Nonparametric regression using deep neural networks with ReLU activation function

Johannes Schmidt-Hieber*

Abstract

Consider the multivariate nonparametric regression model. It is shown that estimators based on sparsely connected deep neural networks with ReLU activation function and properly chosen network architecture achieve the minimax rates of convergence (up to $\log n$-factors) under a general composition assumption on the regression function. The framework includes many well-studied structural constraints such as (generalized) additive models. While there is a lot of flexibility in the network architecture, the tuning parameter is the sparsity of the network. Specifically, we consider large networks with number of potential parameters being much bigger than the sample size. The analysis gives some insights why multilayer feedforward neural networks perform well in practice. Interestingly, the depth (number of layers) of the neural network architectures plays an important role and our theory suggests that scaling the network depth with the logarithm of the sample size is natural.

Keywords: nonparametric regression; multilayer neural networks; ReLU activation function; minimax estimation risk; additive models.

1 Introduction

In the nonparametric regression model with random design, we observe n i.i.d. vectors $\mathbf{X}_i \in [0,1]^d$ and n responses $Y_i \in \mathbb{R}$ from the model

$$Y_i = f(\mathbf{X}_i) + \varepsilon_i, \quad i = 1, \ldots, n. \tag{1.1}$$

The noise variables ε_i are assumed to be i.i.d. standard normal and independent of $(\mathbf{X}_i)_i$. The statistical problem is to recover the unknown function $f : [0,1]^d \to \mathbb{R}$ from the sample

*Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands.
Email: `schmidthieberaj@math.leidenuniv.nl`

$(\mathbf{X}_i, Y_i)_i$. Various methods exist that allow to estimate the regression function nonparametrically, including kernel smoothing, series estimators/wavelets and splines, cf. [13, 40, 38]. In this work, we consider fitting a multilayer feedforward artificial neural network to the data. It is shown that the estimator achieves nearly optimal convergence rates under various constraints on the regression function.

Multilayer (or deep) neural networks have been successfully trained recently to achieve impressive results for complicated tasks such as object detection on images and speech recognition. Deep learning is now considered to be the state-of-the art for these tasks. But there is a lack of theoretical understanding. One problem is that fitting a neural network to data is highly non-linear in the parameters. Moreover, the function class is non-convex and different regularization methods are combined in practice.

This article is inspired by the idea to build a statistical theory that provides some understanding of these procedures. As the full method is too complex to be theoretically tractable, we need to make some selection of important characteristics that we believe are crucial for the success of the procedure.

To fit a neural network, an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ needs to be chosen. Traditionally, sigmoidal activation functions were employed. Recall that an activation function is called sigmoidal if it can be written as cumulative distribution function of a real valued random variable. For deep neural networks, however, there is a clear gain using the non-sigmoidal rectifier linear unit (ReLU) $\sigma(x) = \max(x, 0) = (x)_+$. Indeed, in practice the ReLU outperforms other activation functions with respect to the statistical performance and the computational cost, cf. [11]. While most of the existing statistical theory deals with sigmoidal activation functions, we provide statistical theory for deep neural networks with ReLU activation function.

The statistical analysis for the ReLU activation function is quite different from earlier approaches and we discuss this in more detail in the overview on related literature in Section 4. Viewed as a nonparametric method the ReLU has some surprising properties. To explain this, notice that deep networks with ReLU activation produce functions that are piecewise linear in the input. Nonparametric methods which are based on piecewise linear approximations are typically not able to capture higher-order smoothness in the signal and are rate-optimal only up to smoothness index two. Interestingly, we can show that the ReLU combined with a deep network architecture achieves near minimax rates for arbitrary smoothness of the regression function.

The number of hidden layers of state-of-the-art network architectures has been growing over the past years, cf. [36]. There are versions of the recently developed deep network

ResNet which are based on 152 layers, cf. [15]. Our analysis indicates that for the ReLU activation function the network depth should scale with the logarithm of the sample size. This suggests, that for larger samples, additional hidden layers should be added.

Recent deep architectures include more parameters than training samples. The well-known AlexNet [26] for instance is based on 60 million network parameters using only 1.2 million samples. We account for high-dimensional parameter spaces in our analysis by assuming that the number of potential network parameters is bounded by an arbitrary power of the sample size. To avoid overfitting, some sort of regularization or sparsity has to be incorporated. In the deep networks literature, one option is to make the network thinner assuming that only few parameters are non-zero (or active), cf. [12], Section 7.10. Our analysis shows that the number of non-zero parameters plays the role of the effective model dimension and - as common in non-parametric regression - needs to be chosen carefully.

Existing statistical theory often requires that the size of the network parameters tends to infinity as the sample size increases. In practice, estimated network weights are, however, rather small. We can incorporate small parameters in our theory, proving that it is sufficient to consider neural networks with all network parameters bounded in absolute value by one.

Multilayer neural networks are typically applied to high-dimensional input. Without additional structure in the signal besides smoothness, nonparametric estimation rates are then slow because of the well-known curse of dimensionality. This means that no statistical procedure can do well regarding pointwise reconstruction of the signal. Multilayer neural networks are believed to be able to adapt to many different structures in the signal, therefore avoiding the curse of dimensionality and achieving faster rates in many situations. In this work, we stick to the regression setup and show that deep networks can indeed attain faster rates under a hierarchical composition structure assumption on the regression function, which includes (generalized) additive models and the composition models considered in [18, 19, 3, 23, 6].

Parts of the success of multilayer neural networks can be explained by the fast algorithms that are available to estimate the network weights from data. These iterative algorithms are based on minimization of some empirical loss function using stochastic gradient descent. To regularize the reconstruction, a common method is to stop after few iterations. Because of the non-convex function space, these gradient descent methods converge to one of the many local minima. It is now widely believed that the risk of most of the local minima is not much larger than the risk of the global minimum, cf. [8]. On the contrary, the statistical estimation theory deals with the estimator minimizing the (global) least-squares functional over a class of network functions. Although this estimator is in general not computable it is conceivable that it has similar properties as an estimator obtained using stochastic gradient descent.

