Analytical Variation in the generalization of deep feed-forward neural networks

Carlos Guatimosim Neves

Dissertação apresentada ao
Instituto de Matemática e Estatística da
Universidade de São Paulo para
obtenção do título de Mestre em Ciências

Programa: Matemática Aplicada

Orientador: Renato Vicente

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro do CNPq por meio do processo de número 132965/2017-8

São Paulo, Abril de 2021

Analytical Variation in the generalization of deep feed-forward neural networks

Esta é a versão original da dissertação elaborada pelo
candidato Carlos Guatimosim Neves, tal como
submetida à Comissão Julgadora:
Prof. Renato Vicente (Presidente), Prof Gustavo Rocha, Prof Nestor Caticha

Acknowledgements

First of all, I would like to express my most deep gratitude to my family, without which all this would be unfathomable. I feel truly blessed.

I would like to thank all my friends: Samuel, Lucas, José and Rodrigo, that I met along this journey. Countless hours were invested in insightful and joyful conversations by the coffee table. This partnership made the experience memorable and one in a lifetime. I am deeply move by all of it.

Also, I have much to thank my supervisor, Renato Vicente, for having the courage to accept so willingly someone with a theoretical background and no applied experience. Your guidance was timeless.

Finally, I would like to thank Taís, the light when all other lights had gone.

Outline

Guatimosim, C. N. **Analytical Variation in the generalization of deep feed-forward neural networks**. 2020. 96f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2020.

The essence of Machine Learning modelling may be summarized as: to unravel the implicit pattern in the data by having access to only a finite number of samples. The theory which studies this process is rich, and two quantities are of particular importance: the performance errors inside and outside the sample. The error inside the sample is called the training error, and is calculated over the set used to optimize the model's parameters. The outside error is the average error amongst all samples, and may be understood as the true error. Although the final objective is to construct a model with low true error, we only have access to the training error, which is an empirical estimation. Thus, in order to deduce the general pattern in the data, is necessary that both are similar.

The distance between those errors is called the generalization gap, and much of the theory is dedicated to study its properties and upper bounds, so that we may understand under what circumstances it is controlled. The gap is a measure of the model's ability to properly induce the global pattern, and is a major topic in all applied instances of Machine Learning.

The classical view of statistics correlates the generalization property with the model capacity to fit patterns in the data. The reasoning behind this is that, being capable of fitting many configurations, the model is prone to read noise in the sample, and thus perform poorly in general. However, the definition of a model's complexity is loose, and while it usually translates to the number of parameters, there is one hypothesis space which seemingly escapes this intuition. That is the case of Neural Networks.

Using Neural Networks with many layers (Deep learning) is proving to be the best modelling paradigm for many benchmark problems, and many of the advances in the industry are due to their success. However, this seemingly goes against what classical Statistical Learning theory states about generalization and complexity, since Deep networks are capable of fitting many patterns. Indeed, there has been experiments showing that networks may fit

even random labels.

This apparent paradox is an open question in the field and the main topic of this work. After a introduction and overview of the classical understanding of generalization, we introduce the work of [20], which is the central to our contributions. In it a new approach named Analytical Learning is proposed, aiming to complement the classical one, hopefully bringing some insights about the apparent contradiction emerged from Deep Learning.

Instead of analyzing probabilistic bounds, in this paper the generalization gap is studied in a context where the predictor and the dataset are fixed. By doing so, we prevent the pessimistic cases, and a tighter bound is hopefully achieved. Additionally, it provides a more real scenario, since in practice usually the data is given.

The main result of [20] bounds the gap involving a term related to the data and another related to the loss function Hardy-Kruase Variation. Our main contribution revolves around tracing similarities between this variation term with the stability concept studied in the classical approach of generalization, making parallels with what may be understood in the Analytical case as information.

The main idea is that the loss function variation decreases if the partial derivatives of the predictor, according to the instance space, are close to the oracle's. The derivatives in this sense may be understood as how much information the function is reading, since it measures the impact of a certain dimension in a local prediction. Thus, if the predictor reads information similarly to the oracle, then we guarantee a low gap. With this, we argue that the partial derivatives of the predictor are the main measure of regularization in the analytical sense. One of the advantages of this is simplicity: rewriting the SGD (Stochastic Gradient Descent) optimization step in the function space, we have an easy way to investigate the evolution of the model's complexity during training.

Furthermore, we use this interpretation to develop on relative recent papers trying to tackle the generalization paradox in deep learning, [28] and [37]. In the former we make an extensive analysis. while in the latter we make a more brief qualitative approach, showing how our interpretation relates with their result.

In [28], the complexity of networks is studied through the lens of Fourier Theory. There is shown that the space of ReLU (Rectifier Linear Unit) networks has a high spectral decay: during optimization, the increments in the k-th harmonic caused by the weights updates decreases with at least k^2. This means that high frequencies magnitudes in this space are naturally damped during training, suggesting an inherent regularization property. However, at no moment in [28] the generalization gap is mentioned, and so it is not clear if the spectral decay is enough to guarantee a good estimation of the true error. Motivated by this, we

show a bound using the Hardy-Krause Variation on splines which decreases with the degree, justifying the special properties of ReLU activation functions.

In [37] the main theorem shows that if the architecture of the network follows a funnel pattern (when the number of neurons in the network decreases as we go deeper), then increasing the number of layers actually reduces the generalization gap, thus supporting the deep learning approach. This happens because the funnel like architecture forces a non trivial kernel in the linear transformations, which translates into a loss of information. This implies that as the number of layers increases, the information shared between the final layer and the dataset decreases, making the prediction less data dependant and thus regularized.

This result relates closely to our interpretation of information in the analytical sense. Having a non trivial kernel means that in some cases the prediction is constant with respect to disruptions in certain dimensions. This means that the overall variation (in the sense of derivatives) will be smaller, which according to Analytical Learning translates to a smaller generalization gap.

Abstract

Guatimosim, C. N. **Analytical Variation in the generalization of deep feed-forward neural networks**. 2020. 96f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2020.

Since their overwhelming success in real applications, the overparametrization paradox in Deep Neural Networks has been an open problem in the field. It greatly motivated the study of bounds for the generalization gap, and after the results of [36] it became clear that the classical understanding was not enough and maybe a different approach was required. Here we use the Analytical Theory of learning developed in [20], where the Hardy-Krause Variation of the loss function plays the role of complexity to bound the gap. By making parallels with the stability and information concepts in the classical theory, we attempt to expand the work of [28], show how ReLU architectures provide a bound to the Hardy-Krause Variation.

Summary

*"For a long time, I did **not** go to bed early."*

0.1 List of Notation

Here we set the basic notation used in this work. It will not change unless explicitly stated.

n	Sample size
d	Dimensions
$\mathcal{X}$	Instance Space (Vectorial space from which the data is sampled)
$\mathcal{Y}$	Label Space
$\mathcal{Z}$	$\mathcal{Z} \triangleq \mathcal{X} \times \mathcal{Y}$
$\mathcal{H}$	The Hypothesis Space
S	The dataset already coupled with the labels ($S \subset \mathcal{Z}$)
μ	Distribution from which data is sampled from $\mathcal{Z}$
$L_\mu(h)$	The true error made by the hipothesis h according to the distribution μ
$L_S(h)$	Training error of the hypothesis h on the dataset S
$\ell(z)$	The loss function evaluated in $z \in \mathcal{Z}$
$\langle u, v \rangle$	The inner product between vectors u and v
$\mathbb{1}_A$	Indicator function of the set A
$\|\|x\|\|_2$	Euclidean distance
∇f	Gradient of f
$\ker(T)$	Kernel of the linear transformation T
$I(X, Y)$	Mutual information between the random variables X and Y
$\mu_1 \times \mu_2$	Product distribution induced my the distributionsμ_1 and $\mu 2$
$\mathbb{E}(X)$	Expected value
$z \sim D^m$	a sample of size m from the distribution D

Chapter 1

Introduction

The prevalence of machine learning in our society today is unquestionable. Its use, ranging from social media to security, credit allowance to physics, is overwhelming and its dominance, unrivaled. Although machine learning's date of birth is up for debate[1], it is a fact that its use has risen sharply since, propelled by the exponential growth of processing power and data storage capacity. Knowledge of machine learning is currently in high demand, even when it is not needed, and proficiency in its applications is the best career path in the United States according to Glassdoor's 2016-2019 report [1] and others. It seems inadequate to list detailed applications in order to illustrate the importance of machine learning in our lives, since assuming one's knowledge of it would be, not only reasonable, but also among the tenderest assumptions made in this work. Nevertheless, that to question the importance of machine learning's applicability today is to question modern technology itself.

Although many areas do not allow an explicit division between their applications and the theory which legitimates them, machine learning do so. Due to the development of high level programming languages and libraries, the use of machine learning is becoming not only increasingly popular, but also alienated from its foundations, making the development of real life applications possible without fluent theoretical knowledge. It would be absurd to consider an aeronautical engineer unfamiliar with the Navier-Stokes equation, but it is far from being an exception a machine learning practitioner who is not well versed in the Fundamental Theorem of Statistical Learning, Cybenko's Theorem or the Representer Theorem. Despite this apparent distinction, one would be mistaken to believe that formal machine learning theory did not keep up with the flourishing of its practical counterpart. Nurtured by a vast amount of questions raised in the already heated applied realm, theory is presenting itself incomplete, making research about its mathematical foundations not just evermore present in the academia but also increasingly so inside technology companies.

Among the questions that arose from the ubiquity of machine learning models, are of undeniable importance those regarding the overwhelming success of overparametrized neu-

[1] 1952 when Arthur Samuel first used the term, 1957 when Frank Rosenblatt designed the perceptron, etc

ral networks. Ever increasing data and processing power has allowed data scientists to fool around with such complex architectures, applying them in many areas, such as computer vision, speech recognition, speech translation and natural language processing, unexpectedly obtaining unparalleled performance [14]. However, a theoretical problem arises: the capacity of neural network's hypothesis class is tied with the architecture's complexity, implying that the overparametrized regime is prone to overfitting. This begs the question: how can such elaborate networks perform so well in practice, to the point of completely outclassing more parsimonious models such as logistic regression and SVM, when they seem to contradict the Occam's Razor principle? To answer this, a natural step would be to investigate the generalization capability of neural networks.

However, before delving inside more intimate properties of neural networks's generalization power, an essential theme which motivates our work, we shall first present a broad classical approach to such questions for completeness and to set the notation. Moreover, it would be mistaken to assume that former understanding about generalization cannot help to better grasp the apparently anomalous behaviour of deep networks, persuading us to step back towards the foundations.

1.1 The Foundations

Here, we set the basic framework of supervised learning, which shall be the one used in this work. Although a brief overview is given, knowledge of some fundamental concepts is assumed. We refer the interested reader to [32] for a more careful study. All notation used shall remain unaltered throughout the work unless a change is explicitly stated.

A general classification problem may be described as follows: consider an instance space $\mathcal{X}$ whose elements we wish to label. An element $x \in \mathcal{X}$ might be a person whose gender we wish to predict, or a photo which belongs to some previously specified category. In other words, we are looking for a function which receives x as an input and outputs its class, which in the case of gender could be indexed as 0 or 1.

Firstly, we need to define a loss metric that quantifies the error made by the predictor. Consider a measure space $(\mathcal{Z}, \mu)$, $\mathcal{Z} \triangleq \mathcal{X} \times \{0, 1\}$, where $\mathcal{Z}$ contains each instance together with all possible labels and is equipped with the joint distribution $\mu = p(x, y)$. The loss function would then be, given a hypothesis h, a real mapping ℓ defined in $\mathcal{Z}$, such as $\ell(z, h) = (h(x) - y)^2$. Naturally, we would like to find a predictor h that minimizes $L_\mu(h) \triangleq \mathbb{E}_\mu[\ell(z, h)] = \int_Z \ell(z, h) d\mu$, however, the measure μ is unknown[2]. As a matter of a fact, our only known link to the true labelling rule comes in the form of a dataset, sometimes called the training set, which is any finite subset S of $\mathcal{Z}^n$ sampled according to μ. The general idea would be to approximate the hypothesis which minimizes L_μ by finding one with good performance in the training set. Formally, defining the training error as $L_S(h) \triangleq \frac{1}{|S|} \sum_{z \in S} l(z, h)$, the obvious approach would be to design an algorithm $\mathcal{A}$ that finds a function which minimizes $L_S(h)$ and hope it performs similarly in the whole $\mathcal{X}$. Such class of algorithms are famous and commonly called empirical risk minimizers (ERM). Formally,

Definition 1.1.1. *Let $\bigcup_{i=1}^{\infty} \mathcal{Z}^i \triangleq \Lambda$ and $\Lambda^{\mathcal{H}}$ be the set of all mappings from Λ to $\mathcal{H}$. We say that an algorithm $\mathcal{A}'$ follows the ERM paradigm if it belongs to the set of all risk minimizers, that is,*

$$\mathcal{A}' \in \{\mathcal{A} \in \Lambda^{\mathcal{H}} ; \mathcal{A}(S) = \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h)\} .$$

So, if we are to adopt this paradigm, given a labelled dataset S and a loss function ℓ, we need to fix a hypothesis space $\mathcal{H}$ so that we can choose an optimizer to solve

$$\mathcal{A} : S \to \mathcal{H} , \qquad \mathcal{A}(S) = \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h) .$$

Having defined the main objects of the theory and notation, we can now turn to the heart of the problem which is the core of our work: supposing we adopt, for instance, the

[2]Had we had knowledge of μ the learning problem would be trivial, since the best predictor would be, in the least squares case, $h(x) = \max_{y \in \{0,1\}} p(y|x)$

ERM paradigm, under what circumstances can we guarantee that a function that does well in S will warrant us a good approximation of the labeling rule? In other words, when can we guarantee a low $\left|L_\mu(h) - L_S(h)\right|$? The latter mentioned quantity is known as the generalization gap, an essential notion in machine learning whose impacting agents we shall spend the rest of this section carefully discussing. Being as it may, it should be no surprise that many times $L_\mu(h)$ is much larger than $L_S(h)$ regardless of the algorithmic paradigm, a quite frequent phenomena called overfitting. Effectively, the very act of achieving a small $L_\mu(h)$ through a minimization of L_S is seen as the definition of learning itself.

Definition 1.1.2. *(Agnostic PAC Learnability)*[3] *A hypothesis class $\mathcal{H}$ is agnostic PAC (probably approximately correct) learnable with respect to a set $\mathcal{Z}$ and a loss function ℓ if there exists a function $m_{\mathcal{H}} : (0,1) \to \mathbb{N}$ and a learning algorithm with the following property: for every $\epsilon, \delta \in (0,1)$ and for every distribution μ over S, the algorithm returns $h \in \mathcal{H}$ such that, with probability of at least $1-\delta$,*

$$L_\mu(h) \leq \min_{h' \in \mathcal{H}} L_\mu(h') + \epsilon \, .$$

Not worrying much about the definition's technicalities, probably the first observation is that, regardless of the algorithm or the hypothesis space used, in order to learn we approximate the true expected value of a random variable according to an unknown measure μ with the empirical mean. This kind of approximation is usual and many results are known, one of which being the Hoeffding's inequality, which establishes an upper bound for the error made by approximating the true mean with the empirical expectation. Applying it in machine learning terms we obtain the famous result,

Theorem 1.1.1. [32] *Let S be a subset of $\mathcal{Z}$, $|S| = n$, sampled according to μ. Consider a predictor $h \in \mathcal{H}$, $h : \mathcal{X} \to \mathbb{R}$ and $\ell(z,h)$ a loss function. Then*

$$\mu^n \left(\left|L_\mu(h) - L_S(h)\right| \geq \epsilon \right) \leq 2e^{-2n\epsilon^2} \, . \tag{1.1.1}$$

This result is quite intuitive and relates closely to the Law of Large numbers: as we increase n, the training set S will become an increasingly faithful representation of $\mathcal{Z}$. We call attention that the measure is over S. This means that, as n increases, the datasets which would convey the wrong pattern and yield a large $\left|L_\mu(h) - L_S(h)\right|$ become exponentially unlikely according to μ for a fixed h.

We may then conclude that everything works well as long as we have enough samples. Unfortunately, things are often not so simple in the practice: even in the information era, gathering data is still expensive and we have no guarantee that the measure which produced the training set will be the same one to measure the true error, compelling us to look for other factors that might shorten the generalization gap. Additionally, if we restrict ourselves to the notion of PAC learnability, the above bound is not strong enough since it is not uniform

[3] In [32] other similar, yet weaker, definitions of learnability are presented.

to all hypothesis in $\mathcal{H}$, and thus (1.1.1) must be improved. Such refinement is trivially achieved for finite hypothesis spaces, where one can simply unite over all possible h using the union bound, and obtain

$$\mu^n \left(\left| L_\mu(h) - L_S(h) \right| \geq \epsilon \right) \leq \left| \mathcal{H} \right| 2e^{-2n\epsilon^2} \quad \forall h \in \mathcal{H} \, .$$

Indeed, this proves that any finite hypothesis space is PAC learnable. In the infinite case, however, a more elaborate theory is required. Concepts were developed in order to replace the cardinality of $\mathcal{H}$ for more informative quantities that can faithfully translate the true richness of the hypothesis space without redundancy. We are of course alluding to VC dimension and Rademacher complexity[4].

Definition 1.1.3. *The VC dimension of a function space $\mathcal{H}$, from here on denoted as VC($\mathcal{H}$), is the maximum size of a set $C \in \mathcal{X}$ that can be labelled with -1 or 1 in all possible ways by functions in $\mathcal{H}$.*

Definition 1.1.4. *Given a loss function ℓ, a dataset S of size n and a hypothesis space $\mathcal{H}$, we define the Radamacher complexity of $\mathcal{H}$ with respect to S and ℓ by*

$$R(\ell \circ \mathcal{H} \circ S) \triangleq \frac{1}{m} \mathop{\mathbb{E}}_{\sigma} \left[\sup_{h \in \mathcal{H}} \sum_{i=1}^{m} \sigma_i \ell(h, z_i) \right] .$$

where σ is uniformly distributed in $\{-1, 1\}$ and a_i is the i-th coordinate of a.

Loosely speaking, these concepts aim to gauge the representation capacity of a hypothesis class, capturing its true size, or in other words, its degrees of freedom. For the case of VC dimension, consider a given domain $\mathcal{X}$ and a hypothesis space $\mathcal{H}$ capable of reproducing any set of labeled points S, regardless of size. Then learn this class in the PAC sense would be akin to learn the set of all functions from $\mathcal{X}$ to $\{-1, 1\}$, since S could depict all labelling patterns. However, according to the No-Free Lunch Theorem, learn such class in the PAC sense is impossible. This means that the PAC learnability of a class — and as a consequence the ability to achieve a low gap — depends on its capacity to represent labels, which is precisely what VC dimension and Rademacher complexity measures. With hold of these stronger tools we may now state improved versions of inequality (1.1.1):

Theorem 1.1.2. *(VC generalization bound, [26]) Let $\mathcal{H}$ be a class of functions assuming values in $\{-1, 1\}$ and VC($\mathcal{H}$) $= d$. Then $\forall \delta \geq 0$, with probability at least $1 - \delta$ over the sample S of size n, the following holds.*

$$L_\mu(h) - L_S(h) \leq \sqrt{\frac{2d \log \frac{en}{d}}{n}} + \sqrt{\frac{\log \frac{1}{\delta}}{2n}} \qquad \forall h \in \mathcal{H} \, .$$

[4]Here we define VC dimension restricted to the classification context. For a broader scenario, see for example [25]

Theorem 1.1.3. *(Rademacher generalization bound, [32]) Let $\mathcal{H}$ be a class of functions, ℓ a loss function and S a dataset of size n. Then $\forall \delta \geq 0$, with probability at least $1-\delta$ over S, the following holds.*

$$L_\mu(h) - L_S(h) \leq 2R(\ell \circ \mathcal{H} \circ S) + c\sqrt{\frac{2\ln\frac{4}{\delta}}{n}} \qquad \forall h \in \mathcal{H}\ .$$

where c is some positive constant.

What we should take from the above bounds is the fundamental role the hypothesis space choice plays in the approximation of the true error. If $\mathcal{H}$ is not too complex in the VC or Rademacher sense, that is, if it does not have a large variety of functions, then their empirical loss is fairly close to the true loss for any dataset S. On the other hand, if either the VC dimension or the Rademacher complexity is infinite, then we cannot possibly expect to learn and we should look for simpler solutions[5].

Another way of viewing the influence of $\mathcal{H}$ in learning theory is by decomposing the distance between the true error of a fixed $h \in \mathcal{H}$ and the Bayes error[6] L^* in the following fashion:

$$L_\mu(h) - L^* = \underbrace{L_\mu(h) - L_\mu(h')}_{\epsilon_{est}} + \underbrace{L_\mu(h^{'}) - L^*}_{\epsilon_{app}}\,, \tag{1.1.2}$$

where $h \in \mathcal{H}$ is an arbitrary hypothesis which may be thought of as the output of the algorithm $\mathcal{A}$, and h' is the best performing hypothesis in the class: $h' = \text{argmin}_{h\in\mathcal{H}} L_\mu(h)$.

The first term is called the estimation error and it quantifies how far from the best achievable error in your class $L_\mu(h)$ is. Notice that this error depends on h, and thus, on the algorithm and the dataset. As the complexity of the class increases, ϵ_{est} will increase, since more functions with superior performance will be contemplated. One could think of this error as how far they are from the best possible hypothesis in the chosen $\mathcal{H}$.

The second term is called the approximation error and it quantifies how far from the theoretical smallest error the best performing hypothesis in your class is. This error depends only on $\mathcal{H}$ and it is a measure of how well your class as a whole represents the phenomenon to be learned. Naturally, as $\mathcal{H}$ is enriched, we obtain a smaller ϵ_{app}.

Seeking generalization, we clearly see a trade-off between performing well in S and approximating L_μ, which is analogous to the bias-variance trade-off in Statistics. For instance, Theorem 1.1.2 shows that if the VC dimension is low, we will achieve a small generalization

[5]In the PAC learning sense, indeed, a hypothesis class is learnable if and only if its VC dimension is finite, such is the Fundamental Theorem of Statiscal Learning [33].

[6]The Bayes error is the true error made by the best possible predictor, the Bayes predictor b^*. It always exists and depends solely on the distribution and on the loss function. For example, in the mean squared error case, $b^*(x) \triangleq \mathbb{E}[y|x]$, where the expected value is in respect to the posterior distribution.

gap. However, since we are restricting the representational power of the class, the training error will increase, which implies that both empirical and true error are high. In summary: we wish to minimize L_μ but only L_S is known, so we need to reduce it *and* make it close to the true error. However, these are counteracting processes. Since the training error is the main term in the loss function, usually the problem lies in reducing the generalization gap. Methods which tend to this matter are called regularization methods, and are widely used in practice. The essence of these techniques is to find ways to give up some training accuracy (such as using a simpler hypothesis space), in exchange of a better approximation of the true error.

So far, we restrained our discussion about the generalization to the choice of $\mathcal{H}$, but as we just saw this factor's influence is dependant of other elements, making it extremely restricted, suggesting that other ways to regularize must be considered, demanding thus a change of perspective. Indeed, looking back at (1.1.2) one might argue that is not only harmless but helpful to choose a rich $\mathcal{H}$ as long as one manages to output a function whose true error is close to $L_\mu(h^*)$. With this in mind, we now set aside the hypothesis space matter and turn our attention to the algorithm and the loss function.

Naturally simplifying the hypothesis space is one way to regularize, but most regularizations done in practice are not concerned with the hypothesis space itself, but with the choice of method to search it. In this work, we understand by regularization any technique or decision regarding the learning processes by which one intends to shorten the generalization gap, that is, to make L_S closer to L_μ.

Many such methods have been developed as heuristics to further improve the quality of the hypothesis found by $\mathcal{A}$. Let us now illustrate this with a few examples.

Formal techniques

- **Weight Decay:** If our loss function is only composed in terms of the training error, our algorithm will blindly fit the noise in S and output an overfitting hypothesis. One common way to get around this is to add a term that penalizes complex functions in the optimization[7], such as ones with parameters of high norm, biasing the algorithm's search in $\mathcal{H}$ towards simpler ones. Denoting by $w \in \mathcal{W}$ the parameters [8] of h, the algorithm would do
$$\underset{w \in \mathcal{W}}{\operatorname{argmin}} \; L_S(h_w) + ||w||_p^p \, .$$

- **Bagging:** In order to reduce overfitting, a common technique used in machine learning is ensemble learning, where one combines several models to make predictions. The most

[7] It should be noted that part of the regularization problem is to understand how to define complexity, and for each case this word might have a different technical meaning.

[8] For instance, the coefficients of a polynomial.

common way to generate such "council" of predictors is by using the Bagging method, native from statistics. The procedure is to uniformly sample from S, with replacement, m subsamples $S_i \subset S\,, i = 1, \ldots, m$, and then use each of those as a training set to generate m distinct models, defining the final predictor as their average. It has been demonstrated that bagging reduces the estimators variance by smoothing the predictions, providing great improvements, especially for discrete predictors such as decision trees [8]. Although both are regarded as ensemble learning techniques, we would like to stress that bagging and boosting differ in a central way: the former reduces the variance, while the latter is primarily used to reduce the bias[9].

Heuristics

- **Stochastic Gradient Descent (SGD):** SGD is a variant of the classic gradient descent optimizer. Instead of using the whole dataset to calculate the gradient and start the iteration[10] $w_{i+1} = w_i - \eta \nabla L_S(h_{w_i})$, it considers approximations of $L_S(h_{w_i})$ using not the entire dataset but randomly selected subsets called batches. The iteration would then be stochastic and in average equivalent to the classic gradient descent. Formally, the step iteration would be written as below.

$$w_{i+1} = w_i + \eta \sum_{z \in \beta_{i+1}} \nabla \ell(h_{w_i}, z) \tag{1.1.3}$$

  where β_{i+1} is a randomly selected subset of S of predetermined size used in the i-th step.

