SORBONNE UNIVERSITÉ

Non-parametric regression using deep neural networks with ReLU activation function

Minimax convergence rates under composition assumption

Author :
Nacef Ben Mansour
Tomas Maiuri

Professor :
Gérard Biau

Project completed as part of the Statistical Learning course.

January 11, 2025

Introduction

Deep Neural Networks have emerged as powerful tools in modern statistics and machine learning, demonstrating remarkable success in areas such as computer vision, natural language processing or regression problems. They offer exceptional flexibility in modeling complex relationships between variables while effectively handling large datasets.

This course focuses on estimation in the nonparametric regression model using Deep Neural Networks (DNNs) with specific settings.

First, the mathematical framework and essential definitions for regression analysis are established in Sections 1.1 and 1.2. Then, in sections 1.3 and 1.4, we will mention classical methods of nonparametric estimation and specify their performance in terms of minimax convergence rates. However, one particular limitation of their use, namely the curse of dimensionality, will rapidly be encountered. Therefore, we need to introduce DNNs as a solution to this problem.

This course focuses specifically on Feed Forward Neural Networks, a fundamental type of neural networks that will be presented in chapter 2. In a first part, we will justify our focus on the Rectified Linear Unit (ReLU) activation function, after which we will describe some key learning methods in the case of neural networks. The chapter ends with the construction of the function space in which the estimators considered in our convergence results will live.

Once all the important notions have been detailed, we will finish preparing our set up at the beginning of chapter 3. After that, it will eventually be shown that estimators based on sparsely connected DNNs with ReLU activation function and properly configured network architecture achieve the minimax rates of convergence (up to $\log(n)$-factors) under a general composition assumption. This result establishes the statistical optimality of neural networks in our nonparametric regression settings.

Note for the attention of Mr. Biau :

Following the instructions, we constructed a course at the level of a Master's student. So, in order to keep within the 15-page limit, we decided to make a trade-off between setting up a coherent structure of reasoning and providing detail on each element of the proof of the main result. Because of its length, we sometimes only give the central points for understanding certain steps without writing them down. This might require a little work on the part of a student if they wanted to understand it in depth. We feel that this does not go against the nature of a course handout.

Contents

List of Figures

List of Tables

1 Nonparametric Regression models: Framework and Basics

1.1 Definition of the framework

We consider a scenario with n pairs of i.i.d. random variables $\mathcal{D}_n := \{(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n)\}$ such that

$$Y_i = f_0(\mathbf{X}_i) + \xi_i, \quad \text{for all } i = 1, \dots, n. \tag{1}$$

where $\mathbf{X}_i \in [0,1]^d$ are i.i.d. random vectors and $Y_i \in \mathbb{R}$. We also assume that $\xi_i \sim \mathcal{N}(0,1)$ are i.i.d. standard normal random variables representing noise that is independent of $\mathbf{X}_i$.

Here, the function $f_0 : [0,1]^d \to \mathbb{R}$, known as the regression function, is unknown. The goal in the problem of nonparametric regression is to estimate f_0, under the assumption that it belongs to a specified nonparametric class of functions $\mathcal{F}$.
For example, $\mathcal{F}$ might represent:

$$\mathcal{F} = \{f : [0,1]^d \to \mathbb{R} \mid f \text{ is continuous}\}, \quad \text{or}$$

$$\mathcal{F} = \left\{f : [0,1]^d \to \mathbb{R},\ f^{\lfloor\beta\rfloor} \text{ exists},\ \forall x, y \in A,\ |f^{\lfloor\beta\rfloor}(x) - f^{\lfloor\beta\rfloor}(y)| \leq L|x-y|^{\beta - \lfloor\beta\rfloor}\right\}$$

where the latter defines the set of Hölder functions $\mathcal{H}(\beta, L)$ for $\beta, L \in \mathbb{R}^+$.

Definition 1.1 (Estimator)**.** *A statistic $T(\mathbf{X})$ is a measurable function of the random object $\mathbf{X}$ that can possibly depend on other known parameters, but not on f_0. An estimator of f_0 is a statistic $\widehat{f} = \widehat{f}(\mathbf{X})$ intended to approximate f_0.*

1.2 Loss function & Risk

In this section, we briefly review several common definitions of risk that will become essential later. We then recall the bias-variance trade-off, which will help to analyze and optimize convergence rates.
For more detail, the reader may refer to the course material by Dion-Blanc (2024) and Tsybakov (2009).

To evaluate the performance of an estimator, we introduce a **loss function** ℓ to quantify the penalty associated with this discrepancy.

Definition 1.2 (Loss Function)**.** *A loss function $\ell : \mathbb{R}\times\mathbb{R} \to \mathbb{R}^+$ is a measurable, non-zero, convex function such that $\ell(x,x) = 0$. $\ell(z,y)$ quantifies the loss incurred when predicting z while the true output is y.*

The performance of the estimator $\hat{f}_n$ is then evaluated using its risk.

Definition 1.3 (Risk)**.** *We define the risk as*

$$R(\widehat{f}_n, f) := \mathbb{E}[\ell(\widehat{f}_n(X), f(X))],$$

where $\mathbb{E}$ denotes the expectation taken with respect to $\mathcal{D}_n$ sampled from our regression model and X, with $X \sim X_1$ being independent of $\mathcal{D}_n$.

More particularly, we will consider the quadratic loss $\ell : (x,y) \mapsto (x-y)^2$ and the risk :

$$R(\widehat{f}_n, f) = \mathbb{E}[(\widehat{f}_n(X) - f(X))^2] = \mathbb{E}\int_{\mathbb{R}^d} (\widehat{f}_n(x) - f(x))^2 dP_X(x),$$

which can be seen as the mean squared error integrated on the distribution of X.

Definition 1.4. *For any estimator $\widehat{f}_n$ that returns a network in the class $\mathcal{F}$ with regression function f_0, we define the corresponding quantity*

$$\Delta_n := \Delta_n(\widehat{f}_n, f_0, \mathcal{F}) := E_{f_0}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{f}_n(\mathbf{X}_i))^2 - \inf_{f\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^{n}(Y_i - f(\mathbf{X}_i))^2\right], \quad (2)$$

The sequence Δ_n measures the difference between the expected empirical risk of $\widehat{f}_n$ and the global minimum over all networks in the class. Notice that $\Delta_n \geq 0$ and $\Delta_n = 0$ if $\widehat{f}_n$ is an empirical risk minimizer.