Our setting deviates in two other important features from the computer science literature on deep learning. Firstly, we consider regression and not classification. Secondly, we restrict ourselves in this article to multilayer feedforward artificial neural networks, while most of the many recent deep learning applications have been obtained using convolutional or recurrent neural networks. Although these are limitations, one should be aware that our setting is much more general than previous statistical work on the topic and provides, to the best of our knowledge, for the first time nearly optimal estimation rates for multilayer neural networks with ReLU activation function.

The article is structured as follows. Section 2 introduces multilayer feedforward artificial neural networks. The considered function classes and the main result can be found in Section 3. This part also discusses several specific examples such as additive models. We give an overview of relevant related literature in Section 4. The proof of the main result together with additional discussion can be found in Section 5.

Notation: Vectors are denoted by bold letters, $\mathbf{x} := (x_1, \dots, x_d)$, $\mathbf{0} := (0, \dots, 0), \dots$ As usual, we define $|\mathbf{x}|_p := (\sum_{i=1}^d |x_i|^p)^{1/p}$, $|\mathbf{x}|_\infty := \max_i |x_i|$, $|\mathbf{x}|_0 = \sum_i \mathbf{1}(x_i \neq 0)$, and write $\|f\|_p := \|f\|_{L^p(D)}$ for the L^p-norm on D, whenever there is no ambiguity of the domain D. For two sequences $(a_n)_n$ and $(b_n)_n$, we write $a_n \lesssim b_n$ if there exists a constant C such that $a_n \leq C b_n$ for all n. Moreover, $a_n \asymp b_n$ means that $a_n \lesssim b_n$ and $b_n \lesssim a_n$.

2 Mathematical definition of multilayer neural networks

Fitting a multilayer neural network requires the choice of an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ and the network architecture. Motivated by the importance in deep learning, we study the rectifier linear unit (ReLU) activation function

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

For $\mathbf{v} = (v_1, \dots, v_r) \in \mathbb{R}^r$, define the shifted activation function $\sigma_{\mathbf{v}} : \mathbb{R}^r \to \mathbb{R}^r$ as

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

The network architecture $(L, \mathbf{p})$ consists of a positive integer L called the *number of hidden layers* or *depth* and a *width vector* $\mathbf{p} = (p_0, \dots, p_{L+1}) \in \mathbb{N}^{L+2}$. A neural network with network architecture $(L, \mathbf{p})$ is then any function of the form

$$f : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}, \quad \mathbf{x} \mapsto f(\mathbf{x}) = W_{L+1} \sigma_{\mathbf{v}_L} W_L \sigma_{\mathbf{v}_{L-1}} \cdots W_2 \sigma_{\mathbf{v}_1} W_1 \mathbf{x}, \tag{2.1}$$

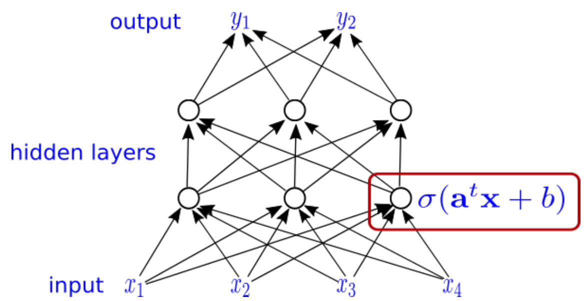

Figure 1: Representation as a direct graph of a network with two hidden layers $L = 2$ and width vector $\mathbf{p} = (4, 3, 3, 2)$.

where W_i is a $p_i \times p_{i-1}$ weight matrix and $\mathbf{v}_i \in \mathbb{R}^{p_i}$ is a shift vector. Network functions are therefore build by alternating matrix-vector multiplications with the action of the non-linear activation function σ. In (2.1), it is also possible to omit the shift vectors by considering the input $(\mathbf{x}, 1)$ and enlarging the weight matrices by one row and one column with appropriate entries. For our analysis it is, however, more convenient to work with the representation (2.1).

In computer science, neural networks are more commonly introduced via their representation as directed acyclic graph, cf. Figure 1. Using this equivalent definition, the nodes in the graph (also called *units*) are arranged in layers. The input layer is the first layer and the output layer the last layer. The layers that lie in between are the hidden layers. The number of hidden layers corresponds to L and the number of units in each layer generates the width vector $\mathbf{p}$. Each node/unit in the graph representation stands for a scalar product of the incoming signal with a weight vector which is then shifted and applied to the activation function.

Given a network function $f(\mathbf{x}) = W_{L+1}\sigma_{\mathbf{v}_L} W_L \sigma_{\mathbf{v}_{L-1}} \cdots W_2 \sigma_{\mathbf{v}_1} W_1 \mathbf{x}$, the network parameters are the entries of the matrices $(W_j)_{j=1,\ldots,L+1}$ and vectors $(\mathbf{v}_j)_{j=1,\ldots,L}$. Denote by $\|W_j\|_0$ the number of non-zero entries of W_j. Similarly, define by $\|W_j\|_\infty$ the maximum-entry norm of W_j. The aim of this article is to study the statistical properties of large but sparse networks with bounded parameters. To this end, define the space of network functions with given network architecture and network parameters bounded by one,

$$\mathcal{F}(L, \mathbf{p}) := \Big\{ f \text{ of the form (2.1)} : \max_{j \in 1,\ldots,L+1} \|W_j\|_\infty \vee |\mathbf{v}_j|_\infty \leq 1 \Big\}, \tag{2.2}$$

with the convention $\mathbf{v}_{L+1} := \mathbf{0}$. Also define

$$\mathcal{F}(L, \mathbf{p}, s) := \mathcal{F}(L, \mathbf{p}, s, F) := \Big\{ f \in \mathcal{F}(L, \mathbf{p}) : \sum_{j=1}^{L+1} \|W_j\|_0 + |\mathbf{v}_j|_0 \leq s, \; \big\| |f|_\infty \big\|_\infty \leq F \Big\}. \tag{2.3}$$

The upper bound on the uniform norm of f is most of the time dispensable and therefore omitted in the notation. We consider cases where the number of network parameters s is small compared to the total number of parameters.