- **Early Stopping** : Early stop is a regularization technique that can be applied in any step oriented algorithm. Rather than running for as long as possible, one breaks the iteration after some kind of training error convergence, giving up a possibly lower $L_S(h)$ in order to obtain a tighter generalization gap.

Firstly, we point out that we are not discussing why the above mentioned improve the gap, since this matter lies beyond the boundaries of this work. However, before turning ourselves to the generalization problem in neural networks, it is worth noticing that these methods revolve around a modification in the algorithm that lowers its tendency to look for a hypothesis that only minimizes the training error. This change aims to prevent overfitting by lowering the predictor's dependency on S. In a way, we are handing over the unbiased characteristic of the algorithm, which would guarantee convergence to the true hypothesis in the limit $n \to \infty$, to grant its output a tighter generalization gap.

[9] For a more detailed investigation between their differences, we reference the reader to [8].
[10] η quantifies the step size towards the gradient and is referred to as the learning rate.

1.2 What about Neural Networks?

We spent the beginning of the last section introducing the reasoning behind regularization with regards to the complexity of the hypothesis space, such as VC dimension and Rademacher complexity. Although those fared relatively well to explain generalization in more traditional models, theory started to reveal limitations with the recent success of overparametrized neural networks, which have presented remarkable performance in various fields and are now heavily sought in many applications. Although these accomplishments (and shortcomings) are seen in various sorts of networks, in this dissertation we are restricting our discussion to deep feedforwards non linear ones, very commonly used in supervised problems. We spend the next few lines formally defining it.

Definition 1.2.1. *(Feedforward Neural Network) Let $\sigma : \mathbb{R} :\to \mathbb{R}$ be any non linear real function and $W \triangleq \{W_i\}_{i=0}^{L}$ a family of real valued matrices of sizes such that its composition $W_L \circ W_{L-1} \circ \cdots \circ W_0$ makes sense. Then, a neural network with activation function σ, parameters W and $L-1$ hidden layers is a function $f_W : \mathcal{X} \to \mathbb{R}^{d_L}$ of the form*

$$f(x) \;=\; W^{(L)} \circ \sigma \circ W^{(L-1)} \circ \sigma \circ \cdots \circ \sigma \circ W^{(1)}(x) \;,$$

where the compositions of σ are applied entrywise and d_l is the dimension of W^l's range.

Remarks:

1. The term "layer" comes from a common interpretation of neural nets as a L-partite graph. Each output vector $\sigma\big(W(l)(\cdot)\big)$ is seen as the l-th set (a layer), and each of its entry, a vertex, as is shown in Figure 1.1.

2. While in principle a feedforward neural network can have many distinct activation functions, and that is often the case in practice, in theoretical studies the choice of restricting it to only one type is often made.

At the risk of being redundant, we stress again that, according to statistical theory, overly rich hypothesis spaces are prone to overfitting. Despite their overwhelming success, deep feedforward networks are probably the best candidate for the most expressive space award, even more so when a more complex architecture is considered. Indeed, in 1989 George Cybenko demonstrated that, as the number of neurons tends to infinity, single-layered feedforward sigmoidal neural networks are capable of uniformly approximating any continuous function on a compact interval [10][11]. This became known as Cybenko's theorem, and after

[11] Two years later Kurt Hornik generalized the result for any activation function, showing that the approximation property is due to the architecture [18].

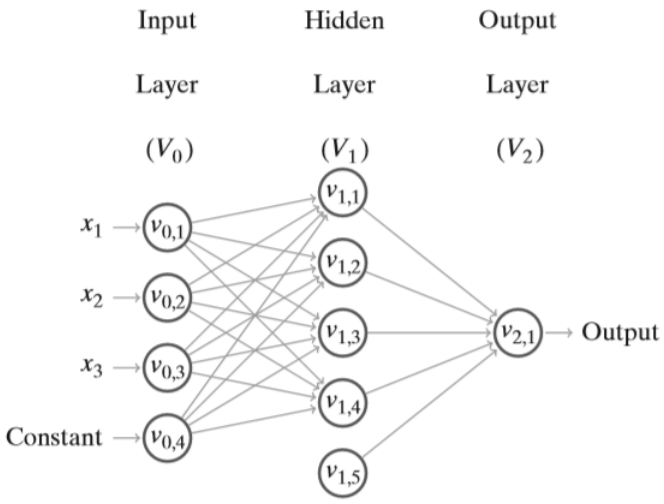


Figure 1.1: ***[Image taken from [32]]*** *In this example, $\mathcal{X} = \mathbb{R}^4$ and $L = 1$, where the dimensions of the inputs are denoted by x_i (x_4 is assumed to be a constant) and they enter in the network rewritten as $v_{0,i}$. Passing through the layer from left to right, they are transformed by W_1 whose entries are denoted by the edges, such that the dimension of the matrix establishes the width of the layers. For instance, we have that $v(1,j) = \sigma\big(\sum_{i=1}^{i} v(0,i) w_i^{(j)}\big)$, where $w^{(j)}$ is the j-th row vector of W_0. All layers that are between the sets containing the input and the output are called hidden layers. We say it has a feedforward behaviour because x flows in a linear fashion through the architecture, going from each independent part of the graph in one directed way.*

it many practitioners[12] stopped working with multi-layered networks since, according to the result, one layer was enough to guarantee asymptotic uniform approximation. In later years, however, deep neural networks started receiving great attention when they proved capable of outperforming the shallow architectures in many applications, going against the expectations.

Thinking in terms of VC theory, the VC dimension of neural networks is in the order of $\mathcal{O}(|E| \log(|E|))$ [25] where $|E|$ denotes the number of edges in the graph. Since this is monotonically increasing in the number of edges, it fails to explain the benefits of over-parametrization in generalization. In particular, it suggests that the bound in (1.1.2) is not tight enough for this case.

The attention to overparametrized networks grew even further with the questionings made in [36]. This paper takes benchmark architectures with number of parameters of order 10^6 and use them to learn the CIFAR10 dataset[13], a benchmark dataset composed of 60.000 images. It was already known that these would perform extremely well, but in [36] they also trained these networks on randomized labels using SGD with the same training hyperparameters, resulting in a complete memorization of the unstructured dataset by the learners, with basically no difference between each learning dynamic. This means that the

[12] Current Chief AI Scientist at Facebook Yann Lecun was an exception.

[13] Alexnet, number of parameters: 1,387,786. Inception, number of parameters: 1,649,402. MLP 3x512, number of parameters: 1,735,178. MLP 1x512, number of parameters: 1,209,866.

hypothesis class's capacity is high enough to memorize random labels of CIFAR10. Additionally, they made the same experiments (true and randomized labels), now using different kinds of regularizations[14], aiming to investigate their effect on the performance. The result is surprising, showing that none of them are really required for learning, pointing towards a possible implicit regularization in deep networks.

By showing how an extremely overparametrized $\mathcal{H}$ with capacity high enough to memorize random labels may be a good learner, the experiments made in [36] defy our understanding of generalization, showing how limited and incomplete is the *status quo*.

Naturally, having in mind what has been said in this section, it would be a misguided conclusion to deduce that parameters have no impact in the out of sample performance of networks, since part of the problem is that we do not know what the "true parameters" are. Instead, what one should take from this analysis is that we are unable to unwind the contribution of each pertinent component in generalization, and thus we must look for other concepts that can better capture the true quantities that explain the capacity of the network, and even more, understand how it relates with the algorithm, loss function and the training set.

Many other approaches came forth trying to better explain this problem, some more directed to the neural network model, some developing a much broader analysis such as the VC theory. In this work we first investigate recent studies, some more vigorously than others, dedicated to this endeavor, exploring the trailed trajectory and the framework constructed by the community to pursue the solution for this apparent paradox. Then we make use of recent results to ground our main contributions-[15], consisting in a new interpretation of hypothesis complexity which proves to be potentially fruitful to better understand neural networks.

[14]Data augmentation, weight decay and dropout.
[15]Better detailed in chapter 7

1.3 Chapter Overviews

Here we present how this work is organized, summarizing each chapter.

Chapter 2 : Aiming to ground the reasoning present in later chapters and provide the reader with a bit of context, a formal treatment over the relationship between stability and generalization is presented, contemplating precise definitions and rigorous results.

Chapter 3 : Here an overall study over the theory exposed in [20] is carried out, followed up by our contributions concerning it, making use of the main theorem in chapter 3. We adapt the stability's concept to the analytical learning framework using Hardy-Krause Variation. In the next chapters we use our insights to analyze two recent studies about generalization in deep networks.

Chapter 4 : With no proper machine learning content, a review of Reproducing Hilbert Space theory is made, culminating in a known result that allows us to write the gradient descent optimization in the function space.

Chapter 5 : In an attempt to use our interpretation in recent studies about generalization, we analyze the work of [28]. Using the analytical learning framework, we prove a Theorem relating their result with the generalization power of neuralnets.

Chapter 6 : In the same spirit of chapter 5, we again use the developed interpretation to parse through the theorems of [37]. We give a new qualitative analysis of the result, using it to further justify the pertinence of our approach developed in Chapter 5.

Chapter 7 : Here we finish our work, listing the main conclusions developed in the last chapters.

Chapter 2

Stability and Generalization

This chapter is dedicated to a brief review about rather recent concepts regarding generalization, some of which inspired by the previously mentioned paradox in deep networks. All of them intend to characterize generalization by presenting bounds for $L_\mu(h) - L_S(h)$ which involves concepts beyond the ones described in the first chapter. In [19] a very minute review over recent works on the subject of generalization is made, whose read is strongly recommended. Here, however, we shall take a more concise approach, caring for one specific notion which shall be of great use in this work: stability.

Hadamard was the first to define well-posedness, concept that brought to light the importance of stability in problems. A well-posed problem satisfies the following three conditions:

1. Admits solution.
2. The solution is unique.
3. The problem is stable, that is, the solutions are continuous on the initial data.

In [27] one finds a very informative discussion about this topic, considering a classical example: given a map $T : A \to B$ and $u \in B$, find $g \in A$ such that

$$u = Tg \, . \tag{2.0.1}$$

If we think T^{-1} as the learning algorithm, then T^{-1} is continuous if and only if small oscilations on its output may be achieved by small oscilations on the dataset. So, intuitively, continuity is related to the problem stableness, making the beforehand study of this more general approach aligned with our endeavors.

In principle, T can be as general as the problem allows, but for the sake of illustration lets consider for now the case where the mapping is a matrix.

If the determinant is not null, then the first two conditions are satisfied, and so the problem is well posed, and therefore stable, if and only if T^{-1} is continuous. This would

mean that

In this particular case this is always true since any linear transformation in a finite dimension space is bounded, so one only has to keep the determinant in mind. In a more general case, however, we must be mindful of continuity, which usually is the key criteria for well posedness[1].

Beyond its undeniable importance for certain applications, theoretically speaking, continuity implies in an important behaviour for solutions, which is often assumed in reality. Again, consider (2.0.1) now with $u_\delta \triangleq u + \delta$ being a perturbed initial data and its associated solution $g_\delta \in A$, which always exists supposing that T is a bijection. Under the only assumption that A and B are metric spaces, it is not always true that $||Tg_\delta - u||_B \leq \epsilon_1$ implies in $||g_\delta - g||_A \leq \epsilon_2$. In other words, by assuming only bijection and continuity of T, one yet does not have the guarantee that a solution for the perturbed problem is close to the true solution, even asymptotically.

Indeed, as is observed in [27], weight decay, also known in mathematics as Tikhonov regularization, was firstly created to solve this exact issue, and its origin lies in the following topological Theorem.

Theorem 2.0.1. *Consider an operator T mapping a compact set $A \subset \mathcal{A}$ onto $B \subset \mathcal{B}$, A and B being metric spaces. If T is continuous and one-to-one, then the inverse mapping T^{-1} is also continuous.*

The above theorem gives us a sufficient condition to mend the possible lack of stability: compactness of the solution space. Additionally, this result explicitates the objects which orchestrate the problem's stableness, namely, the solution space and the mapping between it and the initial data.

With some adaptations, one readily sees its relationship with weight decay: if we use some norm regularization in the optimizer, we are implicitly restricting its size, rendering it compact, and by the above theorem we would now have that $u_\delta \to u$ implies in $g_\delta \to g$. This similarity suggests that stability is, as is weight decay, sufficient for generalization. Indeed, we are dedicating this chapter on this matter because the latter is true. [7] has become a benchmark article on the topic and a brief rundown over its main results could not be more natural.

[1] In the case of the learning problem, is worth remembering that uniqueness of solutions is always meant under the equivalency class induced by the training error. That is, since the algorithm will pick at random some hypothesis with the same L_S, we are regarding them as all belonging to the same class

2.1 A formal approach to Stabilization in Learning

The concept of stability is prominent in many areas, and to the best of our knowledge it was first taken to the learning framework in the seventies, [11] and [12]. Albeit often related to sensitivity analysis, where one measures the changes in some response variable according to perturbations in the input, stability in learning communes with other elements which ought to be considered. In [7], general bounds for the generalization gap are found by studying the learning algorithm's stability with respect to the training set S, giving a positive answer to what is suggested in our analysis of problem (2.0.1).

Although we spent some lines arguing on how continuity is the basis of stability, usually one deals with discrete problems. Since the possible ways to define discrete continuity are many, the before well paved and defined path to understand ill posedness ramifies. Naturally, the learning map is no exception. By taking a countable set as input, conceiving some kind of stability notion that makes a prediction problem inherit the structure of the continuous case is a challenge that must be faced.

One way to tackle this issue is to look for definitions of stability which implies in a controlled generalization gap. Propelled by this , [7] proceeds as follows (the definitions and theorems below are as presented in [19]).

Let $S^{\setminus i} \triangleq \{z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n\}$ and $S^i \triangleq \{z_1, \ldots, z_{i-1}, z_i', z_{i+1}, \ldots, z_n\}$.

Definition 2.1.1. *An algorithm $\mathcal{A}$ has hypothesis stability β with respect to the loss function ℓ if the following holds,*

$$\forall i \in \{1, ..., n\},\ \mathbb{E}_{S,z}[|\ell(\mathcal{A}_S, z) - \ell(\mathcal{A}_{S^{\setminus i}}, z)|] \leq \beta\ ,$$

where $\mathcal{A}_X$ is the output of the learning algorithm A with X as the input.

Definition 2.1.2. *An algorithm $\mathcal{A}$ has point-wise hypothesis stability β with respect to the loss function ℓ if the following holds,*

$$\forall i \in \{1, ..., n\},\ \mathbb{E}_{S}[|\ell(\mathcal{A}_S, z_i) - \ell(\mathcal{A}_{S^{\setminus i}}, z_i)|] \leq \beta\ ,$$

While both of the above measure the change in predictions due to a single modification in the dataset, the former bounds the expected variation in the loss function, while the latter bounds the changes for any $z_i \in S$.

We call attention to the similarities between the above definitions and continuity. Since the input is a random variable, one could regard Definition 2.1.1 as some kind of continuity of the mean, where the deletion of any point in S entails changes in the loss function smaller

than β in expectation. In Definition 2.1.2 the expectation is only over the input S, and its measured a point-wise stability according to the deleted point.

Theorem 2.1.1. [7] *For any learning algorithm $\mathcal{A}$ with hypothesis stability β_1 and point-wise hypothesis stability β_2 with respect to a loss function ℓ such that $0 \leq \ell(f(x), y) \leq M$, we have, with probability $1 - \delta$,*

$$L_\mu(\mathcal{A}_S) \leq L_S(\mathcal{A}_S) + \sqrt{\frac{M^2 + 12Mn\beta_1}{2n\delta}} \text{ , and}$$

$$L_\mu(\mathcal{A}_S) \leq L_{loo}(\mathcal{A}_S) + \sqrt{\frac{M^2 + 6Mn\beta_2}{2n\delta}} ,$$

where $L_{\setminus i}(\mathcal{A}_S) \triangleq \frac{1}{n}\sum_{i=1}^{n} \ell(\mathcal{A}_{S^{\setminus i}}, s_i)$.

This theorem gives a positive answer to what was previously postulated: according to the defined notions, stability implies generalization. As a matter of fact, such relation was already known, and the main contribution of [7] is an even stronger definition which is found to be sufficient for a similar bound as the ones stated in the above theorem, but now with logarithmic components in δ.

Having in mind the sufficiency of stability for generalization, some remarks are at hand. Is intuitive that a stable algorithm contributes for generalization. After all, by producing a hypothesis whose prediction does not vary much with respect to perturbations in the training set S, is expected that the learning process which generated the hypothesis cared not for the specificity's of each learning instance, but rather, tried to grasp the general labelling rule implicit in the sample[2]. However, the interesting question remains: under which assumptions it is equivalent to generalization?

In the ERM case, is a classical result that consistency is equivalent to generalization, which in turn is equivalent to the hypothesis class being uniform Glivenko-Cantelli (for any measure) for uniformly bounded loss functions [33]. Here by consistency we mean universal weak consistency:

Definition 2.1.3. *A learning map is said to be weak universally consistent if, for any ϵ,*

$$\lim_{n\to\infty} \sup_{\mu} \left\{ P\Big(L_S \geq L_\mu + \epsilon\Big) \right\}.$$

The task of finding a similar relation for stability, that is, one which is sufficient for generalization and in turn equivalent to consistency in the ERM paradigm, is achieved in

[2]We insist on "expected" since, by simply ignoring the data, stability (and thus generalization) is trivially achieved despite nothing being learned, the constant predictor being a good example. This is to argue that one must be mindful of the trade off between good generalization and good performance, since a small gap is easily achieved if one does not tend to L_S.

[27]. There is defined the notion of *Cross-validation, error and empirical error Leave-One-Out stability* ($CVEEE_{loo}$ stability) and its demonstrated to yield the same implications, showing how fruitful this approach is for the learning problem.

2.2 Mutual Information and Stability

So far, having made this parallel between well posedness (stability), continuity and generalization, we point out that the learning map has been always considered deterministic. While it does cover a wild range of algorithms, there are a non contemplated few which ought to be attended. Stochastic gradient descent (SGD), being even more used than its deterministic counterpart, comes to mind.

To the best of our knowledge, [29] is the first to take these methods into account. There, stability is measured in terms mutual information shared between the algorithm and the dataset, and again we see a relationship between it and the generalization gap.
However, before going any further, we provide the reader with a brief rundown over the main concepts of Information Theory, which are imperative to understand results in [29], but will also present itself as an insightful contextualization for what follows.

2.2.1 Information Theory background

Here we recapitulate some fundamental notions. For a less abbreviated reading, we recommend [9].

The entropy of a discrete random variable X with alphabet $\mathcal{X}$ and probability mass $p(x)$ is defined as

$$H(X) \triangleq -\sum_{x\in\mathcal{X}} p(x)\log p(x) \ .$$

Entropy may be seen as the amount of uncertainty we have about X, which is completely characterized by $p(X)$. For instance, the entropy of the binomial distribution is maximized when $p = \frac{1}{2}$. If Y is another discrete random variable with alphabet $\mathcal{Y}$, then $H(X)-H(X|Y)$ would be the decrease of uncertainty about X when Y is realized, or in other words, the mutual information between X and Y:

$$I(X;Y) \triangleq H(X) - H(X|Y) = \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)log\frac{p(x,y)}{p(x)p(y)}.$$

Another quantity of great interest is the Kullback-Leibler divergence, which establishes a notion of distance between measures[3] and is defined as

$$D(\rho||\pi) = -\int_\Omega log\frac{d\pi}{d\rho}d\rho \ ,$$

[3]Formally, It is not a distance since it is not reflexive and does not satisfy the triangle inequality.

where Ω is a measurable space with respect to $d\rho$, $\pi << \rho$ and $\frac{d\pi}{d\rho}$ is the Radon–Nikodym derivative (the definition can be easily adapted to discrete random variables). If $\pi = p(X)p(Y)$ and $\rho = p(X,Y)$ then

$$D(p(X,Y)||p(X)(Y)) \triangleq -\int_{\mathcal{X}\times\mathcal{Y}} log\frac{p(x)p(y)}{p(x,y)}p(x,y)d(x,y) \ ,$$

which in turn is equal to the mutual information for continuous random variables, *i.e*:

$$D(p(X,Y)||p(X)(Y)) \triangleq I(X;Y) = H(X) - H(X,Y) \ .$$

Mutual information is always non negative, reflexive, and most notably, vanishes if and only if X and Y are independent.

2.2.2 Stability in Stochastic Algorithms

As it is quite common in the literature, stochastic algorithms may be regarded as a distribution $P(H|S)$ where $P(S,H) = P(S) \times P(H|S) = \mu^n \times P(H|S)$. That is, it will receive a dataset $S \subseteq Z^n$ sampled from μ^n as an input, and sample a hypothesis h according to the induced distribution $P(H|S)$ in the hypothesis class $\mathcal{H}$ parametrized by the weight space $\mathcal{W}$. Notice that H is now a random variable[4] with values in $\mathcal{H}$, whose distribution is characterized by the algorithm, defining a stochastic learning map. Thus, it makes sense to address the information shared by S and the output, or in other words, to how the prior's entropy $H(W)$ induced by the algorithm changes when it reads the dataset[5].

For example, one could think of stochastic gradient descent as the conditional probability in the parametric space induced by the dataset S. When the optimization begins, a distribution over all possible parameters $P(W)$, $W \in \mathcal{W}$ is established, and so each family of weights has its own probability of being chosen[6]. In the ERM paradigm for instance, parameters which correspond to functions of high training error would be unlikely to be selected. In this framework, the loss function is naturally defined as the average over all possible hypothesis for a given S:

$$\ell(P(H|S), z) \triangleq \mathop{\mathbb{E}}_{h\sim P(H|S)}[\ell(h,z)] \ .$$

This entails in the following definition for the gap[7]:

[4]Here we are incurring an abuse of notion, since entropy is also denoted by H. This should not pose a problem given the different context of each term.

[5]We call attention that, since only $P(W|S)$ is given, $P(W)$ and $H(W)$ are unknown.

[6]Here we make an abuse of notation, using interchangeably $P(W)$ and $P(H)$

[7]This is equivalent to redefining L_μ and L_S using $\ell(P(H|S))$

$$\mathop{\mathbb{E}}_{P(H|S)}[L_\mu(h) - L_S(h)] \ .$$

Naturally, mutual information encapsulates the idea of stability discussed in deterministic algorithms: loosely speaking, here a stable algorithm would be one whose distribution does not vary much in expectation over realizations of S. With this in mind, one might ask if this property is also enough for generalization.

Indeed, such relation may be established. One classical result in this direction is the PAC - Bayes bound given by McAllester in 1999:

Theorem 2.2.1 (PAC - Bayes Theorem, [32])**.** *Let μ be an arbitrary distribution over $\mathcal{Z}$ and $\ell : \mathcal{Z} \times \mathcal{W} \to [0,1]$ be a loss function. Let P be a prior distribution over $\mathcal{H}$ and $\delta \in (0,1)$. Then for all distributions Q over $\mathcal{H}$ we have that*

$$\mathop{\mathbb{E}}_{Q}[L_\mu(h)] - \mathop{\mathbb{E}}_{Q}[L_S(h)] \leq \sqrt{\frac{D(Q||P) + \ln(n/\delta)}{2(n-1)}}$$

holds with probability at least $1-\delta$ over the choice of the training set S.

Intuitively, one might think of the prior as the distribution defined by[8] $P(H) \triangleq \mathbb{E}_S[\, P(H|S)P(S)\,]$ and $Q \triangleq P(H|S)$. The idea behind the result is that, if the prior does not change too much with respect to S, then it will generalize with high probability. For instance, if the algorithm is independent of the data (maximum stability), that is, $P(H|S) = P(H)\ \forall S$, then $D(Q||P) = 0$ and the gap is minimized, although L_S will most likely be high.

The main theorem in [34] thus comes with no surprises. It shows that mutual information succeeds at expanding the concept of stability to stochastic learning maps, and further yet, the generalization sufficiency is preserved. We also mention these bounds on behalf of their informational interpretation, of which we shall make use to motivate our later work.

Theorem 2.2.2. [34] *Let H be a random variable representing the hypothesis and $\mathcal{H}$ the chosen hypothesis space. Suppose ℓ is σ-subgaussian loss function[9] under the generating distribution μ for all $h \in \mathcal{H}$. Then*

$$G(\mu, P(H|S)) \triangleq \mathbb{E}[L_\mu(H) - L_S(H)] \leq \sqrt{2(\sigma^2/n)I(S;H)} \ .$$

where the expectation is taken over the joint distribution $P(S,H)$.

Proof. A corollary from Theorem 8.1.1 proved in section 8.1. ■

[8] Notice that under this assumption $D(Q||P) = D(P(H,S)||P(H)P(S)) = I(S,H)$

[9] h is σ-subgaussian if $\log \mathbb{E}[e^{\lambda(h-\mathbb{E}[h])}] \leq \lambda^2\sigma^2/2$ for all λ in $\mathbb{R}$.

In words, the theorem states that small variations of H's entropy when some S is given implies in a better expected generalization. Essentially, if an algorithm $P(H|S)$ induces an *a priori* distribution which already has a good guess as to what the output might be, then, in expectation, it will generalize well. The difference between this result and the PAC - Bayes Theorem is mainly in the bounded object. The latter bounded with probability δ, where the former concerns the expected gap according to S.

Our motivation for presenting the last two results is to further clarify the reasoning which shall orchestrate our study, apart from being quite useful for setting the context and to pave the rationale of generalization theory. The latter, however, still lies behind a few prerequisites which we cared to expose thoroughly in the next chapter.

Chapter 3

Analytical Learning versus Statistical Learning

The success of deep neural networks and the usefulness of their overparametrization poses a challenge to the classical paradigm of statistical learning. As previously mentioned, [36] shows how this hypothesis space, albeit capable of learning random noise, might outperform other benchmark models in the overparametrized regime (number of parameters higher than $|S|$).

This however, only poses a contradiction when considering the stability of the learning map relative to changes in S, since performance is measured through the expected generalization error. So, if we consider S a given fixed variable, does learning in a high capacity hypothesis space emerges as an issue? How stability may be seen in this scenario? This chapter is dedicated to answering these questions.