Proposition 1.1. *For an estimator $\widehat{f}_n$ in the class $\mathcal{F}$ and an empirical risk minimizer $\hat{f}_n^{ERM}$, we have the following result :*

$$\Delta_n = R(\hat{f}_n, f_0) - R(\hat{f}_n^{ERM}, f_0)$$

Proof. First, for all $i \leq n$, we can write:

$$\begin{aligned}(Y_i - \hat{f}_n(X_i))^2 &= (Y_i - f_0(X_i) + f_0(X_i) - \hat{f}_n(X_i))^2 \\ &= (Y_i - f_0(X_i))^2 + (f_0(X_i) - \hat{f}_n(X_i))^2 + 2(Y_i - f_0(X_i))(f_0(X_i) - \hat{f}_n(X_i))\end{aligned}$$

Now, in our model $Y_i - f_0(X_i) = \varepsilon_i$, where the ε_i are i.i.d. with $E[\varepsilon_i] = 0$ and $\mathrm{Var}(\varepsilon_i) = \sigma^2$, taking the expectation gives:

$$E_{f_0}[(Y_i - \hat{f}_n(X_i))^2] = \sigma^2 + E_{f_0}[(f_0(X_i) - \hat{f}_n(X_i))^2] = \sigma^2 + R(\hat{f}_n, f_0)$$

Similarly, since $\widehat{f}_n^{\mathrm{ERM}}$ minimizes the empirical risk, and given that the pairs (X_i, Y_i) are i.i.d., we have:

$$E_{f_0}\left[\inf_{f\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^{n}(Y_i - f(X_i))^2\right] = E_{f_0}[(Y_1 - \hat{f}_n^{\mathrm{ERM}}(X_1))^2] = \sigma^2 + R(\hat{f}_n^{\mathrm{ERM}}, f_0)$$

Finally, by subtracting the expressions:

$$\Delta_n = [R(\hat{f}_n, f_0) + \sigma^2] - [R(\hat{f}_n^{\mathrm{ERM}}, f_0) + \sigma^2] = R(\hat{f}_n, f_0) - R(\hat{f}_n^{\mathrm{ERM}}, f_0)$$

□

Thus, the term $\Delta_n(\hat{f}_n, f_0, \mathcal{F})$ is the key quantity that together with the minimax estimation rate sharply determines the convergence rate of $\hat{f}_n$ (up to $\log n$-factors).

Definition 1.5 (Minimax)**.** *For a class of functions $\mathcal{F}$, the minimax risk is defined as*

$$\inf_{\widehat{f}}\sup_{f\in\mathcal{F}}\{R(f, \widehat{f})\},$$

where the inf is taken over all estimators.

Definition 1.6 (Oracle)**.** *The "oracle" estimator, denoted by $\widehat{f}^*$, is defined by:*

$$\widehat{f}^* = \arg\min_{\widehat{f}} R(f, \widehat{f})$$

Remark 1.1. *It is inaccessible (since f is unknown) and is therefore not a true estimator.*

Definition 1.7 (Oracle-type inequality)**.** *An estimator $\widehat{f}_n$ satisfies an oracle-type inequality if there exists a constant $C > 0$ such that:*

$$\mathbb{E}\left[R(f_0, \widehat{f}_n)\right] \leq C \inf_{\widehat{f}'} R(f_0, \widehat{f}') + R_n,$$

where $R(f, \widehat{f})$ denotes the risk, R_n is a remainder term depending on the sample size n, and the infimum is taken over a suitable class of estimators.

1.3 Estimation methods

Here, we present two classical estimation methods in the non-parametric regression model. We adopt the notation from the course of Dion-Blanc (2024)(Sections 1.3 and 3.1). We will not elaborate on all the concepts, and we encourage the reader to explore the cited sections for further details.

1.3.1 Kernel-based estimators

To estimate a density f on $\mathbb{R}$, a kernel-based estimator is defined, for $h > 0$ and $x \in \mathbb{R}$, as:

Definition 1.8 (Density Estimator)**.**

$$\widehat{f}_h(x) := \frac{1}{nh} \sum_{j=1}^{n} K\left(\frac{x - X_j}{h}\right) = \frac{1}{n} \sum_{j=1}^{n} K_h(x - X_j),$$

where $K_h := (1/h)K(\cdot/h)$, with $K : \mathbb{R} \to \mathbb{R}$ being a kernel function and $h > 0$ referred to as the bandwidth.

We now extend the kernel density estimation concept to the regression setting. Consider the regression model defined in (1), where the goal is to estimate the unknown regression function f.

The **Nadaraya-Watson estimator**, based on kernel density estimation methods, is defined as:

$$\widehat{f}_{h,h'}(x) = \frac{\sum_{j=1}^{n} K_h(x - X_j) Y_j}{\sum_{j=1}^{n} K_{h'}(x - X_j)},$$

with $K : \mathbb{R}^d \to \mathbb{R}$ being a kernel function and $\mathbf{h} = (h, h') > 0$ the bandwidth parameter vector.

1.3.2 Wavelet-based estimators

Suppose $f \in L^2(A)$, where $A \subseteq \mathbb{R}$. Let $(\varphi_j)_{j \geq 1}$ be an orthonormal basis of $L^2(A)$. Then, any function f can be represented as:

$$f = \sum_{j \geq 1} \langle f, \varphi_j \rangle \varphi_j,$$

where $\langle f, \varphi_j \rangle = \int f(x)\varphi_j(x)dx$ denotes the inner product in $L^2(A)$.

Definition 1.9 (Projection estimator)**.** *Let $(\varphi_j)_{j \geq 1}$ be an orthonormal basis of $L^2(A)$, and let $X_1, \ldots, X_n$ be an i.i.d. sample with the same distribution as X. The **projection estimator** of f is given by:*

$$\widehat{f}_m(x) = \sum_{j=1}^{m} \widehat{a}_j \varphi_j(x) \quad \textit{with} \quad \widehat{a}_j = \frac{1}{n} \sum_{i=1}^{n} \varphi_j(X_i),$$

where $\widehat{a}_j$ estimates the theoretical coefficients $\langle f, \varphi_j \rangle$ through empirical averaging.

Wavelets are a special class of orthonormal bases of $L^2(A)$. More formally, following Tsybakov (2009) :

Definition 1.10 (Wavelet Basis)**.** *Let $\psi : \mathbb{R} \to \mathbb{R}$ be a sufficiently smooth function with compact support. Define a system of functions as follows:*

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k), \quad j, k \in \mathbb{Z}$$

where j controls the dilation (scale) and k the translation (location).
Under appropriate conditions on ψ, this system forms an orthonormal basis in $L^2(\mathbb{R})$.

Remark 1.2. *For all $f \in L^2(\mathbb{R})$, we can write:*

$$f = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \theta_{j,k}\psi_{j,k}, \quad \theta_{j,k} = \int f\psi_{j,k}$$

By combining the projection estimator with the wavelet basis, we introduce the wavelet-based estimator.

Definition 1.11 (Wavelet Regression Estimator)**.** *Consider the regression model (1) with observations $(X_i, Y_i)_{1 \le i \le n}$. Given the wavelet basis $\{\psi_{j,k}\}_{j,k \in \mathbb{Z}}$, the wavelet estimator of the regression function f at resolution level J is:*

$$\widehat{f}_J(x) = \sum_{j=-J}^{J} \sum_{k=-\infty}^{\infty} \widehat{\theta}_{j,k}\psi_{j,k}(x)$$

where the empirical wavelet coefficients are:

$$\widehat{\theta}_{j,k} = \frac{1}{n}\sum_{i=1}^{n} Y_i \psi_{j,k}(X_i)$$

In practice, the infinite sum over k is truncated based on the support of ψ.

Remark 1.3. *The parameter J determines the finest resolution level included in the estimator, controlling the balance between fidelity to the data and smoothness of the estimate.*

1.4 Rates of convergence for nonparametric estimation methods

We now define the notion of convergence rate that will be central in the following.