Recall that we consider the d-variate nonparametric regression model. We therefore must have $p_0 = d$ and $p_{L+1} = 1$. The multilayer neural network estimator $\widehat{f}$ is the network in $\mathcal{F}(L, \mathbf{p}, s, F)$ that fits the data best in a least squares sense, that is,

$$\widehat{f} \in \underset{f \in \mathcal{F}(L, \mathbf{p}, s, F)}{\arg\min} \sum_{i=1}^{n} (Y_i - f(\mathbf{X}_i))^2. \tag{2.4}$$

To evaluate the statistical performance of this estimator, we derive bounds on the L^2-risk

$$R(\widehat{f}, f) := E_f\big[\big(\widehat{f}(\mathbf{X}) - f(\mathbf{X})\big)^2\big],$$

with $\mathbf{X} \overset{\mathcal{D}}{=} \mathbf{X}_1$ being independent of the sample $(\mathbf{X}_i, Y_i)_i$. The subscript f in E_f indicates that the expectation is taken with respect to a sample generated from the nonparametric regression model with regression function f.

3 Main results

In this section we show that if the number of parameters s is of the right order, the neural network estimator achieves the optimal nonparametric rates (up to $\log n$-factors) over a wide range of function classes with different structural constraints and smoothness properties. We start by defining appropriate function classes.

A simple structural constraint is that the regression function $f : [0,1]^d \to \mathbb{R}$ can be written as a composition of functions

$$f = g_1 \circ g_0 \tag{3.1}$$

with $g_0 : [0,1]^d \to \mathbb{R}^{d_1}$ and $g_1 : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$. Nonparametric estimation under this composition assumption has been studied in [19]. Notice that (3.1) generalizes additive models $f(\mathbf{x}) = \sum_{j=1}^{d} f_j(x_j)$ by taking $g_0(\mathbf{x}) = (f_1(x_1), \ldots, f_d(x_d))^t$ and $g_1(\mathbf{y}) = \sum_{j=1}^{d} y_j$. Then, g_1 is arbitrarily smooth and the components of g_0 depend on one variable only. This causes the fast rates that can be obtained for additive models. Assumption (3.1) can also be seen as a generalization of shape constraints such as log-concavity.

To represent higher order structure, such as generalized additive models, (3.1) turns out to be insufficient and we consider instead compositions of arbitrary depth

$$f = g_q \circ g_{q-1} \circ \ldots \circ g_1 \circ g_0 \tag{3.2}$$

with $g_i : [a_i, b_i]^{d_i} \to [a_{i+1}, b_{i+1}]^{d_{i+1}}$. Denote by $g_i = (g_{ij})^t_{j=1,\ldots,d_i}$ the components of g_i and let t_i be the maximal number of variables on which each of the g_{ij} depends on. Thus, each g_{ij} is a t_i-variate function and for specific constraints such as additive models, t_i might be much smaller than d_i. The single components $g_0, \ldots, g_q$ are obviously not identifiable. As we are only interested in estimation of f this causes no problems. Since $f : [0,1]^d \to \mathbb{R}$, we must have $d_0 = d$, $a_0 = 0$, $b_0 = 1$ and $d_{q+1} = 1$.

The composition assumption in (3.2) occurs very naturally in combination with multilayer neural networks. Notice that nonparametric estimation of f is a challenging task and many estimation techniques such as series estimators/wavelets might not be able to take advantage from the underlying composition structure in the regression function. Slightly more specific function spaces, which alternate between summations and composition of functions, have been considered in [18, 6]. We provide below an example of a function class that can be decomposed in the form (3.2) but is not contained in these spaces.

We assume that each of the functions g_{ij} has some Hölder smoothness. Recall that a function has Hölder smoothness index β if all partial derivatives up to order $\lfloor \beta \rfloor$ exist and are bounded and the partial derivatives of order $\lfloor \beta \rfloor$ are $\beta - \lfloor \beta \rfloor$ Hölder, where $\lfloor \beta \rfloor$ denotes the largest integer strictly smaller than β. The ball of β-Hölder functions with radius K is then defined as

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

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

We assume that each of the functions g_{ij} has Hölder smoothness β_i. Since g_{ij} is also t_i-variate, $g_{ij} \in \mathcal{C}_{t_i}^{\beta_i}([a_i, b_i]^{t_i}, K_i)$. For estimation rates in the nonparametric regression model, the crucial quantity is the smoothness of f. If, for instance, $q = 1$, $\beta_0, \beta_1 \leq 1$, $d_0 = d_1 = t_0 = t_1 = 1$, then $f = g_1 \circ g_0$ and f has smoothness $\beta_0\beta_1$, cf. [19, 34]. We should then be able to achieve at least the convergence rate $n^{-2\beta_0\beta_1/(2\beta_0\beta_1+1)}$. For $\beta_1 > 1$, the rate changes.

Below we see that the estimation rate is described by the effective smoothness indices

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

Theorem 1. *Consider the d-variate nonparametric regression model* (1.1) *for composite regression function* (3.2) *assuming that $g_{ij} \in \mathcal{C}_{t_i}^{\beta_i}([a_i, b_i]^{t_i}, K_i)$, for all $i = 0, \ldots, q$ and $j = 1, \ldots, d_i$. Let $\widehat{f}$ be the neural network estimator defined in* (2.4) *for a network class $\mathcal{F}(L, (p_i)_{i=0,\ldots,L+1}, s, F)$ satisfying*

(i) $F \geq \|f\|_\infty + 1$,

(ii) $\sum_{i=0}^{q} (2 + \log_2 t_i) \log_2 n \leq L \lesssim \log n$,

(iii) $\max_{i=0,\ldots,q} n^{\frac{t_i}{2\beta_i^* + t_i}} \lesssim \min_{i=1,\ldots,L} p_i \leq \max_{i=1,\ldots,L} p_i \lesssim n^C$, *for some $C > 0$,*

(iv) $s \asymp \max_{i=0,\ldots,q} n^{\frac{t_i}{2\beta_i^* + t_i}} \log n$.

