

*Master's Thesis*

# MATHEMATICAL FOUNDATIONS OF STATISTICAL LEARNING

---

Yash Jakhmola , 20MS028  
(*BS-MS Dual Degree*)

*Supervised by*

Dr. Anirvan Chakraborty<sup>\*</sup>



*Department of Mathematics and Statistics*

*Indian Institute of Science Education and Research, Kolkata*

August 2024 - May 2025

---

<sup>\*</sup>Associate Professor, Department of Mathematics and Statistics, Indian Institute of Science Education and Research, Kolkata

# ABSTRACT

Statistical learning is the process of finding patterns in datasets to make predictions without being explicitly programmed. This is done by using the given data to find a function that generalizes well to unseen data. Techniques range from simple linear regression to complex neural networks, all aiming to extract meaningful information from data.

This thesis studies the mathematical background required to understand the algorithms and techniques used in practice, as well as the attempts made to understand why certain methods work so well in practice.

We start by defining the problem setup and then move onto the major issue of the field - minimizing the error over unseen datasets. We then explore SVMs and their extensions using kernels. Then, we dive into neural networks - one of the best performing algorithms, study their architecture, practical implementation and variants. We also state and prove their approximation capabilities.


# DECLARATION BY THE STUDENT

Date: 9 May 2025

I, *Mr. Yash Jakhmola*, Registration No. *20MS028*, dated *9 May 2025*, a student of the *Department of Mathematics and Statistics* of the *5 Year BS-MS Dual Degree* programme of IISER Kolkata, hereby declare that this thesis is my own work and, to the best of my knowledge, it neither contains materials previously published or written by any other person, nor has it been submitted for any degree/diploma or any other academic award anywhere before. I have used the originality checking service to prevent inappropriate copying.

I also declare that all copyrighted material incorporated into this thesis is in compliance with the Indian Copyright Act, 1957 (amended in 2012) and that I have received written permission from the copyright owners for my use of their work.

I hereby grant permission to IISER Kolkata to store the thesis in a database which can be accessed by others.

 09/05/2025

---

Yash Jakhmola

Department of Mathematics and Statistics,  
Indian Institute of Science Education and Research Kolkata.  
Mohanpur 741246, West Bengal, India.

# CERTIFICATE FROM THE SUPERVISOR

Date: 9 May 2025

This is to certify that the thesis titled “*Mathematical Foundations of Statistical Learning*” submitted by *Mr. Yash Jakhmola*, Registration No. *20MS028*, dated *9 May 2025*, a student of the *Department of Mathematics and Statistics* of the *5 Year BS-MS Dual Degree* programme of IISER Kolkata, is based upon his/her own research work under my supervision. I also certify, to the best of my knowledge, that neither the thesis nor any part of it has been submitted for any degree/diploma or any other academic award anywhere before. In my opinion, the thesis fulfils the requirement for the award of the degree of *Bachelor of Science and Master of Science*.

 8/5/25

Dr. Anirvan Chakraborty

Associate Professor,

Department of Mathematics and Statistics,

Indian Institute of Science Education and Research Kolkata.

Mohanpur 741246, West Bengal, India.

# ACKNOWLEDGMENTS

I am deeply grateful to my supervisor Dr. Anirvan Chakraborty for making the journey of this thesis truly enjoyable and mathematically enriching, and to my parents for their constant support in all ways possible. I would also like to thank my friend Aakash Kumar for all the interesting discussions we have had.

# CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Binary classification . . . . .	3
1.2	Multi-Class classification . . . . .	5
1.3	Regression . . . . .	6
<b>2</b>	<b>GENERALIZATION BOUNDS</b>	<b>7</b>
2.1	Rademacher Complexity . . . . .	8
2.2	VC Dimension . . . . .	11
2.3	Margin-based bounds . . . . .	14
2.4	Multi-class generalization bounds . . . . .	18
2.5	Regression generalization bounds . . . . .	21
<b>3</b>	<b>SVMS AND KERNELS</b>	<b>25</b>
3.1	Separable Case . . . . .	26
3.1.1	Primal Optimization Problem . . . . .	26
3.1.2	Dual Optimization Problem . . . . .	27
3.1.3	Leave-One-Out analysis . . . . .	31
3.2	Non-separable Case . . . . .	32
3.2.1	Primal Optimization Problem . . . . .	33
3.2.2	Dual Optimization Problem . . . . .	33
3.3	Deriving SVMs from margin bounds . . . . .	36
3.4	Multi-class SVM . . . . .	37
3.5	Support Vector Regression . . . . .	40

3.6	Kernels . . . . .	41
3.6.1	PDS Kernels . . . . .	42
3.6.2	SVM with PDS kernels . . . . .	45
3.6.3	Representer theorem . . . . .	46
<b>4</b>	<b>IMPROVING PERFORMANCE</b>	<b>47</b>
4.1	Cross-Validation . . . . .	48
4.2	Boosting . . . . .	48
4.2.1	An anomaly . . . . .	53
4.2.2	Some remarks . . . . .	57
4.3	Dimensionality Reduction . . . . .	59
4.3.1	Principal Component Analysis . . . . .	59
4.3.2	Johnson-Lindenstrauss lemma . . . . .	60
<b>5</b>	<b>NEURAL NETWORKS</b>	<b>63</b>
5.1	Introduction . . . . .	63
5.1.1	Activation functions . . . . .	66
5.1.2	Training . . . . .	67
5.2	Optimization . . . . .	68
5.2.1	Gradient Descent . . . . .	68
5.2.2	Adam . . . . .	72
5.3	Training . . . . .	73
5.3.1	Backpropagation . . . . .	74
5.3.2	Batch Training . . . . .	76
5.3.3	Weights Initialization . . . . .	78
5.4	Approximation . . . . .	79
5.5	Information Processing . . . . .	92
5.5.1	Lost information and Compressible layers . . . . .	93
5.5.2	Bottleneck Principle . . . . .	94
5.6	Other Architectures . . . . .	95
5.6.1	Convolutional Neural Networks . . . . .	95

5.6.2	Recurrent Neural Networks . . . . .	96
5.6.3	Transformers . . . . .	99
5.6.4	Generative Adversarial Network . . . . .	100
<b>6</b>	<b>CONCLUSION</b>	<b>102</b>
<b>A</b>	<b>CONVEX OPTIMIZATION</b>	<b>103</b>
<b>B</b>	<b>LINEAR ALGEBRA</b>	<b>105</b>
<b>C</b>	<b>ANALYSIS</b>	<b>106</b>
<b>D</b>	<b>PROBABILITY</b>	<b>108</b>
<b>E</b>	<b>FUNCTIONAL ANALYSIS</b>	<b>110</b>



# *Chapter 1*

## INTRODUCTION

Statistical learning is the process of using given data to choose a function that generalizes well to unseen data. Methods of statistical learning work best when there is a lot of high dimensional data and the pattern that needs to be found is very complicated. The abundance of data in today's age is why statistical learning methods are much preferred. Statistical learning has more popularly been a practical field, with most of the research being conducted in the implementation of various algorithms to various use-cases.

This thesis shall give a brief but comprehensive and mathematically rigorous introduction to the field of statistical learning, in the context of machine learning and deep learning. Such mathematical treatment is essential to answering questions of why and how - we know statistical learning works well in practice, but why? And how? Such questions need to be answered to develop our understanding of the algorithms used in practice.

Though statistical learning can include all of statistics, this thesis shall restrict to what has colloquially been called machine learning and deep learning. Machine learning broadly refers to the algorithms and techniques that were created using traditional statistics - their implementation can be little involved in terms of the mathematical background needed. Deep learning refers to the algorithms based upon neural networks, which shall be defined in [Chapter 5](#). They are much simpler,

but are much more robust and perform equally if not better than almost any traditional method, most of the times.

Such power of deep learning based algorithms shook the world, but it is still not completely clear why these methods perform so well. As a result, many sub-fields of mathematical research have emerged, trying to understand deep learning from various perspectives.

This thesis shall start off with laying the basic important concepts from machine learning, following the book by Mohri et al., 2018. The rest of this chapter explains the setup for the three different problems that we will be talking about for the most part of this thesis - binary classification (section 1.1), multi-class classification (section 1.2) and regression (section 1.3).

Chapter 2 talks about bounding the generalization error and margin loss (ie. the ‘test’ error or error on the unseen data) using the empirical error (ie. the ‘train’ error or error on the seen data). Chapter 3 talks about support vector machines - one of the best machine learning algorithms and kernels, one of the most powerful tools for studying and extending various algorithms. Chapter 4 goes into improving performance of a machine learning algorithm using cross-validation, boosting, and dimensionality-reduction.

Then, Chapter 5 goes into deep learning, following the book by Calin, 2020 - talking about some basic concepts, approximation results and delving into some recent research trying to give mathematical justification of the absurd performance of deep learning.

All theorems and proofs, unless stated explicitly, are taken from Mohri et al., 2018 for all chapters except Chapter 5, for which the theorems and proofs are taken from Calin, 2020.

Let’s start rigorously defining the setup for classification and regression. To give basic intuition - labelling images of animals as dog or cat is binary classification, labelling images of animals as dog, cat, cow or tiger is multi-class classification (in particular, 4-class classification) and labelling images of animals as a number in

$[0, 1]$ , where 0 represents wild and 1 represents domestic is regression.

## 1.1 Binary classification

Let  $\mathcal{X} \subseteq \mathbb{R}^N$ , for some  $N \in \mathbb{N}$  be the **input/instance space** and  $\mathcal{Y} := \{-1, +1\}$  be the **output/label space**. Let  $\mathcal{C} := \{f : \mathcal{X} \rightarrow \mathcal{Y} \mid f \text{ is measurable}\}$  be the class of all potential ‘patterns’ between the input and output data, called the **concept class**. We want to find a function that captures the pattern between the input and output data.

We will be given a **sample**  $S := (x_1, \dots, x_m)$  (or  $S := ((x_1, y_1), \dots, (x_m, y_m))$ , depending on context) of size  $m \in \mathbb{N}$ , drawn iid according to a **distribution**  $\mathcal{D}$  over  $\mathcal{X}$ , along with labels for each instance  $(c(x_1), \dots, c(x_m))$ , where  $c \in \mathcal{C}$  is the true function between  $\mathcal{X}$  and  $\mathcal{Y}$  that we wish to find (or approximate).

*Remark 1.1.1.* We shall consider the deterministic case, when  $c(x_i) = y_i$  for all  $i \in [m] := \{1, \dots, m\}$ . Also, the distribution  $\mathcal{D}$  is a pdf over the probability space  $(\mathbb{R}^N, \Omega, \mathbb{P})$  where  $\Omega$  is any  $\sigma$ -algebra over  $\mathbb{R}^N$  and  $\mathbb{P}$  is any associated probability measure.

Now, let the **hypothesis set** be a ‘manageable’ and measurable subset  $\mathcal{H}$  of  $\mathcal{C}$ . We want to find  $h \in \mathcal{H}$  with a low **generalization error**, which is defined as

$$R(h) := \mathbb{P}_{x \sim \mathcal{D}} [h(x) \neq c(x)] = \mathbb{E}_{x \sim \mathcal{D}} [1_{h(x) \neq c(x)}] \quad (1.1)$$

Ideally, we want to find  $c \in \mathcal{C}$  that minimizes  $R$ . However, that is not practically feasible - we cannot minimize generalization error over all measurable functions. Moreover, doing so would mean that we would be finding a function that not only captures the patterns in the data, but also the noise. This is called overfitting, and will be explained in [Chapter 2](#). Thus, we find a ‘simpler’ class of functions  $\mathcal{H}$  and minimize  $R$  over that.

Almost all of our analysis will involve trying to bound this error, in hopes that

the bound gives us a good idea of how good can the error be for our particular situation. Since we know neither  $\mathcal{D}$  nor the value of  $c$  for points not in the sample, we will use an empirical version of this error (**empirical error**) to find bounds on  $R(h)$  for all  $h \in \mathcal{H}$ , which is defined as

$$\hat{R}_S(h) := \frac{1}{m} \sum_{i=1}^m 1_{h(x_i) \neq c(x_i)} \quad (1.2)$$

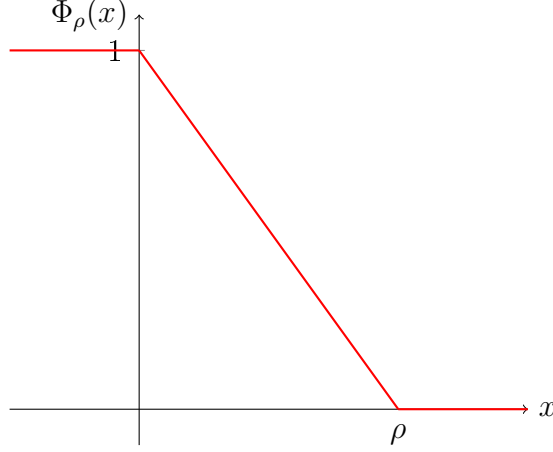
We can also consider  $\mathcal{C}$  (and thus  $\mathcal{H}$ ) to be subsets of functions mapping  $\mathcal{X}$  to  $\mathbb{R}$ . The class of a point  $x \in \mathcal{X}$  can then be simply  $\text{sgn}(h(x))$ . Such a  $\mathcal{H}$  is often called the class of **scoring functions**. This is more used in practice, since defining class of functions mapping to  $\mathbb{R}$  is easier than finding functions mapping to  $\{-1, +1\}$ . Also, they help us define the **confidence margin** of a pair  $(x, y)$  by  $yh(x)$  - larger the confidence, more our belief that the label of  $x$  is  $y$ .

Using this concept, we can define **empirical margin loss**, which is a slightly smoothed version of the 0-1 loss used in the definition of generalization error above. Instead of giving 0 error to every correctly classified example, it will simply give a linear error if the example is classified correctly but its margin is lower than some **margin**  $\rho > 0$ :

$$\hat{R}_{S,\rho}(h) := \frac{1}{m} \sum_{i=1}^m \Phi_\rho(y_i h(x_i)) \quad (1.3)$$

where  $\Phi_\rho : \mathbb{R} \rightarrow \mathbb{R}$  is defined as

$$\Phi_\rho(x) := \min \left\{ 1, \max \left\{ 0, 1 - \frac{x}{\rho} \right\} \right\} = \begin{cases} 1, & x \leq 0 \\ 1 - \frac{x}{\rho}, & 0 < x < \rho \\ 0, & x \geq \rho \end{cases} \quad (1.4)$$



**Figure 1.1:** The margin loss function  $\Phi_\rho(x)$

## 1.2 Multi-Class classification

The setup is exactly the same as binary classification, except for the following changes.

Firstly,  $\mathcal{Y} = \{1, \dots, k\}$ , for some  $k \in \mathbb{N}$ . Here,  $k$  represents the number of classes.

Secondly, for multi-class, we will have to work with empirical margin loss directly. We cannot do something simple like defining the confidence margin as  $yh(x)$  as in the binary case (because then we would be giving more weight to classes having a higher  $y$  label). We need to first have a score of how good a label  $y$  is for a given  $x$ , and then we compare these scores for other labels. Note that we did not need to compare in the binary case as the score itself was good enough to discriminate between two classes. Here, having a high score is not enough - what if two classes have both high scores - we then would need to compare their scores as well.

So, we need a scoring function  $h : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$  that would give us a score of how ‘good’ a label  $y$  is for a given  $x$ . Then, the margin of  $h \in \mathcal{H}$  at  $(x, y) \in \mathcal{X} \times \mathcal{Y}$  is given by

$$\rho_h(x, y) := h(x, y) - \max_{y' \neq y} h(x, y') \quad (1.5)$$

Now,  $\rho_h$  is the confidence margin - it tells us how confident we are with labelling

$x$  as  $y$  and not as any other  $y'$ . Note that  $h$  misclassifies  $x$  iff  $\rho_h(x, y) < 0$ . Finally, we define the empirical margin loss for any  $\rho > 0$  as

$$\hat{R}_{S,\rho}(h) := \frac{1}{m} \sum_{i=1}^m \Phi_\rho(\rho_h(x_i, y_i)) \quad (1.6)$$

where  $\Phi_\rho$  is the margin loss function.

## 1.3 Regression

The setup is again exactly the same as binary classification, except for the following changes.

Firstly,  $\mathcal{Y}$  is a measurable subset of  $\mathbb{R}$ . Thus, in contrast to classification,  $\mathcal{Y}$  can be potentially infinite.

Secondly, generalization error is defined as

$$R(h) := \mathbb{E}_{(x,y) \sim \mathcal{D}} [L(h(x), y)] \quad (1.7)$$

for some **loss function**  $L : \mathcal{Y} \times \mathcal{Y} \rightarrow [0, M]$ , for some  $M > 0$ . The loss function acts as a distance (though it need not be exactly be a distance) over  $\mathcal{Y}$ .

The empirical error is defined analogously:

$$\hat{R}_S(h) := \frac{1}{m} \sum_{i=1}^m L(h(x_i), y_i) \quad (1.8)$$

## Chapter 2

# GENERALIZATION BOUNDS

As mentioned in [section 1.1](#), we want to find bounds for the generalization error in terms of the empirical error. This chapter shall talk about that, and all the prerequisites needed to find the bounds as well.

To start, as mentioned in [section 1.1](#), we want to select the hypothesis set such that it is a ‘manageable’ (and measurable) subset of  $\mathcal{C}$ , ie. we want a hypothesis set that is neither too ‘simple’ nor too ‘complex’ in comparison to the sample. If  $\mathcal{H}$  is too ‘simple’, it may not even contain a hypothesis which is good enough to capture the patterns in the data. This is called **underfitting**. If  $\mathcal{H}$  is too ‘complex’, an algorithm might return a hypothesis that not only reduces error over the ‘patterns’ in the sample, but also the ‘noise’. This is called **overfitting**.

Thus, we need to be able to quantify how ‘manageable’ or ‘complex’ a hypothesis set is, with respect to the sample. We shall look into two such measures of complexity - Rademacher complexity (and generalization bounds using it ([section 2.1](#))), which depends on  $\mathcal{D}$  and is more of an analytical measure and VC dimension (and generalization bounds using it ([section 2.2](#))), which does not depend on  $\mathcal{D}$  and is a combinatorial measure.

Then, we shall look into bounding the margin loss ([section 2.3](#)) and bounding the generalization error for multi-class classification ([section 2.4](#)) and regression ([section 2.5](#)).

## 2.1 Rademacher Complexity

**Definition 2.1.1** (Rademacher complexity). The **empirical Rademacher complexity** of  $\mathcal{H}$  with respect to  $S$  is defined as

$$\hat{\mathfrak{R}}_S(\mathcal{H}) := \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m \sigma_i h(x_i) \right] \quad (2.1)$$

where  $\sigma := (\sigma_1, \dots, \sigma_m)^T$  with  $\sigma_1, \dots, \sigma_m \stackrel{\text{iid}}{\sim} \text{Unif}\{-1, +1\}$ , where the  $\sigma_i$ 's are called **Rademacher variables**. We then define the Rademacher complexity of  $\mathcal{H}$  to be

$$\mathfrak{R}_m(\mathcal{H}) := \mathbb{E}_{S \sim \mathcal{D}^m} [\hat{\mathfrak{R}}_S(\mathcal{H})] \quad (2.2)$$

The sum acts as an inner product between the vectors  $(h(x_1), \dots, h(x_m))$  and  $\sigma$ , which can be interpreted as the ‘similarity’ between them. Thus, if  $\mathfrak{R}_m(\mathcal{H})$  is high, it means that if we take an iid sample from  $\mathcal{D}^m$ , there would exist a hypothesis from  $\mathcal{H}$  which when applied on that sample will be able to ‘approximate’ random noise on the labels very well.

In other words, that hypothesis can map the sample very ‘close’ to random noise. The idea is that if there exists a hypothesis that can even approximate random noise on the labels, then  $\mathcal{H}$  should be complex enough to approximate any patterns that we throw at it.

We now use Rademacher complexity to derive a probabilistic bound on the generalization error of a hypothesis.

**Theorem 2.1.1.** *Let  $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$  and  $G \subseteq \{f : \mathcal{Z} \rightarrow [0, 1]\}$ . Then, for all  $\delta > 0$ , with probability at least  $1 - \delta$  over the draw of an iid sample  $S$  of size  $m$ , for all  $g \in G$ ,*

$$\mathbb{E}[g(z)] \leq \frac{1}{m} \sum_{i=1}^m g(z_i) + 2\mathfrak{R}_m(G) + \sqrt{\frac{\log(1/\delta)}{2m}} \quad (2.3)$$



$$\mathbb{E}[g(z)] \leq \frac{1}{m} \sum_{i=1}^m g(z_i) + 2\hat{\mathfrak{R}}_m(G) + 3\sqrt{\frac{\log(2/\delta)}{2m}} \quad (2.4)$$

*Proof.* For any sample  $S = (z_1, \dots, z_m)$  and any  $g \in G$ , denote by  $\hat{\mathbb{E}}_S[g]$  the empirical average of  $g$  over  $S$ :

$$\hat{\mathbb{E}}_S[g] := \frac{1}{m} \sum_{i=1}^m g(z_i) \quad (2.5)$$

We shall apply McDiarmid's inequality ([Appendix D](#)) to  $\Phi$ :

$$\Phi(S) := \sup_{g \in G} \left( \mathbb{E}[g] - \hat{\mathbb{E}}_S[g] \right) \quad (2.6)$$

Let  $S, S'$  be two samples differing by exactly one point, say  $z_m$  in  $S$  and  $z'_m$  in  $S'$ . Then,

$$\Phi(S) - \Phi(S') = \sup_{g \in G} \left( \mathbb{E}[g] - \hat{\mathbb{E}}_S[g] \right) - \sup_{g \in G} \left( \mathbb{E}[g] - \hat{\mathbb{E}}_{S'}[g] \right) \quad (2.7)$$

$$\leq \sup_{g \in G} \left( \hat{\mathbb{E}}_S[g] - \hat{\mathbb{E}}_{S'}[g] \right) \quad (2.8)$$

$$= \sup_{g \in G} \frac{g(z_m) - g(z'_m)}{m} \quad (2.9)$$

$$\leq \frac{1}{m} \quad (2.10)$$

where last inequality holds since each  $g \in G$  takes values in  $[0, 1]$ . Similarly, we can obtain  $\Phi(S) - \Phi(S') \leq \frac{1}{m}$ , thus  $|\Phi(S) - \Phi(S')| \leq \frac{1}{m}$ . Now, using McDiarmid's inequality with  $f = \Phi$ ,  $c_i = 1/m$  for all  $i \in [m]$ , we get that for any  $\delta > 0$  with probability at least  $1 - \delta/2$

$$\Phi(S) \leq \mathbb{E}_S[\Phi(S)] + \sqrt{\frac{\log(2/\delta)}{2m}} \quad (2.11)$$

Now, we bound the empirical average of  $\Phi$  by the Rademacher complexity of  $G$ .

$$\mathbb{E}_S[\Phi(S)] = \mathbb{E}_S \left[ \sup_{g \in G} \left( \mathbb{E}[g] - \hat{E}S(g) \right) \right] \quad (2.12)$$

$$= \mathbb{E}_S \left[ \sup_{g \in G} \mathbb{E}_{S'} \left( \hat{\mathbb{E}}_{S'}[g] - \hat{E}S(g) \right) \right] \quad (2.13)$$

$$\leq \mathbb{E}_{S, S'} \left[ \sup_{g \in G} \left( \hat{\mathbb{E}}_{S'}[g] - ES(g) \right) \right] \quad (2.14)$$

$$= \mathbb{E}_{S, S'} \left[ \sup_{g \in G} \frac{1}{m} \sum_{i=1}^m (g(z'_i) - g(z_i)) \right] \quad (2.15)$$

$$= \mathbb{E}_{\sigma, S, S'} \left[ \sup_{g \in G} \frac{1}{m} \sum_{i=1}^m \sigma_i (g(z'_i) - g(z_i)) \right] \quad (2.16)$$

$$\leq \mathbb{E}_{\sigma, S'} \left[ \sup_{g \in G} \frac{1}{m} \sum_{i=1}^m \sigma_i (g(z'_i)) \right] + \mathbb{E}_{\sigma, S} \left[ \sup_{g \in G} \frac{1}{m} \sum_{i=1}^m (-\sigma_i (g(z_i))) \right] \quad (2.17)$$

$$= \mathbb{E}_{\sigma, S} \left[ \sup_{g \in G} \frac{1}{m} \sum_{i=1}^m \sigma_i (g(z_i)) \right] \quad (2.18)$$

$$= 2\mathfrak{R}_m(G) \quad (2.19)$$

Thus, we get the first inequality. To get the second one, note that by definition of Rademacher complexity ([Definition 2.1.1](#)), changing one point in  $S$  changes  $\hat{\mathfrak{R}}_S(G)$  by at most  $1/m$ . Then, using McDiarmid's inequality, with probability at least  $1 - \delta/2$ ,

$$\mathfrak{R}_m(G) \leq \hat{\mathfrak{R}}_S(G) + \sqrt{\frac{\log(2/\delta)}{2m}} \quad (2.20)$$

Using union bound to combine the above inequalities, we get that with probability at least  $1 - \delta$ ,

$$\Phi(S) \leq 2\hat{\mathfrak{R}}_S(G) + 3\sqrt{\frac{\log(2/\delta)}{2m}} \quad (2.21)$$

□

**Lemma 2.1.2.** *Let  $\mathcal{H} \subseteq \{f : \mathcal{X} \rightarrow \{-1, +1\}\}$  and  $G := \{(x, y) \mapsto 1_{h(x) \neq y} : h \in \mathcal{H}\}$ . For a sample  $S = ((x_1, y_1), \dots, (x_m, y_m))$ , define  $S_{\mathcal{X}} = (x_1, \dots, x_m)$ . Then,*

$$\hat{\mathfrak{R}}_S(G) = \frac{1}{2} \hat{\mathfrak{R}}_{S_{\mathcal{X}}}(\mathcal{H}) \quad (2.22)$$

*Proof.*

$$\hat{\mathfrak{R}}_S(G) = \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m \sigma_i 1_{h(x_i) \neq y_i} \right] \quad (2.23)$$

$$= \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m \sigma_i \frac{1 - y_i h(x_i)}{2} \right] \quad (2.24)$$

$$= \frac{1}{2} \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m \sigma_i y_i h(x_i) \right] \quad (2.25)$$

$$= \frac{1}{2} \hat{\mathfrak{R}}_{S_{\mathcal{X}}}(\mathcal{H}) \quad (2.26)$$

□

**Theorem 2.1.3** (RC-based generalization bound for binary classification with 0-1 loss). *Let  $\delta > 0$ . Then,*

$$\mathbb{P}_{S \sim \mathcal{D}^m} \left( \forall h \in \mathcal{H}, R(h) \leq \hat{R}_S(h) + \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\ln(1/\delta)}{2m}} \right) \geq 1 - \delta \quad (2.27)$$

$$\mathbb{P}_{S \sim \mathcal{D}^m} \left( \forall h \in \mathcal{H}, R(h) \leq \hat{R}_S(h) + \hat{\mathfrak{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\ln(2/\delta)}{2m}} \right) \geq 1 - \delta \quad (2.28)$$

*Proof.* Use [Lemma 2.1.2](#) in [Theorem 2.1.1](#). □

This is an instance of Occam’s razor (popularly paraphrased as “The simplest explanation is usually the best one”) since the smaller the Rademacher complexity, lower the second term of RHS. Though note that, if we keep on reducing the complexity, we might not be able to decrease the empirical error (the first term). Thus, there is a tradeoff.

## 2.2 VC Dimension

**Definition 2.2.1** (Growth function). The growth function  $\Pi : \mathbb{N} \rightarrow \mathbb{N}$  at  $n \in \mathbb{N}$  gives the maximum number of distinct ways (called **dichotomies**) in which  $n$  points can be classified using a hypothesis in  $\mathcal{H}$ . Thus, for all  $n \in \mathbb{N}$ ,

$$\Pi_{\mathcal{H}}(n) := \max_{\{x_1, \dots, x_n\} \subseteq \mathcal{X}} \left| \{(h(x_1), \dots, h(x_n)) : h \in \mathcal{H}\} \right| \quad (2.29)$$

**Lemma 2.2.1** (Massart). *Let  $A \subseteq \mathbb{R}^m$  be a finite set with  $r = \max_{x \in A} \|x\|_2$ . Then,*

$$\mathbb{E}_\sigma \left[ \frac{1}{m} \sup_{x \in A} \sum_{i=1}^m \sigma_i x_i \right] \leq \frac{r \sqrt{2 \log |A|}}{m} \quad (2.30)$$

*Proof.* Apply maximal inequality (Appendix D) to the random variables  $\sigma_i x_i / m$ . □

**Theorem 2.2.2.** *Let  $G \subseteq \{f : \mathcal{X} \rightarrow [-1, +1]\}$ . Then,*

$$\mathfrak{R}_m(G) \leq \sqrt{\frac{2 \log \Pi_G(m)}{m}} \quad (2.31)$$

*Proof.* For a fixed sample  $S = (x_1, \dots, x_m)$ , denote by  $G|_S$  the set of vectors  $(g(x_1), \dots, g(x_m))^T$ . Then, using Lemma 2.2.1,

$$\mathfrak{R}_m(G) = \mathbb{E}_S \left[ \mathbb{E}_\sigma \left[ \sup_{u \in G|_S} \frac{1}{m} \sum_{i=1}^m \sigma_i u_i \right] \right] \quad (2.32)$$

$$\leq \mathbb{E}_S \left[ \frac{\sqrt{m} \sqrt{2 \log |G|_S}}{m} \right] \quad (2.33)$$

$$\leq \mathbb{E}_S \left[ \frac{\sqrt{m} \sqrt{2 \log \Pi_G(m)}}{m} \right] \quad (2.34)$$

$$= \sqrt{\frac{2 \log \Pi_G(m)}{m}} \quad (2.35)$$

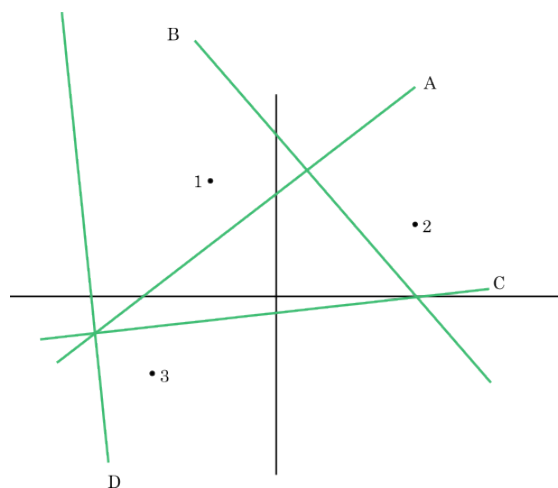
□

**Definition 2.2.2** (VC dimension). VC dimension of  $\mathcal{H}$  is the size of the largest set that can be given all possible dichotomies by  $\mathcal{H}$  (called being **shattered** by  $\mathcal{H}$ ).

