Theory and practice in sigmoidal and ReLU Artificial Neural Networks

Ulysse Faure, Florent Draye

June 18, 2022

Project supervisor : Gabriel Peyré

Abstract

In the last ten years, usage of Artificial Neural Networks (ANNs) algorithms has developed considerably, motivated by groundbreaking successes and regular superiority over previous state-of-the-art, especially in fields like computer vision [16], [8] and natural language processing [9] but also in other fields such as genomics [17], [7]. ANNs are functions that aggregate computations from a large quantity of simple units using a fixed process and hierarchy called network structure. The resulting function (essentially consisting in consecutive matrix-vector products separated by a fixed nonlinear function) depends on a large number of parameters, which are optimised by the network when given a labeled data set, during what is called the training phase. Though the basic algorithm and principles of ANNs are relatively easy to understand, mathematical research in this area is recent and overall quite difficult. This paper is a review of theorems and proofs that have helped theory advance from the 1990's onwards, notably from [4],[5], and [14], coupled with numerical analysis on the MNIST data set. It explores performance of different network structures (fully connected, sparse, convolutional,...), activation functions (sigmoid and ReLU), and optimisation behaviour during training phase. The aim is to provide the reader with a global view of the field, to understand contributions to it from the last thirty years, as well as to present precise mathematical statements and their demonstrations in a structured, coherent frame. This paper is a summary of our study of mathematics on ANNs during the Spring 2022 semester.

Contents

Introduction

Artificial Neural Networks (ANNs) form a special kind of algorithms initially developed from inspiration of the way the human or animal brain works, to solve problems like pattern recognition, regression and classification, or decision-making. These networks have been at the heart of notable successes in the last decade in a wide variety of fields, like image and sound recognition, finance, or medicine and genomics. This paper tries to explain the motivation, mathematical theory and implementation of ANNs.

Motivation. Artificial neural networks considered in this essay are designed to give efficient interpolation/approximation of a target function $f : \mathbb{R}^d \to \mathbb{R}^m$ or $f : \mathbb{R}^d \to [0,1]^m$, that is most of the time largely multivariate ($d \gg 1$), typically $1 \leq m \leq 20$ and often known on a relatively small finite set. This might be used to automate pattern recognition or provide convincing regression and classification for decision-making.

Examples. A postal service company might wish to automate the recognition of m-digit postal code on letters. It will feed a photograph of the handwritten code to the network, which should output the postal code. In order to train the network, the company provides it with a number of photographs that are correctly labeled. Here, the target function $\mathbb{N}_{\leq 255}^n \to \mathbb{N}_{\leq 10^m}$ takes the value of each pixel and returns the correct postal code. The target function is known on the labeled photographs and should be "deduced elsewhere", ie. well approximated by the neural network.

Another example is that of a researcher in gender inequalities who wants to know in what proportion men and women appear on a website's photos. Here the input should be one photo with one person on it and the output be a vector $\mu \in [0,1]^2$ with $\mu \cdot \mathbf{1} = 1$ representing probability (as assessed by the network) that this photo features a man or a woman.

Idea. In a simplified model of the brain, neurons are units that receive electrical signals from input neurons they are connected to. If a neuron receives an electrical stimuli larger than a certain threshold, it emits in turn a signal to a certain set a output neurons. The information moves from a set of Start Neurons (their signal might be received directly from a nerve like the optic nerve) to a set of End Neurons (whose activation may correspond to a certain decision or thought). In ANNs, a neuron is represented by a neuron function $x \to \sigma(b \cdot x + c)$. Here $x \in \mathbb{R}^d$ is the signal from the d input neurons, weighted by b which represents the strength of the connection between the input neurons and the receiving neuron, $-c$ is the neuron's activation threshold and σ is called the activation function, nonlinear and (for an important class of these) typically close to 0 for negative inputs and close to 1 for positive ones.

The neurons are organised in successive layers, so that a layer function (from n_1 neurons to n_2 neurons) is

$$\boldsymbol{x} \in \mathbb{R}^{n_1} \to \boldsymbol{\sigma}(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{c}),\ \boldsymbol{B} \in \mathbb{R}^{n_2 \times n_1}, \boldsymbol{c} \in \mathbb{R}^{n_2} \tag{1}$$

where $\boldsymbol{\sigma}(\cdot)$ applies $\sigma(\cdot)$ to each coordinate. In an ANN, the output is the value of the last layer in the network. Thus, the network function is

$$f_\theta(x) = x_H,\ \begin{cases} x_0 = x,\ \theta = (b_{i,j}, c_{i,j})_{1 \le i \le H,\ 1 \le j \le n_i} \\ x_{i+1} : (x_{i+1})_j = \sigma(b_{i,j} \cdot x_i + c_{i,j}) \end{cases} \tag{2}$$

with H layers, n_i neurons in hidden layer i, σ is the activation function and weights $b_{i,j}, c_{i,j}$ are in $\mathbb{R}$.

These ANNs are also called feed-forward neural networks (FFNNs), or multi-layer perceptrons (MLPs) because they are an iteration of Rosenblatt's 1958 perceptron function (eq. 1). Figure 1 shows the standard way of drawing an ANN (in this case, $\mathbb{R}^3 \to \mathbb{R}^2$ with 2 layers (not counting the input layer).

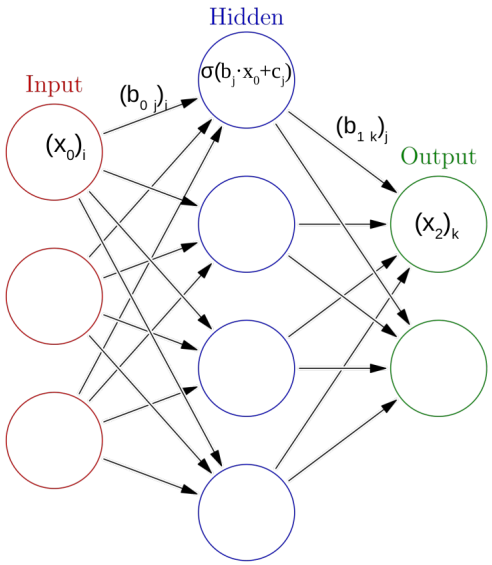


Figure 1: A visualisation of a two-layer perceptron $\mathbb{R}^3 \to \mathbb{R}^2$ with notations as in eq. (18)

Two-layer perceptrons we consider in this essay are a bit special, however, because we impose $m = 1$ and we do not apply the nonlinearity on the output layer. In this case, the network function will can be written more simply as

$$f \in C(\mathbb{R}, \mathbb{R}) : x \to \sum_{i=1}^{n} a_i \sigma(b_i \cdot x + c) \mid n \in \mathbb{N}, (a_i, b_i, c_i) \in \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$$

Approximation and estimation error.

When trying to approach a target function f with an artificial neural network, one will choose network parameters $\theta = (\theta_i)_{i \in I} = (b_{i,j}, c_{i,j})$ given the observations of f: $(X_i, f(X_i))_{i=1}^N$. However, depending on the distribution of the X_i, the chosen parameters θ_{chosen} might not be optimal to minimise the distance between the network function $f_{\theta_{\text{chosen}}}$ and the target function. What's more, even optimal parameters might not be able to approximate f with arbitrary precision. We call

$$\text{AE} = \text{approximation error} := \inf_{\theta} \|f - f_\theta\|$$

if θ is restricted to a compact set, this inf is attained for a certain $\theta_{\min}$. In this case we let

$$\text{EE} = \text{estimation error} := \|f_{\theta_{\text{chosen}}} - f_{\theta_{\min}}\|$$

In this paper, we often aim to minimise $\|f - f_{\theta_{\text{chosen}}}\|$. The immediate bounds

$$\text{AE} \leq \|f - f_{\theta_{\text{chosen}}}\| \leq \text{AE} + \text{EE}$$

follow from the triangular inequality and justify the definition of the approximation and estimation errors. As we will see, multiple theorems concern them. These errors are illustrated in figure 2.

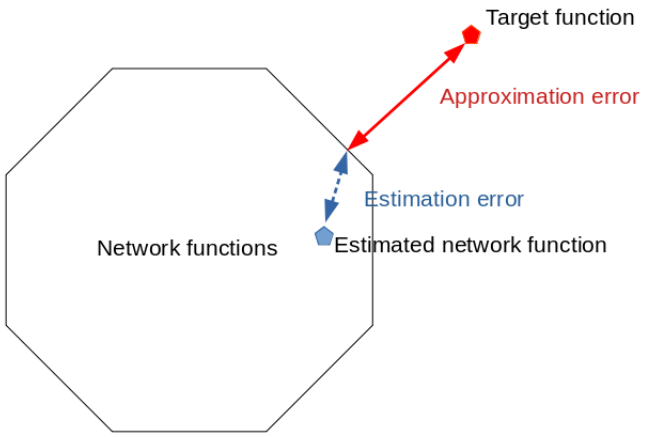


Figure 2: Illustration of approximation and estimation error. Approximation error is the error made when choosing a certain *class* of functions $\{f_\theta \mid \theta \in \Theta\}$ (e.g. 2-layer ANNs, 10-degree polynomials, etc.) to approximate the target function. Estimation error is made when choosing the parameter θ within this class given some data (ie. only partial information on f), as opposed to choosing the best possible parameter $\theta_{\min}$ from Θ that would minimize $\|f - f_{\theta_{\min}}\|$. For ANNs, choosing a good network structure reduces approximation error ; while training that network reduces estimation error.

Overview. In this paper, we first try to give an selective overview of mathematical knowledge in ANNs (we prove most results in the appendix), then confront it with (and present) the common

practices and difficulties in usage of these programs, such as the effectiveness of gradient descent or program architecture choice. Thus, the essay has a few focus points that we divide in sections.

- Section 1 covers 1990's milestones for 2-layer perceptrons theory, notably two results that Andrew Barron proved in consecutive articles [4] and [5]. These theorems demonstrate the potential effectiveness of ANNs in high-dimensional settings by providing an upper bound for the expected total error as a function of the number of neurons in the network.
- Section 2 introduces the ReLU (Rectified Linear Unit) function and compares it with traditional sigmoid activation functions. The section also discusses deep neural networks with multiple hidden layers, which have been used with groundbreaking effectiveness from the 2010's, but for which mathematical theory is notoriously sparse. We present some results from Yarotsky [14] that prove, among other things, that deep ANNs can be asymptotically more efficient than shallow ones.
- Section 3 is the one in which ANNs are faced with real data. It presents the standard algorithms for numerical optimisation (the "training phase") of ANNs and compares their quality in different regimes that depend on the quantity of data provided and the number of neurons in the network. Algorithms are implemented in Python 3 with the PyTorch library.

After all of these, the paper is rounded up with a conclusion and a list of references.

Appendix : Proofs

- Section 4 provides and illustrates proofs of Barron's theorems. These are mathematically rich and involve measure theory, probability theory, Fourier integration and functional analysis.
- Section 5 provides proofs of Yarotsky's results that compare the efficiency of deep and shallow neural networks for approximation. These are very insightful for the question of why the deep regime can be more effective. The fact that they are constructive proofs make them relatively easy to follow and intuitive.
- Section 6 proves that ANN gradient descent makes the empirical risk go to 0 at a linear rate with high probability in overparametrised regimes. This is important because it does not require the parameters to be initialised close to optimality (which seldom happens in reality).

Glossary.

ANNs : Artificial Neural Networks.
S-2LP : Sigmoidal 2 Layer Perceptrons (1 hidden layer ANNs with sigmoidal activation function).
ReLU : Rectified Linear Unit : the function $\max(0, x)$.
MLPs : Multi Layered Perceptrons - often deep neural networks : those with multiple hidden layers.
RFM : Random Feature Model.

Part I
Theory and practice

1 S-2LP Theory

1.1 First results

A common class of functions used for nonlinear activation are the sigmoidal functions. The idea partly stems from neuroscience, because neuron activation in the brain has been found to be close to binary with a certain threshold.

Definition 1.1 (Sigmoidal function)**.** *A sigmoidal function* $\sigma : \mathbb{R} \to \mathbb{R}$ *is a function such that* $\lim_{-\infty} \sigma(x) = 0$ *and* $\lim_{+\infty} \sigma(x) = 1$*, with* $0 \leq \sigma(x) \leq 1$ *on* $\mathbb{R}$*.*

Examples of such functions include heaviside step function $1_{x>0}$, the logistic function $(1 + e^{-x})^{-1}$ and the cumulative distribution function of any probability distribution on $\mathbb{R}$. As we will see, the sigmoidal activation functions have the practical advantage of being bounded as well as the problem (when they are C^1) of having next-to-null gradient far from 0, which can sometimes render the network's learning phase (which is made with gradient descent) inefficient.

In the first part of the present work, we will consider mainly artificial neural networks defined by sigmoidal two-layer perceptrons.

Definition 1.2 (Sigmoidal 2-Layer Perceptrons)**.** *With* σ *a sigmoidal function, we define the sets*

$$\textit{S-2LP} = \left\{ f : x \to \sum_{i=1}^{n} a_i \sigma(b_i \cdot x + c) \middle|\ n \in \mathbb{N}, (a_i, b_i, c_i) \in \mathbb{R} \times \mathbb{R}^d \times \mathbb{R} \right\}$$

$$\textit{S-2LP}(n) = \left\{ f : x \to \sum_{i=1}^{n'} a_i \sigma(b_i \cdot x + c) \middle|\ n' \leq n, (a_i, b_i, c_i) \in \mathbb{R} \times \mathbb{R}^d \times \mathbb{R} \right\}$$

In the following, we note $I^d := [0, 1]^d \subset \mathbb{R}^d$ and $C^0(I^d, \mathbb{R})$ the space of continuous functions $I^d \to \mathbb{R}$.

Theorem 1.3 (Cybenko's density theorem)**.** *[1] Let* σ *be a continuous sigmoidal function. Then*

$$G = \left\{ f : x \to \sum_{i=1}^{n} a_i \sigma(b_i \cdot x + c), (a_i, b_i, c_i) \in \mathbb{R} \times \mathbb{R}^d \times \mathbb{R} \right\} \subset C^0(I^d, \mathbb{R})$$

is dense in $C^0(I^d, \mathbb{R})$ *with the uniform norm* $\|\cdot\|_\infty$*.*

This result shows the potential of approximation of the two-layer perceptron : it can represent any continous function on the d-dimensional unit cube with arbitrary precision. Of course, by scaling, the result also holds for any continuous function with compact support.

1.2 Curse of dimensionality

The difficulty in dealing with problems that involve a high-dimensional input space is often called the *curse of dimensionality*. This is an umbrella term regrouping the problems that arise in interpolation, parametrisation or search for a function's minima in vast spaces like $\mathbb{R}^n$ for $n \gg 1$. For example, a common obstacle in these settings is that available data is almost always sparse, because the number of observations required to ϵ-cover the input space grows like ϵ^{-d}. Another example is the difficulty of finding the "nearest neighbours" when working with a dataset $\mathcal{D}$, ie. for $p \in \mathcal{D}$, finding the k points in $\mathcal{D}$ nearest to p. This problem is useful in nonparametric regression or classification, but its difficulty often scales in $\#\mathcal{D}$ for any evaluation of a new data point.

In the setting of artificial neural networks, one manifestation of the curse of dimensionality takes the form of an exponential lower bound on the number of parameters required to approximate any function from a given regularity class (as can be expected, the exponent is of lesser magnitude when functions are regular).

We denote by $W^{k,\infty}(I^d)$ the Sobolev space of functions on I^d with bounded derivatives up to k-th order, and $F_{k,d}$ the unit ball of this space with the Sobolev norm.

Theorem 1.4 (DeVore lower bound). *Let $M \in \mathbb{N}^*$ and $\eta : \mathbb{R}^M \to C(I^d, \mathbb{R})$. If $A : F_{k,d} \to \mathbb{R}^M$ is continuous and such that $\|f - \eta(A(f))\|_\infty \leq \epsilon$ for all $f \in F_{k,d}$, then $M \geq c_k \epsilon^{-k/n}$, with $c_k \in \mathbb{R}$ constant dependent only on k.*

In this context, $\eta(p)$ is the network function for parameters p and A maps a function in the Sobolev space to parameters that make the network function a good approximation of f. The theorem states that if the parameters for good approximation depend continuously on the target function, at least $O(\epsilon^{-d/k})$ parameters are necessary to uniformly approximate all bounded regular functions (ie. from $W^{k,\infty}(I^d)$) with precision ϵ.

Because of this, Cybenko thought that efficient approximation by two-layer perceptron would be infeasable in practice if the target function's regularity k did not scale like its dimension d : "At this point, we can only say that we suspect quite strongly that the overwhelming majority of approximation problems will require astronomical numbers of terms." ([1]).

A few years later, however, Andrew Barron would make significant contributions to the theory of error of interpolation with sigmoidal functions by bounding both the approximation and the estimation error made by S-2LP networks. Barron started by proving that for a large class of target functions f, not nearly as restrictive as $F_{k,d}$, the optimal sigmoidal approximation with n neurons lies at distance $\leq c_f^{1/2} n^{-1/2}$ from f (with c_f depending only on f), regardless of the dimension of the input space. This means that fixing the target function and increasing the number of neurons yields relatively rapid convergence of the optimal network function to the target function, so long as c_f is not too large. He would then go on to bound S-2LP's average estimation error.

1.3 Barron's error theorems

In an application, it might be more important that certain x from the input space be well approximated compared to some other x'. In the postal code example, images corresponding to real handwritten digits are important to approximate, whereas white noise images, for which the output of the ANN is essentially meaningless, can be considered unimportant. In order to address this, Barron introduces a measure μ on the input space.

Barron's theorem concerns functions with a Fourier representation,

$$f(x) = \int_{\mathbb{R}^d} e^{i\omega \cdot x} \tilde{f}(\omega) d\omega$$

such that its Fourier distribution has finite absolute first moment

$$C_f = \int_{\mathbb{R}^d} \|\omega\| \left\| \tilde{f}(\omega) \right\| d\omega < \infty \tag{3}$$

We call the space of functions f satisfying $C_f < \infty$ the Barron space for sigmoid networks and denote it $\mathcal{B}$.