In [21] discusses how complex hypothesis may solve regression problems for a given dateset, presenting a theorem showing that linear models admit complex hypothesis which have low training and generalization error:

Theorem 3.0.1. ([21]) *Let $\mathbb{R}^{d_y}$ be the label space, $\mathbb{R}^m$ the instance space, S a training set of size n and S_{test} a test set of size n_{test}. Consider a feasible linear regression problem, that is, given an oracle parametrized by $W^* \in \mathbb{R}^{d_y \times m}$ we wish to find $W \in \mathbb{R}^{d_y \times m}$ such that $||WX_{test} - Y_{test}||_2 \leq \epsilon$ by minimizing $||WX - Y||_2 \leq \epsilon$, where X and X_{test} are the training and test instances of S and S_{test} respectively, $W^*X_{test} = Y_{test} \in \mathbb{R}^{n_{test} \times d_y}$ and $W^*X_{test} = Y_{test} \in \mathbb{R}^{n_{test} \times d_y}$.*
Hence, if $n \leq m$, $rank(X) < m$ and $rank(M) < n$ where $M = [X^\intercal, X_{test}^\intercal]$, then there exists a matrix W such that

- $\hat{Y} = Y + \epsilon A$ *for some matrix A with* $||A||_F \leq 1$
- $\hat{Y}_{test} = Y_{test} + \epsilon B$ *for some matrix B with* $||B||_F \leq 1$
- $||W||_F \geq \delta$ *and* $||W - W^*||_F \geq \delta$

where $\hat{Y} \triangleq WX$ *and* $\hat{Y}_{test} \triangleq WX_{test}$

This result shows how there may be, for a given S, a high norm hypothesis which achieves a low generalization error with a low training error. Is also worth mentioning that the above result may be generalized to consider the expected instead of the test error.

By the classical learning paradigm, regularizations such as weight decay will ignore these potential solutions, but this means that there is some other condition of learning that is being disregarded. Indeed, the following simple result from [21] demonstrates the existence of an arbitrarily unstable algorithm that outputs a hypothesis which generalizes for a given S. This further argues that generalization gap bounds based on stability and robustness, while sufficient, are not necessary for generalization.

Theorem 3.0.2. *Consider a prediction problem defined by the pair* (μ, S) *specifying the true distribution* μ *and the training set* $|S| = n$*. Let* $\epsilon > \inf_{f \in \mathcal{Y}^{\mathcal{X}}} L_\mu(f) - L_S(f)$ *be the desired gap and* h_ϵ *be a function such that* $L_\mu(h_\epsilon) - L_S(h_\epsilon) < \epsilon$*, where* $\mathcal{Y}^{\mathcal{X}}$ *is the set of all functions from* $\mathcal{X}$ *to* $\mathcal{Y}$*. Defining* $h_{\mathcal{A}(S)} \triangleq \mathcal{A}(\mathcal{S})$ *as the output of the algorithm* $\mathcal{A}$ *when given* S*, we have that*

(i) for any hypothesis space $\mathcal{H}$ *whose VC dimension is at least* n *and contains the predictor* h_ϵ*, there exists a learning algorithm* $\mathcal{A}_\epsilon$ *such that the generalization gap of* $h_{\mathcal{A}_\epsilon(\mathcal{S})}$ *is at most* ϵ*, and*

(ii) there exist an arbitrarily unstable and arbitrarily non-robust algorithm $\mathcal{A}'$ *such that the generalization gap of* $h_{A(S)}$ *is at most* ϵ*.*

Proof. (i): Let $\mathcal{H}$ be any hypothesis class which contains h_ϵ. We simply choose $\mathcal{A}$ to be any algorithm from the class of all algorithms that outputs h_ϵ when S is given.
(ii): We define the algorithm $\mathcal{A}'$ as follows. If S is the input, then $\mathcal{A}'(S) = h_\epsilon$, and for any other S', $\mathcal{A}'(S') = h' \ \forall S' \neq S$ where h' is any arbitrary non robust and unstable hypothesis. ■

Naturally, in opposition to the probabilistic approach, theorem 3.0.1 and 3.0.2 argue about problems whose framework considers a fixed tuple (P, S), and so they do not contradict the bounds achieved in classical statistical learning, serving only to show that the current analysis of the generalization gap is incomplete, needing something other than stability, VC dimention and so on.
When a triple (S, f, μ) is fixed, $|L_\mu(f) - L_S(f)|$ is completely determined by its elements independently of other factors, such as the complexity of the hypothesis space, algorithmic robustness, stability etc. Investigate sufficient conditions to find hypothesis in this scheme is potentially advantageous not only because it is more realistic [1] but also hopefully it is

[1] Practically speaking, in many cases one works on a problem where the dataset and the distribution are already given.

better suited to explain the success of deep neural networks, since under overparametrization the hypothesis space complexity is very large and the statistical bounds might be too pessimistic.
The theory concerned with this kind of problem is Analytical Learning Theory[2], and to the best of our knowledge was firstly developed in [20]. While classical bounds consider all possible cases, like the ones based on VC dimension for neural networks, analytical bounds aim to analyze the generalization gap in a framework where each given dataset is individually treated, considering only gap bounds that exclusively depends on (S, f, μ), avoiding worst case scenarios.

In this regard, we ask ourselves the following: what properties that depend only on (S, f, μ) can we expect to be beneficial for learning, since stableness, robustness, VC dimension etc might be too pessimistic? Is generalization improved by overparametrization when a fixed problem instance is considered? The next section is dedicated to the study of [20], where a general analysis over the gap under the analytical framework is proposed, and we take it as a natural first step towards answering these questions, leaving [21], which is more oriented towards neural networks, with a strong reading recommendation[3]. We should also remember that, given how recent this approach is, there is not much done yet on the subject and one cannot be certain that it provides the answers we seek, but in [20] and [21] there is enough to convince anyone of its theoretical importance.

[2]Not to be confused with Analytical Learning algorithms, which accepts prior knowledge together with the data as input **??**.

[3]Even though this paper relates closely with our work since it studies specifically neural networks through the lens of analytical learning, its comprehension is not required to understand our main results, and so the choice was made to leave it as a complementary reading.

3.1 A first step

As a first step towards answering the aforementioned questions, we study the main result of [20], an upper bound for $L_\mu(f) - L_S(f)$ which only depends on the triple (S, f, μ).

We first would like to draw attention that such bounds are not common. For instance, the one given by the Hoeffding's inequality (1.1.1) depends on all possible datasets sampled by all possible distributions, yielding a bound which depends on many factors other than (S, f, μ). Another example would be the Radamacher complexity shown in Theorem 1.1.3: when the "expected ability" to fit unstructured data over the hypothesis space is considered, we take into account all possible cases, losing the capability to capture the properties that endows a fixed hypothesis a low generalization gap.

All this to argue how such deterministic bounds, from here on termed as instance dependant bounds, are specific, strong and not so obvious to achieve. To have a grasp on how restrictive this requirement is, notice that any that dependence on the size of the dataset, but not on the data itself, is not instance dependant, since it concerns any dataset of the same size.

In order to derive instance dependant bound presented in [20], we must first define two quantities: *discrepancy* and *variation in the sense of Hardy and Krause*[4]. Also, in order to adapt this new framework in Analytical Learning, there is a technical basis shift from probability and statistics to measure theory.

3.1.1 Discrepancy and Variation

Consider a measurable space $(\mathcal{Z} = [0,1]^d, \mu, \mathcal{B})$ where $\mathcal{B}$ is the Borel σ-algebra and μ some normalized Borel measure. Given a set $T_m = \{t^{(i)}\}_{i=1}^n \subseteq \mathcal{Z}$, suppose we would like to estimate how likely this event is in respect to μ. This raises the definition of discrepancy.

Definition 3.1.1. *Fixed $T_m = \{t^{(i)}\}_{i=1}^m$ and $t = (t_1, t_2, \ldots, t_d) \in [0,1]^d$, let $B_t = [0, t_1] \times [0, t_2] \times \cdots \times [0, t_d]$ be the closed box with a vertex on the origin. Then the local discrepancy of T_m and B_t is*

$$D(B_t, T_n, \mu) \triangleq \frac{1}{n} \sum_{t^{(i)} \in T_n} \mathbb{1}_{t^{(i)} \in B_t} - \mu(B_t) \,. \tag{3.1.1}$$

$D(B_t, T_n, \mu)$ serves to measure how consistent T_n is according to μ. For example, if $\mu(B_t) = \frac{1}{3}$, then a set whose a third of its points $t^{(i)}$ belongs to B_t represents μ, and so the

[4]These concepts, although relatively new to the machine learning field, have already been vastly used in numerical analysis [17].

discrepancy vanishes. Also, it is worth noting that, while one may view T_n as a sample for intuition's sake, we are treating it as a set in a measurable space with no need for a clear distribution. Figure 3.1 serves well to give a visualization about what $D(B_t, T_n, \mu)$ aims to measure.

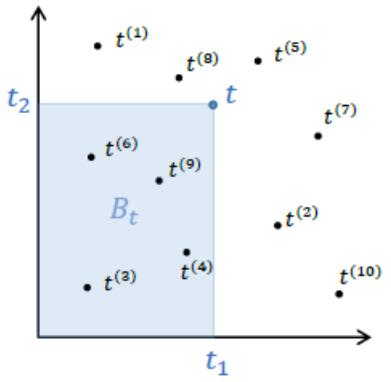


Figure 3.1: ***[Image taken from [20]]*** *Plots of T_n and B_t in $[0,1]^2$. If $t^{(i)}$ is coherent with μ, then the empirical probability is a good approximation for $\mu(B_t)$. Reciprocally, if $\mu(B_t) >> \frac{1}{n}\mathbb{1}_{t^{(i)}}$ then the points are out of the box even though the measure μ sees it as "large", suggesting that the set T_n does not represent it well.*

Since B_t merely compares sizes, we consider the the supremum of (3.1), that is, the scenario with lowest coherence (highest discrepancy) between T_n and $\mu(B_t)$. Thus the star discrepancy is defined as

$$D^*[T_n, \mu] \triangleq \sup_{t \in \mathcal{Z}} |D(B_t, T_n, \mu)|.$$

Now we turn ourselves to the Hardy-Krause variance.

We define a partition P of $\mathcal{Z}$ of size $m_1, m_2, \ldots, m_d$ as a set $\{t_j\}_{j=1}^d$ of non-decreasing finite sequences in $\mathcal{Z}$ where $t_j = t_j^{(0)}, t_j^{(1)}, \ldots, t_j^{(m_j)}$. In other words, P is a set of ordered partitions of each dimension of $[0,1]^d$, each with its own partition size. In our case we define such ordering $\leq$ in $\mathcal{Z}$ as

$$a \leq b \iff a_k \leq b_k \text{ for all } k = 1, \ldots, d\,. \tag{3.1.2}$$

Given points $a = (a_1, \ldots, a_d)$, $b = (b_1, \ldots, b_d)$ in $\mathcal{Z}$ and $\ell \subseteq \{1, \ldots, d\}$, we define $a^\ell : b^{-\ell}$ as being the point whose i-th entry is a_i if $i \in \ell$ and b_i otherwise. Then, given a real function g in $\mathcal{Z}$, the difference operator Δ is defined as

$$\Delta[g, a, b] \triangleq \sum_{\ell \subseteq \{1,\ldots,d\}} (-1)^{|\ell|} g(a^\ell : b^{-\ell})\,,$$

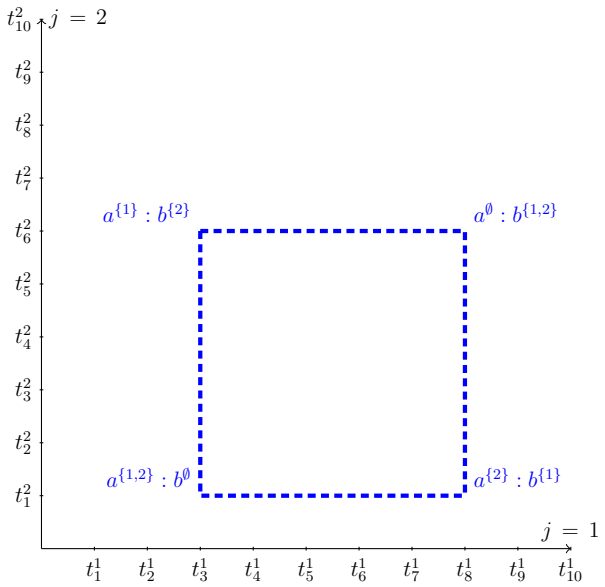


Figure 3.2: *We show above a partition of the positive plane. The parallel rectangle defined by* $a = (t^1_3, t^2_1)$ *and* $b = (t^1_8, t^2_6)$*, highlighting the vertices* $a^\ell : b^{-\ell}$*. Notice that, according to the defined measure, the entries of* a *must all be less than* b*, and so the rectangle defined by them has always non zero area.*

where $-\ell$ is the complement of ℓ.

In essence, $\Delta[g, a, b]$ is the alternating sum of evaluations of g in all vertices of the axis-parallel rectangle[5] $[a, b]$ defined by the ordering in 3.1.2. See figure **??** for more details.

Furthermore, we define the restriction of g to $u = \{j_1, \ldots, j_k\}$ evaluated at $t = (t_1, t_2, \ldots, t_d) \in [0, 1]^d$ as

$$g_u(t) \triangleq g(t')$$

where $t'_i = 1$ if $i \notin \{j_1, \cdots, j_k\}$ and $t'_i = t_i$ otherwise.

Finally, the variation of g in the sense of Vitali is

$$V^{(u)} g \triangleq \sup_{P \in \mathcal{P}_u} \sum_{t \in P} \left| \Delta[g_u, t, t_+] \right| , \tag{3.1.3}$$

where $\mathcal{P}_u$ is the set of all partitions in $[0, 1]^{|u|}$ and t_+ is the successor of t in P according to $\leq$.

In words, by the definition of the difference operator, $\left| \Delta^P[g_u, t, t_+] \right|$ is the sum of differences of g_u evaluated on all vertices of the rectangle defined by $[t, t_+]$. When the supremum is taken, we are considering infinitely finer partitions, and so the Vitali's variation would be similar to a sum of derivatives. Indeed, if g_u is continuous we have the somewhat intuitive

[5]Indeed, since $[a, b]$ is the set of all points x such that $a \leq x \leq b$, considering the defined norm, $[a, b]$ equals the interior and the borders of the rectangle.

result.

Theorem 3.1.1. *if g_u is a function for which $\partial_{j_1,\dots,j_k} g_{j_1,\dots,j_k}$ exists on $[0,1]^d$, then*

$$V^{(u)}g \leq \sup_{(t_{j_1},\dots,t_{j_k}) \in [0,1]^{|u|}} \left| \partial_{j_1,\dots,j_k} g_u(t) \right| , \tag{3.1.4}$$

if $\partial_{j_1,\dots,j_k} g_u$ is also continuous,

$$V^{(u)}g = \int_{[0,1]^d} \left| \partial_{j_1,\dots,j_k} g_u(t) \right| dt_{j_1}, \dots, dt_{j_k} .$$

The demonstration is omitted due to its solely technical nature. For a detailed proof see [20], proposition 1.

At last, the variation in the sense of Hardy-Krause is essentially the sum of Vitali's over all possible dimension restrictions of all possible sizes:

$$V[g] = \sum_{u \subseteq \{1,\dots,d\}} V^{(u)}g .$$

3.1.2 The generalization gap in Analytical Theory

As previously mentioned, the motivation to consider Analytical Learning Theory is to provide a deterministic upper bound to the generalization gap which is strongly instance-dependant. In the first section we gave an intuitive idea of the properties of such bound. Here we render this notion precise.

Definition 3.1.2. *A problem instance is any tuple $(\mu, S, \ell(f))$ whose elements specify some measure, a dataset and a loss function of a hypothesis. For simplicity we leave omitted the existence of a measurable space, although it is also specified by the measure.*

Definition 3.1.3. *Let ϕ be any object which depends on the tuple $(\mu, S, \ell(f))$. ϕ is called strongly instance-dependant with respect to $(\mu, S, \ell(f))$ if it is invariant under any change of any mathematical object that contains or depends on any $\hat{\mu} \neq \mu$, $\hat{S} \neq S$, $\hat{f} \neq f$.*

One example of a ϕ which is not strongly instance-dependant is the bound derived by Hoeffding's inequality. Since it varies according to n, then it would remain invariant for any $\hat{S} \neq S$ of the same size. Our intention is to ignore such bounds since, by translating the concept of representativity to n, it ignores the elements of the dataset, becoming possibly loose. Another example would be any bound that depends on the norm of the hypothesis's parameter, since one could have many hypothesis whose parameter's norm are the same.

The Hardy-Krause variation and the star discrepancy are common concepts in numerical analysis, and in the Hlawka-Koksma inequality ([17], [23]) they are used to upper bound

the estimation error made when approximating an integral with a finite sum. The idea of the main theorem in [20] is to adapt this inequality to the machine learning framework, hopefully arriving at a strongly instance-dependant quantity.

Recovering our usual notation, let $\mathcal{X} = [\mathbf{0}, \mathbf{1}]^d$ be the instance space and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y} = [\mathbf{0}, \mathbf{1}]^{d+1}$ the set of tuples $(x, y) = z$, that is, the instances coupled with the answer variable. In the analytical framework we work with measures, so we consider the input space for the loss function $(\mathcal{Z}, \mu, \mathcal{B})$ a measure space coupled with a normalized Borel measure μ defined in the Borel σ-algebra $\mathcal{B}$.

Theorem 3.1.2. ([20])[6] *Consider $f : \mathcal{X} \to \mathbb{R}$ a measurable function and $g : (\mathcal{Z}\,\mu, \mathcal{B}), \to \mathbb{R}$ a loss function which depends on f such that $V[g]$ is finite. Then, for any given dataset $S = \{z_1, \dots z_n\} \subset \mathcal{Z}^n$,*

$$\left|L_\mu(f) - L_S(f)\right| \leq V[g]\, D^*[\mu, S]\,, \tag{3.1.5}$$

where

$$L_\mu(f) \,-\, L_S(f) \,=\, \int_{\mathcal{Z}} g(z) d\mu \,-\, \frac{1}{n}\sum_{i=1}^{n} g(z_i)\,.$$

Remark: In general, the finiteness of $V[g]$ is a rather strong assumption which excludes many simple non continuous functions, such as the indicator of a ball, as shown in [16]. In this paper a new less restrictive concept of variation is defined, and a similar bound holds, albeit some algebraic properties are lost. This note is not made in [20], and since the usage of this theorem has not seen much applicability in the machine learning scenario yet, is hard to say if it is too restrictive to the point of needing a new idea of variation.

Proof. At first we present a lemma which will be necessary for the demonstration. Its proof is technical enough to redirect the reader to [20].

Lemma 3.1.1. *Any real valued function g in $\mathcal{Z}$ with $V[g] < \infty$ is measurable.*

Also, theorem 3 from [3] will be evoked to relate right-continuous functions with signed measures, but before that, a brief definition is called for.

Definition 3.1.4. *Consider a measure space $\mathcal{Z}$ equipped with a signed measure ν. Let*

$$\begin{aligned}\nu^+ &\triangleq \sup_{A \subseteq \mathcal{Z}} \{\nu(A)\} \\ \nu^- &\triangleq \sup_{A \subseteq \mathcal{Z}} \{-\nu(A)\}\,.\end{aligned}$$

[6] Most part of the demonstration is an adaptation of the one found in [3], where a simplified proof of the generalized Koksma–Hlawka inequality for non-uniform measures is presented.

Then the total variation of ν is defined as

$$|\nu| \triangleq \nu^+ + \nu^- .$$

Theorem 3.1.3. ([3]) *Let g be a right-continuous function on $\mathcal{Z}$ which has bounded Hardy-Krause variation. Then there exists a unique signed Borel measure ν on $[0,1]^{d+1}$ for which:*

$$g(z) = \nu([\mathbf{0}, z]) \quad \forall\, z \in [0,1]^{d+1} \tag{3.1.6}$$

and

$$|\nu|([\mathbf{0},\mathbf{1}]^{d+1}) = V[f] + |f(\mathbf{0})| . \tag{3.1.7}$$

The idea of the proof is to show equation (3.1.5) in the case where g is left continuous, and then expand the result for any loss function with finite variation. In order to use (3.1.6), let $\hat{g}(z) = g(\mathbf{1} - z) - g(\mathbf{1})$. Then $\hat{g}$ is right continuous and there exists $\nu_{\hat{g}}$ such that (3.1.6) and (3.1.7) holds. To have a similar measure for g instead, let us define ν_g as the reflected measure of $\nu_{\hat{g}}$, *i.e* , $\nu_g([a,b]) \triangleq \nu_{\hat{g}}([1-b, 1-a])$. Therefore,

$$g(z) - g(\mathbf{1}) = \hat{g}(\mathbf{1} - z) = \nu_{\hat{g}}([\mathbf{0}, \mathbf{1} - z]) = \nu_g([z, \mathbf{1}]) .$$

The fact that $|\nu_g|([\mathbf{0},\mathbf{1}]^{d+1}) = V[g]$ follows from equation 20 in [3].
With this correspondence between v_g and g at hand, we now rewrite $L_\mu(f) - L_S(f)$ exclusively in terms of ν_g. From the last equation,

$$g(z) = g(\mathbf{1}) + \nu_g([z,\mathbf{1}]) = g(\mathbf{1}) + \int_{\mathcal{Z}} \mathbb{1}_{[z,\mathbf{1}]}(t) d\nu_g(t) = g(\mathbf{1}) + \int_{\mathcal{Z}} \mathbb{1}_{[0,t]}(z) d\nu_g(t) .$$

This means that $g(z)$ may be rewritten as the size of the box $[z,\mathbf{1}]$ according to $d\nu_g$ times $V[g]$. Now, substituting the integral form of g in L_S and L_μ and using Fubini-Tonelli Theorem, we arrive at

$$\text{(a)}: \quad L_S(f) = \frac{1}{n}\sum_{i=1}^{n} g(z_i) = g(\mathbf{1}) + \frac{1}{n}\sum_{i=1}^{n}\int_{\mathcal{Z}} \mathbb{1}_{[0,t]}(z_i) d\nu_g(t) ;$$

$$\text{(b)}: \quad L_\mu(f) = \int_{\mathcal{Z}} g(z)\mu(z) = g(\mathbf{1}) + \int_{\mathcal{Z}}\int_{\mathcal{Z}} \mathbb{1}_{[0,t]}(z) d\mu(z) d\nu_g(t) = g(\mathbf{1}) + \int_{\mathcal{Z}} \mu([0,t]) d\nu_g(t) .$$

Thus, with the left continuity assumption of g, the generalization gap becomes

$$|L_\mu(f) - L_S(f)| \le \int_{\mathcal{Z}} \Big| \mu([0,t]) - \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{[0,t]}(z_i) \Big| d\nu_g(t) = \int_{\mathcal{Z}} D(B_t, S, \mu) d\nu_g(t) \le$$
$$|V_g| D^*[S,\mu] . \tag{3.1.8}$$

We have then demonstrated (3.1.5) for left continuous functions. The idea now is to

create a function g_m which will be left continuous and behave similarly to g, in a sense that shall be made precise. But before, some inequalities must be established.

By the the law of large numbers, for a given ϵ there exists $\hat{A}_m = \{\hat{z}_i\}_{i=1}^m$ such that

$$\left| \int_{\mathcal{Z}} g(z)d\mu - \frac{1}{n}\sum_{i=1}^{n} g(\hat{z}_i) \right| \leq \epsilon$$

Also, the Glivenko-Cantelli Theorem shows us that the empirical measure of a set B converges uniformly to $\mu(B)$, that is [7],

$$\sup_{B \in \mathcal{Z}} \left| \mu(B) - \frac{1}{m}\sum_{i=1}^{n} \mathbb{1}_B(\hat{z}_i) \right| = D^*[\mu, \hat{A}_m] \leq \epsilon \, .$$

Now we claim the following: for any g defined in $\mathcal{Z}$ with finite variation, there exists a left continuous function g_m such that $g_m(z) = g(z) \; \forall \, z \in S \cup \hat{A}_m$ and $V[g_m] \leq V[g]$. We shall proceed the demonstration by assuming the existence of such function and after we construct an example.

Summing and subtracting inside the generalization gap $\int_{\mathcal{Z}} g_m(z)\mu(z)$ and $\frac{1}{n}\sum_{i=1}^{n} g_m(\hat{z}_i)$ we obtain

$$\left| \int_{\mathcal{Z}} g(z)d\mu - \frac{1}{n}\sum_{i=1}^{n} g(z_i) \right| \leq \underbrace{\left| \int_{\mathcal{Z}} g_m(z)d\mu - \frac{1}{n}\sum_{i=1}^{n} g(z_i) \right|}_{(i)} +$$

$$\underbrace{\left| \frac{1}{m}\sum_{i=1}^{m} g_m(\hat{z}_i) - \int_{\mathcal{Z}} g_m(z)d\mu \right|}_{(ii)} + \underbrace{\left| \int_{\mathcal{Z}} g(z)d\mu - \frac{1}{m}\sum_{i=1}^{m} g_m(\hat{z}_i) \right|}_{(iii)} \, .$$

Since g_m is left continuous and $g(z_i) = g_m(z_i)$ for all z_i in S, we may use (3.1.8) and conclude that $(i) \leq V[g_m]D^*[T(S), T_\mu] \leq V[g]D^*[S, \mu]$. Moreover, because $g_m = g$ in $\hat{A}_m$, Glivenko-Cantelli theorem tells us that (ii) and (iii) are at most ϵ. Since ϵ can be made arbitrarily small by choosing an appropriate $\hat{A}_m$, we arrive at (3.1.5).