Definition 1.12 (Convergence Rate)**.** *A sequence $(\phi_n)_n$ is called a convergence rate of an estimator $\widehat{f}$ with sample size n on a class $\mathcal{F}$ for a risk R if:*

1. *$\phi_n \to 0$ as $n \to \infty$*
2. *$\exists C > 0$, $\forall n \in \mathbb{N}$, $\sup_{f \in \mathcal{F}} R(\widehat{f}, f) \le C\phi_n^2$*

Then for the minimax rate, we should define it carefully as well:

Definition 1.13 (Minimax Rate)**.** *A sequence $(\psi_n)_n$ is called a minimax rate of an estimator $\widehat{f}$ with sample size n on a class $\mathcal{F}$ for a risk R if:*

1. *$\psi_n \to 0$ as $n \to \infty$*
2. *$\exists c, C > 0$, $\forall n \in \mathbb{N}$, $c\psi_n^2 \le \inf_{\widehat{f}} \sup_{f \in \mathcal{F}} R(\widehat{f}, f) \le C\psi_n^2$*

1.4.1 Optimal convergence rates for specific methods

For the **Nadaraya-Watson estimator** with $h = h' > 0$, assuming $f \in \mathcal{N}(\beta, L)$ (Nikol'skii space, , which can be viewed as an integrated Hölder space) and K is a kernel of order $\lfloor \beta \rfloor$, then choosing bandwidth

$$h \asymp n^{-1/(2\beta+1)}$$

leads to the classical minimax nonparametric rate; cf. Stone (1982):

$$\inf_{\widehat{f}_h} \sup_{f \in \mathcal{N}(\beta,L)} R(\widehat{f}_h, f) \asymp n^{-2\beta/(2\beta+1)}$$

The **Wavelet Regression estimator** achieves the same classical minimax rate as the **Nadaraya-Watson** estimator. This rate is optimal for estimating functions in Sobolev spaces under L_2-risk.

1.4.2 Curse of dimensionality

The previous rates highlight a fundamental challenge in nonparametric statistics: as the dimension d of the input space increases, the convergence rates deteriorate significantly. For both kernel and wavelet estimators in d dimensions, assuming $f \in \mathcal{N}(\beta, L)$, the minimax rate becomes:

$$n^{-2\beta/(2\beta+d)}$$

To illustrate this phenomenon, consider a function with smoothness $\beta = 2$:

- For $d = 1$, the rate is $n^{-4/5}$
- For $d = 5$, it deteriorates to $n^{-4/9}$

This implies that to maintain the same estimation precision as dimension increases, the required sample size grows exponentially with the dimension : a phenomenon known as the curse of dimensionality.

1.4.3 Deep Neural Networks as a solution

While these classical nonparametric approaches suffer from the curse of dimensionality, Deep Neural Networks (DNNs) have emerged as a powerful alternative. Bauer and Kohler (2019) and Schmidt-Hieber (2020) demonstrated that DNNs can achieve convergence rates that are essentially independent of the dimension d when the target function possesses certain structural properties, such as compositional structure or hierarchical relationships between variables.

2 Deep Neural Network

Neural networks, initially inspired by biological neural systems, have evolved into powerful tools in machine learning and artificial intelligence. Following the pioneering work of McCulloch and Pitts (1943), and the fundamental breakthrough of backpropagation Linnainmaa (1970), these mathematical models have become ubiquitous across diverse fields, from signal processing to computer vision.

This work focuses on their theoretical properties, particularly the convergence characteristics and estimation methods.

2.1 Feed Forward Neural Networks architecture

In this course, we focus on Feed Forward Neural Networks (FFNNs), which constitute the fundamental architecture in neural network theory. Despite their structural simplicity, FFNNs are universal approximators, possessing the remarkable ability to approximate any Borel measurable function; cf. Hornik (1989). FFNNs have introduced the concept of a hidden layer, and this requires us to choose a fixed nonlinear function called the activation function that will be used to compute the hidden layer values.

Definition 2.1. *For $v = (v_1, \ldots, v_r) \in \mathbb{R}^r$ and σ an activation function, we define the shifted activation function $\sigma_v : \mathbb{R}^r \to \mathbb{R}^r$ as*

$$\sigma_v \begin{pmatrix} y_1 \\ \vdots \\ y_r \end{pmatrix} = \begin{pmatrix} \sigma(y_1 - v_1) \\ \vdots \\ \sigma(y_r - v_r) \end{pmatrix}.$$

The activation functions in the neurons are essentially modeling a threshold that decides whether the information that a neuron got will be relevant for the further calculations or not. If the information is relevant it will be passed on to the next layer until the output layer is reached.

Definition 2.2. *For $(L, p) \in \mathbb{N} \times \mathbb{N}^{L+2}$, a neural network with network architecture (L, p) is any function of the form*

$$f : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}, \; x \mapsto f(x) = W_L \sigma_{v_L} W_{L-1} \sigma_{v_{L-1}} \ldots W_1 \sigma_{v_1} W_0 x, \tag{3}$$

where W_i is a $p_{i+1} \times p_i$ weight matrix, $v_i \in \mathbb{R}^{p_i}$ and is a shift vector.

Network functions are therefore built by alternating matrix-vector multiplications with the action of the nonlinear activation function σ.

Remark 2.1. *The network architecture $(L, \boldsymbol{p})$ consists of a positive integer L, called the number of hidden layers or depth, and a width vector $\boldsymbol{p}$. The input layer is the first layer and the output layer the last layer. Note that by convention we refer to a neural network as DNN as long as $L \geq 1$.*

Remark 2.2. *To fit networks to data generated from the d-variate nonparametric regression model we must have $p_0 = d$ and $p_{L+1} = 1$.*

To illustrate this definition, Figure 1 provides a graphical representation of a neural network architecture. Neural networks are visualized as directed acyclic graphs, where nodes (or neurons) are systematically arranged in layers from the input layer to the output layer. Each neuron represents a computational unit that performs a scalar product of the incoming signal with a weight vector, followed by a shift and an application of the activation function.

2.2 More on ReLU activation function

We focus on networks employing the Rectified Linear Unit (ReLU) activation function, defined as:

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

While there exists a vast array of activation functions (sigmoid and *tanh* for example), ReLU has emerged as a cornerstone in deep learning due to its mathematical simplicity and computational efficiency; cf. LeCun et al. (2015).

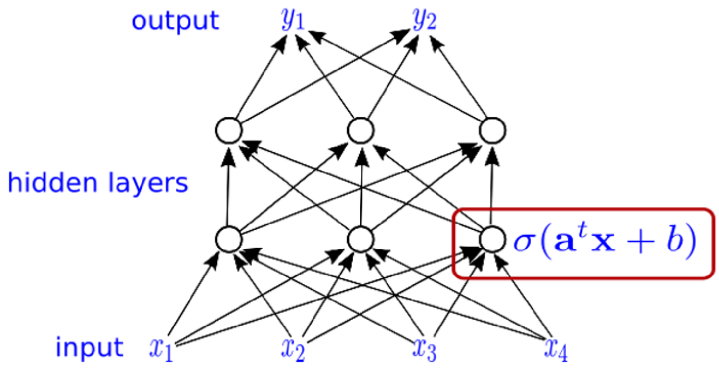

Figure 1: Representation as a direct graph of a network with two hidden layers $L = 2$ and width vector $p = (4, 3, 3, 2)$.