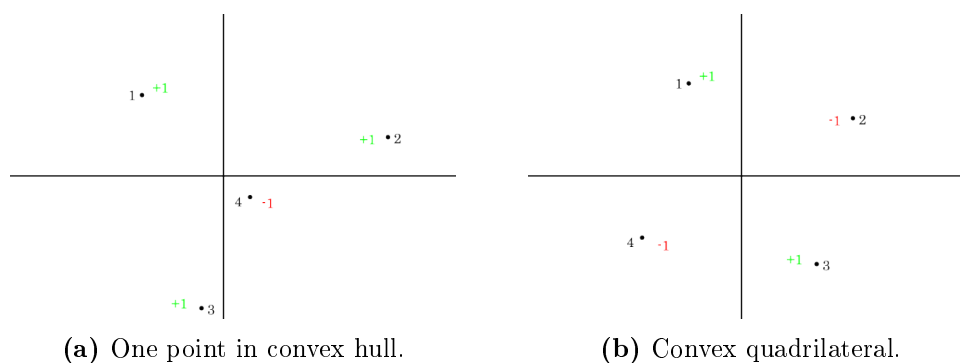
$$\text{VCdim}(\mathcal{H}) := \max\{n \in \mathbb{N} : \Pi_{\mathcal{H}}(n) = 2^n\} \quad (2.36)$$

**Example 2.2.1.** VC dimension of the set of all lines in  $\mathbb{R}^2$  (for binary classifying points in  $\mathbb{R}^2$ ) is at least 3, since there exists a set of three points (in particular, any set of three non-collinear points) that can be shattered (figure 2.1).

Moreover, no set containing 4 points can be shattered, since one point would fall



**Figure 2.1:** The three lines that can shatter a configuration of 3 points (each line represents two dichotomies).



**Figure 2.2:** These dichotomies of four points cannot be realised by lines.

either in the convex hull of the other three points (figure 2.2a), or the four points would form a convex quadrilateral (figure 2.2b).

Thus, VC dimension of the set of all lines in  $\mathbb{R}^2$  (for binary classifying points in  $\mathbb{R}^2$ ) is 3.

**Lemma 2.2.3** (Sauer). *Let  $\mathcal{H}$  be a hypothesis set with  $VCdim(\mathcal{H}) = d$ . Then for all  $m \in \mathbb{N}$ ,*

$$\Pi_{\mathcal{H}}(m) \leq \sum_{i=0}^d \binom{m}{i} \quad (2.37)$$

**Theorem 2.2.4.** *Let  $\mathcal{H}$  be a hypothesis set with  $VCdim(\mathcal{H}) = d$ . Then for all  $m \geq d$ ,*

$$\Pi_{\mathcal{H}}(m) \leq \left(\frac{em}{d}\right)^d \quad (2.38)$$

*Proof.*

$$\Pi_{\mathcal{H}}(m) \leq \sum_{i=0}^d \binom{m}{i} \quad (2.39)$$

$$\leq \sum_{i=0}^d \binom{m}{i} \left(\frac{m}{d}\right)^{d-i} \quad (2.40)$$

$$\leq \sum_{i=0}^m \binom{m}{i} \left(\frac{m}{d}\right)^{d-i} \quad (2.41)$$

$$= \left(\frac{m}{d}\right)^d \sum_{i=0}^m \binom{m}{i} \left(\frac{d}{m}\right)^i \quad (2.42)$$

$$= \left(\frac{m}{d}\right)^d \left(1 + \frac{d}{m}\right)^m \quad (2.43)$$

$$\leq \left(\frac{m}{d}\right)^d e^d \quad (2.44)$$

□

**Theorem 2.2.5** (VC dimension-based generalization bound). *Let  $VCdim(\mathcal{H}) = d < m$  and  $\delta > 0$ . Then,*

$$\mathbb{P}_{S \sim \mathcal{D}^m} \left( \forall h \in \mathcal{H}, R(h) \leq \hat{R}_S(h) + \sqrt{\frac{2d \ln(em/d)}{m}} + \sqrt{\frac{\ln(1/\delta)}{2m}} \right) \geq 1 - \delta \quad (2.45)$$

*Proof.* Apply [Theorem 2.2.4](#) to the last theorem of [section 2.1](#). □

## 2.3 Margin-based bounds

**Lemma 2.3.1** (Talagrand). *Let  $\Phi_1, \dots, \Phi_m : \mathbb{R} \rightarrow \mathbb{R}$  be  $l$ -Lipschitz functions. Let  $\sigma_1, \dots, \sigma_m$  be Rademacher variables. Then,*

$$\frac{1}{m} \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i (\Phi_i \circ h)(x_i) \right] \leq \frac{l}{m} \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i h(x_i) \right] = l \hat{R}_S(\mathcal{H}) \quad (2.46)$$

*Proof.* Define  $u_{m-1}(h) := \sum_{i=1}^{m-1} \sigma_i(\Phi_i \circ h)(x_i)$ . Then,

$$\frac{1}{m} \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i(\Phi_i \circ h)(x_i) \right] = \frac{1}{m} \mathbb{E}_{\sigma_1, \dots, \sigma_{m-1}} \left[ \mathbb{E}_{\sigma_m} \left[ \sup_{h \in \mathcal{H}} (u_{m-1} + \sigma_m(\Phi_m \circ h)(x_m)) \right] \right] \quad (2.47)$$

Now, by definition of sup, for all  $\varepsilon > 0$ , there exist  $h_1, h_2 \in \mathcal{H}$  such that

$$u_{m-1}(h_1) + (\Phi_m \circ h_1)(x_m) \geq (1 - \varepsilon) \left[ \sup_{h \in \mathcal{H}} (u_{m-1} + \sigma_m(\Phi_m \circ h)(x_m)) \right] \quad (2.48)$$

$$u_{m-1}(h_1) - (\Phi_m \circ h_2)(x_m) \geq (1 - \varepsilon) \left[ \sup_{h \in \mathcal{H}} (u_{m-1} - \sigma_m(\Phi_m \circ h)(x_m)) \right] \quad (2.49)$$

Now, let  $\varepsilon > 0$  and  $s = \text{sgn}\{h_1(x_m) - h_2(x_m)\}$ . Then,

$$(1 - \varepsilon) \mathbb{E}_{\sigma_m} \left[ \sup_{h \in \mathcal{H}} (u_{m-1} + \sigma_m(\Phi_m \circ h)(x_m)) \right] \quad (2.50)$$

$$= (1 - \varepsilon) \left[ \sup_{h \in \mathcal{H}} u_{m-1}(h) \right] \quad (2.51)$$

$$= (1 - \varepsilon) \left[ \frac{1}{2} \sup_{h \in \mathcal{H}} u_{m-1}(h) + (\Phi_m \circ h)(x_m) + \frac{1}{2} \sup_{h \in \mathcal{H}} u_{m-1}(h) - (\Phi_m \circ h)(x_m) \right] \quad (2.52)$$

$$\leq \left[ \frac{1}{2} u_{m-1}(h_1) + (\Phi_m \circ h_1)(x_m) + \frac{1}{2} u_{m-1}(h_2) - (\Phi_m \circ h_2)(x_m) \right] \quad (2.53)$$

$$= \frac{1}{2} [u_{m-1}(h_1) + u_{m-1}(h_2) + (\Phi_m \circ h_1)(x_m) - (\Phi_m \circ h_2)(x_m)] \quad (2.54)$$

$$\leq \frac{1}{2} [u_{m-1}(h_1) + u_{m-1}(h_2) + (h_1(x_m) - h_2(x_m))ls] \quad (2.55)$$

$$= \frac{1}{2} [u_{m-1}(h_1) + slh_1(x_m)] + \frac{1}{2} [u_{m-1}(h_2) - slh_2(x_m)] \quad (2.56)$$

$$\leq \frac{1}{2} \left[ \sup_{h \in \mathcal{H}} (u_{m-1}(h) + slh(x_m)) \right] + \frac{1}{2} \left[ \sup_{h \in \mathcal{H}} (u_{m-1}(h) - slh(x_m)) \right] \quad (2.57)$$

$$= \mathbb{E}_{\sigma_m} \left[ \sup_{h \in \mathcal{H}} (u_{m-1}(h) + \sigma_m lh(x_m)) \right] \quad (2.58)$$

Since this holds for all  $\varepsilon > 0$ ,

$$\mathbb{E}_{\sigma_m} \left[ \sup_{h \in \mathcal{H}} (u_{m-1} + \sigma_m(\Phi_m \circ h)(x_m)) \right] \leq \mathbb{E}_{\sigma_m} \left[ \sup_{h \in \mathcal{H}} (u_{m-1}(h) + \sigma_m lh(x_m)) \right] \quad (2.59)$$

□

**Corollary 2.3.1.1.** *If  $\Phi_i = \Phi$  for all  $i \in [m]$ , then  $\hat{R}_S(\Phi \circ \mathcal{H}) \leq l\hat{R}_S(\mathcal{H})$ .*

**Theorem 2.3.2** (Margin-based generalization bound). *Let  $\mathcal{H} \subseteq \{f : \mathcal{X} \rightarrow \mathbb{R}\}$  be a hypothesis set. Let  $\rho > 0$ ,  $\delta > 0$ . Then,*

$$\mathbb{P}_{S \sim \mathcal{D}^m} \left( \forall h \in \mathcal{H}, R(h) \leq \hat{R}_{S,\rho}(h) + \frac{2}{\rho} \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\ln(1/\delta)}{2m}} \right) \geq 1 - \delta \quad (2.60)$$

$$\mathbb{P}_{S \sim \mathcal{D}^m} \left( \forall h \in \mathcal{H}, R(h) \leq \hat{R}_{S,\rho}(h) + \frac{2}{\rho} \hat{\mathfrak{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\ln(2/\delta)}{2m}} \right) \geq 1 - \delta \quad (2.61)$$

*Proof.* Define the sets of confidence margins and margin loss of each hypothesis as

$$\mathcal{H}_C := \{z = (x, y) \mapsto yh(x) : h \in \mathcal{H}\} \quad (2.62)$$

$$\mathcal{H}_M := \{\Phi_\rho \circ f : f \in \mathcal{H}_C\} \quad (2.63)$$

By first theorem of [section 2.1](#), for all  $g \in \mathcal{H}_M$ ,

$$\mathbb{E}[g(z)] \leq \frac{1}{m} \sum_{i=1}^m g(z_i) + 2\mathfrak{R}_m(\mathcal{H}_M) + \sqrt{\frac{\log 1/\delta}{2m}} \quad (2.64)$$

$$\mathbb{E}[g(z)] \leq \frac{1}{m} \sum_{i=1}^m g(z_i) + 2\hat{\mathfrak{R}}_S(\mathcal{H}_M) + 3\sqrt{\frac{\log 2/\delta}{2m}} \quad (2.65)$$

Thus, for all  $h \in \mathcal{H}$ ,

$$\mathbb{E}[\Phi_\rho(yh(x))] \leq \hat{R}_{S,\rho}(h) + 2\mathfrak{R}_m(\Phi_\rho \circ \mathcal{H}) + \sqrt{\frac{\log 1/\delta}{2m}} \quad (2.66)$$

$$\mathbb{E}[\Phi_\rho(yh(x))] \leq \hat{R}_{S,\rho}(h) + 2\hat{\mathfrak{R}}_S(\Phi_\rho \circ \mathcal{H}) + 3\sqrt{\frac{\log 2/\delta}{2m}} \quad (2.67)$$

We are done since  $R(h) \leq \mathbb{E}[\Phi_\rho(yh(x))]$  (because  $1_{u \leq 0} \leq \Phi_\rho(u)$  for all  $u \in \mathbb{R}$ ) and  $\hat{\mathfrak{R}}_S(\Phi_\rho \circ \mathcal{H}) \leq \frac{1}{\rho} \hat{\mathfrak{R}}_S(\mathcal{H})$  by [Corollary 2.3.1.1](#) since  $\Phi_\rho$  is  $1/\rho$ -Lipschitz. Note that  $\mathfrak{R}_m(\Phi_\rho \circ \mathcal{H}) \leq \frac{1}{\rho} \mathfrak{R}_m(\mathcal{H})$  follows by just taking expectation both sides.  $\square$

**Theorem 2.3.3** (Uniform Margin-based generalization bound). *Let  $\mathcal{H} \subseteq \{f : \mathcal{X} \rightarrow \mathbb{R}\}$  be a hypothesis set. Let  $r > 0$ ,  $\delta > 0$ . Then, for all  $\delta > 0$ , with probability at least  $1 - \delta$  over the draw of an iid sample  $S$  of size  $m$ , for all  $h \in \mathcal{H}$*



and  $\rho \in (0, r]$ ,

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{4}{\rho} \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\ln(\log_2(2r/\rho))}{m}} + \sqrt{\frac{\ln(2/\delta)}{2m}} \quad (2.68)$$

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{4}{\rho} \hat{\mathfrak{R}}_S(\mathcal{H}) + \sqrt{\frac{\ln(\log_2(2r/\rho))}{m}} + 3\sqrt{\frac{\ln(4/\delta)}{2m}} \quad (2.69)$$

*Proof.* Consider the positive sequences  $\{\rho_k\}_{k \in \mathbb{N}}$  and  $\{\varepsilon_k\}_{k \in \mathbb{N}}$  with  $\varepsilon_k \in (0, 1]$ . By [Theorem 2.3.2](#), for any  $k \in \mathbb{N}$ ,

$$\mathbb{P} \left[ \sup_{h \in \mathcal{H}} \left( R(h) - \hat{R}_{S,\rho_k}(h) \right) > \frac{2}{\rho_k} \mathfrak{R}_m(\mathcal{H}) + \varepsilon_k \right] \leq e^{-2m\varepsilon_k^2} \quad (2.70)$$

Choose  $\varepsilon_k := \varepsilon + \sqrt{\frac{\log k}{m}}$ . Then,

$$\mathbb{P} \left[ \sup_{h \in \mathcal{H}, k \in \mathbb{N}} \left( R(h) - \hat{R}_{S,\rho_k}(h) \right) - \frac{2}{\rho_k} \mathfrak{R}_m(\mathcal{H}) - \varepsilon_k > 0 \right] \quad (2.71)$$

$$\leq \sum_{k \in \mathbb{N}} e^{-2m\varepsilon_k^2} \quad (2.72)$$

$$= \sum_{k \in \mathbb{N}} e^{-2m \left( \varepsilon + \sqrt{\frac{\log k}{m}} \right)^2} \quad (2.73)$$

$$\leq \sum_{k \in \mathbb{N}} e^{-2m\varepsilon^2} e^{-2 \log k} \quad (2.74)$$

$$= e^{-2m\varepsilon^2} \sum_{k \in \mathbb{N}} \frac{1}{k^2} \quad (2.75)$$

$$= \frac{\pi^2}{6} e^{-2m\varepsilon^2} \quad (2.76)$$

$$< 2e^{-2m\varepsilon^2} \quad (2.77)$$

Now, choose  $\rho_k = \frac{r}{2^k}$ . For any  $\rho \in (0, r]$ , there exists  $k \in \mathbb{N}$  such that  $\rho \in (\rho_k, \rho_{k-1}]$ , with  $\rho_0 = r$ . For that  $k$ ,  $\rho \leq \rho_{k-1} = 2\rho_k$ , which implies  $\frac{1}{\rho_k} \leq \frac{2}{\rho}$ , which implies

$$\sqrt{\log k} = \sqrt{\log \log_2 \frac{r}{\rho_k}} \leq \sqrt{\log \log_2 \frac{2r}{\rho}} \quad (2.78)$$

Also, for all  $h \in \mathcal{H}$ , since  $\rho > \rho_k$ ,  $\hat{R}_{S,\rho_k}(h) \leq \hat{R}_{S,\rho}(h)$ . Thus, replacing  $\rho_k$  by  $\rho$ ,

$$\mathbb{P} \left[ \sup_{h \in \mathcal{H}, \rho \in (0,r]} \left( R(h) - \hat{R}_{S,\rho}(h) \right) - \frac{4}{\rho} \mathfrak{R}_m(\mathcal{H}) - \sqrt{\frac{\log \log_2(2r/\rho)}{m}} - \varepsilon > 0 \right] \leq 2e^{-2m\varepsilon^2} \quad (2.79)$$

□

## 2.4 Multi-class generalization bounds

**Lemma 2.4.1.** *Let  $F_1, \dots, F_l \subseteq \mathcal{H}$ . Let*

$$G := \{\max\{h_1, \dots, h_l\} : h_i \in F_i, \forall i \in [l]\} \quad (2.80)$$

*Then, for all  $S$  of size  $m$ ,  $\hat{R}_S(G) \leq \sum_{j=1}^l \hat{R}_S(F_j)$*

*Proof.* Let  $S = (x_1, \dots, x_m)$  and let  $l = 2$ . We know that for all  $h_1 \in F_1, h_2 \in F_2$ ,

$$\max\{h_1, h_2\} = \frac{1}{2}[h_1 + h_2 + |h_1 - h_2|] \quad (2.81)$$

Thus,

$$\hat{R}_S(G) = \frac{1}{m} \mathbb{E} \left[ \sup_{h_1 \in F_1, h_2 \in F_2} \sum_{i=1}^m \sigma_i \max\{h_1(x_i), h_2(x_i)\} \right] \quad (2.82)$$

$$= \frac{1}{2m} \mathbb{E} \left[ \sup_{h_1 \in F_1, h_2 \in F_2} \sum_{i=1}^m \sigma_i [h_1(x_i) + h_2(x_i) + |h_1(x_i) - h_2(x_i)|] \right] \quad (2.83)$$

$$\leq \frac{1}{2} [\hat{R}_S(F_1) + \hat{R}_S(F_2)] + \frac{1}{2m} \mathbb{E} \left[ \sup_{h_1 \in F_1, h_2 \in F_2} \sum_{i=1}^m \sigma_i |h_1(x_i) - h_2(x_i)| \right] \quad (2.84)$$

Now, using Talagrand's lemma ([section 2.3](#)) with  $\Phi(x) = |x|$  (since it is 1-Lipschitz),

$$\frac{1}{2m} \mathbb{E} \left[ \sup_{h_1 \in F_1, h_2 \in F_2} \sum_{i=1}^m \sigma_i |h_1(x_i) - h_2(x_i)| \right] \quad (2.85)$$

$$\leq \frac{1}{2m} \mathbb{E} \left[ \sup_{h_1 \in F_1, h_2 \in F_2} \sum_{i=1}^m \sigma_i[h_1(x_i) - h_2(x_i)] \right] \quad (2.86)$$

$$\leq \frac{1}{2} [\hat{R}_S(F_1) + \hat{R}_S(F_2)] \quad (2.87)$$

Thus,  $\hat{R}_S(G) \leq \hat{R}_S(F_1) + \hat{R}_S(F_2)$ . Now since

$$\max\{h_1, \dots, h_l\} = \max\{h_1, \max\{h_2, \dots, h_l\}\} \quad (2.88)$$

we can iterate and get the result.  $\square$

**Theorem 2.4.2** (Margin bound). *Let  $\mathcal{H} := \{h : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}\}$  with  $\mathcal{Y} = \{1, \dots, k\}$ .*

*Let  $\rho > 0$ . Define*

$$\Pi_1(\mathcal{H}) := \{x \mapsto h(x, y) : y \in \mathcal{Y}, h \in \mathcal{H}\} \quad (2.89)$$

*Then, for all  $\delta > 0$  with probability at least  $1 - \delta$ , for all  $h \in \mathcal{H}$ ,*

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{4k}{\rho} \mathfrak{R}_m(\Pi_1(\mathcal{H})) + \sqrt{\frac{\log(1/\delta)}{2m}} \quad (2.90)$$

*Proof.* For  $\theta > 0$ , define  $\rho_{\theta,h}(x, y) := \min_{y'} \{h(x, y) - h(x, y') + \theta 1_{y'=y}\}$ . Note that

$$\rho_{\theta,h}(x, y) = \min_{y'} \{h(x, y) - h(x, y') + \theta 1_{y=y'}\} \quad (2.91)$$

$$\leq \min_{y' \neq y} \{h(x, y) - h(x, y') + \theta 1_{y=y'}\} \quad (2.92)$$

$$= \min_{y' \neq y} \{h(x, y) - h(x', y)\} \quad (2.93)$$

$$= \rho_h(x, y) \quad (2.94)$$

Define  $\tilde{\mathcal{H}} := \{(x, y) \mapsto \rho_{\theta,h}(x, y) : h \in \mathcal{H}\}$  and  $H := \Phi_\rho \circ \tilde{\mathcal{H}}$ . Using first theorem of [section 2.1](#) on  $H$ , we get that with probability at least  $1 - \delta$ , for all  $h \in \mathcal{H}$ ,

$$\mathbb{E}[\Phi_\rho(\rho_{\theta,h}(x, y))] \leq \frac{1}{m} \sum_{i=1}^m \Phi_\rho(\rho_{\theta,h}(x_i, y_i)) + 2\mathfrak{R}_m(H) + \sqrt{\frac{\log 1/\delta}{2m}} \quad (2.95)$$

Since  $R(h) \leq \mathbb{E}[\Phi_\rho(\rho_{\theta,h}(x, y))]$  for all  $h \in H$ , we get

$$R(h) \leq \frac{1}{m} \sum_{i=1}^m \Phi_\rho(\rho_{\theta,h}(x_i, y_i)) + 2\mathfrak{R}_m(H) + \sqrt{\frac{\log 1/\delta}{2m}} \quad (2.96)$$

Fixing  $\theta = 2\rho$ ,

$$\Phi_\rho(\rho_{\theta,h}(x_i, y_i)) = \Phi_\rho(\rho_h(x_i, y_i)) \quad (2.97)$$

where  $\rho_h$  is the multi-class scoring function (1.5), since either  $\rho_{\theta,h}(x_i, y_i) = \rho_h(x_i, y_i)$  or  $2\rho = \rho_{\theta,h}(x_i, y_i) \leq \rho_h(x_i, y_i)$ . Furthermore, Talagrand's lemma (section 2.3) implies  $\mathfrak{R}_m(H) = \frac{1}{\rho} \mathfrak{R}_m(\tilde{\mathcal{H}})$ . Thus, for all  $\delta > 0$ , with probability at least  $1 - \delta$ , for all  $h \in \mathcal{H}$ ,

$$R(h) \leq \hat{R}(S, \rho)(h) + \frac{2}{\rho} \mathfrak{R}_m(\tilde{\mathcal{H}}) + \sqrt{\frac{\log 1/\delta}{2m}} \quad (2.98)$$

To complete the proof, we need to show  $\mathfrak{R}_m(\tilde{\mathcal{H}}) \leq 2k\mathfrak{R}_m(\Pi_1(\mathcal{H}))$ .

$$\mathfrak{R}_m(\tilde{\mathcal{H}}) \quad (2.99)$$

$$= \frac{1}{m} \mathbb{E} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i(h(x_i, y_i) - \max_y(h(x_i, y) - 2\rho 1_{y=y_i})) \right] \quad (2.100)$$

$$\leq \frac{1}{m} \mathbb{E} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i(h(x_i, y_i)) \right] + \frac{1}{m} \mathbb{E} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i(\max_y(h(x_i, y) - 2\rho 1_{y=y_i})) \right] \quad (2.101)$$

Now,

$$\frac{1}{m} \mathbb{E} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sum_{y \in \mathcal{Y}} \sigma_i h(x_i, y) 1_{y=y_i} \right] \quad (2.102)$$

$$\leq \frac{1}{m} \sum_{y \in \mathcal{Y}} \mathbb{E}_{S, \sigma} \left[ \sum_{i=1}^m \sigma_i h(x_i, y) 1_{y=y_i} \right] \quad (2.103)$$

$$= \sum_{y \in \mathcal{Y}} \frac{1}{m} \mathbb{E}_{S, \sigma} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i h(x_i, y) \varepsilon_i \right] \quad (2.104)$$

where  $\varepsilon_i = 21_{y_i=y} - 1$ . Since  $\varepsilon_i \in \{-1, +1\}$ ,  $\sigma_i$  and  $\sigma_i \varepsilon_i$  follow the same distribution.

So, for all  $y \in \mathcal{Y}$ ,

$$\frac{1}{m} \mathbb{E}_{S, \sigma} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i h(x_i, y) \varepsilon_i \right] \quad (2.105)$$

$$\frac{1}{2m} \mathbb{E}_{S, \sigma} \left[ \sup_{h \in \mathcal{H}} \sigma_i \varepsilon_i h(x_i, y) \right] + \frac{1}{2m} \mathbb{E}_{S, \sigma} \left[ \sup_{h \in \mathcal{H}} \sigma_i h(x_i, y) \right] \quad (2.106)$$

$$= \hat{\mathfrak{R}}_m(\Pi_1(\mathcal{H})) \quad (2.107)$$

Adding up,

$$\frac{1}{m} \mathbb{E}_{S, \sigma} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sum_{y \in \mathcal{Y}} \sigma_i h(x_i, y) 1_{y=y_i} \right] \leq k \hat{\mathfrak{R}}_m(\Pi_1(\mathcal{H})) \quad (2.108)$$

Finally, by [Lemma 2.4.1](#),

$$\frac{1}{m} \mathbb{E}_{S, \sigma} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i (\max_y (h(x_i, y) - 2\rho 1_{y=y_i})) \right] \quad (2.109)$$

$$\leq \sum_{y \in \mathcal{Y}} \frac{1}{m} \mathbb{E}_{S, \sigma} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i (h(x_i, y) - 2\rho 1_{y=y_i}) \right] \quad (2.110)$$

$$= \sum_{y \in \mathcal{Y}} \frac{1}{m} \mathbb{E}_{S, \sigma} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i (h(x_i, y) - 2\rho \sum_{i=1}^m \sigma_i 1_{y=y_i}) \right] \quad (2.111)$$

$$= \sum_{y \in \mathcal{Y}} \frac{1}{m} \mathbb{E}_{S, \sigma} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i h(x_i, y) \right] \quad (2.112)$$

$$\leq \sum_{y \in \mathcal{Y}} \mathfrak{R}_m(\Pi_1(\mathcal{H})) \quad (2.113)$$

$$= k \mathfrak{R}_m(\Pi_1(\mathcal{H})) \quad (2.114)$$

□

## 2.5 Regression generalization bounds

**Theorem 2.5.1** (Rademacher complexity bound). *For all  $y' \in \mathcal{Y}$ , let  $L(y, y') \leq \mu y$  for all  $y \in \mathcal{Y}$ , for some  $\mu > 0$ . Define  $G := \{(x, y) \mapsto L(h(x), y) : h \in \mathcal{H}\}$ . Then, for any sample  $S$ ,*

$$\hat{\mathfrak{R}}_S(G) \leq \mu \hat{\mathfrak{R}}_S(\mathcal{H}) \quad (2.115)$$

*Proof.* Using Talagrand's lemma from [section 2.3](#),

$$\hat{R}_S(G) = \frac{1}{m} \mathbb{E} \left[ \sum_{i=1}^m \sigma_i L(h(x_i), y_i) \right] \leq \frac{1}{m} \mathbb{E} \left[ \sum_{i=1}^m \sigma_i \mu h(x_i) \right] = \mu \hat{R}_S(\mathcal{H}) \quad (2.116)$$

□

**Corollary 2.5.1.1** (RC-based regression bound). *For all  $y' \in \mathcal{Y}$ , let  $y \mapsto L(y, y')$  be  $\mu$ -Lipschitz for some  $\mu > 0$ . Then,*

$$R(h) \leq \hat{R}_S(h) + 2\mu \mathfrak{R}_m(\mathcal{H}) + M \sqrt{\frac{\log 1/\delta}{2m}} \quad (2.117)$$

$$R(h) \leq \hat{R}_S(h) + 2\mu \hat{\mathfrak{R}}_m(\mathcal{H}) + 3M \sqrt{\frac{\log 2/\delta}{2m}} \quad (2.118)$$

*Proof.* Use [Theorem 2.5.1](#) in last theorem of [section 2.1](#). □

The notion of VC dimension does not apply directly here because  $\mathcal{Y}$  is potentially infinite. So, an analogous notion is needed.

**Definition 2.5.1** (Shattering). A finite set  $\{z_1, \dots, z_m\} \subseteq \mathcal{X} \times \mathcal{Y}$  is said to be shattered by  $G \subseteq \{g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}\}$  if there exist  $t_1, \dots, t_m \in \mathbb{R}$  such that

$$\left| \left\{ \begin{bmatrix} \text{sgn}(g(z_1) - t_1) \\ \vdots \\ \text{sgn}(g(z_m) - t_m) \end{bmatrix} : g \in G \right\} \right| = 2^m \quad (2.119)$$

*Remark 2.5.1.* So, the set  $\{z_1, \dots, z_m\}$  is shattered by  $G$  for witnesses  $t_1, \dots, t_m$  if  $G$  is rich enough to go over a subset  $A \subseteq J := \{(z_i, t_i) : i \in [m]\}$  and below  $J \setminus A$ , for all  $A \subseteq J$ .

**Definition 2.5.2** (Pseudo-dimension). Let  $G \subseteq \{g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}\}$ . Then, the pseudo-dimension of  $G$ , denoted by  $\text{Pdim}(G)$  is the size of the largest set shattered by  $G$ .



**Figure 2.3:**  $\{z_1, z_2\}$  being shattered by witnesses  $t_1, t_2$  (Mohri et al., 2018, pg. 271).

Thus, for  $G \subseteq \{g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}\}$ ,

$$\text{Pdim}(G) = \text{VCdim}\left(\{(x, t) \mapsto 1_{g(x)-t>0} : g \in G\}\right) \quad (2.120)$$

**Theorem 2.5.2** (Pdim-based generalization bound). *Let  $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X}}$  and  $G := \{(x, y) \mapsto L(h(x), y) : h \in \mathcal{H}\}$ . Let  $\text{Pdim}(G) = d$ . Then, for all  $\delta > 0$ , with probability at least  $1 - \delta$ , for all  $h \in \mathcal{H}$ ,*

$$R(h) \leq \hat{R}_S(h) + M \sqrt{\frac{2d \log(em/d)}{m}} + M \sqrt{\frac{\log(1/\delta)}{2m}} \quad (2.121)$$

*Proof.* Let  $\hat{\mathcal{D}}$  be the empirical distribution of a sample  $S$  drawn independently from  $\mathcal{D}$ . For all  $h \in \mathcal{H}$  and  $t \geq 0$ , define a classifier

$$c(h, t)(x, y) := 1_{L(h(x), y) > t} \quad (2.122)$$

Now,  $R(c(h, t)) = \mathbb{P}_{(x, y) \sim \mathcal{D}}[L(h(x), y) > t]$  and  $\hat{R}_S(c(h, t)) = \mathbb{P}_{(x, y) \sim \hat{\mathcal{D}}}[L(h(x), y) > t]$ .