Then,

$$R(\widehat{f}, f) \lesssim \max_{i=0,\ldots,q} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}} \log^2 n.$$

The result shows that there is a lot of flexibility in picking a good network architecture as long as the number of active parameters s is taken to be of the right order. Interestingly, the depth L of the network can be chosen without knowledge of the smoothness indices and it is sufficient to know an upper bound on the $t_i \leq d_i$. The network width can also be chosen independent of the smoothness indices by taking $n \lesssim \min_i p_i \leq \max_i p_i \lesssim n^C$. One might wonder whether the sparsity s can be made adaptive by minimizing a penalized least squares problem with an ℓ^1-penalty on the network weights. As this requires much more machinery, the question will be moved to future work.

The number of network parameters in a fully connected network is of the order $\sum_{i=0}^{L} p_i p_{i+1}$. This shows that Theorem 1 requires sparse networks. More specifically, the network has at least $\sum_{i=1}^{L} p_i - s$ completely inactive nodes, meaning that all incoming signal is zero.

For convenience, Theorem 1 is stated without explicit constants. The proofs, however, are non-asymptotic although we did not make an attempt to minimize the constants. It is well-known that deep learning outperforms other methods only for large sample sizes. This indicates that the method might be able to adapt to underlying structure in the signal and therefore achieving fast convergence rates but with large constants or remainder terms which spoil the results for small samples. The proof of the upper bound on the risk in Theorem 1 is based on the following oracle-type inequality.

Theorem 2. *Consider the d-variate nonparametric regression model* (1.1) *with unknown regression function f_0, satisfying $\|f_0\|_\infty \leq F$ for some $F \geq 1$. Let $\widehat{f}$ be the neural network estimator defined in* (2.4) *for a network class $\mathcal{F}(L, \mathbf{p}, s, F)$. If $V := \prod_{\ell=0}^{L+1}(p_\ell + 1)$, then, for any $\varepsilon > 0$, there exists a constant C_ε, only depending on ε, such that*

$$R(\widehat{f}, f_0) \leq (1+\varepsilon)^2 \inf_{f \in \mathcal{F}(L,\mathbf{p},s,F)} \|f - f_0\|_\infty^2 + C_\varepsilon F^2 \frac{(s+1)\log(2n(L+1)V^2)}{n}.$$

Let us remark on the optimality of the rate in Theorem 1. Obviously, the $\log^2 n$-factor is an artifact of the upper bound. For composition of two functions (3.1), [19] derives the minimax rates for sup-norm loss. Unlike the classical nonparametric regression model, the minimax estimation rates for L^2-loss can be different by a polynomial power in the sample size n. Nevertheless, our rate coincides in most regimes with the rates obtained in Section 4.1 of [19] for $q = 1$.

A complete proof for a lower bound could be established using the standard multiple testing approach, cf. [38], Section 2.6. We only sketch the idea here. For simplicity assume that $t_i = d_i = 1$ for all i. In this case, the functions g_i are univariate and real-valued. Define $i^* \in \arg\max_{i=0,\ldots,q} n^{-2\beta_i^*/(2\beta_i^*+1)}$ as an index for which the estimation rate is obtained. For any $\alpha > 0$, x^α has Hölder smoothness α and for $\alpha = 1$, the function is infinitely often differentiable and has finite Hölder norm for all smoothness indices. Set $g_\ell(x) = x$ for $\ell < i^*$ and $g_\ell(x) = x^{\beta_\ell \wedge 1}$ for $\ell > i^*$. Then,

$$f(x) = g_q \circ g_{q-1} \circ \ldots \circ g_1 \circ g_0(x) = \big(g_{i^*}(x)\big)^{\prod_{\ell=i^*+1}^q \beta_\ell \wedge 1}.$$

In this model, the Kullback-Leibler divergence is $\mathrm{KL}(P_f, P_g) = \frac{n}{2}\|f - g\|_2^2$. Take a kernel function K and consider $\widetilde{g}(x) = h^{\beta_{i^*}} K(x/h)$. Under standard assumptions on K, $\widetilde{g}$ has Hölder smoothness index β_{i^*}. Now we can generate two hypotheses $f_0(x) = 0$ and $f_1(x) = (h^{\beta_{i^*}} K(x/h))^{\prod_{\ell=i^*+1}^q \beta_\ell \wedge 1}$ by taking $g_{i^*}(x) = 0$ and $g_{i^*}(x) = \widetilde{g}(x)$. Therefore, $|f_0(0) - f_1(0)| \gtrsim h^{\beta_{i^*}^*}$ assuming that $K(0) > 0$. For the Kullback-Leibler divergence, we find $\mathrm{KL}(P_{f_0}, P_{f_1}) \lesssim nh^{2\beta_{i^*}^*+1}$. Using Theorem 2.2 (iii) in [38], this shows that the pointwise rate of convergence is $n^{-2\beta_{i^*}^*/(2\beta_{i^*}^*+1)} = \max_{i=0,\ldots,q} n^{-2\beta_i^*/(2\beta_i^*+1)}$. This matches with the upper bound since $t_i = 1$ for all i. In a similar fashion, we can extend the number of hypotheses to obtain rates with respect to L^2-loss.

Examples: There are several well-studied special cases in which the estimation rate in Theorem 1 leads to a simpler expression.

