.d. samples generated by $\mathcal{D}$ and labeled by f, the algorithm returns a hypothesis h such that

$$\mathbb{P}(L_{\mathcal{D},f}(h) \leq \epsilon) \geq 1 - \delta$$

Remark 1. The minimal function $m_{\mathcal{H}}$ determines the sample complexity of learning $\mathcal{H}$.

Corollary 3.1. *Every finite hypothesis class is PAC learnable with sample complexity*

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

Let $\mathcal{J}$ be the joint distribution over $\mathcal{X} \times \mathcal{Y}$. Note that, $\mathcal{D}$ is the marginal distribution of $\mathcal{J}$. Then we can revise the definition for the true error

$$L_{\mathcal{J}}(h) = \mathbb{P}_{(x,y)\sim\mathcal{J}}(h(x) \neq y)$$

Then given $\mathcal{J}$ the best label prediction function is

$$f_{\mathcal{J}}(x) = \begin{cases} 1 & \text{if } \mathbb{P}(y = 1 \mid x) \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

that is, there is no other classifier g with $L_{\mathcal{J}}(g) < L_{\mathcal{J}}(f_{\mathcal{J}})$

Definition: A hypothesis $\mathcal{H}$ is **agnostic PAC learnable** if there exist a function $m_{\mathcal{H}} :]0,1[^2 \to \mathbb{N}$ and a learning alogrithm such that:

- For every $\epsilon, \delta \in]0,1[$, distribution $\mathcal{J}$ over $\mathcal{X} \times \mathcal{Y}$
- Then, when running the algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ of i.i.d. samples generated by $\mathcal{D}$ and labeled by f, the algorithm returns a hypothesis h such that

$$\mathbb{P}\Big(L_{\mathcal{J}}(h) \leq \min_{h'} L_{\mathcal{J}}(h') + \epsilon\Big) \geq 1 - \delta$$

3.1 Generalized loss functions

Given any set $\mathcal{H}$ and some domain Z, let l be any function from $\mathcal{H} \times Z$ to $\mathbb{R}_+$. We call such functions *loss functions*. We then define the risk function to be

$$L_{\mathcal{Z}}(h) = \mathbb{E}_{\mathcal{Z}}[l(h, z)]$$

where $h \in \mathcal{H}$, and $\mathcal{Z}$ is the distribution on Z. Similarly, the empirical risk over a given sample $S \in Z^m$ is

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} l(h, z_i)$$

Then revising the agnostic PAC learnability definition for general loss function gives

Definition: A hypothesis $\mathcal{H}$ is **agnostic PAC learnable** with respect to Z and a loss function $l : \mathcal{H} \times Z \to \mathbb{R}_+$, if there exist a function $m_{\mathcal{H}} :]0,1[^2 \to \mathbb{N}$ and a learning alogrithm such that:

- For every $\epsilon, \delta \in]0,1[$, distribution $\mathcal{Z}$ over Z
- Then, when running the algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ of i.i.d. samples generated by $\mathcal{Z}$ and labeled by f, the algorithm returns a hypothesis h such that

$$\mathbb{P}\Big(L_{\mathcal{Z}}(h) \leq \min_{h'} L_{\mathcal{Z}}(h') + \epsilon\Big) \geq 1 - \delta$$

Remark 2. 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 $\mathcal{H}' \times Z$. In this cases, we may allow the algorithm to return a hypothese $h' \in \mathcal{H}'$ as long as it satisfies the requirement

$$\mathbb{P}\left(L_{\mathcal{Z}}(h') \leq \min_{h \in \mathcal{H}} L_{\mathcal{Z}}(h) + \epsilon\right) \geq 1 - \delta$$

This is called *representation independent* learning, or *improper learning.*