To show that this definition of g_m is not vacuous, we present an example of such function. Consider the grid G generated by $S \cup \hat{A}_m \cup \mathbf{0} \cup \mathbf{1}$, that is,

$$G \triangleq \{z \in \mathcal{Z} \; ; \; \forall k \in \{1, \ldots, d\} \; \exists a \in S \cup \hat{A}_m \cup \{\mathbf{0}\} \cup \{\mathbf{1}\} \; s.t \; z_k = a_k\} \, .$$

We define $g_m(z) \triangleq g(succ_G(z))$ where $succ_G(z)$ outputs the highest t in G such that $z \geq t$ according to the ordering defined in (3.1.2). First note that by construction g_m is left-

[7] Here we are suppressing the step where we take the biggest m such that $\hat{A}_m$ satisfies both inequalities

continuous coordinate wise. Additionally, $S \cup \hat{A}_m \subset G$ and $succ_G(z) = z \iff z \in G$, and so $g_m(z) = g(z)$ for $z \in S \cup \hat{A}_m$. As for variation, g_m is a piecewise constant function whose partitions are given by G, so $V[g_m] \leq V[g]$. ■

First and foremost, one must see that this adaptation of the Hlawka-Koksma bound for Machine Learning considers a problem instance $(\mu, \mathcal{Z}, g(f))$, and as such it is strongly instance-dependant. Indeed, the generalization gap bound is composed by two terms:

- $V[g]$: A measure of variance of the loss function for a given hypothesis f.
- $D^*(S, \mu)$: A term that encases all influences of S in the generalization gap. It gauges how representative the dataset is according to the measure and only depends on μ and S.

Note that (3.1.5) is invariant to any complexity measure of the hypothesis space or algorithm. Given a hypothesis, the bound tells us that all that matters is $V[f]$, regardless of the numbers of parameters, meaning that overparametrization could be valuable as long as it reduces the loss function's variance.

By ignoring other possible outcomes originated by the stochasticity of the sample or the algorithm, (3.1.5) gives a deterministic tight bound, giving a clearer understanding about what matters given a realization. For example, it sustains the concept of one-shot learning, since even in the case where $n = 1$, as long as the hypothesis has zero training error, and the loss functions, low variance, it will have a high performance.

More importantly for our case, it also translates the idea of stability previously presented in the stochastic case. Indeed, when $g = \ell(f) \in C^\infty$, $V^k[g_{j_1,\dots,j_n}]$ becomes the continuous sum of $|\partial_{j_1,\dots,j_k} g_{j_1,\dots,j_k}|$, that is, the sum over all points of variations of $g_{j_1,\dots,j_k}$ according to local changes in $z_{j_1}, \dots, z_{j_k}$, meaning that $V[g]$ gauges the overall prediction's stability according to the inputs. For instance, if g is unstable in a certain open set with respect to disruptions of a certain combination of variables, then it will contribute greatly to Hardy-Krause Variation. In the eyes of classical statistical learning this notion is applied in the context of algorithms, and the disruptions considered are usually in the sample or parameters, as noted in chapter 2.

All this to conclude that in Analytical Learning Theory, being a study of the learning process after S and f are given, the generalization bound depicts the classical regularization factor of stability in another form, obliging us to accordingly change its interpretation. In the next section we push further rational and attempt to link it with information compression.

3.2 A formal analysis of Compression

Unlike in [20], where an algorithmic analysis is developed, here we are mostly interested in the theoretical idea of stability suggested by $V[g]$ and how it relates with mutual information.
As seen in chapter 2, definitions which try to encapsulate this concept are many, but in general, an algorithm is considered stable if similar datasets yield similar hypothesis. In Analytical Learning Theory this concept's importance is somewhat maintained, but in a different context. When a problem instance is fixed, (3.1.5) decouples the factors that depend on the dataset and the hypothesis, evidenciating the Hardy-Krause Variation as the property which characterizes the stability of a predictor, but now relative to changes in the instance space and not S. So this notion remains of great importance in the Analytical Learning perspective just as it is in the classical one, preserved in the form of the loss function's Hardy-Krause variation.

However, before further discussions on information[8], an issue lingers: (3.1.5) establishes a relationship between the generalization gap and the Hardy-Krause variation of the loss function, omitting the hypothesis f. So far we have always referred to the stability of g, but in Machine Learning we wish to know the properties of the predictor which yields a low generalization error, and thus remains to be exposed some kind of similarity between $V[g]$ and $V[f]$. We elaborate on that herein.

We assume a teacher-student scenario: let $\mathcal{X}$ be a vectorial space with dimension d and suppose that the labels are obtained by the rule f^*, where f aims to approximate it according to the loss $g(x) = (f(x) - f^*(x))^2$. We first start with a naive approach, using the partial derivatives to extrapolate properties for the variance.

$$\frac{\partial g(x)}{\partial x_i} = 2(f(x) - f^*(x))(f_{x_i}(x) - f^*_{x_i}(x)) \, .$$

Intuitively, we would like conclude that its is sufficient to control the partial derivatives of $f(x) - f^*(x)$ so that the variation of $V[g]$ is small, or in other words, it is sufficient that the partial derivatives of f are close to the ones of f^* to achieve a low $V[g]$. In the context of stability, this suggests that the predictor should vary according to the oracle, thus maintaining the pertinence of each variable in the prediction of x, hopefully reducing $V[g]$.

Indeed, rewriting the partial derivatives as

$$\partial_{j_1,\dots,j_k} g \triangleq \partial_u g$$

[8]It should be stressed that, since there is no stochastic phenomena being considered in analytical learning, the word "information" becomes imprecise. We use it loosely in order to guide our informational interpretation of $V[g]$, appealing to an intuitive understanding.

where $u = \{\partial_{j_1,\dots,j_k}\} \subseteq \{1,\dots,d\}$, we may write a general partial derivative of g as

$$\partial_u g \;=\; 2\sum_{j\subseteq u}\big(f_j - f_j^*\big)\big(f_{-j} - f_{-j}^*\big)\;,$$

and thus

$$|\,\partial_u g\,| \;\le\; 2\sum_{j\subseteq u}\big|\big(f_j - f_j^*\big)\big(f_{-j} - f_{-j}^*\big)\big|\;. \tag{3.2.1}$$

where $-j$ denotes the complement of j.
Thus, we find that in order to decrease $|\partial_u g(x)|$, and consequently $V[g]$, it is enough to make $\partial_u f(x)$ close to $\partial_u f^*(x)$ for all $k \le d$. Formally, assuming that f and f^* are such that $g \in C^\infty$ and using equation (3.1.1) from Theorem 5.2,

$$V^{(u)}g_u \;=\; \int_{[0,1]^k}\big|\partial_u g_u(t_{j_1},\dots,t_{j_k})\big|dt_{j_1},\dots,dt_{j_k} \;=$$
$$\int_{[0,1]^k}\eta(t_{j_1},\dots,t_{j_k})dt_{j_1},\dots,dt_{j_k}$$

where $\eta(t_{j_1},\dots,t_{j_k})$ is a positive function that converges point-wise to zero as $\partial_u f_u(t_{j_1},\dots,t_{j_k})$ tends to $\partial_u f_u^*(t_{j_1},\dots,t_{j_k})$.

Since $V[g]$ is defined as the sum of all $V^{(k)}g_u$ over all possible combinations of dimensions, the above equality gives us a criteria that is sufficient to control $V[g]$ by simply looking at the partial derivatives of the predictor and of the oracle[9], enabling us to derive a direct and easy to calculate quantity that will conduct the learning process. More concretely, if $\partial_u f_u(t_{j_1},\dots,t_{j_k})$ is close to $\partial_u f_u^*(t_{j_1},\dots,t_{j_k})$ in an open set for all possible restrictions u, then we can guarantee a reduction of $V[g]$ and therefore of the generalization gap. This enriches our interpretation, allowing us to equate the variation of the loss function with the variance disparity of f relative to f^*.

Its worth stressing that if the variation of both f and f^* are close, then $V[g] \le \epsilon$. This because reduce $\big|V^u[f] - V^u[f^*]\big|$ would require a similarity between the derivatives norm, which is not sufficient to control (3.2.1). Indeed,

$$\big|V^u[f] - V^u[f^*]\big| \;\le\; \int_{[0,1]^k}\Big|\,\big|\partial_u f_u(t_{j_1},\dots,t_{j_k})\big| - \big|\partial_u f_u^*(t_{j_1},\dots,t_{j_k})\big|\,\Big|dt_{j_1},\dots,dt_{j_k}\,.$$

This resonates with what we said before: to reduce $V[g]$, f must create dependency with the inputs, not only in the same intensity, but with the same sign.

[9]There maybe are other ways to reduce the loss function's Hardy-Krause variance, but in the machine learning framework, approximating the oracle's derivatives is the most reasonable approach.

Having elaborated on the relationship between $V[f]$ and $V[g]$, we now bring information to the matter. In chapter 2 we explain how mutual information may be seen as an alternative concept of stability, being able to bound the expected generalization gap. Parting ways from [20], here we argue how compression, seen in the lens of information theory, relates to $V[g]$.

In mutual information, we sum variations of the entropy as we change the outcomes of the input variable, and so, if our uncertainty about Y changes as we see realizations of X, then both share information. Similarly in the deterministic case, fixing a hypothesis f, $x = (x_1, \ldots, x_n)$, $\frac{\partial f(x)}{\partial x_i}$ could be seen as a measure of how locally dependant $f(x)$ is of the i-th entry of x: if the function's slope peaks according to x_i on x, then it means that $f(x)$ is highly susceptible to changes in the i-th dimension, suggesting us to interpret $\frac{\partial f(x)}{\partial_{x_i}}$ as a measure of pertinence of x_i for the prediction. From another perspective, given a set of points to interpolate, to choose a function of low $V[f]$ is to choose a function more "oblivious" and invariant relative to the inputs [10]. On the other hand, a function of high oscillation would be one that reads strongly the information in the inputs, making it more dependant and unstable.

Motivated by how the Hardy-Krause variation contributes to learning and relates to stability, we propose an interpretation that sees it as a quantifier of information shared between the inputs and the prediction, measuring the local participation of each dimension and their entanglements.

The translation of the relationship between information and stability from the classical context to the analytical one is rather unique. As far as the statistical approach goes, algorithmic stability is always welcome when it comes to the reduction of the generalization gap. However, while the reduction of $V[g]$ always contributes, equation (3.1.5) tells us that the benefits brought by the hypothesis's variation (in the analytical sense) only goes so far. The partial derivatives of f must be close to the oracle's: loosely speaking, if it is much greater, f is reading more information than it should, yielding in a higher $V[g]$. Conversely, if partial derivatives are close to f^*, then it is reading important information and ignoring what f^* ignores, implying in a tighter gap.

This makes sense when one thinks about the original application of the Koksma–Hlawka inequality, integral estimations. Regardless of the method used to discretely estimate the area, a curve of high oscillations will usually make the approximation poorer. The same reasoning may be applied to Machine Learning when L_S is used to approximate L_μ. From this we draw an important conclusion: when it comes to the reduction of the gap, is in the loss function where stability is truly important, and in order to achieve it, information

[10] In the limit case, the constant predictor, having a null Hardy-Krause variation, would read no "information" from the inputs.

must be correctly compressed, that is, f must be stable only where f^* also is. This means that $V[g]$ is a measure of the hypothesis's compression in relation to f^*, which is in turn, a measure of their derivative's distance.

This chapter was meant to introduce Analytical Learning Theory and our own interpretation of variation as a deterministic information. Having all this in mind, we see these benefits:

(i) By making parallel with the classic statistical concepts, it brings a clearer understanding of the Koksma-Halwaka Inequality in the Machine Learning context, contributing to Analytical Learning Theory.

(ii) Being based on partial derivatives, it is easy to manipulate. In particular, it provides a simple way to analyze information compression during training, unwinding the relation between training error optimization and stability.

(iii) Together with (3.1.5), we believe that it may also shed light on the overparametrization paradox in deep networks.

Our goal now is to elaborate on the listed items using the tools here described. On the next chapter we will use Kernel Theory to detail and formalize (ii), and the last two chapters will be dedicated to develop (iii) by analyzing two papers through the lens of Analytical Learning Theory.

Chapter 4

Reproducing Kernel Hilbert Spaces and Optimization

In a fixed problem instance context, compression can be understood in terms of decay of the derivatives. For example, if $\frac{\partial f(x)}{\partial x_i} = 0$ then f is using no information from the i-th dimension when labeling x, that is, it learned that x_i is not a relevant feature for that prediction, and as such its variations should not change it. Equation (3.1.5) shows that if f compression coincides with the oracle's, then the gap will be small. However, it remains to be shown how information reading evolves during learning, and how it may relate to the training error.

Following this reasoning, in this chapter we ask the question: when one trains a hypothesis using a gradient descent method, is there some kind of pattern governing compression and accumulation of information? In order to answer this question, we need to monitor the evolution of the partial derivatives during optimization.

In supervised learning, once a dataset S and a hypothesis space $\mathcal{H}$ parametrized by a weight space $\mathcal{W}$ is given, an optimization algorithm of the gradient descent class is usually chosen. The model is parametrizable, and the computation takes place in the weight space $\mathcal{W}$, as it is represented in (1.1.3). However, while the equivalence between $\mathcal{W}$ and $\mathcal{H}$ is of fundamental importance in computations, especially when designing the optimization algorithm, this approach is often unfriendly to other approaches in studying learning. For instance, the parametric space representation of functions clouds the generalization analysis, hindering any study of hypothesis complexity that considers objects beyond the weight magnitudes. Since an akin approach is taken in this work, we ignore the parametric form unless explicitly stated.

By taking this decision, albeit granting flexibility, the absence of parametrization disallows the use of the classical gradient descent formula. Aiming to restore something analogous to our resolution before any other further progress, the main objective of this chapter is to retrieve equation (1.1.3) in its non parametric form, now as an optimization in the function

space:

$$f_{t+1}^S \triangleq f_t^S - \frac{2}{n}\sum_{i=1}^{n}(f_t(x_i) - y_i)K_{x_i} \ , \tag{4.0.1}$$

where K_{x_i} is the kernel function with one of its entries evaluated at x_i.

The theory behind this result is that of Reproducing Kernel Hilbert Spaces (RKHS), and the preliminaries shall be thoroughly studied in section 4.1. In section 4.2 we restrict ourselves to the regression problem using square differences as a loss function, following the reasoning in [35][1], using the results gathered so far to demonstrate that gradient descent in the RKHS may be described as in (4.0.1).

[1] In [35]. The formula of the gradient descent in the RKHS is derived in proposition 2.2, but the essential part of the demonstration is omitted.

4.1 A brief overview of Reproducing Kernel Hilbert Spaces

Firstly a brief introduction of Reproducing kernels Hilbert Spaces is provided. We then use it to arrive at a formula for the gradient descent in the function space, which will be the basis of our later analysis. In this section we describe RKHS theory oblivious of any aspect of learning theory, so the reader may fathom its generality.

Kernel is one of those words whose understanding is obscured by its almost ominous presence in a wild range of fields, always assuming different meanings. In the context of RKHS, the definition of Kernel is any real[2] function of two variables $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. This notion has been frequently used in machine learning, being the theoretical basis for the kernel trick and the Representer Theorem, demonstrated by Schölkopf, Herbrich, and Smola in 2001 [31].

Consider a Hilbert space $\mathcal{H}$ of real valued functions defined in $\mathcal{X}$, a domain with, in principle, no structure. We may then ask ourselves whether or not the evaluation functional, defined as $e_x(f) = f(x)$, is continuous. If it is, then it belongs to the dual $\mathcal{H}^*$, and so we are allowed to use the Riesz Representation Theorem [24], giving us the so called *reproducing property*:

$$\forall e_x \in \mathcal{H}^*,\ \exists K_x \in \mathcal{H} \quad \text{such that} \quad e_x(f) = \langle f, K_x \rangle_{\mathcal{H}} = f(x)\ . \tag{4.1.1}$$

When $\mathcal{H}$ is such that (4.1.1) holds, we call it a Reproducing Kernel Hilbert Space. This is because it naturally generates a Mercer kernel, that is, a semi positive-definite, continuous and symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined as

$$K(x, y) \triangleq e_y(K_x) = \langle K_x, K_y \rangle_H = e_x(K_y)\ .$$

When (4.1.1) holds, not only $\mathcal{H}$ produces the above kernel, but it is also generated by the same one up to a isomorfism, such is he Moore-Aronszayn Theorem [4].

Theorem 4.1.1. *(Moore-Aronszayin [4]) Let K be a real Mercer Kernel (positive semi-definite, symmetric and continuous) function on $\mathcal{X} \times \mathcal{X}$. Then there exists only one Hilbert space $\mathcal{H}_K$ of functions on $\mathcal{X}$ with K as the reproducing kernel, which is given by $\mathcal{H}_K \triangleq \overline{span\{K_x / x \in \mathcal{X}\}}$ where $K_x(\cdot) \triangleq K(x, \cdot)$. Moreover, for $f = \sum_{i=1}^{n} \alpha_i K_{x_i}$ and $g = \sum_{j=1}^{m} \beta_j K_{x_j}$, the inner product of $\mathcal{H}$ is defined as*

$$\langle f, g \rangle_K \triangleq \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j K(x_i, x_j)\ .$$

[2]Formally, a kernel might also take complex values, but since we aim for applications in machine learning, this is a case which we happily abdicate.

For f and g limits of Cauchy sequences $f_n, g_n \in \mathcal{H}_K$, the inner product becomes

$$\langle f, g \rangle_K \triangleq \lim_{n\to\infty} \langle f_n, g_n \rangle_K .$$

The proof revolves around the formalities needed to be taken care when extending the inner product to the closure and also as to how convergence is given according to the norm defined by it.

For uniqueness, were there another RKHS $\mathcal{H}'$ with respect to K, we first note that $\mathcal{H}_K \subseteq \mathcal{H}'$. Secondly, for all $f \in \mathcal{H}'$, $f \perp \mathcal{H}_K$ iff $f = 0$ because $\langle f, K_x \rangle_{\mathcal{H}} = f(x) = 0$. Since $\mathcal{H}_K$ is closed, $\mathcal{H}' = \mathcal{H}_K + \mathcal{H}_K^{\perp} = \mathcal{H}'$ would be dense in $\mathcal{H}_K$. For a more detailed proof and deepened study of kernels we recommend [6].

The importance of this result is the characterization of the RKHS exclusively in terms of K, property that is used in the proof of Lemma 4.2.3. For this reason, we shall forgo in this work other pathologic kernels, assuming from here on that K is always Mercer, so that we are supported by the Moore-Aronszayn Theorem.

4.2 Gradient Descent and RKHS

Now to contextualize, we ask: how does RKHS theory may be embedded in the supervised learning framework, culminating in (4.0.1)? As a short answer, it all comes down to an isomorphism between a RKHS and the space of square integrable functions $\mathcal{L}^2(\mu)$. We elaborate as follows.

Consider the instance space $\mathcal{X}$, label space $\mathcal{Y} = [-M, +M]$ and the dataset $S \in \mathcal{Z}^n = (\mathcal{X} \times \mathcal{Y})^n$. We define the loss function as

$$L_\mu(f) \triangleq \int_{\mathcal{Z}} (f(x) - y)^2 \, d\mu \, , \tag{4.2.1}$$

where μ is a probability measure in $\mathcal{Z}$. Since we are now working under the learning theory scheme, we will assume that $\mathcal{X}$ is a compact metric space.

Now, turning our attention to the hypothesis space, we carry out as general as possible, defining $\mathcal{H}$ as the space of all functions f where (4.2.1) is finite, that is, $\mathcal{L}^2(\mu)$. So,

$$L_\mu : \mathcal{H} \, = \, \mathcal{L}^2(\mu) \to \mathbb{R} \, .$$

Defining $\mu_x \triangleq \int_{\mathcal{X}} d\mu$ we have that $\mu = \mu_{y|x} \times \mu_x$. Additionally, the function which minimizes L_μ is

$$f_\mu(x) \triangleq \int_{\mathcal{Y}} y \, d\mu_{y|x} \quad ; \quad f_\mu \, = \, \underset{f \in \mathcal{L}^2(\mu_x)}{\operatorname{argmin}} \, L_\mu(f) \, . \tag{4.2.2}$$

Therefore, the class of functionals that minimize L_μ lie in $\mathcal{L}^2(\mu_x)$, and so, if we are to run an algorithm to optimize it, we might as well set $\mathcal{L}^2(\mu_x)$ as the choice space.

From here on we follow the reasoning in [35]. Although the main result is not of our interest, here we work towards providing a detailed demonstration of its Proposition 2.2, which states the gradient descent formula in the RKHS.

Naturally, the goal of a machine learning model is to find a hypothesis in $L^2(\mu_x)$ whose prediction is as close as possible to those of f_μ. Under a few conditions, however, it is possible to prove that there is a RKHS isomorphic to $L^2(\mu_x)$, allowing us to write the gradient descent algorithm in terms of the kernel. The rest of this chapter is dedicated to prove this claim.

We first state a trivial Corollary of the Moore-Aronszayn theorem, showing us how to construct the unique RKHS given a kernel K.

Corollary 4.2.1. *Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a Mercer kernel. Then the set defined as*

$$\mathcal{H}_K \triangleq \overline{span\{K_x / x \in \mathbb{R}\}} \ , \quad where \quad K_x(\,\cdot\,) \triangleq K(\,x, \cdot\,) \ ,$$

is a RKHS with respect to the linear extension of the inner product $\langle\, K_x, K_y \,\rangle_K \triangleq K(\,x, y\,)$.

The intent is to find a link between $\mathcal{L}^2(\mu_x)$ and the RKHS. Firstly one must notice that, since K is Mercer and $\mathcal{X}$ is compact, $K_x \in \mathcal{L}^2(\mu_x)$ and therefore the elements of $\mathcal{H}_K$ are in $\mathcal{L}^2(\mu_x)$, although they have different topological structures (inner product).

A natural way to construct the isomorphism is to project a given function from $\mathcal{L}^2(\mu_x)$ in the space where K_x acts as some kind of basis, in hopes that property (4.1.1) is valid on that component of f. Formally,

$$\int_{\mathcal{X}} K_y(x) f(x)\, d\mu_x \ = \ \langle\, f, K_y \,\rangle \ ,$$

where $\langle\, \cdot, \cdot \,\rangle$ shall denotes the Euclidean norm and inner product of $\mathcal{L}^2(\mu_x)$.

This quantity is a function of y and measures the alignment of f and K_y. Notice that were K_y an orthogonal basis for $\mathcal{L}^2(\mu_x)$ then the above would be the f's coordinate corresponding to K_y. Indeed, we shall see further on that this reasoning is not far from the truth. Moreover, the inner product of an element from $\mathcal{L}^2(\mu_x)$ and K defines, not by coincidence, a linear transformation which plays a central role in kernel theory[3]:

$$T_K : \mathcal{L}^2(\mu_x) \to \mathcal{C}(\mathcal{X}) \quad ; \quad T_K(f)(\cdot) \ = \ \int_{\mathcal{X}} K(\,x, \cdot\,) f(x)\, d\mu_x \ , \tag{4.2.3}$$

where $\mathcal{C}(\mathcal{X})$ is the Banach space of all real continuous functions in $\mathcal{X}$. This last assertion is not so direct and we justify it in the following lemma.

Lemma 4.2.1. *$T_K(f) \in \mathcal{C}(\mathcal{X})$ for all $f \in \mathcal{L}^2(\mu_x)$.*

Proof. Since $\mathcal{X}$ is compact, K_x is uniformly continuous for all x, that is, given $\epsilon \geq 0$ there is $\delta \geq 0$ such that $|K_x(y) - K_x(z)| \leq \epsilon_x$ for all $||y - z||_{\mathcal{X}} \leq \delta$. Then, for such y and z,

$$\begin{aligned}
&|T_K(f)(y) - T_K(f)(z)| \ = \\
&\Big| \int_{\mathcal{X}} (K(x,y) - K(x,z)) f(x) d\mu_x \Big| \ = \\
&|\langle\, f, K_x(y) - K_x(z) \,\rangle| \ \leq \\
&||K_x(y) - K_x(z)||\, ||f|| \leq \\
&\sqrt{\mu_x(\mathcal{X})} \max_x |K_x(y) - K_x(z)|\, ||f|| \ \leq \ \epsilon ||f|| \quad \text{for all } x, y, z \in \mathcal{X},
\end{aligned}$$

[3]Note that $T(f)(\cdot) = \langle\, f, K(\,\cdot\,, x) \,\rangle$. When this relation holds we say that K is the kernel of T_K, which is unique.

where in the last line we used the fact that $(K_y - K_z)(x)$ admits a finite maximum since $\mathcal{X}$ is compact. ■

Since the image of T_K is continuous, then it is also square integrable. So, with some abuse of notation, we redefine its codomain by adding an inclusion mapping[4], embedding its image in $\mathcal{L}^2$, yielding $T_K : \mathcal{L}^2(\mu_x) \to \mathcal{L}^2(\mu_x)$.

As it just so happens, T_K is actually a compact map, that is, the image of a compact set is relatively compact (a set is relatively compact if its closure is compact). This property unlocks numerous results, most notably the Spectral Theorem for compact operators, unveiling strong consequences. But before proving the compactness of T_K, we must state previously a Theorem that shall aid us:

Theorem 4.2.1. *(Arzelá-Ascoli Theorem) Let $\mathcal{X}$ be a compact metric space. Then a subset B of $C(\mathcal{X})$ is relatively compact in the topology induced by the uniform norm if and only if it is equicontinuous* [5] *and pointwise bounded* [6]*.*

Lemma 4.2.2. *The transformation T_K defined in* (4.2.3) *is compact, that is, the image of a compact set is relatively compact.*

Proof. The idea is to apply the Arzelá-Ascoli Theorem in the image of T_K applied in the ball of radius r, $B_r \subseteq C(\mathcal{X})$.
Let $f \in B_r \subseteq C(\mathcal{X})$ and $f_K \triangleq T_K(f)$. To prove point wise boundness of $T_K(B_r)$ we use the Cauchy-Schwartz inequality and the Reproducing property, giving us

$$|f_K(x)| = |\langle f(\cdot), K(x, \cdot)\rangle| \leq ||f||\,||K_x|| \leq ||K_x||r \quad \forall f_K \in T_K(B_r)\,.$$

For equicontinuity we use again Cauchy-Schwartz paired with the uniform continuity of K,

$$\begin{aligned} &f_K(y) - f_K(y_0) = \int_{\mathcal{X}} [K(x,y) - K(x,y_0)] f(x)\, d\mu_x \implies \\ &|f_K(y) - f_K(y_0)| \leq |\langle f, K_y - K_{y_0}\rangle| \leq \\ &||f||\,||K_y - K_{y_0}|| \leq \epsilon r \quad \forall y \text{ such that } ||y - y_0|| \leq \delta\,. \end{aligned}$$

Note that δ, whilst possibly dependant on y_0 and ϵ, it is constant on f, which implies that the image of any bounded set with respect to T_K is equicontinuous. Gathering these two results, by the Arzelà–Ascoli theorem, all sequences in $T_K(B_r)$ must admit a uniformly convergent subsequence, and so it must also admit one converging in $\mathcal{L}^2(\mu_x)$.