Additive models: In an additive model the regression function has the form

$$f(x_1, \ldots, x_d) = \sum_{i=1}^d f_i(x_i).$$

As mentioned above this can be written as $f = g_1 \circ g_0$ with $d_1 = d$, $t_0 = 1$, and $g_2(y_1, \ldots, y_d) = \sum_{j=1}^d y_j$. Suppose that $f_i \in \mathcal{C}_1^\beta([0,1], K)$. Then, $f: [0,1]^d \xrightarrow{g_0} [-K, K]^d \xrightarrow{g_1} [-Kd, Kd]$. Since for any $\gamma > 1$, $g \in \mathcal{C}_d^\gamma([-K,K]^d, (K+1)d)$, we can set $\gamma = (\beta \vee 1)d$. For network architectures $\mathcal{F}(L, \mathbf{p}, s, F)$ satisfying $F \geq Kd \vee 1$, $(4 + \log_2 d)\log n \leq L \lesssim \log n$, $n^{1/(2\beta+1)} \lesssim \min_i p_i \leq \max_i p_i \lesssim n^C$ and $s \asymp n^{1/(2\beta+1)} \log n$, we thus obtain by Theorem 1,

$$R(\widehat{f}, f) \lesssim n^{-\frac{2\beta}{2\beta+1}} \log^2 n.$$

This coincides up to the $\log^2 n$-factor with the minimax estimation rate.

Generalized additive models: Suppose the regression function is of the form

$$f(x_1, \ldots, x_d) = h\Big(\sum_{i=1}^d f_i(x_i)\Big),$$

for some unknown link function $h: \mathbb{R} \to \mathbb{R}$. This can be written as composition of three functions $f = g_2 \circ g_1 \circ g_0$ with g_0 and g_1 as before and $g_2 = h$. If $f_i \in \mathcal{C}_1^\beta([0,1], K)$ and $h \in \mathcal{C}_1^\gamma(\mathbb{R}, K')$, then $f: [0,1]^d \xrightarrow{g_0} [-K,K]^d \xrightarrow{g_1} \mathbb{R} \xrightarrow{g_2} [-K, K]$. For network architectures satisfying the assumptions of Theorem 1, the bound on the estimation rate becomes

$$R(\widehat{f}, f) \lesssim \Big(n^{-\frac{2\beta(\gamma\wedge 1)}{2\beta(\gamma\wedge 1)+1}} + n^{-\frac{2\gamma}{2\gamma+1}}\Big) \log^2 n.$$

For $\beta = \gamma \geq 2$ and β, γ integers, Theorem 2.1 of [18] establishes the estimation rate $n^{-2\beta/(2\beta+1)}$ which matches with our convergence rate up to the $\log^2 n$-factor.

Multiplicative models: Assume that the function f can be written as

$$f(\mathbf{x}) = \sum_{\ell=1}^N a_\ell \prod_{i=1}^d f_{i\ell}(x_i), \tag{3.3}$$

for fixed N. Especially, if $N = 1$, then $f(\mathbf{x}) = \prod_{i=1}^d f_i(x_i)$. The multiplicative structural constraint can be motivated by series estimators. Series estimators are based on the idea that the unknown function is close to a linear combination of few basis functions, where the approximation error depends on the smoothness of the signal. This means that any L^2-function can be approximated by $f(\mathbf{x}) \approx \sum_{\ell=1}^N a_\ell \prod_{i=1}^d \phi_{i\ell}(x_i)$ for suitable coefficients a_ℓ. Therefore, (3.3) generates a rich function class.

One should also notice that (3.3) is much more general than series expansions. Firstly, (3.3) will be satisfied whenever there exists a basis such that f has an expansion with finite terms, while for series estimators we need to pick a basis. Secondly, the functions $f_{i\ell}$ are arbitrary and hence do not need to be orthogonal.

We can rewrite (3.3) as a composition of functions $f = g_2 \circ g_1 \circ g_0$ with $g_0(\mathbf{x}) = (f_{i\ell}(x_i))_{i,\ell}$, $g_1 = (g_{1j})_{j=1,\ldots,N}$ performing the N multiplications $\prod_{i=1}^d$ and $g_2(\mathbf{y}) = \sum_{\ell=1}^N a_\ell y_\ell$. Observe that $t_0 = 1$ and $t_1 = d$. Assume that $f_{i\ell} \in \mathcal{C}_1^\beta([0,1], K)$ and $\max_\ell |a_\ell| \leq 1$. Because of $g_{1,j} \in \mathcal{C}_d^\gamma([-K,K]^d, d!K^d)$ for all $\gamma \geq d+1$ and $g_2 \in \mathcal{C}_1^{\gamma'}([-d!K^d, d!K^d], Nd!K^d)$ for $\gamma' \geq 1$, we have $[0,1]^d \xrightarrow{g_0} [-K,K]^{Nd} \xrightarrow{g_1} [-d!K^d, d!K^d]^N \xrightarrow{g_2} [-Nd!K^d, Nd!K^d]$. Applying Theorem 1 with $\gamma = (\beta d) \vee (d+1)$ and $\gamma' = N\beta + 1$, the rate for estimators based on suitable network architectures is bounded by

$$R(\widehat{f}, f) \lesssim n^{-\frac{2\beta}{2\beta+1}} \log^2 n,$$

which is unaffected by the curse of dimensionality.

4 A brief summary of related statistical theory for neural networks

This section is intended as a condensed overview on related literature summarizing main proving strategies for bounds on the statistical risk. To control the stochastic error of neural networks, bounds on the covering entropy and VC dimension can be found in the monograph [1]. The challenging part in the analysis of neural networks is the approximation theory for multivariate functions. We first recall results for shallow neural networks, that is, neural networks with one hidden layer.

Shallow neural networks and cosine expansions: A shallow network with one output unit and width vector $(d, m, 1)$ can be written as

$$f_m(\mathbf{x}) = \sum_{j=1}^m c_j \sigma\big(\mathbf{w}_j^t \mathbf{x} + v_j\big), \quad \mathbf{w}_j \in \mathbb{R}^d,\ v_j, c_j \in \mathbb{R}. \tag{4.1}$$

The universal approximation theorem states that a neural network with one hidden layer can approximate any continuous function f arbitrarily well with respect to the uniform norm provided there are enough hidden units, cf. [16, 17, 9, 27, 35]. If f has a derivative f', then the derivative of the neural network also approximates f'. The number of required hidden units might be, however, extremely large, cf. [33] and [32].

