UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques
i Informàtica

BACHELOR IN MATHEMATICS

Final degree Thesis

LEARNING THEORY AND OUT OF DISTRIBUTION DETECTION

Author: Axel Champredon

Supervisors: Dr. Josep Vives & Dr. Santiago Segui
From: Mathematics and Computer Science Department

Barcelona, June 9, 2024

Contents

Abstract

Nowadays, Machines are starting to have a really important relevance in automation tasks. Learning could be considered as one of the hardest tasks we can encounter. The goal of this work is to introduce the theoretical foundations of this topic.

After a short introduction on general learning and machine learning in chapter 1, we will introduce the fundamental concepts of *Learning Theory* in chapter 2, we will find what learning formally means and under which conditions a scenario can be learnable.

In the last chapter we will introduce a pretty recent topic: *Out of Distribution detection*. A theory that appeared for the first time in 2017 and tries to formalize whether or not it would be possible for a machine to detect if we are trying to make predictions on data which it hasn't been trained for. Again, we will try to find conditions under which a machine could learn this skill.

Resum

Avui en dia, les màquines comencen a tenir una rellevància important en tasques d'automatització. Una de les tasques que es podria considerar de les més difícils és aprendre. L'objectiu d'aquest treball serà introduir els fonaments teòrics d'aquest tema.

Després d'una breu introducció sobre l'aprenentatge i l'aprenentatge automàtic per les màquines, introduirem els conceptes fonamentals de la *Teoria de l'aprenentatge*, definirem formalment el significat de l'aprenentatge i posarem condicions sota les quals un escenari pot ser après per una màquina.

A l'últim capítol introduirem un tema d'actualitat: la *Detecció de Fora de distribució*. Una teoria que va néixer l'any 2017 i que intenta formalitzar si seria possible que una màquina detectés si estem intentant fer prediccions sobre dades per les quals no ha sigut entrenada. Altre cop, intentarem trobar condicions sota les quals una màquina podria aprendre aquesta habilitat.

2020 Mathematics Subject Classification. 11G05, 11G10, 14G10

Acknowledgment

I will start by thanking my tutors for this work. Thank you to Santiago Seguí for providing me with a wonderful topic that made me want to continue studying and maybe work in the research area. Also thank you to Josep Vives for revising my work at the last minute and finding every single small inaccuracy.

I also want to thank my parents Olivier and Clarisse, as well as my brother Gabriel, for the unconditional support during all the process.

Also thank you to Build38 team for the support in the hardest moments, especially to Edu for staying until late in the office to make sure I don't lose concentration.

Chapter 1

Introduction to Machine Learning

1.1 What is learning?

The best way to understand what is learning is to see some examples we can easily encounter in the nature. This will allow us to understand the goal of Machine Learning and the way to automate the learning process.

When a rat encounters food (or even objects sometimes), they first smell the item to see if they recognize it, if it is recognized and remembered as eatable, no danger detected, they will eat it. If the item is not recognized they will taste a very small amount, the result of the experiment will depend on the flavour and the physiological effect of the food. If the food produces ill it will be associated to illness and will be remembered as so.

The rat is indeed generating his knowledge based on his own experience. In other words, there is a learning mechanism. Let's see if we can apply this mechanism to a real life case scenario.

Let's say we want to create a machine that detects spam emails. Using the rat's method, we can use a database of spam emails, when an email is received we can compare it to the saved spam emails and see if there is a match, if not, the email will be considered as not spam. If it is then reported as spam by the user it will be added to the database.

The limitation here is that the probability of getting the same email is really low, which means that not a lot of spam emails will be detected. We have to find a more accurate way to find spam emails.

We could try to find patterns in the emails, with the risk of finding these patterns in non spam emails. This is known as superstitious learning and is in fact not accurate in this scenario.

Learning theory is seemingly not trivial or easy to achieve. We will see now in what cases we need more advanced tools to obtain algorithms capable of learning

in more complicated situations.

1.2 When do we need Machine Learning?

Sometimes it is really easy to program a computer to solve specific problems if we already know how to solve them. For example we can create a program that finds the title of a music by listening to it, it can be done by analyzing the spectrum of frequencies of the music, for example the Shazam algorithm distills samples of a song into fingerprints, and matches these fingerprints against fingerprints from known songs, taking into account their timing relative to each other within a song. And it is really effective.

The problem rises when we do not know how to solve a problem or it is too hard to program or to compute. An easy example is chess, given a specific position we humanly cannot tell what the best move is in every scenario. And it is really hard to calculate even with a computer, simply because the number of games that can be played from a given position is too high, the possible outcomes increases exponentially based on the depth of the calculations (the number of moves we calculate ahead).

Clearly this could be programmed but couldn't be executed by any computer, the number of possible games is estimated at 10^{120} according to the Shannon number.

Another limitation of programming an algorithm by hand is the rigidity, once the program has been written and installed it cannot be changed. Sometimes programs need to adapt, either because new data has been found or because data can change over time. The advantage of Machine Learning is that it adapts to its input data.

Using the previous example of chess, DeepMind developed AlphaZero in 2017, a computer program that learned how to play chess by itself, by playing against himself and learning from its own games, without having access to databases or previous knowledge. After only 4 hours of training it reached a better level than Stockfish, the best chess program at that moment.

AlphaZero uses a deep neural network to achieve such a result, but there is many types of learning. We will now see a concise overview of some of the types of learning we can find a Machine Learning.

1.3 Types of learning

As expected, Machine Learning is a very wide domain, as a consequence it has branches to several subfields dealing with different learning scenarios. Here is a

brief explanation of some of these fields.

Supervised and unsupervised learning. The difference between these two branches lies in the labeling of the input data. Supervised learning consists in training an algorithm with the input data being labelled, for example in the spam email detection mentioned before, the training data would consist of a database of emails and if they are spam or not.

Unsupervised learning would rather try to find unusual behaviours in the body of each email because they would be no label attached to it. More abstractly, there is no difference between training and testing 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 objects is a typical example of such a task.

In this work we will mainly focus on supervised learning as it is easier to evaluate the effectiveness of a classifier.

Active and Passive learners. These types of learners can be distinguished by the role of the learner. An active learner will interact with the environment at learning time. Using the example of the spam emails, an active learner could ask the user if a chosen email is spam or not, in general it would be a passive learner, it would only wait for the emails to come to him.

Help of a teacher. The name is quite self explanatory, in a real case scenario, a scientist has to face the entire environment an choose the useful data by himself, unlike a child learning with a teacher, in that case the teacher can choose to give specific data to easier the learning process to the learner.

We can also use an adversarial "teacher", to test the effectiveness of our learner in the worst case scenario: if you manage to learn against an adversarial teacher you are guaranteed to succeed interacting with any odd teacher. For example if the spammer tries to mislead the spam detector.

Online and batch learning. Sometimes the learner has to respond online, throughout the learning process, and will be effective after having processed a large amount of data. For example a stockbroker has to make daily decisions, and at the beginning his decisions will probably be unsuccessfully, but he may become an expert over time. In most of the cases, the learner has large amounts of data he can use to learn before having to take any decision.

1.4 Decision tree classifier

In order to get a Machine learning algorithm we need data, all that we can get. Once we have enough data we divide it into training and testing data, usually 70% of the total data is used as training data. Then we use an algorithm called the

learner to output a predictor using the training data, we finally test and evaluate this predictor using the testing data. The predictor can be either a classifier or a regressor depending on the labels domain. Here we will briefly see the Decision tree, a simple example of Machine Learning algorithm.

Definition 1.1. A **tree** is a data structure in which each node is connected to a certain amount of children (depending on the type of tree) but can be connected to only one parent. The only node that has no parent is the **root**. A node that has no children is called a **leaf**.

Definition 1.2. A **decision tree** is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

Example 1.3. Here we have a decision tree that tries to guess what animal are we observing given some details about the animal.

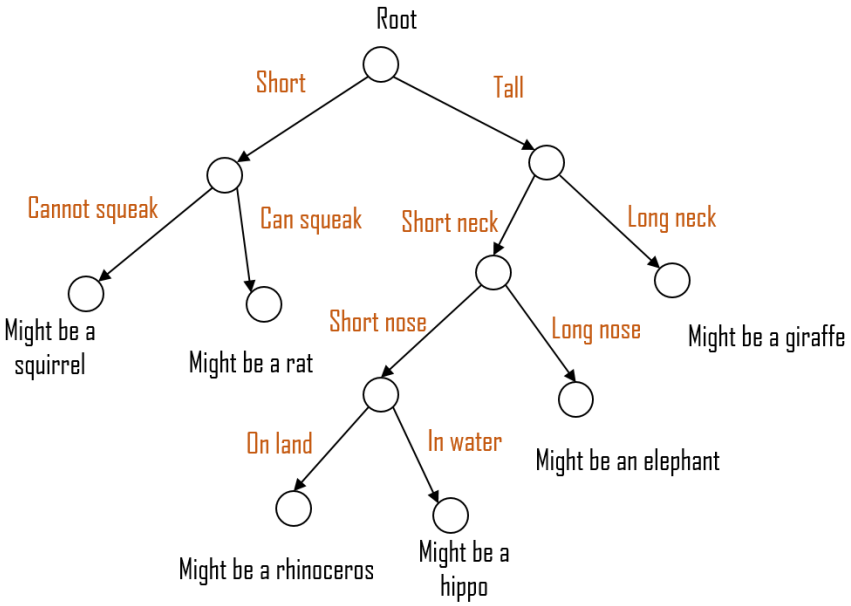

Figure 1.1: Decision tree classifier

Remark 1.4. As we can observe each node of the decision tree contains a condition, if a given data satisfies the condition, the node will bring that data to a certain child, if not it will bring it to the other child node.

1.4.1 Training

We will now see how such algorithm can be learned, or trained in this case scenario.

Many ways can be employed to get a trained decision tree, here we will see one of the easiest ways, using information gain based on the entropy of each node.

Definition 1.5. If we have a dataset X with n classes and $p_1, ..., p_n$ the proportion of every class in X, we define the **entropy** of a dataset as

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Remark 1.6. Notice that if we have only 2 classes, the entropy is $H(X) = 0$ if X contains data from only one class, and it is $H(X) = 1$ if half of the dataset is from one class and the other half of the dataset is from the other class.

Definition 1.7. If we have a dataset X with n classes and a condition C that divides X in the subsets $X_1, ..., X_m$, we define the **information gain** of the condition as:

$$IG_X(C) = H(X) - \sum_{i=1}^{m} P_i H(X_i),$$

where P_i is the number of data items in X_i divided by the number of data items in X.

Now the training process is easy to define, if we have a training dataset X, each data item with n features, we choose a given amount of pairs composed by a random conditions and a random feature for the root node, and we keep the one that has the highest information gain. Each subset $X_1, ..., X_m$ will be inherited by the children nodes and the same process will be applied recursively. The process stops if a node contains only one class of data items, in other words the entropy of the dataset inherited to the node is 0.

Let's see a quick example.

Example 1.8. Let's say we are in the unit square $[0, 1]^2$, if a point is placed in the $[0, \frac{1}{2}]^2$ then the point is red, otherwise it is blue. It is important to note that the learner doesn't know this rule, that's what we want it to learn using the training data. In other words, we want to train a decision tree so it can predict the color of a point based on its coordinates.

We can see that in this case, we only have 2 features, the x and y coordinates of each point.

Let's say we have a set of 5 points as training data; $p_1 = (0,15; 0,31), p_2 = (0,80; 0,99), p_3 = (0,73; 0,53), p_4 = (0,88; 0,85), p_5 = (0,33; 0,70)$. Notice that in our case only p_1 is red.

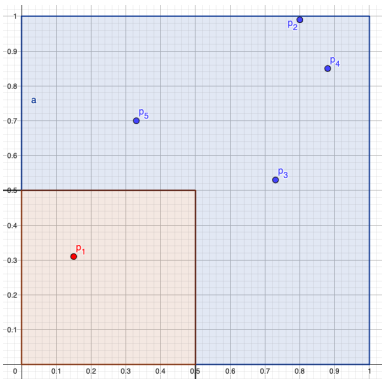

Figure 1.2: Points distribution

Let's train our model:

- **Step 1**: We choose a random feature and a random condition. Let's say we choose the x coordinates and $x < 0,4$ as condition.
- **Step 2**: We observe that p_1 and p_5 will be in the same class, and p_2, p_3 and p_4 will be in the other class. We can calculate the **entropy** of the parent node, and the children nodes.

  $H(parent) = -\frac{1}{5} \cdot log_2(\frac{1}{5}) - \frac{4}{5} \cdot log_2(\frac{4}{5}) \approx 0,72192$

  $H(right\_child) = -\frac{1}{2} \cdot log_2(\frac{1}{2}) - \frac{1}{2} \cdot log_2(\frac{1}{2}) = 1$

  $H(left\_child) = -0 \cdot log_2(0) - 1 \cdot log_2(1) = 0$ (assuming $0 \cdot log_2(0) = 0$)

  We can now calculate the **information gain**

  $IG_{\{p_1,p_2,p_3,p_4,p_5\}}(x < 0,4) = 0,72192 - \frac{2}{5} \cdot 1 - \frac{3}{5} \cdot 0 = 0.32192$
- **Step 3**: We can choose a given amount of other pairs of feature/condition and calculate the information gain of each of them, and keep the one with highest **information gain**.
- **Step 4**: The process creates 2 subsets of point, each subset is inherited by a child, the same process would be applied until the entropy of a node is 0, it would be the case of the left child with the condition we have chosen.

Remark 1.9. As we can see a decision tree can be different from one execution to the other as it requires randomness, the predictions can vary because of that. To

avoid this we can create a **random forest**, a set of a given amount of trees that will be trained separately, the prediction of the forest will be the average or the median (or others) of the predictions made by each tree. Other similar models can be used, they are known as **strong learners**, but we won't cover this topic in this work.

Remark 1.10. In this case we have used the **information gain** to evaluate the efficiency of a threshold, but other methods can be used like the **gini function**.

1.4.2 Evaluation measures

As we discussed previously, we usually keep some data to evaluate the model, in this case the classifier. Here are some tools used for evaluating a model that has to classify over only two classes.

Definition 1.11. If we have only two classes, a positive and a negative one, we define:

- a **false positive** is data that has been classified as positive but is on fact negative.
- a **false negative** is data that has been classified as negative but is in fact positive.
- a **true positive** is data that as been successfully classified a positive.
- a **true negative** is data that has been successfully classified as negative.