■

[4] This is due to a technicality, since while an element of $\mathcal{C}(\mathcal{X})$ is a function, in $\mathcal{L}^2(\mu_x)$ it is a class.

[5] We say that a set $A \subseteq C(\mathcal{X})$ is equicontinuous if $\forall x and \epsilon \geq 0$, exists a neighborhood of x U_x such that for every $f \in A$, $y \in \mathcal{X}$, $|f(y) - f(x)| \leq \epsilon$.

[6] We say that a set $A \subseteq C(\mathcal{X})$ is pointwise bounded if for every x, $sup_{f\in A}\{f(x)\} \leq \infty$.

Hence, by the Spectral Theorem for compact operators, there exists $\{\lambda_i\}_{i=1}^{\infty}$ and an orthonormal basis $\{e_i\}_{i=1}^{\infty}$ of $\mathcal{L}^2(\mu_x)$ such that $T_K(e_i) = \lambda_i e_i$. Or in other words, the eingenfunctions of T_K form a basis in the domain. Also, since the mapping T_K is continuous, we can choose e_i to be continuous[7].
With this in mind, We are now able to state the result demonstrated by James Mercer in 1909, known as the Mercer's Theorem. It shows that the kernel's coordinates are equal to the eigenvalues of T_K, each of which corresponding to an eigenfunction, implicating that the kernel and the operator defined by 4.2.1 are related one to one. This result is the essence of the representer theorem and essential for our work.

Theorem 4.2.2. *(Mercer's Theorem) Let K be a Mercer kernel defined in $\mathcal{X} \times \mathcal{X}$, then $\{\lambda_i\}_{i=1}^{\infty}$ are strictly positive and*

$$K(x, y) = \sum_{i=1}^{\infty} \lambda_i e_i(x) e_i(y) , \tag{4.2.4}$$

where the convergence is uniform in $\mathcal{X} \times \mathcal{X}$ and absolute for each $(x, y) \in \mathcal{X}$.

Proof. [6] ■

Indeed, all Mercer kernels may be written in terms of functions mapping the set of instances to a Hilbert Space $\phi : \mathcal{X} \to \mathcal{H}$, commonly called feature maps, such that $K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ holds. The above theorem shows this correspondence by setting $\mathcal{H} = \ell^2$ and $\phi(x) = \{\sqrt{\lambda_i} e_i(x)\}_{i=1}^{\infty}$, where ℓ^2 is the space of real sequences $x = \{x_i\}$ such that $\sum x_i^2 \leq \infty$.
Since the kernel's action is completely characterized by the operator, a natural step is to characterize the whole space $\mathcal{H}_K$ in terms of the eigenfunctions and eigenvalues. Finally, the isomorphism follows as a corollary.

Lemma 4.2.3. *Given a Mercer kernel and T_K a compact operator defined in* (4.2.3)*, then*

$$\mathcal{H}_K = \mathcal{H}_0 \triangleq \Big\{ f \in \mathcal{L}^2(\mu_x) \, ; \, \sum_i^{\infty} \frac{\langle f, e_i \rangle^2}{\lambda_i} < \infty \Big\} . \quad \textit{Moreover, for } f, g \in H_K ,$$

$$\langle f, g \rangle_K = \langle f, g \rangle_{\mathcal{H}_0} \triangleq \sum_i^{\infty} \frac{\langle f, e_i \rangle \langle g, e_i \rangle}{\lambda_i} . \tag{4.2.5}$$

Proof. Note that all there is to prove is that $\mathcal{H}_0$ is a RKHS with respect to K and to the inner product defined in (4.2.5), since by the Moore-Aronszayn theorem, the RKHS must be unique.
Obviously, $\mathcal{H}_0$ is a normed vectorial space under $|| \cdot ||_{\mathcal{H}_0}$ and (4.2.5) is an inner product. As to completeness, the demonstration is analogous to that of the completeness of ℓ^2, and so

[7] Given an equivalence class $\tilde{e}_j$, we may assign to it a continuous function $e_j \triangleq \lambda_j^{-1} \langle \tilde{e}(x), K(\cdot, x) \rangle \in \mathcal{C}(\mathcal{X})$.

$\mathcal{H}_0$ is a Hilbert Space.
As for the reproducing property, using equation (4.2.4),

$$\langle K_x, e_j \rangle = \lambda_j e_j(x) \Longrightarrow \sum_j^{\infty} \frac{\langle K_x, e_j \rangle^2}{\lambda_j} = \sum_j^{\infty} \lambda_j e_j(x)^2 = K(x,x) \leq \infty .$$

Therefore $K_x \in \mathcal{H}_0$, and so, for $f \in \mathcal{H}_0$, $\langle f, K_x \rangle_K$ is well defined, giving

$$\langle f, K_x \rangle_{\mathcal{H}_0} = \sum_k^{\infty} \frac{\langle f, e_k \rangle \langle K_x, e_k \rangle}{\lambda_k} = \sum_k^{\infty} \frac{\langle f, e_k \rangle}{\lambda_k} \sum_{i=1}^{\infty} \lambda_i e_i(x) \langle e_i, e_k \rangle =$$
$$\sum_k^{\infty} \langle f, e_k \rangle e_k(x) = f(x) .$$

Thus, $\mathcal{H}_0$ has the reproducing property. By the Moore-Aronszajn Theorem, it must be then isomorph to the closed span of K_x. ■

Corollary 4.2.2. *$T_K^{\frac{1}{2}}$ is an isomorphism between $\mathcal{H}_K$ and $\mathcal{L}^2(\mu)/ker(T_K)$, where the fractional operator T_K^r is defined as $T_K^r(e_i) = \lambda_i^r e_i \ \ \forall r \in \mathbb{R}$.*

Proof. Since $T_K^{\frac{1}{2}}$ is linear, the algebraic structure is preserved, and so is enough to show that it is an isometry. Using the decomposition of f in terms of the basis,

$$|| T_k^{\frac{1}{2}}(f) ||_K^2 = \Big|\Big| \sum_i^{\infty} \sqrt{\lambda_i} \langle f, e_i \rangle e_i \Big|\Big|_K = \sum_j^{\infty} \frac{\sum_i \lambda_i \langle \langle f, e_i \rangle e_i, e_j \rangle^2}{\lambda_j} = \sum_j^{\infty} \frac{\lambda_j \langle f, e_j \rangle^2}{\lambda_j} =$$
$$||f||^2 .$$

■

Hence, $T^{\frac{1}{2}}$ maps each feasible point of the optimization in (4.2.2) to a point in H_K, preserving both topological and algebraic structures. The function to be minimized is yet the same, but not y the optimization of $L_p(f)$ via gradient descent in $\mathcal{L}^2_{\mu_x}$ is equivalent to the optimization in $\mathcal{H}_K$. The reason to use this equivalency is because gradient descent may be written in a simple way in the RKHS. In this endeavor we remind the definition of a Fréchet's derivative.

Definition 4.2.1. *Let V and W be normed vector spaces and $U \subset V$ be an open subset of V. A function $L : U \to W$ is called Fréchet differentiable at $v \in U$ if there exists a bounded linear operator $T : V \to W$ such that*

$$\lim_{h \to 0} \frac{|| L(v+h) - L(v) - T_v(h) ||_W}{|| h ||_V} = 0 . \tag{4.2.6}$$

We call $D_L(v) \triangleq T_v$ the Fréchet derivative of f at v.

Theorem 4.2.3 ([35]). *The derivative of the functional* $V : \mathcal{H}_K \to \mathbb{R}$, $V(f) \triangleq (f(x) - y)^2$ *at* f *is given by*

$$D_V(f)(\cdot) = \langle 2(f(x) - y)K_x, \cdot \rangle_K.$$

Proof. Developing the numerator in (4.2.6),

$$\begin{aligned}
&| V(v + h) - V(v) - D_V(f)(h) | = \\
&| (f(x) + h(x))^2 - 2(f + h)(x)y + 2f(x)y - f(x)^2 - D_V(f)(h) | = \\
&| 2f(x)h(x) + h(x)^2 - 2h(x)y - D_V(f)(h) | = \\
&| 2f(x)h(x) + h(x)^2 - 2h(x)y - 2(f(x) - y)h(x) | = h(x)^2 .
\end{aligned}$$

Now, using the definition,

$$\begin{aligned}
&\frac{| V(v + h) - V(v) - D_V(f)(h) |}{|| h ||_K} = \frac{h(x)^2}{|| h ||_K} \leq \frac{|| K_x ||_K^2 || h ||_K^2}{|| h ||_K} = || K_x ||_K^2 || h ||_K \implies \\
&\lim_{h \to 0} \frac{| V(v + h) - V(v) - D_V(f)(h) |}{|| h ||_K} = 0
\end{aligned}$$

where in order to achieve the inequality we used

$$h(x)^2 = \langle h, K_x \rangle_K^2 \leq || K_x ||_K^2 || h ||_K^2 .$$

■

Therefore, the gradient map of V is $gradV(f) = 2(f(x) - y)K_x$. That is, for each f, $gradV$ maps f to the function in $\mathcal{H}$ which represents the functional $D_V(f) \in \mathcal{H}^*$.
Given a dataset $S = \{(x_i, y_i)\}_{i=1}^n$ and using the linearity of the Fréchet's gradient, $gradL_S(f)$ may be written as

$$gradL_S(f) = grad(\hat{\mathbb{E}}_S[V(f)] = \hat{\mathbb{E}}_S[gradV(f)] = \frac{2}{n} \sum_{i=1}^{n} (f(x_i) - y_i)K_{x_i} ,$$

where $\hat{\mathbb{E}}_S$ is the empirical average over S.
Therefore, defining f_t^S as the hypothesis given by gradient descent in the t-th step, we arrive at the iteration rule in the RKHS mentioned in (4.0.1):

$$f_{t+1}^S \triangleq f_t^S - \frac{2}{n} \sum_{i=1}^{n} (f_t(x_i) - y_i)K_{x_i} .$$

As previously mentioned, the advantage of this reformulation is the means it provides to track the derivatives's evolution during the optimization. Indeed, to describe the behaviour of $\partial_u f$ we may simply derive the above equation, as states Theorem 1 in [38]:

Theorem 4.2.4. ([38] *Let* $s \in \mathbb{N}$ *and* K *a Mercer kernel defined in* $\mathcal{X} \times \mathcal{X}$ *such that* $K \in C^{2s}(\mathcal{X} \times \mathcal{X})$. *Then follows that, For all* $x \in \mathcal{X}$ *and* u *:*

(i) $\partial_u(K_x) = (\partial_u K)_x \in \mathcal{H}_K$.

(ii) For $f \in \mathcal{H}_K$, $\langle f, \partial_u K_x \rangle_{\mathcal{H}_K} = \partial_u f(x)$.

Thus, since the gradient descent is contained inside $\mathcal{H}_K$, which in turn contains the derivatives of power less or equal to s of K_x , we may write

$$\partial_u f_{t+1}^S \triangleq \partial_u f_t^S - \frac{2}{n} \sum_{i=1}^{n} (f_t(x_i) - y_i) \partial_u K_{x_i} .$$

We have then derived an analytical way to describe the derivatives of the function in each step of the gradient descent.

Assuming again the existence of an oracle f^*, the quality of information compression may be measured by the distances of all partial derivatives $|\partial_u f - \partial_u f^*|$. Having this is mind, a natural question is how the information compression relates to the training error. The above equation shows that the derivatives evolve according to $\partial_u K_{x_i}$, weighted by the error in each point of S. This means that the convergence of the variation depends only on the error in each point of the dataset. Furthermore, assuming that the K_{x_i} are linearly independent, the derivatives of f will be skewed towards the direction whose corresponding point had the most contribution to error during training.

Also, the result is consistent with the early stop heuristic, since by interrupting learning before a certain threshold one is preventing the variation to increase even further.

Finally, it should be mentioned that this result is well known in kernel theory. Our intention presenting it is to expand its utility, not simply because it is the correct approach in displaying the optimization for analytical learning, but also because it sheds new light on our concept of information compression.

4.3 Final Remarks

Having settled the formalities, in this section we attend to brief observations which we hope clarify the discussion in the last two sections. The following reading is by no means a necessity for the final objective of this work and the reader may disregard it if need be.
Notice that during this chapter the Mercer kernel K was always assumed arbitrary, and thus are free to choose the space H_K in which he wishes to optimize. We list a few examples for concreteness so that the reader may have a firmer grasp about these mappings.

• **Polynomial Kernel:** Let $x,\, x', c \in \mathcal{X} = \mathbb{R}^m$. The polynomial kernel is given by

$$K(x, x') \;=\; (x \cdot x' + c)^d \;.$$

Developing the equation, we may derive the corresponding feature map. For simplicity we incorporate the constant c by setting $x_0 \,=\, c,\, x'_0 \,=\, 1$.

$$K(a,b) \;=\; \left(\sum_{i=1}^{m} x_i x'_j + c\right)^d \;=\; \left(\sum_{i=0}^{m} x_i x'_j\right)^d \;=$$

$$\sum_{J\in\{0,1,\ldots,m\}^d} \prod_{i=1}^{d} x_{J_i} x'_{J_i} \;=\; \sum_{J\in\{0,1,\ldots,m\}^d} \prod_{i=1}^{d} x_{J_i} \prod_{i=1}^{d} x'_{J_i} \;,$$

where J_i is the i-th entry of the vector J.
Hence we define $\psi : \mathcal{X} \to \mathbb{R}^{(m+1)^k}$ as $\psi_J(x) \triangleq \prod_{i=1}^{m} x_{J_i}$ for every $J \in \{0, 1, \ldots, n\}^k$, and so $\langle \psi(x), \psi(x') \rangle \,=\, K(x, x')$.
Notice that

$$\sum_{J\in\{0,1,\ldots,m\}^d} w_J \psi(x)_J$$

is an polynomial of degree d in its general form. If one were to solve a classification task by learning the space of all halfspaces $h \in \mathcal{H}$, $h_w(x) \triangleq \langle w, x \rangle_{\mathcal{X}}$, then by means of the kernel trick one could choose to learn it in the feature space $\langle w, \psi(x) \rangle_{\mathcal{X}}$, which corresponds to a polynomial regression of degree k.

• **Gaussian Kernel:** Let $x,\, x' \in \mathcal{X} = \mathbb{R}$. Then the Gaussian kernel is given by

$$K(x, x') \;=\; e^{-\frac{||x-x'||^2}{2\sigma}} \;.$$

Here the instance space is mapped to the space of square convergent sequences ℓ^2. Indeed, consider $\psi : \mathcal{X} \to \ell^2$ where the n-th element of the sequence is $\psi(x)_m \triangleq$

$\frac{1}{\sqrt{m!}}e^{-\frac{x^2}{2}}x^m$. Then

$$\langle \psi(x), \psi(x') \rangle_{\ell^2} = \sum_{n=0}^{\infty} \left(\frac{1}{\sqrt{m!}} e^{-\frac{x^2}{2\sigma}} x^m \right) \left(\frac{1}{\sqrt{m!}} e^{-\frac{(x')^2}{2\sigma}} (x')^m \right) =$$
$$e^{-\frac{x^2+(x')^2}{2\sigma}} \sum_{i=0}^{\infty} \left(\frac{(xx')^m}{n!} \right) = e^{-\frac{(x-x')^2}{2\sigma}} ,$$

where in the last line we used the formulation in infinite series of the exponential. Note that if $x = x'$ and $\sigma = 1$ then $||\psi(x)||_{\ell^2} = 1$, so in this case the Gaussian kernel normalizes the previous notion of similarity in the original space.

The kernels listed above are the most common in machine learning practice. Maybe unsatisfied, one might ask how to conceive other Mercer kernels, and the answer is surprisingly simple: there are as many Mercer kernels on a Hilbert space $\mathcal{X}$ as there are bounded linear operators $T : \mathcal{X} \to \mathcal{X}$ which are self adjoint and positive semi-definite. Indeed, these spaces are isomorphic and correspond to each other in the following manner.

Lemma 4.3.1. *Let $\mathcal{X}$ be a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{X}}$. Then $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a Mercer kernel if and only if there exists an operator T following the above assumptions and*

$$K(x, x') = \langle x, T\big(x'\big) \rangle_{\mathcal{X}} .$$

Notice that if T can be factored into fractional operators $T^{1/2}$ such that $T^{\frac{1}{2}} \circ T^{\frac{1}{2}} = T$, then we can write $K(x, x') = \langle T^{\frac{1}{2}}\big(x\big), T^{\frac{1}{2}}\big(x'\big) \rangle_{\mathcal{X}}$, and we also have the feature map.
Finally, its worth mentioning that the feature map corresponding to a kernel K may not be unique. For example, consider $\mathcal{X} = \mathbb{R}$. Then $K(x, x') = xx'$ admits $\psi_1(x) = x$ and $\psi_2(x) = \frac{x}{\sqrt{2}}(e_1 + e_2)$ as its feature maps, with corresponding feature space being $\mathbb{R}$ and $\mathbb{R}^2$ respectively.

Chapter 5

A Fourier analysis of Neural Networks

So far, after introducing Reproducing Kernel Theory, we developed on the main concept which will be used from now on, the Hardy-Krause Variation. On this chapter, we use it to guide us through the reasoning and results from [28]. On this article, by the usage of spectral theory, a regularization approach using frequencies is taken, and ReLU Neural Networks are found to show a special behaviour. However, nothing is said about the generalization gap or overparametrization.

On section 5.1 we present the main concepts and results of [28], making parallels with generalization. On section 5.2 we attempt to link the results drawn from Fourier theory to with the Hardy-Krause Variation and its stability interpretation described in chapter 4. In subsection 5.2.1 we generalize the main results of [28] to $V[f]$, and in 5.2.2 we give a qualitative argument towards a possible contribution to generalization in the overparametrized regime.

5.1 An implicit regularization

As exposed before, many evidences point to a possible implicit regularization in neural networks, as if the very choice of this hypothesis space favored simple functions. In the last chapter, we used a deterministic bound to attempt to track down a regularization factor that might help understand learning in neural networks, and by using a compression oriented interpretation, we studied how it evolved during gradient descent in an arbitrary hypothesis space. Here we complete our work by proceeding with our analysis, now restrained to ReLU neural networks, arriving at interesting results. In such endeavor, however, new tools are required.

We will explore the work of [28] which displays an investigation of ReLU networks using Fourier theory, showing that choosing this hypothesis space implies in a certain inclination for fitting lower frequencies. Before delving into it we stress that the generalization error is not the main object of this chapter (and neither of [28]), but in order to elucidate DNN's learning behaviour we ought to understand the implications of choosing it as the hypothesis space.

Many results have been derived for linear neural networks, including an almost complete analytical description of their training dynamic [1]. However, these have limited relevance in practice and are studied to expand the results to the non linear networks, hopefully generalizing some properties. ReLU activation functions represent the simplest non linearity, making their study a natural first step towards understanding general architectures, one that we take in this chapter.

5.1.1 The Main theorem

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a ReLU neural network with $L-1$ hidden layers, all composed by N neurons. Throughout this work we assume that f has a compact support, a reasonable supposition for all practical matters. We may then write f as

$$f = T^{(L)} \circ r \circ \cdots r \circ T^{(2)} \circ r \circ T^{(1)} , \tag{5.1.1}$$

where $r(x) = max(0, x)$ is applied coordinatewise, $T^{(l)}$ is an affine transformation from $\mathbb{R}^N$ to $\mathbb{R}^N$ for $1 < l < L$, and $T^L : \mathbb{R}^N \to \mathbb{R}, T^1 : \mathbb{R}^d \to \mathbb{R}^N$.

To analyze the behaviour of networks in the frequency domain, one must have some kind of access to its Fourier's transform. However, is too complicated to simply apply the transform, so we need another way to make this approach tractable. This is a first advan-

[1] [30]

tage gained from the ReLU assumption: being a composition of piecewise linear functions, f must then also be piecewise linear[2]. Hence, the first objective of [28] is to write f as the sum of affine functions times an indicator, unwinding the contribution of each T^l in each linear region.

Therefore, we must first try to discriminate such regions. Since the non-linearity of f comes solely from the presence of r in (6.1.1), a linear region P is one where the neurons clipped by the activation function are the same neurons, that is, the region of all x where $r \circ T^l(x)|_P$ acts as a linear transformation for each l. For the example where $l = 1$, defining $T_i^{(1)}$ as $T^{(1)}$ with the i-th row set to zero, then $r \circ T^{(1)}(x) = T_i^{(1)}(x)$ for all x where $[T^{(1)}(x)]_i \leq 0$. Thus, the linear region P_i of $T_i^{(1)}(x)$ is the one which induces the activation pattern where the i-th neuron is clipped. Naturally, there will be as many linear regions as there are activation patterns, which in turn increases exponentially with the number of layers and vertices. In this example, the number of combinations of null rows in $T^{(1)}$ equals the number of linear regions of $r \circ T^l(x)|_P$, that is, 2^d.

Formally, given an activation pattern $e^{(1)}, e^{(1)} \in \{-1, 1\}^N$, we define

$$[T^{(1)}_{\epsilon^{(1)}}(x)]_i = \begin{cases} [T^{(1)}(x)]_i & \text{if } \epsilon_i^{(1)} = 1. \\ 0 & \text{otherwise.} \end{cases} \tag{5.1.2}$$

Hence, for a given activation pattern, the points x which reproduces it satisfy

$$\left(\epsilon_i^{(1)}\right) \left[\left(T^{(1)}_{\epsilon^{(1)}}\right)(x)\right]_i \geq 0 \quad i = 1, \ldots, N \, .$$

Notice that the points where the above is negative are exactly the points where 5.1.2 does not behave like $r \circ T^1(x)$.

Expanding this reasoning for any number of layers, consider a set of L binary vectors $e = \{\epsilon^{(1)}, \ldots, \epsilon^{(L)}\}$ with $\epsilon^{(l)} \in \{-1, 1\}^N$, where $f(x)$ will behave linearly when $\epsilon_i^{(l)} = -1$ under the x's which nullify the i-th entry of the l-th layer (view figure 5.1). This then allows us to parametrize the linear regions (or equivalently, activation patterns) with a binary vector. Also, let $[T^{(l)}_{\epsilon^{(l)}}(x)]_i$ be defined as in (5.1.2). Hence, fixed e, the points that belong to the linear region associated with it are characterized by the following inequality (for all $l = 1, \cdots, L$):

$$\left(\epsilon_i^{(l)}\right) \left[\left(T^{(l)} \circ T^{(l-1)}_{\epsilon^{(l-1)}} \circ \cdots \circ T^{(1)}_{\epsilon^{(1)}}\right)(x)\right]_i \geq 0 \, , \quad i = 1, \ldots, N \, .$$

We now structure the above conclusions in the following Theorem:

Theorem 5.1.1. ([28])

[2]Indeed, given two piecewise linear functions f_1 and f_2, then the linear regions of f defined by $f(x) = f_2(f_1(x))$ will simply be $f_1^{-1}(A_i)$, where A_i are the linear regions of f_1

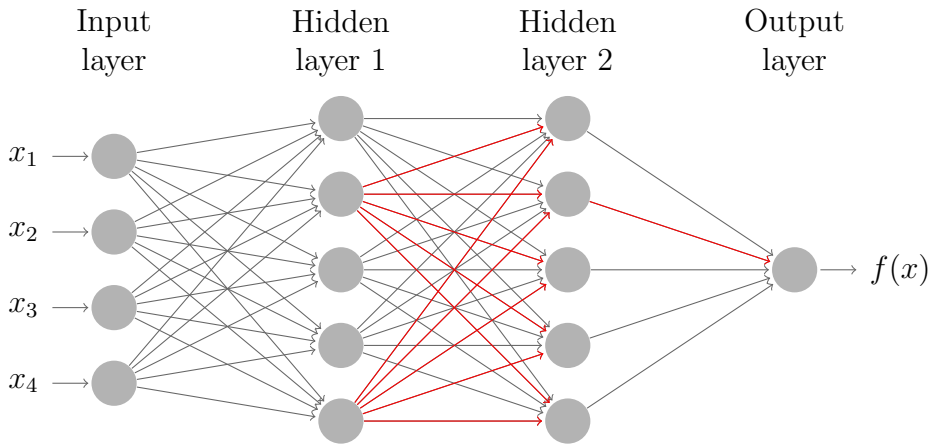


Figure 5.1: *A graph representation of a ReLU network with $d = 4$, $L = 3$, $N = 5$, $e^1 = \{1, -1, 1, 1, -1\}$, $e^1 = \{1, -1, 1, 1, 1\}$. The red lines irradiating from neuron 2 and 3 in the first layer indicate that $[T^1(x)]_2$ and $[T^1(x)]_5$ are negative for a fixed choice of x (analogously for layer 2). All x which convey this pattern in the network will belong to the same linear region, since this behaviour is equivalent to ignoring r and simply nullifying the second and fifth row of T^1 and the second one of T^2.*

Consider f defined in 6.1.1. Then it may be written as

$$f(x) = \sum_{\epsilon^{(1)},\cdots,\epsilon^{(L)}} \mathbb{1}_{P_\epsilon}(x)\Big(T^{(L)} \circ T^{(L)}_{\epsilon^{(L)}} \circ \cdots \circ T^{(1)}_{\epsilon^{(1)}}\Big)(x) \ ,$$

where $\epsilon = \{\epsilon^{(i)}\}_{i=1}^{L}$ and 1_{P_ϵ} denotes the indicator function of the N dimensional polytope [3] *$P_\epsilon \in \mathbb{R}^d$, defined as the set of solutions for the following inequalities (for all $l = 1, \cdots, L$):*

$$\big(\epsilon_i^{(l)}\big)\Big[\Big(T^{(l)} \circ T^{(l-1)}_{\epsilon^{(l-1)}} \circ \cdots \circ T^{(1)}_{\epsilon^{(1)}}\Big)(x)\Big]_i \geq 0 \ , \quad i = 1, \ldots, N \ .$$

Corollary 5.1.1. *Given a ReLU neural network defined as in* (6.1.1)*, then we may write it in a linear spline format*

$$f(x) = \sum_{\epsilon} \mathbb{1}_{P_\epsilon}(x)(w_\epsilon x + b_\epsilon) \ , \tag{5.1.3}$$

where $w_\epsilon \triangleq \prod_{l=1}^{L+1} W_\epsilon^{(l)}$, and $W_\epsilon^{(l)}$ is the matrix of $T^{(l)}$ with its i-th row set to zero whenever $\epsilon_i^l = -1$[4]*.*

Needless to say, being able to write f in its piecewise linear form is extremely useful. In our case, it provides the means to calculate its Fourier transform, here defined as

$$\tilde{f}(k) = \int_{\mathbb{R}^d} f(x)e^{-ikx}dx \quad ; \quad f(x) = \frac{1}{2\pi}\int_{\mathbb{R}^d} \tilde{f}(k)e^{ikx}dk \ .$$

We remind the reader that since we are assuming that the network f is a real continuous

[3] For us a polytope is a finite intersection of half spaces.