Now,

$$|R(h) - \hat{R}_S(h)| = \left| \mathbb{E}_{(x, y) \sim \mathcal{D}}[L(h(x), y)] - \mathbb{E}_{(x, y) \sim \hat{\mathcal{D}}}[L(h(x), y)] \right| \quad (2.123)$$

$$= \left| \int_0^M \left( \mathbb{P}_{(x, y) \sim \mathcal{D}}[L(h(x), y) > t] - \mathbb{P}_{(x, y) \sim \hat{\mathcal{D}}}[L(h(x), y) > t] \right) dt \right| \quad (2.124)$$

$$\leq M \sup_{t \in [0, M]} \left| \mathbb{P}_{(x, y) \sim \mathcal{D}} [L(h(x), y) > t] - \mathbb{P}_{(x, y) \sim \hat{\mathcal{D}}} [L(h(x), y) > t] \right| \quad (2.125)$$

$$= M \sup_{t \in [0, M]} |R(c(h, t)) - \hat{R}_S(c(h, t))| \quad (2.126)$$

where second inequality holds because the following holds for any nonnegative  $f$ :

$$\mathbb{E}_{z \sim \mathcal{D}} [f(z)] = \int_0^\infty \mathbb{P}_{z \sim \mathcal{D}} [f(z) > t] dt \quad (2.127)$$

Thus, we get

$$\mathbb{P}[\sup_{h \in \mathcal{H}} |R(h) - \hat{R}_S(h)| > \varepsilon] \leq \mathbb{P} \left[ \sup_{h \in \mathcal{H}, t \in [0, M]} |R(c(h, t)) - \hat{R}_S(c(h, t))| > \frac{\varepsilon}{M} \right] \quad (2.128)$$

where RHS can be bounded by last theorem of [section 2.2](#) using [Theorem 2.5.2](#).  $\square$



## Chapter 3

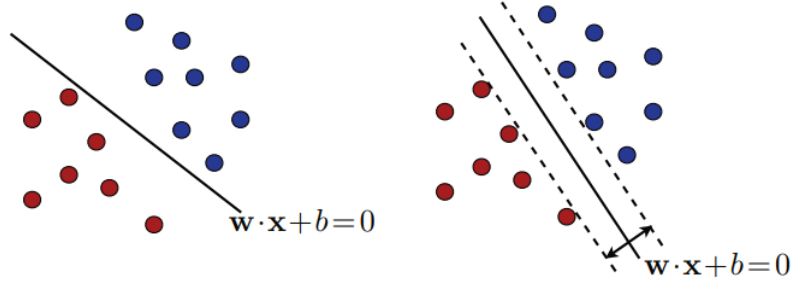
# SVMs AND KERNELS

Finding ways (usually some sort of algorithms) to find the hypothesis minimizing the empirical risk is one of the biggest and central challenges in the domain of machine learning and artificial intelligence. One of the most theoretically well-motivated and practically effective algorithms is support vector machines (SVMs).

We shall work with the following setup for this chapter :  $\mathcal{X} \subset \mathbb{R}^N$  for some  $N \in \mathbb{N}$ ,  $S = ((x_1, y_1), \dots, (x_m, y_m)) \stackrel{\text{iid}}{\sim} \mathcal{D}$  and  $\mathcal{H} := \{x \mapsto \text{sgn}(w^T x + b) : w \in \mathbb{R}^N \setminus \{0\}, b \in \mathbb{R}\}$ .

We start with  $\mathcal{Y} := \{-1, +1\}$ , ie. the problem of binary classification addressed using hypotheses of the form of hyperplanes. We shall consider two cases : when there exists a hyperplane that can separate the training data perfectly ([section 3.1](#)), and when there does not exist such a hyperplane ([section 3.2](#)).

We then look into how we can derive the SVM algorithm by minimizing an upper bound on the margin loss ([section 3.3](#)), look into SVM variants for multi-class classification ([section 3.4](#)) and regression ([section 3.5](#)) and then end with studying kernels in a general setting, and using it to extend functionality of SVMs ([section 3.6](#)).



**Figure 3.1:** A bad vs. good separating hyperplane (Mohri et al., 2018, pg. 80).

## 3.1 Separable Case

We shall assume that  $S$  is **linearly separable**, ie. there exists  $(w, b) \in (\mathbb{R}^N \setminus \{0\}) \times \mathbb{R}$  such that

$$y_i(w^T x_i + b) \geq 0 \quad (3.1)$$

for all  $i \in [m]$ .

### 3.1.1 Primal Optimization Problem

A good separating hyperplane would be the one that is not very ‘close’ to any of the training data points, since then we are the most confident with the separation. We don’t want a hyperplane that is ‘close’ to some data points. In other words, we want to maximize the minimum distance between the hyperplane and the training dataset.

Distance between a hyperplane and a point is given by the **geometric margin**.

For a hyperplane  $h \in \mathcal{H}$ ,

$$\rho_h(x) := \frac{|w^T x + b|}{\|w\|} \quad (3.2)$$

for all  $x \in \mathbb{R}^N$ . The SVM solution must maximize  $\rho_h$ , where

$$\rho_h := \min_{i \in [m]} \rho_h(x_i) \quad (3.3)$$

Thus, the primal optimization problem is:

$$\rho_{\text{SVM}} := \max_{w, b: \min_{i \in [m]} y_i(w^T x_i + b) \geq 0} \min_{i \in [m]} \frac{|w^T x_i + b|}{\|w\|} = \max_{w, b \in \mathbb{R}^N \times \mathbb{R}} \min_{i \in [m]} \frac{y_i(w^T x_i + b)}{\|w\|} \quad (3.4)$$

where second equality holds because  $S$  is linearly separable. Note that the quantity  $\frac{y_i(w^T x_i + b)}{\|w\|}$  does not change under  $(w, b)$  being scaled by a constant. Thus, we scale  $(w, b)$  such that  $\min_{i \in [m]} y_i(w^T x_i + b) = 1$ , which changes our optimization problem to

$$\rho_{\text{SVM}} = \max_{w, b: \min_{i \in [m]} y_i(w^T x_i + b) = 1} \frac{1}{\|w\|} = \max_{w, b: y_i(w^T x_i + b) \geq 1 \forall i \in [m]} \frac{1}{\|w\|} \quad (3.5)$$

where the second equality holds because the minimum of  $y_i(w^T x_i + b)$  cannot be strictly more than 1, since if it were say  $\alpha > 1$ , we could divide  $(w, b)$  by  $\alpha$ , which would yield the minimum to be 1 and give us a larger value of  $\frac{1}{\|w\|}$ , thus giving a contradiction.

We finally state the primal optimization problem as follows.

$$\begin{aligned} & \min_{w, b \in \mathbb{R}^N \times \mathbb{R}} \frac{1}{2} \|w\|^2 \\ & \text{subject to: } y_i(w^T x_i + b) \geq 1 \quad \forall i \in [m] \end{aligned} \quad (3.6)$$

since maximizing  $\frac{1}{\|w\|}$  or minimizing  $\frac{1}{2} \|w\|^2$  would yield the same  $(w, b)$ . Define  $F : \mathbb{R}^N \rightarrow \mathbb{R}$  by  $F(w) := \frac{1}{2} \|w\|^2$ , for all  $w \in \mathbb{R}^N$ .

### 3.1.2 Dual Optimization Problem

The Lagrangian for the above optimization problem is  $\mathcal{L} : \mathbb{R}^N \times \mathbb{R} \times \mathbb{R}^m \rightarrow \mathbb{R}$  defined by

$$\mathcal{L}(w, b, \alpha) := \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i (y_i(w^T x_i + b) - 1) \quad (3.7)$$

for all  $(w, b, \alpha) \in \mathbb{R}^N \times \mathbb{R} \times \mathbb{R}^m$ . Thus, the dual problem is

$$\max_{\alpha \in \mathbb{R}^m} \left( \min_{(w, b) \in \mathbb{R}^N \times \mathbb{R}} \mathcal{L}(w, b, \alpha) \right) \quad (3.8)$$

subject to:  $\alpha \geq 0$

Strong duality holds by Slater's conditions ([Appendix A](#)), since  $F$  and the constraints are convex, the constraints are affine linear and there exists  $(w, b) \in \mathbb{R}^N \times \mathbb{R}$  where all constraints are satisfied (by definition of linear separability). Since  $F$  is differentiable, we can now apply KKT theorem ([Appendix A](#)) to say the following : let the following conditions be satisfied. Then,  $(w^*, b^*), \alpha^*$  are the primal and dual solutions respectively.

1. (Stationarity)

$$\nabla_w \mathcal{L}(w^*, b^*, \alpha^*) = 0 \implies w^* = \sum_{i=1}^m \alpha_i^* y_i x_i \quad (3.9)$$

$$\nabla_b \mathcal{L}(w^*, b^*, \alpha^*) = 0 \implies \sum_{i=1}^m \alpha_i^* y_i = 0 \quad (3.10)$$

2. (Complementary Slackness) For all  $i \in [m]$ ,

$$\alpha_i^* (y_i (w^{*T} x_i + b^*) - 1) = 0 \implies \alpha_i^* = 0 \quad \text{or} \quad y_i (w^{*T} x_i + b^*) - 1 = 0 \quad (3.11)$$

3. (Primal Feasibility) For all  $i \in [m]$ ,

$$y_i (w^{*T} x_i + b^*) \geq 1 \quad (3.12)$$

4. (Dual Feasibility) For all  $i \in [m]$ ,

$$\alpha_i^* \geq 0 \quad (3.13)$$

Thus, the dual optimization problem can be written as

$$\max_{\alpha \in \mathbb{R}^m} \left( \min_{(w, b) \in \mathbb{R}^N \times \mathbb{R}} \mathcal{L}(w, b, \alpha) \right) \quad (3.14)$$

$$= \max_{\alpha \in \mathbb{R}^m} \mathcal{L}(w^*, b^*, \alpha) \quad (3.15)$$

$$= \max_{\alpha \in \mathbb{R}^m} \frac{1}{2} \|w^*\|^2 - \sum_{i=1}^m \alpha_i (y_i (w^{*T} x_i + b) - 1) \quad (3.16)$$

$$= \max_{\alpha \in \mathbb{R}^m} \frac{1}{2} \left\| \sum_{i=1}^m \alpha_i y_i x_i \right\|^2 - \sum_{i=1}^m \alpha_i \left( y_i \left( \sum_{j=1}^m \alpha_j y_j (x_j^T x_i) + b \right) - 1 \right) \quad (3.17)$$

$$= \max_{\alpha \in \mathbb{R}^m} \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j x_i^T x_j - b \sum_{i=1}^m \alpha_i y_i + \sum_{i=1}^m \alpha_i \quad (3.18)$$

$$= \max_{\alpha \in \mathbb{R}^m} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (x_i^T x_j) \quad (3.19)$$

(where (3.15) holds due to last remark of [Appendix A](#), (3.17) holds due to (3.9) and (3.19) holds due to (3.10)) subject to the constraints imposed on  $\alpha$  by the conditions:

$$\alpha \geq 0 \quad \text{and} \quad \sum_{i=1}^m \alpha_i y_i = 0 \quad (3.20)$$

Thus, the final dual problem is

$$\begin{aligned} & \max_{\alpha \in \mathbb{R}^m} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (x_i^T x_j) \\ & \text{subject to: } \alpha \geq 0 \quad \text{and} \quad \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned} \quad (3.21)$$

*Remark 3.1.1.* We can denote the above objective function by  $G : \mathbb{R}^m \rightarrow \mathbb{R}$ , defined by  $G(\alpha) := \mathbf{1}^T \alpha - \frac{1}{2} \alpha^T A \alpha$  for all  $\alpha \in \mathbb{R}^m$ , where  $A = V^T V$  for  $V = (x_1 y_1 \cdots x_m y_m)$ . Then,  $\nabla G = -A$ , which implies  $G$  is concave (since  $A$  is nonnegative definite) and the constraints are affine linear. Thus, the dual (maximization) problem is convex.

By (3.9),  $w^* = \sum_{i=1}^m \alpha_i^* y_i x_i$ . Thus,  $x_i$  affects  $w^*$  iff  $\alpha_i \neq 0$ . Such sample vectors  $x_i$  for which  $\alpha_i^* \neq 0$  are called **support vectors**, since the solution  $w^*$  is ‘supported’ by them. Note that for support vectors, by complementary slackness,  $y_i (w^{*T} x_i + b^*) = 1$ . Thus, support vectors lie on the hyperplane  $y(w^{*T} x + b^*) = 1$ , which is defined to be the **marginal hyperplane**.

We can recover the primal solution  $(w^*, b^*)$  from the dual solution  $\alpha^*$  as follows.

Let  $h : \mathbb{R}^N \rightarrow \mathbb{R}$  be the separating hyperplane.

$$h(x) := \text{sgn}(w^{*T}x + b^*) = \text{sgn}\left(\sum_{i=1}^m \alpha_i^* y_i (x_i^T x) + b^*\right) \quad (3.22)$$

So, we just need to find  $b^*$ . Choose any support vector  $x_i$  (index for such a point is the index of a nonzero element of  $\alpha^*$ ). Then,  $w^{*T}x_i + b^* = y_i$  (since we are working with binary classification). Thus,

$$b^* = y_i - \sum_{j=1}^m \alpha_j y_j (x_j^T x_i) \quad (3.23)$$

*Remark 3.1.2.* Note that the solution hyperplane depends only on the inner product between the sample points, and not directly on the values of the sample points themselves.

Multiplying by  $\alpha_i^* y_i$  on both sides of the above equation, and summing over  $i = 1$  to  $m$ , we get

$$\sum_{i=1}^m \alpha_i^* y_i b^* = \sum_{i=1}^m \alpha_i^* y_i^2 - \sum_{i,j=1}^m \alpha_i^* \alpha_j^* y_i y_j (x_j^T x_i) \quad (3.24)$$

We can sum over all  $i$  since the above equation holds for non-support vectors as well (since  $\alpha_i^* = 0$  for them). Thus,

$$0 = \sum_{i=1}^m \alpha_i^* - \|w^*\|_2^2 \iff \|w^*\|_2^2 = \|\alpha^*\|_1 \quad (3.25)$$

since  $\alpha^* \geq 0$ . Thus, the (square of) **hard margin**, ie. the distance between the separating hyperplane and the marginal hyperplane is given by

$$\rho^2 = \frac{1}{\|w^*\|_2^2} = \frac{1}{\|\alpha^*\|_1} \quad (3.26)$$

### 3.1.3 Leave-One-Out analysis

**Lemma 3.1.1.** *Let  $h_S$  be the hypothesis returned by some algorithm when trained on sample  $S$ . Then, for  $m \geq 2$ ,*

$$\mathbb{E}_{S \sim \mathcal{D}^m} [\hat{R}_{LOO}] = \mathbb{E}_{S' \sim \mathcal{D}^{m-1}} [\hat{R}(h_{S'})]$$

where  $\hat{R}_{LOO} := \frac{1}{m} \sum_{i=1}^m 1_{h_{S \setminus \{x_i\}}(x_i) \neq y_i}$ .

*Proof.*

$$\mathbb{E}_{S \sim \mathcal{D}^m} [\hat{R}_{LOO}] = \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{S \sim \mathcal{D}^m} [1_{h_{S \setminus \{x_i\}}(x_i) \neq y_i}] \quad (3.27)$$

$$= \mathbb{E}_{S \sim \mathcal{D}^m} [1_{h_{S \setminus \{x_1\}}(x_1) \neq y_1}] \quad (3.28)$$

$$= \mathbb{E}_{S' \sim \mathcal{D}^{m-1}, x_1 \sim \mathcal{D}} [1_{h_{S'}(x_1) \neq y_1}] \quad (3.29)$$

$$= \mathbb{E}_{S' \sim \mathcal{D}^{m-1}} \left[ \mathbb{E}_{x_1 \sim \mathcal{D}} [1_{h_{S'}(x_1) \neq y_1}] \right] \quad (3.30)$$

$$= \mathbb{E}_{S' \sim \mathcal{D}^{m-1}} [R(h_{S'})] \quad (3.31)$$

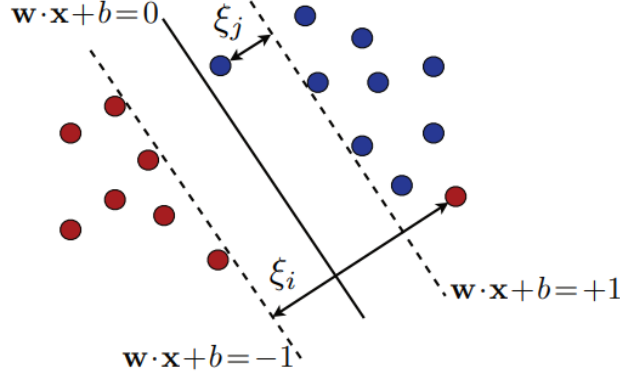
□

**Theorem 3.1.2.** *Let  $h_S$  be the hypothesis returned by SVM for a linear separable sample  $S$  of size  $m$ . Let  $N_{SV}(S)$  be the number of support vectors defining  $h_S$ . Then,*

$$\mathbb{E}_{S \sim \mathcal{D}^m} [\hat{R}(h_S)] \leq \mathbb{E}_{S \sim \mathcal{D}^{m+1}} \left[ \frac{N_{SV}(S)}{m+1} \right]$$

*Proof.* Let  $S$  be a linearly separable sample of size  $m+1$ . If  $x$  is not a support vector, then  $h_{S \setminus \{x\}} = h_S$  and thus it classifies  $x$  correctly. Contrapositively, if  $h_{S \setminus \{x\}}$  misclassifies  $x$ , then  $x$  must be a support vector. Thus, for SVM,  $\hat{R}_{LOO} \leq \frac{N_{SV}(S)}{m+1}$ , which means, using [Lemma 3.1.1](#),

$$\mathbb{E}_{S \sim \mathcal{D}^m} [R(h_S)] = \mathbb{E}_{S \sim \mathcal{D}^{m+1}} [\hat{R}_{LOO}] \leq \mathbb{E}_{S \sim \mathcal{D}^{m+1}} \left[ \frac{N_{SV}(S)}{m+1} \right] \quad (3.32)$$



**Figure 3.2:** The non-separable case (Mohri et al., 2018, pg. 87).

□

## 3.2 Non-separable Case

Now, let  $S$  be **not linearly separable**, ie. for all  $(w, b) \in (\mathbb{R}^N \setminus \{0\}) \times \mathbb{R}$ , there exists  $i \in [m]$  such that

$$y_i(w^T x_i + b) < 1 \quad (3.33)$$

Note that this is not the same as inverse of linear separability directly, ie. we are not violating  $y_i(w^T x_i + b) \geq 0$  for all  $i \in [m]$ , but rather the constraint of the primal optimization problem in the separable case, since violating that constraint directly makes more sense.

We try to bring this closer to our previous analysis by introducing **slack variables**  $\xi_1, \dots, \xi_m \geq 0$  for a given  $(w, b) \in (\mathbb{R}^N \setminus \{0\}) \times \mathbb{R}$  such that

$$y_i(w^T x_i + b) \geq 1 - \xi_i \quad (3.34)$$

for all  $i \in [m]$ . Thus,  $\xi_i$  is the ‘amount’ by which  $x_i$  violates  $y_i(w^T x_i + b) \geq 1$ . A point  $x_i$  needs to be present on the correct side of the marginal hyperplane (and not just the separating hyperplane) to not be considered an **outlier**, ie. to have  $\xi_i = 0$ .



*Remark 3.2.1.* If we omit the outliers, the training sample becomes separable for that  $(w, b)$  with the **soft margin**  $\rho = \frac{1}{\|w\|_2}$ .

### 3.2.1 Primal Optimization Problem

Drawing from the previous discussions, we state the primal optimization problem as follows.

$$\begin{aligned} \min_{w, b, \xi \in \mathbb{R}^N \times \mathbb{R} \times \mathbb{R}^m} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\ \text{subject to: } & y_i(w^T x_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \quad \forall i \in [m] \end{aligned} \quad (3.35)$$

where  $C > 0$  tunes the trade-off between margin maximization and slack minimization. This is a hyperparameter of the algorithm that can be chosen via cross validation.

### 3.2.2 Dual Optimization Problem

The Lagrangian is  $\mathcal{L} : \mathbb{R}^N \times \mathbb{R} \times \mathbb{R}_{\geq 0}^m \times \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$  defined by

$$\mathcal{L}(w, b, \xi, \alpha, \beta) := \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i - \sum_{i=1}^m \alpha_i (y_i(w^T x_i + b) - 1 + \xi_i) - \sum_{i=1}^m \beta_i \xi_i \quad (3.36)$$

for all  $(w, b, \xi, \alpha, \beta) \in \mathbb{R}^N \times \mathbb{R} \times \mathbb{R}_{\geq 0}^m \times \mathbb{R}^m \times \mathbb{R}^m$ . Thus, the dual problem is

$$\begin{aligned} \max_{\alpha, \beta \in \mathbb{R}^m} \quad & \left( \min_{(w, b, \xi) \in \mathbb{R}^N \times \mathbb{R} \times \mathbb{R}_{\geq 0}^m} \mathcal{L}(w, b, \xi, \alpha, \beta) \right) \\ \text{subject to: } & \alpha \geq 0 \text{ and } \beta \geq 0 \end{aligned} \quad (3.37)$$

Strong duality holds again by Slater's conditions ([Appendix A](#)), since the objective and the constraints are convex, the constraints are affine linear and there exists  $(w, b, \xi) \in \mathbb{R}^N \times \mathbb{R} \times \mathbb{R}_{\geq 0}^m$  where all constraints are satisfied (by definition of slack variables). Since the objective is differentiable, we can now apply KKT theorem

([Appendix A](#)) to say the following : let the following conditions be satisfied. Then,  $(w^*, b^*, \xi^*), (\alpha^*, \beta^*)$  are the primal and dual solutions respectively.

1. (Stationarity)

$$\nabla_w \mathcal{L}(w^*, b^*, \xi^*, \alpha^*, \beta^*) = 0 \implies w^* = \sum_{i=1}^m \alpha_i^* y_i x_i \quad (3.38)$$

$$\nabla_b \mathcal{L}(w^*, b^*, \xi^*, \alpha^*, \beta^*) = 0 \implies \sum_{i=1}^m \alpha_i^* y_i = 0 \quad (3.39)$$

$$\nabla_\xi \mathcal{L}(w^*, b^*, \xi^*, \alpha^*, \beta^*) = 0 \implies c\mathbf{1} = \alpha^* + \beta^* \quad (3.40)$$

2. (Complementary Slackness) For all  $i \in [m]$ ,

$$\alpha_i^* (y_i(w^{*T} x_i + b^*) - 1 + \xi_i^*) = 0 \implies \alpha_i^* = 0 \quad \text{or} \quad y_i(w^{*T} x_i + b^*) = 1 - \xi_i^* \quad (3.41)$$

$$\beta_i^* \xi_i^* = 0 \implies \beta_i^* = 0 \quad \text{or} \quad \xi_i^* = 0 \quad (3.42)$$

3. (Primal Feasibility) For all  $i \in [m]$ ,

$$y_i(w^{*T} x_i + b^*) \geq 1 - \xi_i^* \quad (3.43)$$

$$\xi_i \geq 0 \quad (3.44)$$

4. (Dual Feasibility) For all  $i \in [m]$ ,

$$\alpha_i^* \geq 0 \quad (3.45)$$

$$\beta_i^* \geq 0 \quad (3.46)$$

Thus, the dual optimization problem can be written as

$$\max_{\alpha, \beta \in \mathbb{R}^m} \left( \min_{(w, b, \xi) \in \mathbb{R}^N \times \mathbb{R} \times \mathbb{R}_{\geq 0}^m} \mathcal{L}(w, b, \xi, \alpha, \beta) \right) \quad (3.47)$$

$$= \max_{\alpha, \beta \in \mathbb{R}^m} \mathcal{L}(w^*, b^*, \xi^*, \alpha, \beta) \quad (3.48)$$

$$= \max_{\alpha, \beta \in \mathbb{R}^m} \frac{1}{2} \|w^*\|^2 + C \sum_{i=1}^m \xi_i - \sum_{i=1}^m \alpha_i (y_i (w^{*T} x_i + b) - 1 + \xi_i) - \sum_{i=1}^m \beta_i \xi_i \quad (3.49)$$

$$= \max_{\alpha, \beta \in \mathbb{R}^m} \frac{1}{2} \left\| \sum_{i=1}^m \alpha_i y_i x_i \right\|^2 + C \sum_{i=1}^m \xi_i - \sum_{i=1}^m \alpha_i \left( y_i \left( \sum_{j=1}^m \alpha_j y_j (x_j^T x_i) + b \right) - 1 + \xi_i \right) - \sum_{i=1}^m \beta_i \xi_i \quad (3.50)$$

$$= \max_{\alpha \in \mathbb{R}^m} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (x_i^T x_j) \quad (3.51)$$

(where (3.48) holds due to last remark of [Appendix A](#)) subject to the constraints imposed on  $\alpha$  by the conditions:

$$\alpha \geq 0 \quad \text{and} \quad \sum_{i=1}^m \alpha_i y_i = 0 \quad (3.52)$$

But note that, the second dual feasibility condition and the third stationarity condition implies that  $\alpha_i^* \leq C$ . Thus, the final dual problem is

$$\max_{\alpha \in \mathbb{R}^m} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j (x_i^T x_j) \quad (3.53)$$

$$\text{subject to: } \alpha \in [0, C] \quad \text{and} \quad \sum_{i=1}^m \alpha_i y_i = 0 \quad (3.54)$$

*Remark 3.2.2.* Since the objective is exactly the same as before, the dual (maximization) problem is convex by the same argument.

Again, note that if  $\alpha_i^* \neq 0$ , then by the first complementarity slackness condition  $y_i (w^{*T} x_i + b^*) = 1 - \xi_i$ . If  $\xi_i = 0$ , then  $x_i$  lies on a marginal hyperplane. Otherwise,  $x_i$  is an outlier and thus  $\beta_i^* = 0$  (by the second complementarity slackness condition), which implies that  $\alpha_i = C$  (by third stationarity condition). Thus, support vectors here can be either on marginal hyperplanes or outliers.

Again, we can recover the solution hyperplane from the dual solution  $(\alpha^*, \beta^*)$  using the exact same argument as before. Again, the solution hyperplane will depend only on the inner products between the sample points. And again, as before, the

square of the hard margin is given by the reciprocal of the sum of values of  $\alpha^*$ .

### 3.3 Deriving SVMs from margin bounds

If we consider the hypothesis set as the set of hyperplanes passing through the origin, we can get a nice upper bound on the sample Rademacher complexity.

**Theorem 3.3.1.** *Let  $S \subseteq \{x \in \mathbb{R}^N : \|x\|_2 \leq r\}$  be of size  $m$ . Let  $\mathcal{H} = \{x \mapsto w^T x : \|w\|_2 \leq \Lambda\}$ . Then,*

$$\hat{R}_S(\mathcal{H}) \leq \sqrt{\frac{r^2 \Lambda^2}{m}} \quad (3.55)$$

*Proof.*

$$\hat{R}_S(\mathcal{H}) = \frac{1}{m} \mathbb{E}_{\sigma} \left[ \sup_{\|w\| \leq \Lambda} \sum_{i=1}^m \sigma_i(w^T x_i) \right] \quad (3.56)$$

$$= \frac{1}{m} \mathbb{E}_{\sigma} \left[ \sup_{\|w\| \leq \Lambda} w^T \sum_{i=1}^m \sigma_i x_i \right] \quad (3.57)$$

$$\leq \frac{\Lambda}{m} \mathbb{E}_{\sigma} \left[ \left\| \sum_{i=1}^m \sigma_i x_i \right\| \right] \quad (3.58)$$

$$\leq \frac{\Lambda}{m} \left( \mathbb{E}_{\sigma} \left[ \left\| \sum_{i=1}^m \sigma_i x_i \right\|^2 \right] \right)^{1/2} \quad (3.59)$$

$$= \frac{\Lambda}{m} \left( \mathbb{E}_{\sigma} \left[ \sum_{i,j=1}^m \sigma_i \sigma_j (x_i^T x_j) \right] \right)^{1/2} \quad (3.60)$$

$$= \frac{\Lambda}{m} \left[ \sum_{i=1}^m \|x_i\|^2 \right]^{1/2} \quad (3.61)$$

$$= \frac{\Lambda}{m} \sqrt{m r^2} \quad (3.62)$$

where we used Cauchy-Schwartz on (3.58) and Jensen's inequality on (3.59) (Appendix D).  $\square$

*Remark 3.3.1.* The good thing about the above generalization bounds is that they depend only on the margin  $\rho$ , and not on the feature space dimension  $N$ .

*Remark 3.3.2.* Using last theorem of [section 2.3](#) and [Theorem 3.3.1](#), we get the following : let  $S \subseteq \{x \in \mathbb{R}^N : \|x\|_2 \leq r\}$  be of size  $m$ ,  $\mathcal{H} = \{x \mapsto w^T x : \|w\|_2 \leq 1\}$ ,  $r > 0$  and  $\delta > 0$ . Then, with probability at least  $1 - \delta$ , for all  $h \in \mathcal{H}$  and  $\rho \in (0, r]$ ,

$$R(h) \leq \frac{1}{m} \sum_{i=1}^m \max \left\{ 0, 1 - \frac{y_i(w^T x_i)}{\rho} \right\} + 4\sqrt{\frac{r^2}{m\rho^2}} + \sqrt{\frac{\ln(\log_2(2r/\rho))}{m}} + 3\sqrt{\frac{\ln(4/\delta)}{2m}} \quad (3.63)$$

since  $\Phi_\rho(u) \leq \max\{0, 1 - \frac{u}{\rho}\}$  for all  $u \in \mathbb{R}$ . We can write this inequality as

$$R(h) \leq \frac{1}{m} \sum_{i=1}^m \max \{0, 1 - y_i(w^T x_i)\} + 4\sqrt{\frac{r^2}{m\rho^2}} + \sqrt{\frac{\ln(\log_2(2r/\rho))}{m}} + 3\sqrt{\frac{\ln(4/\delta)}{2m}} \quad (3.64)$$

subject to  $\|w\|_2 \leq 1/\rho$ . Now, if we want to find a hypothesis  $h$  for which  $R(h)$  is small in this scenario, it would make sense to minimize the above upper bound. This can be done by minimizing with respect to  $w$  (such that  $\|w\| \leq 1/\rho$ ) and choosing  $\rho$  by cross-validation. Thus, we would want to solve

$$\min_{w \in \mathbb{R}^N} \lambda \|w\|^2 + \frac{1}{m} \sum_{i=1}^m \max \{0, 1 - y_i(w^T x_i)\} \quad (3.65)$$

and select  $\lambda$  by cross-validation. So, we are minimizing the norm of  $w$  along with minimizing the points for which  $y_i(w^T x_i) < 1$ , which is exactly the SVM primal problem (with  $b = 0$ ). Thus, minimizing the above margin-based generalization bound results in the SVM algorithm!