The essential idea of the universal approximation theory can be described as follows. Recall the addition theorem $\cos(u)\cos(v) = \frac{1}{2}(\cos(u+v) + \cos(u-v))$. Using this together with the fact that any function $f \in L^2[0,1]^d$ can be expanded in the tensorized cosine basis, we obtain $f(x_1, \ldots, x_d) = \sum_{(i_1,\ldots,i_d) \in \mathbb{N}^d} a_{i_1 \ldots i_d} \prod_{j=1}^d \cos(i_j \pi x_j) = \sum_j \widetilde{c}_j \cos(\widetilde{\mathbf{w}}_j^t \mathbf{x})$ for suitable $\widetilde{\mathbf{w}}_j \in \mathbb{R}^d$ and $\widetilde{c}_j \in \mathbb{R}$. We can bring this then in the form (4.1) by approximating $\cos(\cdot)$ through linear combinations of $\sigma(a \cdot + b)$.

This idea can be sharpened in order to obtain rates of convergence. In [30] the convergence rate $n^{-2\beta/(2\beta+d+5)}$ is derived. Compared with the minimax estimation rate this is suboptimal by a polynomial factor. The reason for the loss of performance with this approach is that rewriting the function as a network requires too many parameters.

In [4, 5, 20, 21] a similar strategy is used to derive the rate $C_f(d\frac{\log n}{n})^{1/2}$ for the squared L^2-risk, where $C_f := \int |\omega|_1 |\mathcal{F}f(\omega)| d\omega$ and $\mathcal{F}f$ denotes the Fourier transform of f. If $C_f < \infty$ and d is fixed the rate is always $n^{-1/2}$ up to logarithmic factors. Since $\sum_i \|\partial_i f\|_\infty \leq C_f$, this means that $C_f < \infty$ can only hold if f has Hölder smoothness at least one. This rate is difficult to compare with the standard nonparametric rates except for the special case $d = 1$, where the rate is suboptimal compared with the minimax rate $n^{-2/(2+d)}$ for d-variate functions with smoothness one. More interestingly, the rate $C_f(d\frac{\log n}{n})^{1/2}$ shows that neural networks can converge fast if the underlying function satisfies some additional structural constraint. In a similar fashion, [2] studies abstract function spaces on which shallow networks achieve fast convergence rates.

Results for multilayer neural networks: [22, 14, 24, 23] use two-layer neural networks with sigmoidal activation function and achieve the nonparametric rate $n^{-2\beta/(2\beta+d)}$ up to $\log n$-factors. Unfortunately, the result requires that the activation function is at least as smooth as the signal, cf. Theorem 1 in [6]. It therefore rules out the ReLU activation function.

The activation function $\sigma(x) = x^2$ is not of practical relevance but has some interesting theory. Indeed, with one hidden layer, we can generate quadratic polynomials and with L hidden layers polynomials of degree 2^L. For this activation function, the role of the network depth is the polynomial degree and we can use standard results to approximate functions in common function classes. A natural generalization is the class of activation functions satisfying $\lim_{x\to-\infty} x^{-k}\sigma(x) = 0$ and $\lim_{x\to+\infty} x^{-k}\sigma(x) = 1$.

If the growth is at least quadratic ($k \geq 2$), the approximation theory has been derived in [31] for deep networks with number of layers scaling with $\log d$. The same class has also been considered recently in [7]. For the approximations to work, the assumption $k \geq 2$ is crucial and the same approach does not generalize to the ReLU activation function, which satisfies the growth condition with $k = 1$, and always produces functions that are piecewise linear in the input.

Approximation theory for the ReLU activation function has been only recently developed in [37, 28, 41]. The key observation is that there are specific deep networks with few units which approximate the square function well. In particular, the function approximation presented in [41] is essential for our approach and we use a similar strategy to construct networks that are close to a given function. An additional difficulty in our approach is the explicit control on the network architecture and the network parameters.

5 Proofs

5.1 Embedding properties of network function classes

For the approximation of a function by a network, we first construct smaller networks computing simpler objects. To combine networks, we make frequently use of the following rules. Recall that $\mathbf{p} = (p_0, \ldots, p_{L+1})$ and $\mathbf{p}' = (p'_0, \ldots, p'_{L+1})$.

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

Composition: 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 $\mathcal{F}(L + L' + 1, (\mathbf{p}, p'_1, \ldots, p'_{L'+1}))$. In most of the cases that we consider, the output of the first network is non-negative and the shift vector $\mathbf{v}$ will be taken to be zero.

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

$$\mathcal{F}(L, \mathbf{p}, s) \subset \mathcal{F}(L + q, (\underbrace{p_0, \ldots, p_0}_{q \text{ times}}, \mathbf{p}), s + qd). \tag{5.1}$$

Parallelization: Given two networks with the same number of hidden layers and the same input dimension, that is, $f \in \mathcal{F}(L, \mathbf{p})$ 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 $\mathcal{F}(L, (p_0, p_1 + p'_1, \ldots, p_{L+1} + p'_{L+1}))$.

We frequently make use of the fact that for a fully connected network in $\mathcal{F}(L, \mathbf{p})$ the number of parameters is

$$\sum_{\ell=0}^{L} (p_\ell + 1) p_{\ell+1} - p_{L+1}. \tag{5.2}$$

5.2 Approximation of polynomials by neural networks

In a first step, we construct a network, with all network parameters bounded by one, which approximately computes xy given input x and y. Let $T^k : [0, 2^{2-2k}] \to [0, 2^{-2k}]$,

$$T^k(x) := (x/2) \wedge (2^{1-2k} - x/2) = (x/2)_+ - (x - 2^{1-2k})_+$$

and $R^k : [0,1] \to [0, 2^{-2k}]$,

$$R^k := T^k \circ T^{k-1} \circ \dots T^1.$$

The next result shows that $\sum_{k=1}^m R^k(x)$ approximates $x(1-x)$ exponentially fast in m and that in particular $x(1-x) = \sum_{k=1}^\infty R^k(x)$ in $L^\infty[0,1]$. This lemma can be viewed as a slightly sharper variation of Lemma 2.4 in [37] and Proposition 2 in [41]. In contrast to the existing results, we can use it to build networks with parameters bounded by one. It also provides an explicit bound on the approximation error.