The ReLU activation function is defined very simply. Its piecewise linear nature makes it computationally efficient and easy to implement. ReLU avoids vanishing gradient issues and propagates gradients effectively for $x > 0$ (note that in this case $\sigma'(x) = \mathbb{1}_{x>0}$), enabling efficient training of deep networks; cf. Inturrisi et al. (2021). Finally, the projection property $\sigma \circ \sigma = \sigma$ of this function allows for depth synchronization and facilitates combining subnetworks with differing depths.

We finish this subsection with a proposition showing that using the ReLU activation function is not much different from using any other piecewise linear activation function with finitely many breakpoints. This is formalized in the following result.

Proposition 2.1. *Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous piecewise linear function with M breakpoints, where $1 \leq M < \infty$.*

(a) Let ξ be a network with the activation function ρ, having depth L, W weights and U computation units. Then there exists a ReLU network η that has depth L, not more than $(M+1)2W$ weights and not more than $(M+1)U$ units, and that computes the same function as ξ.

(b) Conversely, let η be a ReLU network of depth L with W weights and U computation units. Let D be a bounded subset of $\mathbb{R}^n$, where n is the input dimension of η. Then there exists a network with the activation function ρ that has depth L, $4W$ weights and $2U$ units, and that computes the same function as η on the set D.

Proof. For reasons of brevity, and because it is the first point that really matters, point (b) is not proved here, but the interested reader can refer to section 2 of Yarotsky (2017)

(a) Let $a_1 < \cdots < a_M$ be the breakpoints of ρ, i.e., the points where its derivative is discontinuous: $\rho'(a_k+) \neq \rho'(a_k-)$. We can then express ρ via the ReLU function σ, as a linear combination

$$\rho(x) = c_0\sigma(a_1 - x) + \sum_{m=1}^{M} c_m\sigma(x - a_m) + h$$

with appropriately chosen coefficients $(c_m)_{m=0}^{M}$ and h. It follows that computation performed by a single ρ-unit,

$$x_1, \ldots, x_N \mapsto \rho(\sum_{k=1}^{K} w_k x_k + b),$$

can be equivalently represented by a linear combination of a constant function and com-

putations of $M+1$ σ-units,

$$x_1, \ldots, x_N \mapsto \begin{cases} \sigma(\sum_{k=1}^{N} w_k x_k + b - a_m) & m = 1, \ldots, M \\ \sigma(a_1 - b - \sum_{k=1}^{N} w_k x_k) & m = 0 \end{cases}$$

(here m is the index of a ρ-unit). We can then replace one-by-one all the ρ-units in the network ξ by σ-units, without changing the output of the network. Obviously, these replacements do not change the network depth. Since each hidden unit gets replaced by $M+1$ new units, the number of units in the new network is not greater than $M+1$ times their number in the original network. Note also that the number of connections in the network is multiplied, at most, by $(M+1)^2$. Indeed, each unit replacement entails replacing each of the incoming and outgoing connections of this unit by $M+1$. New connections, and each connection is replaced twice: as an incoming and as an outgoing one. These considerations imply the claimed complexity bounds for the resulting σ-network η. □

Furthermore, it has been showed that piecewise linear units improve the learning performance of DNNs; cf. Inturrisi et al. (2021). This justifies our limited attention to the ReLU activation function.

2.3 Learning in DNN: Backpropagation

Given a network function with form (3), the network parameters are the entries of the matrices $(W_j)_{j=0,\ldots,L}$ and vectors $(v_j)_{j=1,\ldots,L}$. These parameters need to be estimated/learned from the data.
Note : In the following, no upper bound on the number of network parameters is needed. Thus, we consider high-dimensional settings with more parameters than training data.

The learning process starts by defining a **risk function** $R(\widehat{f}_n, f_0)$, which evaluates the performance of an estimator $\widehat{f}_n$ (see Section 1.2).
However, in machine learning, the true data distribution is typically unknown. Therefore, we cannot directly minimize the risk $R(\widehat{f}_n, f)$. Instead, we minimize the empirical risk R_{emp}, which serves as an approximation of the true risk $R(\widehat{f}_n, f_0)$. The empirical risk R_{emp} is computed over a finite training dataset $\mathcal{D}_n$ as:

$$R_{emp}(\widehat{f}_n, f_0) = \frac{1}{n} \sum_{i=1}^{n} \ell(\widehat{f}_n(x_i), f_0(x_i).$$

In the **forward pass** computes the network output $\widehat{f}_n(x)$ for a given input x by alternating matrix-vector multiplications with the action of the non-linear activation function σ (see Definition 2.2).This allows evaluation of the empirical risk R_{emp} over the training dataset $\mathcal{D}_n$.

Once the **forward pass** is complete and R_{emp} is computed, **backpropagation** begins by computing the gradient of the loss with respect to the output of the final layer. The gradients are then propagated backward through the network using the chain rule.

After computing the gradients for all layers, the network parameters $(W_j, v_j)_{j=0,\ldots,L}$ are updated using **gradient descent**. Given a learning rate η, the updates are performed as:

$$W_j \leftarrow W_j - \eta \frac{\partial R_{emp}}{\partial W_j}, \quad v_j \leftarrow v_j - \eta \frac{\partial R_{emp}}{\partial v_j}, \quad j = 1, \ldots, L$$

The forward and backward passes are repeated for many iterations (epochs) over the training data until the network converges to a set of parameters that minimize the R_{emp}.

Typically, techniques such as mini-batch gradient descent, Adam optimizer (see Kingma and Ba (2015)), or stochastic gradient descent (SGD) (see Bottou (2010)) are used to improve convergence speed and stability.

2.4 Space of functions and network sparsity

To be more in line with what is observed in practice, we consider networks with all parameters bounded by one. This constraint can be easily built into the deep learning algorithm by projecting the network parameters in each iteration onto the interval $[-1, 1]$.

Thus, given a network architecture (L, P), we introduce the space of network functions with parameters bounded by one :

$$\mathcal{F}(L, \mathbf{p}) := \left\{ f \text{ of the form (3)} : \max_{j=0,\dots,L} \|W_j\|_\infty \vee \|\mathbf{v}_j\|_\infty \leq 1 \right\}, \tag{4}$$

where $\|W_j\|_\infty$ is the maximum-entry norm of W_j and with the convention that $\mathbf{v}_0$ is the zero vector.

Furthermore, in deep learning, sparsity of the neural network is a key feature that is enforced through regularization or specific forms of networks; cf. Denil et al. (2014). Dropout for instance sets randomly units to zero and has the effect that each unit will be active only for a small fraction of the data; cf. Srivastava et al. (2014).

In this course we model the network sparsity assuming that there are only few nonzero/active network parameters. We define s-sparse networks as follows:

$$\mathcal{F}(L, \mathbf{p}, s, F) := \left\{ f \in \mathcal{F}(L, \mathbf{p}) : \sum_{j=0}^{L} \|W_j\|_0 + |\mathbf{v}_j|_0 \leq s, \| |f|_\infty \|_\infty \leq F \right\}, \tag{5}$$

where $\|W_j\|_0$ denotes the number of nonzero entries of W_j and $\| |f|_\infty \|_\infty$ stands for the sup-norm of the function $x \mapsto |f(x)|_\infty$. We consider cases where the number of network parameters s is small compared to the total number of parameters in the network.