Theorem 1.5 (Barron approximation theorem). *([4]) For any function $f \in \mathcal{B}$, any sigmoidal function σ, any $r > 0$ and μ a probability measure on $B(0, r) \subseteq \mathbb{R}^d$, there exists a neural network f_n defined by*

weights $(a_i, b_i, c_i) \in \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$ *for* $1 \leq i \leq n$, $f_n : x \to \sum_{i=1}^n a_i \sigma(b_i \cdot x + c_i)$ *such that*

$$\|f - f_n\|_\mu^2 = \int_{B_r} (f(x) - f_n(x))^2 d\mu \leq \frac{c_f}{n}$$

with $c_f = (2rC_f)^2$ *and* $\|a\|_1 = \sum_i |a_i| \leq 2C$.

This means that approximating a function with μ-mean precision ϵ requires at most $O(c_f \epsilon^{-2})$ neurons and thus $O(c_f \epsilon^{-2})$ parameters, contrasting with the $O(\epsilon^{-d/n})$ of the DeVore lower bound. A general idea for the proof of this theorem is that f can be approximated with cosines *via* the Fourier transform, and cosines on bounded domains can be approximated by sigmoid functions.

What's more, parameters (b_i, c_i) can be restricted in growth speed :

Proposition 1.6. *In the conditions of theorem 1.5, for any* $\tau > 0$ *there exists* $f_n \in$ *S-2LP*(n) *with weights* a_i, b_i, c_i *satisfying* $\|a\|_1 \leq 2C$ *and for all* $1 \leq i \leq n$, $|b_i| \leq \tau$ *and* $|c_i| \leq \tau$, *such that*

$$\left\|\bar{f} - f_n\right\|_\mu \leq 2C\left(\frac{1}{\sqrt{n}} + \delta_\tau\right)$$

with $\bar{f}(x) = f(x) - f(0)$,

$$\delta_\tau = \inf_{0<\epsilon\leq 1/2} \left[2\epsilon + \sup_{|z|\geq\epsilon} |\sigma(\tau z) - \mathbf{1}_{z>0}|\right]$$

Thus if $(\tau_n)_{n=1}^\infty$ *is such that*

$$\delta_{\tau_n} \leq \frac{1}{\sqrt{n}} \tag{4}$$

we have

$$\left\|\bar{f} - f_n\right\|_\mu \leq \frac{4C}{\sqrt{n}}.$$

For example, if σ is the step function then $\delta_\tau = 0$; if σ is the logistic sigmoidal function $(1+e^{-z})^{-1}$, taking $\tau \geq \sqrt{n}\log(n)$ yields error in $O(\frac{1}{\sqrt{n}})$. Nevertheless, as Barron acknowledges, the term $+\delta_\tau$ in the formula has the problem of loosening the bound on b_i, c_i ; this means that regression on these parameters would be harder, because the input space is larger. We will see that this term disappears under certain conditions with the ReLU activation function.

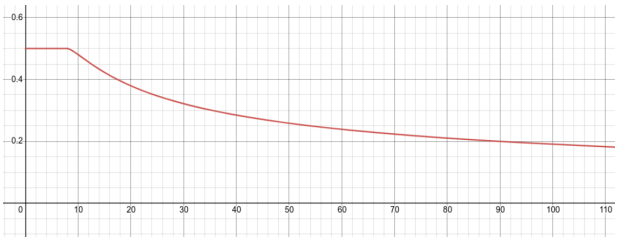


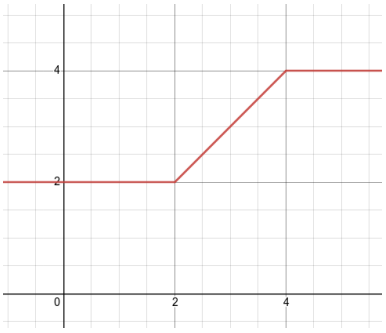
Figure 3: $\delta_\tau(\tau)$ for the logistic sigmoidal function

.

Barron added to this important result an impressive, highly nontrivial theorem in a following paper which incorporated the quantity of given data in the bound for the mean squared distance $d(f_n, f)$ (for bounded f), thus expressing both the approximation and estimation bounds at the same time. In order to state this theorem, we first introduce the clip function.

Definition 1.7 (Clip function)**.** *For $x \leq y \in \mathbb{R}$, we define the clip function*

$$c = c_{x,y} : \ a \to \begin{cases} x & \text{if } a \leq x \\ a & \text{if } a \in [x, y] \\ y & \text{if } a \geq y \end{cases}$$



The clip function for $[x, y] = [2, 4]$

Theorem 1.8 (Barron error theorem)**.** *[5]*

Let X_i $(1 \leq i \leq N)$ be drawn from any distribution μ on $B(0, r) \subset \mathbb{R}^d$, $f \in \mathcal{B}$ with $Im(f) \subseteq [x, y] \subset \mathbb{R}$. There exist constants λ_1, λ_2, independent of f, such that for every N, d, n, there exists an artificial neural network $\hat{f} \in$ S-2LP(n) chosen by minimisation of error on the X_i on a finite set of admissible $\hat{f}$, such that

$$\mathbb{E}_\mu \left\| f - c(\hat{f}) \right\|^2 \leq \lambda_1 \cdot \frac{C_f^2}{n} + \lambda_2 \cdot \frac{nd}{N} \log(N)$$

Where $c(\cdot)$ is the clip function on $[x, y]$. This time σ is assumed to be v_1-Lipschitz with $v_1 > 0$, and approaching its limits in $\pm\infty$ at least polynomially fast.

(Recall that C_f is the first absolute moment of the Fourier transform of f (eq. (3)). A general idea for the proof of this theorem is that in order to find the parameters θ for good approximation, it is sufficient to look on a finite grid on the parameter space, whose tightness depends on the number of neurons but also on the number of observations (this avoids overfitting). This is encapsulated in a variable dependent on f called the Index of Resolvability, whose definition takes inspiration from information theory - the more f is approachable with only a few neurons, the lower the index.

Barron's error theorem shows that there is a known finite-time method for choosing $\hat{f}$, given the $(X_i, f(X_i))$, that yields total error in $O(\frac{C_f^2}{n}) + O(\frac{nd}{N} \log(N))$. In this bound, the first term is the approximation bound given by theorem 1.5 and the second term is a new estimation bound. This result also yields a best growth factor on n (the number of neurons) as a function of C_f and the number of observations N to optimise the total bound : $n \sim C_f \sqrt{\frac{N}{d \log(N)}}$.

The theorem might look like a magic formula to find the best sigmoidal neural network for interpolation, because it gives a theoretical best number of neurons and a finite-time algorithm to find the parameters. However, it suffers two important problems :

- In most problems C_f is unknown, so in these cases the theorem gives no obvious choice for the number of neurons that will yield best interpolation of f for a certain number of observations.
- The algorithm requires minimisation of a loss function over a bounded grid Θ on the parameter space, which grows exponentially in the number of parameters, so the best bound for $\#\Theta$ is $\log \#\Theta = O(nd \log(CndN))$, which means the algorithm is probably impractical for even a small number of neuron in small dimension.

In both articles that produced the theorems, what Barron regularly emphasized is that these bounds showed that two-layer sigmoidal neural networks avoided the curse of dimensionality in terms of accuracy of approximation and estimation, but not in terms of computational complexity, which is why the problem of interpolation remains a difficult one.

1.4 S-2LP attain DeVore's lower bound

Another test of strength for artificial neural networks is the *complexity problem* : how many neurons or parameters are necessary to ϵ-approximate *all* functions in a certain class ? DeVore provides a lower bound on the supposition that parameter choice depend continously on the target function ; in this case, at least $O(\epsilon^{-d/k})$ parameters are necessary for functions in $C^k(\mathbb{R}^d, \mathbb{R})$. In 1996, H.N. Mhaskar demonstrated that this optimal bound is attained by S-2LP under DeVore's condition of continuity and some technical conditions for σ. This result is interesting, but not as strong as Barron's because it is dependent on input dimension and thus weak for high-dimensional problems. Mhaskar showed the following, where $\sigma^{(m)}$ denotes the m-th derivative of σ :

Theorem 1.9 (Mhaskar [6])**.** *Let $k, d, n \geq 1$ be integers and $1 \leq p \leq \infty$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be C^∞ in ± 1, with $\sigma^{(m)}(1) \neq 0$ for all $m \in \mathbb{N}$. There exist $b_i \in \mathbb{R}^d$, for $1 \leq i \leq n$ such that for any $f \in W^{k,p}(\mathbb{R}^d)$, there exist $a_i = a_i(f) \in \mathbb{R}$, $1 \leq i \leq n$, with*

$$\left\| f - \underbrace{\sum_{i=1}^{n} a_i \sigma(b_i \cdot (\cdot) + 1)}_{:=\eta(A(f))} \right\|_{W^{k,p}} \leq cn^{-k/d} \|f\|_{W^{k,p}} .$$

where c depends on k, d, n but is independent of f. What's more, the function $\Psi_n : f \to (a_i)_{i=1}^n$ is continuous. In particular, for any $f \in F_{k,d}$,

$$\|f - \eta(A(f))\|_{W^{k,p}} \leq cn^{-k/d}.$$

Note that this result doesn't even require the b_i to change in function of f and provides the same fixed value 1 for the c_i. The two-layer sigmoidal perceptron can attain a certain approximation bound under continuity assumptions, but that is not where its real interests lie.

1.5 Superiority of S-2LP over approximation by linear subspaces

Dropping the DeVore continuity assumption and the complexity problem for a while, we shall see that the strength of S-2LP resides in the parameters b_i, c_i inside the nonlinearity σ that allow for translation and scaling of the activation function in conjunction with the target function f. Without this there is no avoiding a $1/d$ in the exponent of the error bound, even when restricting target functions to the set of functions in the Barron space with absolute first moment $C_f \leq C$, which we denote $\mathcal{B}_C$.

Definition 1.10. *Let $H_n = span(h_1, .., h_n) \subseteq \mathcal{B}_C$. With μ a probability distribution on the unit cube $B = [0, 1]^d$ and $d(f, g) = \|f - g\|_\mu = (\int_{\mathcal{B}_C} (f(x) - g(x))^2 d\mu)^{1/2}$, we let*

$$W_n = \inf_{h_1,..,h_n} \sup_{f \in \mathcal{B}_C} d(f, H_n) \tag{5}$$

and call W_n the Kolmogorov n-width of $\mathcal{B}_C$. Here

$$d(f, H_n) = \inf_{h \in H_n} d(f, h)$$

In this definition, we find the h_i whose span best covers $\mathcal{B}_C$. W_n thus represents how well $\mathcal{B}_C$ can be approximated by a n-dimensional linear subspace.

Proposition 1.11. *We have*

$$W_n \geq \kappa \frac{C}{d} \left(\frac{1}{n}\right)^{1/d}$$

where $\kappa \geq 1/(8\pi e^{\pi - 1})$ is a constant.

This means that approximating a target function f by linear combinations of fixed functions ϕ_i, $f \approx \sum_{i=1}^{n} a_i \phi_i$ is asymptotically very weak for large d compared to changing the input of a single activation function σ as in Barron's approximation bound (theorem 1.5).

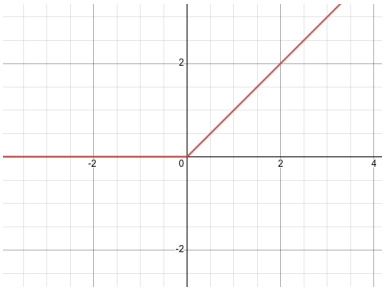
2 ReLU and Deep Neural Networks

2.1 The ReLU activation function

A general numerical problem with using a sigmoid for the activation function is that they tend to saturate to 0 for small values and saturate to 1 for large values. This leads to the concept of vanishing gradient which will be explained in the section on numerical optimization. Moreover, as the size of the neural networks and the capability of hardware increased, it was realized that sigmoidal functions can be hard to train. In order to tackle these problems, multiple different activation functions were tested ; the one that is now most in use is called the ReLU (Rectified Linear Unit) activation function. As we will see, this function efficiently answers both issues presented above, but has the flaw of being unbounded and more difficult to use in theorems like Barron's (therefore with a sparser theoretical background).

The ReLU function is given by

$$ReLU(x) = (x)_+ = \begin{cases} x \text{ if } x \geq 0 \\ 0 \text{ otherwise} \end{cases} \tag{6}$$



The ReLU activation function

Obviously the problem of complexity of evaluation is solved, since it consists in a simple test ≥ 0, for ReLU as for its derivative $\mathbf{1}_{x \geq 0}$ (with convention $ReLU'(0) = 1$). Furthermore, the gradient of ReLU is independent of the magnitude of its input, so gradient descent is efficient whatever the X_i (cf. section 3). This function has only a single point of nonlinearity at 0. One might ask whether ReLU pays its simplicity in limitations in terms of effectiveness in approximation. As we shall see, this is not true.

2.1.1 Theoretical results with the ReLU activation function

Because of its relative young age, the ReLU function has been much less studied than its C^∞ sigmoidal counterparts and research is relatively new. In recent theory, the equivalent of the Barron space for the ReLU activation function is defined as the set of functions $f : \mathbb{R}^d \to \mathbb{R}$ that admit the representation

$$f(x) = \int_\Omega a\sigma(b \cdot x + c)\rho(da, db, dc) \tag{7}$$

where $\Omega = \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$, ρ is a probability distribution on Ω and $\sigma = \max\{0, x\}$, with finite norm

$$\|f\|_{\mathcal{B}_p} = \inf_{\rho \in P_f} \left(\mathbb{E}_\rho \left[|a|^p \left(\|b\|^p + |c|^p\right)\right)^{1/p} \right. \tag{8}$$

where $P_f := \{\rho : f(x) = \int_\Omega a\sigma(b \cdot x + c)\rho(da, db, dc)\}, 1 \leq p \leq \infty$. This can be confusing given the original definition of the Barron space from [4]. It was proved in [25] that for the ReLU activation

function $\|\cdot\|_{\mathcal{B}_p} = \|\cdot\|_{\mathcal{B}_q}, 1 \leq p, q \leq \infty$ and thus we can talk about a unique Barron norm $\|\cdot\|_{\mathcal{B}}$. The name Barron is justified by the following approximation theorem proved in [19].

Theorem 2.1 (Approx RelU)**.** *For any $f \in \mathcal{B}$, μ a probability distribution on $\mathbb{R}^d$, there exists a two-layer neural network $f_n = \sum_{i=1}^n a_i \sigma(b_i \cdot x + c_i)$ such that*

$$\|f - f_n\|_{L^2(\mu)} \leq \frac{3\|f\|_{\mathcal{B}}}{\sqrt{n}} \tag{9}$$

Furthermore, we have

$$\sum_{j=1}^{n} |a_j| (\|b_j\| + |c_j|) \leq 2\|f\|_{\mathcal{B}} \tag{10}$$

Remark. Since for $\sigma(x) = \max(0, x)$, $\sigma(x+1) - \sigma(x)$ is a sigmoidal function, one could use the original Barron theorem for a neural network with the ReLU activation function. However, in the original Barron theory, there is no satisfying bound on the term b_k. In the previous theorem, we have an l_1 bound on the norm of b_k which is essential in practice. This is the reason for defining another Barron space which depends on ReLu activation function.

2.1.2 The Barron space in relation with ReLU

In the last section, we gave another definition of the Barron space $\mathcal{B}$. The objective of this section is to understand what kind of function are in this space. The following theorem is proved in [11].

Theorem 2.2 (Barron and Klusowski (2016))**.** *If*

$$\gamma(f) := \int_{\mathbb{R}^d} \|w\|^2 \left|\hat{f}(w)\right| dw < +\infty \tag{11}$$

where $\hat{f}$ is the Fourier transform of f, then f can be represented as

$$f(x) = \int_{\Omega} a\sigma(b \cdot x + c)\rho(da, db, dc) \tag{12}$$

where $\sigma(x) = \max(0, x)$. Moreover,

$$\|f\|_{\mathcal{B}} \leq 2\gamma(f) + 2\|\Delta f(0)\| + 2|f(0)| \tag{13}$$

With this condition, it is not complicated to prove for instance that smooth positive definite functions ($\sum x_i x_j f(x_i - x_j) \geq 0$ for all $x_1, ..., x_k$) are Barron. Indeed, positive definite functions are characterized as functions that have a Fourier representation $f(x) = \int e^{iw \cdot x} F(dw)$ in terms of a positive real valued measure F, and if f is twice continuously differentiable,

$$\int_{\mathbb{R}^d} \|w\|^2 \left|\hat{f}(w)\right| dw = \int_{\mathbb{R}^d} \|w\|^2 F(dw) = -\sum_{i=1}^{d} \frac{\partial^2 f}{\partial^2 x_i}(0) < +\infty \tag{14}$$

Next, a corollary to the previous theorem is that every sufficiently smooth function is Barron.

Theorem 2.3. *Let $s > \frac{d}{2} + 2$. Then $H^s(\mathbb{R}^d)$ embeds continuously into $\mathcal{B}$.*

The proof of this assertion is sufficiently short to make it in the text.

Proof. Suppose that f has partial derivatives of order less than or equal to $s+1$ that are square-integrable on $\mathbb{R}^d$. In this case using the Cauchy-Schwarz inequality

$$\int |w|^2 |\hat{f}(w)| dw \leq \left(\int (1+|w|^{2k})^{-1} dw \right)^{1/2} \left(\int |w|^4 |\hat{f}(w)|^2 (1+|w|^{2k}) dw \right)^{1/2} \tag{15}$$

where $k = s-1$. Now the integral $\int (1+|w|^{2k})^{-1} dw$ is finite for $2k > d$, and using Parseval's identity,

$$\int |\hat{f}(w)|^2 (|w|^4 + |w|^{2(s+1)}) dw = \int \left| f^{(2)} \right|^2 dw + \int \left| f^{(s+1)} \right|^2 dw < +\infty \tag{16}$$

We conclude by theorem 2.2. □

Remark. Suppose f has partial derivatives of order equal or less to $d/2+2$ that are continuous and we take $r > 0$. We consider a function ρ s-times continuously differentiable such that there exists $r' > r$ with $\rho(z) = 1$ for $\|z\| \leq r$ and $\rho(z) = 0$ for $r' \leq \|z\|$. Then the function $f(x)\rho(x)$ satisfies the conditions of the theorem. This proves that all functions that are smooth enough with respect to their dimension are Barron on any compact set. However, in practice, using derivatives to characterize the approximation bounds is not efficient because the constant C_f can become quite large.