This being established, we can define the evalutaion measures.

Definition 1.12. Let **FN** be the false positives, **FP** the false positives, **TN** the true negatives and **TP** the true negatives. If **N** is the total size of the sample:

- The **Accuracy** is: $\frac{TP+TN}{N}$
- The **Precision** is: $\frac{TP}{TP+FP}$
- The **Recall** is: $\frac{TP}{TP+FN}$
- The **Specificity** is: $\frac{TN}{TN+FP}$
- The **F1-Score** is: $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$

Remark 1.13. We will see in the next chapters that researchers combine these evaluation measures to estimate the performance of the learners, the out of distribution detection in our case.

Chapter 2

Learning Theory

2.1 The learning framework

We have seen what learning is all about, and we have introduced an example of machine learning algorithm. Let's now deep into the mathematical concepts of learning.

In order to introduce the learning theory we first have to define rigorously the key statistical concepts of learning.

In this chapter we will be using knowledge from the books [2], [3] and [7].

2.1.1 Previous definitions

Definition 2.1.

- *The **domain set** is an arbitrary set $\mathcal{X}$ that we may wish to label. It is composed by vectors of **features** also known as **instances**. In the decision tree example seen in the previous chapter we would have $\mathcal{X} = [0,1]^2$*
- *The **label set** is the set $\mathcal{Y}$ of possible labels. For the example mentioned earlier it would be $\mathcal{Y} = \{red, blue\}$.*
- *The training set is a set $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\} \subseteq \mathcal{X} \times \mathcal{Y}$ of finite pairs of labeled domain points. This is the set that the learner has access to during the learning phase. In our example it would be the set $S = \{p_1, p_2, p_3, p_4, p_5\}$.*
- *The **learner's output** also known as the **predictor** will be the algorithm that will result from the learning phase. In our example it would be the decision tree itself as it's the algorithm that we will use to make our predictions on unseen instances.*

Something important to point at is how will we know that our algorithm is learning in the right way. That is when we introduce the measure of success. In

the decision tree example we used the entropy function to evaluate how well a threshold was dividing our inherited dataset.

More generally we have to use a function that returns the probability of our predictor not predicting correctly on a random instance. Here we define formally this concept.

Definition 2.2. *Given a domain subset $A \subset \mathcal{X}$ and the indicator function $\mathbb{1}_A$, we define the distribution $\mathcal{D}(A) = \mathbb{P}_{x\sim\mathcal{D}}(\mathbb{1}_A(x))$. In other words it will assign a number which determines how likely it is to observe a point $x \in A$.*

*Now given a prediction $h : \mathcal{X} \to \mathcal{Y}$, we define the **error**, or the **risk**, as follows:*

$$L_{(\mathcal{D},f)} \overset{def}{=} \mathbb{P}_{x\sim\mathcal{D}}[h(x) \neq f(x)] \overset{def}{=} \mathcal{D}(\{x \in \mathcal{X} : h(x) \neq f(x)\})$$

*As we can observe, this function returns the probability of choosing a random point x such as $h(x) \neq f(x)$. Let us highlight that L stands for **loss**.*

It is now easy to know if our learning process is effective or not, as we mentioned in the decision tree example, we can now choose between many possibilities based on which one return the minimal error, this is known as **Risk Minimization**.

But a clear problem arises when defining such concepts: we are assuming that we know $\mathcal{D}$, but generally we won't. We cannot calculate the error using the probability distribution. However we can calculate an approximation of the error by using the data from the training set, the only data we have access to, which helps us define the following concept.

Definition 2.3. *Given a training set $S = \{(x_i, y_i), i = 1, \ldots, m\}$ and a predictor $h : \mathcal{X} \to \mathcal{Y}$, we can define the **training error**, **empirical error** or **empirical risk** as:*

$$L_S(h) \overset{def}{=} \frac{|\{i \in \{1, \ldots, m\} : h(x_i) \neq y_i\}|}{m}$$

Now it makes sense to evaluate the progression of our learning process by using the empirical risk. This is known as the **empirical risk minimization** (ERM). It is actually what we were using in the decision tree example, we were using the training set to evaluate the entropy of each node.

2.1.2 First case of overfitting

Let's use this method in our example. Let's choose a random condition and calculate the entropy of it, let's take the condition $x < 0.2$. The division of the dataset would result as follows

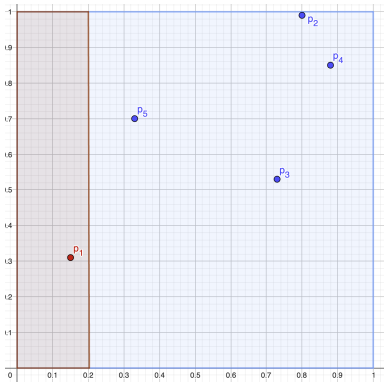

Figure 2.1: Result of the condition $x < 0.2$

If is clear that the entropy is 0 in this case, which means that the learning process would stop here, and the predictor would result in a tree with only one node that would predict that a point is red if his abscissa coordinate is less than 0.2 and blue otherwise.

As we can see, this is really far from what it should be. This is due to the fact that the predictor is adapting too closely to the training dataset (also because the dataset is too small in this particular case). This is known as **overfitting**.

We indeed want to avoid overfitting as it can lead to undesired outputs. A solution to this is creating a **hypothesis class** to restrict the search space during the learning process.

Definition 2.4. *The **hypothesis class** $\mathcal{H}$ is a set of hypothesis or predictors*

$$h \in \mathcal{H} : \mathcal{X} \to \mathcal{Y}$$

.

The $ERM_{\mathcal{H}}$ is the empirical risk minimization algorithm restricted to the hypothesis class $\mathcal{H}$, in other words it it the empirical risk minimization that chooses hypothesis from $\mathcal{H}$. Formally,

$$ERM_{\mathcal{H}}(S) \in argmin_{h \in \mathcal{H}} L_S(h)$$

Remark 2.5. The hypothesis class can also be known as concept class.

2.1.3 Solving overfitting

The simplest restriction we can impose is that $\mathcal{H}$ is finite. We will proceed to prove that under that restriction and a sufficiently large training set, we can make sure that $\text{ERM}_{\mathcal{H}}$ won't overfit.

For this purpose we will assume that there exists $h^* \in \mathcal{H}$ such that $L_{(\mathcal{D},f)}(h^*) = 0$. We will call this the **Realizability Assumption**.

h_S will be the result of applying $\text{ERM}_{\mathcal{H}}$ to the training set S.

We will also assume that S is a set of independent and identically distributed (i.i.d) points according to $\mathcal{D}$. We will denote this assumption as $S \sim \mathcal{D}^m$ where m is the size of S and $\mathcal{D}^m$ denotes the probability over m-tuples induced by applying $\mathcal{D}$ to pick each element of the tuple independently of the other members of the tuple.

We can deduce that $L_{\mathcal{D},f}(h_S)$ is a random variable as it depends on S, which is picked randomly. We cannot make sure that S will lead to an accurate classifier, it will always be possible that the sample S is not representative of the distribution $\mathcal{D}$, as we have seen in the decision tree example. The only thing we can do is calculate the probability of getting a predictor with a given accuracy.

Let's say δ is that probability then $1 - \delta$ is the **confidence parameter**, and ϵ will be the **accuracy** of the output predictor.

That being said we can interpret $L_{(\mathcal{D},f)}(h_S) > \epsilon$ as a failure and we will try to upper bound the probability of getting a sample of m-tuples of instances that will lead to failure, formally if $S|_x = (x_1, \ldots, x_m)$, we want to upper bound:

$$\mathcal{D}^m(\{S|_x : L_{\mathcal{D},f}(h_S) > \epsilon\}) \tag{2.1}$$

Let $\mathcal{H}_B \subset \mathcal{H}$ the set of bad hypotheses, and M the set of misleading samples.

$$\mathcal{H}_B = \{h \in \mathcal{H} : L_{\mathcal{D},f}(h) > \epsilon\} \tag{2.2}$$

$$M = \{S|_x : \exists h \in \mathcal{H}_B, L_S(h) = 0\} = \bigcup_{h \in \mathcal{H}_B} \{S|_x : L_S(h) = 0\} \tag{2.3}$$

Remember that we want to bound the probability of the event $L_{(\mathcal{H},f)}(h_S) > \epsilon$. Since the realizability assumption implies that $L_S(h_S) = 0$ it follows that the event $L_{(\mathcal{H},f)}(h_S) > \epsilon$ is possible only if for some $h \in \mathcal{H}_B$ we have $L_S(h) = 0$ which is possible only if $S \in M$. We have shown that:

$$\{S|_x : L_{(\mathcal{D},f)}(h_S) > \epsilon\} \subseteq M \tag{2.4}$$

Hence,

$$\mathcal{D}^m(\{S|_x : L_{\mathcal{D},f}(h_S) > \epsilon\}) \leq \mathcal{D}^m(M) = \mathcal{D}^m(\cup_{h \in \mathcal{H}_B} \{S|_x : L_S(h) = 0\}) \tag{2.5}$$

We can use the union bound lemma to bound the right hand side of the inequality.

Lemma 2.6. *(Union bound) Given two sets A and B and a distribution $\mathcal{D}$ we have*

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

Applying the lemma to the previous equation (2.5) we have that

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

We can observe that the event $L_S(h) = 0$ is equivalent to the event $\forall i, h(x_i) = f(x_i)$ and since we have assumed that all the x_i in the training set are i.i.d, we have

$$\mathcal{D}^m(\{S|_x : L_S(h) = 0\}) = \mathcal{D}^m(\{S|_x : \forall i, h(x_i) = f(x_i)\}) = \prod_{i=1}^{m} \mathcal{D}(\{x_i : h(x_i) = f(x_i)\}) \tag{2.7}$$

But we have $\mathcal{D}(\{x_i : h(x_i) = y_i\}) = 1 - L_{(\mathcal{D},f)}(h) \leq 1 - \epsilon$
Where the last inequality comes from from the fact that $h \in \mathcal{H}_B$.
Combining this inequality with the equation (2.7) we have that

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

where the last inequality comes from $1 - \epsilon \leq e^{-\epsilon}$. We can conclude that

$$\mathcal{D}^m(\{S|_x : L_{(\mathcal{D},f)}(h_S) > \epsilon\}) \leq |\mathcal{H}_B| e^{-\epsilon m} \leq |\mathcal{H}| e^{-\epsilon m} \tag{2.9}$$

And this makes the first result that can be formulated as follows

Corollary 2.7. *Let $\mathcal{H}$ be a finite hypotheses class, let $\delta \in (0,1)$ and $\epsilon > 0$ and let $m \in \mathbb{Z}$ such that*

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

Then, for any labeling function, f, and for any distribution, $\mathcal{D}$, for which the realizability assumption holds, with probability of at least $1 - \delta$ over the choice of an i.i.d. sample S of size m, we have that for every ERM hypothesis, h_S

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

In other words, if we have a sufficiently large training set, the $\text{ERM}_{\mathcal{H}}$ rule over a finite hypotheses class $\mathcal{H}$ will **probably** (with confidence $1 - \delta$) return an output **approximately** (with confidence ϵ) correct. This introduces the **probably approximately correct** learning theory, we will deep into this in the next section.

2.2 Probably Approximately Correct learning theory

We have just seen in the previous section that if we have a finite hypothesis class and a training set of size m, with m large enough, then we are sure that the ERM will return an approximately correct hypothesis. We will now define this concept in a more general way.

2.2.1 PAC Learning

Definition 2.8. *A hypothesis class $\mathcal{H}$ is said to be **Probably Approximately Correct learnable** (PAC learnable) if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: $\forall \epsilon, \delta \in (0,1)$, for all distribution $\mathcal{D}$ over $\mathcal{X}$ and $\forall f : \mathcal{X} \to \mathcal{Y}$ and if the realizability assumption holds with respect to $\mathcal{H}, \mathcal{D}, f$, then we can run the algorithm on a training set of size $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ and it will return a hypothesis such that, with probability $1 - \delta$, $L_{(\mathcal{D},f)}(h) \leq \epsilon$*

Remark 2.9. The definition of Probably Approximately Correct learnable hypothesis class appeared for the first time in 1984, in a paper published by L.G. Valiant [1].

This definition is basically saying that given the approximation parameters we can find the size of the training dataset such as there exists an algorithm that will return a probably (with confidence parameter $1 - \delta$) approximately (with accuracy parameter ϵ) correct predictor for any distribution and labeling function. Note that we are assuming that the realizabilty assumption holds, we will see later how we can get rid of it.

But we won't be able to get rid of the approximation parameters, we cannot make sure that our training set is representative of $\mathcal{D}$, and even if we do, we can't make sure that our outputted predictor is totally accurate as our training set is always finite.

Definition 2.10. *$m_{\mathcal{H}}$ is also known as **sample complexity** of learning $\mathcal{H}$.*

Remark 2.11. if $m_{\mathcal{H}}$ is a polynomial function we say that $\mathcal{H}$ is efficiently PAC learnable.

Example 2.12. Let's say we want our output to be able to predict if a person will throw rock, paper or scissors in a game. If the person plays randomly it is impossible to get such output, in other words, there is no training set size that would allow us to train a model and get a PAC predictor.

However, if we take the example of the decision tree, if the training set is big enough, the output will be accurate enough as the tree will get deeper since it will only stop once the inherited training sets are perfectly split into their respective classes.

2.2.2 Agnostic PAC Learning

Recall that the realizability assumption states that $\exists h^* \in \mathcal{D}$ s.t. $\mathbb{P}_{x\sim\mathcal{D}}[h^*(x) = f(x)] = 1$. Unfortunately this is not always true, using again the example of the decision tree, if instead of the the $[0, \frac{1}{2}]^2$ square as a red zone we had a fractal, it maybe couldn't be possible to even get a labelling function f in practice. This is why we would like to wave the realizability assumption, here is why we will introduce the agnostic PAC learnability in this section.

For this purpose, from now on, $\mathcal{D}$ will be the **joint distribution** of $\mathcal{X} \times \mathcal{Y}$. In other words $\mathcal{D}_x$ will be the marginal distribution of a random variable $x \in \mathcal{X}$, how likely does x belongs to a certain class, similarly for $\mathcal{D}_y$. That being said we can now redefine what the empirical and true errors are.