3 Rate of convergence using DNN

3.1 Regularity classes for composition

Recall that the theoretical performance of any nonparametric estimator, including neural networks, depends on the properties of the underlying function class being estimated. For β-smooth regression functions, the minimax estimation rate for the prediction error is $n^{-2\beta/(2\beta+d)}$. Since the input dimension d in neural networks is typically large, these rates are impractically slow.

We consider a function class with low-dimensional structure leading to dimension-free estimation rates. The regression function f_0 is assumed to be a composition of functions:

$$f_0 = g_q \circ g_{q-1} \circ \cdots \circ g_1 \circ g_0 \tag{6}$$

with $g_i : [a_i, b_i]^{d_i} \to [a_{i+1}, b_{i+1}]^{d_{i+1}}$. Here, $g_i = (g_{ij})_{j=1,\dots,d_{i+1}}$ denotes the components of g_i, and t_i is the maximal number of variables each g_{ij} depends on. We assume that $t_i \leq d_i$ for all i, and the number of layers L can differ from q.
For example, consider $f_0(x_1, x_2, x_3) = g_{11}(g_{01}(x_1, x_3), g_{02}(x_1, x_2))$ where $d_0 = 3$, $t_0 = 2$, $d_1 = t_1 = 2$, $d_2 = 1$.

In our regression model (1), we have $d_0 = d$, $a_0 = 0$, $b_0 = 1$ and $d_{q+1} = 1$.

Now recall that we define the ball of β-Hölder functions with radius K as :

$$\mathcal{C}_r^\beta(D, K) = \left\{ f : D \subset \mathbb{R}^r \to \mathbb{R} : \sum_{|\boldsymbol{\alpha}| \leq \beta} \|\partial^{\boldsymbol{\alpha}} f\|_\infty + \sum_{|\boldsymbol{\alpha}| = \lfloor \beta \rfloor} \sup_{\substack{\mathbf{x}, \mathbf{y} \in D \\ \mathbf{x} \neq \mathbf{y}}} \frac{|\partial^{\boldsymbol{\alpha}} f(\mathbf{x}) - \partial^{\boldsymbol{\alpha}} f(\mathbf{y})|}{\|\mathbf{x} - \mathbf{y}\|_\infty^{\beta - \lfloor \beta \rfloor}} \leq K \right\},$$

where we used multi-index notation, that is, $\partial^{\boldsymbol{\alpha}} = \partial^{\alpha_1} \cdots \partial^{\alpha_r}$ with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r) \in \mathbb{N}^r$ and $|\boldsymbol{\alpha}| := |\alpha|_1$.

We assume that each of the functions g_{ij} has Hölder smoothness β_i. Since g_{ij} is also t_i-variate, $g_{ij} \in \mathcal{C}_{t_i}^{\beta_i}([a_i, b_i]^{t_i}, K_i)$, and the underlying function space becomes

$$\mathcal{G}(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, K) := \Big\{ f = g_q \circ \cdots \circ g_0 : g_i = (g_{ij})_j : [a_i, b_i]^{d_i} \to [a_{i+1}, b_{i+1}]^{d_{i+1}}, \\ g_{ij} \in \mathcal{C}_{t_i}^{\beta_i}([a_i, b_i]^{t_i}, K_i), \text{ for some } |a_i|, |b_i| \leq K \Big\}$$

with $\mathbf{d} := (d_0, \ldots, d_{q+1})$, $\mathbf{t} := (t_0, \ldots, t_q)$, and $\boldsymbol{\beta} := (\beta_0, \ldots, \beta_q)$.

After imposing conditions on the smoothness of each g_i, the smoothness of f_0 can be effectively described using the following notations.

$$\beta_i^* := \beta_i \prod_{\ell=i+1}^{q} (\beta_\ell \wedge 1) \tag{7}$$

via the rate

$$\phi_n := \max_{i=0,\ldots,q} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}}. \tag{8}$$

3.2 Minimax convergence rates

3.2.1 Preparatory work

We begin this section with an overview of some useful properties of network function classes. For the approximation of a function by a network, we first construct smaller networks, computing simpler objects, after which we combine those networks.

Composition: Let $\mathbf{p} = (p_0, \ldots, p_{L+1})$ and $\mathbf{p}' = (p_0', \ldots, p_{L'+1}')$. Suppose that $f \in \mathcal{F}(L, \mathbf{p})$ and $g \in \mathcal{F}(L', \mathbf{p}')$ with $p_{L+1} = p_0'$. For a vector $\mathbf{v} \in \mathbb{R}^{p_{L+1}}$, we define the composed network $g \circ \sigma_{\mathbf{v}}(f)$ which is in the space $\mathcal{F}(L + L' + 1, (p_0, p_1', \ldots, p_{L'+1}'))$.

Enlarging: $\mathcal{F}(L, \mathbf{p}, s) \subset \mathcal{F}(L, \mathbf{q}, s')$ whenever $\mathbf{p} \leq \mathbf{q}$ componentwise and $s \leq s'$.

Additional layers/depth synchronization: To synchronize the number of hidden layers for two networks, we can add additional layers with identity weight matrix, such that

$$\mathcal{F}(L, \mathbf{p}, s) \subset \mathcal{F}(L + q, (\underbrace{p_0, \ldots, p_0}_{q \text{ times}}, \mathbf{p}), s + q p_0). \tag{9}$$

Parallelization: Suppose that f, g are two networks with the same number of hidden layers and the same input dimension, i.e. $f \in \mathcal{F}(L, \mathbf{p})$ and $g \in \mathcal{F}(L, \mathbf{p}')$ with $p_0 = p_0'$. The parallelized network (f, g) computes f and g simultaneously in a joint network in the class $\mathcal{F}(L, (p_0, p_1 + p_1', \ldots, p_{L+1} + p_{L'+1}'))$.

Removal of inactive nodes: We have

$$\mathcal{F}(L, \mathbf{p}, s) = \mathcal{F}(L, (p_0, p_1 \wedge s, p_2 \wedge s, \ldots, p_L \wedge s, p_{L+1}), s). \tag{10}$$

To see this, let $f(\mathbf{x}) = W_L \sigma_{\mathbf{v}_L} W_{L-1} \ldots \sigma_{\mathbf{v}_1} W_0 \mathbf{x} \in \mathcal{F}(L, \mathbf{p}, s)$. If all entries of the jth column of W_i are zero, then we can remove this column together with the jth row in W_{i-1} and the jth entry of $\mathbf{v}_i$ without changing the function. This shows then that $f \in \mathcal{F}(L, (p_0, \ldots, p_{i-1}, p_i - 1, p_{i+1}, \ldots, p_{L+1}), s)$. Because there are s active parameters, we can iterate this procedure at least $p_i - s$ times for any $i = 1, \ldots, L$. This proves $f \in \mathcal{F}(L, (p_0, p_1 \wedge s, p_2 \wedge s, \ldots, p_L \wedge s, p_{L+1}), s)$.

Now, before stating the main theorem, we need to introduce a few results that will help us with its proof. More details about the following statements can be found on the supplementary material of Schmidt-Hieber (2020).