### 3.4 Multi-class SVM

**Theorem 3.4.1** (RC of multi-class kernel-based hypothesis). *Let  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  be a PDS kernel and  $\Phi : \mathcal{X} \rightarrow \mathbb{H}$  be the feature mapping associated to  $K$ . Let there exist  $r > 0$  such that  $K(x, x) \leq r^2$  for all  $x \in \mathcal{X}$ . Based on  $k$  weight vectors*

$w_1, \dots, w_k \in \mathbb{H}$ , the class associated to  $x \in \mathcal{X}$  is given by

$$\operatorname{argmax}_{y \in \mathcal{Y}} \langle w_y, \Phi(x) \rangle \quad (3.66)$$

For any  $p \geq 1$ , we define the  $L_{\mathbb{H},p}$  group norm of  $W$  by

$$\|W\|_{\mathbb{H},p} := \left( \sum_{l=1}^k \|w_l\|_{\mathbb{H}}^p \right)^{1/p} \quad (3.67)$$

Let  $W = (w_1, \dots, w_k)^T$  and  $\Lambda > 0$ . Then, for all  $p \geq 1$ , we define

$$\mathcal{H}_{K,p} := \{(x, y) \mapsto \langle w_y, \Phi(x) \rangle : \|W\|_{\mathbb{H},p} \leq \Lambda\} \quad (3.68)$$

Then, for all  $m \in \mathbb{N}$ ,

$$\mathfrak{R}_m(\Pi_1(\mathcal{H}_{K,p})) \leq \sqrt{\frac{r^2 \Lambda^2}{m}} \quad (3.69)$$

*Proof.* Let  $S = (x_1, \dots, x_m)$  be a sample. For all  $l \in [k]$ ,  $\|w_l\|_{\mathbb{H}} \leq \|W\|_{\mathbb{H},p}$ . Thus,  $\|w_l\|_{\mathbb{H}} \leq \Lambda$  for all  $l \in [k]$ . Thus,

$$\mathfrak{R}_m(\Pi_1(\mathcal{H}_{K,p})) = \frac{1}{m} \mathbb{E}_{S,\sigma} \left[ \sup_{y \in \mathcal{Y}, \|W\| \leq \Lambda} w_y^T \sum_{i=1}^m \sigma_i \Phi(x_i) \right] \quad (3.70)$$

$$\leq \frac{1}{m} \mathbb{E}_{S,\sigma} \left[ \sup_{y \in \mathcal{Y}, \|W\| \leq \Lambda} \|w_y\|_{\mathbb{H}} \left\| \sum_{i=1}^m \sigma_i \Phi(x_i) \right\|_{\mathbb{H}} \right] \quad (3.71)$$

$$\leq \frac{\Lambda}{m} \mathbb{E}_{S,\sigma} \left[ \left\| \sum_{i=1}^m \sigma_i \Phi(x_i) \right\|^2 \right]^{1/2} \quad (3.72)$$

$$\leq \frac{\Lambda}{m} \mathbb{E}_S \left[ \left\| \sum_{i=1}^m \Phi(x_i) \right\|^2 \right]^{1/2} \quad (3.73)$$

$$\leq \frac{\Lambda}{m} \mathbb{E}_S \left[ \left\| \sum_{i=1}^m K(x_i, x_i) \right\|^2 \right]^{1/2} \quad (3.74)$$

$$\leq \frac{\Lambda}{m} \sqrt{mr^2} \quad (3.75)$$

where we use Cauchy-Schwartz in (3.71) and Jensen's inequality in (3.72) (Appendix D).  $\square$

**Corollary 3.4.1.1.** *Let  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  be a PDS kernel and  $\Phi : \mathcal{X} \rightarrow \mathbb{H}$  be the feature mapping associated to  $K$ . Let there exist  $r > 0$  such that  $K(x, x) \leq r^2$  for all  $x \in \mathcal{X}$ . Then, for all  $\delta > 0$ , with probability at least  $1 - \delta$ , for all  $h \in \mathcal{H}_{k,2}$ ,*

$$R(h) \leq \frac{1}{m} \sum_{i=1}^m \xi_i + 4k \sqrt{\frac{r^2 \Lambda^2}{m}} + \sqrt{\frac{\log(1/\delta)}{2m}} \quad (3.76)$$

where, for all  $i \in [m]$ ,

$$\xi_i := \max \left\{ 1 - \left( \langle w_{y_i}, \Phi(x_i) \rangle - \max_{y' \neq y_i} \langle w_{y'}, \Phi(x_i) \rangle \right), 0 \right\} \quad (3.77)$$

The min goes away from  $\hat{R}_{S,1}$  assuming all points in the sample are classified correctly.

The multi-class SVM algorithm simply minimizes this upper bound:

$$\min_{W, \xi \in \mathbb{H}^k \times \mathbb{R}^m} \frac{1}{2} \sum_{l=1}^k \|w_l\|^2 + C \sum_{i=1}^m \xi_i \quad (3.78)$$

subject to:  $\langle w_{y_i}, \Phi(x_i) \rangle \geq \langle w_l, \Phi(x_i) \rangle + 1 - \xi_i$  and  $\xi_i \geq 0 \forall i \in [m]$ ,  $l \in \mathcal{Y} \setminus \{y_i\}$

So, we want to minimize  $\Lambda$  and  $\sum_{i=1}^m \xi_i$ , which can be done by minimizing the above objective. We do this subject to the constraint that all sample points are correctly classified, which is true iff  $\xi_i \leq 1$ , which is true iff the above constraints are satisfied.

Moreover, the above problem is convex and KKT conditions hold at optimum.

Thus, the dual optimization problem can be written as

$$\max_{\alpha \in \mathbb{R}^{m \times k}} \sum_{i=1}^m \alpha_i^T e_{y_i} - \frac{1}{2} \sum_{i=1}^m (\alpha_i^T \alpha_j) K(x_i, x_j) \quad (3.79)$$

subject to:  $0 \leq \alpha_{iy_i} \leq C$  and  $\alpha_{ij} \leq 0 \forall j \neq y_i$  and  $\alpha_i^T 1 = 0, \forall i \in [m]$

But, both size of the solution and number of constraints are  $\Omega(mk)$ . Thus, we can also instead decompose the problem into  $m$  disjoint sets of constraints instead.

### 3.5 Support Vector Regression

This is SVMs for regression. We shall fit a ‘tube’ of width  $\varepsilon > 0$ , and penalize linearly the points that are outside the tube.

Let  $\mathcal{H} := \{x \mapsto w^T \Phi(x) + b : w \in \mathbb{R}^N, b \in \mathbb{R}\}$ , where  $\Phi$  is a feature map corresponding to some PDS kernel  $K$ . The primal SVR problem is as follows.

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m |y_i - (w^T \Phi(x_i) + b)|_\varepsilon \quad (3.80)$$

where  $|y' - y|_\varepsilon := \max\{0, |y' - y| - \varepsilon\}$  for all  $y, y' \in \mathcal{Y}$ . This leads to solutions being sparse.

This can be written equivalently as

$$\min_{w,b,\xi,\xi'} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m (\xi_i + \xi'_i) \quad (3.81)$$

$$\text{subject to: } w^T \Phi(x_i) + b - y_i \leq \varepsilon + \xi_i \quad (3.82)$$

$$y_i - (w^T \Phi(x_i) + b) \leq \varepsilon + \xi'_i$$

$$\xi_i \geq 0, \xi'_i \geq 0, \forall i \in [m]$$

The dual optimization problem becomes

$$\max_{\alpha, \alpha'} -\varepsilon(\alpha' + \alpha)^T \mathbf{1} + (\alpha' - \alpha)^T y - \frac{1}{2}(\alpha' - \alpha)^T K(\alpha' - \alpha)$$

$$\text{subject to: } \alpha, \alpha' \in [0, C]$$

$$(\alpha - \alpha')^T \mathbf{1} = 0 \quad (3.83)$$

The solution becomes

$$h(x) = \sum_{i=1}^m (\alpha'_i - \alpha_i) K(x_i, x) + b \quad (3.84)$$



where

$$b = - \sum_{i=1}^m (\alpha'_i - \alpha_i) K(x_i, x_j) + y_j + \varepsilon \quad (3.85)$$

for  $x_j$  such that  $\alpha_j \in (0, C)$ , or

$$b = - \sum_{i=1}^m (\alpha'_i - \alpha_i) K(x_i, x_j) + y_j - \varepsilon \quad (3.86)$$

for  $x_j$  such that  $\alpha'_j \in (0, C)$ .

By complementarity conditions, for all  $i \in [m]$ ,

$$\alpha_i (w^T \Phi(x_i) + b - y_i - \varepsilon - \xi_i) = 0 \quad (3.87)$$

$$\alpha'_i (w^T \Phi(x_i) + b - y_i + \varepsilon + \xi'_i) = 0 \quad (3.88)$$

So, if  $\alpha_i \neq 0$  or  $\alpha'_i \neq 0$ , then  $x_i$  is a support vector and either  $w^T \Phi(x_i) + b - y_i - \varepsilon = \xi_i$  or  $w^T \Phi(x_i) + b - y_i + \varepsilon = -\xi'_i$ . Thus, support vectors lie outside the  $\varepsilon$ -tube. Note that at most one of  $\alpha_i$  or  $\alpha'_i$  is nonzero, since  $x_i$  is either overestimated or underestimated. For points in the tube,  $\alpha_i = \alpha'_i = 0$ . So, if the number of points in the tube is large, the hypothesis will be sparse but innaccurate.

## 3.6 Kernels

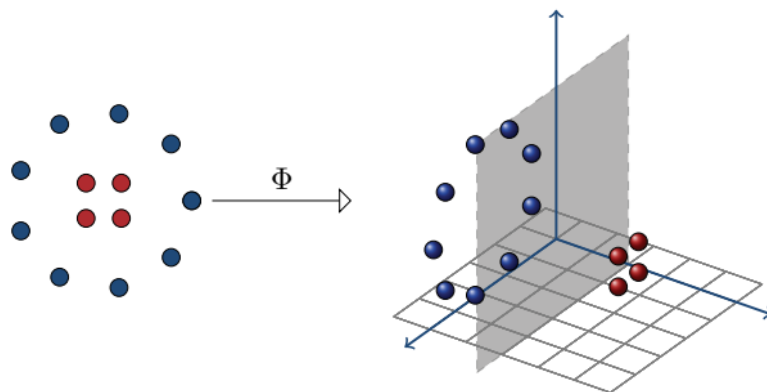
SVMs, by definition, can create only linear decision boundaries. But most real world problems will need highly nonlinear functions to solve. Kernels are a very elegant way to extending the functionality of SVMs.

**Definition 3.6.1** (Kernel). A kernel is a map  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ .

We want to define  $K$  such that for all  $x, x' \in \mathcal{X}$ ,

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle \quad (3.89)$$

for some  $\Phi : \mathcal{X} \rightarrow \mathbb{H}$ , where  $(\mathbb{H}, \langle \cdot \rangle)$  is a Hilbert space. Thus,  $K(x, x')$  is the inner



**Figure 3.3:** Example of a non-linearly separable sample becoming linearly separable by projecting to a higher dimensional space (Mohri et al., 2018, pg. 107).

product between  $x, x'$  when projected to a Hilbert space. Intuitively,  $K(x, x')$  tells the ‘similarity’ between  $x, x'$ .

Usually,  $\Phi$  (which is called the **feature mapping**) is a nonlinear map from  $\mathcal{X}$  to a higher dimensional space  $\mathbb{H}$  where linear separation is possible.

Kernels are useful in practice because they are often computationally more efficient than applying  $\Phi$  and then taking the inner product. Also, kernels are often flexible in the sense that we need not define  $\Phi$  explicitly every time.

But how can we ensure that we can define  $K$  such that the above equation holds? To solve this problem, we need to define PDS kernels.

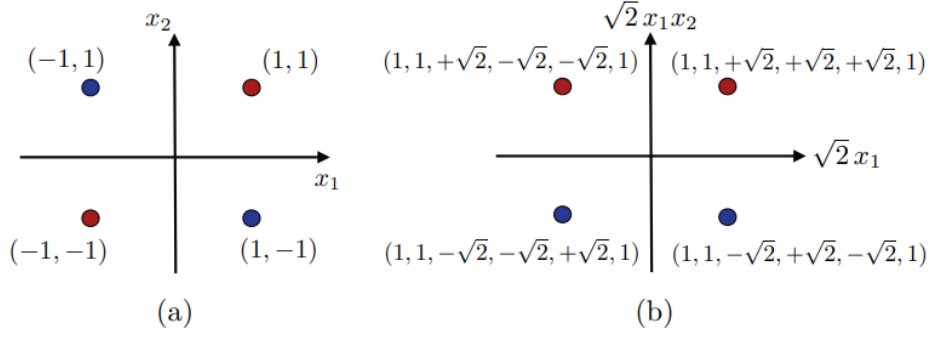
### 3.6.1 PDS Kernels

**Definition 3.6.2** (PDS Kernel). A kernel  $K$  is called PDS (Positive-Definite Symmetric) if for all  $\{x_1, \dots, x_m\} \subseteq \mathcal{X}$ ,

$$\mathbf{K} := [K(x_i, x_j)]_{i,j=1}^m \quad (3.90)$$

is symmetric positive semi-definite.  $\mathbf{K}$  is called the **kernel matrix**.

*Remark 3.6.1.*  $K$  is positive-definite iff  $\mathbf{K}$  is positive semi-definite.



**Figure 3.4:** XOR classification problem becomes linearly separable using the polynomial kernel  $K(x, x') := (x^T x' + 1)^2$  (Mohri et al., 2018, pg. 109).

**Example 3.6.1.** The polynomial kernel  $K : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$  defined by  $K(x, x') := (x^T x' + 1)^2$  for all  $x, x' \in \mathbb{R}$  can make the XOR problem linearly separable (figure 3.4).

**Lemma 3.6.1** (Cauchy-Schwartz for PDS kernels). *Let  $K$  be a PDS kernel. Then,*

$$K(x, x')^2 \leq K(x, x)K(x', x') \quad (3.91)$$

for all  $x, x' \in \mathcal{X}$ .

*Proof.* Let  $x, x' \in \mathcal{X}$ . Consider

$$\mathbf{K} = \begin{pmatrix} K(x, x) & K(x, x') \\ K(x', x) & K(x', x') \end{pmatrix} \quad (3.92)$$

Then,  $K$  is PDS iff  $\mathbf{K}$  is positive-definite symmetric, which implies  $\det(\mathbf{K}) \geq 0$ .  $\square$

**Theorem 3.6.2** (RKHS). *Let  $K$  be a PDS kernel. Then, there exists a Hilbert space  $(\mathbb{H}, \langle \cdot, \cdot \rangle)$  and a feature mapping  $\Phi : \mathcal{X} \rightarrow \mathbb{H}$  such that*

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle \quad (3.93)$$

for all  $x, x' \in \mathcal{X}$ . Moreover,  $\mathbb{H}$  is called the RKHS for  $K$  and it has the reproducing

property : for all  $h \in \mathbb{H}$ ,  $x \in \mathcal{X}$ ,

$$h(x)(\cdot) = \langle h, K(x, \cdot) \rangle$$

*Proof.* For all  $x \in \mathcal{X}$ , define  $\Phi(x) : \mathcal{X} \rightarrow \mathbb{R}^{\mathcal{X}}$  as

$$\Phi(x)(x') := K(x, x') \quad (3.94)$$

for all  $x' \in \mathcal{X}$ . Define

$$\mathbb{H}_0 := \left\{ \sum_{i \in I} a_i \Phi(x_i) : a_i \in \mathbb{R}, x_i \in \mathcal{X}, |I| < \infty \right\} \quad (3.95)$$

For all  $f, g \in \mathbb{H}_0$  of forms  $f = \sum_{i \in I} a_i \Phi(x_i)$ ,  $g = \sum_{j \in J} b_j \Phi(x'_j)$ , define  $\langle \cdot, \cdot \rangle$  on  $\mathbb{H}_0 \times \mathbb{H}_0$  as

$$\langle f, g \rangle := \sum_{i, j \in I} a_i b_j K(x_i, x'_j) \quad (3.96)$$

$$= \sum_{j \in J} b_j \sum_{i \in I} a_i \Phi(x_i)(x'_j) = \sum_{j \in J} b_j f(x'_j) \quad (3.97)$$

$$= \sum_{i \in I} a_i \sum_{j \in J} b_j \Phi(x'_j)(x_i) = \sum_{i \in I} a_i g(x_i) \quad (3.98)$$

Thus,  $\langle \cdot, \cdot \rangle$  is symmetric and  $f, g$  do not depend on their particular representations.

Moreover,  $\langle \cdot, \cdot \rangle$  is bilinear. Now, for all  $f = \sum_{i \in I} a_i \Phi(x_i) \in \mathbb{H}_0$ ,

$$\langle f, f \rangle = \sum_{i, j \in I} a_i a_j K(x_i, x_j) = c^T \mathbf{K} c \geq 0 \quad (3.99)$$

with  $c = (a_1, \dots, a_{|I|})^T$ . Also, for all  $f_1, \dots, f_m \in \mathbb{H}_0$ , for all  $c_1, \dots, c_m \in \mathbb{R}$ ,

$$c^T P c = \sum_{i, j=1}^m c_i c_j \langle f_i, f_j \rangle = \left\langle \sum_{i=1}^m c_i f_i, \sum_{j=1}^m c_j f_j \right\rangle \geq 0 \quad (3.100)$$

Thus,  $\langle \cdot, \cdot \rangle$  is a PDS kernel on  $\mathbb{H}_0$ . Now, using [Lemma 3.6.1](#), for all  $f \in \mathbb{H}_0$  and

for all  $x \in \mathcal{X}$ ,

$$\langle f, \Phi(x) \rangle^2 \leq \langle f, f \rangle \langle \Phi(x), \Phi(x) \rangle \quad (3.101)$$

And, for all  $f = \sum_{i \in I} a_i \Phi(x_i) \in \mathbb{H}_0$ , for all  $x \in \mathcal{X}$ ,

$$f(x) = \sum_{i \in I} a_i K(x_i, x) = \langle \sum_{i \in I} a_i \Phi(x_i), \Phi(x) \rangle = \langle f, \Phi(x) \rangle \quad (3.102)$$

Thus,  $(f(x))^2 \leq \langle f, f \rangle K(x, x)$  for all  $x \in \mathcal{X}$ . Thus  $\langle \cdot, \cdot \rangle$  defined an inner product on  $\mathbb{H}_0$ . To make  $\mathbb{H}_0$  complete, we can add all limits of Cauchy sequences. And, for all  $x \in \mathcal{X}$ ,  $f \mapsto \langle f, \Phi(x) \rangle$  is Lipschitz and thus continuous.  $\square$

*Remark 3.6.2.* Note that  $\mathbb{H}$  is a subset of functions. Thus, points in  $\mathcal{X}$  are projected to functionals over  $\mathcal{X}$ . The functional associated to a point  $x \in \mathcal{X}$  gives the similarity (in terms of the kernel  $K$ ) of  $x$  and every other point in  $\mathcal{X}$ .

### 3.6.2 SVM with PDS kernels

Simply replace  $x^T x'$  by  $K(x, x')$  for a PDS kernel  $K$ . For example, the dual problem of SVMs in non separable case becomes the following.

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^m} \quad & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \text{subject to: } & \alpha \in [0, C] \quad \text{and} \quad \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned} \quad (3.103)$$

And, the solution becomes

$$h(x) = \text{sgn} \left( \sum_{i=1}^m \alpha_i^* y_i K(x_i, x) + b^* \right) \quad (3.104)$$

with  $b^* = y_i - \sum_{j=1}^m \alpha_j^* y_j K(x_j, x_i)$  for some  $x_i$  with  $\alpha_i^* \in (0, C)$ .

### 3.6.3 Representer theorem

Solution of many regularization-based optimization problems (for example, regularization based SRM) can be written in terms of PDS kernels.

**Theorem 3.6.3** (Representer). *Let  $K$  be a PDS kernel and  $\mathbb{H}$  be its RKHS. Let the objective  $G : \mathbb{R} \rightarrow \mathbb{R}$  be nondecreasing. Let the regularization loss/penalty be  $L : \mathbb{R}^m \rightarrow \mathbb{R} \cup \{+\infty\}$ . Then, the problem*

$$\operatorname{argmin}_{h \in \mathbb{H}} G(\|h\|_{\mathbb{H}}) + L(h(x_1), \dots, h(x_m)) \quad (3.105)$$

*has a solution of the form  $h^*(\cdot) = \sum_{i=1}^m \alpha_i K(x_i, \cdot)$  for some  $\alpha_i \in \mathbb{R}$ . Furthermore, if  $G$  is increasing, then any solution has this form.*

*Proof.* Define  $\mathbb{H}_1 := \operatorname{span}\{K(x_i, \cdot) : i \in [m]\}$ . For all  $h \in \mathbb{H}$ ,  $h = h_1 + h^\perp$  for some  $h_1 \in \mathbb{H}_1, h^\perp \in \mathbb{H}_1^\perp$  where  $\mathbb{H}_1^\perp$  is the orthogonal complement of  $\mathbb{H}_1$ . So,

$$G(\|h\|_{\mathbb{H}}) \leq G(\sqrt{\|h_1\|_{\mathbb{H}}^2 + \|h_1^\perp\|_{\mathbb{H}}^2}) = G(\|h_1\|_{\mathbb{H}}) \quad (3.106)$$

By representation property, for all  $i \in [m]$ ,

$$h(x_i) = \langle h, K(x_i, \cdot) \rangle = \langle h_1, K(x_i, \cdot) \rangle = h_1(x_i) \quad (3.107)$$

Thus,  $L(h(x_1), \dots, h(x_m)) = L(h_1(x_1), \dots, h_1(x_m))$ , which implies  $F(h_1) \leq F(h)$ .

Moreover, this inequality becomes strict if  $G$  is strictly increasing.  $\square$

## Chapter 4

# IMPROVING PERFORMANCE

In practice, an algorithm does not work perfectly well right out of the box. There are multiple issues that need to be dealt with. Some of them are as follows.

1. The hyperparameters of the model, which are the parameters that are not decided by the model but instead given by the user (for example, the hyperparameter  $C$  in the non-separable case of SVMs ([section 3.2](#))), need to be tuned properly. We can do so by cross-validation ([section 4.1](#)).
2. Many times, it is hard to find an algorithm that works good for a problem, but not so hard to find an algorithm that works better than random chance. It is possible to combine these better-than-random algorithms to create a good algorithm. We can do so by boosting ([section 4.2](#)).
3. Usually, real-world data is very high-dimensional. Most of these dimensions are usually not so useful for the problem at hand. We can get rid of these dimensions using various dimensionality reduction techniques like PCA ([section 4.3](#)). Moreover, we will talk about the Johnson-Lindenstrauss lemma, which says that we can always find a dimensionality reduction function that is close to being isometric.

## 4.1 Cross-Validation

Let the sample  $S$  of size  $m$  be divided into a subsample  $S_1$  of size  $(1 - \alpha)m$  and a subsample  $S_2$  of size  $\alpha m$  (where  $\alpha \in (0, 1)$  is typically chosen to be relatively small).  $S_1$  is for training,  $S_2$  is for validation.

For any  $k \in \mathbb{N}$ , define  $h_{S_1, k}^{\text{ERM}}$  to be the ERM (Empirical-Risk Minimization) solution on  $S_1$  using  $\mathcal{H}_k$ . Then, the cross-validation solution is defined as

$$h_S^{\text{CV}} := \underset{h \in \{h_{S_1, k}^{\text{ERM}} : k \in \mathbb{N}\}}{\text{argmin}} \quad \hat{R}_{S_2}(h) \quad (4.1)$$

However, an algorithm trained on  $(1 - \alpha)m$  points may have a significantly worse performance than when trained on  $m$  points. Avoiding this ‘phase transition’ issue can be done by  $n$ -fold cross validation.

Let  $\theta$  denote the vector of hyperparameters of the algorithm. Partition  $S$  into  $n$  subsamples, or folds  $S_1, \dots, S_n$ . The  $i^{\text{th}}$  fold is a labelled subsample of size  $m_i$ . Then, for all  $i \in [n]$ , the learning algorithm is trained on all but the  $i^{\text{th}}$  fold to generate  $h_i$ , and performance of  $h_i$  is tested on the  $i^{\text{th}}$  fold.

Finally, that  $\theta$  is chosen (from a small number of possible values) which minimizes the CV error:

$$\hat{R}_{\text{CV}}(\theta) := \frac{1}{n} \sum_{i=1}^n \hat{R}_{S_i}(h_i) \quad (4.2)$$

## 4.2 Boosting

Boosting refers to ‘combining’ several better-than-random hypotheses to create a hypothesis that is good. As stated in the introduction, it is hard to find an algorithm that works good for a problem, but not so hard to find an algorithm that works better than random chance. For classification, the better-than-random hypotheses used are also called base classifiers.



AdaBoost is a classic method of boosting. Its name is short for ‘Adaptive Boosting’, since the algorithm adapts to the accuracy of the base classifiers. Let  $\mathcal{H}$  be the class of base classifiers. We shall assume them to be binary classifiers. For a given sample  $S = ((x_1, y_1), \dots, (x_m, y_m))$  and number of iterations  $T \in \mathbb{N}$ , define the AdaBoost algorithm as follows:

---

**Algorithm 1:** AdaBoost Algorithm
 

---

**Input** : Training set  $S = ((x_1, y_1), \dots, (x_m, y_m))$

**Output:** Final classifier  $f$

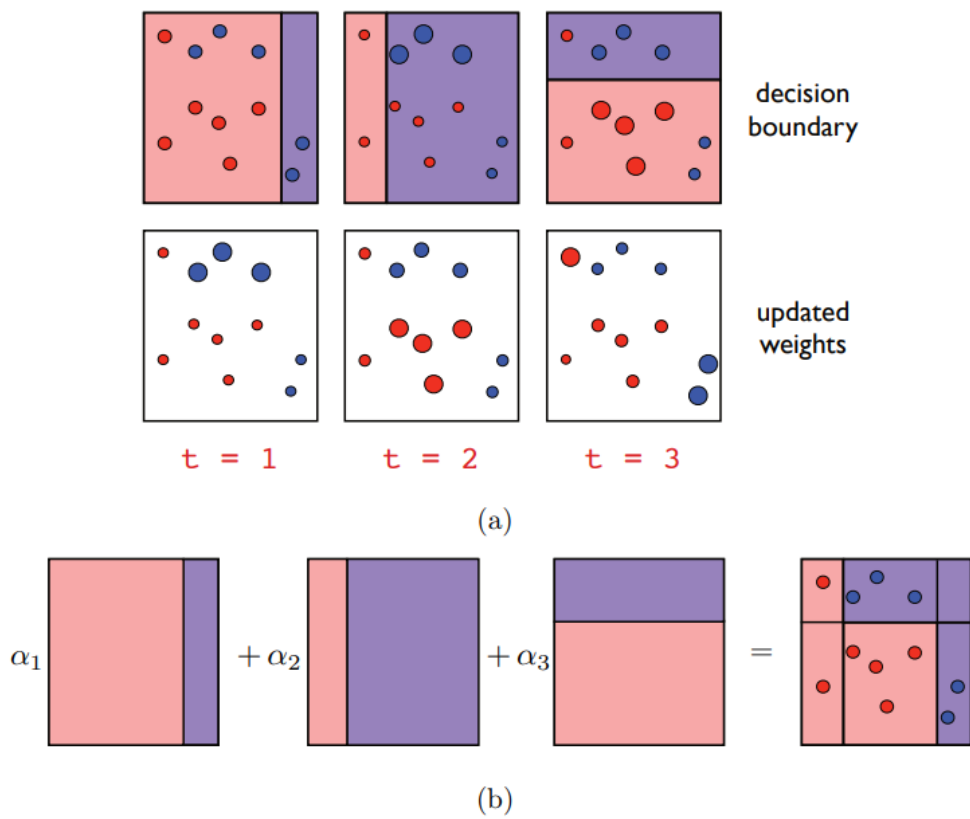
```

1 for  $i \leftarrow 1$  to  $m$  do
2    $\mathcal{D}_1(i) \leftarrow \frac{1}{m}$ 
3 for  $t \leftarrow 1$  to  $T$  do
4   Train base classifier  $h_t \in \mathcal{H}$  with error  $\epsilon_t = \mathbb{P}_{i \sim \mathcal{D}_t}[h_t(x_i) \neq y_i]$ 
5    $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$ 
6    $Z_t \leftarrow 2 [\epsilon_t(1 - \epsilon_t)]^{\frac{1}{2}}$ 
7   for  $i \leftarrow 1$  to  $m$  do
8      $\mathcal{D}_{t+1}(i) \leftarrow \frac{\mathcal{D}_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$ 
9  $f \leftarrow \sum_{t=1}^T \alpha_t h_t$ 
10 return  $f$ 
    
```

---

Note that  $\mathcal{D}_t$  is the pmf over  $[m]$  before iteration  $t$ .  $Z_t$  is simply the normalizing constant for the  $t^{\text{th}}$  iteration.

The algorithm initializes the distribution over sample indices to be uniform using the loop in line 1. Then, for every iteration  $t$ , it chooses the hypothesis that performs best on the sample with respect to the empirical error being weighted by the distribution  $\mathcal{D}_t$ . Thus, if the weight (or probability) of the  $i^{\text{th}}$  sample is more, it would contribute more to the error, and thus the minimization will prioritize making sure that the  $i^{\text{th}}$  sample is classified correctly. Then,  $\alpha_t$  is defined such that if  $\epsilon_t < 1/2$  (which will always be the case as the base classifiers are weak learners, ie. better than random), then  $\alpha_t > 0$ . This ensures that when we define the distribution for the next iteration in line 8, we will give more weight to the



**Figure 4.1:** Example of AdaBoost combining linear classifiers (Mohri et al., 2018, pg. 147).

points classified incorrectly in the current iteration.