Finally, to understand functions that are not Barron, there is the following theorem proved in [27]. The idea is that a single neuron $\sigma(b \cdot x + c)$ is only non differentiable along the hyperplane $\{b \cdot x + c = 0\}$ and similarly finite two layer neural networks have a singular part that is contained in an union of hyperplane. This can be extended to Barron functions.

Theorem 2.4. *If $f \in \mathcal{B}$, then there exists $(f_i)_{i=1}^{\infty}$ such that $f_i \in C^1\left(\mathbb{R}^d \setminus V_i\right)$ where V_i is a k_i-dimensional affine subspace of $\mathbb{R}^d$ for $0 \leq k_i \leq d-1$, and $f = \sum_{i=1}^{\infty} f_i$.*

In particular, the singular set of f is contained in a countable union of affine spaces. As a consequence, distance functions to curved surfaces are not Barron functions because they have curved singular sets of co-dimension 1. For $d \geq 3$,the function $f(x) = \max\{x_1, ..., x_d\}$ has for singular set the union

$$\bigcup_{i \neq j} \{x_i = x_j = \max\{x_1, ..., x_d\}\} \tag{17}$$

which is not compatible with the previous theorem.

2.2 Deep neural networks

Neural networks have found a surge in popularity in recent years with the successes of *deep* neural network architectures, in opposition to two-layer perceptrons (which therefore are *shallow* networks). Deep neural networks are multi-layered perceptrons with more than one hidden layer ; they can use from two to dozens of hidden layers. The idea is that neurons may be better placed back-to-back rather than all parallel in the same layer. Recall that the network function f_θ is

$$f_\theta(x) = x_H, \begin{cases} x_0 = x, \ \theta = (b_{i,j}, c_{i,j})_{1 \leq i \leq H, \ 1 \leq j \leq n_i} \\ x_{i+1} : (x_{i+1})_j = \sigma(b_{i,j} \cdot x_i + c_{i,j}) \end{cases} \tag{18}$$

with H hidden layers, n_i neurons in hidden layer i, σ is the activation function and weights $b_{i,j}, c_{i,j}$ are in $\mathbb{R}$. In the case that $\#\text{neurons} = \sum_{i=1}^{H} n_i$ is large, ReLU is often preferred to sigmoids for the activation σ because of the simplicity of its evaluation and gradient evaluation.

Because the number of parameters has to remain manageable, we might want to step away from fully connected networks as we've seen so far and allow different *network architectures*. In order to limit $\dim(\theta)$, we set some parameters $b_{i,j}$ to be null ($c_{i,j}$ can also be null) for all network functions. Schematically, this cuts the link between neurons from consecutive layers. Therefore we define :

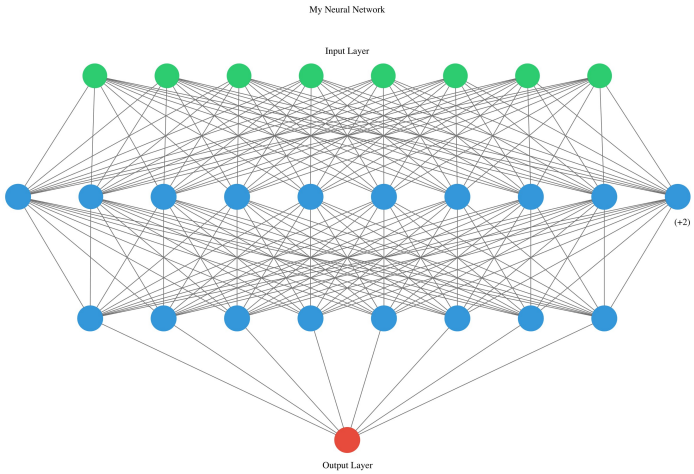


Figure 4: Graphical representation of a deep neural network $\mathbb{R}^8 \to \mathbb{R}$, $H = 3$.

Definition 2.5. *A Network Architecture is the collection A of $H \in \mathbb{N}$, $(n_i)_{i=0}^{H-1} \in \mathbb{N}^h$, and $G \subseteq \mathbb{N}_H \times \mathbb{N}_{\max n_i}$ $(A = H, (n_i), G)$. H is the number of layers, the n_i are the size of the layers and G is the set of parameters fixed to 0. The network functions in this architecture are*

$$\mathcal{A} = \left\{ f_\theta \text{ as in } (18) \text{ with } (i,j) \in G \implies b_{i,j}, c_{i,j} = 0 \right\}$$

We then say that the number of parameters in A is $\sum_{i=0}^{H-1} n_i - 2|G|$, instead of $\sum_i n_i$ for a fully connected network.

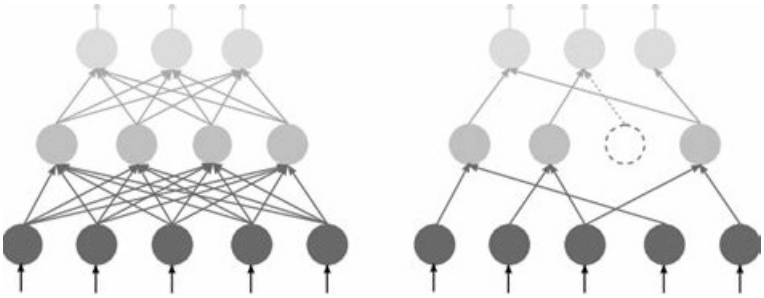


Figure 5: Fully-connected versus sparse architecture in an ANN.

It might seem surprising that deep neural networks should have any superior power of prediction compared to two-layer perceptrons, as we have strong results in the latter case. Why bother to make a network deep when Mhaskar showed that S-2LP attains optimal bound for approximation of the whole $W^{k,\infty}([-1,1])$ space of C^k regular functions on the cube ? The answer is that in this case, practise precedes theory : deep neural networks have had remarkable results in the last ten years compared to two-layer perceptrons ; and machine learning surged with it, helped by the larger availability of data and fast progress in computational speed. This new interest in artificial neural networks has left a hole in its theory for deep networks, emphasised by the difficulty of its study induced by the composition of nonlinearities of the network function.

Nevertheless, some results have been shown that prove the effectiveness of deep ANNs compared to shallow ones. In the following, we introduce a few results from Dmitry Yarotsky in the article *Error bounds for approximations with deep ReLU networks*, published in 2017 [14].

Some results in deep ReLU networks.

Yarotsky considers the approximation of regular functions. Recall DeVore's theorem which states that if parameter choice depends continuously on the target function, then at least $O(\epsilon^{-d/k})$ parameters are required to approximate any C^k function $[-1,1]^d \to \mathbb{R}$ in the Sobolev unit ball. We saw that Mhaskar showed that this bound is attained in S-2LP. First recall our convenient notation :

Definition 2.6. *We let $F_{k,d}$ be the Sobolev unit ball for functions in $W^{k,\infty}([-1,1]^d, \mathbb{R})$, ie.*

$$F_{k,d} = \left\{ f \in W^{k,\infty}([-1,1]^d, \mathbb{R}) \mid \|f\|_{W^{k,\infty}} \leq 1 \right\}$$

Yarotsky uncovers four different results.

Proposition 2.7 (Yarotsky 1)**.** *For any d, k and $\epsilon \in (0,1)$, there exists a ReLU network architecture that is capable of expressing any element of $F_{k,d}$ with precision ϵ (in the* sup *norm), has depth at most $c(\log(1/\epsilon)+1)$ and at most $c\epsilon^{-d/k}(\log(1/\epsilon)+1)$ parameters, with parameters depending continuously on the target function.*

Thus, in DeVore's conditions, ReLU networks graze optimal approximation bound, falling short from a small factor $\log(1/\epsilon)$. However, as we will see, this is lower than fixed depth networks for sufficiently regular functions. To prove this proposition, Yarotsky first approximates the multiplication function $(x,y) \to xy$ with a ReLU network, which allows him to approximate Taylor polynomials for f at different $x \in \mathbb{R}^n$ evenly spaced in $[-1,1]^d$, with spacing dependent on ϵ (but larger than $1/\epsilon$). Then he combines these with partitions of unity to approximate f and bounds the total error made by all of these approximations.

Proposition 2.8 (Yarotsky 2)**.** *If network architecture is allowed to depend on the target function, then all functions in $F_{1,1}$ can be approximated by a depth 6 ReLU network with $O(\frac{1}{\epsilon \log(1/\epsilon)})$ parameters.*

This is lower than DeVore's theoretical lower bound by a factor of $-\ln(\epsilon)$. Here we have a glimpse at the high expressiveness of deep ReLU, but the non-fixed architecture is an obstacle. Reading the proof of this proposition is recommended, because it illustrates and proves the advantages of adding multiple layers in a network in order to store patterns. In a nutshell, the idea is to use one part of the network to approximate f by linear components on intervals of size $1/m$; then another part to approximate $f_2 := f -$ the first part, which is a C^2 function taking value 0 every $1/m$. Because of these constraints, the function f_2 can only take a few different shapes on the m intervals ; these shapes can be coded once in the network and re-used for several of the m intervals, yielding a "recycling" of the information and reducing the total number of neurons needed for ϵ-precision.

Proposition 2.9 (Yarotsky 3)**.** *If one ReLU network architecture can approximate any element of $F_{d,k}$, with parameter choice not necessarily varying continously on the target function, this architecture has at least $O(\epsilon^{-d/2n})$ parameters.*

Dropping the continuity assumption obviously loosens the lower bound. Finally, we have

Proposition 2.10 (Yarotsky 4)**.** *Let $f \in C^2([0,1]^d)$ be a nonlinear function. For any L, a ReLU network of depth L that ϵ-approximates f must have at least $O(\epsilon^{-1/(2(L-2))})$ parameters.*

Here is the lower bound for fixed depth that allows us to prove asymptotic superiority of deep networks when $\epsilon \to 0$, in combination with Yarotsky 1. Indeed, when

$$k > 2d(L-2)$$

then deep neural networks have faster convergence to any $f \in W^{k,\infty}([-1,1]^d)$ $(k \geq 2)$ than fixed L-depth networks by a factor of

$$\log(\frac{1}{\epsilon})\epsilon^{\frac{-1}{2(L-2)}+\frac{d}{k}}.$$

In particular, this condition becomes $k > 2d$ when comparing deep with a single hidden-layer perceptron (Yarotsky counts the input layer in his layer count L). This result is important because it provides some justification that deep neural networks work so well ; much better in some applications than shallow ones. The idea of the proof of proposition 2.10 is that fixed depth networks have linear components that are inevitably larger than some constant a dependent on their depth ; and that thus they cannot approximate a nonlinear function with arbitrary accuracy without adding many neurons.

3 Numerical optimization

In this more applied part of the project, we will study the use of neural networks in the more realistic situation where the training data is of finite size n. Most of the theoretical results cover neural networks with the ReLu activation function.

3.1 Difficulties of S-2LP theory

Strong theoretical results have been shown for sigmoidal two layer perceptron networks in regression and classification problems. As we have seen, Barron showed that this architecture avoids the curse of dimensionality in a certain sense because it can approximate any fixed target function f with precision ϵ with $O(\epsilon^{-2})$ parameters (where the exponent is independent of the dimension of the domain of the target function d). He further showed that the estimation error made when choosing parameters given only N observations of f is in $O(\frac{nd}{N}\log(N))$ where d appears only linearly. Mhaskar showed that S-2LP attain the theoretical lower bound given by DeVore of $O(\epsilon^{-d/k})$ parameters when approximating all functions in $C^k(\mathbb{R}^d, \mathbb{R})$ with Sobolev norm ≤ 1.

However, the theory of S-2LP is often unverified in practice, stemming from two main problems.

- **Best parameter choice.** Finding the best parameters for $\hat{f}$ to minimise $\left\|f - \hat{f}\right\|$ given f (minimising the approximation error) is a NP-hard problem (as proved by Stephen Judd in [3]), which means they can never be found exactly (or with arbitrary precision). When given only N observations of f, with Barron's method, finding parameters that attain the Barron upper bound for total error (including estimation error) requires searching for a minimum in a finite set whose cardinality grows exponentially with d and n (the number of neurons) ; which again is undoable in practice.

- **Regularity of target function**. Barron starts from the hypothesis that the first moment of the Fourier transform of the target function is finite. For the theorem to be of interest, this first moment should also not scale exponentially with the dimension d. Mhaskar, meanwhile, only considers C^k functions. These regularity conditions are imposed on the target function in order to yield good upper bounds for approximation and estimation, but are often difficult to justify in practice. For example, why should a classification function for photographs of cats and dogs be discrete-C^k regular with respect to the pixel values ?

3.2 Problem setup

Given a function f defined on $\mathbb{R}^d$ and a training data set $\{(x_i, f(x_i))_{i=1}^N\}$, x_i independent identically distributed (iid) with distribution μ, the ultimate goal is to find the weights $(a_i, b_i, c_i) \in \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$ for

$1 \leq i \leq n$ defining $f_n \in S-2LP(n)$, that minimize the population risk also known as the generalization error

$$R(\theta) = \frac{1}{2}\mathbb{E}_{x\sim\mu}\left[(f_n(x,\theta) - f(x))^2\right] \tag{19}$$

In practice, the objective is to minimize the

Pour les réseaux à une couche, $\delta > 0$, si $n \geq \mathcal{O}(N^2/\delta)$, alors avec probabilité plus grande que $1-\delta$,

$$R_N(\theta(t)) \leq e^{-tnC} R_N(\theta(0)) \tag{20}$$

$\rightarrow$ une seule couche cachée et $n \gg N$, alors

following empirical risk

$$\hat{R}_N(\mathbf{a},\mathbf{b},\mathbf{c}) = \frac{1}{2N}\sum_{i=1}^{N}(f_n(x_i,\mathbf{a},\mathbf{b},\mathbf{c}) - f(x_i))^2 \tag{21}$$

In general, the data is divided into a training and a test set. The empirical risk $\hat{R}_N$ is equivalent to the training error and is used for training purposes whereas the population risk is approximated using the test error. To simplify notations, we will identify $x \in \mathbb{R}^d$ with $(x,1)^{d+1}$ and $(b,c) \in \mathbb{R}^{d+1}$ as b. In particular, we can write $b \cdot x = b \cdot x + c$ and

$$f_n(x) = \mathbf{a}\sigma(\mathbf{b} \cdot x) \tag{22}$$

3.3 Gradient descent

As we have seen, best parameter choice is impossible in a real-world machine learning problem. Therefore, an imperfect but applicable method must be chosen to find the a_i, b_i, c_i. The standard heuristic is based on gradient descent. The idea is that parameters are initially set to values that don't correspond the desired approximation of f (*eg.* random values) and are gradually changed in order to improve this approximation, simply by taking a step in the opposite direction of the gradient of the risk $\hat{R}_N$. More formally, writing the parameters a_i, b_i, c_i as a single parameter $\theta = \theta_j$ (dependent on the optimisation step j), the **training phase** of the neural network consists in finding θ_J, where J is the number of descent steps we wish to make in optimisation, with

$$\theta_{j+1} = \theta_j - \tau\partial_\theta \hat{R}_N(X,\theta_j) \tag{23}$$

where τ is the learning rate which is generally defined numerically. Call the network function with n neurons f_n, and the signed quality of interpolation at point i Δ_i, so that

$$\Delta_i = f_n(x_i, \mathbf{a}(t), \mathbf{b}(t)) - f(x_i).$$

In this case, gradient descent means, for $1 \leq j \leq n$,

$$a_j(t+1) = a_j(t) - \tau\dot{a}_j(t) \tag{24}$$

$$b_j(t+1) = b_j(t) - \tau\dot{b}_j(t) \tag{25}$$

with

$$\dot{a}_j(t) = \frac{1}{2N}\sum_{i=1}^{N} \Delta_i \sigma(b_j(t)\cdot x_i) \tag{26}$$

$$\dot{b}_j(t) = \frac{1}{2N}\sum_{i=1}^{N} \Delta_i a_j(t)\sigma'(b_j(t)\cdot x_i)x_i \tag{27}$$

For the initialization of the gradient descent, there are different possibilities. First, one could set all the weights $a_j(0), b_j(0)$ to zero but this leads to all neurons learning the same features. The next option is the random initialization. We can use the uniform distribution or the gaussian distribution:

$$a_j(0) \sim \mathcal{N}(0,1), b_j(0) = \mathcal{N}(0, I_d) \tag{28}$$

which is better because it avoids this problem of symmetry. However, as the number of neurons m increases, the range of $\sum_{i=1}^{n} a_i\sigma(b_i\cdot x)$ would increase proportionally. Moreover, as the dimension d of the input increases, the variance of the term $b\cdot x$ would increase too. But with sigmoidal activation functions like the logistic regression $(1+e^{-z})^{-1}$, the gradient changes slowly for large or small values. This can be seen in the expression of the derivatives

$$\dot{b}_j(t) = \frac{1}{2N}\sum_{i=1}^{N} \Delta_i a_j(t) \underbrace{\sigma'(b_j(t)\cdot x_i)}_{\approx 0 \text{ if } |b_j(t)\cdot x_i| \gg 1} x_i \tag{29}$$

This problem is called vanishing gradients. For the ReLu activation function, a problem is that the variance $\sigma(b\cdot x)$ would increase with the number of dimensions. To avoid these issues, there is the Xavier-type initialization

$$a_j(0) \sim \mathcal{N}(0, \beta^2), b_j(0) = \mathcal{N}(0, I_d/d) \tag{30}$$

where $\beta = 1/\sqrt{n}$. There are other types of initilization for the gradient descent. For instance, there is the He initialization

$$a_j(0) \sim \mathcal{N}(0, \beta^2), b_j(0) = \mathcal{N}(0, 2I_d/d) \tag{31}$$

which was found to be best for the ReLu activation function in deep neural networks.