First, we state a theorem that guarantees, for any function f of the β-Hölder ball with radius K, the existence of a network that approximates f, with constraints on its architecture.

Theorem 3.1. *For any function $f \in C_r^\beta([0,1]^r, K)$ and any integers $m \geq 1$ and $N \geq (\beta+1)^r \vee (K+1)e^r$, there exists a network*

$$\tilde{f} \in \mathcal{F}(L, (r, 6(r + \lceil\beta\rceil)N, \ldots, 6(r + \lceil\beta\rceil)N, 1), s, \infty)$$

with depth

$$L = 8 + (m+5)(1 + \lceil \log_2(r \vee \beta) \rceil)$$

and number of parameters

$$s \leq 141(r + \beta + 1)^{3+r} N(m+6),$$

such that

$$\|\tilde{f} - f\|_{L^\infty([0,1]^r)} \leq (2K+1)(1 + r^2 + \beta^2)6^r N 2^{-m} + K 3^\beta N^{-\frac{\beta}{r}}.$$

With this in mind, we can start to construct such a network. For the rest of this course, we will consider functions g_{ij} such that $g_{ij} \in C_{t_i}^{\beta_i}([a_i, b_i]^{t_i}, K_i)$, where we assume that $K_i \geq 1$. For $i = 1, \ldots, q-1$, we define

$$h_0 := \frac{g_0}{2K_0} + \frac{1}{2}, \quad h_i := \frac{g_i(2K_{i-1} \cdot - K_{i-1})}{2K_i} + \frac{1}{2}, \quad h_q = g_q(2K_{q-1} \cdot - K_{q-1}).$$

Here, $2K_{i-1}x - K_{i-1}$ means that the entries are transformed by $2K_{i-1}x_j - K_{i-1}$ for all j. Hence, we have

$$f = g_q \circ \cdots \circ g_0 = h_q \circ \cdots \circ h_0. \tag{11}$$

Thus, by definition of the Hölder balls $C_r^\beta(D, K)$, h_{0j} takes values in $[0,1]$, $h_{0j} \in C_{t_0}^{\beta_0}([0,1]^{t_0}, 1)$, $h_{ij} \in C_{t_i}^{\beta_i}([0,1]^{t_i}, (2K_{i-1})^{\beta_i})$ for $i = 1, \ldots, q-1$ and $h_{qj} \in C_{t_q}^{\beta_q}([0,1]^{t_q}, K_q(2K_{q-1})^{\beta_q})$.

It means that f can always be written as a composition of functions defined on hypercubes $[0,1]^{t_i}$.

Then, we introduce a few technical lemmas :

Lemma 3.1. *Let h_{ij} be as above with $K_i \geq 1$. Then, for any functions $\tilde{h}_i = (\tilde{h}_{ij})_j^\top$ with $\tilde{h}_{ij} : [0,1]^{t_i} \to [0,1]$,*

$$\|h_q \circ \cdots \circ h_0 - \tilde{h}_q \circ \cdots \circ \tilde{h}_0\|_{L^\infty[0,1]^d} \leq K_q \prod_{\ell=0}^{q-1} (2K_\ell)^{\beta_{\ell+1}} \sum_{i=0}^{q} \| |h_i - \tilde{h}_i|_\infty \|_{L^\infty[0,1]^{d_i}}^{\prod_{\ell=i+1}^{q} \beta_\ell \wedge 1}.$$

The two following lemmas will eventually lead us to an oracle inequality on the risk, where Lemma 3.1 will allow us to bound the first term of its upper bound.

Lemma 3.2. *Consider the d-variate nonparametric regression model (1) with unknown regression function f_0. Let $\hat{f}$ be any estimator taking values in $\mathcal{F}$. Assume $\{f_0\} \cup \mathcal{F} \subset \{f : [0,1]^d \to [-F, F]\}$ for some $F \geq 1$.*

Let $\mathcal{N}(\delta, \mathcal{F}, \|\cdot\|_\infty)$ be the covering number, i.e. the minimal number of $\|\cdot\|_\infty$-balls with radius δ that covers $\mathcal{F}$ (the centers do not need to be in $\mathcal{F}$). Δ_n is the quantity defined in (2). If $\mathcal{N}_n := \mathcal{N}(\delta, \mathcal{F}, \|\cdot\|_\infty) \geq 3$, then, for all $\delta, \varepsilon \in (0,1]$,

$$(1-\varepsilon)^2 \Delta_n - F^2 \frac{18 \log \mathcal{N}_n + 76}{n\varepsilon} - 38\delta F \leq R(\hat{f}, f_0)$$

$$\leq (1+\varepsilon)^2 \left[\inf_{f \in \mathcal{F}} E[(f(\mathbf{X}) - f_0(\mathbf{X}))^2] + F^2 \frac{18 \log \mathcal{N}_n + 72}{n\varepsilon} + 32\delta F + \Delta_n \right].$$

And we have a covering entropy bound.

Lemma 3.3. *If $V := \prod_{\ell=0}^{L+1} (p_\ell + 1)$, then, for any $\delta > 0$,*

$$\log \mathcal{N}(\delta, \mathcal{F}(L, \mathbf{p}, s, \infty), \|\cdot\|_\infty) \leq (s+1) \log(2\delta^{-1}(L+1)V^2).$$

Remark 3.1. *Identity (10) applied to Lemma 3.3 gives*

$$\log \mathcal{N}(\delta, \mathcal{F}(L, \mathbf{p}, s, \infty), \|\cdot\|_\infty) \leq (s+1) \log(2^{2L+5}\delta^{-1}(L+1)p_0^2 p_{L+1}^{2L}).$$

Hence, since $F \geq 1$, with Lemma 3.3 with $\delta = 1/n$, Lemma 3.2 and Remark 3.1 we can state the following theorem giving an oracle-type inequality.

Theorem 3.2. *Consider the d-variate nonparametric regression model (1) with unknown regression function f_0, satisfying $\|f_0\|_\infty \leq F$ for some $F \geq 1$. Let $\hat{f}_n$ be any estimator taking values in the class $\mathcal{F}(L, \mathbf{p}, s, F)$ and $\Delta_n(\widehat{f}_n, f_0) = \Delta_n(\widehat{f}_n, f_0, \mathcal{F}(L, \mathbf{p}, s, F))$. For any $\varepsilon \in (0,1]$, there exists a constant C_ε, only depending on ε, such that with*

$$\tau_{\varepsilon,n} := C_\varepsilon F^2 \frac{(s+1)\log(n(s+1)^L p_0 p_{L+1})}{n},$$

$$(1-\varepsilon)^2 \Delta_n(\hat{f}_n, f_0) - \tau_{\varepsilon,n} \leq R(\hat{f}_n, f_0)$$

$$\leq (1+\varepsilon)^2 \left(\inf_{f \in \mathcal{F}(L,\mathbf{p},s,F)} \|f - f_0\|_\infty^2 + \Delta_n(\hat{f}_n, f_0) \right) + \tau_{\varepsilon,n}.$$

3.2.2 Optimality of the rates

Having introduced these results, we now present the main theorem of this course. It gives a convergence rate for any estimator belonging to the class of functions $\mathcal{F}$ defined earlier, provided that the parameters are well chosen.