Thus, with each iteration, we keep on giving more weight to the hard-to-classify sample points and keep on finding hypotheses that can classify these points correctly. We then end with our final hypothesis as the weighted average of the hypothesis from all of the iterations, weighted by how good their error on the sample was.

As a generalization of the above algorithm, we can try finding  $h_t$  using  $\mathcal{D}_t$ . The above algorithm also works when  $\mathcal{H} \subseteq [-1, +1]^{\mathcal{X}}$  (or any bounded subset of  $\mathbb{R}$ ) with the appropriate (minor) changes.

*Remark 4.2.1.* Note that

$$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t}{\varepsilon_t} \iff (1 - \varepsilon_t)e^{-\alpha_t} = \varepsilon_t e^{\alpha_t}$$

Thus, at each iteration, AdaBoost assigns equal distribution mass to correct and incorrect points. This is not contradicting the above reasoning, since we start with the number of correctly classified points larger than the number of incorrectly classified points (as we are working with weak learners). So, at each iteration, we will be giving more mass to at least one incorrect point than to any correct point.

**Theorem 4.2.1** (AdaBoost empirical error bound).

$$\hat{R}_S(f) \leq \prod_{t=1}^T Z_t \leq \exp \left( -2 \sum_{t=1}^T \left( \frac{1}{2} - \varepsilon_t \right)^2 \right)$$

*Proof.*

$$\hat{R}_S(f) = \frac{1}{m} \sum_{i=1}^m 1_{y_i f(x_i) \leq 0} \tag{4.3}$$

$$\leq \sum_{i=1}^m e^{-y_i f(x_i)} \tag{4.4}$$

$$= \frac{1}{m} \sum_{i=1}^m \left[ \mathcal{D}_{T+1}(i) m \prod_{t=1}^T Z_t \right] \tag{4.5}$$

$$= \prod_{t=1}^T Z_t \quad (4.6)$$

$$= \prod_{t=1}^T 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \quad (4.7)$$

$$= \prod_{t=1}^T \sqrt{1 - 4\left(\frac{1}{2} - \varepsilon_t\right)^2} \quad (4.8)$$

$$\leq \prod_{t=1}^T \sqrt{e^{-4\left(\frac{1}{2} - \varepsilon_t\right)^2}} \quad (4.9)$$

$$= \prod_{t=1}^T \sqrt{e^{-2\left(\frac{1}{2} - \varepsilon_t\right)^2}} \quad (4.10)$$

$$= \exp\left(-2 \sum_{t=1}^T \left(\frac{1}{2} - \varepsilon_t\right)^2\right) \quad (4.11)$$

where (4.4) holds because  $1_{u \leq 0} \leq e^{-u}$  for all  $u \in \mathbb{R}$ , (4.5) holds by ‘unfolding’ the recursive definition of  $\mathcal{D}_{t+1}$  from algorithm 1 and (4.9) holds because  $1 - x \leq e^{-x}$  for all  $x \in \mathbb{R}$ .  $\square$

*Remark 4.2.2.* We motivate choosing  $\alpha_t$  as the one that minimizes the above upper bound on the empirical error. It is sufficient to instead minimize  $Z_t$ . Since it can be shown easily that  $Z_t = (1 - \varepsilon_t)e^{-\alpha_t} + \varepsilon_t e^{\alpha_t}$ , we define  $\varphi : \mathbb{R} \rightarrow \mathbb{R}$  by  $\varphi(\alpha) := (1 - \varepsilon_t)e^{-\alpha} + \varepsilon_t e^{\alpha}$  for all  $\alpha \in \mathbb{R}$ . We thus want to find the minimizer of  $\varphi$ .

Since  $\varphi$  is convex and differentiable, finding the minima is as simple as finding solution of  $\varphi'(\alpha) = 0$ , which is  $\alpha = \frac{1}{2} \log \frac{1-\varepsilon_t}{\varepsilon_t}$ . Note that this is not circular reasoning as we do not need the explicit definition of  $\alpha_t$  to either find the first inequality of the above theorem or to find the above expression for  $Z_t$ .

*Remark 4.2.3.* AdaBoost coincides with the coordinate descent algorithm applied to  $F : \mathbb{R}^T \rightarrow \mathbb{R}$  defined by

$$F(\alpha) := \frac{1}{m} \sum_{i=1}^m \exp\left(-y_i \sum_{t=1}^T \alpha_t h_t\right)$$

for all  $\alpha \in \mathbb{R}^T$ . Also, if we use the logistic loss instead of exponential loss, ie.

$$\tilde{F}(\alpha) := \frac{1}{m} \sum_{i=1}^m \log_2 \left( 1 + e^{y_i \sum_{t=1}^T \alpha_t h_t} \right)$$

then coordinate descent will coincide with logistic regression.

*Remark 4.2.4.* Practically, boosting is often applied using decision trees of depth 1 (called stumps), which are just a linear partition of the space. Though they are good in practice, they are not exactly weak learners (since they cannot classify XOR with at least 50% accuracy).

### 4.2.1 An anomaly

Define

$$\mathcal{F}_T := \left\{ \text{sgn} \left( \sum_{t=1}^T \alpha_t h_t \right) : \alpha_t \geq 0, h_t \in \mathcal{H} \forall t \in [T] \right\} \quad (4.12)$$

for some  $T \in \mathbb{N}$  and some hypothesis set  $\mathcal{H}$ . Then,

$$\text{VCdim}(\mathcal{F}_T) = 2(d+1)(T+1) \log_2((T+1)e) = \mathcal{O}(dT \log T) \quad (4.13)$$

Thus, it appears that for large  $T$ , AdaBoost should lead to overfitting. However, empirically, it does not!

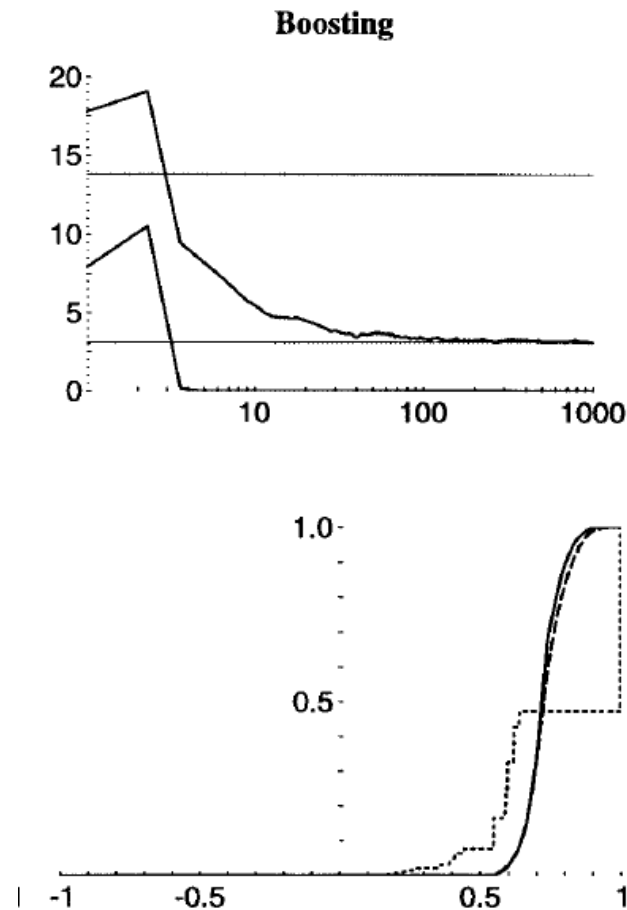
To explain this phenomena, we define the  $L_1$  geometric margin of  $f = \sum_{t=1}^T \alpha_t h_t$  (with  $\alpha \neq 0$ ) at  $x \in \mathcal{X}$  to be

$$\rho_f(x) := \frac{|f(x)|}{\|\alpha\|_1} = \frac{|\alpha^T h(x)|}{\|\alpha\|_1} \quad (4.14)$$

And,  $\rho_f := \min_{i \in [m]} \rho_f(x_i)$ . If  $x$  is correctly classified by  $f$ , then

$$\rho_f(x) = \frac{yf(x)}{\|\alpha\|_1} =: y\bar{f}(x) \quad (4.15)$$

**Theorem 4.2.2.** *Let  $\mathcal{H}$  be a hypothesis set of real-valued functions,  $S$  be a sample.*



**Figure 4.2:** Error curve and margin distribution graph for boosting C4.5 (decision-tree learning algorithm). Learning curve is shown directly above corresponding margin distribution graph. The learning curve figure shows the training and test error curves (lower and upper curves, respectively) of the combined classifier as a function of the number of classifiers combined. Horizontal lines indicate the test error rate of the base classifier as well as the test error of the final combined classifier. The margin distribution graph show the cumulative distribution of margins of the training instances after 5, 100 and 1000 iterations, indicated by short-dashed, long-dashed and solid curves, respectively. (Bartlett et al., 1998, pg. 2).

Define

$$\text{conv}(\mathcal{H}) := \left\{ \sum_{k=1}^p \mu_k h_k : p \in \mathbb{N}, \mu_k \geq 0 \text{ and } h_k \in \mathcal{H} \forall k \in [p], \sum_{k=1}^p \mu_k \leq 1 \right\} \quad (4.16)$$

Then,  $\hat{\mathfrak{R}}_S(\text{conv}(\mathcal{H})) = \hat{\mathfrak{R}}_S(\mathcal{H})$ .

*Proof.*

$$\hat{\mathfrak{R}}_S(\text{conv}(\mathcal{H})) = \frac{1}{m \sigma} \mathbb{E} \left[ \sup_{h_k \in \mathcal{H}, \mu \geq 0, \|\mu\|_1 \leq 1} \sum_{i=1}^m \sigma_i \sum_{k=1}^p \mu_k h_k(x_i) \right] \quad (4.17)$$

$$= \frac{1}{m \sigma} \mathbb{E} \left[ \sup_{h_k \in \mathcal{H}, \mu \geq 0, \|\mu\|_1 \leq 1} \sum_{k=1}^p \mu_k \sum_{i=1}^m \sigma_i h_k(x_i) \right] \quad (4.18)$$

$$= \frac{1}{m \sigma} \mathbb{E} \left[ \sup_{h_k \in \mathcal{H}} \max_{k \in [p]} \sum_{i=1}^m \sigma_i h_k(x_i) \right] \quad (4.19)$$

$$= \frac{1}{m \sigma} \mathbb{E} \left[ \sup_{h \in \mathcal{H}} \sum_{i=1}^m \sigma_i h(x_i) \right] \quad (4.20)$$

$$= \hat{R}_S(\mathcal{H}) \quad (4.21)$$

Where (4.19) holds because maximizer of a convex combination places all weight on the largest term.  $\square$

*Remark 4.2.5.* Applying [Theorem 2.3.2](#) with the hypothesis set being

$$\text{conv}(\mathcal{H}) = \left\{ \sum_{t=1}^T \frac{\alpha_t h_t}{\|\alpha\|_1} : T \in \mathbb{N}, \alpha_k \geq 0 \text{ and } h_k \in \mathcal{H} \forall t \in [T] \right\} \quad (4.22)$$

we get

$$R(\bar{f}) \leq \hat{R}_{S,\rho}(\bar{f}) + \frac{2}{\rho} \mathfrak{R}_m(\text{conv}(\mathcal{H})) + \sqrt{\frac{\ln(1/\delta)}{2m}} \quad (4.23)$$

Using the above theorem and the fact that  $R(\bar{f}) = R(f)$ ,

$$R(f) \leq \hat{R}_{S,\rho}(\bar{f}) + \frac{2}{\rho} \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\ln(1/\delta)}{2m}} \quad (4.24)$$

We can now bound  $\hat{R}_{S,\rho}(\bar{f})$  using the below theorem.

**Theorem 4.2.3** (AdaBoost empirical margin-loss bound). *Let  $f = \sum_{t=1}^T \alpha_t h_t$  be returned by AdaBoost. Let  $\varepsilon_t \leq 1/2$  for all  $t \in [T]$ . Then, for all  $\rho > 0$ ,*

$$\hat{R}_{S,\rho}(\bar{f}) \leq 2^T \prod_{t=1}^T \sqrt{\varepsilon_t^{1-\rho} (1 - \varepsilon_t)^{1+\rho}} \quad (4.25)$$

*Proof.*

$$\hat{R}_{S,\rho}(\bar{f}) = \frac{1}{m} \sum_{i=1}^m 1_{y_i f(x_i) - \rho \|\alpha\|_1 \leq 0} \quad (4.26)$$

$$\leq \frac{1}{m} \sum_{i=1}^m e^{-y_i f(x_i) + \rho \|\alpha\|_1} \quad (4.27)$$

$$= \frac{1}{m} \sum_{i=1}^m e^{\rho \|\alpha\|_1} \left( m \prod_{t=1}^T Z_t \right) \mathcal{D}_{t+1}(i) \quad (4.28)$$

$$= e^{\rho \|\alpha\|_1} \prod_{t=1}^T Z_t \quad (4.29)$$

$$= e^{\rho \sum_{t'} \alpha_{t'}} \prod_{t=1}^T Z_t \quad (4.30)$$

$$= \left( \prod_{t=1}^T \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}} \right)^\rho \prod_{t=1}^T 2\sqrt{\varepsilon_t(1 - \varepsilon_t)} \quad (4.31)$$

$$= 2^T \left( \prod_{t=1}^T \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}} \right)^\rho \prod_{t=1}^T \sqrt{\varepsilon_t(1 - \varepsilon_t)} \quad (4.32)$$

where (4.27) holds because  $1_{u \leq 0} \leq e^{-u}$  for all  $u \in \mathbb{R}$ , (4.28) holds by ‘unfolding’ the recursive definition of  $\mathcal{D}_{t+1}$  from algorithm 1 and (4.31) holds by summing both sides of definition of  $\alpha_t$  (from algorithm 1) and rearranging, and since  $Z_t$  is the normalizing factor,

$$Z_t = \sum_{i=1}^m \mathcal{D}_t(i) e^{-\alpha_t y_i h_t(x_i)} \quad (4.33)$$

$$= \sum_{i: y_i h_t(x_i)=1} \mathcal{D}_t(i) e^{-\alpha_t} + \sum_{i: y_i h_t(x_i)=-1} \mathcal{D}_t(i) e^{\alpha_t} \quad (4.34)$$

$$= (1 - \varepsilon_t) e^{-\alpha_t} + \varepsilon_t e^{\alpha_t} \quad (4.35)$$

$$= 2\sqrt{\varepsilon_t(1 - \varepsilon_t)} \quad (4.36)$$



where last equation holds by definition of  $\alpha_t$  (from algorithm 1).  $\square$

*Remark 4.2.6.* In practice,  $\varepsilon_t$  increases with  $t$  because we are giving more important to hard-to-classify points. So, if  $\varepsilon_t$  reaches  $1/2$  quickly, the above theorem becomes useless.

Define edge of the algorithm as some  $\gamma > 0$  such that  $\gamma \leq \frac{1}{2} - \varepsilon_t$  for all  $t \in [T]$ . For  $\rho \leq \gamma$ , it can be shown that  $\hat{R}_{S,\rho}(\bar{f}) \rightarrow 0$  as  $T \rightarrow \infty$ .

Thus, it can be said that AdaBoost increases the confidence margin of the sample points as the number of boosting rounds increases. If  $\hat{R}_{S,\rho}(\bar{f})$  reaches zero, we can increase  $\rho$  while keeping  $\hat{R}_{S,\rho}(\bar{f}) = 0$ , thus reducing the above upper bound on the generalization error.

Though this explanation is not mathematically exact (how does reducing upper bound on the error reduce the error ?), it serves as a good explanation of what might be actually happening.

### 4.2.2 Some remarks

*Remark 4.2.7.* Consider an algorithm for solving the following optimization problem:

$$\max_{\alpha \in \mathbb{R}^T} \rho \tag{4.37}$$

subject to:  $y_i(\alpha^T h(x_i)) \geq \rho \forall i \in [m]$  and  $\|\alpha\|_1 = 1$  and  $\alpha \geq 0$

Then, by definition, the above problem's solution will have an  $L_1$  margin at least as large as that of AdaBoost. But empirically, it is seen that AdaBoost still performs better. Thus, our current theory is not enough to explain the effectiveness of AdaBoost.

*Remark 4.2.8.* If the training sample is not linearly separable, or if the edge is positive but small, AdaBoost may give large weight to just some of the sample points, which will lead to poor overall performance. To overcome this problem, we

can limit the number of boosting rounds (called early stopping) or we can control the magnitude of the weights using  $L_1$  regularization, by minimizing the following function instead:

$$G(\alpha) := \frac{1}{m} \sum_{i=1}^m e^{-y_i f(x_i)} + \lambda \|\alpha\|_1 \quad (4.38)$$

This can be motivated better with the following reasoning.

Using similar logic as [Remark 4.2.5](#), we can use last theorem of [section 2.3](#) to say that for all  $\delta > 0$ , with probability at least  $1 - \delta$ , for all  $f = \sum_{t=1}^T \alpha_t h_t$  with  $\|\alpha\|_1 \leq 1$  and  $\rho \in (0, 1]$ ,

$$R(f) \leq \frac{1}{m} \sum_{i=1}^m 1_{f(x_i) \leq \rho} + \frac{2}{\rho} \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\ln(\log_2(2/\rho))}{m}} + \sqrt{\frac{\ln(2/\delta)}{2m}} \quad (4.39)$$

Note that this also holds for  $\rho > 1$ , since first term becomes 1. Upper bounding the first term,

$$R(f) \leq \frac{1}{m} \sum_{i=1}^m e^{1 - \frac{f(x_i)}{\rho}} + \frac{2}{\rho} \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\ln(\log_2(2/\rho))}{m}} + \sqrt{\frac{\ln(2/\delta)}{2m}} \quad (4.40)$$

Since generalization error of  $f/\rho$  equals that of  $f$ , we can say that for all  $\|\alpha\|_1 \leq 1/\rho$ ,

$$R(f) \leq \frac{1}{m} \sum_{i=1}^m e^{1 - f(x_i)} + \frac{2}{\rho} \mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\ln(\log_2(2/\rho))}{m}} + \sqrt{\frac{\ln(2/\delta)}{2m}} \quad (4.41)$$

Minimizing this upper bound with respect to  $\alpha$ , we get the problem

$$\begin{aligned} & \max_{\alpha \in \mathbb{R}^T} \frac{1}{m} \sum_{i=1}^m e^{-f(x_i)} \\ & \text{subject to: } \|\alpha\|_1 \leq \frac{1}{\rho} \end{aligned} \quad (4.42)$$

which is same as the problem that we were dealing with while making the coordinate descent connection, with the added constraint on  $\|\alpha\|_1$ .

## 4.3 Dimensionality Reduction

Let there be a sample  $S = (x_1, \dots, x_m)$  and a feature mapping  $\Phi : \mathcal{X} \rightarrow \mathbb{R}^N$ . Let  $X = (\Phi(x_1), \dots, \Phi(x_m)) \in \mathbb{R}^{N \times m}$ . We want to find  $k < N$  such that  $Y \in \mathbb{R}^{k \times m}$  is a ‘faithful’ representation of  $X$ .

### 4.3.1 Principal Component Analysis

Fix  $k \in [N]$  and let  $X$  have zero mean. Let  $P_k$  be the set of  $N$ -dimensional  $k$ -rank orthogonal projection matrices. We shall project  $X$  onto that  $k$ -dimensional linear subspace that minimizes the sum of squared  $L_2$ -distances between the original and projected data. Thus, the PCA solution  $P^*$  is given by

$$P^* := \min_{P \in P_k} \|PX - X\|_F^2 \quad (4.43)$$

**Theorem 4.3.1.**  $P^* = U_k U_k^T$ , where  $U_k \in \mathbb{R}^{N \times k}$  is the matrix formed by top  $k$  singular vectors of  $\frac{1}{m} X X^T$  (the sample covariance matrix). Moreover, the desired ‘representation’ of  $X$  is given by the projection  $Y = U_k^T X$ .

*Proof.* Let  $P$  be a projection matrix. Then,

$$\|PX - X\|_F^2 = \text{tr}((PX - X)^T(PX - X)) \quad (4.44)$$

$$= \text{tr}((X^T P - X^T)(PX - X)) \quad (4.45)$$

$$= \text{tr}(X^T P X - X^T P X - X^T P X + X^T X) \quad (4.46)$$

$$= \text{tr}(X^T X) - \text{tr}(X^T P X) \quad (4.47)$$

Therefore,

$$\underset{P \in P_k}{\text{argmin}} \|PX - X\|_F^2 = \underset{P \in P_k}{\text{argmax}} \text{tr}(X^T P X) \quad (4.48)$$

By definition of orthogonal projections in  $P_k$ ,  $P = U U^T$  for some  $U \in \mathbb{R}^{N \times k}$

containing orthogonal columns ([Appendix B](#)). Thus,

$$\text{tr}(X^T P X) = \text{tr}(X^T U U^T X) = \text{tr}(U^T X X^T U) = \sum_{i=1}^k u_i^T X X^T u_i \quad (4.49)$$

Now, we know the largest  $k$  singular vectors of  $X X^T$  maximize the above sum. Since  $X X^T$  and  $\frac{1}{m} X X^T$  have the same singular vectors, we conclude that  $U_k$  maximizes the sum. Moreover,

$$P X = U_k U_k^T X \implies Y = U_k^T X \quad (4.50)$$

□

### 4.3.2 Johnson-Lindenstrauss lemma

**Lemma 4.3.2.** *Let  $Q \sim \chi_{(k)}^2$ . Then, for all  $\varepsilon \in (0, 1/2)$ ,*

$$\mathbb{P}[(1 - \varepsilon)k \leq Q \leq (1 + \varepsilon)k] \geq 1 - 2e^{-(\varepsilon^2 - \varepsilon^3)k/4} \quad (4.51)$$

*Proof.* By Markov's inequality,

$$\mathbb{P}[Q \geq (1 + \varepsilon)k] = \mathbb{P}[e^{\lambda Q} \geq e^{\lambda(1+\varepsilon)k}] \quad (4.52)$$

$$\leq \frac{\mathbb{E}[e^{\lambda Q}]}{e^{\lambda(1+\varepsilon)k}} \quad (4.53)$$

$$= \frac{(1 - 2\lambda)^{-k/2}}{e^{\lambda(1+\varepsilon)k}} \quad (4.54)$$

for  $\lambda < \frac{1}{2}$ . Now, choosing  $\lambda = \frac{\varepsilon}{2(1+\varepsilon)} < \frac{1}{2}$  to minimize RHS, and using the fact that  $1 + \varepsilon \leq e^{\varepsilon - \frac{\varepsilon^2 - \varepsilon^3}{2}}$ , we get

$$\mathbb{P}[Q \geq (1 + \varepsilon)k] \leq \left( \frac{1 + \varepsilon}{e^\varepsilon} \right)^{k/2} \quad (4.55)$$

$$\leq \left( \frac{e^{\varepsilon - \frac{\varepsilon^2 - \varepsilon^3}{2}}}{e^\varepsilon} \right)^{k/2} \quad (4.56)$$

$$= e^{\frac{-k}{4}(\varepsilon^2 - \varepsilon^3)} \quad (4.57)$$

Using similar technique to bound  $\mathbb{P}[Q \leq (1 - \varepsilon)k]$ , we can get the claim of the theorem by using union bound.  $\square$

**Lemma 4.3.3.** *Let  $x \in \mathbb{R}^N$  and  $k < N$ . Entries of a matrix  $A \in \mathbb{R}^{k \times N}$  are drawn independently from  $N(0, 1)$ . Then, for all  $\varepsilon \in (0, 1/2)$ ,*

$$\mathbb{P} \left[ (1 - \varepsilon) \|x\|^2 \leq \left\| \frac{1}{\sqrt{k}} Ax \right\|^2 \leq (1 + \varepsilon) \|x\|^2 \right] \geq 1 - 2e^{-(\varepsilon^2 - \varepsilon^3)k/4} \quad (4.58)$$

*Proof.* Let  $\hat{x} = Ax$ . Then,

$$\mathbb{E}[\hat{x}_j^2] = \mathbb{E} \left[ \left( \sum_{i=1}^N A_{ji} x_i \right)^2 \right] \quad (4.59)$$

$$= \mathbb{E} \left[ \sum_{i=1}^N A_{ji}^2 x_i^2 \right] \quad (4.60)$$

$$= \sum_{i=1}^N x_i^2 = \|x\|^2 \quad (4.61)$$

Define  $T_j := \frac{\hat{x}_j}{\|x\|}$ . Since  $T_j$ 's are iid  $N(0, 1)$ ,  $Q := \sum_{j=1}^k T_j^2 \sim \chi_{(k)}^2$ . Thus,

$$\mathbb{P} \left[ (1 - \varepsilon) \|x\|^2 \leq \frac{\|\hat{x}\|^2}{k} \leq (1 + \varepsilon) \|x\|^2 \right] \quad (4.62)$$

$$= \mathbb{P} \left[ (1 - \varepsilon)k \leq \sum_{j=1}^k T_j^2 \leq (1 + \varepsilon)k \right] \quad (4.63)$$

$$= \mathbb{P}[(1 - \varepsilon)k \leq Q \leq (1 + \varepsilon)k] \quad (4.64)$$

$$\geq 1 - 2e^{-(\varepsilon^2 - \varepsilon^3)k/4} \quad (4.65)$$

where (4.65) holds from Lemma 4.3.2.  $\square$

**Theorem 4.3.4** (Johnson-Lindenstrauss). *Let  $\varepsilon \in (0, 1/2)$ ,  $k = \frac{20 \log(m)}{\varepsilon^2}$  and  $V \subset \mathbb{R}^N$  such that  $|V| = m > 4$ . Then, there exists  $f : \mathbb{R}^N \rightarrow \mathbb{R}^k$  such that for all  $u, v \in V$ ,*

$$(1 - \varepsilon) \|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \varepsilon) \|u - v\|^2 \quad (4.66)$$

*Proof.* Let  $f = \frac{A}{\sqrt{k}}$  such that entries of  $A$  are drawn independently from  $N(0, 1)$ . For fixed  $u, v \in V$ , apply [Lemma 4.3.3](#) with  $x = u - v$  and then use union bound over all pairs of points in  $V$  to get

$$\mathbb{P}[\text{success}] \geq 1 - 2m^2 e^{-(\varepsilon^2 - \varepsilon^3)k/4} \quad (4.67)$$

$$= 1 - 2m^2 e^{-(\varepsilon^2 - \varepsilon^3)20 \log(m)/4\varepsilon^2} \quad (4.68)$$

$$= 1 - 2m^2 m^{-(\varepsilon^2 - \varepsilon^3)5/\varepsilon^2} \quad (4.69)$$

$$= 1 - 2m^{5\varepsilon - 3} \quad (4.70)$$

$$> 1 - 2m^{-1/2} \quad (4.71)$$

$$> 0 \quad (4.72)$$

where success means finding an  $f$  such that it satisfies [\(4.66\)](#) for all  $u, v \in V$ , [\(4.71\)](#) holds iff  $\varepsilon < 1/2$  and [\(4.72\)](#) hold iff  $m > 4$ .  $\square$

Thus, for any sample of size  $m$  in  $\mathbb{R}^N$ , we can find a mapping to  $\mathbb{R}^k$  that is close to being isometric.

## *Chapter 5*

# NEURAL NETWORKS

Until now, we have looked at one type of hypothesis set - that of linear hyperplanes - SVMs and extended their functionality using kernels. However, in practice, SVMs are usually not the ones used, since they do not provide enough complexity for real world cases. We want a hypothesis set whose complexity can be tweaked as per use case. Neural networks are one example of such hypothesis sets. They are very easy to implement, can be tweaked easily, and benefit from many theoretical guarantees.

We start by defining a neural network and the basic concepts surrounding it ([section 5.1](#)) and then go into what algorithms ([section 5.2](#)) can be used to train ([section 5.3](#)) them. We then go over the approximation ([section 5.4](#)) and information processing ([section 5.5](#)) capabilities of neural networks. Finally, we study some simple neural network variants ([section 5.6](#)) that are used widely in practice for various uses.

## 5.1 Introduction

A neural network is a function made up of compositions of affine linear functions with nonlinear functions. Composition of a nonlinear function with an affine linear

function is called a **neuron**. For example,  $X : \mathbb{R}^n \rightarrow \mathbb{R}$  is a neuron, defined as

$$X(x) := \sigma(w^T x + b) \quad (5.1)$$

where  $w \in \mathbb{R}^n$  is the weights vector of the neuron and  $b \in \mathbb{R}$  is the bias of the neuron. Both these parameters are tuned while fitting the neural network, which is also called **training** the network.  $\sigma$  is a nonlinear function (also called the **activation function**) which is put to remove the linearity. Otherwise, a neuron will always only take form of a linear function, which would mean that, for example, it would be able to classify only linearly separable classes.

These neurons are assembled in layers, with input of neuron in a layer being all of the outputs of neurons in the previous layer. A neural network is also called a **fully connected neural network** or a **multi-layer perceptron**.

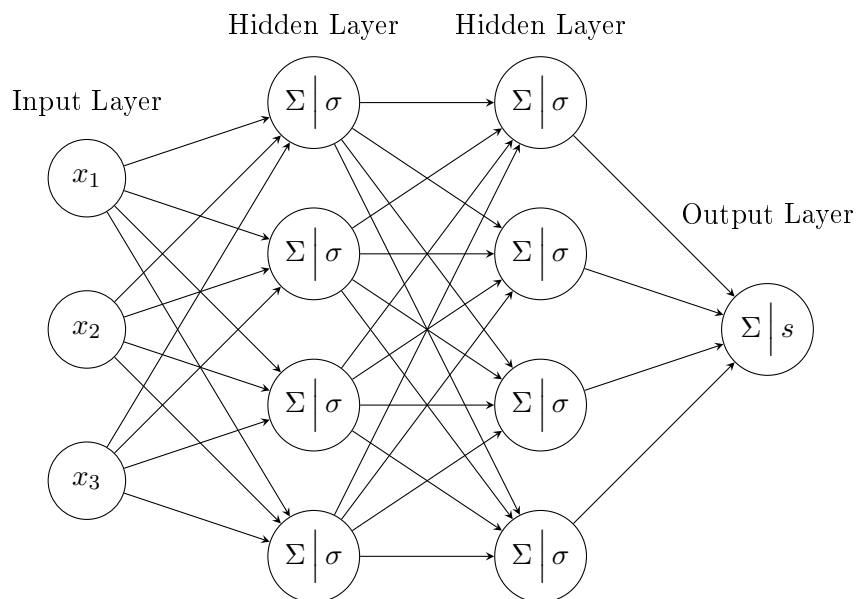
The first layer is called the **input layer**, with each neuron being each dimension of the input data. The final layer is called the **output layer**, with each neuron being the dimension of the output data. All layers between the input and output layers are called the **hidden layers**.