[4] We use the notation w_ϵ because $\prod_{l=1}^{L+1} W_\epsilon^{(l)}$ is a vector since the network's output is a real number.

function with a compact support, its Fourier transform is well defined. Also, as it was in the other chapters, $||\cdot||$ will denote the Euclidean norm.

Theorem 5.1.2. ([28]) *The Fourier transform of f defined in* (5.1.3) *is given by*

$$\tilde{f}(k) \;=\; \sum_{\epsilon} \frac{w_\epsilon k}{i||k||^2}\, \tilde{\mathbb{1}}_{P_\epsilon}(k)\,. \tag{5.1.4}$$

Proof. For a fixed frequency vector k, the vector-valued function $(f(x)e^{ik\cdot x})\,k$ is continuous and has continuous derivatives almost everywhere (because of $f(x)$ piecewise linearity). Since f has a finite support, the divergence Theorem gives us

$$\int_{\mathbb{R}^d} \nabla_x \cdot [kf(x)e^{-ik\cdot x}]dx \;=\; \oint_{\Gamma} kf(x)e^{-ik\cdot x} \cdot \mathbf{n}dx \;=\; 0$$

for Γ being any closed surface outside the support, $\mathbf{n}$ its normal and ∇_x the divergent. Because in the above equation k is seen as a constant, the integrand on the left may be calculated as follows[5]:

$$\nabla_x \cdot \Big[kf(x)e^{-ik\cdot x}\Big] \;=\; \sum_{j=1}^{n} \partial_{x_j}\Big[k_j f(x)e^{-ik\cdot x}\Big] \;=\; \sum_{j=1}^{n} k_j f_{x_j}(x)e^{-ik\cdot x} + \sum_{j=1}^{n} -ik_j^2 f(x)e^{-ik\cdot x}$$
$$=\; k\cdot(grad_x f)(x)e^{-ik\cdot x} - i||k||^2 f(x)e^{-ik\cdot x}\,.$$

Substituting the above result in the integral,

$$\int_{\mathbb{R}^d} k\cdot(grad_x f)(x)e^{-ik\cdot x} - i||k||^2 f(x)e^{-ik\cdot x}\,dx \;=\; 0 \;\longrightarrow$$
$$\int_{\mathbb{R}^d} k\cdot(grad_x f)(x)e^{-ik\cdot x}dx \;=\; \int_{\mathbb{R}^d} i||k||^2 f(x)e^{-ik\cdot x}dx \;=\; i||k||^2\tilde{f}(k) \;\longrightarrow$$
$$\tilde{f}(k) \;=\; \frac{1}{i||k||^2}k\cdot\int_{\mathbb{R}^d}(grad_x f)(x)e^{-ik\cdot x}dx\,,$$

where the last integral is taken over each component of the gradient vector. Using (5.1.3), $grad_x f(x) \;=\; \sum_\epsilon w_\epsilon \mathbb{1}_{P_\epsilon}(x)$, and so

$$\tilde{f}(k) \;=\; \frac{1}{i||k||^2}\sum_{\epsilon} k\cdot w_\epsilon \int \mathbb{1}_{P\epsilon}(x)\, e^{-ik\cdot x}dx \;=\; \sum_{\epsilon}\frac{w_\epsilon\cdot k}{i||k||^2}\,\tilde{\mathbb{1}}_{P_\epsilon}(k)\,.$$

■

Now we are left with the Fourier transform of an indicator function of a polytope. Fact is that $\tilde{\mathbb{1}}_{P_\epsilon}$ is a fairly intricate object, and it is not so easily derived. In [13] a recursive application of Stokes Theorem is used in order to calculate it, the demonstration, however, is beyond the scope of this work. Moreover, we are not interested in the exact Fourier

[5] We remember the reader that the divergent for a vector valued function u is defined as $\nabla_x u \;=\; \sum_{j=1}^{n}\partial_{x_j}u_j$.

transform of f, but in how its spectral magnitude depends on $||k||$, so its enough for us to evoke lemma 2 from [28].

Lemma 5.1.1. *Let P be a full dimensional polytope in $\mathbb{R}^d$. Then*

$$|\tilde{\mathbb{1}}_P(k)| \;=\; \mathcal{O}\left(\frac{1}{k^{\Delta_k^{(P)}}}\right). \tag{5.1.5}$$

where $1 \leq \Delta_k^{(P)} \leq d$ and $\Delta_k^{(P)} = j$ when k is orthogonal to some $(d-j)$ dimensional face of P.

For our later conclusions in this section $\Delta_k^{(P)}$ will play a central role, so it is worth investing time discussing it.

A polytope is a generalization of a polyhedron to higher dimensions. P may be seen as a gathering of j-dimensional polytopes, $j < d$, which are called its j-th facet. For example, if P is a cube, then the planes are the 2-th facet and the edges the 1-th one. The theorem above states that the spectrum's decay of $|\tilde{\mathbb{1}}_P(k)|$ has order equal to the smallest codimension[6] among all facets of P to which k is orthogonal. In the case when k is not orthogonal to any facet of P_ϵ, then $\Delta^{(P_\epsilon)} = d$.

This result shows that the behaviour of $|\tilde{\mathbb{1}}_P(k)|$ is highly anisotropic, that is, the frequency magnitude's decay depends heavily on the direction of the vector k, since orthogonality is paramount. Additionally, since $\mathbb{R}^n$ may be decomposed into a direct sum $A \oplus A^\perp$, it means that the space of all k's that are orthogonal to a $(d-j)$-dimensional subspace has dimension j. Therefore, as we increase j, almost all directions will have a high $\Delta_k^{(P)}$ according to a Lebesgue measure[7].

Putting together (5.1.4) and (5.1.5), we arrive at the main Theorem of [28], which gives the spectral decay of ReLU networks.

Theorem 5.1.3. ([28]) *The coordinates norm of a ReLU neural network f in its Fourier domain satisfy*

$$|\tilde{f}(k)| = \mathcal{O}\left(\frac{N_f L_f}{||k||^{\Delta_k^{(P)}+1}}\right),$$

where N_f is the number of linear regions, $\Delta_k^{(P)} \triangleq \min_\epsilon \Delta_k^{(P_\epsilon)}$ and $L_f \triangleq \max_\epsilon ||w_\epsilon||$ is the Lipschitz constant of f.

Proof. Using (5.1.4), (5.1.5) and Cauchy-Schwartz,

$$|\tilde{f}(k)| \;\leq\; \sum_\epsilon \left|\frac{w_\epsilon k}{i||k||^2}\right| |\tilde{\mathbb{1}}_{P_\epsilon}(k)| \;\leq\; \sum_\epsilon \frac{||w_\epsilon||}{||k||} |\tilde{\mathbb{1}}_{P_\epsilon}(k)| \;\leq\; \frac{N_f \max_\epsilon ||w_\epsilon||}{||k||} \sum_\epsilon |\tilde{\mathbb{1}}_{P_\epsilon}(k)|.$$

[6] Given V a vectorial space and W a subspace, then $codim W = dim V - dim W$.

[7] Here we are using the fact that the Lebesque measure of lower dimensional subspaces is zero.

Since we are interested in the order, to make the above independent on ϵ we must choose it aiming to minimize $\Delta_k^{(P_\epsilon)}$.
Let $\Delta_k^{(P)} = \min_\epsilon \Delta_k^{(P_\epsilon)}$. Then, in light of lemma 5.1.1, $\Delta_k^{(P)}$ is the minimal integer such that we can find a $\Delta^{(P)}$-codimensional facet of some P_ϵ to which k is orthogonal. We may then write

$$|\tilde{f}(k)| = \mathcal{O}\left(\frac{N_f L_f}{||k||^{\Delta_k^{(P)}+1}}\right).$$

■

Hence, the coefficients of ReLU neural networks in the Fourier domain decrease, at the very worst case, quadratically with the frequency norm. This means that, as $|k|$ increases, the space of parameters which achieves a fixed magnitude for a certain frequency k shrinks. This means that there are, for fixed weights, less networks capable of fitting high frequencies.

Such decay rate is not so easily achieved even among almost everywhere C^1 function's. For instance, $\sqrt{|x|}$ is $C^1(\mathbb{R} - \{0\})$ and decays with $k^{-1.5}$, as is noted in [28].

Looking back to the network, we see that by translating f to its linear spline form, despite the analytical advantage, we lost track of the network parameter's contribution to the Fourier's coefficients. To mend this technicality, we may further develop the last result to unwind the Lipschitz constant's dependency on the parameters.

Corollary 5.1.2. ([28]) *Defining $\sigma(A)$ as the spectral norm of A, we have*

$$|\tilde{f}(k)| = \mathcal{O}\left(\frac{N_f \prod_{\ell=1}^{L} \sigma(T^{(\ell)})}{||k||^{\Delta_k^{(P)}+1}}\right). \tag{5.1.6}$$

Proof. Consider a family $\{f_i\}_{i=1}^n$ of composable c_i - Lipschitz functions. Then

$$\begin{aligned} &|(f_n \circ \ \dots \ \circ f_1)(x) - (f_n \circ \ \dots \ \circ f_1)(y)| \leq \\ &c_n|(f_{n-1} \circ \ \dots \ \circ f_1)(x) - (f_{n-1} \circ \ \dots \ \circ f_1)(y)| \leq \prod_{i=1}^{n} c_i|x-y|, \end{aligned} \tag{5.1.7}$$

which means that $\prod_{i=1}^n c_i$ is the Lipschitz constant of the function defined as the composition of $\{f_i\}_{i=1}^n$.
Since f is a composition of linear transformations and ReLU functions[8], then
$L_f \leq \prod_{k=1}^{L} \sigma(T^{(k)})$.

■

We now take a brief step back to fathom these two last results under the lens of the machine learning framework. First of all, the magnitude of high frequency amplitudes is a natural way to measure the complexity of a function: since noise often comes in disruptive

[8]We remind that $sup\left(\frac{||Tx||}{||x||}\right) = \sigma(T)$.

oscillations, in order to overfit the hypothesis would need to have high magnitudes associated with high frequencies[9]. The main theorem of [28] shows that these magnitudes are controlled by the frequency's norm and some quantity representing the network's parameters.

As shall be made precise in the next Lemma, the ReLU activation function gives us a higher exponent in $||k||$, making so that the weights needed to achieve a certain fit in high frequencies be much larger in comparison to other models. This shows that increments in the parameters during optimization have a smaller contribution to the capacity of the hypothesis space in the Fourier domain, supporting the idea that this space endows the algorithm with an extra regularization factor.

So far we have restrained our discussion to ReLU neural networks. However, in order to understand why they are special, one must not only show that they have pivotal properties for learning but also show that these properties are not present in other common models. With this in mind, one might wonder about the spectral bias of other architectures. Of course, for any function with a convergent Fourier series, the magnitude of the amplitudes must decrease with $||k||$, the matter at hand is the order with which it decays.

In the following [28] develops a result similar to Equation 5.1.5, but now for any Lipschitz activation function. It presents a bound with a slower rate, leading us to believe that ReLU network's order is indeed not so easily found in other models. The demonstration relies on the following lemma, which relates the Lipschitz constant to Fourier's coefficients.

Lemma 5.1.2. *For any real c-Lipschitz function h defined $\mathbb{R}^n$ with a finite support, it holds that*

$$|\tilde{h}(k)| \leq \frac{Vc}{||k||} ,$$

where $V = Vol(supp(h))$.

Proof. Using that $\exp\{-ik(x - \frac{\pi k}{||k||^2})\} = -\exp\{-ikx\}$, we may translate x by a factor of $-\frac{\pi k}{||k||^2}$ in the Fourier Transform, arriving at

$$\tilde{h}(k) = \frac{1}{2\pi}\int_{\mathcal{X}} h(x)e^{-ikx}dx = -\frac{1}{2\pi}\int_{\mathcal{X}} h\left(x - \frac{\pi k}{||k||^2}\right)e^{-ikx}dx .$$

[9]This notion is not formal and will be discussed in section 5.2.1.

Multiplying the equation by $\frac{1}{2}$ and summing both sides, we have

$$\tilde{h}(k) = \frac{1}{4\pi}\int_{\mathcal{X}} h(x)\, e^{-ikx}dx - \frac{1}{4\pi}\int_{\mathcal{X}} h\left(x - \frac{\pi k}{||k||^2}\right) e^{-ikx}dx \quad \longrightarrow$$

$$|\tilde{h}(k)| = \left| \frac{1}{4\pi}\int_{\mathcal{X}} h(x)\, e^{-ikx}dx - \frac{1}{4\pi}\int_{\mathcal{X}} h\left(x - \frac{\pi k}{||k||^2}\right) e^{-ikx}dx \right| \leq \frac{1}{4\pi}\int_{\mathcal{X}} \left| h(x) - h\left(x - \frac{\pi k}{||k||^2}\right)\right| dx$$

$$\leq \frac{c}{4\pi}\frac{\pi}{||k||}V = \frac{Vc}{||k||}\,.$$

■

Theorem 5.1.4. [10] *Let f be a neural network with $L-1$ hidden layers, width N (for all layers) and composed by A-Lipschitz activation functions. Let T^l be the transformations of f, where its entries $w^l_{i,j}$ are uniformly bounded by K. Then the magnitude of the Fourier coefficient $|\tilde{f}_W(k)|$ is bounded by*

$$|\tilde{f}(k)| \leq \frac{VA^LN^LK^L}{||k||}\,.$$

Proof. We use again (5.1.7), now applying the result for a general A - Lipchitz activation function, giving

$$L_f \leq \prod_{l=1}^{L} A\sigma\big(T^l\big)\,. \tag{5.1.8}$$

Because each entry of $T^{(l)}$ is bounded, we are able to derive a relationship between K and the spectral norm, as we explain in the following. Here we omit the index l for simplicity.

Defining w_i as the i-th row vector of T, we have $||T(x)||^2 = \sum_i^N \langle\, w_i, x\,\rangle^2$ and $|\langle\, w_i, x\,\rangle| \leq ||w_i||\,||x|| = ||w_i||$ for $||x|| = 1$. Furthermore, the entries of T are uniformly bounded by K implying that $||w_i||^2 = \sum_j^N w_{i,j}^2 \leq NK^2$, giving us $||T^l(x)||^2 \leq \sum_i^N NK^2 = N^2K^2$ for all $x, ||x|| = 1$. Since the linear transformation is bounded and the unit circle is compact we have

$$\sigma(T^l) = \max_{||x||=1} ||T^l(x)|| \leq NK\,.$$

Using the above inequality together with (5.1.8) we conclude that $L_f \leq A^LN^LK^L$. Plugging this result to (5.1.2) we get the desired result,

$$|\tilde{f}(k)| \leq \frac{VA^LN^LK^L}{||k||}\,.$$

■

So, not only giving up the ReLU assumption we lost the guaranteed quadratic con-

[10] While we found this theorem to be relevant for the present discussion, it is omitted in the updated version of [28].

vergence, but also Lipschitzness only gives us linear decay. Moreover, notice that in both theorems the bound increases polynomially with the number of neurons and exponentially with the number of layers, suggesting that depth plays an important role in fitting high frequencies. This shall be discussed in section 5.2.2.

5.1.2 The Spectral Bias

Comprising what we have viewed so far, the main theorem in [28] conveys the following interpretation: the space of ReLU neural networks have a natural struggle to fit high frequency signals. According to (5.1.6), as we increase the frequency, the magnitude of the corresponding Fourier's coefficients decrease with $||k||^{-\Delta_k^{(P)}-1}$. This would imply that if, for instance, high frequency noise is present in the dataset, the optimization would be regularized, biased to choose a predictor that would be incapable to fit it. This confirms what was alluded previously in the preamble of this chapter: by choosing the space of ReLU neural networks, we are, in some aspect, favoring simpler functions.

How is this behaviour depicted in the optimization? Suppose gradient descent is the optimizer of choice for training. If the optimization is initialized close to zero, which is usually the case, the learning process would start in a region of low spectral norm, yielding a strong upper bound for higher frequency components, as is pointed out in (5.1.6). As the weights are updated, the gradient will consider these simpler networks first before advancing farther into the space. If the high frequency present in the dataset is just noise, the SGD would stop in a region of low spectral norm. If the rule to be learned has relevant harmonics of higher frequency, the gradient will be guided by the loss function to leave the region close to the origin due to underfitting. Naturally, this regularization only works if training is initialized close to zero, further supporting this heuristic.

To observe empirically this behaviour, experiments are made in [28]. The details are as follows.

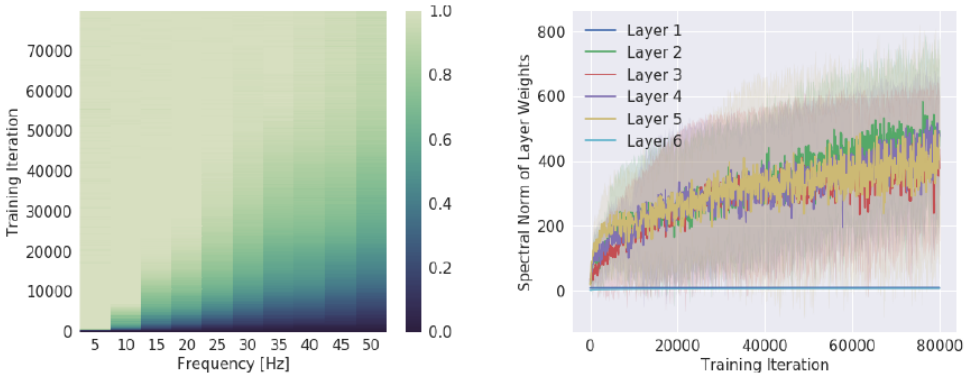
Experiment 1

Two target functions are considered, $\lambda^{(1)}, \lambda^{(2)} : [0,1] \to \mathbb{R}$

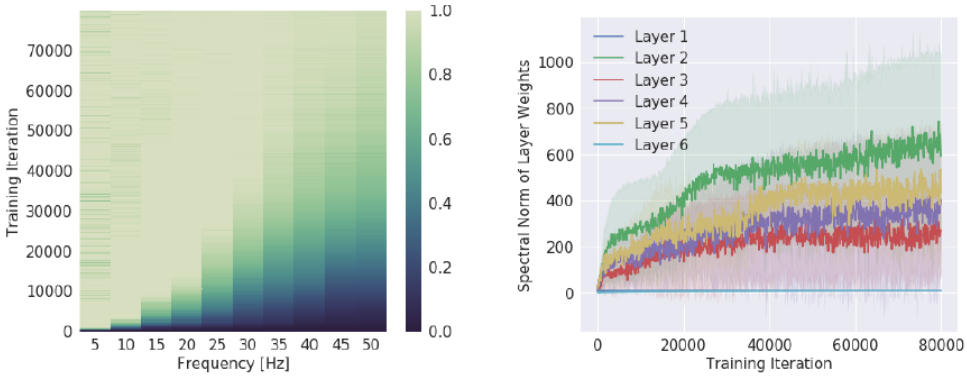
$$\lambda^{(1)}(z) = \sum_{i=1}^{10} A_i^{(1)} sin(2\pi z k_i + \phi_i) \quad and \quad \lambda^{(2)}(z) = \sum_{i=1}^{10} A_i^{(2)} sin(2\pi z k_i + \phi_i)\,.$$

where $A_i^{(1)} = 1$ for all i, $A_i^{(2)} = 0.1 \times i$, $\phi_i \sim U(0, 2\pi)$ and $k_i = 5i$.
The architecture chosen to learn both these oracles is a 6 layered ReLU network with 256 neurons in all layers. Both targets were sampled on 200 uniformly spaced points in $[0, 1]$ and

trained for 80000 steps of full batch gradient descent with Adam [22]. In order to monitor the magnitude of their Fourier transform, here denoted by $|\tilde{f}(k_i)|$, the neural network is evaluated on the dataset every 100 steps in the training process. To further confirm the conjectures regarding ReLU networks and how the gradient descent should fit low frequencies, the spectral norm of the parameters is plotted against the training iterations. Gradient Descent was chosen instead of the more common SGD so that the to be investigated behaviour is not disturbed from the noise caused by randomness.



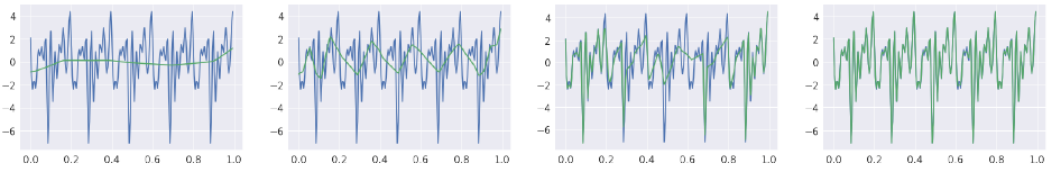
(a) Equal Amplitudes



(b) Increasing Amplitudes

Figure 5.2: ***[Image taken from [28]]*** *(Left side) Plot of the training steps (y axis) against the frequencies (x-axis), where for each* $i = 1, 2, \cdots, 10$ $\frac{|\tilde{f}(k_i)|}{A_i}$ *is measured, here illustrated by the color bar, clipped at* 0 *and* 1.
(Right side) Plot of the spectral norm (y axis), measured by the power iteration algorithm, against the training time (x axis), each color corresponding to a layer. Both $\frac{|\tilde{f}(k_i)|}{A_i}$ *and the spectral norm are averaged over ten runs with different phases* ϕ_i*. Figure set a) is according to* $\lambda^{(1)}$ *and Figure set b) is according to* $\lambda^{(2)}$*. We see that in both cases the learning went very similarly: the lower frequencies were learned first and the spectral norm increased so the higher frequencies would also be fitted, as presumed.*

The theorem states that the harmonics amplitudes increase with $\mathcal{O}\left(||k||^{-\Delta_k^{(P)}-1}\right)$ and thus will not be able to fit anything whose amplitude decays slower, unless the weights are updated accordingly. Together with the fact that initialization was close to zero, this makes us expect an increase in the spectral norm during the optimization in order to reduce the training error. Indeed, as observed in the right hand side of Figures 5.2 and 5.3, that is



(a) Iteration 100. (b) Iteration 1000. (c) Iteration 10000. (d) Iteration 80000.

Figure 5.3: *[**Image taken from [28]**] Sequential plots showing the goodness of fit of the target function $\lambda^{(1)}$ (blue) against the learning function (green).*

exactly what happens.

It is important to note that these dynamics does not result of a possible bias induced by the loss function weighting more in errors made higher frequencies. Let x_i be a point in our dataset S, then the unitarity[11] of the Fourier transform guarantees that

$$\sum_{i=1}^{N}(f(x_i) - g(x_i))^2 = \sum_{i=1}^{N}(\tilde{f}(k_i) - \tilde{g}(k_i))^2 \ .$$

That is, since the error is gauged according to the discrete inner product induced by S, the error's measure is preserved in the spectral domain.

Another way to intuitively understand the spectral decay is to think of it in terms of volume in the parametric space. Let Θ_k be the set of parameters from which we sample the weights of f. Given ϵ and k, we define

$$\Theta_k(\epsilon) \triangleq \{\, \theta \in \Theta \ / \ \exists k' \geq k \ \textit{such that} \ |\tilde{f}_\theta(k')| \geq \epsilon \,\}\,.$$

Then, the relative volume inherits the spectral decay:

$$\frac{Vol(\Theta_k(\epsilon))}{Vol(\Theta)} = \mathcal{O}\Big(||k||^{-\Delta_k^{(P)}-1}\Big)\,.$$

That is, given a certain frequency k and a magnitude benchmark ϵ, the density of weights that generates a f that admits a higher frequency with a higher magnitude follows the spectral decay.
The demonstration is intuitive and may be found in Appendix D of [28].

[11] Unitary transformations are isometries, and so the inner product is preserved.

5.2 Analytical learning and the Fourier Domain

The spectral bias shown in the last section opens a new interpretation for regularization in neural networks, bringing to light special properties of ReLU architectures. It shows that these kinds of networks have a natural delay to reach farther in the parametric space due to a high spectral decay. However, not much is said in [28] on how it relates with the generalization gap or with the overparametrization paradox, or if it relates at all. Propelled by these considerations, we consider the possibility of linking it to the Hardy-Krause variation. On section 5.2.1 we study how variation relates to harmonics, and then we demonstrate the formula for $V[f]$ in the ReLU scenario, concluding something similar to the spectral decay for the Hardy-Krause Variation in polynomial splines. Next on section 5.2.2 we present an experiment which suggests that overparametrization acts against the effects of spectral decay. We then argue on how it may, despite this, contribute to reducing the gap.