Definition 2.13. *We redefine the **true error** as:*

$$L_{\mathcal{D}}(h) \stackrel{def}{=} \mathbb{P}_{(x,y)\in\mathcal{D}}[h(x) \neq y] \stackrel{def}{=} \mathcal{D}(\{(x,y) : h(x) \neq y\})$$

As we tried earlier, our main goal is to find a predictor that minimizes this error, but again the learner won't have access to the distribution $\mathcal{D}$, as before it will only have access to the training data. We will only be able to calculate the empirical risk, which remains the same as in the previous section:

$$L_S(h) \stackrel{def}{=} \frac{|\{i = 1, \ldots, m : h(x_i) \neq y_i|\}}{m}$$

We can now proceed to define what the agnostic PAC learning concept is.

Definition 2.14. *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 that satisfies that $\forall \epsilon, \delta \in (0,1)$ and every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, when running the algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d examples granted by $\mathcal{D}$, it will return a hypothesis h s.t with probability of at least $1 - \delta$*

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

Remark 2.15. 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.

Notice that until now we were using the error function, or the risk function, to evaluate how accurate our hypothesis are, we can generalize this concept to scenarios where we are not talking about predictions.

Definition 2.16. *A **loss function** is a function $l : \mathcal{H} \times Z \to \mathbb{R}_+$ where $\mathcal{H}$ is a hypothesis set and Z is a domain set.*

Notice that we can use this definition of loss function to define what the error is. We can now define the error function in a more general way.

Definition 2.17. *The **risk function** can be redefined, if l is a loss function*

$$L_{\mathcal{D}}(h) \overset{def}{=} \underset{z\sim\mathcal{D}}{\mathbb{E}}[l(h,z)]$$

*And the **empirical risk function** is*

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

Example 2.18.

- The most common loss function is the *square loss function*, defined as follows:

$$l_{sq}(h,(x,y)) = (h(x)-y)^2$$

- Another popular loss function is the *absolute loss function*

$$l_{abs}(h,(x,y)) = |h(x)-y|$$

  This loss function is not as popular as the first one as it can be complicated to apply some calculations to the absolute value, for example when doing some differentiation, as the absolute value function could not be differentiable in all $\mathbb{R}^n$.

- The simplest loss function that one can use is the *0-1 loss function* and it is defined as follows

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

We can now review our definition of agnostic PAC learnable hypothesis using this generalized empirical risk definition.

Definition 2.19. *A hypothesis class $\mathcal{H}$ is **agnostic PAC learnable** with respect to a set Z and a loss function $l : \mathcal{H} \times Z \to \mathbb{R}_+$ if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm that satisfies that $\forall \epsilon, \delta \in (0,1)$ and every distribution $\mathcal{D}$ over Z, when running the algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d examples granted by $\mathcal{D}$, it will return a hypothesis h s.t with probability of at least $1 - \delta$*

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

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

2.3 Uniform Convergence

We have seen in the previous section of this chapter that a finite hypothesis class is PAC learnable. In this section section we will introduce the *uniform convergence*, we will show that every finite hypothesis class is learnable in the agnostic PAC learning model with general loss function, as long as the range loss function is bounded.

2.3.1 Learnability with uniform convergence

Definition 2.20. *Given Z a domain, $\mathcal{H}$ a hypothesis class, l a loss function and $\mathcal{D}$ a distribution, a training set S is said to be **ϵ-representative** if*

$$\forall h \in \mathcal{H}, \quad |L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon$$

Basically, this says that the empirical risk is close enough to the true risk. We will use this in the following lemma.

Lemma 2.21. *Let Z be a domain set, $\mathcal{H}$ a hypothesis set, l a loss function and $\mathcal{D}$ a distribution. If a training set S is $\frac{\epsilon}{2}$-representative, then the output of the $ERM_{\mathcal{H}}(S)$, in other words, $h_S \in argmin_{h \in \mathcal{H}} L_S(h)$, satisfies*

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

Remark 2.22. This lemma basically tells us that if the empirical risk of the training set is close to the true risk with a certain tolerance ($\frac{\epsilon}{2}$-representative training set), then the ERM learning rule will return a good hypothesis.

Proof.

$$\begin{aligned}
S \text{ is } \frac{\epsilon}{2}\text{-representative} &\implies \forall h \in \mathcal{H}, \quad |L_S(h) - L_{\mathcal{D}}(h)| \leq \frac{\epsilon}{2} \\
&\implies L_{\mathcal{D}}(h_S) - L_S(h_S) \leq \frac{\epsilon}{2} \\
&\implies L_{\mathcal{D}}(h_S) \leq L_S(h_S) + \frac{\epsilon}{2} \\
h_S \text{ is an ERM predictor} &\implies L_S(h_S) + \frac{\epsilon}{2} \leq L_S(h) + \frac{\epsilon}{2} \\
\text{again as S is } \frac{\epsilon}{2}\text{-representative} &\implies L_S(h) + \frac{\epsilon}{2} \leq L_{\mathcal{D}}(h) + \frac{\epsilon}{2} + \frac{\epsilon}{2} = L_{\mathcal{D}}(h) + \epsilon
\end{aligned}$$

As this is true $\forall h \in \mathcal{H}$ we can conclude that $L_{\mathcal{D}}(h_S) \leq \min_{h \in \mathcal{H}} \quad L_{\mathcal{D}}(h) + \epsilon$ □

That being said we only have to make sure that we are choosing an ϵ-representative training set (with ϵ small enough) with a certain probability, in our case $1 - \delta$, to confirm the PAC learnability. This is what uniform convergence will be used for we can introduce it as follows.

Definition 2.23. *Given a domain Z and a loss function l, we say that a hypothesis class $\mathcal{H}$ has the **uniform convergence property** if there exists a function $m_{\mathcal{H}}^{UC} : (0,1)^2 \to \mathbb{N}$ such that $\forall \epsilon, \delta \in (0,1)$ and for all probability distribution $\mathcal{D}$ over Z, if S is a sample of size $m \geq m_{\mathcal{H}}^{UC}(\epsilon, \delta)$ drawn according to $\mathcal{D}$ then with probability of at least $1 - \delta$, S is ϵ-representative.*

Similarly to what we saw with PAC learning, $m_{\mathcal{H}}^{UC}$ is called the sample complexity and fixes what the size of the sample should be depending on ϵ and δ to assure that we get an ϵ-representative sample with probability $1 - \delta$. Of course we can only assure such thing with a certain probability, we are never safe from being unlucky and get a non-representative sample.

We can conclude with a Corollary that follows directly from Lemma 2.20 and the definition of uniform convergence.

Corollary 2.24. *If a hypothesis class $\mathcal{H}$ has the uniform convergence property with a function $m_{\mathcal{H}}^{UC}$ then the class is agnostically PAC learnable with sample complexity $m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{UC}(\frac{\epsilon}{2}, \delta)$. Moreover, the $ERM_{\mathcal{H}}$ paradigm is a successful agnostic PAC learner for $\mathcal{H}$.*

2.3.2 Finite classes are Agnostic PAC Learnable

We have seen in section 2.1.3 that finite hypothesis classes are PAC learnable, we will extend this to the agnostic PAC learning method in this section.

We would like to use the Corollary 2.23 to prove this. To do so, we have to prove that uniform convergence holds for finite hypothesis classes.

Corollary 2.25. *Let $\mathcal{H}$ be a hypothesis set, Z a domain and let $l : \mathcal{H} \times Z \to [0,1]$ be a loss function. Then $\mathcal{H}$ enjoys the uniform convergence property with sample complexity*

$$m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq \left\lceil \frac{log\left(\frac{2|\mathcal{H}|}{\delta}\right)}{2\epsilon^2} \right\rceil$$

Furthermore, the class is agnostically PAC learnable using ERM algorithm with sample complexity

$$m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq \left\lceil \frac{log\left(\frac{2|\mathcal{H}|}{\delta}\right)}{2\epsilon^2} \right\rceil$$

We will first prove Hoeffding's inequality

Lemma 2.26. *(Hoeffding's Inequality) Let $\theta_1, \ldots, \theta_m$ a sequence of i.i.d random variables and assume that $\forall i, \mathbb{E}[\theta_i] = \mu, \mathbb{P}[a \leq \theta_i \leq b] = 1$ then for any $\epsilon > 0$*

$$\mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m}\theta_i - \mu\right| > \epsilon\right] \leq 2\exp\left(\frac{-2m\epsilon^2}{(b-a)^2}\right)$$

Proof. (of Corollary 2.24) Fix some ϵ, δ. We have to find a sample size m that guarantees that for every distribution $\mathcal{D}$, with probability $1-\delta$ of the choice of $S = (z_1, \ldots, z_m)$ sampled i.i.d from $\mathcal{D}$ we have that $\forall h \in \mathcal{H}, \quad |L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon$. That is,

$$\mathcal{D}^m(\{S : \forall h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon\}) \geq 1 - \delta$$

Equivalently we have to show that,

$$\mathcal{D}^m(\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) < \delta$$

We can rewrite,

$$\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\} = \cup_{h \in \mathcal{D}}\{S : |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}$$

As we did in a previous proof, we can apply the union bound Lemma 2.6

$$\mathcal{D}^m(\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) \leq \sum_{h \in \mathcal{H}} \mathcal{D}^m(\{S : |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) \tag{2.10}$$

Recall that $L_{\mathcal{D}}(h) = \mathbb{E}_{z\sim\mathbb{D}}[l(h,z)]$. In other words $L_{\mathcal{D}}(H)$ is the expected value of $l(h,z)$. Also $L_S(h) = \frac{1}{m}\sum_{i=1}^{m} l(h,z_i)$, by linearity of expectations it follows that $L_{\mathcal{D}}(h)$ is also the expected value of $L_S(h)$.

In other words $|L_S(h) - L_{\mathcal{D}}(h)|$ is the deviation of a random variable $L_S(h)$ from it's expected value $L_{\mathcal{D}}(h)$. We then have to find the size m for which the probable values of $L_S(h)$ are concentrated enough around $L_S(h)$.

We could use the *law of large numbers* to prove that if m gets bigger, the empirical average converge to the expected value. The problem is that such a law is asymptotically true, this is why we will use the *Hoeffding's inequality* (Lemma 2.25).

For more simplicity in the calculations we will set $l(h,z_i) = \theta_i$ and $\mu = L_{\mathcal{D}}(h)$. Hence, $L_S(h) = \frac{1}{m}\sum_{i=1}^{m}\theta_i$

It is also reasonable to assume that the error function is between 0 and 1. Hence $\theta_i \in [0,1]$. We obtain that each of the terms of the sum in the equation (2.10) are

$$\mathcal{D}^m(\{S : |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) = \mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m}\theta_i - \mu\right| > \epsilon\right] \leq 2e^{-2m\epsilon^2} \tag{2.11}$$

We then obtain from equation (2.10)

$$\mathcal{D}^m(\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) \leq \sum_{h\in\mathcal{H}} 2e^{-2m\epsilon^2} = 2|\mathcal{H}|e^{-2m\epsilon^2} \tag{2.12}$$

Finally we only have to choose

$$m \geq \frac{log\left(\frac{2|\mathcal{H}|}{\delta}\right)}{2\epsilon^2} \tag{2.13}$$

to get the desired result

$$\mathcal{D}^m(\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) \leq \delta. \tag{2.14}$$

□

2.4 The No-Free Lunch Theorem

We have seen in the previous chapter that learning algorithm can lead to overfitting a training set unless we restrict the hypothesis space. In other words we are using some prior knowledge to influence the behavior of the learning process.

A reasonable question that we can ask is which learning model is the most optimal and doesn't require prior knowledge? Even before that we can ask ourselves

if there is a learning algorithm that can give us good result in every scenario? For example, we have used the decision tree algorithm to classify points into two classes, could this same algorithm work for image processing? Could we use it to return the value of a handwritten number?

We will see in this chapter that no such universal learner exists, this is the No-Free Lunch Theorem. Actually, the NFL Theorem is a general Theorem in mathematical optimization, we will state a version of that Theorem that is more applied to our case and prove it, we will state the general Theorem at the end of this section as additional knowledge.

2.4.1 Formulation of the NFL Theorem

Here we introduce a fundamental theorem in learning theory.

Theorem 2.27. *(No-Free Lunch Theorem) Let A be any learning algorithm for the task of binary classification with respect to the 0-1 loss function over a domain set $\mathcal{X}$. Let m be any number smaller than $\frac{|\mathcal{X}|}{2}$, representing the size of a training dataset. Then, there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ such that:*

1. *There exists a function $f : \mathcal{X} \to \{0,1\}$ with $L_{\mathcal{D}}(f) = 0$.*
2. *With probability of at least $\frac{1}{7}$ over the choice of $S \sim \mathcal{D}^m$ we have that $L_{\mathcal{D}}(A(S)) \geq \frac{1}{8}$*

Basically, this Theorem states that for every learning algorithm there exists a task for which it fails but another learner would succeed.

Proof. Let $C \subset \mathcal{X}$ be a subset of size $2m$. Note that there exist $T = 2^{2m}$ possible labelling functions from C to $\{0,1\}$, we denote $f_1, f_2, \ldots, f_T$ these labelling functions. For each function f_i, $\mathcal{D}_i$ will be the distribution over $C \times \{0,1\}$ defined by

$$\mathcal{D}_i(x,y) = \begin{cases} \frac{1}{|C|} & \text{if} \quad y = f_i(x) \\ 0 & \text{otherwise} \end{cases}$$

Basically the probability to choose a pair (x,y) is $\frac{1}{|C|}$ if y is the true label according to f and the probability is 0 if $y \neq f_i(x)$. Clearly, $L_{\mathcal{D}_i}(f_i) = 0$.