Theorem 3.3. *Consider the d-variate nonparametric regression model (1) for composite regression function (6) in the class $\mathcal{G}(q, d, t, \beta, K)$. Let $\hat{f}_n$ be an estimator taking values in the network class $\mathcal{F}(L, (p_i)_{i=0,\dots,L+1}, s, F)$ satisfying:*

(i) $F \geq \max(K, 1)$,

(ii) $\sum_{j=0}^{L} \log_2(4t_j \vee 4B_j) \log_2 n \leq L \leq n\phi_n$,

(iii) $n\phi_n \lesssim \min_{i=1,\dots,L} p_i$,

(iv) $s \asymp n\phi_n \log n$.

There exist constants C, C' only depending on q, d, t, β, F, such that if

$$\Delta_n(\hat{f}_n, f_0) \leq C\phi_n L \log^2 n,$$

then

$$R(\hat{f}_n, f_0) \leq C'\phi_n L \log^2 n, \tag{12}$$

and if $\Delta_n(\hat{f}_n, f_0) \geq C\phi_n L \log^2 n$, then

$$\frac{1}{C'}\Delta_n(\hat{f}_n, f_0) \leq R(\hat{f}_n, f_0) \leq C'\Delta_n(\hat{f}_n, f_0). \tag{13}$$

In order to minimize the rate $\phi_n L \log^2 n$, the best choice is to choose L of the order of $\log_2 n$. The rate in the regime $\Delta_n(\hat{f}_n, f_0) \leq C\phi_n \log^3 n$ becomes then

$$R(\hat{f}_n, f_0) \leq C'\phi_n \log^3 n.$$

The convergence rate in Theorem 3.3 depends on ϕ_n and $\Delta_n(\hat{f}_n, f_0)$. Below we show that ϕ_n is a lower bound for the minimax estimation risk over this class. Recall that the term $\Delta_n(\hat{f}_n, f_0)$ is large if $\hat{f}_n$ has a large empirical risk compared to an empirical risk minimizer. Having this term in the convergence rate is unavoidable as it also appears in the lower bound in (13). Since for any empirical risk minimizer the Δ_n-term is zero by definition, we have the following direct consequence of the main theorem:

Corollary 3.1. *Let $\tilde{f}_n \in \arg\min_{f \in \mathcal{F}(L,p,S,F)} \sum_{i=1}^{n} (Y_i - f(X_i))^2$ be an empirical risk minimizer. Under the same conditions as for Theorem 3.3, there exists a constant C', only depending on q, d, t, β, F, such that*

$$R(\tilde{f}_n, f_0) \leq C'\phi_n L \log^2 n.$$

Proof. This demonstration holds for n sufficiently large, which is the case in practice. During the proof, we will often refer to a constant C', only depending on (q, d, t, β, F), but note that this constant may vary from line to line.

First, with the assumed bounds on $s \asymp n\phi_n \log(n)$, where we recall that $\phi_n := \max_{i=0,\dots,q} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}}$, we get $\log(s+1) \leq C' \times \log(n)$ where the term $\log(\log(n))$ is ignored. Now, knowing that $p_0 p_{L+1} = d$, we have :

$$\begin{aligned}
\tau_{\varepsilon,n} &= C_\varepsilon F^2 \frac{(s+1)\log(n(s+1)^L p_0 p_{L+1})}{n} \\
&= C' \frac{(s+1)(\log(n) + L \times \log(s+1) + \log(d))}{n} \\
&\leq C' \frac{(n\phi_n \log(n))(\log(n) + L \times \log(n))}{n} \\
&\leq C' \phi_n L \log^2(n)
\end{aligned}$$

Hence, combined with Theorem 3.2, it follows for $n \geq 3$:

$$\frac{1}{4}\Delta_n(\widehat{f}_n, f_0) - C'\phi_n L \log^2 n \leq R(\widehat{f}_n, f_0$$
$$\leq 4 \inf_{f^* \in \mathcal{F}(L,p,s,F)} \|f^* - f_0\|_\infty^2 + 4\Delta_n(\widehat{f}_n, f_0) + C'\phi_n L \log^2 n$$

where we used $\epsilon = 1/2$ for the lower bound and $\epsilon = 1$ for the upper bound. Taking $C = 8C'$, we find that $\frac{1}{8}\Delta_n(\widehat{f}_n, f_0) \leq R(\widehat{f}_n, f_0)$ whenever $\Delta_n(\widehat{f}_n, f_0) \geq C\phi_n L \log^2 n$. This proves the lower bound in (13).

As for the upper bounds in (12) and (13), we need to bound the approximation error. To do this, we rewrite the regression function f_0 as in (11), that is,

$$f_0 = h_q \circ \cdots \circ h_0 \quad \text{with} \quad h_i = (h_{ij})_j^\top,$$

h_{ij} defined on $[0,1]^{t_i}$, and for any $i < q$, h_{ij} mapping to $[0,1]$.

We apply Theorem 3.1 to each function h_{ij} separately. Take $m = \lceil \log_2 n \rceil$, and let $L_i' := 8 + (\lceil \log_2 n \rceil + 5)(1 + \lceil \log_2(t_i \vee \beta_i) \rceil)$. This means there exists a network $\tilde{h}_{ij} \in \mathcal{F}(L_i', (t_i, 6(t_i + \lceil \beta_i \rceil)N, \ldots, 6(t_i + \lceil \beta_i \rceil)N, 1), s_i)$ with $s_i \leq 141(t_i + \beta_i + 1)^{3+t_i} N(\lceil \log_2 n \rceil + 6)$, such that

$$\|\tilde{h}_{ij} - h_{ij}\|_{L^\infty([0,1]^{t_i})} \leq (2Q_i + 1)(1 + t_i^2 + \beta_i^2)6^{t_i} N n^{-1} + Q_i 3^{\beta_i} N^{-\frac{\beta_i}{t_i}}, \tag{14}$$

where Q_i is any upper bound of the Hölder norms of h_{ij}. If $i < q$, then we apply to the output the two additional layers $1 - (1 - x)_+$. This requires four additional parameters (two operations, namely a multiplication by -1 and a sum by 1, on two successive layers). Let h_{ij}^* be the resulting network such that $h_{ij}^* \in \mathcal{F}(L_i' + 2, (t_i, 6(t_i + \lceil \beta_i \rceil)N, \ldots, 6(t_i + \lceil \beta_i \rceil)N, 1), s_i + 4)$. The previous step guarantees that $\sigma(h_{ij}^*)$ is in $[0,1]$. Indeed, observe that $\sigma(h_{ij}^*) = (\tilde{h}_{ij}(x) \vee 0) \wedge 1$.