3.4 Under/Over-parameterized regime

The over-parameterized regime in a two-layer neural networks takes place when the number of neurons n in the network is much greater than the data size N and the dimension d. Neural networks are used all over the industry for their ability to fit all training labels. A believed explanation for the efficiency of neural networks is their over-parametrization. Since the loss function of the neural network is highly non-convex, even for two-layer neural networks, it is unclear why gradient descent would find the global minimum. For two-layer neural network, a lot is known in the so-called highly over-parametrized regime. It has been proven in [18] that the system rapidly converges to a global minima at an exponential rate. We define the Gram matrices

$$K^{(a)}_{i,j} = \frac{1}{N}\mathbb{E}_{b\sim\pi_0}\left[\sigma(bx_i)\sigma(bx_j)\right] \tag{32}$$

$$K^{(b)}_{i,j} = \frac{1}{N}\mathbb{E}_{b\sim\pi_0}\left[\sigma'(bx_i)\sigma'(bx_j)\langle x_i, x_j\rangle\right] \tag{33}$$

and their smallest eigenvalues $\lambda_N^{(a)} := \lambda_{min}(K^{(a)}), \lambda_N^{(b)} := \lambda_{min}(K^{(b)})$.

Theorem 3.1. *Let $0 < \delta < 1$. Assume that $\lambda_N^{(a)}, \lambda_N^{(b)} > 0$, and for all i, $\|x_i\| = 1$, that $\mathbb{P}\{a_k(0) = \beta\} = 1/2, \mathbb{P}\{a_k(0) = -\beta\} = 1/2$ with $\beta = \frac{1}{\sqrt{n}}$ and that $b(0)$ is drawn from the uniform distribution over the unit sphere. If $n \gtrsim \min\{\lambda_n^{(a)}, \lambda_N^{(b)}\}^{-4} N^2 \delta^{-1} \ln(N^2/\delta)$, then with probability greater than $1 - \delta$ over the initialization, we have for each step $t \geq 0$*

$$R_N(a(t), b(t)) \leq e^{-tn(\lambda^{(a)} + \beta^2 \lambda^{(b)})} R_N(a(0), b(0)) \tag{34}$$

where

$$R_N(a, b) = \frac{1}{2N} \sum_{i=1}^{N} \Delta_i^2 \tag{35}$$

is the empirical risk.

This result is proved in appendix. Unfortunately, for regression in two-layer neural networks, the over-parameterized regime is also known to be no better than the associated random feature model defined by freezing the terms $(b_j)_{j=1}^n$ in the optimization, so where the approximation function at time t is defined by

$$f_n(a(t), b(0), x) = \sum_{i=1}^{n} a_i(t) \sigma(b_i(0) \cdot x) \tag{36}$$

and the gradient is

$$\dot{a}_j(t) = \frac{1}{2N} \sum_{i=1}^{N} \Delta_i \sigma(b_j(0) \cdot x_i) \tag{37}$$

This implies that in the over-parameterized regime, there is no "implicit regularization" making the algorithm more efficient than a simple random feature model. We have the following result proved in [21] stating that the entire gradient path can be uniformly close to the associated random feature model as the number of neurons increases.

Theorem 3.2. *If $\tilde{a}(t)$ denotes the parameter of the associated random feature model, under the same assumptions as in theorem 3.1,*

$$\sup_{x \in S^{d-1}} |f_n(a(t), b(t), x) - f_n(\tilde{a}(t), b(0), x)| \leq \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) \tag{38}$$

We now turn ourselves to numerical experiments to illustrate the previous theorems. We first consider random samples $(x_i)_{i=1}^{30}$ drawn from the uniform distribution over the set $[-1, 1]^{10}$ and $(y_i)_{i=1}^{30}$ independent samples drawn from the uniform distribution over the set $[0, 1]$. The objective here is to see if a two-layer over-parametrized neural network can fit any sample $(y_i)_{i=1}^{30}$ at an exponential rate. In this setting, there is no test error. We use the Xavier-like initialization,

$$a_j(0) \sim \mathcal{N}(0, \beta^2), b_j(0) = \mathcal{N}(0, I_d/d) \tag{39}$$

where $\beta = 1/\sqrt{n}$. We consider the classical gradient with the rate $\eta = 0.05$. The results in figure 6 illustrate the previous theorems with exponential convergence proportionate to the number of neurons.

In the second experiment, we consider the single neuron target function defined by $\sigma(e_1 \cdot x)$, where $e_1 = (1, 0, .., 0) \in \mathbb{R}^d$. The objective is to display the relation between a two-layer neural network and its corresponding random feature model (RFM). In particular, we want to see if the over-parametrized networks are uniformly close to their RFM. The single target function is perfect to understand the over-parametrized regime because it represents these functions that can be approximated by a small number of neurons compared to n. The setting is same as previously with a step size $\eta = 0.05$. For the

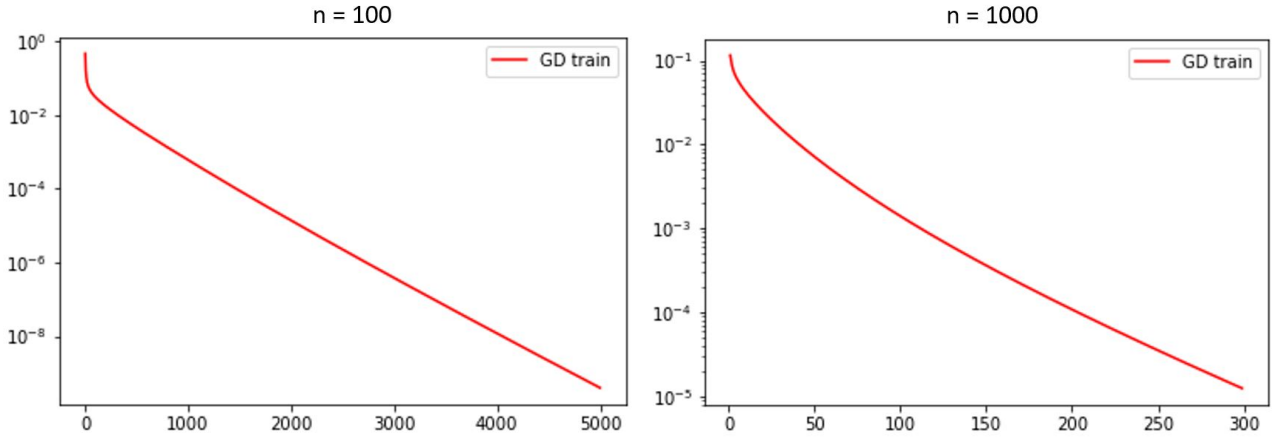


Figure 6: Gradient descent loss, $N = 30$, $d = 10$, with n neurons

test error, we consider a sample of N independent vectors $(\tilde{x}_i)_{i=1}^N$ drawn from the uniform distribution over the set $[-1, 1]^d$. The results in figure 7 are in accordance with the theory. We can see that as the number of neurons n increases, the distance between the train losses of the RFM and the GD decreases. Moreover, we can observe a first phase where the test losses and the train losses have the same behavior. Quickly, a second phase takes place, where the test loss starts to saturate.

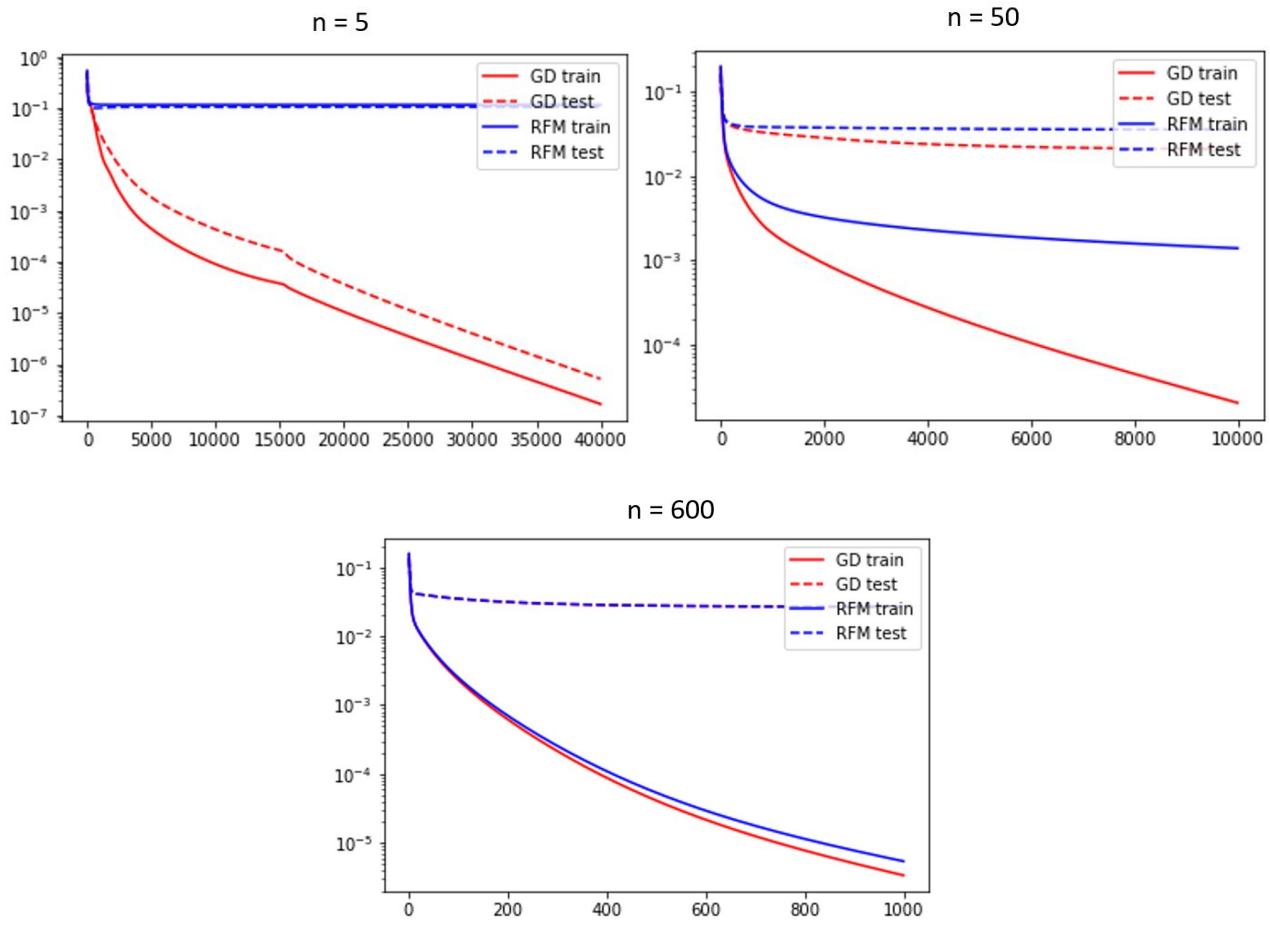


Figure 7: Second experiment: gradient descent of a two-layer neural network and its corresponding random feature model with the parameters $d = 10, N = 50$.

The theory in the under-parameterized and midly-parameterized regime is still weak. We have seen that for two layer neural networks, in the over-parametrized regime, the behavior is the same as the corresponding random feature model. This means that the parameters b_i barely change. Is this behavior also present in other regimes ? We considered the single neuron target function in dimension 20, with 200 data points $(x_i)_{i=1}^{200}$, and with two networks composed of 10 and respectively 30 hidden neurons. First, we can observe that the train loss converges exponentially in both cases which is expected for such a simple target function. For $n = 10$, only two neurons seem to get activated at the beginning of the GD and start contributing more than others that are sent to 0. For $n = 30$, the behavior is similar but less amplified. 4 neurons are activated and the other neurons seem to quicly become stable as in the over-parametrized regime.

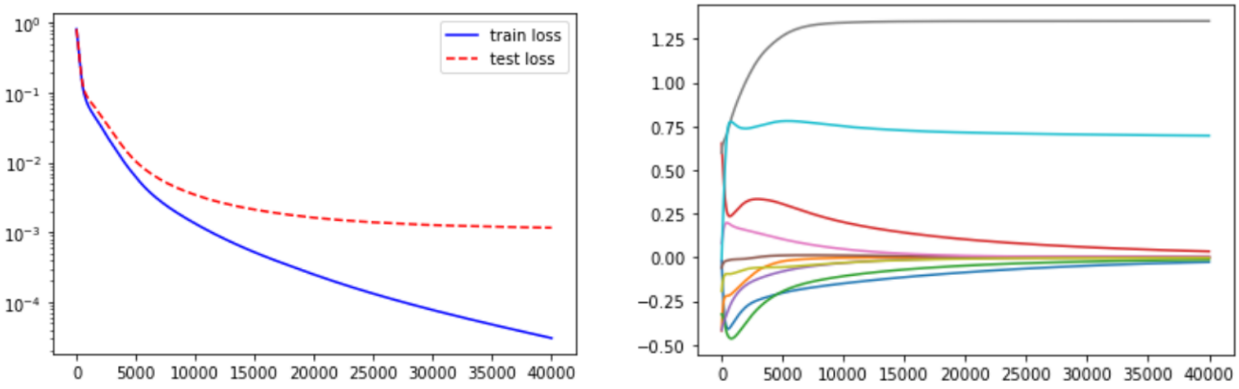


Figure 8: GD for the single neuron target function with the parameters $d = 20, n = 10, N = 200, \eta = 0.01$. On the right, we plot the terms $a_i \|b_i\|$

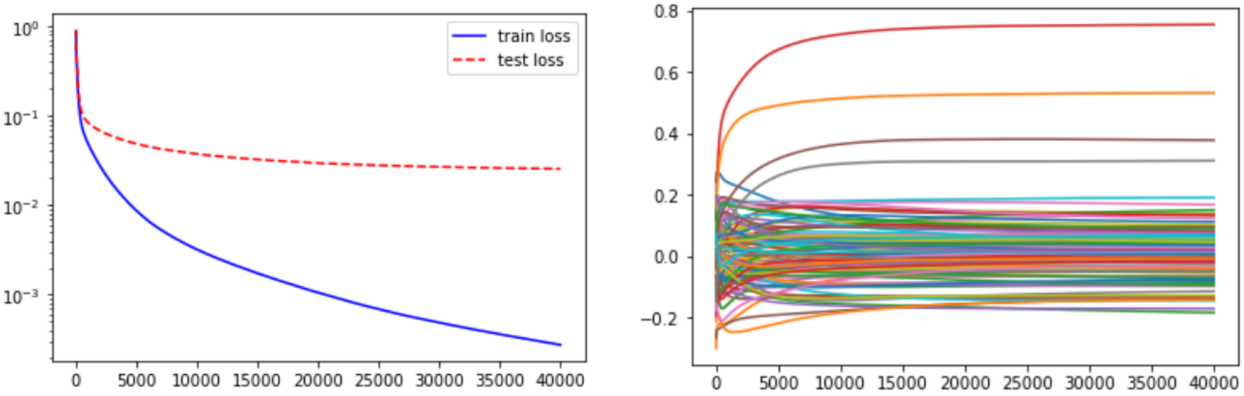


Figure 9: GD for the single neuron target function with the parameters $d = 20, n = 30, N = 200, \eta = 0.01$. On the right, we plot the terms $a_i \|b_i\|$

3.5 Parameter norm evolution under ReLU and sigmoid regimes

In practice, the ReLU activation function is the most popular activation function. The problem with sigmoidal activation functions like the logistic regression $(1 + e^{-z})^{-1}$ is that their gradient is almost

null except around 0, thus rendering gradient descent inefficient. We saw that

$$\partial_\theta R_N(X_i, \theta) = -2 \sum_{i=1}^{N} \partial_\theta f(X_i, \theta) \Delta_i.$$

Given $1 \leq k \leq n$,

$$\begin{cases} \partial_{b_k} \hat{R}_N(X_i, \theta) = -2 \sum_i a_k X_i \sigma'(b_k X_i + c_k) \Delta_i \\ \partial_{c_k} \hat{R}_N(X_i, \theta) = -2 \sum_i a_k \sigma'(b_k X_i + c_k) \Delta_i \end{cases}$$

thus if all $b_k X_i + c_k$ are far from 0, we have $\partial_{b_k} \hat{R}_N(X_i, \theta) \approx \partial_{c_k} \hat{R}_N(X_i, \theta) \approx 0$. In a bad configuration, gradient descent only optimises the a_k, and yields poor final approximation results. Though this result might look specific to this loss function or this simple algorithm given in (23), actually most loss functions and more sophisticated algorithms (like the widely-used stochastic gradient descent) also face this problem. Another difficulty with sigmoidal activation functions is simply the computational cost of their (and their derivative's) evaluation, which can cause the training phase to be too slow to be done in problems that involve a high number of data observations or a high number of neurons. The ReLU activation function does not suffer from vanishing gradients. Its derivative is much more simple to evaluate than the derivative of sigmoidal function which makes it computationally more interesting. Finally, the ReLU activation function sets all negative values to 0. This tends to induce sparsity in the network and make it lighter. Interestingly, this can be observed on our simple two-layer neural networks. For this, we consider the single neuron target function and a network composed of 10 hidden neurons for the ReLu and the sigmoid activation function and we plot the terms $a_i \|b_i\|$ for each iteration of the GD. We can clearly observe the tendency of the ReLU function to shut down some neurons compared to the sigmoid function.