We want to show that for every algorithm A that receives a training set of size m from $C \times \{0,1\}$ and returns $A(S) : C \to \{0,1\}$, it holds that

$$\max_{i=1,\ldots T} \quad \mathbb{E}_{S \sim \mathcal{D}_i^m}[L_{\mathcal{D}_i}(A(S))] \geq \frac{1}{4} \tag{2.15}$$

Clearly this means that for every algorithm, A' that receives a training set of size m from $\mathcal{X} \times \{0,1\}$ there exists a function $f : \mathcal{X} \to \{0,1\}$ and a distribution $\mathcal{D}$ over $X \times \{0,1\}$, such that $L_{\mathcal{D}}(f) = 0$ and

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(A'(S))] \geq \frac{1}{4}. \tag{2.16}$$

We will now proceed to prove equation (2.15). We have $k = (2m)^m$ possible sequences of m examples from C. Denote these sequences by $S_1, \ldots, S_k$. Also, if $S_j = (x_1, \ldots, x_m)$, we denote by $S_j^i = ((x_1, f_i(x_1)), \ldots, (x_m, f_i(x_m)))$. If the distribution is $\mathcal{D}_i$ then the possible training sets A can receive $S_i^i, \ldots, S_k^i$ and all these training sets have the same probability of being sampled. Therefore,

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}_i^m}[L_{\mathcal{D}_i}(A(S))] = \frac{1}{k}\sum_{j=1}^{k} L_{\mathcal{D}_i}(A(S_j^i)). \tag{2.17}$$

We can use the fact that the maximum is always greater that the average which is greater than the minimum, we then have

$$\begin{aligned} \max_{i=1,\ldots,T} \frac{1}{k}\sum_{j=1}^{k} L_{\mathcal{D}_i}(A(S_j^i)) &\geq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{k}\sum_{j=1}^{k} L_{\mathcal{D}_i}(A(S_j^i)) \\ &= \frac{1}{k}\sum_{j=1}^{k}\frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)) \\ &\geq \min_{j=1,\ldots,k} \frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)). \end{aligned} \tag{2.18}$$

We can now fix some $j = 1, \ldots, k$. Denote $S_j = (x_1, \ldots, x_m)$ and let $v_1, \ldots, v_p$ be the examples in C that do not appear in S_j. Clearly $p \geq m$. Therefore, for every function $h : C \to \{0,1\}$ and every i we have

$$\begin{aligned} L_{\mathcal{D}_i}(h) &= \frac{1}{2m}\sum_{x \in C} \mathbb{1}_{[h(x) \neq f_i(x)]} \\ &\geq \frac{1}{2m}\sum_{r=1}^{p} \mathbb{1}_{[h(v_r) \neq f_i(v_r)]} \\ &\geq \frac{1}{2p}\sum_{r=1}^{p} \mathbb{1}_{[h(v_r) \neq f_i(v_r)]}. \end{aligned} \tag{2.19}$$

Hence,

$$\begin{aligned}\frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)) &\geq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{2p}\sum_{r=1}^{p}\mathbb{1}_{[A(S_j^i)(v_r)\neq f_i(v_r)]} \\ &= \frac{1}{2p}\sum_{r=1}^{p}\frac{1}{T}\sum_{i=1}^{T}\mathbb{1}_{[A(S_j^i)(v_r)\neq f_i(v_r)]} \\ &\geq \frac{1}{2}.\min_{r=1,\dots,p}\frac{1}{T}\sum_{i=1}^{T}\mathbb{1}_{[A(S_j^i)(v_r)\neq f_i(v_r)]}.\end{aligned} \tag{2.20}$$

Now fix some $r = 1, \dots, p$. We can partition all the functions in $f_1, \dots, f_T$ into $\frac{T}{2}$ disjoint pairs, where for a pair $(f_i, f_{i'})$ we have that for every $c \in C$, $\quad f_i(c) \neq f_{i'}(c) \iff c = v_r$. Since for such a pair we must have $S_j^i = S_j^{i'}$, it follows that

$$\mathbb{1}_{[A(S_j^i)(v_r)\neq f_i(v_r)]} + \mathbb{1}_{[A(S_j^{i'})(v_r)\neq f_{i'}(v_r)]} = \frac{1}{2}. \tag{2.21}$$

which yields

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{1}_{[A(S_j^i)(v_r)\neq f_i(v_r)]}. \tag{2.22}$$

Combining this equation with the equations (2.20), (2.18), (2.17) we finally obtain the equation (2.15), which concludes the proof. □

We will now state the general No-Free Lunch theorem as additional knowledge, as we mentioned earlier.

Theorem 2.28. *(No-Free Lunch Theorem Generalized) Given a finite set V and a finite set S of real numbers, assume that $f : V \to S$ is chosen randomly according to uniform distribution on the set S^V of all possible functions from V to S. For the problem of optimizing f over the set V, then no algorithm performs than blind search.*

This Theorem states that when all functions f are equally likely, the probability of observing an arbitrary sequence of m values in the course of optimization does not depend upon the algorithm.

2.5 The VC-dimension

We have seen until now that finite hypothesis classes can be learnt throw the ERM rule. Nonetheless we could ask ourselves if classes have to be finite to be learnable, in other words, are infinite classes learnable? What does actually make classes learnable?

We will begin the section showing that infinite classes can be learnt, we will then present the VC-dimension (Vapnik-Chervonenkis dimension) and we will explore its correlation with learnability of hypothesis classes. We will indeed state the fundamental theorem of PAC learning.

2.5.1 Infinite classes can be learnt

We will see an example of infinite hypothesis class that can be learnable in this section.

We could imagine that $\mathcal{H}$ is the set of all the thresholds over the real numbers: $\mathcal{H} = \{h_a : a \in \mathbb{R}\}$ where $h_a : \mathbb{R} \to \{0,1\}$, indeed $h_a = \mathbb{1}_{[x<a]}$. It is clear that the hypothesis class $\mathcal{H}$ is infinite. Nonetheless the following lemma proves that this hypothesis class can be learnt.

Lemma 2.29. *Let $\mathcal{H}$ be the set of all the threshold functions over $\mathbb{R}$. Then, $\mathcal{H}$ is PAC learnable, using the ERM rule with sample complexity of $m_{\mathcal{H}}(\epsilon,\delta) = \left\lceil \frac{log(\frac{2}{\delta})}{\epsilon} \right\rceil$*

Proof. Let a^* be the threshold such that the hypothesis $h^*(x) = \mathbb{1}_{[x<a^*]}$ achieves $L_{\mathcal{D}}(h^*) = 0$. Let $\mathcal{D}_x$ be the marginal distribution over the domain $\mathcal{X}$ and let $a_0 < a^* < a_1$ such that

$$\mathop{\mathbb{P}}_{x\sim\mathcal{D}_x}[x \in (a_0, a^*)] = \mathop{\mathbb{P}}_{x\sim\mathcal{D}_x}[x \in (a^*, a_1)] = \epsilon \tag{2.23}$$

In the case that $\mathcal{D}_x(-\infty, a^*) \leq \epsilon$ we can set $a_0 = -\infty$, similarly for a_1. Given a training set S, let $b_0 = max\{x : (x,1) \in S\}$ and $b_1 = min\{x : (x,0) \in S\}$, if no example in S is positive we can set $b_0 = -\infty$ and if no example in S is negative, we can set $b_1 = \infty$. Let b_S be the threshold of h_S, the hypothesis that we obtain by running the ERM rule on S. We can deduce that $b_S \in (b_0, b_1)$, therefore, a sufficient condition for $L_{\mathcal{D}}(h_S) \leq \epsilon$ is that both $b_0 \geq a_0$ and $b_1 \leq a_1$. In other words,

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(h_S) > \epsilon] \leq \mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[(b_0 < a_0) \cup (b_1 > a_1)] \tag{2.24}$$

Using the union bound we can deduce:

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(h_S) > \epsilon] \leq \mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[b_0 < a_0] + \mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[b_1 > a_1] \tag{2.25}$$

The event $b_0 < a_0$ happens if and only if all examples in S are not in the interval (a_0, a^*), whose probability is ϵ as defined:

$$\mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[b_0 < a_0] = \mathop{\mathbb{P}}_{S\sim\mathcal{D}^m}[\forall (x,y) \in S, x \notin (a_0, a^*)] = (1-\epsilon)^m \leq e^{-\epsilon m} \tag{2.26}$$

Since we are assuming that $m > \frac{log(\frac{2}{\delta})}{\epsilon}$ it follows that the equation (2.26) is less than $\frac{\delta}{2}$. With the same argument we can say that $\underset{S\sim\mathcal{D}^m}{\mathbb{P}}[b_1 > a_1] \leq \frac{\delta}{2}$. Combining this result with the equation (2.25) we obtain the desired result, which concludes the proof. □

2.5.2 The VC-dimension

We have seen in the previous section that the finiteness of a hypothesis class is not necessary for the learnability of that class. We have also seen the No-Free Lunch Theorem, which says that without any restriction on the hypothesis class, given a learning algorithm, we will always find an adversary distribution where the learning process fails, but another one could succeed.

In this section we will introduce the VC-dimension of a hypothesis class, which will help us determine its learnability. We will first rigorously define what a restriction stands for.

Definition 2.30. *Let $\mathcal{H}$ be a class of functions from $\mathcal{X}$ to $\{0,1\}$ and let $\mathcal{C} = \{c_1, \dots, c_m\} \subset \mathcal{X}$. The **restriction of $\mathcal{H}$ to** $\mathcal{C}$ is the set of functions from $\mathcal{C}$ to $\{0,1\}$ that can be derived from $\mathcal{H}$. That is,*

$$\mathcal{H}_\mathcal{C} = \{(h(c_1), \dots, h(c_m)) : h \in \mathcal{H}\}$$

We will represent each function from $\mathcal{C}$ to $\{0,1\}$ as a vector in $\{0,1\}^{|\mathcal{C}|}$

This introduces the definition of *shattering*, which says:

Definition 2.31. *A hypothesis class $\mathcal{H}$ **shatters** a finite set $\mathcal{C} \subset \mathcal{X}$ if the restriction of $\mathcal{H}$ to $\mathcal{C}$ is the set of all functions from $\mathcal{C}$ to $\{0,1\}$. That is, $|\mathcal{H}_\mathcal{C}| = 2^{|\mathcal{C}|}$.*

Example 2.32. We can see an easy example taking $\mathcal{H}$ as the set of all possible thresholds functions over $\mathbb{R}$. If $\mathcal{C} = \{c_1\}$, we can see that if $a = c_1 + 1$, then $h_a(c_1) = 1$, and if $a = c_1 - 1$, then we have that $h_a(c_1) = 0$. Therefore $\mathcal{H}_\mathcal{C}$ is the set of all functions from $\mathcal{C}$ to $\{0,1\}$, and $\mathcal{H}$ shatters $\mathcal{C}$.

A counter example could be taking $\mathcal{C} = \{c_1, c_2\}$ with $c_1 < c_2$, then any threshold that assigns 0 to c_1 will also assign 0 to c_2, therefore not all functions from $\mathcal{C}$ to $\{0,1\}$ are included in $\mathcal{H}_\mathcal{C}$, hence $\mathcal{C}$ is not shattered by $\mathcal{H}$.

Corollary 2.33. *Let $\mathcal{H}$ be a hypothesis class of functions from $\mathcal{X}$ to $\{0,1\}$. Let m be a training set size. Assume that there exists a set $C \subset \mathcal{X}$ of size $2m$ that is shattered by $\mathcal{H}$. Then, for any learning algorithm, A, there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ and a predictor $h \in \mathcal{H}$ such that $L_\mathcal{D}(h) = 0$ but with probability of at least $\frac{1}{7}$ over the choice of $S \sim \mathcal{D}^m$ we have that $L_\mathcal{D}(A(S)) \geq \frac{1}{8}$.*

Remark 2.34. This corollary basically tells us that if $\mathcal{H}$ shatters some set $\mathcal{C}$ of size $2m$ then we cannot learn $\mathcal{H}$ using m examples. Intuitively, if a set $\mathcal{C}$ is shattered by a hypothesis class $\mathcal{H}$ and we receive a training set containing half the instances of $\mathcal{C}$, the labels of these instances give us no information about the labels of the rest of the instances in $\mathcal{C}$, every possible labelling of the rest of the instances can be explained by some hypothesis in $\mathcal{H}$. Summering up, *if someone can explain every phenomenon, his explanations are worthless*.

Definition 2.35. *(VC-dimension) The **VC-dimension** of a hypothesis class $\mathcal{H}$ is the maximal size of a set $\mathcal{C} \subset \mathcal{X}$ that can be shattered by $\mathcal{H}$. If $\mathcal{H}$ can shatter any set of any size we say that $\mathcal{H}$ has an infinite VC-dimension. We will denote the VC-dimension of a hypothesis class $\mathcal{H}$ as $VCdim(\mathcal{H})$.*

From this definition we can use the Corollary (2.32) to get our first result about VC-dimension:

Theorem 2.36. *Let $\mathcal{H}$ be a hypothesis class such as $VCdim(\mathcal{H}) = \infty$. Then, $\mathcal{H}$ is not PAC learnable.*

Proof. Since $\mathcal{H}$ has an infinite VC-dimension, for any training set size m, there exists a shattered set size $2m$, and the Corollary (2.32) tells us that $\mathcal{H}$ is not PAC learnable. □

We will see later that the converse is also true, that is, if a hypothesis class has a finite VC-dimension then it is PAC learnable. Hence, the VC-dimension characterizes the PAC learnability of a hypothesis class.

We will now see some examples of how to calculate the VC-dimension of hypothesis classes.

Example 2.37.

- **Threshold functions**

  As we have seen in a previous example if we take $\mathcal{C} = \{c_1\}$, then $\mathcal{H}$ shatters $\mathcal{C}$, therefore $VCdim(\mathcal{H}) \geq 1$. Moreover, if $\mathcal{C} = \{c_1, c_2\}$ with $c_1 \leq c_2$, $\mathcal{H}$ does not shatter $\mathcal{C}$. We conclude the $VCdim(\mathcal{H}) = 1$.

- **Intervals**

  Let $\mathcal{H}$ be the class of all intervals in $\mathbb{R}$, namely, $\mathcal{H} = \{h_{a,b} : a, b \in \mathbb{R}, a < b\}$, where $h_{a,b} : \mathbb{R} \to \{0, 1\}$ is a function such that $h_{a,b} = \mathbb{1}_{[x \in (a,b)]}$. If $\mathcal{C} = \{1, 2\}$. We can take $a < b < 1$ then $h_{a,b}(1) = 0$ and $h_{a,b}(2) = 0$. If $a < 1 < b < 2$, then $h_{a,b}(1) = 1$ and $h_{a,b}(2) = 0$. Finally, if $1 < a < b < 2$, then $h_{a,b}(1) = 1$ and $h_{a,b}(1) = 1$. In other words, for every function from C to $\{0, 1\}$, we can find

a function in $\mathcal{H}$ that gives the same output, so $\mathcal{H}$ shatters C and therefore $VCdim(\mathcal{H}) \geq 2$. But if we take $\mathcal{C} = \{c_1, c_2, c_3\}$, assuming $c1 < c2 < c3$, we cannot find a function in $\mathcal{H}$ such that $h_{a,b}(c_1) = 1$, $h_{a,b}(c_2) = 0$ and $h_{a,b}(c_3) = 1$. This is because $h_{a,b}(c_1) = 1 \implies a < c_1$
$h_{a,b}(c_3) = 1 \implies b > c_3 \implies c_2 \in (a, b) \implies h_{a,b}(c_2) = 1 \neq 0$. Therefore $\mathcal{H}$ does not shatter $\mathcal{C}$, and $VCdim(\mathcal{H}) = 2$.