5.2.1 Spectral Decay and Variation

Intuitively, it makes sense to argue that by choosing a hypothesis space with a "faster than average" spectral decay one contributes to learning, albeit lacking a formal link. On the other hand, by Theorem 3.1.2 we are able to connect the Hardy-Krause variation with the generalization gap, but how the latter might be related with harmonics is not clear.
Since variations might come from any frequency given the right amplitude, it makes sense that spectral decay is just one regularization factor, whereas $V[f]$ has a broader approach. This suggests that there is no way to bound the generalization gap with the conclusions drawn from [28].
One simple way to develop this intuition would be to bound the derivative's norm with a term that follows the spectral decay. Indeed, assuming invertibility of the Fourier Transform for a function g,

$$\partial_{x_{j_1},\dots,x_{j_\ell}} g(x) \;=\; \frac{1}{2\pi}\partial_{x_{j_1},\dots,x_{j_\ell}} \int_{\mathbb{R}^d} \tilde{g}(k)e^{ikx}dk \;=\; \frac{1}{2\pi}\int_{\mathbb{R}^d} i^\ell \prod_{i=1}^{\ell} k_i \tilde{g}(k)e^{ikx}dk \tag{5.2.1}$$

$$\longrightarrow \; \Big|\, \partial_{x_{j_1},\dots,x_{j_\ell}} g(x) \,\Big| \;\leq\; \frac{1}{2\pi}\int_{\mathbb{R}^d} \Big| \prod_{i=1}^{\ell} k_i \tilde{g}(k)e^{ikx}\Big| dk \;\leq\; \frac{1}{2\pi}\int_{\mathbb{R}^d} \prod_{i=1}^{\ell} |k_i|\big|\tilde{g}(k)\big| dk \;.$$

First thing one should notice from the above is that, in order to the inequalities be relevant, $\prod_{i=1}^{\ell} k_i\tilde{g}(k)e^{ikx}$ must be absolutely integrable. In fact, to guarantee that the right hand side is finite for all derivatives, we must assume that $\tilde{g}$ belongs to a Schwartz Space[12]. In this case, the right hand side would be a term that would be reduced as the spectral decay

[12]Informally, a function g belongs to a Schwartz Space if all its derivatives exists in $\mathbb{R}^d$ and all of them decrease faster than any inverse power of x

increases, allowing us to control the derivatives norm. Unfortunately this is not the case of ReLU networks, and even if the order of the spectral decay is equal to $\mathcal{O}\big(||k||^n\big)$ for some n, the right hand side might still be unbounded. In other words, because the derivative's norm is the weighted continuous sum of all harmonics, $|\tilde{g}(k)|$ decreasing with $||k||^n$ is not enough for the integral's convergence.

As to why the elusiveness of the relation between harmonics decay and derivatives, it is not clear. As already stated in this work, regularization may manifest itself on many stages of the learning process. Maybe, by trying to measure the spectral decay as a regularization agent using it as a bound to $V[g]$, we are implicitly making an average of its contributions to regularization over all learning stages, and since $V[g]$ is so general, the bound does not hold. This suggests that the spectral decay only acts on a few steps of learning, optimization being the one where we think lies the core of its role as a regularizer. Were otherwise, it alone would contemplate all cases needed to prevent overfitting and thus be sufficient to guarantee a low generalization gap. But in Analytical Learning we still might have a function that came from a high spectral decay space and has a high variation due to its large weights[13].

Having understood the limits of comparing the spectral bias and the Hardy-Krause variation in the general case, we turn ourselves to the ReLU architecture.

Before anything, we must first demonstrate that given an ReLU network f, then $V[f] < \infty$. We actually go even further, unwinding the relation of its Lipschitz constant with the Hardy-Krause variation.

Lemma 5.2.1. *Let $f : \Omega = [0,1]^d \to \mathbb{R}$ be a ReLU neural network with Lipschitz constant L_f. Then*

$$V[f] \leq d\, 2^{2d-1} L_f \,.$$

Proof. The idea is straightforward: we will first bound that the Vitali's variation, and then the result will follow. Unfortunately, since the derivatives of f do not exist on the frontier of linear regions, we can not use the simplified formula of $V^{(u)}f$ shown in (3.1.4)[14].

With this in mind, we recall the definition of the difference operator Δ and $V^u[f]$ given in (3.1.3):

$$\Delta[f,a,b] \triangleq \sum_{\ell \subseteq \{1,\dots,d\}} (-1)^{|\ell|} f(a^\ell : b^{-\ell}) \,,$$

$$V^{(u)}f \triangleq \sup_{P \in \mathcal{P}_u} \sum_{x \in P} \Big| \Delta[f_u, x, x_+] \Big| \,,$$

[13]Maybe in a classical approach one could presume that this case would only happen with low probability, and so the gap would be low with a certain confidence given a high enough decay rate.

[14]Indeed, in [5] is given an example of a piecewise linear Lipschitz function defined in a compact support which has infinite Hardy-Krause variation. This is not a counter example because in our case we have a finite number of linear regions, whereas in the former this is crucial.

where the restriction of f to $u = \{j_1, \ldots, j_k\}$ evaluated at $x = (x_1, x_2, \ldots, x_d) \in [0,1]^d$ is denoted as

$$f_u(x) \triangleq f(x')$$

for $x'_i = 1$ if $i \notin \{j_1, \cdots, j_k\}$ and $x'_i = x_i$ otherwise.

Now, fixed $u = \{j_1, \ldots, j_k\}$ and x a point in the partition $\mathcal{P}_u$, since $\Delta[f_u, x, x_+]$ is an alternating sum of 2^d terms, we may group them in differences, yielding 2^{d-1} pairs of the form $f_u(x^{\ell_i} : x_+^{-\ell_i}) - f_u(x^{\ell_j} : x_+^{-\ell_j})$. Furthermore[15], $\max_{i,j} ||(x^{\ell_i} : x_+^{-\ell_i}) - (x^{\ell_j} : x_+^{-\ell_j})|| = ||x - x_+||$.

Using these two observations, it follows that

$$\sum_{\ell \subseteq \{1,\ldots,d\}} (-1)^{|\ell|} f_u(x^\ell : x_+^{-\ell}) \leq 2^{d-1} L_f \, ||x - x_+|| \leq 2^{d-1} L_f \, ||x - x_+||_1 \, .$$

Putting together the last inequality with the definition of $V^{(u)} f$,

$$V^{(u)} f \leq 2^{d-1} L_f \sup_{P \in \mathcal{P}_u} \sum_{x \in P} ||x - x_+||_1 = d \, 2^{d-1} L_f \, ,$$

where the last part follows from the fact that $\sum_{x \in P} ||x - x_+||_1 = d$ since P_u is defined as the Cartesian product of d partitions of $[0,1]$.
We have then shown that the Vitali's variation is finite, now we just plug the result in the definition of $V[f]$, yielding

$$V[f] = \sum_{u \subseteq \{1,\ldots,d\}} V^{(u)} f \leq d \, 2^{2d-1} L_f \, .$$

■

What is interesting about this result is its relation with the main idea of this chapter: just like in (5.1), we bounded a term measuring complexity using the Lipschitz constant. Hence, if the weights start close to zero, so will the harmonics amplitude and the variation of f. In the case when the error is high on low frequencies, the weights will be updated and the harmonics will increase along with L_f, and the same would be expected of $V[f]$. Furthermore, it was shown the ReLU has a high spectral decay, which makes the SGD run farther than usual to find something with equal complexity. Likewise, the high order derivatives are null for linear splines, implying that the only source of contribution to the Hardy Krause Variation in the ReLU case are the first order derivatives, making $V[f]$ increase slower. This last remark is the most pertinent since it is the analogue of the spectral decay, and is made formal in the following Corollary.

[15] Here we are using the fact that the hypotenuse maximizes the distance.

Lemma 5.2.2. *Let $p : [-M, M]^d \to \mathbb{R}$ be a polynomial of degree n, and $w_{i,j}$ the coefficient associated with the j-th dimension and i-th degree.*
Hence its Lipschitz constant L_p is bounded by

$$L_p \leq d\,W\left(\sum_{i=1}^{n} i\,M^{i-1}\right),$$

where $W \triangleq \max_{i,j} |w_{i,j}|$.

Proof. The Lipschitz constant will be the gradient of the polynomial. Thus,

$$\left|\frac{\partial p}{\partial x_j}\right| = \left|\sum_{i=1}^{n} i\,w_{i,j}\,x_j^{i-1}\right| \leq \sum_{i=1}^{n} |\,i\,w_{i,j}\,M^{i-1}\,| \leq W\sum_{i=1}^{n} |\,i\,M^{i-1}\,|\,.$$

This implies that

$$L_P \leq d^{\frac{3}{2}}\,W\left(\sum_{i=1}^{n} i\,M^{i-1}\right).$$

■

Corollary 5.2.1. *Let $f_n : \Omega = [0,1]^d \to \mathbb{R}$ be a spline of degree n and $L_f \triangleq \max_\epsilon L_\epsilon$, where L_ϵ is the Lipschitz constant of the polynomial defined in the region P_ϵ. Then*

$$V[f_n] \leq 2^{2d-1}L_f \leq 2^{2d-1}d^{\frac{3}{2}}\,W\left(\sum_{i=1}^{n} i\,M^{i-1}\right)$$

where $W_j \triangleq \max_i w_{i,j}$ and $w_{i,j}$ are the coefficients of the polynomial whose Lipschitz constant is L_f.

Proof. The first inequality is just a corollary of lemma (5.2.1). The second one is derived by simply plugging the formula of the Lipschitz constant given in lemma (5.2.2), considering that the Lipschitz constant of f_n is L_f. ■

Proceeding with a reasoning similar to the spectral decay, for fixed $w_{i,j}$, the above Theorem shows that the updates are more relevant on the weights associated with a higher degree, as if it would have a higher learning rate. Indeed, an increment on the weights represents an increment of the power of n to the Lipschitz constant, and by reducing the degree, just like increasing the order of $|k|$ in (5.1), we diminish its impact, requiring higher weights to reach the same complexity. With this result we demonstrate an analogue of the spectral decay for polynomial spline, now using the Hardy-Krause Variation, displaying further advantages of the ReLU architecture.

Our result is not as general as is the spectral bias, since we only argue the "low variation bias" for linear splines relative to other splines, while in Theorem 5.1.4 it is shown that ReLU architectures are advantageous in comparison to any other Neural Network with a Lipschitz activation function. On the other hand, the link between our bias and the generalization

gap is much more explicit. Furthermore, in (5.2.1) we suggest that these two dynamics are independent, that is, low spectral decay does not guarantee low Hardy-Krause Variation.

5.2.2 Spectral Decay and Overparametrization

In view of what we learned in respect to the Hardy-Krause variation and the spectral decay, how does it all couple to the overparametrization problem? The main benefit of the ReLU architecture being biased towards low frequencies is the safe optimization: in comparison with other activation functions, increments in the spectral norm will correspond to smaller increments in harmonics amplitude. In other words, by choosing such space the step size becomes effectively smaller, inheriting the same advantages to learning.

On the other hand, we saw in theorem Theorem 5.1.4 that the Lipschitz constant of f may be bounded by

$$L_f \leq \prod_{l=1}^{L} A\sigma(T^{(l)}) \leq A^L N^L K^L ,$$

where L is the number of layers and N the number of neurons in each layer.
This means that, by considering a hypothesis space with a complex architecture (high L and N), the algorithm will not have to reach far to achieve high complexity , since the Lipschitz constant will increase exponentially with L, turning small weight increments into the source of leaps of complexity, allowing the rapid increase of capacity.
Thus, the number of parameters and the spectral decay's order have counter acting affects on learning's speed, both of which potentially beneficial and harmful for learning.

Beyond the obvious benefits, training faster has strong implications on the generalization gap, as is shown in [15]. In this paper, under a few assumptions over loss function smoothness, it is shown that SGD is algorithmic stable[16] with a stability coefficient ϵ_{stab} that satisfies

$$\epsilon_{stab} \leq \frac{2C^2}{n} \sum_{t=1}^{T} \alpha_t ,$$

where $|S| = n$, α_t is the step size and C is some constant.

Hence, learning with fewer steps improves generalization[17], since, as discussed in chapter 2, the stability constant bounds the gap. This is the main theorem of [15], and tying it up with previous presented results, we use it together with the following experiment to ground our assertions on the advantages of complex architectures.

[16] In [15] they refer to stability in the sense of Bousquet - Elissef [7], as viewd in chapter 2.
[17] Here we are referring to the classical framework of the generalization gap, and not the Analytical one.

Experiment 2

Our data generating function is $y : \mathbb{R} \to \mathbb{R}$,

$$y(x) = \sum_{i=1}^{4} A_i sin(k_i x),$$

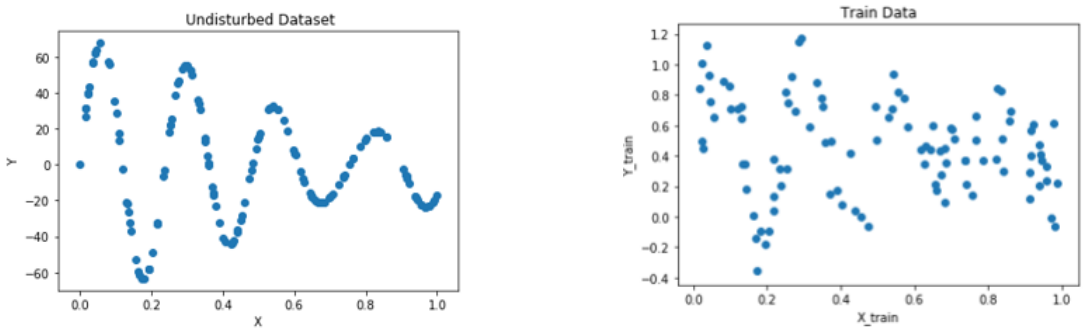


Figure 5.4: *(Left) A plot of* 200 *points generated according y(x),* $x \sim U(0, 1)$*. (Right) A plot of* 150 *disturbed normalized points which will be used for training. Notice that the scale is beyond* 1 *due to the applied noise.*

where $A_i \sim U(10, 20)$, $k_i \sim U(20, 30)$ and $x \in [0, 1]$. 200 points uniformly sampled in $[0, 1]$ were used to generated the dataset S, where 150 were used to train and 50 to validate. In order to force some regularization, after normalizing S, all points in the dataset were disturbed by $\epsilon \sim \mathcal{N}(0, 0.2)$, as is shown in Figure 5.4.
To compare the effect of overparametrization, four ReLU neural networks were chosen with different architectures to learn S. All were trained using full batch Adam for 5000 iterations and initialized using Xavier's initialization, which starts the optimization close to the origin. The performance after training for each model are shown in Figure 5.5.

Model 1: 4 layers, each composed by 60 neurons.
Model 2: 8 layers, each composed by 60 neurons.
Model 3: 4 layers, each composed by 30 neurons.
Model 4: 8 layers, each composed by 30 neurons.

Naturally, the regressors correctly fitted the dataset, since all had architectures large enough to endow the hypothesis space with sufficient capacity to represent it. Also, no overfitting is to be seen even by the most complex network, Model 2, which has 480 neurons, much larger than the size of the entire dataset. Since we are interested in the impact of parametrization on the validation loss and on the learning's speed, we show below the history of the validation loss for each model.

Note how the validation error is rising by the end of training, with the exception of Model 3. It was important for our study to train until overtraining so we could see when each model achieved its best validation error. The logs of Figure 5.6 are:

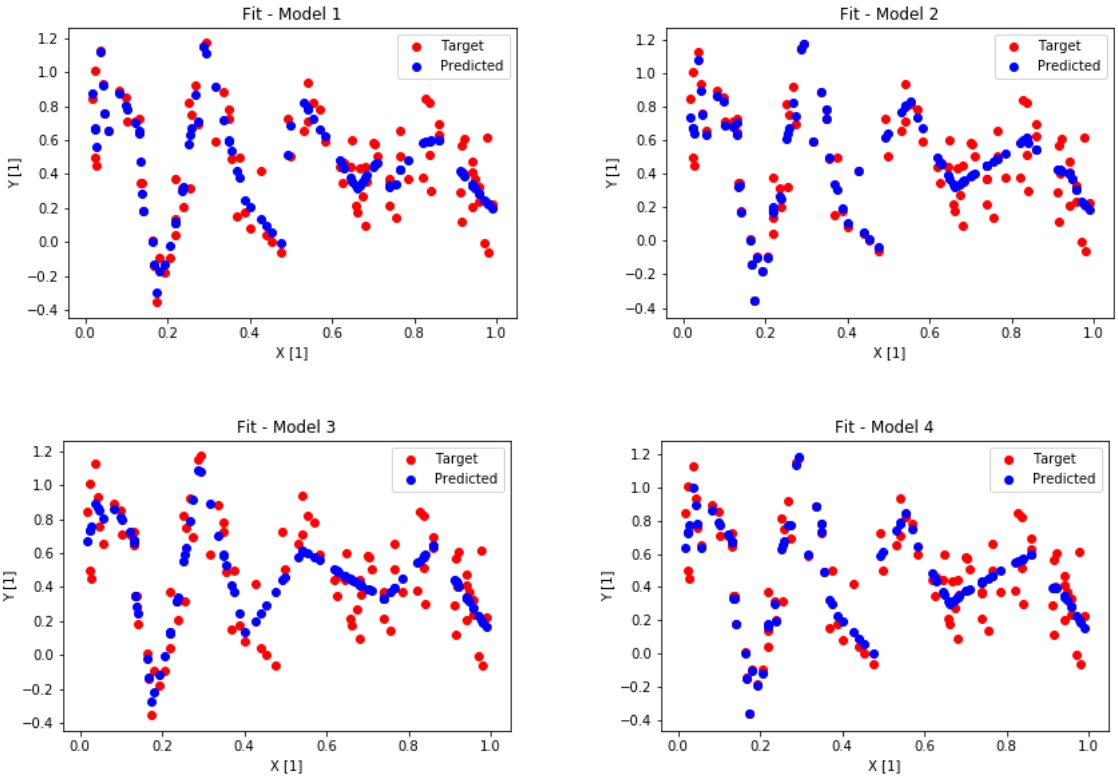


Figure 5.5: *A set of four subfigures:*

Model 1: Best validation loss was 0.063731 achieved at epoch 2810
Model 2: Best validation loss was 0.067901 achieved at epoch 1154
Model 3: Best validation loss was 0.065027 achieved at epoch 4950
Model 4: Best validation loss was 0.061734 achieved at epoch 3095

These results, of course, contribute greatly to our conjectures. We see that the most complex architecture, Model 2, achieved its best validation error the fastest, while the simplest one, Model 3, has yet to learn. We point out that even though Model 2 did not have the best validation error among all other models, the difference is theoretically negligible and can not be related to the architecture.

From now on we will only analyze the differences between models 1 and 2, since the analysis and conclusions are analogous when comparing models 3 and 4.
Observing the differences between the plots in Figure 5.7, its clear that the escalation of the spectral norm saturates much quicker in complex architectures, where in simpler ones it takes much longer to converge. Additionally, in the overparametrized regime, the spectral norm of Model 2 saturated in 3.5, where in its simpler counter part it converged at 5. Moreover, the spectral norm of Model 3, whose training error still have not converged, is still evolving, meaning that the algorithm did not reach a region of enough capacity to fit important frequencies present in the dataset.

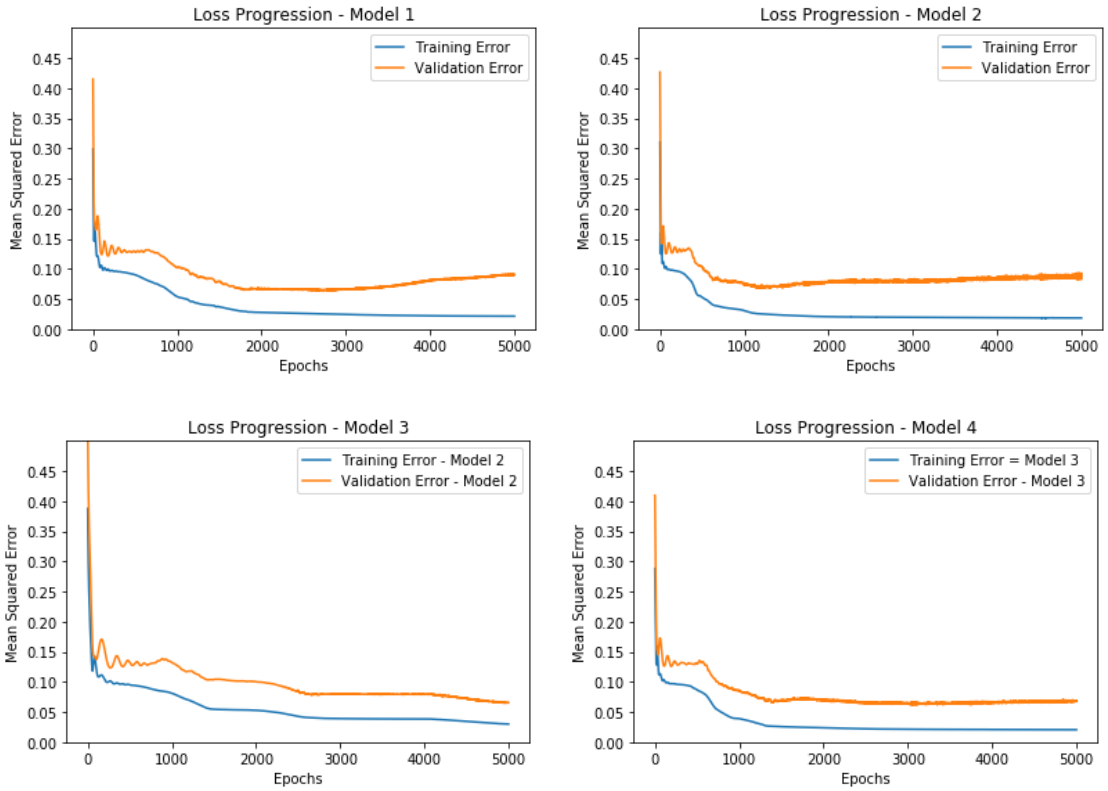


Figure 5.6: *Training and validation error history for the four models. Observe how the training error of the simplest architecture, Model 3, still has not saturated.*

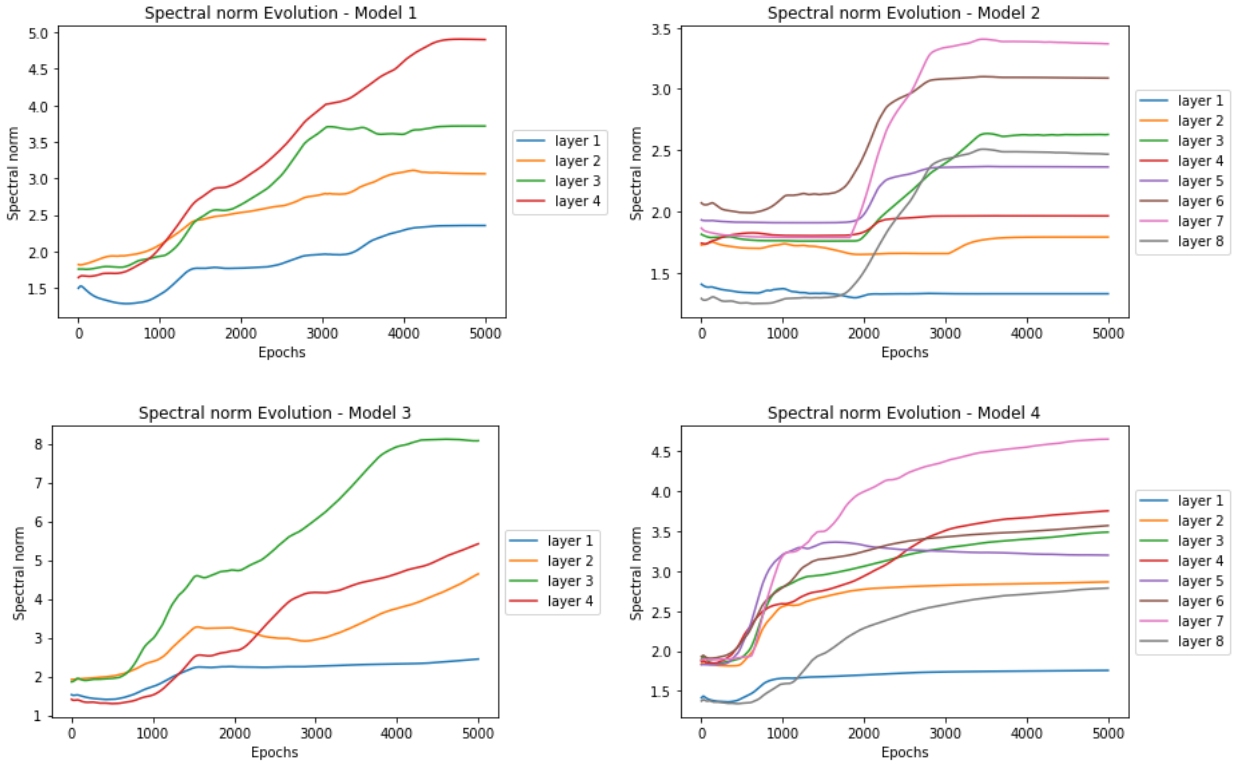


Figure 5.7: *Plots of the spectral norm's evolution of each layer during training. Notice the overall increase and sharp saturation of σ in complex architectures.*

Naturally, the essence of this behaviour lies in (5.1.2). By increasing the number of layers and parameters, the spectral norm required to reach a given complexity is decreased, reducing the numbers of steps needed to reach a given capacity.