One of the big advantages of using neural networks is their robustness. Just by changing the number of neurons in input/output layer, we can account for multi-dimensional input/output problems. By changing the output layer activation, we can account for regression or classification (binary or multi-class). If the model is overfitting, we can try reducing the number of layers or neurons. Such simple fixes gives rise to tremendous flexibility that is not present in traditional methods.

**Definition 5.1.1** (Neural network). A neural network with

1.  $L \in \mathbb{N}$  being the number of layers
2.  $d^{(l)} \in \mathbb{N}$  being number of neurons in  $l^{\text{th}}$  layer
3.  $w_{ij}^{(l)} \in \mathbb{R}$  being the weight from  $i^{\text{th}}$  neuron in layer  $l - 1$  to  $j^{\text{th}}$  neuron in layer  $l$





**Figure 5.1:** A neural network schematic.

4.  $b_j^{(l)} \in \mathbb{R}$  being the bias of the  $j^{\text{th}}$  neuron in layer  $l$
5.  $\phi_{(l)} : \mathbb{R} \rightarrow \mathbb{R}$  being the activation function of layer  $l$

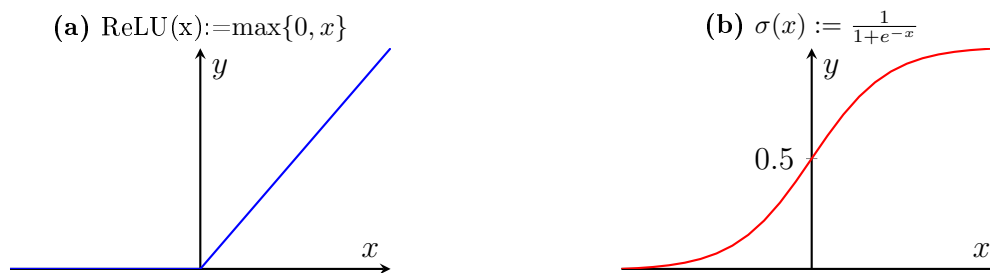
is a function  $F : [0, 1]^{d^{(0)}} \rightarrow \mathbb{R}^{d^{(L)}}$  defined as

$$F(z) := (X^{(L)} \circ \dots \circ X^{(1)})(z) \quad (5.2)$$

for all  $z \in [0, 1]^{d^{(0)}}$  with the output of the  $l^{\text{th}}$  layer  $X^{(l)} : \mathbb{R}^{d^{(l-1)}} \rightarrow \mathbb{R}^{d^{(l)}}$  being defined recursively as

$$X^{(l)} := \phi_{(l)}(W^{(l)T} X^{(l-1)} - B^{(l)}) \quad (5.3)$$

with  $X^{(l)} = (x_1^{(l)}, \dots, x_{d^{(l)}}^{(l)})^T$ ,  $W^{(l)} = (w_{ij}^{(l)})_{i,j} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$  and  $B^{(l)} = (b_1^{(l)}, \dots, b_{d^{(l)}}^{(l)})^T$  for all  $l \in [L]$ ,  $X^{(0)} \equiv z$  being the vector of inputs and  $\phi_{(l)}$  being applied component-wise.



**Figure 5.2:** The ReLU and sigmoid activation functions

### 5.1.1 Activation functions

The activation functions usually have to be provided. Usually, all of the hidden layers have the same activation, the input layer does not need any activation, and the output layer has an activation depending upon whether the task is classification or regression.

There is an umpteen number of activation functions that have been used. Hidden layer activation is usually either Rectified Linear Unit ReLU:  $\mathbb{R} \rightarrow \mathbb{R}$  or sigmoid  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ , defined as follows.

$$\text{ReLU}(x) := \max\{0, x\} \quad (5.4)$$

$$\sigma(x) := \frac{1}{1 + e^{-x}} \quad (5.5)$$

for all  $x \in \mathbb{R}$ . Output layer activation is usually either just sigmoid for binary classification, softmax:  $\mathbb{R}^k \rightarrow \mathbb{R}^k$  for  $k$ -class classification or nothing (ie. linear) for regression. softmax is defined as

$$\text{softmax}(x) := \left( \frac{e^{x_1}}{\sum_{i=1}^k e^{x_i}}, \dots, \frac{e^{x_k}}{\sum_{i=1}^k e^{x_i}} \right) \quad (5.6)$$

for all  $x \in \mathbb{R}^k$ . Sigmoid or softmax convert the output into a form that can be interpreted as probabilities for each class, which then makes classification easier - just take the class as the one having the highest probability.

### 5.1.2 Training

Training a neural network refers to finding those values of the weights and biases of all the neurons in the network such that the network has low error on the given training data. This error is measured using **cost functions**.

As before, we partition our given dataset into a **training** dataset - to find the appropriate weights and biases, a **validation** dataset - to find the appropriate values of the hyperparameters and a **testing** dataset - to test how good our neural network is. We want the value of the cost function on both the test and train datasets to be low.

There is a plethora of cost functions that can be used. An example of a simple cost function for regression and ( $k$ -class) classification each is mean squared error  $MSE : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$  and cross-entropy  $S : \mathbb{R}^k \times \mathbb{R}_+^k \rightarrow \mathbb{R}$ , defined as follows.

$$MSE(y, y') := \frac{1}{m} \sum_{i=1}^m (y - y'_i)^2 \quad (5.7)$$

$$S(y, y') := - \sum_{i=1}^m y_i \ln(y'_i) \quad (5.8)$$

where  $y, y' \in \mathbb{R}^m$  for MSE and  $y \in \mathbb{R}^m, y' \in \mathbb{R}_+^m$  for cross-entropy (which will usually be true as components of output of softmax are all positive).

To prevent overfitting, sometimes the cost can also be penalized with the norm of the weights and biases.

An important property of cost functions is that they must be able to be written as a sum of the cost over the training data points, ie. a cost function  $C$  should be of the form  $C = \sum_x C_x$  where sum is over all training data points. This is to ensure that the backpropagation algorithm can be implemented, which will be discussed later.

## 5.2 Optimization

While training a neural network, we are interested in minimizing  $C \circ F$  over the training data, where  $C$  is the cost function and  $F$  is the neural network. This minimization is not at all straightforward for any practically relevant neural network. Thus, we resort to numerical methods like gradient descent and adam to find approximate global minimas.

### 5.2.1 Gradient Descent

Given a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  and a value in its domain  $x \in \mathbb{R}^n$ , we (greedily) push  $x$  in the direction that leads to the highest decrease in  $f(x)$ . This is called gradient descent. Since the direction of steepest descent is given by  $\nabla f$ , the gradient descent algorithm is defined as follows. Starting from an initial estimate  $x_0 \in \mathbb{R}^n$ , form a sequence of points in the domain of  $f$  by

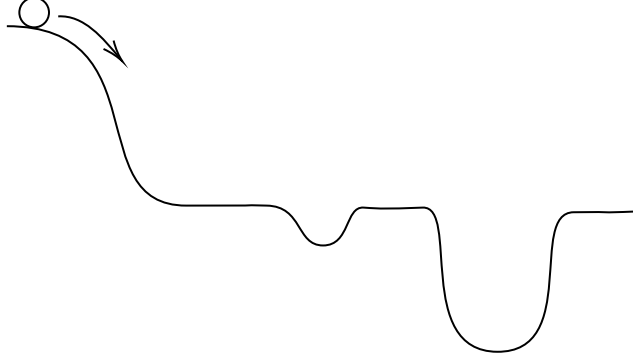
$$x_{t+1} = x_t - \eta \nabla f(x_t) \tag{5.9}$$

for all  $t \in \mathbb{N}_0$ , where  $\eta > 0$  is called the **learning rate** - it is the size of the step that we take. In practice, we obviously cannot continue this algorithm forever. We stop at some iteration  $T \in \mathbb{N}$ .

It can be beneficial to decrease the value of  $\eta$  as  $T$  increases, since taking a large step while near the minima might cause the algorithm to ‘bounce around’ and take longer to converge (if it does) than expected.

Gradient descent need not converge to a global minimum for all functions. It is easy to see that the algorithm can get stuck in local minimas or plateaus. To deal with these issues, we will introduce Adam in the next section. However, convergence guarantees can be given for good enough functions. The following lemmas and theorem were taken from Garrigos and Gower, [2023](#).

**Lemma 5.2.1.** *Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be differentiable and  $\nabla f$  be  $L$ -lipschitz. Then, for*



**Figure 5.3:** A function with plateaus and local minimas.

all  $x, y \in \mathbb{R}^d$ ,

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2 \quad (5.10)$$

*Proof.* Let  $x, y \in \mathbb{R}^d$ . Define  $\phi : \mathbb{R} \rightarrow \mathbb{R}$  by  $\phi(t) := f(x + t(y - x))$  for all  $t \in \mathbb{R}$ . Using fundamental theorem of calculus,

$$f(y) - f(x) = \int_0^1 \langle \nabla f(x + t(y - x)), y - x \rangle dt \quad (5.11)$$

$$= \langle \nabla f(x), y - x \rangle + \int_0^1 \langle \nabla f(x + t(y - x)) - \nabla f(x), y - x \rangle dt \quad (5.12)$$

$$\leq \langle \nabla f(x), y - x \rangle + \int_0^1 \|\nabla f(x + t(y - x)) - \nabla f(x)\| \|y - x\| dt \quad (5.13)$$

$$\leq \langle \nabla f(x), y - x \rangle + \int_0^1 Lt \|y - x\|^2 dt \quad (5.14)$$

$$= \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2 \quad (5.15)$$

$$(5.16)$$

where we used Cauchy-Schwartz inequality in (5.13) ([Appendix D](#)).  $\square$

**Lemma 5.2.2.** Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be differentiable and  $\nabla f$  be  $L$ -lipschitz and  $\lambda > 0$ . Then, for all  $x, y \in \mathbb{R}^d$ ,

$$f(x - \lambda \nabla f(x)) - f(x) \leq -\lambda \left(1 - \frac{\lambda L}{2}\right) \|\nabla f(x)\|^2 \quad (5.17)$$

*Proof.* Put  $y = x - \lambda \nabla f(x)$  in [Lemma 5.2.1](#).  $\square$

**Theorem 5.2.3** (Gradient Descent convergence rate). *Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be differentiable, convex and  $\nabla f$  be  $L$ -lipschitz. Let  $\operatorname{argmin} f \neq \emptyset$ . Let  $(x_t)_{t \in \mathbb{N}}$  be the sequence generated by gradient descent with learning rate  $\eta \in (0, \frac{1}{L}]$ . Then, for all  $x^* \in \operatorname{argmin} f$ , for all  $t \in \mathbb{N}$ ,*

$$f(x_t) - \inf f \leq \frac{\|x_0 - x^*\|^2}{2\eta t}$$

*Proof.* Let  $x^* \in \operatorname{argmin} f$ . From Lemma 5.2.2,  $(f(x_t))_t$  is decreasing (since  $\eta L \leq 1$ ):

$$f(x_{t+1}) - f(x_t) \leq -\eta \left(1 - \frac{\eta L}{2}\right) \|\nabla f(x_t)\|^2 \leq 0 \quad (5.18)$$

Now, we show  $(\|x_t - x^*\|^2)_t$  is decreasing.

$$\|x_{t+1} - x^*\|^2 - \|x_t - x^*\|^2 \quad (5.19)$$

$$= \|x_{t+1}\|^2 - \|x_t\|^2 - 2\langle x_{t+1}, x^* \rangle + 2\langle x_t, x^* \rangle \quad (5.20)$$

$$= \|x_{t+1}\|^2 - \|x_t\|^2 - 2\langle x_{t+1} - x_t, x^* \rangle \quad (5.21)$$

$$= \|x_{t+1} - x_t\|^2 + 2\langle x_{t+1}, x_t \rangle - 2\langle x^*, x_{t+1} - x_t \rangle - 2\|x_t\|^2 \quad (5.22)$$

$$= \|x_{t+1} - x_t\|^2 + 2\langle x_t - \eta \nabla f(x_t), x_t \rangle - 2\langle x^*, \eta \nabla f(x_t) \rangle - 2\|x_t\|^2 \quad (5.23)$$

$$= \|x_{t+1} - x_t\|^2 - 2\eta \langle \nabla f(x_t), x_t \rangle - 2\eta \langle x^*, \nabla f(x_t) \rangle \quad (5.24)$$

$$= \|x_{t+1} - x_t\|^2 - 2\eta \langle \nabla f(x_t), x_t - x^* \rangle \quad (5.25)$$

$$= \|x_{t+1} - x_t\|^2 - 2\eta \langle \nabla f(x_t), x_{t+1} + \eta \nabla f(x_t) - x^* \rangle \quad (5.26)$$

$$= \|x_{t+1} - x_t\|^2 - 2\eta \langle \nabla f(x_t), x_{t+1} - x^* \rangle - 2\eta^2 \|\nabla f(x_t)\|^2 \quad (5.27)$$

$$= -\|x_{t+1} - x_t\|^2 + 2\|x_{t+1} - x_t\|^2 - 2\eta \langle \nabla f(x_t), x_{t+1} - x^* \rangle - 2\eta^2 \|\nabla f(x_t)\|^2 \quad (5.28)$$

$$= -\|x_{t+1} - x_t\|^2 + 2\|-\eta \nabla f(x_t)\|^2 - 2\eta \langle \nabla f(x_t), x_{t+1} - x^* \rangle - 2\eta^2 \|\nabla f(x_t)\|^2 \quad (5.29)$$

$$= -\|x_{t+1} - x_t\|^2 - 2\eta \langle \nabla f(x_t), x_{t+1} - x^* \rangle \quad (5.30)$$

$$= -\|x_{t+1} - x_t\|^2 - 2\eta \langle \nabla f(x_t), x_{t+1} - x_t \rangle + 2\eta \langle \nabla f(x_t), x^* - x_t \rangle \quad (5.31)$$

By convexity of  $f$ ,

$$\langle \nabla f(x_t), x^* - x_t \rangle \leq f(x^*) - f(x_t) = \inf f - f(x_t) \quad (5.32)$$

And by [Lemma 5.2.1](#),

$$-\langle \nabla f(x_t), x_{t+1} - x_t \rangle \leq \frac{L}{2} \|x_{t+1} - x_t\|^2 + f(x_t) = f(x_{t+1}) \quad (5.33)$$

Putting these into [\(5.31\)](#),

$$\frac{1}{2\eta} [\|x_{t+1} - x^*\|^2 - \|x_t - x^*\|^2] \quad (5.34)$$

$$\leq \frac{-1}{2\eta} \|x_{t+1} - x_t\|^2 + \frac{L}{2} \|x_{t+1} - x_t\|^2 - (f(x_{t+1}) - \inf f) \quad (5.35)$$

$$\leq -(f(x_{t+1}) - \inf f) \quad (5.36)$$

since  $L\eta \leq 1$ . And since  $\inf f \leq f(x_{t+1})$ , we conclude that  $(\|x_t - x^*\|^2)_t$  is decreasing. Now, define Lyapunov energy as

$$E_t := \frac{1}{2\eta} \|x_t - x^*\|^2 + t(f(x_t) - \inf f) \quad (5.37)$$

$(E_t)_t$  is also a decreasing sequence:

$$E_{t+1} - E_t \quad (5.38)$$

$$= (t+1)(f(x_{t+1}) - \inf f) - t(f(x_t) - \inf f) + \frac{1}{2\eta} [\|x_{t+1}\|^2 - \|x_t - x^*\|^2] \quad (5.39)$$

$$= f(x_{t+1}) - \inf f + t[f(x_{t+1}) - f(x_t)] + \frac{1}{2\eta} [\|x_{t+1}\|^2 - \|x_t - x^*\|^2] \quad (5.40)$$

$$\leq f(x_{t+1}) - \inf f \quad (5.41)$$

$$\leq 0 \quad (5.42)$$

where [\(5.41\)](#) holds because both  $(f(x_t))_t$  and  $(\|x_t - x^*\|^2)_t$  are decreasing. There-

fore, for any  $t \in \mathbb{N}$ ,  $E_t \leq E_0$ , which implies

$$\frac{1}{2\eta} \|x_t - x^*\|^2 + t(f(x_t) - \inf f) \leq \frac{1}{2\eta} \|x_0 - x^*\|^2 \quad (5.43)$$

$$\implies t(f(x_t) - \inf f) \leq \frac{1}{2\eta} \|x_0 - x^*\|^2 \quad (5.44)$$

$$\iff f(x_t) - \inf f \leq \frac{1}{2\eta t} \|x_0 - x^*\|^2 \quad (5.45)$$

□

### 5.2.2 Adam

Imagine a ball rolling down a hill - it will always go over the local minimas and plateaus if it has enough momentum. We simulate this with Adaptive moments, or Adam for short, which is the most popular algorithm for neural network fitting. It is defined as follows. Start with defining

$$m_t := \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (5.46)$$

$$v_t := \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (5.47)$$

for all  $t \in \mathbb{N}$ , with  $m_0 = v_0 = 0$ , where  $g_t := \nabla f(x_t)$ ,  $g_t^2$  refers to the vector of element-wise squares of  $g_t$  and  $\beta_1, \beta_2 \in (0, 1]$ . Assuming some stochasticity built in  $g_t$ 's,  $m_t$  and  $v_t$  can be seen as biased estimates of the first and second moments of  $g_t$  respectively (assuming  $\mathbb{E}(g_t), \mathbb{E}(g_t^2)$  are constants for all  $t \in \mathbb{N}$ ),

$$\mathbb{E}[m_t] = (1 - \beta_1^t) \mathbb{E}(g_t) \quad (5.48)$$

$$\mathbb{E}[v_t] = (1 - \beta_2^t) \mathbb{E}(g_t^2) \quad (5.49)$$

Thus, the unbiased estimates of the first and second moments are

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (5.50)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (5.51)$$



Finally, the update rule becomes

$$x_{t+1} = x_t - \eta \frac{\hat{m}_t}{\sqrt{|\hat{v}_t|} + \varepsilon} \quad (5.52)$$

where  $\varepsilon > 0$  is some small number to prevent division by zero.

The update rule for  $m_t$  gives a sense of memory to the gradient. In gradient descent, the update depends only on the current gradient - without any regard for any previous gradients. Taking into account the previous gradients gives momentum to the update. Thus, getting stuck in valleys or plateaus becomes harder.

However, if we keep on adding the gradients - the updating term will keep on monotonically increasing. Thus, stopping near a minima will become harder as the number of iterations increase. To dampen this effect, we divide by the square root of norm of the momentum updated squared gradient. This acts as a normalization factor - decreasing the value of the updating term more and more as the number of iterations increase.

But still, as with gradient descent, it is not guaranteed that adam will always converge to a global minimum in general. However, Li et al., 2023 have proved its convergence, at least to a stationary point, for a good enough class of functions.

## 5.3 Training

In previous section, we had introduced some basic concepts regarding training neural networks. Now, we shall see how backpropagation exactly works, and what are some common practical problems that one might need to deal with when training neural networks.

### 5.3.1 Backpropagation

Backpropagation is simply an application of the chain rule to find  $\nabla C$ . Once we know  $\nabla C$ , we can use any algorithm that uses it - in particular gradient descent and adam. We assume activation function  $\phi : \mathbb{R} \rightarrow \mathbb{R}$  is differentiable.

We know output of the  $l^{\text{th}}$  layer is defined as

$$x_j^{(l)} := \phi(s_j^{(l)}) \quad (5.53)$$

$$s_j^{(l)} := \sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} - b_j^{(l)} \quad (5.54)$$

for all  $j \in [d^{(l)}]$  and  $l \in [L]$  (assuming same activation  $\phi$  for all hidden layers), with  $x_1^{(0)}, \dots, x_{d^{(0)}}^{(0)}$  being the input. First, we calculate all the layer outputs for all of the training data points. This is called a **forward pass**. Then, we begin finding the derivatives.

By chain rule, for all  $i \in [d^{(l-1)}], j \in [d^{(l)}]$  and  $l \in [L]$ ,

$$\frac{\partial C}{\partial w_{ij}^{(l)}} = \frac{\partial C}{\partial s_j^{(l)}} \frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} x_i^{(l-1)} \quad (5.55)$$

for  $\delta_j^{(l)} := \frac{\partial C}{\partial s_j^{(l)}}$ . Similarly,

$$\frac{\partial C}{\partial b_j^{(l)}} = \frac{\partial C}{\partial s_j^{(l)}} \frac{\partial s_j^{(l)}}{\partial b_j^{(l)}} = -\delta_j^{(l)} \quad (5.56)$$

So, we can write

$$\nabla C = (\nabla_w C, \nabla_b C) = \left( \delta_j^{(l)}(x_i^{(l-1)}, -1) \right)_{i,j,l} \quad (5.57)$$

Thus, we now need to find the  $\delta_j^{(l)}$ 's. To do so, we will find  $\delta_j^{(L)}$  and use it to find all other  $\delta_j^{(l)}$ 's recursively. Now, note that only  $\delta_j^{(L)}$  depends on the cost function. This is an advantage of backpropagation - changing the cost means only changing

the following expression for  $\delta_j^{(L)}$ . Everything else remains the same.

To continue, let's take the cost to be MSE :  $C(w, b) := \frac{1}{2} \sum_{i=1}^{d^{(L)}} (x_i^{(L)} - z_i)^2$ . Then,

$$\delta_j^{(L)} = \frac{\partial C}{\partial s_j^{(L)}} = (x_j^{(L)} - z_j) \phi'(s_j^{(L)}) \quad (5.58)$$

We now begin going backwards. Since each  $s_i^{(l-1)}$  is affected by all  $s_j^{(l)}$ 's, by chain rule,

$$\delta_i^{(l-1)} = \frac{\partial C}{\partial s_i^{(l-1)}} = \sum_{j=0}^{d^{(l)}} \frac{\partial C}{\partial s_j^{(l)}} \frac{\partial s_j^{(l)}}{\partial s_i^{(l-1)}} \quad (5.59)$$

We are done once we find  $\frac{\partial s_j^{(l)}}{\partial s_i^{(l-1)}}$ .

$$\frac{\partial s_j^{(l)}}{\partial s_i^{(l-1)}} = w_{ij}^{(l)} \phi'(s_i^{(l-1)}) \quad (5.60)$$

Thus,

$$\delta_i^{(l-1)} = \phi'(s_i^{(l-1)}) \sum_{j=0}^{d^{(l)}} \delta_j^{(l)} w_{ij}^{(l)} \quad (5.61)$$

And, we are done! Putting everything together in nice matrix equations, we get the following. For each  $l \in [L]$ , find the layer outputs (forward pass).

$$X^{(l)} = \phi(W^{(l)T} X^{(l-1)} - B^{(l)}) \quad (5.62)$$

Then, begin backpropagation by finding  $\delta^{(L)}$ . For MSE, the expression for  $\delta^{(L)}$  is

$$\delta^{(L)} = (x^{(L)} - z) \odot \phi'(s^{(L)}) \quad (5.63)$$

Next, propagate the values backwards, finding  $\delta^{(l-1)}$  for each  $l = 2, \dots, L$ .

$$\delta^{(l-1)} = (W^{(l)} \delta^{(l)}) \odot \phi'(s^{(l-1)}) \quad (5.64)$$

Finally, the gradient components are

$$\frac{\partial C}{\partial W^{(l)}} = X^{(l-1)} \delta^{(l)T} \quad \frac{\partial C}{\partial B^{(l)}} = -\delta^{(l)} \quad (5.65)$$

for all  $l \in [L]$ .

*Remark 5.3.1.* For functions like ReLU which are not differentiable but convex, we can replace all differentiations with sub-differentiations.

*Remark 5.3.2.* From the above equations, values of  $\delta^{(l)}$  depend on the derivative of the activation function, for all  $l \in [L]$ . So, if the activation is sigmoid, then backpropagation through each layer results in reduction in the value of  $\delta^{(l)}$  by a factor of at least 4, since  $\sigma' \in (0, \frac{1}{4}]$ . Thus, for a deep neural network, learning the weights of the initial few layers will become very slow. This is the **vanishing gradient problem**. To counter this, we can use the ReLU activation instead.

### 5.3.2 Batch Training

The backpropagation algorithm explained above gave us the value of  $\nabla C$  at one training sample point. Thus, to use this in an algorithm like gradient descent or adam, we can do several things.

We can take mean of the gradient over all training points and then use it in the algorithm:

$$w_{t+1} = w_t - \eta \nabla \bar{C}(w_t) \quad (5.66)$$

where  $\bar{C} = \frac{1}{m} \sum_{i=1}^m \nabla C_{x_i}(w_t)$ , where  $m$  is number of training data points and  $\nabla C_{x_i}(w_t)$  is the value of the gradient of cost calculated at  $x_i$  for weight  $w_t$  (or iteration  $t$ ).

Or on the other hand, for each iteration of the algorithm, we can use each individual gradient for the update. This is also called **stochastic gradient descent**:

$$\begin{aligned} w_{t+1} &= w_t - \eta \nabla C_{x_1}(w_t) \\ &\vdots \end{aligned} \quad (5.67)$$

$$w_{t+1} = w_t - \eta \nabla C_{x_m}(w_t)$$

So, we can either update the parameters with the mean gradient of the data points, or we can update the parameters with the gradient of each data point. But, these are two extreme cases and are not usually used in practice.

Finding the mean gradient over all data points results in a lot of computation for each update, which makes the algorithm slow. But updating the parameters for each training point results in a very haphazard descent.

In other words, we are trying to estimate the true gradient using the sample gradients. If the sample we use to estimate is too small, variance of the estimate increases, but it is quicker to compute. And if the sample is too large, variance of the estimate decreases, but it is slower to compute.

Thus, it is best to find a mid point - to estimate the gradient using a batch of training points. This is called **batch stochastic gradient descent**. For example, if we divide the training sample into two batches, we can do the following for each iteration.

$$w_{t+1} = w_t - \eta \nabla \bar{C}_1(w_t) \tag{5.68}$$

$$w_{t+1} = w_t - \eta \nabla \bar{C}_2(w_t)$$

where  $\bar{C}_1 = \frac{2}{m} \sum_{i=1}^{\lfloor m/2 \rfloor} \nabla C_{x_i}(w_t)$  and  $\bar{C}_2 = \frac{2}{m} \sum_{i=\lceil m/2 \rceil}^m \nabla C_{x_i}(w_t)$ .

When the parameters have been updated using all of the training sample points, one **epoch** is said to have passed. So, each iteration of the gradient descent algorithm, where an update has been made using all of the batches, is called an epoch.



**Figure 5.4:** GD vs. SGD training curves

### 5.3.3 Weights Initialization

Both gradient descent and adam require some initial value of the weights and biases  $w_0, b_0$  to start. Choosing these initial parameters properly is crucial, as choosing them suboptimally results in longer or worse training.

Consider initializing weights and biases for a single layer. They should not be very close to zero - as that decreases the input signal variance rapidly with each layer. They should also not be too large - as that amplifies the input signal variance dramatically. We have

$$Y_j = \phi \left( \sum_i w_{ij} X_i + b_j \right) \quad (5.69)$$

Let's initialize the biases to zero and treat layer input and output, and the weights as random variables. Let's approximate  $Y_j$  linearly around  $\mathbb{E}[\sum_i w_{ij} X_i]$  and then take its variance:

$$\text{Var}(Y_j) \approx \phi' \left( \sum_i \mathbb{E}[w_{ij} X_i] \right)^2 \text{Var} \left( \sum_i w_{ij} X_i \right) \quad (5.70)$$

We want to preserve input signal variance as otherwise the signal might either die or explode as it progresses through the layers, ie. we want  $\text{Var}(Y_j) = \text{Var}(X_i)$  for

each  $i, j$ .

It makes sense to assume  $w_{ij}$  are iid for all  $i$ , and  $X_i$  are iid for all  $i$ , with each pair of  $w_{ij}, X_i$  being independent. We can also choose  $w_{ij}$  such that  $\mathbb{E}[w_{ij}] = 0$  for all  $i, j$ . Then, we get

$$\text{Var}(Y_j) \approx \phi'(0)^2 N \text{Var}(w_{ij}) \text{Var}(X_i) \quad (5.71)$$

Imposing  $\text{Var}(Y_j) = \text{Var}(X_i)$ , we get

$$\text{Var}(w_{ij}) = \frac{1}{N \phi'(0)^2} \quad (5.72)$$

Thus, we can initialize the weight as, for example

$$w_{ij} \sim N\left(0, \frac{1}{N \phi'(0)^2}\right) \quad (5.73)$$

Or,

$$w_{ij} \sim \text{Unif}\left[\frac{-\sqrt{3}}{\phi'(0)\sqrt{N}}, \frac{\sqrt{3}}{\phi'(0)\sqrt{N}}\right] \quad (5.74)$$

This is called LeCun initialization.

Instead of just preserving forward signal variance, we can also preserve backward signal variance too. This is called Xavier initialization.

$$\text{Var}(w_{ij}) = \frac{2}{N + N'} \quad (5.75)$$

where  $N'$  is number of neurons in the previous layer.

## 5.4 Approximation

It has been seen empirically that neural networks are very powerful for many tasks. But why? One of the reasons can be that they can approximate any continuous functions on a compact interval. This property is called universal approximation.

We start with proving that shallow (1-layer) ReLU networks are dense in  $C[0, 1]$ .

**Lemma 5.4.1.** *Let  $g \in C[0, 1]$ . Then, for all  $\varepsilon > 0$ , there exists an equidistant partition  $a = x_0 < x_1 < \dots < x_N = b$  such that the piecewise linear function  $g_\varepsilon$  passing through  $(x_i, g(x_i))$ 's satisfies*

$$|g(x) - g_\varepsilon(x)| < \varepsilon \quad (5.76)$$

for all  $x \in [a, b]$ .