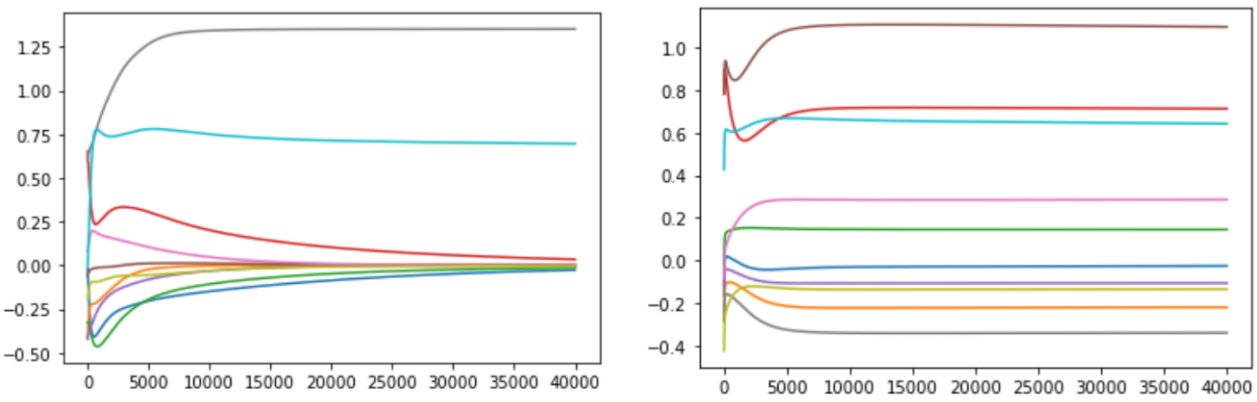


Figure 10: Plot of $a_i \|b_i\|$ for the ReLU activation function (left) and the Sigmoid activation function (right) in a network with parameters $d = 20$ and $n = 10$.

3.6 Experiment on real data

The objective of this section is to illustrate the effectiveness of neural networks on real data. We train a two-layer neural network and deep neural network on the famous data set MNIST using the library PyTorch. This data set consists of images in format 28×28 of the integers from 0 to 9 with their corresponding labels. There are 60000 images in the train set and 10000 images in the test set. The objective here is to approach the function that sends an image in the data set to its corresponding number which is a discrete function.

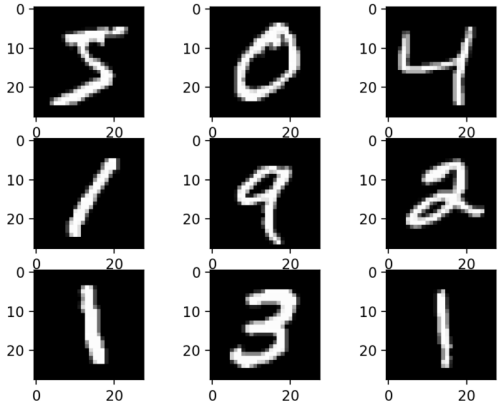


Figure 11: Images from the MNIST data set.

We first consider a two-layer neural network with fully connected hidden layer of size $n = 32$ and the ReLU activation function. The second layer is also fully connected, but has output dimension 10 corresponding to the integers $\{0, 1, ..., 9\}$ drawn in the images. Finally, we apply the Softmax function:

$$\text{Softmax}(x_1, .., x_{10}) = \frac{e^{x_i}}{\sum_{i=1}^{10} e^{x_i}} \tag{40}$$

to convert these 10 numbers into a probability distribution. To optimize this network, we apply the stochastic gradient descent (SGD) with step size $\eta = 0.01$. SGD is a variant of the regular gradient descent where at step, the optimization

$$\theta(t+1) = \theta(t) - \eta \nabla_\theta \left(\sum_{i=1}^{N} (f_n(\theta, x_i) - y_i)^2 \right) \tag{41}$$

is replaced by

$$\theta(t+1) = \theta(t) - \eta \nabla_\theta \left((f_n(\theta, x_{\tilde{i}}) - y_{\tilde{i}})^2 \right) \tag{42}$$

where $\tilde{i}$ is randomly chosen over the set $[1, N]$. In practice, the variant can be computationally more efficient than GD. We use a batch size of 1000 for the train set and the test set, meaning that the images go through the network by blocks of 1000. We use the parameter epochs = 30 which is the number of times that we run through the data set. For the loss function, the natural quadratic loss is not effective. Instead, we use the categorical cross-entropy loss, famous in classification tasks,

$$l(x, y) = -\sum_{i=1}^{10} \log \left(\frac{e^{x_i}}{\sum_{k=1}^{10} e^{x_k}} \right) \tag{43}$$

where x_i are the output from the network. We obtain a precision on the test set of around 93 %. We can also plot the parameters b of the 32 hidden neurons as 28×28 images. We propose to plot the first 9 neurons.

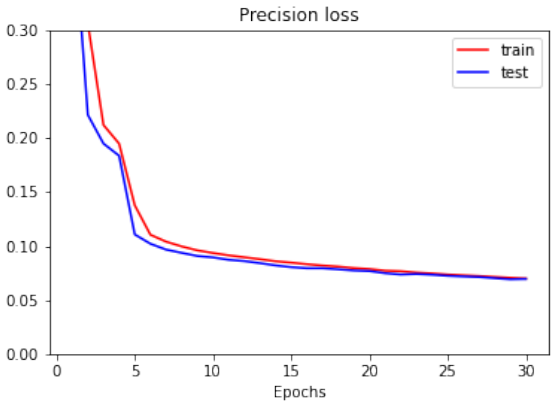


Figure 12: Two-layer neural network with 32 hidden neurons trained on MNIST for classification.

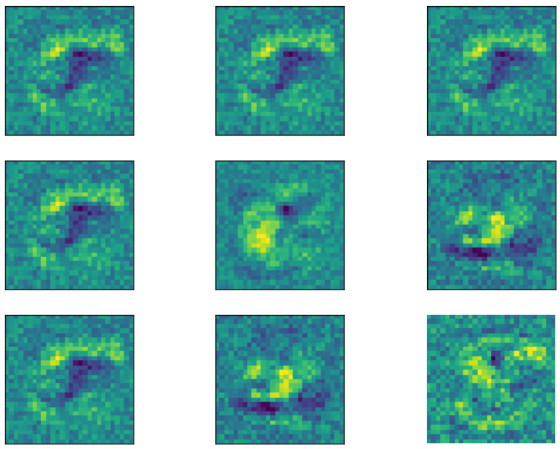


Figure 13: First nine neurons of the network plot as 28×28 images.

To achieve an even better estimation, we propose here to implement a convolutional neural network (CNN). The convolution layer is the builing block of the CNN. The convolution layer receives an image to which it applies a convolution with a kernel of size typically 3×3 or more.

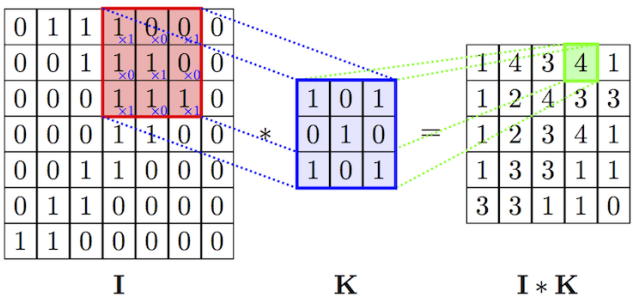


Figure 14: Example of image convolution in 2d. Figure from [13].

A convolution layer of size M applied to N output images, will construct $M * N$ convolutions. This increases quickly the dimension of the network and to avoid computional complications, we generally place a maxpooling layer behind a convolution layer to reduce the dimension of the image. Typical deep neural network for image analysis in the industry are convolutional neural networks. Famous pre-trained network include the VGG16 with the following architecture.

Figure 15: General architecture of VGG-16, convolution kernel size 3·3, pooling size 2·2. Figure from [26].

In our experiment, we considered a network composed of 225034 parameters with the following architecture: $\{Conv, ReLU, MaxPool, Conv, ReLU, MaxPool, Linear, ReLU, Linear, Softmax\}$. For the optimization, we used the stochastic gradient descent (SGD) with stepsize 0.01. We used 10 epochs which took approximately 20 minutes for the networks to be trained on a GPU. The test precision loss reached 97 %, as shown in the next figure. This means that the network guessed the correct handwritten number in 97 % of the images !

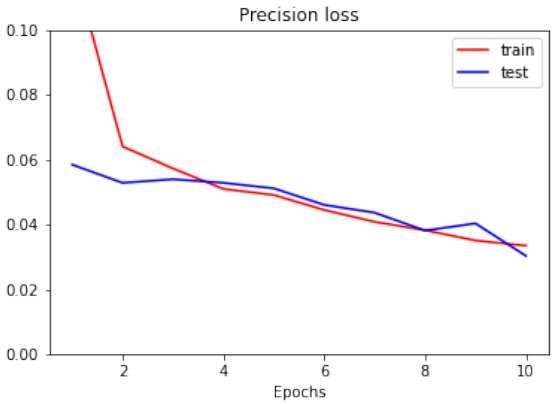


Figure 16: Precision loss for our convolutional neural network on MNIST.

A note on the appendix.

The appendix of this document provides proof to a large part of results that were cited above. Most of these proofs were present in the research papers we cited ; we have adapted them and their conventions to fit in best with our essay. The goal is i) to explain the intuition and key ideas behind each theorem, ii) to illustrate them with relevant graphics, and iii) to provide clarification for some parts that were originally left to the reader or skimmed by the author. We have also deemed it necessary to include a few complete proofs, for example of the key Barron Approximation theorem. Every proof is interesting and relevant, of course, but the least technical and most insightful proofs are those of Barron's approximation theorem (1.5) and Yarotsky's constructive proofs for deep neural networks

(2.7, 2.8, 2.10).

Conclusion

This paper has presented an introduction to the theory and practice in the field of artificial neural networks.

- We have seen and proved the expressive power of the two-layer perceptron both with sigmoid and ReLU activation function (the latter being the standard today), with best mean square approximation error in $O(1/n)$ (with $n = \#\text{neurons}$) when knowing the target function f on the whole of $[-R, R]^d$ and in $O(1/n) + O(nd/N \log(N))$ when given only N observations of f. We have noted that the dimension d only appears in a linear term and therefore ANNs avoid the curse of dimensionality in terms of theoretical expressiveness - but not necessarily in terms of computational complexity. One key condition to these strong results is that the target function is fixed and we vary the number of neurons ; in the reverse case where we want to approximate a whole class of functions like $F_{k,d}$ with a fixed number of neurons, ANNs are optimal (in the conditions of Mhaskar) but do not avoid the curse of dimensionality. This is why choosing the best number of neurons and best network architecture in a real problem is an important challenge.

- We have compared the pros and cons of the Rectified Linear Unit and sigmoidal activation functions such as the logistic sigmoid function. In particular, sigmoids have the advantage of being bounded, but ReLU is faster to compute and does not present the problem of vanishing gradients. Because of this, ReLU has become the canonical activation in deep neural networks. We have proved the convergence, during gradient descent, of ReLU two-layer perceptron parameters to the global empirical-error arg-minimum in the overparametrised regime (large number of neurons for small datasets and dimensions).

- We have explained why strong theoretical results in deep architectures are hard to come by, but also presented some recent results from Yarotsky that provide important insights for their startling effectiveness as opposed to shallow networks. In the proofs, we have constructed some of these deep networks and understood their structure.

- We have tested different ANNs in detail and demonstrated their efficiency on the MNIST dataset, both in the case of 2-layer perceptrons and in the case of convolutional deep neural networks. We have studied the global behaviour of parameter optimisation during the training phase and again highlighted differences between sigmoid and ReLU. In the overparametrised regime, we have validated in practice the theorems that stated that ANNs are efficient but not better than random feature models.

Acknowledgments. We thank our project supervisor Gabriel Peyré for his guidance, advices and comments during the development of this text.

References

[1] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[2] EN Barron and R Jensen. "Semicontinuous viscosity solutions for Hamilton–Jacobi equations with convex Hamiltonians". In: *Communications in Partial Differential Equations* 15.12 (1990), pp. 293–309.

[3] J Stephen Judd. *Neural network design and the complexity of learning*. MIT press, 1990.

[4] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

[5] Andrew R Barron. "Approximation and estimation bounds for artificial neural networks". In: *Machine learning* 14.1 (1994), pp. 115–133.

[6] Hrushikesh N Mhaskar. "Neural networks for optimal approximation of smooth and analytic functions". In: *Neural computation* 8.1 (1996), pp. 164–177.

[7] JIŘÍ VOHRADSKY. "Neural network model of gene expression". In: *the FASEB journal* 15.3 (2001), pp. 846–854.

[8] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. "Deep Neural Networks for Object Detection". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/f7cade80b7cc92b991cf4d2806d6bd78-Paper.pdf.

[9] Jiwei Li et al. "Visualizing and understanding neural models in nlp". In: *arXiv preprint arXiv:1506.01066* (2015).

[10] Matus Telgarsky. "Representation benefits of deep feedforward networks". In: *arXiv preprint arXiv:1509.08101* (2015).

[11] Jason M. Klusowski and Andrew R. Barron. "Risk Bounds for High-dimensional Ridge Function Combinations Including Neural Networks". In: (2016).

[12] Siddharth Krishna Kumar. "On weight initialization in deep neural networks". In: *arXiv preprint arXiv:1704.08863* (2017).

[13] Ihab S. Mohamed. "Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques". In: (Sept. 2017). DOI: 10.13140/RG.2.2.30795.69926.

[14] Dmitry Yarotsky. "Error bounds for approximations with deep ReLU networks". In: *Neural Networks* 94 (2017), pp. 103–114.

[15] Simon S. Du et al. "Gradient Descent Provably Optimizes Over-parameterized Neural Networks". In: (2018). DOI: 10.48550/ARXIV.1810.02054. URL: https://arxiv.org/abs/1810.02054.

[16] Salman Khan et al. "A guide to convolutional neural networks for computer vision". In: *Synthesis Lectures on Computer Vision* 8.1 (2018), pp. 1–207.

[17] Yang Liu et al. "Learning structural motif representations for efficient protein structure search". In: *Bioinformatics* 34.17 (Sept. 2018), pp. i773–i780. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty585. eprint: https://academic.oup.com/bioinformatics/article-pdf/34/17/i773/25702394/bty585.pdf. URL: https://doi.org/10.1093/bioinformatics/bty585.

[18] Simon S. Du et al. "Gradient Descent Provably Optimizes Over-parameterized Neural Networks". In: (2019). arXiv: 1810.02054 [cs.LG].

[19] E Weinan, Chao Ma, and Lei Wu. "Barron spaces and the compositional function spaces for neural network models". In: *arXiv preprint arXiv:1906.08039* (2019).

[20] Difan Zou and Quanquan Gu. "An Improved Analysis of Training Over-parameterized Deep Neural Networks". In: (2019). DOI: 10.48550/ARXIV.1906.04688. URL: https://arxiv.org/abs/1906.04688.

[21] Weinan E, Chao Ma, and Lei Wu. "A comparative analysis of optimization and generalization properties of two-layer neural network and random feature models under gradient descent dynamics." In: *Sci. China Math.* 63.1235–1258 (2020), pp. 115–133. DOI: https://doi.org/10.1007/s11425-019-1628-5.

[22] Chao Ma, Stephan Wojtowytsch, Lei Wu, et al. "Towards a Mathematical Understanding of Neural Network-Based Machine Learning: what we know and what we don't". In: *arXiv preprint arXiv:2009.10713* (2020).

[23] Chao Ma, Lei Wu, et al. "The quenching-activation behavior of the gradient descent dynamics for two-layer neural network models". In: *arXiv preprint arXiv:2006.14450* (2020).

[24] Tomasz Szandała. "Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks". In: (Oct. 2020).

[25] Weinan E, Chao Ma, and Lei Wu. "The Barron Space and the Flow-induced Function Spaces for Neural Network Models". In: (2021). arXiv: 1906.08039 [cs.LG].

[26] Yumei Gao, Xiaobing Kang, and Yajun Chen. "A robust video zero-watermarking based on deep convolutional neural network and self-organizing map in polar complex exponential transform domain". In: *Multimedia Tools and Applications* 80 (Feb. 2021), pp. 1–21. DOI: 10.1007/s11042-020-09904-4.

[27] E Weinan and Stephan Wojtowytsch. "Representation formulas and pointwise properties for barron functions". In: *Calculus of Variations and Partial Differential Equations* 61.2 (2022), p. 46.

Part II
Appendix : Proofs

4 Proofs of section 2 (Sigmoid ANNs)

4.1 Theorem 1.5 (Barron's approximation theorem)

Proof. The idea is to prove that f is in the closure of the convex hull of the set of bounded neuron functions G_ϕ. Then functional analysis tells us that the square error between f and a certain convex combination of n elements of G_ϕ (which is an element of S-2LP) scales like $O(n^{-1})$, which is what we want. In order to prove that $f \in \bar{co}(G_\phi)$, Barron first uses the Fourier transform on f to see that $f \in \bar{co}(G_{\cos})$, where elements in $G_{\cos}$ are differences of cosines, then he approximates functions in $G_{\cos}$ using functions in G_ϕ to prove $\bar{co}(G_{\cos}) \subseteq \bar{co}(G_\phi)$. Therefore the proof can be divided into three lemmas.

First fix $f \in \mathcal{B}$ with $f(0) = 0$, $C \geq C_f$, μ a probability distribution with support $\subseteq B(0, r)$ for some $r > 0$. In the following, the closure is taken with respect to $L_2(\mu, B)$.

Lemma 4.1. *Let*

$$G_{\cos} = \left\{ g : x \to \tfrac{\gamma}{\|\omega\|} \left(\cos(\omega \cdot x + b) - \cos(b) \right), \omega \neq 0, |\gamma| \leq rC \right\}.$$

Then $f \in \bar{co}(G_{\cos})$, the closure of the convex hull of $G_{\cos}$.

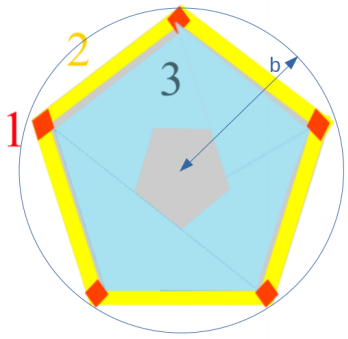
Lemma 4.2. *Let*

$$G_\phi = \left\{ g : x \to \gamma\phi(a \cdot x + b), |\gamma| \leq 2rC, a \in \mathbb{R}^d, b \in \mathbb{R} \right\}$$

Then $\bar{co}(G_{\cos}) \subseteq \bar{co}(G_\phi)$

Lemma 4.3. *Let $G \subseteq H$ a Hilbert space, with $\sup_G \|g\| \leq b$, let $f \in \bar{co}(G)$. Then $\forall n \geq 1, c' > b^2 - \|f\|^2$, there exists f_n in the convex hull of n points of G such that $\|f - f_n\|^2 \leq \frac{c'}{n}$.*

The bound is lower when $\|f\|$ is close to b, because points near the boundary of the convex hull of G are combinations of less points in G.