- **Finite classes**

  Let $\mathcal{H}$ be a finite class. Then we have that for any set $\mathcal{C}$ we have $|\mathcal{H}_\mathcal{C}| \leq |\mathcal{H}|$ and thus $\mathcal{C}$ cannot be shattered if $|\mathcal{H}| < 2^{|\mathcal{C}|}$. This implies that $VCdim(\mathcal{H}) \leq log_2(|\mathcal{H}|)$. This shows that PAC learnability of finite classes follows from the more general statement of PAC learnability of classes of finite VC-dimension, which we shall see in the next section. Note, however, that the VC-dimension of a finite class $\mathcal{H}$ can be significantly smaller than $log_2(|\mathcal{H}|)$. For example, let $\mathcal{X} = \{1, \ldots, k\}$, for some integer k, and consider the class of thresholds functions (as defined in the previous example). Then, $|\mathcal{H}| = k$ but $VCdim(\mathcal{H}) = 1$. Since k can be arbitrarily large, the gap between $log_2(|\mathcal{H}|)$ and $VCdim(\mathcal{H})$ can be arbitrarily large.

2.5.3 The Fundamental Theorem of PAC learning

As we mentioned in the previous section, a hypothesis class with infinite VC-dimension is not PAC learnable, we will now see the converse of that, that is, a hypothesis class with finite VC-dimension is learnable.

Theorem 2.38. *(The Fundamental Theorem of Statistical Learning) Let $\mathcal{H}$ be a hypothesis class of functions from a domain set $\mathcal{X}$ to $\{0, 1\}$ and let the loss function be the 0-1 loss function. Then, the following are equivalent:*

1.$\mathcal{H}$ has the uniform convergence property.
2.Any ERM rule is a successful agnostic PAC learner for $\mathcal{H}$.
3.$\mathcal{H}$ is agnostic PAC learnable.
4.$\mathcal{H}$ is PAC learnable.
5. Any ERM rule is a successful PAC learner for $\mathcal{H}$.
6.$\mathcal{H}$ has a finite VC-dimension.

To prove this Theorem we will first state some previous results.

Definition 2.39. *(Growth function) Let $\mathcal{H}$ be a hypothesis class. Then the **growth function** of $\mathcal{H}$ is defined as:*

$$\tau_\mathcal{H}(m) = \max_{\mathcal{C} \subset \mathcal{H} : |\mathcal{C}| = m} |\mathcal{H}_\mathcal{C}|$$

In other words, $\tau_{\mathcal{H}}(m)$ is the number of distinct functions from a set $\mathcal{C}$ of size m to $\{0,1\}$ that can be obtained by restricting $\mathcal{H}$ to $\mathcal{C}$.

We can remark that if $VCdim(\mathcal{H}) = d$ then for any $m \leq d$ we have that $\tau_{\mathcal{H}}(m) = 2^m$. We will now see a result that states that when m becomes greater than the VC-dimension, the growth function increases polynomially rather than exponentially with m.

Lemma 2.40. *(Sauer-Shelah-Perles) Let $\mathcal{H}$ be a hypothesis class with $VCdim(\mathcal{H}) \leq d < \infty$. Then, for all m, $\tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{d} \binom{m}{i}$. In particular, if $m > d+1$ then $\tau_{\mathcal{H}}(m) \leq (\frac{em}{d})^d$*

Proof. Observe that we only have to prove that for any $\mathcal{C} = \{c_1, \ldots, c_m\}$ we have

$$\forall \mathcal{H}, \quad |\mathcal{H}_{\mathcal{C}}| \leq |\{\mathcal{B} \subseteq \mathcal{C} : \mathcal{H} \text{ shatters } \mathcal{B}\}| \tag{2.27}$$

The reason why this is sufficient is that if $VCdim(\mathcal{H}) \leq d$ then no set whose size is larger than d is shattered by $\mathcal{H}$ and therefore:

$$|\{\mathcal{B} \subseteq \mathcal{C} : \mathcal{H} \text{ shatters } \mathcal{B}\}| \leq \sum_{i=0}^{d} \binom{m}{i}$$

When $m > d+1$ the right-hand side of the preceding is at most $(\frac{em}{d})^d$. This can be proved easily by induction.

We will now use induction in order to prove that the equation (2.27) is true.

For $m = 1$ we will always have the two sides of the equation (2.27) equal to 1 or 2 (the empty set is considered to be shattered by $\mathcal{H}$). Assume that the equation (2.27) holds for all the sets of size $k < m$, let's prove that it also holds for sets of size m.

Fix $\mathcal{H}$ and let $\mathcal{C} = \{c_1, \ldots, c_m\}$. Denote $\mathcal{C}' = \{c_2, \ldots, c_m\}$. We define also

$$Y_0 = \{(y_2, \ldots, y_m) : (0, y_2, \ldots, y_m) \in \mathcal{H}_{\mathcal{C}} \text{ or } (1, y_2, \ldots, y_m) \in \mathcal{H}_{\mathcal{C}}\}$$

and

$$Y_1 = \{(y_2, \ldots, y_m) : (0, y_2, \ldots, y_m) \in \mathcal{H}_{\mathcal{C}} \text{ and } (1, y_2, \ldots, y_m) \in \mathcal{H}_{\mathcal{C}}\}$$

We can see that $|\mathcal{H}_{\mathcal{C}}| = |Y_0| + |Y_1|$. Moreover, since $Y_0 = \mathcal{H}_{\mathcal{C}'}$, using the induction assumption (applied on $\mathcal{H}$ and $\mathcal{C}'$) we have that

$$|Y_0| = |\mathcal{H}_{\mathcal{C}'}| \leq |\{\mathcal{B} \subseteq \mathcal{C}' : \text{ shatters } \mathcal{B}\}| = |\{\mathcal{B} \subseteq \mathcal{C} : c_1 \notin \mathcal{B} \text{ and } \mathcal{H} \text{ shatters } \mathcal{B}\}|$$

Next, define $\mathcal{H}' \subseteq \mathcal{H}$ to be

$$\begin{aligned}\mathcal{H}' = \{h \in \mathcal{H} : \exists h' \in \mathcal{H} \text{ s.t } (1 - h'(c_1), h'(c_2), \ldots, h'(c_m)) \\ = (h(c_1), \ldots, h(c_m))\}\end{aligned} \tag{2.28}$$

Namely, $\mathcal{H}'$ contains pairs of hypothesis that agree on $\mathcal{C}'$ and differ on c_1. Using this definition, it is clear that if $\mathcal{H}'$ shatters a set $\mathcal{B} \subseteq \mathcal{C}'$ then it also shatters the set $\mathcal{B} \cup \{c_1\}$ and vice versa. Combining this with the fact that $Y_1 = \mathcal{H}'_{\mathcal{C}}$, and using the inductive assumption (now on $\mathcal{H}'$) and $\mathcal{C}'$ we obtain that

$$\begin{aligned}|Y_1| &= |\mathcal{H}'_{\mathcal{C}}| \\ &= |\{\mathcal{B} \subseteq \mathcal{C}' : \mathcal{H}' \text{ shatters } \mathcal{B}\}| \\ &= |\{\mathcal{B} \subseteq \mathcal{C}' : \mathcal{H}' \text{ shatters } \mathcal{B} \cup \{c_1\}\}| \\ &= |\{\mathcal{B} \subseteq \mathcal{C} : \{c_1\} \in \mathcal{B} \text{ and } \mathcal{H}' \text{ shatters } \mathcal{B}\}| \\ &= |\{\mathcal{B} \subseteq \mathcal{C} : \{c_1\} \in \mathcal{B} \text{ and } \mathcal{H} \text{ shatters } \mathcal{B}\}|\end{aligned} \tag{2.29}$$

Overall, we obtain that

$$\begin{aligned}|\mathcal{H}_{\mathcal{C}}| &= |Y_0| + |Y_1| \\ &\leq |\{\mathcal{B} \subseteq \mathcal{C} : \{c_1\} \notin \mathcal{B} \text{ and } \mathcal{H} \text{ shatters } \mathcal{B}\}| + |\{\mathcal{B} \subseteq \mathcal{C} : \{c_1\} \in \mathcal{B} \text{ and } \mathcal{H} \text{ shatters } \mathcal{B}\}| \\ &= |\{\mathcal{B} \subseteq \mathcal{C} : \mathcal{H} \text{ shatters } \mathcal{B}\}|\end{aligned} \tag{2.30}$$

Which concludes the proof □

Another important result that we will need in order to prove the Fundamental Theorem of PAC Learning is that if $\mathcal{H}$ has a *small effective size* then it enjoys the uniform convergence property. By *small effective size* we mean a class for which $|\mathcal{H}_{\mathcal{C}}|$ grows polynomially with $|\mathcal{C}|$. Formally,

Theorem 2.41. *Let $\mathcal{H}$ be a a class and let $\tau_{\mathcal{H}}$ be its growth function. Then, for every $\mathcal{D}$ and every $\delta \in (0,1)$, with probability of at least $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$ we have*

$$|L_{\mathcal{D}}(h) - L_S(h)| \leq \frac{4 + \sqrt{log(\tau_{\mathcal{H}}(2m))}}{\delta\sqrt{2m}}$$

Proof. Let's start by showing that

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}\left[sup_{h \in \mathcal{H}} |L_{\mathcal{D}}(h) - L_S(h)|\right] \leq \frac{4 + \sqrt{log(\tau_{\mathcal{H}}(2m))}}{\sqrt{2m}} \tag{2.31}$$

To bound the left-hand side of the equation (2.31) we first note that for every $h \in \mathcal{H}$, we can rewrite $L_{\mathcal{H}}(h) = \mathop{\mathbb{E}}_{S' \sim \mathcal{H}^m}[L_{S'}(h)]$, where $S' = \{z'_1, \ldots, z'_m\}$ is an additional i.i.d sample. Therefore,

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}\left[sup_{h \in \mathcal{H}} |L_{\mathcal{D}}(h) - L_S(h)|\right] = \mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}\left[sup_{h \in \mathcal{H}} \left| \mathop{\mathbb{E}}_{S' \sim \mathcal{D}^m}\left[L_{\mathcal{D}}(h) - L_S(h)\right]\right|\right] \tag{2.32}$$

A generalization of the triangle inequality yields

$$\left| \mathop{\mathbb{E}}_{S' \sim \mathcal{D}^m}\left[L_{\mathcal{D}}(h) - L_S(h)\right]\right| \leq \mathop{\mathbb{E}}_{S' \sim \mathcal{D}^m}\left|L_{\mathcal{D}}(h) - L_S(h)\right|$$

And the fact that supremum of expectation is smaller than the expectation of supremum yields

$$sup_{h \in \mathcal{H}} \mathop{\mathbb{E}}_{S' \sim \mathcal{D}^m}\left|L_{\mathcal{D}}(h) - L_S(h)\right| \leq \mathop{\mathbb{E}}_{S' \sim \mathcal{D}^m} sup_{h \in \mathcal{H}'}\left|L_{\mathcal{D}}(h) - L_S(h)\right|$$

Combining all we obtain

$$\begin{aligned}\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}\left[sup_{h \in \mathcal{H}} |L_{\mathcal{D}}(h) - L_S(h)|\right] &\leq \mathop{\mathbb{E}}_{S, S' \sim \mathcal{D}^m}\left[sup_{h \in \mathcal{H}}\left|L_{\mathcal{D}}(h) - L_S(h)\right|\right] \\ &= \mathop{\mathbb{E}}_{S, S' \sim \mathcal{D}^m}\left[sup_{h \in \mathcal{H}} \frac{1}{m}\left|\sum_{i=1}^{m}(l(h, z'_i) - l(h, z_i))\right|\right]\end{aligned} \tag{2.33}$$

Since all the choices in S and S' are i.i.d, nothing will change if we change the name of the random vectors z_i with the names of the vectors z'_i. This would replace the term $(l(h, z'_i) - l(h, z_i))$ by the term $-(l(h, z'_i) - l(h, z_i))$, it follows that for every $\sigma \in \{-1, 1\}^m$ we have that the equation (2.33) equals

$$\mathop{\mathbb{E}}_{S, S' \sim \mathcal{D}^m}\left[sup_{h \in \mathcal{H}} \frac{1}{m}\left|\sum_{i=1}^{m} \sigma_i(l(h, z'_i) - l(h, z_i))\right|\right]$$

Since this is true for every $\sigma \in \{-1, 1\}^m$, it is also true if we sample each component of σ uniformly at random from the uniform distribution over $\{-1, 1\}^m$ (denoted here as $\mathbf{U}_{\pm}$). This give us,

$$\mathop{\mathbb{E}}_{\sigma \sim \mathbf{U}_{\pm}^m} \mathop{\mathbb{E}}_{S, S' \sim \mathcal{D}^m}\left[sup_{h \in \mathcal{H}} \frac{1}{m}\left|\sum_{i=1}^{m} \sigma_i(l(h, z'_i) - l(h, z_i))\right|\right]$$

By the linearity of the expectations we get that

$$\mathop{\mathbb{E}}_{S, S' \sim \mathcal{D}^m} \mathop{\mathbb{E}}_{\sigma \sim \mathbf{U}_{\pm}^m}\left[sup_{h \in \mathcal{H}} \frac{1}{m}\left|\sum_{i=1}^{m} \sigma_i(l(h, z'_i) - l(h, z_i))\right|\right]$$

We can now fix S and S' and let C be the instances appearing in S and S', then we can take the supremum only over $h \in \mathcal{H}_C$. Then,

$$\mathop{\mathbb{E}}_{\sigma\sim\mathbf{U}_{\pm}^m}\left[sup_{h\in\mathcal{H}}\frac{1}{m}\left|\sum_{i=1}^{m}\sigma_i(l(h,z_i')-l(h,z_i))\right|\right] = \mathop{\mathbb{E}}_{\sigma\sim\mathbf{U}_{\pm}^m}\left[sup_{h\in\mathcal{H}}\frac{1}{m}\left|\sum_{i=1}^{m}\sigma_i(l(h,z_i')-l(h,z_i))\right|\right]$$

If we fix some $h \in \mathcal{H}_C$ and denote $\theta_h = \frac{1}{m}\sum_{i=1}^{m}\sigma_i(l(h,z_i')-l(h,z_i))$. Since $\mathbb{E}[\theta_h] = 0$ and θ_h is an average of independent variables, each of which takes values in the interval $[-1,1]$, we can use Hoeffding's Inequality (Lemma 2.25) and say that we have that for every $\rho > 0$

$$\mathbb{P}[|\theta_h| > \rho] \leq 2exp(-2m\rho^2)$$

We can use the following lemma,

Lemma 2.42. *Let X be a random variable and $x' \in \mathbb{R}$ be a scalar and assume that there exists $a > 0$ and $b \geq e$ such that for all $t \geq 0$ we have $\mathbb{P}[|X - x'| > t] \leq 2be^{-t^2/a^2}$. Then, $\mathbb{E}[|X - x'|] \leq a(2 + \sqrt{log(b)})$.*

Using this lemma we can deduce that

$$\mathbb{E}\left[\max_{h\in\mathcal{H}_C}|\theta_h| > p\right] \leq \frac{4 + \sqrt{log(\tau_{\mathcal{H}}(2m))}}{\sqrt{2m}}$$

Combining all with the definition of $\tau_{\mathcal{H}}$, we have shown that

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}\left[sup_{h\in\mathcal{H}}|L_{\mathcal{D}}(h) - L_S(h)|\right] \leq \frac{4 + \sqrt{log(\tau_{\mathcal{H}}(2m))}}{\sqrt{2m}}$$

And it is now easy to deduce the equation from the Theorem we are proving, which concludes the proof. □

With these results being settled we can now proceed onto proving the Fundamental Theorem of PAC Learning.