Lemma 1.

$$\Big|x(1-x) - \sum_{k=1}^m R^k(x)\Big| \le 2^{-m}.$$

Proof. In a first step, we show by induction that R^k is a triangle wave. More precisely, R^k is piecewise linear on the intervals $[\ell/2^k, (\ell+1)/2^k]$ with endpoints $R^k(\ell/2^k) = 2^{-2k}$ if ℓ is odd and $R^k(\ell/2^k) = 0$ if ℓ is even. For $R^1 = T^1$ this is obviously true. For the inductive step, suppose this is true for R^k. Write $\ell \equiv r \mod 4$ if $\ell - r$ is divisible by 4 and consider $x \in [\ell/2^{k+1}, (\ell+1)/2^{k+1}]$. If $\ell \equiv 0 \mod 4$ then, $R^k(x) = 2^{-k}(x - \ell/2^{k+1})$. Similar for $\ell \equiv 2 \mod 4$, $R^k(x) = 2^{-2k} - 2^{-k}(x - \ell/2^{k+1})$; for $\ell \equiv 1 \mod 4$, we have $\ell + 1 \equiv 2 \mod 4$ and $R^k(x) = 2^{-2k-1} + 2^{-k}(x - \ell/2^{k+1})$; and for $\ell \equiv 3 \mod 4$, $R^k(x) = 2^{-2k-1} - 2^{-k}(x - \ell/2^{k+1})$. Together with

$$\begin{aligned} R^{k+1}(x) &= T^{k+1} \circ R^k(x) \\ &= \frac{R^k(x)}{2} \mathbf{1}(R^k(x) \le 2^{-2k-1}) + \Big(2^{-2k-1} - \frac{R^k(x)}{2}\Big) \mathbf{1}(R^k(x) > 2^{-2k-1}). \end{aligned}$$

this shows the claim for R^{k+1} and completes the induction.

For convenience, write $g(x) = x(1-x)$. In the next step, we show that for any $m \ge 1$ and any $\ell \in \{0, 1, \dots, 2^m\}$,

$$g(\ell 2^{-m}) = \sum_{k=1}^m R^k(\ell 2^{-m}).$$

We prove this by induction over m. For $m = 1$ the result can be checked directly. For the inductive step, suppose that the claim holds for m. If ℓ is even we use that $R^{m+1}(\ell 2^{-m-1}) = 0$ to obtain that $g(\ell 2^{-m-1}) = \sum_{k=1}^m R^k(\ell 2^{-m-1}) = \sum_{k=1}^{m+1} R^k(\ell 2^{-m-1})$. It thus remains to consider ℓ odd. Recall that $x \mapsto \sum_{k=1}^m R^k(x)$ is linear on $[(\ell-1)2^{-m-1}, (\ell+1)2^{-m-1}]$ and observe that for any real t,

$$g(x) - \frac{g(x+t) + g(x-t)}{2} = t^2.$$

Using this for $x = \ell 2^{-m-1}$ and $t = 2^{-m-1}$ yields for odd ℓ due to $R^{m+1}(\ell 2^{-m-1}) = 2^{-2m-2}$,

$$g(\ell 2^{-m-1}) = 2^{-2m-2} + \sum_{k=1}^{m} R^k(\ell 2^{-m-1}) = \sum_{k=1}^{m+1} R^k(\ell 2^{-m-1}).$$

This completes the inductive step.

So far we proved that $\sum_{k=1}^m R^k(x)$ interpolates g at the points $\ell 2^{-m}$ and is linear on the intervals $[\ell 2^{-m}, (\ell+1)2^{-m}]$. Observe also that g is Lipschitz with Lipschitz constant one. Thus, for any x, there exists an ℓ such that

$$\Big|g(x) - \sum_{k=1}^{m} R^k(x)\Big| = \Big|g(x) - (2^m x - \ell) g\big((\ell+1)2^{-m}\big) - (\ell + 1 - 2^m x) g(\ell 2^{-m})\Big| \leq 2^{-m}.$$

$\square$

Let $g(x) = x(1-x)$ as in the last proof. To construct a network which returns approximately xy given input x and y, we use the polarization type identity

$$g\Big(\frac{x-y+1}{2}\Big) - g\Big(\frac{x+y}{2}\Big) + \frac{x+y}{2} - \frac{1}{4} = xy. \tag{5.3}$$

Lemma 2. *There exists a network* $\mathrm{Mult}_m \in \mathcal{F}(m+4, (2, 6, 6, \ldots, 6, 1))$, *such that* $\mathrm{Mult}_m(x, y) \in [0, 1]$ *and*

$$\big|\,\mathrm{Mult}_m(x, y) - xy\big| \leq 2^{-m}, \quad \textit{for all } x, y \in [0, 1].$$

Proof. Write $T_k(x) = (x/2)_+ - (x - 2^{1-2k})_+ = T_+(x) - T_-^k(x)$ with $T_+(x) = (x/2)_+$ and $T_-^k(x) = (x - 2^{1-2k})_+$ and let $h : [0,1] \to [0, \infty)$ be a non-negative function. In a first step we show that there is a network N_m with m hidden layers and width vector $(3, 3, \ldots, 3, 1)$ that computes the function

$$\big(T_+(u), T_-^1(u), h(u)\big) \mapsto \sum_{k=1}^{m+1} R^k(u) + h(u),$$

for all $u \in [0, 1]$. The proof is given in Figure 2. Notice that all parameters in this networks are bounded by one. In a next step, we show that there is a network with $m+3$ hidden layers that computes the function

$$(x, y) \mapsto \Big(\sum_{k=1}^{m+1} R^k\Big(\frac{x-y+1}{2}\Big) - \sum_{k=1}^{m+1} R^k\Big(\frac{x+y}{2}\Big) + \frac{x+y}{2} - \frac{1}{4}\Big)_+ \wedge 1.$$