Points in the convex hull of five points well approximated by 1,2 or 3 points.

Note that this bound is independent of the size or dimension of H, thus avoiding the curse of dimensionality. In a sense, the strength of neural networks in high-dimensional approximation can be traced to this lemma.

Putting the lemmas together yields the proof of theorem 1.5. Indeed, for $f \in \mathcal{B}$, by lemmas 4.1 and 4.2 choosing $C = C_f$, we have $f \in \bar{co}(G_\phi)$. Furthermore, because $|\phi| \leq 1$ we have $\|g\|_\mu \leq \|g\|_\infty$

(because μ has total weight 1) and $\|g\|_\infty \leq |\gamma| = 2rC_f$ for any $g \in \bar{co}(G_\phi)$. We can take $c' = (2rC_f)^2$ as long as f is non-null (otherwise the theorem is trivial). Therefore, by lemma 4.3,

$$\forall n \geq 1, \exists \left(\gamma_i, a_i, b_i\right)_{i=1}^n, f_n : x \to \sum_{i=1}^n \gamma_i \phi(a_i \cdot x + b_i)$$

$$\|f - f_n\|^2 \leq \frac{(2rC_f)^2}{n}$$

□

It remains to prove the lemmas.

Proof. (Of lemma 4.1.) Actually we will see that $G_{\cos}$ arises naturally when considering the Fourier transform of f. Since $f \in \mathcal{B}$, we have

$$f(x) - f(0) = \mathrm{Re} \int_{\mathbb{R}^d} (e^{i\omega \cdot x} - 1)\tilde{f}(\omega) d\omega = \int_{\mathbb{R}^d} \mathrm{Re}\left[(e^{i\omega \cdot x} - 1)\tilde{f}(\omega)\right] d\omega$$

because f is real-valued. Writing $\tilde{f}$ in polar form $\tilde{f}(\omega) = e^{i\theta(\omega)}\hat{f}(\omega)$ with $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ and $\bar{f}(x) = f(x) - f(0)$, we have

$$\bar{f}(x) = \int_{\mathbb{R}^d} \underbrace{(\cos(\omega \cdot x + \theta(\omega) - \cos(\theta(\omega)))}_{=:\Delta \cos(x,\omega)} \hat{f}(\omega) d\omega$$

In this form, we see that $\bar{f}$ can be written as an uncountable sum of elements of $G_{\cos}$. In order to prove that this involves $\bar{f} \in \bar{co}(G_{\cos})$, Barron uses the probabilistic method to approach $\bar{f}$ in $co(G_{\cos})$.

Note that $\Lambda(\omega) := \frac{\|\omega\|}{C_f}\hat{f}(\omega)$ is a probability distribution because of the definition of C_f (equation 3). Thus, we can write

$$\bar{f}(x) = \mathbb{E}_\Lambda(g(x, \omega)),$$

with $g(x, \omega) = \frac{C_f}{\|\omega\|}\Delta \cos(x, \omega)$ and ω is seen as a random variable. Since $\|\cos'\|_\infty = \|\sin\|_\infty \leq 1$, the mean value theorem gives $|\Delta \cos(x, \omega)| \leq |\omega \cdot x| \leq r \|\omega\|$ for all $x \in B(0, r)$, thus $|g(x, \omega)| \leq rC_f$.

Drawing $\omega_1, .., \omega_n$ from Λ, taking $\bar{g}(x, \omega_i) = \frac{1}{n}\sum_{i=1}^n g(x, \omega_i) \in G_{\cos}$, defining a loss function $\ell(x, \omega_i) = |\bar{f}(x) - \bar{g}(x, \omega_i)|^2$ (such that $\mathbb{E}_\Lambda \ell(x, \omega_i) = \mathrm{var}_\Lambda\ \bar{g}$), we have

$$\ell(n) := \mathbb{E}_\Lambda \left\|\bar{f}(x) - \bar{g}(x, \omega_i)\right\|_\mu^2 = \mathbb{E}_\Lambda\ \mathbb{E}_\mu\ \ell(x, \omega_i)$$

if f is essentially bounded, by Fubini's theorem,

$$\ell(n) = \mathbb{E}_\mu\ \mathbb{E}_\Lambda\ \ell(x, \omega_i) = \mathbb{E}_\mu\ \mathrm{var}_\Lambda\ \bar{g} = \frac{1}{n}\left[\mathbb{E}_\mu\ \mathrm{var}_\Lambda\ g\right] \leq \frac{(rC_f)^2}{n}.$$

Recall that for any $\epsilon > 0$ we wish to approach $\bar{f}$ in $G_{\cos}$ with precision ϵ in the $L_2(\mu, B)$ space. Let $n \geq 1$ such that $\ell(n) \leq \epsilon$. Since $\ell(n)$ is the Λ-expected loss in μ, there must exist at least one $\bar{g}$ with $\left\|\bar{f} - \bar{g}\right\|_\mu^2 \leq \ell(n) \leq \epsilon$. Since $\bar{g} \in co(G_{\cos})$ for all ϵ and ϵ is arbitrary, we deduce $\bar{f} \in \bar{co}(G_{\cos})$. □

Proof. (Of lemma 4.2.) The idea is to approximate an element of $G_{\cos}$ by sums of step functions, which in turn are approximated by ϕ using its sigmoidal properties.

Take $g \in G_{\cos}$. Then $g = m \circ l$, where $l(x) = \frac{\omega}{\|\omega\| r} \cdot x,\ B(0,r) \to [-1,1]$ and $m(z) = \frac{\gamma}{\|\omega\|}(\cos(r\,\|\omega\|\, z + b) - \cos(b)),\ [-1,1] \to \mathbb{R}$. m can be uniformely approximated by

$$m_+(z) = \sum_{i=1}^{n} (m(t_i) - m(t_{i-1})) \mathbf{1}_{z \geq t_i}$$

where the t_i, $0 \leq i \leq n$ are uniformely distributed on $[-1,1]$. By the mean value theorem, $\sum_i |n(t_i) - n(t_{i-1})| \leq 2r\gamma$ and $\max |m(z) - m_+(z)| \leq 2r\gamma/n$. Thus $m_+ \circ l$ uniformely approaches g on $B(0,r)$ (and therefore also in $L_2(\mu)$). We have proven $g \in \bar{co}(G_{\text{step}})$,

$$G_{\text{step}} := \left\{ f : x \to \Gamma(\text{step}(\alpha \cdot x - t)), \Gamma \leq 2rC, \|\alpha\| = 1/r, |t| \leq 1 \right\}.$$

Next take $h \in G_{\text{step}}$, we prove $h \in \bar{co}(G_\phi)$. If $h(x) = \Gamma\ \text{step}(\alpha \cdot x - t)$, then by definition of sigmoidal functions, the sequence $\phi_n(x) = \Gamma\phi(n(\alpha \cdot x - t))$ converges pointwise to h. By the dominated convergence theorem, this convergence holds in $L_2(\mu, B)$. Thus $h \in \bar{co}(G_\phi)$.

To conclude this lemma, we use the fact that inclusion is stable under closure and the closure of a convex set is convex. Thus

$$\bar{co}(G_{\cos}) \subseteq \bar{co}(G_{\text{step}}) \subseteq \bar{co}(G_\phi).$$

□

Proof. (Of lemma 4.3.) We again make use of the probabilistic method. f is only in the closure of the convex hull of G, so we first take f^* (we will make f^* close to f) in the convex hull of G. This means $f^* = \sum_{i=1}^{m} p_i g_i^*$, $\sum p_i = 1$. Draw n iid. samples $g_1, ..., g_n$ taking value g_i^* with probability p_i and take $f_n = \bar{g_i}$ the mean of the g_i. Since $\mathbb{E}(g_i) = \mathbb{E}(g_1) = f^*$, we have $\mathbb{E}\,\|f^* - f_n\|^2 = \text{var}(f_n) = \frac{1}{n}\text{var}(g_1)$. Noting that

$$\begin{aligned} \text{var}(g_1) = \mathbb{E}\langle f^* - g_1, f^* - g_1 \rangle &= \underbrace{\mathbb{E}\langle f^*, f^* \rangle}_{\|f^*\|^2} + \mathbb{E}\langle g_1, g_1 \rangle - 2\underbrace{\mathbb{E}\langle f^*, g_1 \rangle}_{\|f^*\|^2} \\ &= \mathbb{E}\,\|g_1\|^2 - \|f^*\|^2 \leq b^2 - \|f^*\|^2 \end{aligned}$$

where the last inequality comes from $\mathbb{P}(\|g_1\|^2 \leq b^2) = 1$ since $g_1 \in co(G)$. So $\mathbb{E}\,\|f^* - f_n\|^2 \leq \frac{1}{n}(b^2 - \|f^*\|^2)$, which means there must exist at least one drawing of f_n with $\|f^* - f_n\|^2 \leq \frac{1}{n}(b^2 - \|f^*\|^2)$. Thus for any $c' > (b - \|f\|^2)$, taking $\|f - f^*\|$ sufficiently small gives, by the triangle inequality,

$$\|f - f_n\|^2 \leq \frac{c'}{n}$$

□

4.2 Theorem 1.8 (Barron's error theorem)

Proof. (Of theorem 1.8). Barron structures the proof around the key notion of Index of Resolvability (IR) which incorporates both the approximation and the estimation error to the target function. One idea of IR is to penalize a too fine tuning of the parameters in order to avoid overfitting. The value of IR is then minimized over a discrete set to give an approximation $\hat{f}$ of the target function with convergence speed as in the theorem.

As in Barron's aprroximation theorem, we let μ be a probability distribution on $B(0,r)$. N is the number of observations of $(X_i, f(X_i))$ where X_i are drawn from μ. n is the "number of neurons", ie. we will have our approximation $f_n \in \text{S-2LP}(n)$. The parameter space $\mathcal{P} = \mathcal{P}(n)$ is $\mathbb{R}^n \times (\mathbb{R}^d)^n \times \mathbb{R}^n$,

incorporating the a_i, b_i, c_i necessary to define an element of S-2LP(n). Recall that $c_{x,y}(\cdot)$ is the clip function.

Let Θ be a discrete set of parameters in a parameter space $\mathcal{P}$. Let $L : \mathcal{P} \to \mathbb{R}^+$ be such that $\sum_\Theta e^{-L(\theta)} \leq 1$. We use L to define the complexity of a parameter $\theta \in \mathcal{P}$. (If $L(\theta)$ was the length of the codeword used for θ in signal compression, the condition would be equivalent to the code being uniquely decompressible).

Definition 4.4. *The index of resolvability of a function $f \in \mathcal{B}$ where f takes values in an interval I_0, given a maximum number of neurons n, a number of observations N and a discrete set of admissible parameters $\Theta = \Theta_{n,N} \subset \mathcal{P}(n)$, is*

$$R(f) = R_{n,N}(f) = \min_{\Theta} \left[\|f - c(f_{n,\theta})\|_\mu^2 + \lambda \frac{L(\theta)}{N} \right]$$

with $c(\cdot)$ the clip function for I_0 and

$$\theta = (a_i, b_i, c_i)_{i=1}^n, \;\; f_{n,\theta}(x) = \sum_{i=1}^n a_i \phi(b_i \cdot x + c_i)$$

The IR allows us to define the *minimum complexity estimator* :

Definition 4.5. *The minimum complexity estimator (MCE) of a given size n is $\hat{f}_{n,N} := c(f_{n,\hat{\theta}_{n,N}})$ where*

$$\hat{\theta}_{n,N} = \text{argmin}_{\Theta_{n,N}} \left[\frac{1}{N} \sum_{i=1}^n (Y_i - c(f_{n,\theta}(X_i)))^2 + \lambda \frac{L_{n,N}(\theta)}{N} \right]$$

Thus MCE is the minimisor of the index of resolvability for the observed distribution $\mu' = \frac{1}{N} \sum \delta_{X_i}$.

The interest of $R(f)$ and $\hat{f}_{n,N}$ lie in this important following lemma :

Lemma 4.6. *For $f \in \mathcal{B}$ with range(f) in an interval I_0 of length b, for any $\lambda \geq 5b^2/3$, any $n, N \geq 1$, we have*

$$\mathbb{E}_\mu \left\| f - \hat{f}_{n,N} \right\|^2 \leq \gamma R_{n,N}(f) + \frac{2\gamma\lambda}{N}$$

Note that both the left and the right-hand side depend on the choice of Θ. This lemma is admitted as true as its proof goes beyond the scope of this essay. A sketch of proof is available in [2].

Since we know that growth (with the number of neurons) of magnitude of parameters in a neural network can be controlled, we will let the finite set of admissible parameters Θ_n be a ϵ-net on a certain "ball" of $\mathcal{P}$.

Definition 4.7. $\Theta_{n,\tau} := \left\{ \theta = (a_i, b_i, c_i) |\ |b_i|_1 \leq \tau, |c_i| \leq \tau \right\}$. *If $I_0 = [i_1, i_2]$, we let*

$$\Theta_{n,\tau,C} := \left\{ \theta \in \Theta_{n,\tau} \ |\ \|a\|_1 = \textstyle\sum_i |a_i| \leq C, a_0 \in I_C = [i_1 - C, i_2 + C] \right\}.$$

For any $\epsilon > 0$, we let

$$\Theta_{n,\tau,C,\epsilon} = \textit{a finite set that } \epsilon\textit{-covers } \Theta_{n,\tau,C}.$$

More precisely, such that for any $\theta \in \Theta_{n,\tau,C}$ and $\theta^* \in \Theta_{n,\tau,C,\epsilon}$, we have

$$\begin{cases} \|a - a^*\|_1 \leq C\epsilon \\ |c_0 - c_0^*| \leq C\epsilon \\ |b_i - b_i^*| \leq \epsilon \\ |c_i - c_i^*| \leq \epsilon \end{cases} \quad \forall 1 \leq i \leq n. \tag{44}$$

Lemma 4.8. *For $f \in \mathcal{B}$ with image in an interval I_0, $C \geq C_f$, for any $n \in \mathbb{N}$, if we have $(\tau_i)_{i=1}^{\infty}$ and $(\epsilon_i)_{i=1}^{\infty}$ with*

$$\epsilon_n = O(\frac{1}{\sqrt{n}}) \tag{45}$$

and equations 4 and 44 hold for τ_n and $\Theta_n = \Theta_{n,\tau_n,C,\epsilon_n}$, then there exists $\theta_n^* \in \Theta_n$ such that $(\theta_i^*)_{i=1}^{\infty}$ satisfies $\|f - c_{I_0}(f_{n,\theta^*})\|_\mu = O(\frac{C}{\sqrt{n}})$, where $f_{n,\theta^*} \in$ S-2LP(n) has parameters given by θ^* and c_{I_0} is the clip function on I_0.

Proof. (Sketch.)

- Let θ be the best parameter in $\Theta_{n,\tau_n,C}$ to minimise $\|f - c(f_{n,\theta})\|$. Then if τ_n verifies equation 4, Barron's approximation theorem (with bound on parameters) guarantees that $\|f - c(f_{n,\theta})\|_\mu \leq O(\frac{C}{\sqrt{n}})$.
- We can show that for any $\theta \in \Theta_{n,\tau_n,C}$, there exists $\theta^* \in \Theta_n = \Theta_{\text{finite}}$ such that $\|f_{n,\theta} - f_{n,\theta^*}\| \leq 4vC\epsilon$, where $v = \max(1, v_1)$ and ϕ is v_1-lipschitz (this was a condition of the theorem). We can show this in the supremum norm by triangular inequalities, and therefore in the $L_2(\mu)$ norm.
- If $\epsilon_n = O(n^{-1/2})$, then

$$\begin{aligned}\|f - f_{n,\theta^*}\|_\mu &\leq \|f - f_{n,\theta}\|_\mu + \|f_{n,\theta} - f_{n,\theta^*}\|_\mu \\ &\leq O(\frac{C}{\sqrt{n}}) + O(C\epsilon_n) \leq O(\frac{C}{\sqrt{n}})\end{aligned}$$

□

In other words, a precise enough finite grid on the parameter ball $\Theta_{n,\tau,C}$ also yields approximation error in $O(Cn^{-1/2})$. This grid also gives a natural $L : L(\theta) = \log \#\Theta_{n,\tau_n,C,\epsilon_n}$ where $L(\theta)$ reflects (as suggested) a level of complexity for θ. It is this L that will help prove lemma 4.9.

Lemma 4.9. *If equations 4, 44 and 45 hold, if we also have*

$$\epsilon_n \geq (CndN)^{-p} \text{ for some } p \geq 1 \tag{46}$$

$$\tau_n \text{ bounded by a polynomial in } n \tag{47}$$

then

$$R_{n,N}(f) = O(\frac{C^2}{n}) + O(\frac{nd}{N}\log(CndN))$$

for any $C \geq C_f$.

Proof. (Sketch). The idea is that if we choose $L(\theta) = \log \#\Theta_{n,\tau_n,C,\epsilon_n}$, conditions (46) and (47) guarantee that the regular grid on $\Theta_{n,\tau_n,C} \subseteq \mathcal{P}$ (with constant spacing ϵ on the a_i, b_i, c_i coordinates) verifies the conditions in (44) and has cardinality bounded by $O(nd\log(CndN))$. Thus, because L is constant on $\Theta_n = \Theta_{n,\tau_n,C,\epsilon_n}$,

$$\begin{aligned}R_{n,N}(f) &= \min_{\Theta_n}\left[\|f - c_{I_0}(f_{n,\theta})\|^2 + \lambda\frac{L(\theta)}{N}\right] \\ &\leq O(\frac{C^2}{n}) + O(\frac{nd}{N}\log(CndN))\end{aligned}$$

□

Armed with these lemmas, we can prove Barron's error theorem : combining lemmas 4.6 and 4.9, we have

$$E := \mathbb{E}_\mu \left\| f - c_{I_0}(\hat{f}_{n,N}) \right\|^2 = O(R_{n,N}(f)) + O(\frac{1}{N}) \tag{48}$$

$$= O(\frac{C^2}{n}) + O(\frac{nd}{N}\log(CndN)) + O(\frac{1}{N}) \tag{49}$$

$$= O(\frac{C^2}{n}) + O(\frac{nd}{N}\log(CndN)) \tag{50}$$

with $\hat{f}_{n,N}$ minimising the index of resolvability for the observed distribution $\mu' = \sum_i \delta_{X_i}$ for any parameter $\lambda \geq 5b^2/3$ over the finite set Θ_n verifying (44) with parameters verifying (4), (45), (46) and (47).