So if $\tilde{h}_{ij} < 0$, then $\sigma(h_{ij}^*) = 0$, and since $h_{ij}(x) \in [0,1]$, $|\sigma(h_{ij}^*) - h_{ij}| \leq |\tilde{h}_{ij} - h_{ij}|$. The inequality remains valid if $\tilde{h}_{ij} \in [0,1]$, since $\sigma(h_{ij}^*) = \tilde{h}_{ij}$ in this case. Finally, if $\tilde{h}_{ij} > 1$, then $\sigma(h_{ij}^*) = 1$ and $0 \leq \sigma(h_{ij}^*) - h_{ij} \leq \tilde{h}_{ij} - h_{ij}$. Thus we have :

$$\|\sigma(h_{ij}^*) - h_{ij}\|_{L^\infty([0,1]^{t_i})} \leq \|\tilde{h}_{ij} - h_{ij}\|_{L^\infty([0,1]^{t_i})}. \tag{15}$$

We compute the networks h_{ij}^* in parallel. Thus, due to the properties of parallelization and enlarging, $h_i^* = (h_{ij}^*)_{j=1,\ldots,d_{i+1}}$ is in the class

$$\mathcal{F}\left(L_i' + 2, (d_i, 6r_i N, \ldots, 6r_i N, d_{i+1}), d_{i+1}(s_i + 4)\right),$$

with $r_i := \max d_{i+1}(t_i + \lceil \beta_i \rceil)$, where the term d_{i+1} comes from the summation.

Finally, we construct the composite network $f^* = h_{q1}^* \circ \sigma(h_{q-1}^*) \circ \cdots \circ \sigma(h_0^*)$, which by the composition rule mentioned in the preparatory work, is in the class

$$\mathcal{F}\left(E, (d, 6r_i N, \ldots, 6r_i N, 1), \sum_{i=0}^{q} d_{i+1}(s_i + 4)\right),$$

where $p_0 = d$ and $p'_{L+1} = 1$ as expected, and $E := 3(q-1) + \sum_{i=0}^{q} L_i'$. The term $3(q-1)$ appears due to the depth synchronization also mentioned at the beginning of the preparatory work.

Then, due to our choice of L_i' and the hypothesis (ii) of the theorem, we can show that, for all sufficiently large n :

$$E \leq L.$$

By (9) and for sufficiently large n, the space of f^* can be embedded into $\mathcal{F}(L, p, s)$ with L, p, s satisfying the assumptions of the theorem by choosing $N = \lceil c \max_{i=0,\dots,q} n^{\frac{t_i}{2\beta_i^* + t_i}} \rceil$ for a sufficiently small constant $c > 0$ only depending on q, d, t, β. Combining Lemma 3.1 with (14) and (15), we obtain

$$\inf_{f^* \in \mathcal{F}(L,p,s)} \|f^* - f_0\|_\infty^2 \leq C' \max_{i=0,\dots,q} N^{-\frac{2\beta_i^*}{t_i}} \leq C' \max_{i=0,\dots,q} c^{\frac{-2\beta_i^*}{t_i}} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}} \leq C' \phi_n. \tag{16}$$

The last step is to find a network function that is bounded in sup-norm by F. By the previous inequality, there exists a sequence $(\tilde{f}_n)_n$ such that for all sufficiently large n, $\tilde{f}_n \in \mathcal{F}(L, p, s)$ and $\|\tilde{f}_n - f_0\|_\infty^2 \leq 2C \max_{i=0,\dots,q} c^{\frac{-2\beta_i^*}{t_i}} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}}$. Define $f_n^* := \tilde{f}_n(\|f_0\|_\infty / \|\tilde{f}_n\|_\infty \wedge 1)$. Then, $\|f_n^*\|_\infty \leq \|f_0\|_\infty \leq K \leq F$, where the last inequality follows from assumption (i). Moreover, $f_n^* \in \mathcal{F}(L, p, s, F)$. Writing $f_n^* - f_0 = (f_n^* - \tilde{f}_n) + (\tilde{f}_n - f_0)$, we obtain $\|f_n^* - f_0\|_\infty \leq 2\|\tilde{f}_n - f_0\|_\infty$.

This shows that (16) also holds (with constants multiplied by 8) if the infimum is taken over the smaller space $\mathcal{F}(L, p, s, F)$. We apply this bound on our oracle-type inequality and the upper bounds in (12) and (13) follow for any constant C. That allows us to conclude the proof.

□

This proof introduce a $\log^2 n$ factor in the convergence rate $\phi_n L \log^2 n$. In order to conclude this course, we give a lower bound for the minimax estimation risk over the class $\mathcal{G}(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, K)$ in the regime $t_i \leq \min(d_0, \dots, d_{i-1})$ for all i. Outside of this regime, it is hard to determine the minimax rate. In this case, no dimensions are added on deeper abstraction levels in the composition of functions. In particular, it avoids that t_i is larger than the input dimension d_0.

Theorem 3.4. *Consider the nonparametric regression model (1) with X_i drawn from a distribution with Lebesgue density on $[0,1]^d$ which is lower and upper bounded by positive constants. For any nonnegative integer q, any dimension vectors $\mathbf{d}$ and $\mathbf{t}$ satisfying $t_i \leq \min(d_0, \dots, d_{i-1})$ for all i, any smoothness vector $\boldsymbol{\beta}$ and all sufficiently large constants $K > 0$, there exists a positive constant c such that*

$$\inf_{\hat{f}_n} \sup_{f_0 \in \mathcal{G}(q,\mathbf{d},\mathbf{t},\boldsymbol{\beta},K)} R(\hat{f}_n, f_0) \geq c\phi_n,$$

where the inf *is taken over all estimators $\hat{f}_n$.*

With this result, we have shown that estimators based on sparsely connected DNNs with ReLU activation function and well designed network architecture achieve the minimax rates of convergence, up to $\log^2 n$ factors, under a general composition assumption on the regression function.

References

Bauer, B. and Kohler, M. (2019) On deep learning as a remedy for the curse of dimensionality in nonparametric regression. URL: https://projecteuclid.org/journals/annals-of-statistics/volume-47/issue-4/On-deep-learning-as-a-remedy-for-the-curse-of/10.1214/18-AOS1747.full.

Bottou, L. (2010) Large-scale machine learning with stochastic gradient descent.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M. and de Freitas, N. (2014) Predicting parameters in deep learning. URL: https://arxiv.org/abs/1306.0543.

Dion-Blanc, C. (2024) Cours m2 : Estimation non-paramétrique.

Hornik, K. (1989) Multilayer feedforward networks are universal approximators. URL: https://www.cs.cmu.edu/~epxing/Class/10715/reading/Kornick_et_al.pdf.

Inturrisi, J., Khoo, S. Y. and Kouzani, A. (2021) Piecewise linear units improve deep neural networks. URL: https://arxiv.org/pdf/2108.00700.

Kingma, D. P. and Ba, J. (2015) Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

LeCun, Y., Bengio, Y. and Hinton, G. (2015) Deep learning. URL: https://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf.

Linnainmaa, S. (1970) *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. Master's thesis, University of Helsinki.

McCulloch, W. S. and Pitts, W. (1943) A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**, 115–133.

Schmidt-Hieber, J. (2020) Nonparametric regression using deep neural networks with relu activation function. URL: https://arxiv.org/abs/1708.06633.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) Dropout: A simple way to prevent neural networks from overfitting. URL: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf.

Stone, C. J. (1982) Optimal global rates of convergence for nonparametric regression. *The Annals of Statistics*, 1040–1053.

Tsybakov, A. B. (2009) *Introduction to Nonparametric Estimation*. Springer Series in Statistics. New York: Springer. Revised and extended from the 2004 French original, translated by Vladimir Zaiats.

Yarotsky, D. (2017) Error bounds for approximations with deep relu networks. URL: https://arxiv.org/abs/1610.01145.