Proof. (of Theorem 2.37) We have seen in the previous section about Uniform Convergence that $1 \to 2$. The implications $2 \to 3$ and $3 \to 4$ are trivial as we have seen that the definition of agnostic PAC learning is stronger than the definition of PAC learning. The implications of $4 \to 6$ and $5 \to 6$ follow from the No-Free Lunch Theorem. The difficult part is to prove that $6 \to 1$.

Let's start by proving that if the VC-dimension is finite then the uniform convergence property holds. From Sauer-Shelah-Perles's Lemma 2.39 we have that for

$m > d, \quad \tau_{\mathcal{H}}(2m) \leq (2em/d)^d$. Combining this result with the previous Theorem (2.40) we obtain that with probability of at least $1 - \delta$

$$|L_S(h) - L_{\mathcal{D}}(h)| \leq \frac{4 + \sqrt{d \log(2em/d)}}{\delta\sqrt{2m}}$$

For simplicity we can assume that $\sqrt{d \log(2em/d)} \geq 4$, hence

$$|L_S(h) - L_{\mathcal{D}}(h)| \leq \frac{1}{\delta}\sqrt{\frac{2d \log(2em/d)}{m}}$$

To ensure that the proceeding is at most ϵ we need that

$$m \geq \frac{2d \log(m)}{(\delta\epsilon)^2} + \frac{2d \log(2\epsilon/d)}{(\delta\epsilon)^2}$$

It is easy to see that after some standard algebraic manipulation we obtain that it is sufficient to assume that

$$m \geq 4\frac{2d}{(\delta\epsilon)^2} \log\left(\frac{2d}{(\delta\epsilon)^2}\right) + \frac{4d \log(2\epsilon/d)}{(\delta\epsilon)^2}$$

We can deduce from this result that the $|\mathcal{H}_{\mathcal{C}}|$ grows polynomially with $|\mathcal{C}|$. Which means it has a small effective size, and we have seen that this means it has the uniform convergence property. □

2.6 Summary

In this chapter we have introduced the fundamentals of Learning Theory, first of all we have established methods to obtain predictors by minimizing their error with a training dataset, but as we cannot calculate the true error as we don't know the expected result for every prediction, we have used the empirical error. Like this we have established the empirical risk minimization rule.

We have then formally defined what learning means, making it clear that it always depends on a probability, as our learning process has an inevitable chance to fail, and a tolerance, as the output predictor cannot be completely accurate, this is the probably approximately correct learning theory. We have also seen a more general approach removing the realizabilty assumption, the agnostic PAC learning theory.

We have then defined when does a hypothesis class has the uniform convergence property and we have used this condition to prove that every finite hypothesis class is learnable if it has the mentioned property.

We have then seen our first fundamental result in learning theory, the No-Free Lunch Theorem, which states that there is no learning algorithm that can work in every possible learning scenario, in other words, for every learning algorithm, there exists a learning scenario where it fails, but another algorithm would succeed.

In the last section we have discussed about the VC-dimension, which gives us even more information about the learning possibilities. Formally, we have proved the Fundamental Theorem of PAC learning, which states that a hypothesis class is learnable if and only if its VC-dimension is finite.

Now we will explore a topic that is a direct application of what we have just seen, and that is also relevant nowadays: the out of distribution detection.

Chapter 3

Out of distribution Detection

Imagine we want to train an algorithm that is able to detect which number has been handwritten. First we will collect a lot of handwritten numbers and we will use them as training data for our learner. Let's say we achieve a high accuracy, for the purpose let's assume we get up to a 100% accuracy, in other words a true loss of 0. This means that every number that we could write would be recognized by our algorithm.

What would happen if instead of a number we tried to predict a letter, let's say an L? We could argue that the algorithm would predict that it is a 1, as they have similar shapes.

We observe that even with a perfect algorithm we are not able to predict this instance correctly. This is because the algorithm has been trained to detect only numbers, not letter, indeed, the letter L is a novelty and trying to detect it is known as novelty detection (see [6]). More generally we want to know if it would be possible for an algorithm to learn to detect when we are making predictions on instances out of the distribution of the label's training set.

This is known as out of distribution detection (OOD detection) and it has a huge impact on nowadays learning requirements for algorithms. It can be applied to medicine to detect new diseases, in automotive so autonomous cars can react in situations they haven't been trained for, or even in cybersecurity to detect some suspicious behaviours in a user or software.

The term of out of distribution detection appeared for the first time in 2017 and since then has received increasing attention from the research community, leading to plenty of methods developed, ranging from classification-based to density-based to distance-based ones.

Since this topic is extremely vast and still being researched, we will only be able to introduce it and see some results using the knowledge from the previous chapters. Certain proofs are rather extensive, for this reason we won't see them

is this work. Nonetheless we will define the key concepts in order to formalize whether out of distribution detection is learnable or not, and we will prove a few results. We will try to find conditions that are necessary (and hopefully sufficient) for learnability of the OOD detection.

3.1 Redefining the key concepts

As we may expect, some key concepts introduced in the previous chapter have to be revised as not all of them can be used in the same way. The most obvious one is the label space. We know that we don't necessarily want to classify OOD data into the correct OOD classes, but we still want to detect it as OOD data. For this reason and without loss of generality, let all OOD data be allocated into one big OOD class.

Definition 3.1. *Let $\mathcal{Y} = \{1, \dots, K\}$ be the labels space of the training data. If $\mathcal{Y}_O = \{K+1\}$ is the class of OOD data, we can redefine the **labeling space** as $\mathcal{Y}_{all} = \mathcal{Y} \cup \mathcal{Y}_O$. The **feature space** remains the same as before $\mathcal{X} \subset \mathbb{R}^d$.*

We also have to redefine the concept of distribution, as now, the training and testing data are from different distributions.

Definition 3.2. *We define the **in distribution (ID) joint distribution** $\mathcal{D}_{X_I Y_I}$ over $\mathcal{X} \times \mathcal{Y}$, where $X_I \in \mathcal{X}$ and $Y_I \in \mathcal{Y}$ are random variables. We also define the **out of distribution (OOD) joint distribution** $\mathcal{D}_{X_O Y_O}$, where X_O is a random variable from $\mathcal{X}$, and Y_O is a random variable whose outputs do not belong to $\mathcal{Y}$. We can now define the **mixture distribution** as $\mathcal{D}_{XY} = (1 - \pi^{out})\mathcal{D}_{X_I Y_I} + \pi^{out}\mathcal{D}_{X_O Y_O}$ where π^{out} is the **unknown class-prior probability**.*

We also have to redefine the hypothesis class, as the label space is different from the previous chapter.

Definition 3.3. *We define the **hypothesis space** as $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}_{all}\}$. We also set $\mathcal{H}^{in} = \{h : \mathcal{X} \to \mathcal{Y}\}$ as the **ID hypothesis space**. Also $\mathcal{H}^b = \{h : \mathcal{X} \to \{1, 2\}\}$ will be the **hypothesis space for binary classification**, where 1 represents the ID data, and 2 represents the OOD data.*

The definition of loss and risk function is pretty similar

Definition 3.4. *Given a **loss function** $l : \mathcal{Y}_{all} \times \mathcal{Y}_{all} \to \mathbb{R}_{\geq 0}$ satisfying that $l(y_1, y_2) = 0 \iff y_1 = y_2$ and any $h \in \mathcal{H}$, then the **risk function** with respect to $\mathcal{D}_{XY}$ is:*

$$R_{\mathcal{D}}(h) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{XY}} l(h(x), y)$$

Similarly the α***-risk function*** *is defined as*

$$R^{\alpha}_{\mathcal{D}} = (1-\alpha)R^{in}_{\mathcal{D}}(h) + \alpha R^{out}_{\mathcal{D}}(h), \forall \alpha \in [0,1]$$

where

$$R^{in}_{\mathcal{D}}(h) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{X_I Y_I}} l(h(x),y), \quad R^{out}_{\mathcal{D}}(h) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{X_O}} l(h(x),K+1)$$

As we can observe, all the definition are a simple extension of the ones we saw in the previous chapter, adding the OOD data that we could encounter with a certain unknown probability (π^{out}).

We can now proceed onto defining the key concepts of this chapter. First we will define formally what an algorithm is to be able to then define the learnability of OOD detection.

Definition 3.5. *an* ***algorithm*** *is a mapping from* $\cup_{n=1}^{+\infty}(\mathcal{X}\times\mathcal{Y})^n$ *to* $\mathcal{H}$.

Definition 3.6. *Given a domain space* $\mathcal{D}_{XY}$ *and a hypothesis space* $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}_{all}\}$, *we say that* ***OOD detection is learnable*** *in* $\mathcal{D}_{XY}$ *for* $\mathcal{H}$ *if there exists an algorithm* $A : \cup_{n=1}^{+\infty}(\mathcal{X}\times\mathcal{Y})^n \to \mathcal{H}$ *and a monotonically decreasing sequence* $\epsilon_{cons}(n)$, *such that* $\epsilon_{cons}(n) \underset{n\to+\infty}{\to} 0$ *and for any domain* $\mathcal{D}_{XY}$,

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^n_{X_I Y_I}} [R_{\mathcal{D}}(A(S)) - \underset{h\in\mathcal{H}}{inf}(R_{\mathcal{D}}(h))] \leq \epsilon_{cont}(n)$$

An algorithm A for which this holds is said to be consistent with respect to $\mathcal{D}_{XY}$

3.2 Relation with PAC learning

We will now see our first result, which is going to help us link this chapter to the previous one. We will see that the previous definition is a natural extension of agnostic PAC learning when l is bounded.

Corollary 3.7. *Given a domain space* $\mathcal{D}_{XY}$ *and a hypothesis space* $\mathcal{H}$, *out of distribution detection is learnable in* $\mathcal{D}_{XY}$ *for* $\mathcal{H}$ *if and only if H is PAC learnable in* $\mathcal{D}_{XY}$.

Proof. We will prove in the first place that OOD detection learnability implies PAC learnability.

According to definition (3.6)

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^n_{X_I Y_I}} [R_{\mathcal{D}}(A(S)) - \underset{h\in\mathcal{H}}{inf}(R_{\mathcal{D}}(h))] \leq \epsilon_{cont}(n)$$

We will use the Markov's inequality which tells us:

Lemma 3.8. *(Markov's Inequality) If Z is a non-negative random variable, we have*

$$\forall a \geq 0, \quad \mathbb{P}[Z \geq a] \leq \frac{\mathbb{E}[\mathbb{Z}]}{a}$$

Because $\mathbb{E}[R_{\mathcal{D}}(A(S)) - \underset{h\in\mathcal{H}}{inf}(R_{\mathcal{D}}(h)) \geq 0$ we can use this inequality and have,

$$\begin{aligned}\mathbb{P}[R_{\mathcal{D}}(A(S)) - inf_{h\in\mathcal{H}} R_{\mathcal{D}}(h) < \epsilon] &> 1 - \underset{S\sim\mathcal{D}^n_{X_I Y_I}}{\mathbb{E}}[R_{\mathcal{D}}(A(S)) - \underset{h\in\mathcal{H}}{inf}(R_{\mathcal{D}}(h))]/\epsilon \\ &\geq 1 - \epsilon_{cons}(n)/\epsilon\end{aligned}$$

Because $\epsilon_{cons}(n)$ is monotonically decreasing, we can find a smallest m such that $\epsilon_{cons} \geq \epsilon\delta$ and $\epsilon_{cons}(m-1) > \epsilon\delta$, for $\delta \in (0,1)$. We define that $m = m(\epsilon,\delta)$. Therefore, for any $\epsilon > 0$ and $\delta \in (0,1)$, there exists a function $m(\epsilon,\delta)$, such that when $n > m(\epsilon,\delta)$, with the probability at least $1-\delta$, we have

$$R_{\mathcal{D}}(A(S)) - \underset{h\in\mathcal{H}}{inf} R_{\mathcal{D}}(h) > \epsilon$$

which is the definition of PAC learnability.

Now we will proceed onto proving that PAC learnability implies OOD detection learnability.