Recall that we want λ_1, λ_2 such that

$$E \leq \lambda_1 \cdot \frac{C^2}{n} + \lambda_2 \cdot \frac{nd}{N}\log(N) \tag{51}$$

for all possible values of C, n, d, N. Now recall that f and $c_{I_0}(\hat{f}_{n,N})$ take values in I_0 which has length b. Therefore, because μ has weight 1, $E \leq (2b)^2$. This means that taking $\lambda_1, \lambda_2 \geq (2b)^2$, the bound is trivial for any values of C, n, d, N with $C^2 \geq n$ or $nd \geq N/\log(N)$. However, if $n \geq C^2$ and $nd \leq N/\log(N)$, we have $\log(nd) \leq \log(N/\log(N))$, implying

$$\frac{\frac{nd}{N}\log(NCnd)}{\frac{nd}{N}\log(N)} = 1 + \frac{\log(ndC)}{\log(N)} \leq 1 + 2\frac{\log(nd)}{\log(N)} = 3 - \frac{\log(\log(N))}{\log(N)} \leq 3,$$

which means that $O(\frac{nd}{N}\log(N)) = O(\frac{nd}{N}\log(CndN))$, thus (51) follows from (50) as long as we take the constants λ_1, λ_2 to be $\geq (2b)^2$. We have proved the theorem.

□

5 Proofs of section 3 (ReLU ANNs)

5.1 Proposition 2.7

Yarotsky's proofs are constructive.

Proof. (Of proposition 2.7) (Sketch). Recall we'd like to show that there exists a ReLU network architecture which ϵ-approximates any $f \in F_{n,d}$ with only $c\epsilon^{-d/n}(\log(1/\epsilon) + 1)$ parameters.

In order to use the regularity of $F_{n,d}$, the author divides $[0,1]^d$ into multiple small cubes of length $1/N$ and centers a Taylor polynomial approximating f in each cube. He then expresses the deep ReLU network as an approximation of a combination of these polynomials. This way, he can approximate by construction a known number of sums of products rather than an unknown regular f.

- Take $N \in \mathbb{N}$, let $\mathbf{m'} \in [0,..N]^d$ and $\mathbf{m} = \frac{\mathbf{m'}}{N} \in [0,1]^d$. Let $P_\mathbf{m}$ be the $n-1$ order Taylor polynomial for f at $\mathbf{m}$,

$$P_\mathbf{m}(\mathbf{x}) = \sum_{|\boldsymbol{n}|<n} \frac{D^{\boldsymbol{n}} f}{\boldsymbol{n}!}(\mathbf{m})(\mathbf{x}-\mathbf{m})^{\boldsymbol{n}}$$

  Let $\{\phi_\mathbf{m}\}$ be a partition of unity ($\sum_\mathbf{m} \phi_\mathbf{m}(\mathbf{x}) = 1$) with $\phi_\mathbf{m}$ continuous centered at $\mathbf{m}$,

$$\phi_\mathbf{m}(\mathbf{x}) = \prod_{i=1}^d \psi_N(\mathbf{x}_i - \mathbf{m}_i),\ \psi_N(x) = \begin{cases} 1,\ |x| \leq 1/N \\ 2 - N|x|,\ 1/N \leq |x| \leq 2/N \\ 0,\ |x| \geq 2/N \end{cases}$$

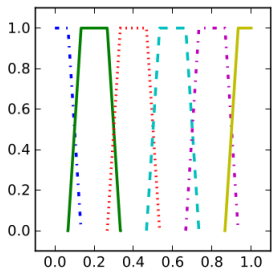


Figure 17: $(\phi_m)_{m=0}^5$ for $d = 1$.

Note that as a piecewise linear function, $\phi_{\mathbf{m}}$ is easily expressible as a ReLU network function. We can then probably expect that $f_1(\mathbf{x}) = \sum_{\mathbf{m}} P_{\mathbf{m}}(\mathbf{x})\phi_{\mathbf{m}}(\mathbf{x})$ is a good approximation of f. In fact, Yarotsky shows that because of the regularity of f, we have $\|f - f_1\|_\infty \leq \frac{2^d d^n}{n!}(\frac{1}{N})^n$.

- Yarotsky's idea is to approximate f_1 by construction, because

$$f_1(\mathbf{x}) = \sum_{\mathbf{m},\ |\boldsymbol{n}|<n} a_{\mathbf{m},\boldsymbol{n}}\phi_{\mathbf{m}}(\mathbf{x})(\mathbf{x}-\mathbf{m})^{\boldsymbol{n}}$$

  is a sum of products of linear and piecewise linear functions. The sum function $\mathbb{R}^2 \to \mathbb{R}$ for bounded inputs can be expressed exactly by a ReLU network with $O(1)$ parameters. The product function $\mathbb{R}^2 \to \mathbb{R}$ is more complex ; Yarotsky shows that it can be ϵ-approximated for any $x, y \in [-M, M]$ with $c_1 \ln(1/\epsilon) + c_2$ weights and computation units, with an absolute constant c_1 and with c_2 dependent on M. His proof relies on the observation that $xy = \frac{1}{2}\left((x+y)^2 - x^2 - y^2\right)$. Yarotsky builds a ReLU network that approximates $x \to x^2$ geometrically well on $[0, 1]$, then concludes by combining with sums and scalings, and counting the number of weights he has to use.

- In order to prove the proposition for any ϵ, Yarotsky $\epsilon/2$ approximates f with f_1 then estimates the number of weights necessary to $\epsilon/2$-approximate f_1 with a ReLU network as a result of approximation of the product function. He obtains an upper bound of $c\epsilon^{-d/n}(\log(1/\epsilon) + 1)$ weights. Note that the network architecture (representing the organisation of sums and products) is independent of f ; only the $a_{\mathbf{m},\boldsymbol{n}}$ depend on f. Thus the proposition is proved.

□

5.2 Proposition 2.8

Proof. (Of 2.8)(Sketch). (Allowing varying network architecture allows us to go under DeVore's bound for $F_{1,1}$). We'd like to show for any $f \in F_{1,1}$, there exists a ReLU network of depth 6 that ϵ-approximates f with $O(\frac{1}{\epsilon \ln(1/\epsilon)})$ parameters only (and not $O(1/\epsilon)$ parameters).

The idea of the proof is that unlike networks with one hidden layer, deep networks can store m-neuron patterns and use them l times at the cost of only $l + m$ connections, instead of lm connections. This is illustrated in figure 18. Yarotsky divides approximation in two scales : the T scale, which corresponds to classic linear interpolation, and the m scale, which corresponds to what he calls "cache", ie. pattern storing.

- In order to structure the proof, Yarotski divides the approximation of f in f_1 and f_2, ie. the network function is $f_1 + f_2 \approx f$. f_1 represents approximation on the the T scale. It is given $O(T)$

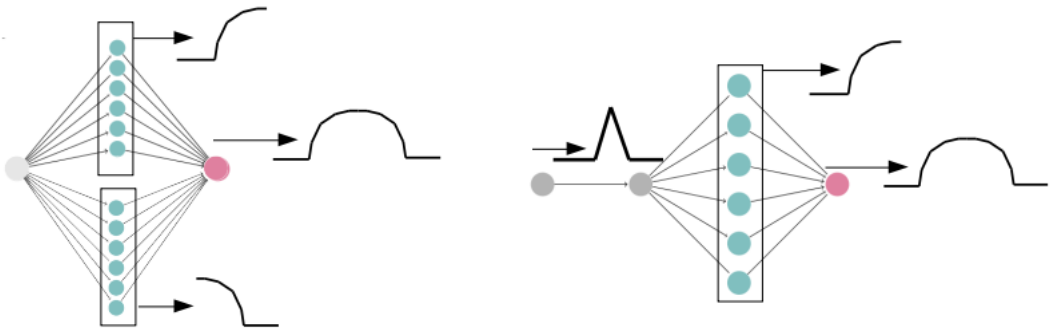


Figure 18: The power of "pattern memorisation" in deep networks. Shallow and deep networks attempt to approximate $\sqrt{1-x^2}$. Symmetry can be used with multiple hidden layers, but not with a single one.

neurons and weights and linearly interpolates f at T regular intervals :

$$f_1(x) = T(f(t_{i+1}) - f(t_i))(x - t_i) + f(t_i) \text{ on } [t_i; t_{i+1}] = [i/T; (i+1)/T]$$

- The problem is now to approximate $f - f_1 =: g$. Note that $g(t_i) = 0$ and g is 2-Lipschiz (as f 1-Lipschiz $\implies$ f_1 1-Lipschiz). This is where we make use of "cached" functions on the m scale.
- Let m be an integer and Γ be the set of continuous funtions $h : [0,1] \to \mathbb{R}$ linear on all $[t_r, t_{r+1}]$, $t_r = r/m$, and such that

$$h(t_{r+1}) - h(t_r) \in \left\{-\tfrac{2}{m}, 0, \tfrac{2}{m}\right\}, h(0) = h(1) = 0$$

There are at most 3^m such functions. Furthermore, for any h_1 continous 2-Lipschiz on $[0,1]$ with $h_1(0) = h_1(1) = 0$ there exists $h \in \Gamma$ with $\|h - h_1\|_\infty \leq 1/m$. With rescaling, we can apply this to g on $[t_1, t_{i+1}]$. Thus there exists $\{\gamma_t\}_{t=1}^T \subseteq \Gamma$ such that

$$f_2(x) = \frac{1}{T}\gamma_t(Tx - t) \text{ on } I_t = [t/T, (t+1)/T]$$

$$\|g - f_2\|_\infty \leq \frac{1}{Tm}$$

The idea is now to encode all $\gamma \in \Gamma$ in the neural network, and to link each time interval to the corresponding γ_t. The structure of the f_2-part of the neural network is shown in figure 19.

- The total number of neurons and weights in Yarotsky's network is $O(\underbrace{T}_{\text{for } f_1} + \underbrace{m|\Gamma|}_{\text{storing cached functions}}) =$ $O(T + m3^m)$ and the error is $\epsilon \leq \frac{1}{Tm}$. Thus, for any $\epsilon > 0$ if we take $m = \lceil \log_3(\frac{1}{\epsilon}) \rceil$ and $T = \lceil \frac{1}{m\epsilon} \rceil$, we have

$$\|f - (f_1 + f_2)\|_\infty = \|g - f_2\|_\infty \leq \frac{1}{Tm} \leq \epsilon$$

with only $O(\frac{1}{\epsilon \ln(\epsilon)})$ parameters.

□

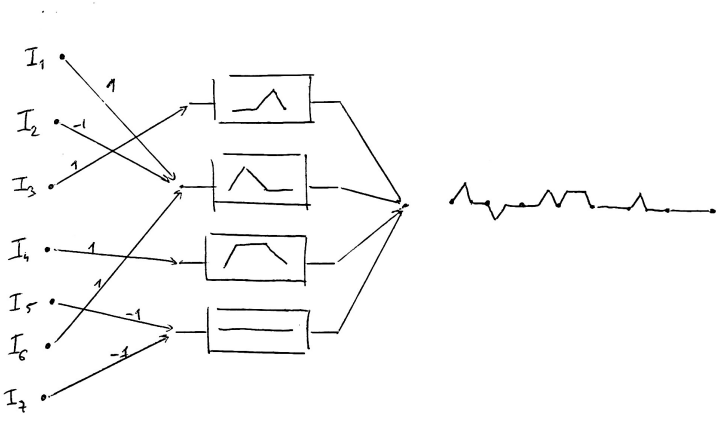


Figure 19: Memory part of Yarotsky's ReLU network for $m = 3$. The boxes hide a network comprising one hidden layer of m neurons and computing the drawn network function. The I_t dots represent neurons which are activated only on I_t, with activation going from 0 to 1 in that interval. Weights on the right are 1. The total network function, drawn on the right, is $f_2 \approx g = f - f_1$.

5.3 Proposition 2.10

Proof. (Of 2.10).

We must show that fixed-depth networks of depth $\leq L$ that ϵ-approximate a nonlinear C^2 function must have at least $c\epsilon^{-1/2(L-2)}$ weights and computation units. This shows the superiority of non-fixed depth networks in the case where $f \in F_{n,d}$ and $n > 2d(L-2)$. We start by showing that we can restrict ourselves to strict convex functions on $[-1,1]$, then we examine the error cost of linear interpolation of those functions depending on the number of interpolation points. We conclude by examining the number of neurons required for such interpolations.

Lemma 5.1. *If $f \in C^2([-1,1]^d, \mathbb{R})$ is non-affine (ie. $f \neq b + \langle a, \cdot \rangle \ \forall b, a$), then there exists $\boldsymbol{x} \in [-1,1]^d$, $\boldsymbol{v} \in \mathbb{R}^d$ with $\boldsymbol{x} + t\boldsymbol{v} \in [-1,1]^d$ on $t \in [-1,1]$ and $f_1 : t \to f(\boldsymbol{x} + t\boldsymbol{v})$ is strictly convex or concave on $[-1,1]$.*

Proof. Since f is C^2 non-affine, there must exist $x, y \in [-1,1]^d$ with f non-affine on the 1-dimensional segment $[x,y]$ (otherwise, f is not even C^1). Here f on $[x,y]$ is f_1 on $[0,1]$. There must exist a point $t_0 \in [0,1]$ with $f_1''(t_0) \neq 0$. Because f is C^2, f_1'' is continuous and there exists $\delta > 0$ with $f_1'' \neq 0$ on $[t_0 - \delta, t_0 + \delta]$, which means that the sign of f_1'' is unchanged on that interval, so f_1 is strictly convex or concave on $[t_0, t_0 + \delta]$. Taking $\mathbf{x} = x + t_0(y - x)$ and $\mathbf{v} = \delta(y - x)$, the lemma is proved. □

Call the network and network function $\tilde{f}$. If we have $\left\| f - \tilde{f} \right\|_\infty \leq \epsilon$, then in particular $\left\| f_1 - \tilde{f}_1 \right\|_\infty \leq \epsilon$, where $f_1, \tilde{f}_1 : [-1,1] \to \mathbb{R}$ is $f, \tilde{f}(\mathbf{x} + t\mathbf{v})$ as in lemma 5.1. Because $\tilde{f}_1$ is from a ReLU network, it is continuous and piecewise linear. If U is the number of neurons in $\tilde{f}$, then the number of linear pieces in $\tilde{f}$ (and thus in $\tilde{f}_1$) is not greater than $(2U)^{L-2}$ [This counting assertion is shown by adapting a lemma from Telgarsky [10]]. Thus there exists an interval $[a,b] \subseteq [-1,1]$ of length at least $2(2U)^{-(L-2)}$ with $\tilde{f}_1$ affine on $[a,b]$.

Suppose in full generality that f_1 is strictly convex (and not strictly concave) on $[a,b]$ and call $c_1 := \min_{[a,b]} f_1''(x) > 0$, $g := f_1 - \tilde{f}_1$. Notice we also have $g'' \geq c_1$ on $[a,b]$, as well as $\|g\|_\infty \leq \epsilon$. The

inequality

$$\max(g(a), g(b)) \geq g(\frac{a+b}{2}) + \frac{c_1}{2}\left(\frac{b-a}{2}\right)^2$$

can be deduced *ab absurdo* by applying the mean value theorem twice consecutively. Thus,

$$\epsilon \geq \left[\max(g(a), g(b)) - g(\frac{a+b}{2})\right] \geq \frac{c_1}{2}\left(\frac{b-a}{2}\right)^2 \geq \frac{c_1}{2}(2U)^{-2(L-2)}$$

Reversing the inequality, we get

$$U \geq \underbrace{\frac{1}{2}(\frac{2\epsilon}{c_1})^{-1/2(L-2)}}_{:=c(f,L)} \cdot \epsilon^{-1/2(L-2)}$$

□

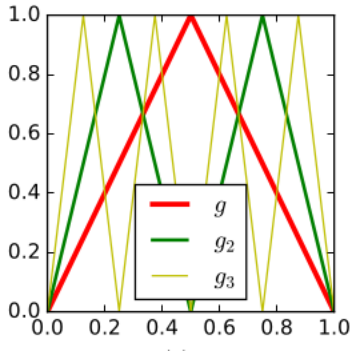


Figure 20: $g_{i+1} = g \circ g_i$ and g_1 is the triangle/sawtooth function. g can be represented as a 3-neuron layer and g_i with a depth i network of 3-neuron layers. The number of linear components, unlike the number of neurons, scales like 2^i.

Note. $(2U)^{L-2}$ might seem a large bound for the number of linear pieces of a ReLU network function. This is in fact a large relaxation of Telgarsky who uncovered the bound $(2m)^{L-2}$ for neural networks with layers of $\leq m$ neurons, each layer being connected only to its neighbour layers. The exponent in L comes from the fact that each new layer adds a "composition" to the network function ($\tilde{f_{new}} = \Psi(\tilde{f_{old}})$), which geometrically scales the network function complexity. An example is shown in figure 20.