*Proof.* Let  $\varepsilon > 0$ . Since  $g$  is uniformly continuous on  $[a, b]$ , there exists  $\delta > 0$  such that for all  $x', x'' \in [a, b]$  with  $|x' - x''| < \delta$ , we have  $|g(x') - g(x'')| < \varepsilon/2$ . Let  $N \in \mathbb{N}$  be large enough such that  $\frac{b-a}{N} < \delta$ . Define the partition as

$$x_j := a + j \left( \frac{b-a}{N} \right) \quad (5.77)$$

for all  $j = 0, \dots, N$ . Define  $g_\varepsilon(x) := g(x_{i-1}) + m_i(x - x_{i-1})$  for all  $x \in [x_{i-1}, x_i]$  with  $m_i = \frac{g(x_i) - g(x_{i-1})}{x_i - x_{i-1}}$ . Now, let  $x \in [a, b]$ . Then,  $x \in [x_i, x_{i+1})$  for some  $i = 0, \dots, N$  (the same proof holds for  $x = b$  as well). We know  $|g(x) - g(x_i)| < \varepsilon/2$ . Since  $g_\varepsilon$  is affine linear on  $[x_i, x_{i+1}]$ ,

$$|g_\varepsilon(x_i) - g_\varepsilon(x)| < |g_\varepsilon(x_i) - g_\varepsilon(x_{i-1})| = |g(x_i) - g(x_{i-1})| < \frac{\varepsilon}{2} \quad (5.78)$$

Therefore, by triangle inequality,

$$|g(x) - g_\varepsilon(x)| \leq |g(x) - g(x_i)| + |g(x_i) - g_\varepsilon(x)| \quad (5.79)$$

$$\leq |g(x) - g(x_i)| + |g_\varepsilon(x_i) - g_\varepsilon(x)| \quad (5.80)$$

$$\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon \quad (5.81)$$

□

**Theorem 5.4.2** (Shallow ReLU networks are universal approximators). *Define*



the set of all shallow ReLU networks as

$$G := \left\{ g \in C[0, 1] : g(x) := \sum_{j=1}^N \alpha_j \text{ReLU}(x + \theta_j) - \beta \text{ for some } \alpha_j, \theta_j, \beta \in \mathbb{R}, N \in \mathbb{N} \right\} \quad (5.82)$$

Then,  $G$  is dense in  $C[0, 1]$  under the sup-norm.

*Proof.* Consider the equidistant partition  $0 = x_0 < x_1 < \dots < x_N = 1$ . We need to show  $\alpha_i, \theta_i, \beta$  can be chosen such that

$$G(x) = \sum_{j=0}^{N-1} \alpha_j \text{ReLU}(x + \theta_j) + \beta \quad (5.83)$$

becomes piecewise linear as in [Lemma 5.4.1](#). Set  $\theta_j := -x_j$ . Then,

$$G(x_k) = \sum_{j=0}^{N-1} \alpha_j \text{ReLU}(x_k - x_j) + \beta = \frac{1}{N} \sum_{j < k} \alpha_j (k - j) + \beta \quad (5.84)$$

The  $N + 1$  parameters  $\alpha_0, \dots, \alpha_{N-1}, \beta$  are uniquely determined by the constraints  $G(x_k) = g(x_k)$  for all  $k = 0, \dots, N$ . Denoting  $y_k := g(x_k)$  for ease of notation,

$$y_0 = G(x_0) = \beta \quad (5.85)$$

$$y_1 = G(x_1) = \frac{\alpha_0}{N} + \beta \quad (5.86)$$

$$y_2 = G(x_2) = \frac{\alpha_0}{N}(2) + \frac{\alpha_1}{N} + \beta \quad (5.87)$$

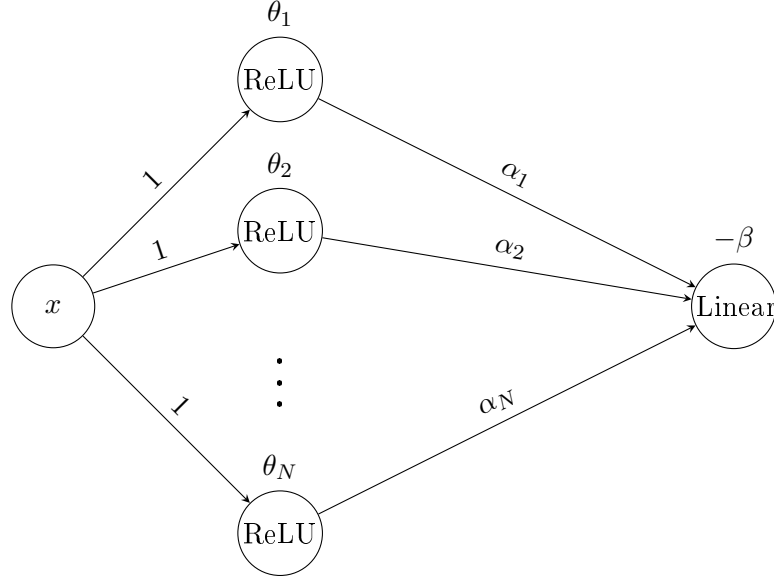
$$\vdots \quad (5.88)$$

$$y_N = G(x_N) = \frac{\alpha_0}{N}(N) + \dots + \frac{\alpha_{N-1}}{N}(1) + \beta \quad (5.89)$$

which reduces to

$$\beta = y_0 \quad (5.90)$$

$$\alpha_0 = N(y_1 - y_0) \quad (5.91)$$



**Figure 5.5:** Shallow ReLU network mentioned in [Theorem 5.4.2](#)

$$\alpha_1 = N(y_2 - 2y_1 - y_0) \quad (5.92)$$

$$\vdots \quad (5.93)$$

$$\alpha_{N-1} = N(y_N - 2y_{N-1} - y_{N-2}) \quad (5.94)$$

□

Now, we shall show that shallow (1-layer) discriminatory (defined ahead) are in dense in  $c[0, 1]^n$ . This is the famous Cybenko's theorem. Proving this is not as easy as it was in the one-dimensional case - we need tools from functional analysis.

**Lemma 5.4.3.** *Let  $U$  be a linear subspace of a normed linear space  $X$ . Let  $x_0 \in X$  such that  $\|x_0 - u\| \geq \delta$  for all  $u \in U$ . Then, there exists a bounded linear functional  $L$  on  $X$  such that*

1.  $\|L\| \leq 1$
2.  $L(u) = 0$  for all  $u \in U$
3.  $L(x_0) = \delta$ .

*Proof.* Define  $T := \{t \in X : t = u + \lambda x_0 \text{ for some } u \in U, \lambda \in \mathbb{R}\}$ . Define

$L : T \rightarrow \mathbb{R}$  by  $L(t) := L(u + \lambda x_0) =: \lambda \delta$ . Note that  $L$  is a well-defined linear functional. Now, let  $u \in U$ . Then,  $-\frac{u}{\lambda} \in U$  for any  $\lambda$ . Thus,

$$\left\| x_0 + \frac{u}{\lambda} \right\| = \left\| x_0 - \left( \frac{-u}{\lambda} \right) \right\| \geq \delta \quad (5.95)$$

$$\iff |\lambda| \delta \leq \|u + \lambda x_0\| \quad (5.96)$$

Thus, for all  $t \in T$ ,

$$L(t) = L(u + \lambda x_0) = \lambda \delta \leq \|u + \lambda x_0\| = \|t\| \quad (5.97)$$

which means that  $L$  is a bounded linear functional. Using Hahn-Banach extension theorem ([Appendix E](#)) with  $V = X$ ,  $p = \|\cdot\|$ ,  $W = T$  and  $g = L$ , we conclude that  $L$  can be extended to a linear functional on  $X$  such that  $L(x) \leq \|x\| = p(x)$  for all  $x \in X$ . Thus,  $\|L\| \leq 1$ , so it is bounded. Moreover,

$$L(u) = L(u + 0 \cdot x_0) = 0 \cdot \delta = 0, \quad \forall u \in U \quad (5.98)$$

$$L(x_0) = L(0 + 1 \cdot x_0) = 1 \cdot \delta = \delta > 0 \quad (5.99)$$

□

Now, consider the normed linear space  $(C(I_n), \|\cdot\|_\infty)$  where  $I_n := [0, 1]^n$ . Define  $M(I_n)$  to be the space of all finite, signed, Borel measures on  $I_n$  ([Appendix C](#)).

**Lemma 5.4.4.** *Let  $U$  be a linear, non-dense subspace of  $C(I_n)$ . Then, there exists  $\mu \in M(I_n)$  such that*

$$\int_{I_n} h d\mu = 0 \quad (5.100)$$

for all  $h \in U$ .

*Proof.* From [Lemma 5.4.3](#), there exists a bounded linear functional  $L : C(I_n) \rightarrow \mathbb{R}$  such that  $L \not\equiv 0$  on  $C(I_n)$  and  $L|_U = 0$ . Applying representation theorem on  $C(I_n)$

([Appendix E](#)), there exists a unique  $\mu \in M(I_n)$  such that

$$L(f) = \int_{I_n} f d\mu \quad (5.101)$$

for all  $f \in C(I_n)$ . In particular, for all  $h \in U$ ,  $L(h) = \int_{I_n} h d\mu = 0$ .  $\square$

**Definition 5.4.1** (Discriminatory function). Let  $\mu \in M(I_n)$ . A function  $\sigma$  is called discriminatory for  $\mu$  if

$$\int_{I_n} \sigma(w^T x + \theta) d\mu(x) = 0 \quad \forall w \in \mathbb{R}^n, \theta \in \mathbb{R} \implies \mu = 0 \quad (5.102)$$

**Theorem 5.4.5** (Cybenko). *Let  $\sigma$  be a continuous, discriminatory function. Define the set of all shallow discriminatory networks as*

$$G := \left\{ g \in C[I_n] : g(x) := \sum_{j=1}^N \alpha_j \sigma(w_j^T x + \theta_j) \text{ for some } \right. \\ \left. w_j \in \mathbb{R}^n, \alpha_j, \theta_j \in \mathbb{R}, N \in \mathbb{N} \right\} \quad (5.103)$$

*Then,  $G$  is dense in  $C[0, 1]^n$  under the sup-norm.*

*Proof.* Since  $\sigma$  is continuous,  $G$  is a linear subspace of  $C(I_n)$ . Now, assume  $G$  is not dense in  $C(I_n)$ . By [Lemma 5.4.4](#), there exists  $\mu \in M(I_n)$  such that

$$\int_{I_n} h d\mu = 0 \quad \forall h \in G \quad (5.104)$$

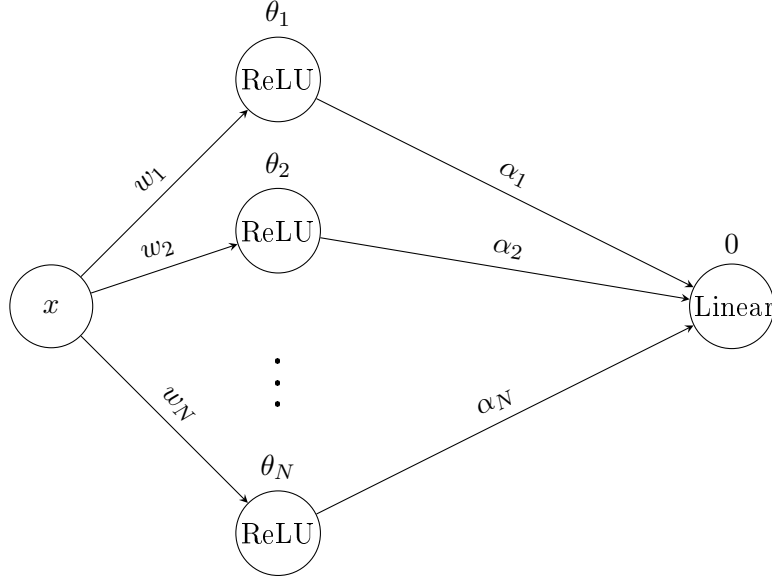
$$\iff \sum_{j=1}^N \alpha_j \int_{I_n} \sigma(w_j^T x + \theta_j) d\mu = 0 \quad \forall w_j \in \mathbb{R}^n, \alpha_j, \theta_j \in \mathbb{R}, N \in \mathbb{N} \quad (5.105)$$

$$\implies \int_{I_n} \sigma(w^T x + \theta) d\mu = 0 \quad \forall w \in \mathbb{R}^n, \theta \in \mathbb{R} \quad (5.106)$$

$$\implies \mu \equiv 0 \quad (5.107)$$

$$\implies L \equiv 0 \quad (5.108)$$

which is a contradiction since  $L \not\equiv 0$  ([Lemma 5.4.4](#)).  $\square$



**Figure 5.6:** Shallow discriminatory network mentioned in [Theorem 5.4.5](#)

Not just continuous functions, it can even be shown that neural networks are dense in measurable functions (under a weaker distance however).

The density in measurable functions is shown with respect to the following distance over Borel-measurable functions from  $\mathbb{R}^n$  to  $\mathbb{R}$ .

$$d_\mu(f, g) := \inf\{\varepsilon > 0 : \mu(|f - g| > \varepsilon) < \varepsilon\} \quad (5.109)$$

where  $\mu$  is a probability measure on  $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n))$ . We start with a few characterizations of convergence in this distance. Define  $x \wedge y := \min\{x, y\}$ .

**Lemma 5.4.6.** *The following are equivalent.*

1.  $\lim_{j \rightarrow \infty} d_\mu(f_j, f) = 0$
2. For all  $\varepsilon > 0$ ,  $\lim_{j \rightarrow \infty} \mu(\{|f_j - f| > \varepsilon\}) = 0$
3.  $\lim_{j \rightarrow \infty} \int (|f_j - f| \wedge 1) d\mu = 0$

*Proof.* We first prove  $1 \implies 2$ . From definition of  $d_\mu$ , for all  $\varepsilon > 0$  such that  $\mu(|f - g| > \varepsilon) < \varepsilon$ , we have  $d_\mu(f_j, f) \leq \varepsilon$ . Since  $d_\mu(f_j, f) \rightarrow 0$ , choose sequence

$\varepsilon = \frac{1}{n_j} \rightarrow 0$  such that

$$\mu\left(|f - g| > \frac{1}{n_j}\right) < \frac{1}{n_j} \quad (5.110)$$

Let 2 not hold. Then, there exists  $\varepsilon_0 > 0$  such that  $\mu(|f - g| > \varepsilon) > \varepsilon_0$ , a contradiction.

2  $\implies$  1 is trivially true by definition of limit. We shall now prove equivalence of 2 and 3. Let  $\varepsilon \in (0, 1)$ . Then,

$$\varepsilon 1_{(\varepsilon, \infty)}(x) \leq x \wedge 1 \leq \varepsilon + 1_{(\varepsilon, \infty)}(x) \quad (5.111)$$

for all  $x \geq 0$ . Replacing  $x$  by  $|f_j(x) - f(x)|$  and integrating with respect to  $\mu$  gives

$$\varepsilon \mu(\{|f_j - f| > \varepsilon\}) \leq \int (|f_j - f| \wedge 1) d\mu \quad (5.112)$$

$$\varepsilon + \mu(\{|f_j - f| > \varepsilon\}) \geq \int (|f_j - f| \wedge 1) d\mu \quad (5.113)$$

Now, 3  $\implies$  2 holds due to (5.112) and 2  $\implies$  3 holds due to (5.113).  $\square$

*Remark 5.4.1.* The above lemma tells that convergence in probability ([Appendix D](#)) is characterized by convergence in  $d_\mu$  metric. Moreover, the topologies induced by  $\rho_\mu := \mathbb{E}[|f - g| \wedge 1]$  and  $d_\mu$  are equivalent. Thus, a set is  $d_\mu$ -dense iff it's  $\rho_\mu$ -dense. This is used in the next lemma.

We shall now prove a crucial theorem that allows us to jump from uniform convergence over compact sets to convergence in  $d_\mu$ -distance.

**Lemma 5.4.7.** *Let  $\{f_j\}_j$  be a sequence of measurable functions that converges uniformly on compact sets to  $f$ , ie. for all compact sets  $K \subset \mathbb{R}^n$ ,*

$$\sup_{x \in K} |f_j(x) - f(x)| \rightarrow 0 \quad (5.114)$$

*Then,  $d_\mu(f_j, f) \rightarrow 0$ .*

*Proof.* By [Lemma 5.4.6](#), it suffices to show that for all  $\varepsilon > 0$ , there exists  $N \in \mathbb{N}$

such that for all  $j \geq N$ ,

$$\int (|f_j - f| \wedge 1) d\mu < \varepsilon \quad (5.115)$$

We first construct a compact set  $K$  such that  $\mu(\mathbb{R}^n \setminus K) < \frac{\varepsilon}{2}$ . We know that

$$\mathbb{R}^n = \bigcup_{k \in \mathbb{N}} B(0, k) = \limsup_{k \in \mathbb{N}} B(0, k) \quad (5.116)$$

From continuity from below of the probability measure  $\mu$ ,

$$1 = \mu(\mathbb{R}^n) = \lim_{k \rightarrow \infty} \mu(B(0, k)) \quad (5.117)$$

Thus, for  $k$  large enough,  $\mu(B(0, k)) > 1 - \frac{\varepsilon}{2}$ . Setting  $K := B(0, k)$ , we get that  $\mu(\mathbb{R}^n \setminus K) < \frac{\varepsilon}{2}$ .

Now, we shall bound the integral (5.115). First, note that

$$\int_{\mathbb{R}^n \setminus K} (|f_j(x) - f(x)| \wedge 1) d\mu(x) \leq \int_{\mathbb{R}^n \setminus K} 1 d\mu(x) = \mu(\mathbb{R}^n \setminus K) < \frac{\varepsilon}{2} \quad (5.118)$$

Now, since  $\sup_{x \in K} |f_j(x) - f(x)| \rightarrow 0$ , we can find  $N \in \mathbb{N}$  such that for all  $j \geq N$ ,

$$\sup_{x \in K} |f_j(x) - f(x)| < \frac{\varepsilon}{2} \quad (5.119)$$

Thus, we get

$$\int_K (|f_j(x) - f(x)| \wedge 1) d\mu(x) \leq \int_K |f_j(x) - f(x)| d\mu(x) \leq \frac{\varepsilon}{2} \mu(K) < \frac{\varepsilon}{2} \quad (5.120)$$

since  $\mu(K) < 1$ . Adding up (5.118) and (5.120), we get  $\int (|f_j - f| \wedge 1) d\mu < \varepsilon$  for all  $j \geq N$ .  $\square$

Owing to Lemma 5.4.7, we prove an extension to Cybenko's theorem to uniform convergence over compact sets, which will be useful for our final theorem later on.

**Theorem 5.4.8** (Extension to Cybenko). *Let  $\sigma$  be a continuous, discriminatory*

function. Then, finite sums of the form

$$G(x) = \sum_{j=1}^N \alpha_j \sigma(w_j^T x + \theta_j) \quad (5.121)$$

for all  $w_j \in \mathbb{R}^n$ ,  $\theta_j, \alpha_j \in \mathbb{R}$  and  $N \in \mathbb{N}$  are uniformly dense on compact sets in  $C(\mathbb{R}^n)$ , ie. for any  $f \in C(\mathbb{R}^n)$ , there exists a sequence  $\{G_m\}_m$  of functions of the above form such that  $G_m \rightrightarrows f$  on any compact subset of  $\mathbb{R}^n$ .

*Proof.* We start by proving that for any  $f \in C(\mathbb{R}^n)$  and compact subset  $K$  of  $\mathbb{R}^n$ , there exists a sequence  $\{G_m^{(K)}\}_m$  of functions of form (5.121) such that it converges uniformly to  $f$  on  $K$ .

To do so, note that Cybenko's theorem holds for any  $C(K)$  instead of  $C(I_n)$ , with an analogous proof. Thus, for a fixed compact set  $K$ , finite sums of form (5.121) are dense in  $C(K)$ . Since for any  $f \in C(\mathbb{R}^n)$ ,  $f|_K \in C(K)$ , there exists a sequence of (5.121) such that  $G_m^{(K)} \rightrightarrows f$  on  $K$ .

We now define an ascending sequence of compact sets  $(K_j)$  that goes to  $\mathbb{R}^n$ :

$$K_j := \{x \in \mathbb{R}^n : \|x\| \leq j\} \quad (5.122)$$

We will find the desired sequence using a diagonalization argument over these  $K_j$ 's. For each  $K_j$ , consider the sequence  $\{G_m^{(j)}\}_m$  such that  $G_m^{(j)} \rightrightarrows f$  on  $K_j$ . Since  $K_j \subset K_{j+1}$ ,  $G_m^{(p)} \rightrightarrows f$  on  $K_j$  for all  $j \leq p$ .

Now, construct the sequence  $G_k := G_k^{(k)}$ , for all  $k \in \mathbb{N}$ . Select the smallest  $j_0$  such that  $K \subset K_{j_0} \subset K_{j_0+1} \subset \dots$ . Using the previous property,  $\{G_k\}_{k \geq j_0}$  converges to  $f$  uniformly on  $K_{j_0}$  and hence on  $K$ .  $\square$

We shall now show the main result that allows us to jump to measurable functions - continuous functions are dense in measurable functions under the  $d_\mu$  distance.

**Theorem 5.4.9.**  *$C(\mathbb{R}^n)$  is dense in the set of all measurable functions under the  $d_\mu$  distance.*



*Proof.* Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be measurable. Fix  $\varepsilon > 0$ . We shall show there exists  $g \in C(\mathbb{R}^n)$  such that  $d_\mu(f, g) < \varepsilon$ . Or equivalently, we can show that there exists  $g \in C(\mathbb{R}^n)$  such that  $\rho_\mu(f, g) < \varepsilon$ , due to [Lemma 5.4.6](#).

We start by showing there exists  $N \in \mathbb{N}$  such that  $\mu(\{|f| > N\}) < \frac{\varepsilon}{2}$ . Define  $A_n := \{|f| > n\}$ . Since  $A_{n+1} \subseteq A_n$  and  $A_n = f^{-1}(n, \infty) \cup f^{-1}(-\infty, -n)$  is Borel measurable,  $\{A_n\}_n$  is a decreasing sequence of measurable sets converging to  $\{|f| = \infty\}$ . From sequential continuity and the fact that  $f$  is finite a.e.,  $\mu(A_n) \rightarrow 0$ . Thus, we can choose  $N$  large enough such that  $\mu(\{|f| > N\}) < \frac{\varepsilon}{2}$ .

We shall now show that  $\rho_\mu(f, h) < \frac{\varepsilon}{2}$ , where  $h(x) := f(x)1_{|f| < N}(x)$ . Now,

$$|f(x) - h(x)| = \begin{cases} 0, & |f(x)| < N \\ |f(x)|, & |f(x)| \geq N \end{cases} \quad (5.123)$$

Thus, by previous argument,  $|f(x) - h(x)|$  is nonzero on a set with small measure.

Now,

$$\int_{\mathbb{R}^n} |f - h| \wedge 1 d\mu = \int_{|f| < N} 0 \wedge 1 d\mu + \int_{|f| \geq N} |f| \wedge 1 d\mu \quad (5.124)$$

$$= \int_{|f| \geq N} |f| \wedge 1 d\mu \quad (5.125)$$

$$= \int_{|f| \geq N} 1 d\mu \quad (5.126)$$

$$= \mu(\{|f| \geq N\}) \quad (5.127)$$

$$\leq \frac{\varepsilon}{2} \quad (5.128)$$

since  $N \geq 1$ . Now, we prove that there exists  $g \in C(\mathbb{R}^n)$  such that  $\rho_\mu(h, g) < \frac{\varepsilon}{2}$ . Note that  $h$  is measurable and bounded (with  $|h| < N$ ). Thus,  $h$  can be written as limit of a sequence  $\{s_k\}$  of simple functions. We may further assume that  $|s_k| < N$  for large  $k$ .

Since  $\mu(\mathbb{R}^n) = 1 < \infty$  and  $s_k \rightarrow h$ , by bounded convergence theorem,

$$\lim_{k \rightarrow \infty} \int_{\mathbb{R}^n} |h(x) - s_k(x)| \wedge 1 d\mu(x) = 0 \quad (5.129)$$

ie. there exists  $k_0 \in \mathbb{N}$  such that  $\rho_\mu(h, s_k) < \frac{\varepsilon}{4}$  for all  $k \geq k_0$ .

Now, let  $s_k(x) := \sum_{i=1}^k \alpha_i 1_{A_i}(x)$  and consider compact sets  $K_i \subset A_i$  such that  $\mu(A_i \setminus K_i) < \frac{\varepsilon}{4kN}$ . Let

$$g_i(x) := \begin{cases} 1, & x \in K_i \\ 0, & x \notin A_i \end{cases} \quad (5.130)$$

such that  $g_i$  is continuous ([Appendix C](#)). Let  $g(x) := \sum_{i=1}^k \alpha_i g_i(x) \in C(\mathbb{R}^n)$ . Then,

$$\int_{\mathbb{R}^n} |g - s_k| \wedge 1 d\mu \leq \sum_{i=1}^k |\alpha_i| \int_{\mathbb{R}^n} |g_i(x) - 1_{A_i}(x)| \wedge 1 d\mu(x) \quad (5.131)$$

$$\leq \sum_{i=1}^k |\alpha_i| \int_{A_i \setminus K_i} 1 d\mu(x) \quad (5.132)$$

$$= \sum_{i=1}^k |\alpha_i| \mu(A_i \setminus K_i) \quad (5.133)$$

$$\leq \frac{\varepsilon}{4kN} \sum_{i=1}^k |\alpha_i| \quad (5.134)$$

$$\leq \frac{\varepsilon}{4} \quad (5.135)$$

ie.  $\rho_\mu(g, s_k) < \frac{\varepsilon}{4}$ . Thus, by triangle inequality,  $\rho_\mu(h, g) < \frac{\varepsilon}{2}$ . Finally, by triangle inequality again,

$$\rho_\mu(f, g) \leq \rho_\mu(f, h) + \rho_\mu(h, g) \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon \quad (5.136)$$

□

**Theorem 5.4.10** (Shallow discriminatory networks are dense in measurable func-

tions). Define the set of all shallow sigmoidal networks as

$$G := \left\{ g \in C[0, 1]^n : g(x) := \sum_{j=1}^N \alpha_j \sigma(w_j^T x + \theta_j) \text{ for some } \right. \\ \left. w_j \in \mathbb{R}^n, \alpha_j, \theta_j \in \mathbb{R}, N \in \mathbb{N} \right\} \quad (5.137)$$

where  $\sigma$  is some continuous discriminatory activation function and  $n \in \mathbb{N}$ . Then,  $G$  is dense in the space of all Borel measurable functions on  $\mathbb{R}^n$  under the  $d_\mu$  distance.

*Proof.* Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be measurable. Fix  $\varepsilon > 0$ . From [Theorem 5.4.9](#) there exists  $g \in C(\mathbb{R}^n)$  such that  $d_\mu(f, g) < \frac{\varepsilon}{2}$ . From [Theorem 5.4.8](#) there exists sequence  $\{G_k\}_k$  in  $G$  such that  $G_k \rightrightarrows g$  on compact sets. From [Lemma 5.4.7](#)  $d_\mu(G_k, g) \rightarrow 0$ , so  $d_\mu(G_{k_0}, g) < \frac{\varepsilon}{2}$  for large enough  $k_0$ . Thus, using triangle inequality,

$$d_\mu(f, G_{k_0}) \leq d_\mu(f, g) + d_\mu(g, G_{k_0}) \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon \quad (5.138)$$

□

A major issue with all these results is that they are existence results - they tell a ‘good’ neural network always exists, but does not tell us how to get it. This is perhaps not possible to solve in general.

However, we can improve upon these results if we could also give the rate at which we can approximate a function, given a particular method of optimization of course. One such result was recently proven by Kohler, [2025](#), who showed that neural networks can approximate any  $(p, C)$ -smooth function with near optimal rate of convergence. The optimal rate of convergence for  $(p, C)$ -smooth functions was given by Stone in the seminal paper - Stone, [1982](#).

**Definition 5.4.2** ( $(p, C)$ -smooth function). Let  $p = q + s$  for some  $q \in \mathbb{N}_0, s \in (0, 1]$ .  $m : \mathbb{R}^d \rightarrow \mathbb{R}$  is called  $(p, C)$ -smooth if for all  $\alpha \in \mathbb{N}_0^d$  with  $\|\alpha\|_1 = q$ , partial

derivatives  $\partial^q m / \partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}$  exist and

$$\left| \frac{\partial^q m}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}(x) - \frac{\partial^q m}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}(z) \right| \leq C \|x - z\|^s \quad (5.139)$$

for all  $x, z \in \mathbb{R}^d$ , where  $C > 0$ .

**Theorem 5.4.11** (Stone, 1982). *For a  $(p, C)$ -smooth regression function defined on  $\mathbb{R}^d$ , the optimal minimax rate of convergence for expected  $L_2$ -error is*

$$n^{\frac{-2p}{2p+d}} \quad (5.140)$$

**Theorem 5.4.12** (Kohler, 2025). *Consider a  $(p, C)$ -smooth regression function. Then, there exists a suitable choice of hyperparameters (depth, width, number of sub-NNs, learning rate, etc.) such that expected  $L_2$ -error of linear combination of sub-NNs converges to zero with near optimal rate, ie. for all  $\varepsilon > 0$ ,*

$$\mathbb{E} \int |m_n(x) - m(x)|^2 P_X(dx) \leq c_0 n^{(\frac{-2p}{2p+d} + \varepsilon)} \quad (5.141)$$

## 5.5 Information Processing

Now, let's take a somewhat more algebraic approach to understanding neural networks, by studying the  $\sigma$ -algebras generated by the input and output random variables.

Let  $(\Omega, \mathcal{H}, \mathbb{P})$  be the input probability space. Let  $X : \Omega \rightarrow \mathbb{R}^n$  be the input random vector and  $Y = g(X)$  be the output random vector, where  $g$  is the neural network.

Define  $\mathcal{I} := \sigma(X)$  and  $\mathcal{E} := \sigma(Y)$ .  $\mathcal{I}$  contains events that can be observed from input data and  $\mathcal{E}$  contains events observed at the output of the network. Since  $Y = g(X)$ ,  $\mathcal{E} \subseteq \mathcal{I}$ , ie. output information is coarser than input information.

### 5.5.1 Lost information and Compressible layers

Let  $X^{(l)}$  be the random variable defining output of the  $l^{\text{th}}$  layer. Define  $I^{(l)} := \sigma(X^{(l)})$ . Then, information lost by the  $l^{\text{th}}$  layer is

$$\mathcal{L}^{(l)} := \sigma(I^{(l-1)} \setminus I^{(l)}) \quad (5.142)$$

We say the  $l^{\text{th}}$  layer is uncompressed if lost information is trivial, ie.

$$\mathcal{L}^{(l)} = \{\phi, \Omega\} \quad (5.143)$$

It would perhaps make sense to not lose any ‘important’ information during processing. Thus, it would make sense to have each layer as uncompressed.