From the definition of PAC learnability we know that for $\epsilon > 0$ and $\delta \in (0,1)$, there exists a function $m(\epsilon,\delta)$ such that when the sample size $n > m(\epsilon,\delta)$, we have that with probability at least $1-\delta$,

$$R_{\mathcal{D}}(A(S)) - \underset{h\in\mathcal{H}}{inf} R_{\mathcal{D}}(h) \leq \epsilon$$

Note that the loss function l has upper bound because $\mathcal{Y}_{all}$ is finite, we assume that the upper bound of l is M. Hence according to the definition of PAC learnability, when the sample size $n > m(\epsilon,\delta)$, we have that

$$\mathbb{E}_S[R_{\mathcal{D}}(A(S)) - \underset{h\in\mathcal{H}}{inf} R_{\mathcal{D}}(h)] \leq \epsilon(1-\delta) + 2M\delta < \epsilon + 2M\delta$$

If we set $\delta = \epsilon$, then when the sample size $n > m(\epsilon,\epsilon)$, we have that

$$\mathbb{E}_S[R_{\mathcal{D}}(A(S)) - \underset{h\in\mathcal{D}}{inf} R_{\mathcal{D}}(h)] < (2M+1)\epsilon$$

This implies that

$$\underset{n\to+\infty}{lim}\, \mathbb{E}_S[R_{\mathcal{D}}(A(S)) - \underset{h\in\mathcal{D}}{inf} R_{\mathcal{D}}(h)] = 0,$$

which implies the OOD detection learnability. □

Since OOD data is not available during the training phase, it is impossible to know the class-prior probability, indeed, in the real world, π^{out} can be any value in $[0,1)$. Therefore, the imbalance issue between ID and OOD data, and the prior-unknown issue are the core challenges. To eliminate this issue we can revise the equation from the definition (3.6)

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^n_{X_IY_I}}[R^{\alpha}_{\mathcal{D}}(A(S)) - \mathop{inf}_{h\in\mathcal{H}}(R^{\alpha}_{\mathcal{D}}(h))] \leq \epsilon_{cont}(n), \quad \forall\alpha\in[0,1]$$

If an algorithm A satisfies this equation, then the imbalance issue and the prior-unknown issue disappear. That is, A can simultaneously classify ID data and detect OOD data. Based on this, we can define the strong learnability of OOD detection.

Definition 3.9. *Given a domain space $\mathcal{D}_{XY}$ and a hypothesis space $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}_{all}\}$, we say that **OOD detection is strongly learnable** in $\mathcal{D}_{XY}$ for $\mathcal{H}$ if there exists an algorithm $A : \cup_{n=1}^{+\infty}(\mathcal{X}\times\mathcal{Y})^n \to \mathcal{H}$ and a monotonically decreasing sequence $\epsilon_{cons}(n)$, such that $\epsilon_{cons}(n) \underset{n\to+\infty}{\to} 0$ and for any domain $\mathcal{D}_{XY}$,*

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^n_{X_IY_I}}[R^{\alpha}_{\mathcal{D}}(A(S)) - \mathop{inf}_{h\in\mathcal{H}}(R^{\alpha}_{\mathcal{D}}(h))] \leq \epsilon_{cont}(n)$$

An algorithm A for which this holds is said to be consistent with respect to $\mathcal{D}_{XY}$

Recall the the learnability of supervised learning depends only on the hypothesis space, as we have seen in the previous chapter the learnability depends only on the finitness of the VC-dimension, which is calculated based on properties of the hypothesis class. In other words, PAC learnability was distribution free, i.e., the set of domains was the set of all domains. However, due to the absence of OOD data, during the training process, it is obvious that the learnability of OOD detection is not distribution free. The goal of the theory is now pretty clear:

Goal: given a hypothesis space $\mathcal{H}$ and several representative domain spaces $\mathcal{D}_{XY}$, what are the **conditions** to ensure the learnability of OOD detection? Are these conditions sufficient in some scenarios?

3.3 Learning in prior-unknown spaces

Definition 3.10. *If $\mathbb{D}_{XY}$ is the set of all possible domains, we say that $\mathbb{D}_{XY}$ is a priori-unknown space, if for any domain $\mathcal{D}_{XY} \in \mathbb{D}_{XY}$ and any $\alpha \in [0,1)$, we have $\mathcal{D}^{\alpha}_{XY} = (1-\alpha)\mathcal{D}_{X_IY_I} + \alpha\mathcal{D}_{X_OY_O} \in \mathbb{D}_{XY}$.*

Theorem 3.11. *Given a domain space* $\mathbb{D}_{XY}$ *and* $\mathbb{D}'_{XY} = \{\mathcal{D}^{\alpha}_{XY} : \forall \mathcal{D}_{XY} \in \mathbb{D}_{XY}, \forall \alpha \in [0,1)\}$, *then*

1. $\mathbb{D}'_{XY}$ *is a priori-unknown space and* $\mathbb{D}_{XY} \subset \mathbb{D}'_{XY}$
2. *if* $\mathbb{D}_{XY}$ *is a priori- unknown space, then definition (3.6) and definition (3.9) are equivalent.*
3. *OOD detection is strongly learnable in* $\mathbb{D}_{XY}$ *if and only if OOD detection is learnable in* $\mathbb{D}'_{XD}$

We won't demonstrate in detail this result but we can find the proof in [4] Appendix E.

3.4 Impossibility Theorems for OOD detection

We will now state a condition that determines whether OOD detection is learnable or not.

Condition 1 (Linear condition). For any $\mathcal{D}_{XY} \in \mathbb{D}_{XY}$ and any $\alpha \in [0,1)$

$$\inf_{h\in\mathcal{H}} R^{\alpha}_{\mathcal{D}}(h) = (1-\alpha)\inf_{h\in\mathcal{H}} R^{in}_{\mathcal{D}} + \alpha \inf_{h\in\mathcal{H}} R^{out}_{\mathcal{D}}(h).$$

Theorem 3.12. *Given a hypothesis set* $\mathcal{H}$ *and a domain* $\mathcal{D}_{XY}$, *OOD detection is learnable in the single-distribution space* $\mathbb{D}^{\mathcal{D}_{XY}}_{XY} = \{\mathcal{D}^{\alpha}_{XY} : \forall \alpha \in [0,1)\}$ *for* $\mathcal{H}$ *if and only if the linear condition holds.*

This result tells us that Condition 1 is important for learnability of OOD detection. Due to the simplicity of single-distribution spaces, Theorem (3.12) implies that Condition 1 is necessary for learnability. The proof of this Theorem is rather extensive and requires previous definitions and results, for this reason we won't see it here, but it can be found in [4] Appendix F. We will now see another result that shows that Condition 1 is not sufficient, especially, when there is an *overlap* between ID and OOD distributions.

Definition 3.13. *We say that the domain* $\mathcal{D}_{XY}$ *has* ***overlap*** *between ID and OOD distributions if there is a* σ*-finite measure* $\tilde{\mu}$ *such that* $\mathcal{D}_X$ *is absolutely continuous with respect to* $\tilde{\mu}$ *and* $\tilde{\mu}(A_{overlap}) > 0$, *where* $A_{overlap} = \{x \in \mathcal{X} : f_I(x) > 0 \text{ and } f_O(x) > 0\}$. *Here* f_I *and* f_O *are the representers of* D_{X_I} *and* D_{X_O} *such that*

$$D_{X_I} = \int f_I d\tilde{\mu} \qquad D_{X_O} = \int f_O d\tilde{\mu}$$

Theorem 3.14. *Given a hypothesis space $\mathcal{H}$ and a prior-unknown space $\mathbb{D}_{XY}$, if there is $\mathcal{D}_{XY} \in \mathbb{D}_{XY}$, which has overlap between ID and OOD, and $inf_{h\in\mathcal{H}} R^{in}_{\mathcal{D}} = 0$ and $inf_{h\in\mathcal{H}} R^{out}_{\mathcal{D}} = 0$, then Condition 1 does not hold. Therefore, OOD detection is not learnable in $\mathbb{D}_{XY}$ for $\mathcal{H}$.*

Proof. We will first explain how we get f_I and f_O. Since $\mathcal{D}_X$ is absolutely continuous respect to $\mu(\mathcal{D}_X << \mu)$, then $\mathcal{D}_{X_I} << \mu$ and $\mathcal{D}_{X_O}$. By Radon-Nikodym Theorem (see [5]), we know there exist two non-negative functions defined over $\mathcal{X}$: f_I and f_O such that for any μ-measurable set $A \subset \mathcal{X}$

$$\mathcal{D}_{X_I}(A) = \int_A f_I(x)d\mu(x), \qquad \mathcal{D}_{X_O}(A) = \int_A f_O(x)d\mu(x)$$

Secondly, we prove that for any $\alpha \in (0,1)$, $inf_{h\in\mathcal{H}} D^{\alpha}_{\mathcal{D}}(h) > 0$
We define $A_m = \{x \in \mathcal{X} : f_I(x) \geq \frac{1}{m} \text{ and } f_O(x) \geq \frac{1}{m}\}$. It is clear that

$$\cup_{m=1}^{+\infty} A_m = \{x \in \mathcal{X} : f_I(x) > 0 \text{ and } f_O(x) > 0\} = A_{overlap}$$

and

$$A_m \subset A_{m+1}$$

Therefore,

$$\lim_{m\to+\infty} \mu(A_m) = \mu(A_{overlap}) > 0$$

which implies that there exists m_0 such that

$$\mu(A_{m_0}) > 0$$

For any $\alpha \in (0,1)$, we define

$$c_\alpha = min_{y_1\in\mathcal{Y}_{all}}((1-\alpha)min_{y_2\in\mathcal{Y}} l(y_1,y_2) + \alpha l(y_1, K+1)).$$

It is clear that $c_\alpha > 0$ for $\alpha \in (0,1)$. Then, for any $h \in \mathcal{H}$,

$$\begin{aligned}
R^\alpha_{\mathcal{D}}(h) &= \int_{\mathcal{X}\times\mathcal{Y}_{all}} l(h(x),y)d\mathcal{D}^\alpha_{XY}(x,y) \\
&= \int_{\mathcal{X}\times\mathcal{Y}} (1-\alpha) l(h(x),y)d\mathcal{D}_{\mathcal{X}_\mathcal{I}\mathcal{Y}_\mathcal{I}}(x,y) + \int_{\mathcal{X}\times\{K+1\}} \alpha l(h(x),y)d\mathcal{D}_{X_O Y_O}(x,y) \\
&\geq \int_{A_{m_0}\times\mathcal{Y}} (1-\alpha) l(h(x),y)d\mathcal{D}_{\mathcal{X}_\mathcal{I}\mathcal{Y}_\mathcal{I}}(x,y) + \int_{A_{m_0}\times\{K+1\}} \alpha l(h(x),y)d\mathcal{D}_{X_O Y_O}(x,y) \\
&= \int_{A_{m_0}} ((1-\alpha)\int_{\mathcal{Y}} l(h(x),y)d\mathcal{D}_{Y_I|X_I}(y|x))d\mathcal{D}_{X_I}(x) + \int_{A_{m_0}} \alpha l(h(x),K+1)d\mathcal{D}_{X_O}(x) \\
&\geq \int_{A_{m_0}} (1-\alpha) \min_{y_2\in\mathcal{Y}} \quad l(h(x),y_2)d\mathcal{D}_{X_I}(x) + \int_{A_{m_0}} \alpha l(h(x),K+1)d\mathcal{D}_{X_O}(x) \\
&\geq \int_{A_{m_0}} (1-\alpha) \min_{y_2\in\mathcal{Y}} \quad l(h(x),y_2)f_I(x)d\mu(x) + \int_{A_{m_0}} \alpha l(h(x),K+1)f_O(x)d\mu(x) \\
&\geq \frac{1}{m_0}\int_{A_{m_0}} (1-\alpha) \min_{y_2\in\mathcal{Y}} \quad l(h(x),y_2)d\mu(x) + \frac{1}{m_0}\int_{A_{m_0}} \alpha l(h(x),K+1)d\mu(x) \\
&\geq \frac{1}{m_0}\int_{A_{m_0}} (1-\alpha) \min_{y_2\in\mathcal{Y}} \quad l(h(x),y_2) + \alpha l(h(x),K+1)d\mu(x) \\
&\geq \frac{c_\alpha}{m_0}\mu(A_{m_0}) > 0
\end{aligned} \tag{3.1}$$

Therefore,

$$\inf_{h\in\mathcal{H}} R^\alpha_{\mathcal{D}}(c) \geq \frac{c_\alpha}{m_0}\mu(A_{m_0}) > 0$$

Condition 1 indicates that

$$\inf_{h\in\mathcal{H}} R^\alpha_{\mathcal{D}}(h) = (1-\alpha)\inf_{h\in\mathcal{H}} R^{in}_{\mathcal{D}} + \alpha \inf_{h\in\mathcal{H}} R^{out}_{\mathcal{D}}(h) = 0$$

which contradicts what we have just seen. Therefore, Condition 1 does not hold, we obtain with this that OOD detection is not learnable for $\mathcal{H}$, which completes the proof. □

This Theorem shows us that under proper conditions, Condition 1 does not hold, if there exists a domain whose ID and OOD distribution have an overlap. Therefore OOD detection is not learnable. A similar result can be proved for the *total space* for any non trivial hypothesis space.

Definition 3.15. *The **total space** $\mathbb{D}^{all}_{XY}$ is the set of all possible domains.*

Theorem 3.16. *OOD detection is not learnable in the total space $\mathbb{D}^{all}_{XY}$ for $\mathcal{H}$, if $|\phi \circ \mathcal{H}| > 1$, where ϕ maps ID labels to 1 and maps OOD labels to 2.*

Proof. We need to prove that OOD detection is not learnable in the total space $\mathbb{D}_{XY}^{all}$ for $\mathcal{H}$, if $\mathcal{H}$ is non-trivial, *i.e.*, $\{x \in \mathcal{X} : \exists h_1, h_2 \in \mathcal{H}, s.t, h_1(x) \in \mathcal{Y}, h_2(x) = K+1\} \neq 0$.

The main idea is to construct a domain $\mathcal{D}_{XY}$ satisfying that:

1. the ID and OOD distribution have an overlap
2. $R_{\mathcal{D}}^{in}(h_1) = 0, R_{\mathcal{D}}^{out}(h_2) = 0$

According to the condition that $\mathcal{H}$ is non-trivial, we know that there exists $h_1, h_2 \in \mathcal{H}$ such that $h_1(x_1) \in \mathcal{Y}, h_2(x_1) = K+1$, for some $x_1 \in \mathcal{X}$. We set $\mathcal{D}_{XY} = 0.5 * \delta_{\{x_1, h_1(x_1)\}} + 0.5 * \delta_{\{x_1, h_2(x_1)\}}$, where δ is the Dirac measure. It is easy to check that $R_{\mathcal{D}}^{in}(h_1) = 0, R_{\mathcal{D}}^{out}(h_2) = 0$, which implies that $inf_{h \in \mathcal{H}} R_{\mathcal{D}}^{in}(h) = 0$ and $inf_{h \in \mathcal{H}} R_{\mathcal{D}}^{out}(h) = 0$. In addition, the ID distribution $\delta_{\{x_1, h_1(x_1)\}}$ and the OOD distribution $\delta_{\{x_1, h_2(x_1)\}}$ has an overlap x_1. By using the Theorem (3.14) we complete the proof. □

Since the overlaps in the ID and OOD distributions may cause that Condition 1 does not hold we can now try to consider studying the learnability of OOD detection in *separate spaces* where there are no overlaps between ID and OOD distributions.