This experiment depicts overparametrization as an agent that endows the weight's updates with a larger impact to fit high frequencies, making the algorithm converge faster. Counteractive, as said in the beginning of this section, the spectral decay diminishes the contribution of the spectral norm in the learning of high oscillations, making it hard to reach complex regions.

With this in mind, one might argue that the spectral decay of a neural network like function space, together with the number of layers and vertices, are then just parameters to tune the impact on the capacity caused by weight updates, changing the number of steps required to achieve a certain complexity level. This might suggest that the effects of overparametrization can be simulated by other means: if one wishes to speed up learning, instead of increasing the number of layers or vertices, one could simply choose an activation function with a slower decay, therefore reproducing the same effect.

While its true that both play the same role in contributing for the spectral norm, we claim that there is at least one additional reason to complexify the architecture, at least in the ReLU scenario.

Looking back to Chapter 4, we discussed how the loss function's variation may be seen as a similarity measure between the way the hypothesis and the oracle read information from the input space. This quantity would, in some way, replace the role of stability in the analytical learning context by using the derivative norm.

In the ReLU architecture, since derivatives are constant on linear regions, we can only approximate the oracle's compression on a given polytope by a constant, which might not be enough, since it is committed to the same compression for each region. However, as shown in 5.1.1, by increasing L and N we increase the number of linear regions, endowing a flexible informational assignment in the input space, not only contributing for fitting, but also assigning a different Hardy-krauser variation for each region. This makes possible to reduce the loss function's variation, and consequently, the generalization gap.

Notice that nothing in the optimization induces this process of making the hypothesis's derivatives close to the oracle's, and as said in the end of section 3.2, the optimization of the training error occurs independently.

In the end, our argument is not that, in the ReLU case, by increasing L and N a better generalization gap will be achieved, but that there is an aspect of overparametrization that might contribute to generalization.

As for other activation functions, is not clear how overparametrization affects variation or even if it is required, since their high non linearity, like the sigmoid, might be enough to create a flexible variance.

Chapter 6

Classical Information and Analytical Learning

Taking as a pillar the Analytical Learning paradigm, we have so far exposed how derivatives might be interpreted as the stability quantifier variant when a problem instance is fixed. In this scenario, the objects which define the learning problem are non stochastic, striving for a tighter bound for the generalization error.

We use this chapter to discuss the main result in [37], which relates closely with the overparametrization paradox in neural networks. There is derived a bound which decreases with the number of layers, showing the potential regularization power in some complex architectures. We unwind close similarities between the demonstration of this result and our reasoning based on analytical learning, aiming to further develop the understanding of variance and how it relates to other more mature notions.

6.1 Information flow in Feed Forward Neural Networks

In this section we work under the interpretation of neural networks as encoders and decoders, inducing a Markov process on the outputs of each layer, as is detailed in subsection 2.2.1. Since its a information theory scheme, the context is completely different from that of Analytical Learning and as such we try to keep the notation as concise as possible. In Appendix A we review the basic concepts used in this chapter.

Here the same interpretation from chapter 2 for the stochastic algorithm is used: $A : \mathcal{S} \to \mathcal{H}$ is seen as a conditional distribution $P(H|S)$, H and S are now random variables with values in $\mathcal{S}$ and $\mathcal{H}$ respectively.

Since we are considering a stochastic algorithm, we must account for it in the generalization error as well. In this chapter we use the concept of the expected generalization error:

$$G(D, A) \triangleq \mathbb{E}[L_\mu(H) - L_S(H)] ,$$

where the expectation is taken over the joint distribution $P(H, S) = P(H|S) \times \mu^n$.

Looking at neural networks as a predictor coupled with a trained feature pre-processing step, one might recall that there is no information to be gained by transforming the data, but only lost. Indeed, one of the ideas behind preprocessing features is to filter useless information, namely, noise.

Having this in mind, a general fact is that invertible transformations of random variables preserve information, so it would be natural to regard the preprocessing step as one composed only of non invertible transformations. In the case of a layer in DNN's, a non invertible transformation would be one which involves a singular matrix, since its kernel colapses a whole dimension to zero. Indeed, viewing the network's transformations between layers as a Markov chain, in [2] it is shown that such property is sufficient and the result is known as the Strong Data Processing Inequality (SDPI's).

Theorem 6.1.1. *(Strong Data Processing Inequality) Consider the Markov chain $X \Rightarrow Y \Rightarrow Z$ and the corresponding transition probability $p(Z|Y)$. If there exists z, $p(z) \neq 0$, and more than one y with $p(z|y) \neq 0$, that is, the channel is noisy, then $\exists\, \eta \in [0, 1)$ such that*

$$I(X; Z) \leq \eta I(X; Y) .$$

In other words, for our particular case, if we have a non invertible transformation between Y and Z To apply the above result in DNN's and analyze architectures which compress

information, we must first adapt our framework to Markov Chains. Let $f_W : \mathbb{R}^d \to \mathbb{R}^k$ be a neural network with $L-1$ layers and n_l neurons defined[1] as

$$f_W = W^{(L)} \circ r \circ \cdots r \circ W^{(2)} \circ r \circ W^{(1)} , \tag{6.1.1}$$

where r is any activation function and $W^{(l)} : \mathbb{R}^{n_l - 1} \to \mathbb{R}^{n_l}$ for $l \in \{1, 2, ..., L\}$, $n_0 = d$ and $n_L = k$

Since the input of the network, $Z^0 \triangleq X \in \mathbb{R}^0$ is a random variable, the input of the l-th layer defined by $Z^l \in \mathbb{R}^{n_l}$ may also be seen as random variables.

The previous theorem naturally suggests the following definition:

Definition 6.1.1. *We call the l-th layer of a neural network a contraction layer if the linear transformation $W^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ is such that $rank(W^{(l)}) < n_l$.*

Throughout this section we will be considering only contraction layers, since they have strict information loss, as we show below.

Theorem 6.1.2. [37] *Let $W^{(l)}$ be a linear transformation from $\mathbb{R}^{n_{l-1}}$ to $\mathbb{R}^{n_l}$. If $W^{(l)}$ is such that $rank(W^{(l)}) < n_l$ then the channel $Z^{l-1} \Rightarrow Z^l$ is noisy.*

Proof. By the hypothesis, $\exists\, a \in \mathbb{R}^{n_l}$ such that $a \in ker\{W^{(l)}\}$. Thus, for $z \notin ker\{W^{(l)}\}$,

$$r \circ W^{(l)}(z) = y = r \circ W^{(l)}(z + a) .$$

■

The idea is simple: if the transformation has a non trivial kernel, then the transformation output is invariant according to disruptions in the kernel's dimension.

The last two result imply trivially in the following corollary,

Corollary 6.1.1. *If W^l is a contraction layer then*

$$I(Z^{l-1}, Z^{l+1}) \leq \eta I(Z^{l-1}, Z^l) . \tag{6.1.2}$$

Motivated by this information loss result, we now detail the Markov chain inside a feed forward network. Instead of see it as a predictor as a whole, we shall view f_W as a $L-1$-fold sequential transformation of the inputs, and in the last layer a regressor parametrized by W^L is applied on the transformed inputs Z^{L-1}, as we see in (6.1). To formalize this interpretation we make a slight change of notation, $W^{(L)} = h$ and $W^* \triangleq W - h =$

[1] We note that this is essentially the same definition as the one given in (6.1.1), but because the parametric form is important here, we chose to write in terms of matrices instead of transformations. Also, while in (6.1.1) r was the ReLU , here it can be any activation function.

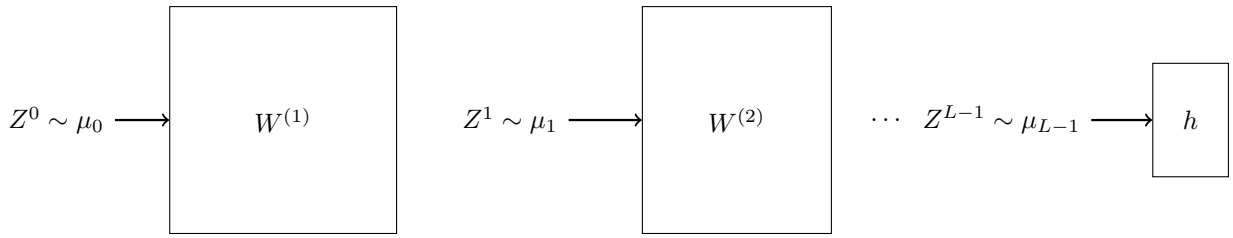


Figure 6.1: *A graphical description of the feed forward network viewed as a Markov chain of transformed features*

$\{W^{(1)}, W^{(2)}, \cdots, W^{(L-1)}\}$.

We now define recursively the variables which define the Markov chain in the network:

$$
\begin{aligned}
&U^0 \triangleq S \sim \mu^n \\
&U^{l-1} \triangleq \sigma \circ W^{(l)}(U^{l-1}) = \{Z_1^{l-1}, Z_2^l, \cdots, Z_{n_{l-1}}^l\} \sim \mu_{l-1}^n \quad , \quad l \in \{1, \ldots, L\}.
\end{aligned}
$$

Assuming that the weights are known, we have then that

$$
\begin{aligned}
h \Rightarrow \{W^{(L-1)}, h\} \Rightarrow \cdots \Rightarrow \{W^{(2)}, \ldots, W^{(L-1)}, h\} \Rightarrow W \\
\Downarrow \\
U^{L-1}|W^* \Leftarrow U^{L-2}|W^* \Leftarrow \cdots \Leftarrow U^1|W^* \Leftarrow S
\end{aligned}
$$

The first part of the chain is trivial since the weights W^* are given. The second part follows because, in order to derive the distribution of U^{L-1}, all that matters is U^{L-2}. Having translated the network behaviour in a Markov Chain, we may use (6.1.2), and so

$$
I(U^{l-1}, U^{l-2}|W^*) \leq \eta_l I(U^{l-1}, U^{l-3}|W^*) \quad \forall l = 1, 2, \ldots, L \, .
$$

The idea behind the above equation is that, in an architecture where all matrices have non trivial kernels, as the features are transformed, information is lost. It would be natural then that, the more layers there are between the inputs and the regressor, the stabler are the transformed inputs with respect to S. This reasoning is the essence of the main result depicted in [37], where algorithmic stability is measured in terms of mutual information, yielding an upper bound for the expected generalization error all while considering an architecture composed by contraction layers.

Theorem 6.1.3. *For a DNN with $L-1$ hidden layers, input S, and parameters W, assume that the loss function $l(W; Z)$ is σ-sub-gaussian with respect to Z for any $W \in \mathcal{W}$. Without loss of generality, let all $L-1$ hidden layers be contraction layers. Then the expected*

generalization error can be upper bounded as

$$G(D, A) \leq \exp\left(-\frac{L-1}{2}\log\frac{1}{\eta}\right)\sqrt{\frac{2\sigma^2}{n}I(S, W)}\,, \tag{6.1.3}$$

where $\eta \leq 1$ *is the geometric mean over all loss factors.*

Proof. For the full demonstration we reference the reader to [37]. However, in 8.1 we show a key result used in the demonstration which relates $G(D, A)$ with mutual information. ■

This result gives us a bound over the generalization error in terms of architecture and algorithmic dependent factors. Firstly we call attention that η is a non increasing function of L, since $\eta_k \leq 1\ \forall k$, and so as the number of layers increases the distance between the training error and the generalization error will decrease, implying that the number of contraction layers can be seen as a regularization factor.
The uniqueness of this bound lies on the fact that each layer behaves as a compression agent. Bounds for generalization error of a general hypothesis class usually are strictly increasing functions of the parameters, which fails to explain the the success of deep neural networks. The above theorem characterizes which kinds of layers yield a better generalization, under mild assumptions over the loss function. This result however does not contradict the bias variance trade off, as is argued in [37], and we present inthe following.

Consider the best predictor among all networks with a fixed architecture, f_{W*} where

$$W^* \triangleq \operatorname*{argmin}_{W\in\mathcal{W}} L_\mu(W)\,.$$

Obviously,

$$\mathbb{E}\left[L^* - L_S(H)\right] \leq \exp\left(-\frac{L}{2}\log\frac{1}{\eta}\right)\sqrt{\frac{2\sigma^2}{n}I(S, W)}\,,$$

where $L^* \triangleq L_D(f_{W*})$.

As discussed before, increasing the amount of contraction layers decreases the capacity of the hypothesis class. This is because the only information used by the regressor to predict is in the last input representation Z^{L-1}, and assuming all layers contract, prediction on Z^{L-1} can't do better than on Z^{L-2} since the latter contains all information used to create the former, implying that L^* is a non decreasing function of L.

If L^* increases as we deepen the network, that is, our hypothesis class is being oversimplified, the training error will as well, meaning that the learner transformations are losing important information and so losing prediction power at the cost of generalization.

But what if L^* stays the same? This would mean that there is still unimportant information to be compressed, and increasing the number of layers will not only bring us closer to the optimal classifier's error (in expectation), but also will decrease the sample complexity in order to achieve it. This last assertion follows from the fact that in the limit $L \to \infty$, information has been completely dissipated and the prediction becomes as good as a coin toss, completely independent of S, meaning that n is important no longer.

6.2 Regularization and algorithmic stability in Deep Neural Networks

As elaborated in chapter 2, many methods known to make the empirical error a better approximation of the true error may be seen as a technique to make the output predictor less dependent of the dataset. In our case, (6.1.3) shows how contraction layers act as a regularization agent, filtering information from the inputs.

Contributing to the work of [37] we provide an additional interpretation of this theorem, now through the lens of Analytical Learning. For simplicity's sake we shall assume throughout this analysis that $n_{l-1} > n_l$ for all l, that is, as we go through the network the dimensions of Z^l will decrease.

What makes this filter-like architecture special is the forcing of non trivial kernels in the linear transformations, since $n_{l-1} > n_l$ is enough for a layer to be a contraction. The kernels are related to the regions where the network is invariant and this is what creates the information loss in 6.1.2. If the inputs of a given layer are disrupted by a vector which belongs to its kernel, then the transformation will ignore the noise. This notion is essential since it shows the relationship between invariance (and therefore the simplicity) of the network with the compression created by each processing step.

Naturally, due to non linearity, a kernel is a property of the transformation and not of the whole network. However, assuming a ReLU activation function $r = \max(0, x)$, the network f will be piecewise linear and we may write it as we did in chapter 5,

$$f(x) = \sum_{\epsilon} w_\epsilon x \;=\; W_\epsilon^{(L)} \cdots W_\epsilon^{(1)} x \quad \forall x \in P_\epsilon \;,$$

where ϵ parametrizes the linear regions and $W_\epsilon^{(l)}$ is the matrix of $T^{(l)}$ with its i-th row set to zero whenever $\epsilon_i^l \;=\; -1$.

In this case, the non linearity of f originated by its activation function makes its invariant space conditioned on the region of the input. This is very important for learning, seeing that the relevance of a given dimension should depend on x. For instance, the pertinence of certain pixels for the prediction of an image depends on other pixels, and only a few can be totally ignored for any given image. Was the network linear, the invariant space would be a vectorial space $ker(f) \;=\; span\{k_j\}$, implying that $x \;=\; \sum_i \alpha v_i + \sum_j \beta_j k_j$ and therefore,

$$f(x) \;=\; f\Big(\sum_i \alpha v_i\Big) + f\Big(\sum_j \beta_j k_j\Big) \;=\; f\Big(\sum_i \alpha v_i\Big) \;.$$

That is, invariant spaces of linear networks are always global. In practice, it would be more realistic to have invariant dimensions associated with only certain regions of x, providing flexibility to discriminate noise.

Back to the piecewise linear case, for a fixed[2] $x \in P_\epsilon$, $u \in ker(w_\epsilon)$ and $\eta \leq 1$ such that $x + \eta u \in P_\epsilon$, we have [3]

$$f(x + \eta u) \;=\; w_\epsilon(x + \eta u) \;=\; w_\epsilon(x) \;=\; f(x) \; .$$

Also, for any linear transformation g_1, g_2,

$$dim(ker(g_2 \circ g_1)) \;=\; dim(ker(g_1)) + dim(range(g_1) \cap ker(g_2)) \; .$$

Therefore, by choosing a ReLU architecture such that $p_{l-1} \leq p_l$ for all l, we are forcing the layers to have kernels, and by increasing the number of layers not only more directions become invariant in a certain region, but also the number of regions grows making the invariance more flexible. Analytically speaking, using the basis of $ker(W_\epsilon)$ as coordinates, the partial derivatives of f according to the kernel's dimensions will be zero in P_ϵ.

All this, as one might expect, ties back to the Hardy-Krause variation and the generalization gap. Supposing the existence of an oracle f^*, these last observations conveys the idea that f should learn the oracle's kernel, so that their invariant spaces coincide, reducing loss function variance and therefore tightening the generalization gap.

From an algorithmic point of view, this matches with the Fourier analysis and the spectral norm dynamic studied in chapter 5. When f is initiallized close to zero, basically all its derivatives are low, since it is piecewise linear, and thus it shall ignore mostly everything. As training progresses, it will start to create dependence in some areas to reduce the training error, and in the end, hopefully, the regions which are still invariant will coincide with the oracle's.

[2]Here we are recovering the notation used in 5 for the linear regions parametrized by ϵ

[3]Notice that $f(x + u) \neq f(x) + f(u)$, since in principle $u \notin P_\epsilon$.

Chapter 7

Conclusion

The broad objective aimed in this work is to, by the usage of Analytical Learning Theory concepts and its paradigm, attempt to contribute to the understanding of generalization in deep feed forward neural networks, using the ReLU activation function example when a concrete case is required.

Our rational is guided by the interpretation of the Hardy-Krause Variation as a measure of information compression quality of the hypothesis, alluding to stability bounds for the generalization error in classical learning theory. After linking it to the partial derivatives of the hypothesis, we recreate the analytical form of the gradient descent, providing a simple way to analyze information compression during training, unwinding the relation between training error optimization and stability. We then used this interpretation to explore the results of [28], from which the main contributions of this work are drawn:

(i) We have shown that the Hardy - Krause Variation behaves analogously to the spectral decay: it increases in order to provide capacity for fitting, and furthermore, its rate of growth is slower in the ReLU architecture when compared to higher degree polynomials. Although the spectral decay phenomena holds for a more general architecture, the Hardy-Krause Variation has a direct connection to the generalization gap.

(ii) Together with (3.1.5), we believe that it may also shed light on the overparametrization paradox in deep networks. Since the number of linear regions increases with the number of layers, not only the capacity to fit, but also the flexibility of the derivatives increases, enabling a possibly lower Hardy-Krause Variation of the loss function, and thus a better a generalization. This behaviour is further endorsed by the fact that gradient descent and its variants test hypothesis with lower derivatives first, turning the event that a highly variant hypothesis is chosen less likely.

Chapter 8

Appendix

8.1 Appendix - Information and Stability

Theorem 8.1.1. [34] *Consider X and Y random variables with a joint distribution $P(X,Y)$ and let $\bar{X}$ be an independent clone of X and $\bar{Y}$ be an independent clone of Y such that $P(\bar{X},\bar{Y}) = P(X)P(Y)$. If $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a σ-subgaussian function with respect to $P(\bar{X},\bar{Y})$ then*

$$|\,\mathbb{E}[f(X,Y)] - \mathbb{E}[f(\bar{X},\bar{Y})]| \,\leq\, \sqrt{2\sigma^2 I(X;Y)}\,, \tag{8.1.1}$$

where the expectations are taken according to the joint probability distributions of the arguments.

Proof. We will make use of Donsker-Varadhan variational representation of the Kullback-Leibler divergent.

$$D(\pi||\rho) \,=\, \sup_F \left\{ \int_\Omega F(x)\, d\pi - \log \int_\Omega e^{F(x)}\, d\rho \right\}.$$

Then, for a fixed f we have:

$$\begin{aligned} &D(P(X,Y)||P(\bar{X},\bar{Y})) \,\geq\, \mathbb{E}_{\,P(X,Y)}[\lambda f(X,Y)] - \log \mathbb{E}_{\,P(\bar{X},\bar{Y})}[e^{\lambda f(\bar{X},\bar{Y})}] \,\geq \\ &\lambda(\mathbb{E}_{\,P(X,Y)}[f(X,Y)] - \mathbb{E}_{\,P(\bar{X},\bar{Y})}[f(\bar{X},\bar{Y})]) - \lambda^2\sigma^2/2\,, \end{aligned}$$

where the second inequality follows from the σ-subgaussian assumption. From now on we shall omit the subscript of $\mathbb{E}$ for simplicity sake.
Since $D(P(X,Y)||P(\bar{X},\bar{Y})) = I(X;Y)$ is always non negative, we have

$$0 \,\geq\, -\lambda^2\sigma^2/2 + \lambda(\mathbb{E}[f(X,Y)] - \mathbb{E}[f(\bar{X},\bar{Y})]) - I(X;Y)\,.$$

The above is a quadratic equation in λ that cannot have two distinct roots, so its determinant must be non positive:

$$(\mathbb{E}[f(X,Y)] - \mathbb{E}[f(\bar{X},\bar{Y})])^2 - 2\sigma^2 I(X;Y) \,\leq\, 0\,,$$

which proves our statement.

■

Now we adapt (8.1.1) to the machine learning framework. Letting $X = S$, $Y = H$ and $f(X,Y) = L_S(H)$, we have that $L_S(H) = \frac{1}{n}\sum_{i=1}^n l(s_i,H)$ is $\sigma\sqrt{n}$-subgaussian if l is σ-subgaussian. Moreover, the expected generalization gap $G(\mu,A)$ can be written as

$$\begin{aligned} &\mathbb{E}_{\,P(S)P(H)}[L_S(H)] - \mathbb{E}_{\,P(S,H)}[L_S(H)] \,=\, \mathbb{E}_{\,P(H)}[L_\mu(H)] \,- \mathbb{E}_{\,P(S,H)}[L_S(H)] \,= \\ &\mathbb{E}_{\,P(S,H)}[L_\mu(H) - L_S(H)] \,=\, G(\mu,A)\,. \end{aligned}$$

Thus we have the following corollary:

Corollary 8.1.1. [34] *If $l(H, S)$ is a σ-subgaussian function under μ then*

$$|G(\mu, A)| \leq \sqrt{2(\sigma^2/n)I(S; H)} \, .$$

The aforementioned result conveys the idea that, the less dependent from your dataset your learning algorithm is, the less it will overfit. Observe that this is consistent with the characterization of $G(D, A)$ in terms of algorithmic stability, since if $I(S; H)$ is small in expectation then minor changes on the dataset will reflect minor changes in your predictions. Note that we are not taking in consideration $L_S(H)$: losing information from S will inevitably increase training error, such is the bias - variance trade off.

Bibliography

[1] `https://www.glassdoor.com/List/Best-Jobs-in-America-LST_KQ0,20.htm`, note = Accessed: 2010-09-30.

[2] Rudolf Ahlswede and Peter Gács. Spreading of sets in product spaces and hypercontraction of the markov operator. *The annals of probability*, pages 925–939, 1976.

[3] Christoph Aistleitner and Josef Dick. Functions of bounded variation, signed measures, and a general koksma-hlawka inequality. *arXiv preprint arXiv:1406.0230*, 2014.

[4] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[5] Kinjal Basu and Art B Owen. Transformations and hardy–krause variation. *SIAM Journal on Numerical Analysis*, 54(3):1946–1966, 2016.

[6] Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[7] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[8] Peter Bühlmann. Bagging, boosting and ensemble methods. In *Handbook of Computational Statistics*, pages 985–1022. Springer, 2012.

[9] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[10] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[11] Luc Devroye and T Wagner. Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Transactions on Information Theory*, 25(2):202–207, 1979.

[12] Luc Devroye and Terry Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604, 1979.

[13] Ricardo Diaz, Quang-Nhat Le, and Sinai Robins. Fourier transforms of polytopes, solid angle sums, and discrete volume. *arXiv preprint arXiv:1602.08593*, 2016.

[14] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[15] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

[16] Glyn Harman. Variations on the koksma-hlawka inequality. *Unif. Distrib. Theory*, 5(1):65–78, 2010.

[17] Edmund Hlawka. Funktionen von beschränkter variatiou in der theorie der gleichverteilung. *Annali di Matematica Pura ed Applicata*, 54(1):325–333, 1961.

[18] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[19] Daniel Jakubovitz, Raja Giryes, and Miguel RD Rodrigues. Generalization error in deep learning. *arXiv preprint arXiv:1808.01174*, 2018.

[20] Kenji Kawaguchi and Yoshua Bengio. Generalization in machine learning via analytical learning theory. *arXiv preprint arXiv:1802.07426*, 2018.

[21] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] JF Koksma. A general theorem from the theory of uniform distribution modulo 1. *Mathematica B (Zutphen)*, 1(7-11):43, 1942.

[24] Erwin Kreyszig. *Introductory functional analysis with applications*, volume 1. wiley New York, 1978.

[25] Wolfgang Maass. Vapnik-chervonenkis dimension of neural nets. *The handbook of brain theory and neural networks*, pages 1000–1003, 1995.

[26] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[27] Sayan Mukherjee, Partha Niyogi, Tomaso Poggio, and Ryan Rifkin. Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, 25(1-3):161–193, 2006.

[28] Nasim Rahaman, Devansh Arpit, Aristide Baratin, Felix Draxler, Min Lin, Fred A Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of deep neural networks. *arXiv preprint arXiv:1806.08734*, 2018.

[29] Daniel Russo and James Zou. How much does your data exploration overfit? controlling bias via information usage. *arXiv preprint arXiv:1511.05219*, 2015.

[30] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[31] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

[32] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[33] V Vapnik. Statistical learning theory. new york: John willey & sons, 1998.

[34] Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2524–2533, 2017.

[35] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[36] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[37] Jingwei Zhang, Tongliang Liu, and Dacheng Tao. An information-theoretic view for deep learning. *arXiv preprint arXiv:1804.09060*, 2018.

[38] Ding-Xuan Zhou. Derivative reproducing properties for kernel methods in learning theory. *Journal of computational and Applied Mathematics*, 220(1-2):456–463, 2008.