**Theorem 5.5.1** (Sufficient conditions to have an uncompressed layer). *Consider a layer of a neural network such that*

1. *Activation  $\phi$  is invertible.*
2. *Number of neurons in previous and current layers are same.*
3. *Weights matrix of weights between the previous and current layer is nonsingular.*

*Then, the current layer is uncompressed.*

*Proof.* Since  $\phi$  is invertible,

$$W^{(l)T} X^{(l-1)} = \phi^{-1}(X^{(l)}) + B^{(l)} \quad (5.144)$$

$$\implies X^{(l-1)} = W^{(l)-T}(\phi^{-1}(X^{(l)}) + B^{(l)}) \quad (5.145)$$

since  $W^{(l)}$  is a square, invertible matrix. Thus,  $\sigma(X^{l-1}) \subseteq \sigma(X^l)$ , which means that  $\sigma(X^{l-1}) = \sigma(X^l)$ , since because of the forward pass relation we always have  $\sigma(X^l) \subseteq \sigma(X^{l-1})$ .  $\square$

However, not losing out on any information gives worse results in practice because the input data usually contains information that is not useful for the task at hand. Thus, we usually need to get rid of useless information.

A classic example is classifying images - an image is usually fed into a neural network as a huge vector with each component telling the value of a pixel. It is intuitively clear that each individual pixel's information is not needed to classify an image. Thus, we need to 'coarsen' the input information.

For images, this is popularly done by Convolutional Neural Networks (CNNs), which will be discussed in the next section.

### 5.5.2 Bottleneck Principle

Let  $Z$  be the target random variable.

**Definition 5.5.1** (Entropy). The information contained in  $X$  is given by

$$H(X) := - \sum_k p_k \ln(p_k) \quad (5.146)$$

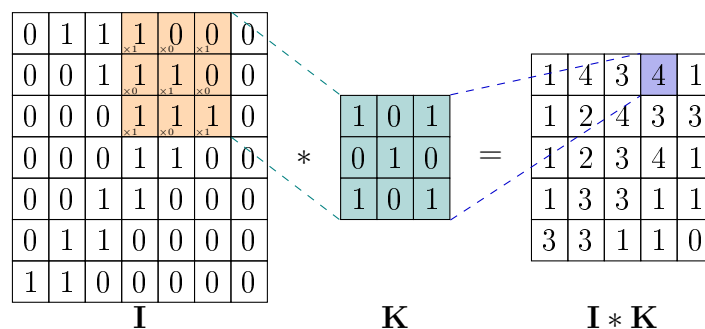
**Definition 5.5.2** (Conditional entropy). The information contained in  $Y$  if  $X$  is known is given by

$$H(Y|X) := \sum_i \sum_j p(x_i, y_j) \ln(p(y_j|x_i)) \quad (5.147)$$

**Definition 5.5.3** (Mutual Information). The information conveyed by  $X$  about  $Y$  is given by

$$I(Y|X) := H(Y) - H(Y|X) \quad (5.148)$$

The bottleneck principle is as follows: 'Pass information provided by  $X$  about  $Z$  through a 'bottleneck' formed by  $Y$  in the most efficient way, ie. to minimize  $I(X, Y)$  subject to fixed  $I(X, Z)$ '.



**Figure 5.7:** Convolution operation (<https://tikz.net/conv2d/>)

## 5.6 Other Architectures

The fully-connected neural networks we have been seeing until now have, despite being very powerful, quite a few drawbacks. In particular, they are not so great at working with pictures or sequences. We shall now look at these drawbacks and see how we can get beyond them.

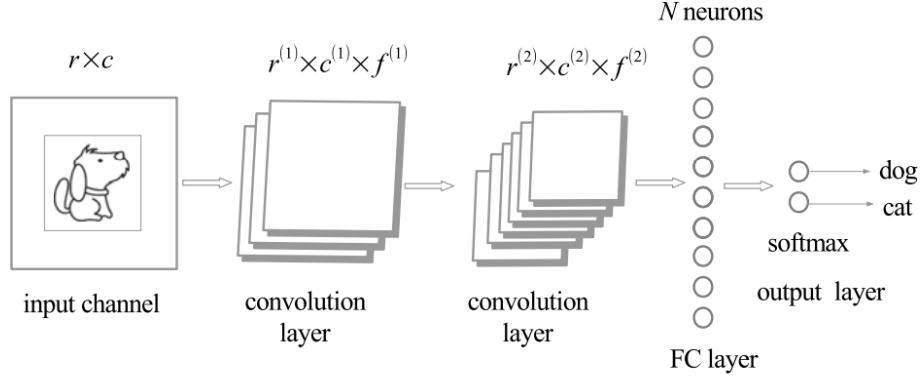
### 5.6.1 Convolutional Neural Networks

A CNN is a standard NN but with sparse weights matrix with weight sharing, ie. the same weights/biases at multiple places. This leads to a huge loss of unnecessary information and also leads to faster training times by taking advantage of the weight sharing. Convolutions are usually paired with pooling, which empirically leads to better results.

Say we want to classify a color image. Color images have three ‘channels’ of information - one for each primary color - red, blue, green. Each image can be represented as a triple of matrices, with each matrix defining a color channel and entry in  $\{0, \dots, 225\}$  representing how much of that color is present in that pixel.

Then, we can define the  $l^{\text{th}}$  layer output of the  $(i, j)^{\text{th}}$  entry of the  $k^{\text{th}}$  channel to be

$$X_{ijk}^{(l)} = \phi \left( \sum_r \sum_p X_{i+p, j+r, k}^{(l-1)} w_{prk}^{(l)} + b_k^{(l)} \right) \quad (5.149)$$



**Figure 5.8:** CNN architecture (Calin, 2020, pg. 528).

where  $i \in [r(l)], j \in [c(l)]$  run over the number of rows and columns for the  $l^{\text{th}}$  layer,  $k \in [3]$  for the three channels. Thus, the output becomes  $X^{(l)} \in \mathbb{R}^{r(l) \times c(l) \times 3 \times f(l)}$ , where  $f(l)$  is the number of kernels used for the  $l^{\text{th}}$  layer.

The idea behind using multiple kernels for each layer is that each kernel is supposed to extract different kinds of features of the image. Customarily, as  $l$  increases,  $r(l)$  and  $c(l)$  decrease while  $f(l)$  increases. At the end, a standard fully connected layer is introduced to combine all the information extracted by the different kernels of the last convolutional layer.

### 5.6.2 Recurrent Neural Networks

Standard NNs don't perform well on sequential data due to their inability to account for correlations between data points. For sequential data, Recurrent Neural Networks (RNNs) were introduced.

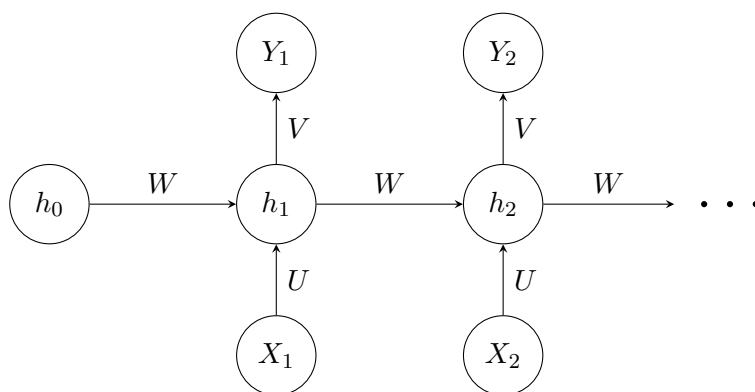
A simple RNN can be defined as follows.

$$h_t = \tanh(Wh_{t-1} + UX_t + b) \quad (5.150)$$

$$Y_t = Vh_t + c \quad (5.151)$$

However, RNNs suffer from many problems - a major one being that if the length





**Figure 5.9:** Simple RNN architecture (Calin, 2020, pg. 546).

of a sequence is large, it is hard for a RNN to account for long term dependencies. To fix this problem, the LSTMs were introduced.

An LSTM network is made up of LSTM cells joined in series that are capable of learning long-term dependencies by using gates to optionally allow information to pass. For each LSTM cell  $t$ , there is a internal cell state, denoted by  $C_t$  and a hidden cell state, denoted by  $h_t$ . The internal cell state is like a conveyor belt - information can pass along it without being altered much. This allows LSTMs to preserve long-term dependencies. Each cell uses three gates: forget, update and output.

**Forget gate** represents fraction of the past state that will be forgotten. It tells the cell what to forget.

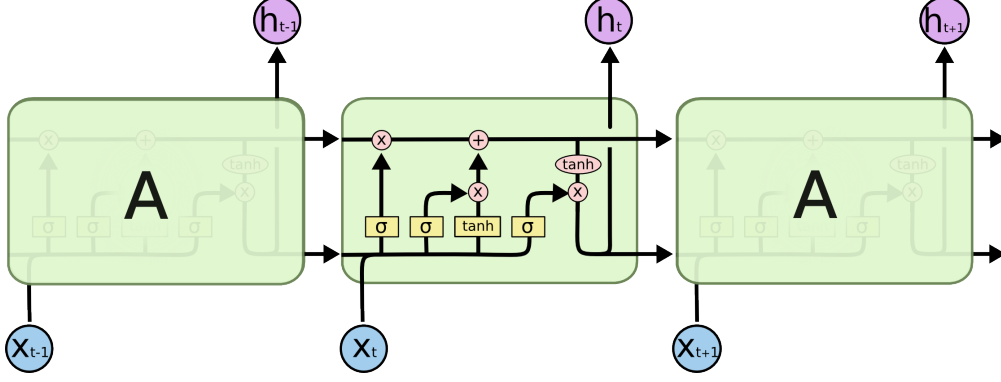
$$f_t = \sigma(W_f h_{t-1} + U_f X_t + b_f) \quad (5.152)$$

**Update gate** updates the current internal cell state using the previous internal cell state and a new candidate. It tells the cell how to update itself.

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \quad (5.153)$$

where  $i_t$  is a scale factor defined by

$$i_t = \sigma(W_i h_{t-1} + U_i X_t + b_i) \quad (5.154)$$



**Figure 5.10:** The LSTM architecture (<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>)

and  $\tilde{C}_t$  represents the candidate that could be added to the internal state.

$$\tilde{C}_t = \tanh(W_c h_{t-1} + U_c X_t + b_c) \quad (5.155)$$

Thus,  $f_t C_{t-1}$  is the information left after forgetting from previous internal state value and  $i_t \tilde{C}_t$  is the fraction of the new candidate that updates  $C_t$ .

**Output gate** provides value of the hidden state as a product between itself and the current state. It tells the cell what to output.

$$o_t = \sigma(W_o h_{t-1} + U_o X_t + b_o) \quad (5.156)$$

and the hidden state value is given by

$$h_t = o_t \tanh(C_t) \quad (5.157)$$

Note that gates are represented with a sigmoid layer followed by element-wise multiplication  $\otimes$ . Since sigmoid outputs values in  $(0, 1)$ , a value of 0 means no information will be passed on (since it gets multiplied by 0) and a value of 1 means all information will be passed on (since it gets multiplied by 1).

On the other hand, tanh push values to  $(-1, 1)$  which is not meant to represent a gate but rather just a standard neuron meant to be processing information.

However, due to the complicated structure of an LSTM cell, there were some problems. Firstly, the vanishing gradient problem was not fully solved - it still occurred noticeably if the sequence length was large enough. Secondly, LSTMs processed information sequentially - to get  $C_t$ , we need to calculate all the previous states. This made LSTMs computationally intensive.

To fix these problems, transformers were introduced by Vaswani et al., [2017](#).

### 5.6.3 Transformers

Say the input and output are a sequences in  $\mathbb{R}^d$  of length  $n$ . Then, we define  $\text{AttHead}: \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times d}$  as

$$\text{AttHead}(X) := \text{softmax} \left( \frac{(XW_Q)(XW_K)^T}{\sqrt{d}} \right) XW_V \quad (5.158)$$

for trainable matrices  $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$ .  $XW_Q$  is called the query matrix - it is meant to ask a relevant question to the data. The key matrix  $XW_K$  is meant to answer it. Their inner product is meant to represent how good of an answer it was. If the inner product is high, we say that the key ‘attends to’ the query. After dividing by  $\sqrt{d}$  to prevent the product from becoming too large if  $d$  is large, we pass it through softmax to convert it into a probability distribution that represents which coordinates do the keys attend the most to the query to. Finally, we multiply this by the value matrix  $XW_V$  to update the original sequence with the appropriate changes.

One attention head to meant to ask one type of questions to the data. So, for a more effective architecture, there should be multiple heads asking multiple questions. We can combine outputs of these multiple heads using another trainable matrix. A self-attention layer  $\text{Att}: \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times d}$  is thus defined as

$$\text{Att}(X) := [\text{AttHead}_1(X), \dots, \text{AttHead}_h(X)]W_O \quad (5.159)$$

for number of heads  $h \in \mathbb{N}$  and  $W_O \in \mathbb{R}^{hd \times d}$ . We can then put these attention

layers in a sequence and combine them to get a network. A transformer block  $\text{Block}: \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times d}$  is defined as

$$\text{Block}(X) := \text{FC}(X + \text{Att}(X)) \quad (5.160)$$

where FC is a fully connected neural network applied row-wise. Finally, a transformer network is defined as

$$T(X) := (\text{Block}_l \circ \dots \circ \text{Block}_1)(X) \quad (5.161)$$

This gets rid of the problem of not being able to handle long-term dependencies because all entries of the sequence are being processed entirely at once (by the attention head). Moreover, this also means the process can be parallelized (because of the matrix products). Thus, transformers become much better and faster in practice than LSTMs.

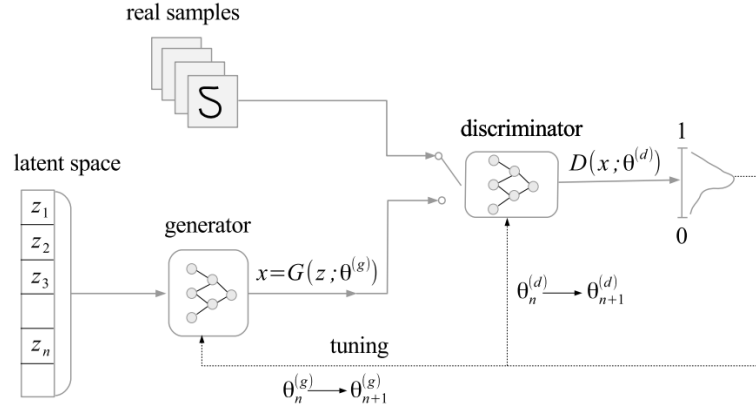
#### 5.6.4 Generative Adversarial Network

Until now, we have looked into regression and classification using neural networks. There is another important problem that can be solved by neural networks - generation.

Given a training sample, we want to train a neural network to generate more sample points of a similar kind to that of the training sample. To do this, Generative Adversarial Networks (GANs) were introduced.

A generator network  $G$  generates sample points using some random noise  $z$ . A discriminator network  $D$  is given a sample which contains sample points both from the generator network and the training data. It has to classify each of these as real or fake - it produces the probability  $D(x; \theta^{(d)}) \in [0, 1]$  that a sample  $x$  is real for some discriminator network parameters  $\theta^{(d)}$ .

Thus, the generator want  $D(G(z))$  to be close to 1 for all  $z$ , and the discriminator



**Figure 5.11:** The GAN architecture (Calin, 2020, pg. 599).

wants  $D(G(z))$  to be close to 0 for all  $z$ . This creates a minimax kind of situation, which motivates the loss function:

$$V(G, D) := \mathbb{E}_{x \sim p_{\text{data}}} [\ln D(x)] + \mathbb{E}_{x \sim p_{\text{model}}} [\ln(1 - D(x))] \quad (5.162)$$

Thus, the optimal network is

$$G^* := \operatorname{argmin}_G \max_D V(G, D) \quad (5.163)$$

*Remark 5.6.1.* Note that  $\operatorname{argmax}_D \min_G V(G, D)$  is incorrect, as the generator might just produce the points that are most likely in the training data, thus fooling the discriminator. This is called mode collapse - where the GAN produces the same output each time.

## *Chapter 6*

# CONCLUSION

In conclusion, we looked into the basic mathematical theory of statistical learning including generalization bounds ([Chapter 2](#)), SVMs and kernels ([Chapter 3](#)), boosting and dimensionality reduction ([Chapter 4](#)), and then studied neural networks - their definition, some practical hurdles, universal approximation and rates of approximation, and then ending with some popular variants of the standard neural network ([Chapter 5](#)).

This has been a very basic primer - due to the extremely high-paced nature of deep learning research, there is a lot of content that has not been covered. On the practical side, many new architectures like graph neural networks and diffusion models have been very promising. Many techniques like skip connections and dropout layers have shown great improvement of performance.

On the theoretical side as well there is a lot that was not covered - universality and rates of convergence for various architectures have been found. Apart from the purely analytical treatment, there has been lot of work involving geometry and algebraic topology to understand neural networks. Reinforcement learning has been another machine learning technique that has shown great performance and has been supported with ample theoretical results.

## Appendix A

# CONVEX OPTIMIZATION

Consider minimizing a function  $f : \mathbb{R}^p \rightarrow \mathbb{R}$  subject to some constraints  $g_i : \mathbb{R}^p \rightarrow \mathbb{R}$  for  $i \in [r]$  ( $p, r \in \mathbb{N}$ ). We shall call this the **primal optimization problem**:

$$\begin{aligned} \min_{x \in \mathbb{R}^p} \quad & f(x) \\ \text{subject to: } & g_i(x) \leq 0 \quad \forall i \in [r] \end{aligned}$$

Define now the **Lagrangian** for this problem as  $\mathcal{L} : \mathbb{R}^p \times \mathbb{R}^r \rightarrow \mathbb{R}$  by

$$\mathcal{L}(x, \alpha) := f(x) + \sum_{i=1}^r \alpha_i g_i(x)$$

for all  $(x, \alpha) \in \mathbb{R}^p \times \mathbb{R}^r$ . The **dual optimization problem** is now defined as:

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^r} \quad & \left( \min_{x \in \mathbb{R}^p} \mathcal{L}(x, \alpha) \right) \\ \text{subject to: } & \alpha \geq 0 \end{aligned}$$

Finally, we say that **strong duality** holds if the optimal values of  $f, \mathcal{L}$  are equal.

**Lemma A.0.1** (Slater's strong duality conditions). *Strong duality holds if*

1.  $f, g_1, \dots, g_r$  are convex.

2. *There exists  $x \in \mathbb{R}^p$  such that  $g_i(x) \leq 0$  for those  $g_i$ 's which are affine linear, and  $g_i(x) < 0$  for other  $g_i$ 's.*

**Theorem A.0.2** (Karush-Kuhn Tucker (KKT)). *Assume  $f, g_1, \dots, g_r$  are convex and differentiable and let strong duality hold. Then,  $x^*, \alpha^*$  are primal and dual solutions respectively iff KKT conditions hold:*

1. *(Stationarity)*

$$\nabla_x \mathcal{L}(x^*, \alpha^*) = 0$$

2. *(Complementary Slackness) For all  $i \in [r]$ ,*

$$\alpha_i^* g_i(x^*) = 0$$

3. *(Primal Feasibility) For all  $i \in [r]$ ,*

$$g_i(x^*) \leq 0$$

4. *(Dual Feasibility) For all  $i \in [r]$ ,*

$$\alpha_i \geq 0$$

*Remark A.0.1.* Under assumptions of KKT theorem,  $\mathcal{L}$  is convex and differentiable. Thus, the first condition says that  $\mathcal{L}(\cdot, \alpha^*)$  is minimum at  $x^*$ .



## Appendix B

# LINEAR ALGEBRA

**Theorem B.0.1** (Singular Value Decomposition). *Let  $A \in \mathbb{R}^{m \times n}$ . Then,  $A = U\Sigma V^T$ , where  $U, V$  are orthogonal matrices and  $\Sigma$  is the diagonal matrix of eigenvalues of  $A^T A$ .*

**Theorem B.0.2** (Rank Decomposition). *Let  $A \in \mathbb{R}^{m \times n}$  be a matrix of rank  $r$ . Then, there exist  $B \in \mathbb{R}^{m \times r}$  and  $C \in \mathbb{R}^{r \times n}$  such that  $A = BC$ .*

**Definition B.0.1** (Projection Matrix). A matrix  $P \in \mathbb{R}^{n \times n}$  is called the projection matrix on a (vector) subspace  $S$  of  $\mathbb{R}^n$  if

1.  $Pv \in S$  for all  $v \in \mathbb{R}^n$
2.  $Pw = w$  for all  $w \in S$ .

**Definition B.0.2** (Orthogonal Projection Matrix). A projection matrix  $P \in \mathbb{R}^{n \times n}$  on  $S$  will be called an orthogonal projection matrix if  $\mathbb{I}_n - P$  is a projection matrix on  $S^\perp$ , ie.  $\mathcal{C}(I_n - P) = S^\perp$ .

## Appendix C

### ANALYSIS

**Definition C.0.1** (Borel  $\sigma$ -algebra). The Borel  $\sigma$ -algebra  $\mathcal{B}(X)$  on a topological space  $(X, \tau)$  is defined as the  $\sigma$ -algebra generated by  $S$ , where  $S$  is the set of all open sets in  $\tau$ .

**Lemma C.0.1** (Urysohn). Let  $A, B$  be two disjoint closed subspaces of  $\mathbb{R}$ . Then, there exists a continuous function  $f : \mathbb{R} \rightarrow [0, 1]$  such that  $f(A) = 0$  and  $f(B) = 1$ .

**Lemma C.0.2.** If  $B \subset A$  and  $\mu(B) < \infty$ , then  $\mu(A \setminus B) = \mu(A) - \mu(B)$ .

**Lemma C.0.3** (Continuity from below). Let  $(X, \mathcal{M}, \mu)$  be a measure space. Let  $A_i \in \mathcal{M}$  such that  $A_1 \subset A_2 \subset A_3 \subset \dots$ , and  $\cup A_i = A$ . Then,  $\lim_{n \rightarrow \infty} \mu(A_n) = \mu(A)$ .

**Theorem C.0.4.** Let  $f : X \rightarrow [0, \infty)$  be a nonnegative measurable function. Then, there exists a sequence  $(s_n)$  of simple functions such that

1.  $\lim s_n(x) = f(x)$  for all  $x \in X$
2.  $0 \leq s_1(x) \leq s_2(x) \leq \dots \leq f(x)$  for all  $x \in X$ .

**Theorem C.0.5** (Bounded Convergence theorem). Let  $\{f_n\}_n$  be a bounded sequence of measurable functions on  $E$  with  $\mu(E) < \infty$ . If  $\lim_{n \rightarrow \infty} f_n$  exists, then

$$\lim_{n \rightarrow \infty} \int f_n d\mu = \int f d\mu \tag{C.1}$$

**Definition C.0.2** (Signed Measure). Let  $(X, \mathcal{M})$  be a measurable space. Let  $\nu : \mathcal{M} \rightarrow \bar{\mathbb{R}} = [-\infty, \infty]$  be a function such that

1.  $\nu(\emptyset) = 0$
2.  $\nu$  assumes at most one of the values  $\pm\infty$
3.  $\nu(\cup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} \nu(E_i)$ , where  $E_i \cap E_j = \emptyset$  for all  $i \neq j$ .

Then,  $\nu$  is called a signed measure and  $(X, \mathcal{M}, \nu)$  is a signed measure space.

**Definition C.0.3** (Regular measure). A positive Borel measure  $\mu$  on  $X$  is said to be regular if it is both outer regular and inner regular, ie.

$$\begin{aligned} \mu(E) &= \inf\{\mu(U) : E \subset U, E \text{ is open}\} \\ &= \sup\{\mu(K) : K \subset E, K \text{ is compact}\} \end{aligned}$$

for any Borel set  $E$ .

## Appendix D

# PROBABILITY

**Theorem D.0.1** (Markov's Inequality). *Let  $X$  be a nonnegative random variable with  $\mu = \mathbb{E}[X] < \infty$ . Then, for all  $t > 0$ ,*

$$\mathbb{P}[X \geq t\mu] \leq \frac{1}{t} \quad (\text{D.1})$$

**Theorem D.0.2** (Jensen's inequality). *Let  $X$  be a real-valued random variable and  $\varphi : \mathbb{R} \rightarrow \mathbb{R}$  be a convex function. Then,*

$$\varphi(\mathbb{E}X) \leq \mathbb{E}[\varphi(X)] \quad (\text{D.2})$$

*Remark D.0.1.* In particular, for  $\varphi(x) = x^2$ ,  $(\mathbb{E}[X])^2 \leq \mathbb{E}[X^2]$ .

**Definition D.0.1** (Convergence in probability). A sequence  $(X_n)$  of random variables is said to converge in probability to a random variable  $X$ , denoted by  $X_n \xrightarrow{p} X$  if, for all  $\varepsilon > 0$ ,

$$\lim_{n \rightarrow \infty} \mathbb{P}\{\omega \in \Omega : |X_n(\omega) - X(\omega)| > \varepsilon\} = 0$$

**Theorem D.0.3.** *Any probability measure over a compact subset of  $\mathbb{R}^n$  is regular.*

**Theorem D.0.4** (McDiarmid's Inequality). *Let  $X_1, \dots, X_m \in \mathcal{X}$  be  $m \in \mathbb{N}$  independent random variables. Assume there exist  $c_1, \dots, c_m > 0$  such that  $f : \mathcal{X}^m \rightarrow$*

$\mathbb{R}$  satisfies

$$|f(x_1, \dots, x_i, \dots, x_m) - f(x_1, \dots, x'_i, \dots, x_m)| \leq c_i \quad (\text{D.3})$$

for all  $i \in [m]$  and for all  $x_1, \dots, x_m, x'_i \in \mathcal{X}$ . Let  $f(S)$  denote  $f(X_1, \dots, X_m)$ .

Then, for all  $\varepsilon > 0$ ,

$$\mathbb{P}[f(S) - \mathbb{E}[f(S)] \geq \varepsilon] \leq \exp \left\{ \frac{-2\varepsilon^2}{\sum_{i=1}^m c_i^2} \right\} \quad (\text{D.4})$$

$$\mathbb{P}[f(S) - \mathbb{E}[f(S)] \leq -\varepsilon] \leq \exp \left\{ \frac{-2\varepsilon^2}{\sum_{i=1}^m c_i^2} \right\} \quad (\text{D.5})$$

**Theorem D.0.5** (Maximal Inequality). *Let  $X_1, \dots, X_n$  be  $n \in \mathbb{N}$  real-valued random variables such that for all  $j \in [n]$ ,  $X_j = \sum_{i=1}^m Y_{ij}$ , where for each fixed  $j \in [n]$ ,  $Y_{ij}$  are independent zero mean random variables taking values in  $[-r_i, r_i]$ , for some  $r_i > 0$ . Then,*

$$\mathbb{E} \left[ \max_{j \in [n]} X_j \right] \leq r \sqrt{2 \log(n)} \quad (\text{D.6})$$

where  $r = \sqrt{\sum_{i=1}^m r_i^2}$ .

## Appendix E

# FUNCTIONAL ANALYSIS

**Definition E.0.1** (Continuous Linear Transformation). Let  $(V, \|\cdot\|_V)$ ,  $(W, \|\cdot\|_W)$  be two NLS over  $\mathbb{R}$ . A continuous linear map  $T : V \rightarrow W$  is called a continuous linear transformation.

**Theorem E.0.1.** *Let  $T : (V, \|\cdot\|_V) \rightarrow (W, \|\cdot\|_W)$  be a linear map. Then,  $T$  is continuous iff there exists a  $c > 0$  such that  $\|T(x)\|_W \leq c\|x\|_V$  for all  $x \in V$ . Thus, a continuous linear transformation is also a bounded linear transformation, and vice-versa.*

**Definition E.0.2** (Operator Norm). Let  $T : (V, \|\cdot\|_V) \rightarrow (W, \|\cdot\|_W)$  be a continuous linear transformation. We define the operator norm of  $T$  to be

$$\|T\| := \sup_{\|x\|_V \leq 1} \|T(x)\|_W$$

**Theorem E.0.2** (Cauchy-Schwartz Inequality). *Let  $V$  be an inner product space and  $x, y \in V$ . Then,*

$$(\langle x, y \rangle)^2 \leq \langle x, x \rangle \langle y, y \rangle \tag{E.1}$$

**Theorem E.0.3** (Hahn-Banach extension). *Let  $V$  be a vector space over  $\mathbb{R}$ ,  $p : V \rightarrow \mathbb{R}$  be a sub-linear map, ie. for all  $x, y \in V$ ,  $\alpha \geq 0$ ,*

1.  $p(x + y) \leq p(x) + p(y)$

$$2. p(\alpha x) = \alpha p(x)$$

Let  $W$  be a subspace of  $V$  and  $g : W \rightarrow \mathbb{R}$  be a linear functional such that  $g \leq p$  on  $W$ . Then, there exists a linear functional  $f : V \rightarrow \mathbb{R}$  such that

$$1. f = g \text{ on } W$$

$$2. f \leq p \text{ on } V$$

**Theorem E.0.4** (Representation theorem for bounded linear functionals). *Let  $F$  be a bounded linear functional on  $C(K)$ , where  $K$  is a compact subset of  $\mathbb{R}^n$ . Then, there exists a unique finite, signed, Borel measure  $\mu$  on  $K$  such that*

$$F(f) = \int_K f(x) d\mu(x) \tag{E.2}$$

for all  $f \in C(K)$ . Moreover,  $\|F\| = |\mu|(K)$ .

**Definition E.0.3** (Hilbert Space). A Hilbert space  $H$  is an inner product space which is a complete NLS with respect to the norm defined as  $\|x\| = \langle x, x \rangle^{\frac{1}{2}}$  for all  $x \in H$ .

# BIBLIOGRAPHY

- Stone, C. J. (1982). Optimal global rates of convergence for nonparametric regression. *The annals of statistics*, 1040–1053.
- Bartlett, P., Freund, Y., Lee, W. S., & Schapire, R. E. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5), 1651–1686.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30, 5998–6008.
- Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning* (2nd). MIT Press.
- Calin, O. (2020). *Deep learning architectures : A mathematical approach*. Springer.
- Garrigos, G., & Gower, R. M. (2023). Handbook of convergence theorems for (stochastic) gradient methods. *arXiv preprint arXiv:2301.11235*.
- Li, H., Rakhlin, A., & Jadbabaie, A. (2023). Convergence of adam under relaxed assumptions. *36*, 52166–52196.
- Kohler, M. (2025). *On the rate of convergence of an over-parametrized deep neural network regression estimate learned by gradient descent* [arXiv preprint arXiv:2504.03405].