Definition 3.17. *The **separate space** $\mathbb{D}_{XY}^{s}$ is the set of the domains that satisfy the separate condition, that is for any $\mathcal{D}_{XY} \in \mathbb{D}_{XY}^{s}$, $supp(\mathcal{D}_{X_O}) \cap supp(\mathcal{D}_{X_I}) \neq \varnothing$, $supp$ is the support space defined as $supp(f) = \{x \in \mathcal{X} : f(x) \neq 0\}$*

Definition 3.18. *We say that a hypothesis space $\mathcal{H}$ is **separate for OOD**, if for each data point $x \in \mathcal{X}$, there exists at least one hypothesis function $h_x \in \mathcal{H}$ such that $h_x(x) = K+1$.*

With this definition we can state the last Theorem of impossibility of the learnability of OOD detection.

Theorem 3.19. *If a hypothesis set $\mathcal{H}$ is separate for OOD, $VCdim(\phi \circ \mathcal{H}) < +\infty$ and $sup_{h \in \mathcal{H}} |\{x \in \mathcal{X} : h(x) = y\}| = +\infty$. Then OOD detection is not learnable in separate space $\mathbb{D}_{XY}$ for $\mathcal{H}$, where ϕ maps ID data to 1, and maps OOD data to 2.*

As we saw in the previous chapter, the finiteness of VC-dimension should imply learnability of OOD, but this Theorem states that in this particular case, it cannot guarantee the learnability as we are in separate spaces. The proof of this result can be found in [4] Appendix H. We will now see some scenarios where the learnability of OOD detection is possible.

3.5 Possibility Theorems of OOD detection

In this section we will try to find some conditions where OOD detection can be learnable, we will start where we left it in the previous section: in the case of separate spaces.

The last stated Theorem shows us conditions under which OOD detection is not learnable in separate spaces. Formally, we say that $VCdim(\phi \circ \mathcal{H}) < +\infty$ and $sup_{h\in\mathcal{H}}|\{x \in \mathcal{X} : h(x) = y\}| = +\infty$ are necessary conditions so that OOD detection is not learnable, we can ask ourselves what would happen if we remove one of the conditions.

Generally, hypothesis spaces generated in practice have finite VC-dimension, therefore we will study the case $|\mathcal{X}| < +\infty$ which implies that $sup_{h\in\mathcal{H}}|\{x \in \mathcal{X} : h(x) = y\}| = +\infty$.

For simplicity we will consider the special case where $K = 1$. This is also known as the one class novelty detection, as our learning distribution consists of only one class, and we are trying to learn to detect any new class. We will show the necessary and sufficient condition for learnability of OOD detection in $\mathbb{D}^s_{XY}$, when $|\mathcal{X}| < +\infty$.

Theorem 3.20. *Let $K = 1$ and $|\mathcal{X}| < +\infty$. Also let $\mathcal{H}$ be a hypothesis class which is separate for OOD data and the constant $h^{in} := 1 \in \mathcal{H}$. Then OOD detection is learnable in $\mathbb{D}^s_{XY}$ for $\mathcal{H}$ if and only if $\mathcal{H}^{all} - \{h^{out}\} \subset \mathcal{H}$, where $\mathcal{H}^{all}$ is the set of all hypothesis functions and h^{out} is a constant function $h^{out} := 2$, here 1 represents ID data and 2 represents OOD data.*

The proof of this result is also pretty long and requires some previous results and unseen definitions, but the proof can be found in [4] Appendix I. We will state and prove an extension of this result to a more general case, for $K > 1$. We will use a binary classifier h^b to classify ID and OOD data. Then another classifier can be used to classify the ID data in each ID class.

For that purpose we will construct a hypothesis class $\mathcal{H}$ composed of $\mathcal{H}^{in}$, the hypothesis class for ID data, and $\mathcal{H}^b$, the hypothesis class for binary classification for ID or OOD data. The hypothesis in $\mathcal{H}$ will have the form of

$$h(x) = \begin{cases} i & \text{if } h^{in}(x) = i \text{ and } h^b(x) = 1 \\ K+1 & \text{otherwise} \end{cases}$$

Let's first state a necessary condition first.

Condition 2: $l(y_2, y_1) \leq l(K+1, y_1)$, for any ID labels y_1 and $y_2 \in \mathcal{Y}$.

Theorem 3.21. *Let $|\mathcal{X}| < +\infty$ and $\mathcal{H} = \mathcal{H}^{in} \bullet \mathcal{H}^{b}$. If $\mathcal{H}^{all} - \{h^{out}\} \subset \mathcal{H}^{b}$ and Condition 2 holds, then OOD detection is learnable in $\mathbb{D}^{s}_{XY}$ for $\mathcal{H}$, where $\mathcal{H}^{all}$ and h^{out} are defined as in the previous Theorem.*

Proof. Since $|\mathcal{X}| < +\infty$ we know that $|\mathcal{H}| < +\infty$. This implies that $\mathcal{H}^{in}$ is agnostic PAC learnable for supervised learning. Therefore, there exists an algorithm $A^{in} : \cup_{n=1}^{+\infty}(\mathcal{X} \times \mathcal{Y})^n \to \mathcal{H}^{in}$ and a monotonically decreasing sequence $\epsilon(n) \underset{n\to+\infty}{\to} 0$ and for any $\mathcal{D}_{XY} \in \mathbb{D}^{s}_{XY}$

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^{n}_{X_IY_I}} R^{in}_{\mathcal{D}}(A^{in}(S)) \leq \inf_{h\in\mathcal{H}^{in}} R^{in}_{\mathcal{D}}(h) + \epsilon(n) \tag{3.2}$$

Since $|\mathcal{X}| < +\infty$ and $\mathcal{H}^{b}$ almost contains all binary classifiers, then using Theorem (3.20) and Theorem (3.11), we obtain that there exists an algorithm $A^{b} : \cup_{n=1}^{+\infty}(\mathcal{X} \times \{1,2\})^n \to \mathcal{H}^{b}$ and a monotonically decreasing sequence $\epsilon'(n) \underset{n\to+\infty}{\to} 0$ and for any $\mathcal{D}_{XY} \in \mathbb{D}^{s}_{XY}$

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^{n}_{X_IY_I}} R^{in}_{\phi(\mathcal{D})}(A^{b}(\phi(S))) \leq \inf_{h\in\mathcal{H}^{b}} R^{in}_{\phi(\mathcal{D})}(h) + \epsilon'(n) \tag{3.3}$$

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^{n}_{X_IY_I}} R^{out}_{\phi(\mathcal{D})}(A^{b}(\phi(S))) \leq \inf_{h\in\mathcal{H}^{b}} R^{out}_{\phi(\mathcal{D})}(h) + \epsilon'(n), \tag{3.4}$$

where ϕ maps ID data to 1 and OOD data to 2.

$$R^{in}_{\phi(\mathcal{D})}(A^{b}(\phi(S))) = \int_{\mathcal{X}\times\mathcal{Y}} l(A^{b}(\phi(S))(x), \phi(y)) d\mathcal{D}_{X_IY_I}(x,y) \tag{3.5}$$

$$R^{in}_{\phi(\mathcal{D})}(h) = \int_{\mathcal{X}\times\mathcal{Y}} l(h(x), \phi(y)) d\mathcal{D}_{X_IY_I}(x,y) \tag{3.6}$$

$$R^{out}_{\phi(\mathcal{D})}(A^{b}(\phi(S))) = \int_{\mathcal{X}\times\{K+1\}} l(A^{b}(\phi(S))(x), \phi(y)) d\mathcal{D}_{X_OY_O}(x,y) \tag{3.7}$$

$$R^{out}_{\phi(\mathcal{D})}(h) = \int_{\mathcal{X}\times\{K+1\}} l(h(x), \phi(y)) d\mathcal{D}_{X_OY_O}(x,y), \tag{3.8}$$

here $\phi(S) = \{(x_1, \phi(y_1) \ldots, (x_n, \phi(y_n))\}$ if $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

Note that $\mathcal{H}^{b}$ almost contains all classifiers, and $\mathbf{D}^{s}_{XY}$ is the separate space. Hence,

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^{n}_{X_IY_I}} R^{in}_{\phi(\mathcal{D})}(A^{b}(\phi(S))) \leq \epsilon'(n), \quad \mathop{\mathbb{E}}_{S\sim\mathcal{D}^{n}_{X_IY_I}} R^{out}_{\phi(\mathcal{D})}(A^{b}(\phi(S))) \leq \epsilon'(n) \tag{3.9}$$

Next, let's construct an algorithm A using A^{in} and A^{b}.

$$A(S)(x) = \begin{cases} K+1, & \text{if } A^b(\phi(S))(x) = 2 \\ A^{in}(S)(x) & \text{if } A^b(\phi(S))(x) = 1 \end{cases}$$

Since $\inf_{h\in\mathcal{H}} R^{in}_{\phi(\mathcal{D})}(\phi \circ h) = \inf_{h\in\mathcal{H}} R^{out}_{\phi(\mathcal{D})}(\phi \circ h) = 0$, we can use Condition 2 to check that

$$\inf_{h\in\mathcal{H}^{in}} R^{in}_{\mathcal{D}}(h) = \inf_{h\in\mathcal{H}} R^{in}_{\mathcal{D}}(h)$$

Additionally, the risk function $R^{in}_{\mathcal{D}}(A(S))$ is from two parts: one part is about ID data detected as OOD data and the other part about ID data detected as ID data, but classified in the wrong class. Therefore, we have the inequality

$$\begin{aligned} \mathop{\mathbb{E}}_{S\sim\mathcal{D}^n_{X_IY_I}} R^{in}_{\mathcal{D}}(A(S)) &\leq \mathop{\mathbb{E}}_{S\sim\mathcal{D}^n_{X_IY_I}} R^{in}_{\mathcal{D}}(A(S)) + c \mathop{\mathbb{E}}_{S\sim\mathcal{D}^n_{X_IY_I}} R^{in}_{\phi(\mathcal{D})}(A^b(\phi(S))) \\ &\leq \inf_{h\in\mathcal{H}^{in}} R^{in}_{\mathcal{D}}(h) + \epsilon(n) + c\epsilon'(n) \\ &= \inf_{h\in\mathcal{H}} R^{in}_{\mathcal{D}}(h) + \epsilon(n) + c\epsilon'(n), \end{aligned} \tag{3.10}$$

where $c = \max_{y_1,y_2\in\mathcal{Y}} l(y_1,y_2)/min(l(1,2),l(2,1))$

Note that the risk $R^{out}(A(S))$ is about OOD being detected as ID data. Therefore,

$$\begin{aligned} \mathop{\mathbb{E}}_{S\sim\mathcal{D}^n_{X_IY_I}} R^{out}_{\mathcal{D}}(A(S)) &\leq c \mathop{\mathbb{E}}_{S\sim\mathcal{D}^n_{X_IY_I}} R^{out}_{\phi(\mathcal{D})}(A^b(\phi(S))) \\ &\leq c\epsilon'(n) \\ &\leq \inf_{h\in\mathcal{H}} R^{out}_{\mathcal{D}}(h) + c\epsilon'(n) \end{aligned} \tag{3.11}$$

Note that $(1-\alpha)\inf_{h\in\mathcal{H}} R^{in}_{\mathcal{D}}(h) + \alpha \inf_{h\in\mathcal{H}} R^{in}_{\mathcal{D}}(h) \leq \inf_{h\in\mathcal{H}} R^{\alpha}_{\mathcal{D}}(h)$. Then, using the equation (3.10) and the equation (3.11), we obtain that for any $\alpha \in [0,1]$

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^n_{X_IY_I}} R^{\alpha}_{\mathcal{D}}(A(S)) \leq \inf_{h\in\mathcal{H}} R^{\alpha}_{\mathcal{D}}(h) + \epsilon(n) + c\epsilon'(n)$$

According to the second result of theorem (3.11), we complete the proof. □

3.6 Conclusion and continuation

We have introduced in this chapter the concept of out of distribution detection, which is, in other words, a generalization of learning theory, as we are assuming that the training data and the testing data may differ in a more practical scenario.

We may wish to find a Fundamental Theorem that would state some necessary and sufficient conditions for learnability of out of distribution detection. But as this topic is recent and still being investigated such result still doesn't exist.

Nowadays, researchers can estimate the performance of out of distribution detection by using functions like AUROC (Area Under Receiving Operator Characteristic), AUPR (Area Under Precision Recall) and FPR95 (False Positive Rate at 95%).

Also some other results exist in other scenarios. For example it would be worth investigating the conditions and efficiency of OOD detection where the ID space is finite. Moreover this is a more realistic scenario, as in practice we will only be able to collect a finite ID dataset.

Some other spaces can be studied, like density-based spaces $\mathbb{D}_{XY}^{\mu,b}$, which are prior-unknown spaces consisting of some domains satisfying that: for any $\mathcal{D}_{XY}$, there exists a density function f with

$$\frac{1}{b} \leq f \leq b \text{ in } supp(\mu) \text{ and } 0.5 \cdot \mathcal{D}_{X_I} + 0.5 \cdot \mathcal{D}_{X_O} = \int f d\mu,$$

where μ is a measure defined over $\mathcal{X}$.

In any case, I would definitely keep an eye on this topic as it will evolve quickly in the next years and will have a huge impact on Learning Theory and Machine Learning.

Bibliography

[1] L.G. Valiant, *A theory of the learnable*, David Waltz Editor, (1984).

[2] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From theory to algorithms*, Cambridge University, Cambridge University Press, (2014), 19-78.

[3] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, MIT Press, ed. 2, (2018), 1-50.

[4] Z.Fang, Y. Li, J.Lu, J. Dong, B. Han, F. Liu *Is Out-Of-Distribution Detection Learnable*, in NeurIPS, (2022)

[5] Donald L Cohn, *Measure theory*, Springer, (2013)

[6] Jingkang Yang, Kaiyang Zhou, Yixuan Li, Ziwei Liu, *Generalized Out-of-Distribution Detection: A Survey*, (2024)

[7] Michael J. Kearns, Umesh V. Vazirani, *An Introduction to Computational Learning Theory*, The MIT Press, (1994).