6 Proofs of section 4 (Numerical Optimisation)

6.1 Theorem 3.1

Proof. For simplicity, we will focus on the continuous version of the gradient descent:

$$\frac{d\theta_t}{dt} = -\nabla\left(\hat{R}_n(\theta_t)\right) \tag{52}$$

In this proof, we will use other notations: m will be the number of neurons and n the number of data points. We remember the definition of the Gram matrices

$$K^{(a)}_{i,j} = \frac{1}{n}\mathbb{E}_{b\sim\pi_0}\left[\sigma(bx_i)\sigma(bx_j)\right] \tag{53}$$

$$K^{(b)}_{i,j} = \frac{1}{n}\mathbb{E}_{b\sim\pi_0}\left[\sigma'(bx_i)\sigma'(bx_j)\langle x_i, x_j\rangle\right] \tag{54}$$

with the assumption that $\lambda_n^{(a)} := \lambda_{min}(K^a) > 0$ and $\lambda_n^{(b)} := \lambda_{min}(K^b) > 0$. We also define

$$G_{i,j}^{(a)} = \frac{1}{nm}\sum_{k=1}^{m}\sigma(b_k x_i)\sigma(b_k x_j) \tag{55}$$

$$G_{i,j}^{(b)} = \frac{1}{nm}\sum_{k=1}^{m}a_k^2\sigma'(b_k x_i)\sigma'(b_k x_j)x_i^T x_j \tag{56}$$

and $G = G^{(a)} + G^{(b)}$. We have

$$\partial_{a_k}\hat{R}_n = \frac{1}{n}\sum_{i=1}^{n}(f_m(x_i,a,b) - y_i)\sigma(b_k x_i) \tag{57}$$

$$\partial_{b_k}\hat{R}_n = \frac{1}{n}\sum_{i=1}^{n}a_k(f_m(x_i,a,b) - y_i)\sigma'(b_k x_i)x_i \tag{58}$$

Thus, with $u_i = f_m(x_i,a,b) - y_i$

$$\begin{aligned}
&\left\|\nabla_\theta \hat{R}_n\right\|^2 = \left\|\nabla_a \hat{R}_n\right\|^2 + \left\|\nabla_b \hat{R}_n\right\|^2 \\
=&\frac{1}{n^2}\sum_{k=1}^{m}\sum_{i,j=1}^{n}u_i u_j\sigma(b_k x_i)\sigma(b_k x_j) + \frac{1}{n^2}\sum_{k=1}^{m}\sum_{i,j=1}^{n}a_k^2 u_i u_j\sigma'(b_k x_i)\sigma'(b_k x_j)x_i^T x_j \\
=&\frac{m}{n}\sum_{i,j=1}^{n}u_i u_j G_{i,j}^{(a)} + \frac{m}{n}\sum_{i,j=1}^{n}u_i u_j G_{i,j}^{(b)} \\
=&\frac{m}{n}u^T G u
\end{aligned} \tag{59}$$

implying that

$$\left\|\nabla_\theta \hat{R}_n\right\|^2 \geq \frac{m}{n}\lambda_{\min}(G)u^T u = 2m\lambda_{\min}(G)\hat{R}_n \tag{60}$$

and so,

$$\frac{d\hat{R}_n}{dt} = (\nabla_\theta \hat{R}_n)^T\frac{d\theta}{dt} = -\left\|\nabla_\theta \hat{R}_n\right\|^2 \leq -2m\lambda_{\min}(G)\hat{R}_n \tag{61}$$

It remains to prove the appropriate lower bound for the term $\lambda_{\min}(G(\theta_t))$. Since,

$$\lambda_{\min}(G(\theta_t)) \geq \lambda_{\min}(G(\theta_0)) - \|G(\theta_t) - G(\theta_0)\|_F \tag{62}$$

The first observation is that $G^{(a)}(\theta_0) \to K^{(a)}$ and $G^{(b)}(\theta_0) \to \beta^2 K^{(b)}$ pointwise as the number of neurons increases. We can prove the following lemma

Lemma 6.1. *For $\delta > 0$, if $m \geq \frac{8}{\min\{\lambda^{(a)},\lambda^{(b)}\}}\ln(2n^2/\delta)$, then with probability at least $1-\delta$ over the random initialization of θ_0*

$$\lambda_{\min}(G(\theta_0)) \geq \frac{3}{4}(\lambda_n^{(a)} + \beta^2\lambda_n^{(b)}) \tag{63}$$

This would lead to

$$\lambda_{\min}(G(\theta_t)) \geq \frac{3}{4}(\lambda_n^{(a)} + \beta^2\lambda_n^{(b)}) - \|G(\theta_t) - G(\theta_0)\|_F \tag{64}$$

If we prove that $\forall t \geq 0$, $t \in I$, where

$$I = \{t : \|G(\theta_t) - G(\theta_0)\| \leq \frac{1}{4}(\lambda_n^{(a)} + \beta^2\lambda_n^{(b)})\} \tag{65}$$

we would obtain that

$$\frac{d\hat{R}_n}{dt} \leq -2m\lambda_{\min}(G)\hat{R}_n \leq -m(\lambda_n^{(a)} + \beta^2\lambda_n^{(b)})\hat{R}_n(\theta_t) \tag{66}$$

which would conclude the proof using Grönwall's inequality. Thus we wish to prove that

$$t_0 = +\infty \text{ for } t_0 = \inf_{t\geq 0}\{t \notin I\} \tag{67}$$

By continuity, small values of t are contained in the set I. We suppose that $t_0 < +\infty$ and we try to find a contradiction. For this, we will need the following lemma

Lemma 6.2. *For any $\delta > 0$, if $m \geq 1024 \min\{\lambda_n^{(a)}, \lambda_n^{(b)}\}^{-2} \ln(n^2/\delta)$, then with probability at least $1-\delta$, for all $t \in [0, t_0]$,*

$$|a_k(t) - a_k(0)| \leq 2p_n \tag{68}$$

$$\|b_k(t) - b_k(0)\| \leq 2q_n \tag{69}$$

$$p_n := \frac{4\sqrt{\hat{R}_n(\theta_0)}}{m(\lambda_n^{(a)} + \beta^2\lambda_n^{(b)})} \text{ and } q_n := p_n^2 + \beta p_n \tag{70}$$

We can also bound the values p_n and q_n

Lemma 6.3. *For $\delta > 0$, if $m \geq 1024 \min\{\lambda_n^{(a)} + \lambda_n^{(b)}\}^{-2} \ln(n^2/\delta)$. If $\beta \leq 1$, with $C = 10(2 + \sqrt{\ln(1/\delta)})^2\delta$,*

$$p_n \leq \frac{C}{\sqrt{m}\lambda_n^{(a)}}\left(\frac{1}{\sqrt{m}} + \beta\right) \tag{71}$$

$$q_n \leq \frac{C}{\sqrt{m}(\lambda_n^{(a)})^2}\left(\frac{1}{\sqrt{m}} + \frac{2\beta}{\sqrt{m}} + \beta^2\right) + \frac{C\beta}{m\lambda_n^{(a)}} + \frac{C\beta^2}{\sqrt{m}\lambda_n^{(a)}} \tag{72}$$

If $\beta > 1$,

$$p_n \leq \frac{C}{\sqrt{m\lambda_n^{(a)}\lambda_n^{(b)}}} \tag{73}$$

$$q_n \leq \frac{C}{\sqrt{m}\lambda_n^{(b)}} \tag{74}$$

To find a contradiction, We first write

$$\|G(t_0) - G(0)\|_F \leq \left\|G^{(a)}(t_0) - G^{(a)}(0)\right\|_F + \left\|G^{(b)}(t_0) - G^{(b)}(0)\right\|_F \tag{75}$$

For the first term, using the fact that σ is 1-Lipschitz and that $\max_k \|b_k(t_0) - b_k(0)\| \leq q_n \leq 1$ using lemma 6.3 and the assumption on m, we have

$$\begin{aligned}
&\left|G_{i,j}^{(a)}(t_0) - G_{i,j}^{(a)}(0)\right| = \frac{1}{nm}\sum_{k=1}^{m}(\sigma(b_k(t_0)x_i)\sigma(b_k(t_0)x_j) - \sigma(b_k(0)x_i)\sigma(b_k(0)x_i)) \\
&= \frac{1}{nm}\sum_{k=1}^{m}(\sigma(b_k(t_0)x_i)(\sigma(b_k(t_0)x_j) - \sigma(b_k(0)x_j)) + \sigma(b_k(0)x_j)(\sigma(b_k(t_0)x_i) - \sigma(b_k(0)x_i)) \\
&\leq \frac{1}{nm}\sum_{k=1}^{m}\left(2\|b_k(t_0) - b_k(0)\| + \|b_k(t_0) - b_k(0)\|^2\right) \\
&\leq 3\frac{q_n}{n}
\end{aligned} \tag{76}$$

which implies that

$$\left\|G^{(a)}(t_0) - G^{(a)}(0)\right\|_F \leq 3q_n \tag{77}$$

For the Gram matrix $G^{(b)}$, we consider the expectation

$$\begin{aligned}
&n\mathbb{E}\left[\left|G^{(b)}_{i,j}(t_0) - G^{(b)}_{i,j}(0)\right|\right] \\
&\quad\leq \mathbb{E}\left[\frac{1}{m^2}\sum_{k=1}^{m}(a_k^2(t_0)\sigma'(b_k(t_0)x_i)\sigma'(b_k(t_0)x_j) - a_k^2(0)\sigma'(b_k(0)x_i)\sigma'(b_k(0)x_j)\right] \\
&\quad\leq \frac{1}{m^2}\sum_{k=1}^{m}\mathbb{E}\left[a_k^2(t_0)\left|(\sigma'(b_k(t_0)x_i)\sigma'(b_k(t_0)x_j) - \sigma'(b_k(0)x_j)\sigma'(b_k(0)x_j)\right|\right] + \mathbb{E}\left[\left|a_k^2(t_0) - a_k^2(0)\right|\right]
\end{aligned} \tag{78}$$

Where we used that $\sigma'(x) = \mathbf{1}\{x > 0\}$. Next, we observe that

$$\begin{aligned}
&\mathbb{P}\left(\left|(\sigma'(b_k(t_0)x_i)\sigma'(b_k(t_0)x_j) - \sigma'(b_k(0)x_j)\sigma'(b_k(0)x_j)\right| = 1\right) \\
&\quad\leq \mathbb{P}\left(\sigma'(b_k(t_0)x_i) \neq \sigma'(b_k(0)x_i)\right) + \mathbb{P}\left(\sigma'(b_k(t_0)x_j) \neq \sigma'(b_k(0)x_j)\right)
\end{aligned} \tag{79}$$

But since $\|b_k(t_0) - b_k(0)\| \leq q_n$, if $|x_i b_k(0)| > q_n$, then $\sigma'(b_k(t_0)x_i) = \sigma'(b_k(0)x_i)$. Using the fact that $\|x_i\| = 1$ and $b_k(0)$ follows the uniform distribution over the sphere, we know that $\mathbb{P}\left(|x_i b_k(0)| \leq q_n\right) \lesssim q_n$. Thus,

$$\mathbb{E}\left[\left|(\sigma'(b_k(t_0)x_i)\sigma'(b_k(t_0)x_j) - a_k^2(0)\sigma'(b_k(t_0)x_i)\right|\right] \lesssim q_n \tag{80}$$

We also have the following bounds using $a_k^2(0) = 1/2$ and lemma 6.3,

$$|a_k^2(t_0) - a_k^2(0)| \leq (\beta^2 + 2p_n)^2 - \beta^2 \lesssim q_n \tag{81}$$

$$a_k^2(t_0) \leq |a_k^2(t_0) - a_k^2(0)| + a_k^2(0) \lesssim \beta^2 + q_n \tag{82}$$

Combining these results, we obtain that

$$n\mathbb{E}\left[\left|G^{(b)}_{i,j}(t_0) - G^{(b)}_{i,j}(0)\right|\right] \lesssim q_n(\beta^2 + q_n) + q_n \lesssim (1+\beta^2)q_n \tag{83}$$

Where we used again that $q_n \leq 1$. Using Markov's inequality, we have $\mathbb{P}\{D_{i,j}\}geq 1 - \delta/n$, where $D_{i,j}$ is defined by

$$\left|G^{(b)}_{i,j}(t_0) - G^{(b)}_{i,j}(0)\right| \leq \frac{(1+\beta^2)q_n}{\delta} \tag{84}$$

Thus, with probability at least $1 - \delta$,

$$\left\|G^{(b)}(t_0) - G^{(b)}(0)\right\|_F \lesssim \frac{(1+\beta^2)nq_n}{\delta} \tag{85}$$

Finally, this implies that

$$\begin{aligned}
\|G(t_0) - G(0)\|_F &\leq \left\|G^{(a)}(t_0) - G^{(a)}(0)\right\|_F + \left\|G^{(b)}(t_0) - G^{(b)}(0)\right\|_F \\
&\leq 3q_n + \frac{(1+\beta^2)nq_n}{\delta}
\end{aligned} \tag{86}$$

Using lemma 6.3, if $m \gtrsim \min\{\lambda_n^{(a)} + \lambda_n^{(b)}\}^{-4} n^2\delta^{-1}\ln(n^2/\delta)$, we find that

$$\|G(t_0) - G(0)\|_F < \frac{1}{4}(\lambda_n^{(a)} + \beta^2\lambda_n^{(b)}) \tag{87}$$

which concludes the proof because it contradicts the definition of t_0. We prove here the lemmas that were used.

Lemma 6.1. Using $\|b_0\| = 1$, $\|x_i\| = 1$, the random variables $X_k^{i,j} := \sigma(b(0)x_i)\sigma(b(0)x_j)$ are contained in the interval $[0, 1]$. Writing $nG_{i,j}^{(a)}(0) = \sum_k X_k^{i,j}$, we can use Hoeffding's inequality to write

$$\mathbb{P}\left(\left|nG_{i,j}^{(a)}(0) - K_{i,j}^{(a)}\right| \leq \epsilon\right) = \mathbb{P}\left(\left|nG_{i,j}^{(a)}(0) - \mathbb{E}[nG_{i,j}^{(a)}(0)]\right| \leq \epsilon\right) \leq 1 - e^{-2m\epsilon^2} \tag{88}$$

This implies that with probability at least $(1 - e^{-2m\epsilon^2})^{2n^2}$,

$$\left\|G^{(a)} - K^{(a)}\right\| \leq \epsilon \tag{89}$$

$$\tag{90}$$

Similarly, with probability at least $(1 - e^{-2m\epsilon^2})^{2n^2}$.

$$\left\|G^{(b)} - K^{(b)}\right\| \leq \epsilon \tag{91}$$

Further, $(1 - e^{-2m\epsilon^2})^{2n^2} \geq 1 - 2n^2e^{-2m\epsilon^2}$. Finally, using Weyl's theorem, we obtain that

$$\begin{aligned}
\lambda_{\min}\left(G(0)\right) &\geq \lambda_{\min}\left(G^{(a)}(0)\right) + \beta^2\lambda_{\min}\left(G^{(b)}(0)\right) \\
&\geq \lambda_n^{(a)} + \beta^2\lambda_n^{(b)} + \left\|G^{(a)}(0) - K^{(a)}\right\|_F + \beta^2\left\|G^{(b)}(0) - K^{(b)}\right\|_F \\
&\geq \lambda_n^{(a)} + \beta^2\lambda_n^{(b)} - (1 + \beta^2)\epsilon
\end{aligned} \tag{92}$$

Taking $\epsilon = \frac{\min\{\lambda^{(a)}, \lambda^{(b)}\}}{4}$ completes the proof.

□

Lemma 6.2. In this lemma, $t \in I$, and so

$$R_n(a(t), b(t), c(t)) \leq e^{-tm(\lambda^{(a)} + \frac{\lambda^{(b)}}{m})} R_n(a(0), b(0), c(0)) \tag{93}$$

We write,

$$\begin{aligned}
|a_k(t) - a_k(0)| &\leq \int_0^t \left|\nabla_{a_k}\hat{R}_n(s)\right| ds \\
&= \int_0^t \left|\frac{1}{n}\sum_{i=1}^n u_i\sigma(b_k x_i)\right| ds \\
&\leq \int_0^t \frac{1}{n}\|u\|\,\|b_k\|\, ds \\
&= \int_0^t 2\|b_k\|\sqrt{\hat{R}_n} ds \\
2 &\leq \int_0^t \max_{0\leq s\leq t}\|b_k(s)\|\, e^{-sm(\lambda^{(a)} + \frac{\lambda^{(b)}}{m})}\sqrt{R_n(0)} ds \\
&= \frac{4\max_{0\leq s\leq t}\|b_k(s)\|\sqrt{R_n(0)}}{m(\lambda^{(a)} + \frac{\lambda^{(b)}}{m})} = p_n \max_{0\leq s\leq t}\|b_k(s)\|
\end{aligned} \tag{94}$$

Similarly, we find that

$$\|b_k(t) - b_k(0)\| \leq p_n \max_{0\leq s\leq t}|a_k(s)| \tag{95}$$

Combining these results,

$$\max_{0\leq s\leq t}|a_k(s)| \leq |a_k(0)| + p_n(\|b_k(0)\| + p_n \max_{0\leq s\leq t}|a_k(s)|) \tag{96}$$

using $\|b_k(0)\| = 1$ and rearranging,

$$\max_{0\leq s\leq t}|a_k(s)| \leq \frac{a_k(0) + p_n}{1 - p_n^2} \leq 2(p_n + \beta) \tag{97}$$

where we used that $p_n \leq 1/2$ for $m \geq \frac{16}{\lambda_n^{(a)}} \max\{1, \frac{4(2+\sqrt{\ln(1/\delta)})^2}{\lambda_n^{(b)}}\}$. For b this implies that

$$\|b_k(t) - b_k(0)\| \leq 2p_n(p_n + \beta) \leq 1 \tag{98}$$

if we suppose that $m \geq 16 \max\{\frac{1}{\sqrt{\lambda_n^{(a)}\lambda_n^{(b)}}}, \frac{64(2+\sqrt{\ln(1/\delta)})^2}{(\lambda_n^{(b)})^2}\}$ for the second inequality. Finally, we proved that

$$|a_k(t) - a_k(0)| \leq p_n \max_{0\leq s\leq t}\|b_k(s)\| \leq p_n(1 + \max_{0\leq s\leq t}\|b_k(t) - b_k(0)\|) \leq 2p_n \tag{99}$$

which concludes the proof.

□

Lemma 6.3. It is a direct consequence of the results used in lemma 6.2 □

□