This network computes in the first layer

$$(x, y) \mapsto \Big(T_+\Big(\frac{x-y+1}{2}\Big), T_-^1\Big(\frac{x-y+1}{2}\Big), \Big(\frac{x+y}{2}\Big)_+, T_+\Big(\frac{x+y}{2}\Big), T_-^1\Big(\frac{x+y}{2}\Big), \frac{1}{4}\Big).$$

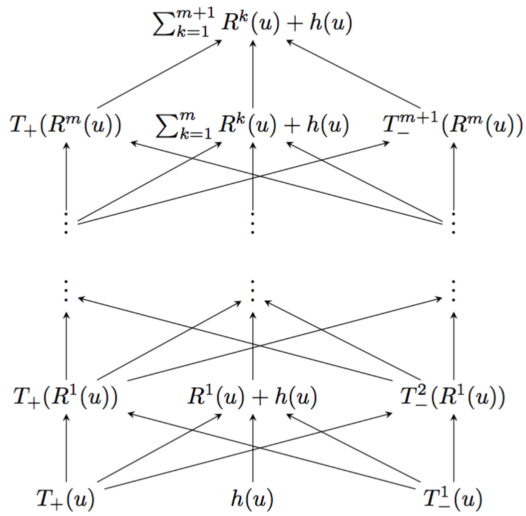

Figure 2: The network $(T_+(u), T_-^1(u), h(u)) \mapsto \sum_{k=1}^{m+1} R^k(u) + h(u)$.

On the first three and the last three components, we apply the network N_m. This gives a network with $m+1$ hidden layers and width vector $(2, 3, \ldots, 3, 2)$ that computes

$$(x, y) \mapsto \Big(\sum_{k=1}^{m+1} R^k\Big(\frac{x-y+1}{2}\Big) + \frac{x+y}{2}, \sum_{k=1}^{m+1} R^k\Big(\frac{x+y}{2}\Big) + \frac{1}{4}\Big).$$

Apply to the output the two hidden layer network $(u, v) \mapsto (1-(1-(u-v))_+)_+ = (u-v)_+ \wedge 1$. The combined network $\mathrm{Mult}_m(x, y)$ has thus $m+4$ hidden layers and computes

$$(x, y) \mapsto \Big(\sum_{k=1}^{m+1} R^k\Big(\frac{x-y+1}{2}\Big) - \sum_{k=1}^{m+1} R^k\Big(\frac{x+y}{2}\Big) + \frac{x+y}{2} - \frac{1}{4}\Big)_+ \wedge 1.$$

This shows that the output is always in $[0,1]$. By (5.3) and Lemma 1, $|\,\mathrm{Mult}_m(x, y) - xy| \leq 2^{-m}$. $\square$

Denote by $\log_2$ the logarithm with respect to the basis two and write $\lceil x \rceil$ for the smallest integer $\geq x$.

Lemma 3. *There exists a network* $\mathrm{Mult}_m^r \in \mathcal{F}((m+5)\lceil \log_2 r \rceil, (r, 6r, 6r, \ldots, 6r, 1))$ *such that* $\mathrm{Mult}_m^r \in [0,1]$ *and*

$$\Big| \mathrm{Mult}_m^r(\mathbf{x}) - \prod_{i=1}^{r} x_i \Big| \leq 3^r 2^{-m}, \quad \textit{for all} \;\; \mathbf{x} = (x_1, \ldots, x_r) \in [0,1]^r.$$

Proof. Let $q := \lceil \log_2(r) \rceil$. Let us now describe the construction of the Mult_m^r network. In the first hidden layer the network computes

$$(x_1, \ldots, x_r) \mapsto (x_1, \ldots, x_r, \underbrace{1, \ldots, 1}_{2^q - r}).$$

Next, apply the network Mult_m in Lemma 2 to the pairs (x_1, x_2), (x_3, x_4),…, $(1, 1)$ in order to compute the vector $(\mathrm{Mult}_m(x_1, x_2), \mathrm{Mult}_m(x_3, x_4), \ldots, \mathrm{Mult}_m(1, 1))$ which has length 2^{q-1}. Now, we pair neighboring entries and apply Mult_m again. This procedure is continued until there is only one entry left. The resulting network is called Mult_m^r and has $q(m+5)$ hidden layers and all parameters bounded by one.

If $a, b, c, d \in [0, 1]$, then by Lemma 2 and triangle inequality, $|\mathrm{Mult}_m(a, b) - cd| \leq 2^{-m} + |a - c| + |b - d|$. By induction on the number of iterated multiplications q, we therefore find that $|\mathrm{Mult}_m^r(\mathbf{x}) - \prod_{i=1}^r x_i| \leq 3^{q-1} 2^{-m}$. □

In the next step, we construct a sufficiently large network that approximates all monomials $x_1^{\alpha_1} \cdot \ldots \cdot x_r^{\alpha_r}$ for non-negative integers α_i up to a certain degree. As common, we use multi-index notation $\mathbf{x}^{\boldsymbol{\alpha}} := x_1^{\alpha_1} \cdot \ldots \cdot x_r^{\alpha_r}$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)$ and $|\boldsymbol{\alpha}| := \sum_\ell |\alpha_\ell|$ is the degree of the monomial. The number of monomials with degree $0 < |\boldsymbol{\alpha}| < \gamma$ is denoted by $C_{r,\gamma}$. Obviously, $C_{r,\gamma} + 1 \leq (\gamma + 1)^r$ since each α_i has to take values in $\{0, 1, \ldots, \lfloor \gamma \rfloor\}$.

Lemma 4. *Given* $\gamma > 0$, *there exists a network*

$$\mathrm{Mon}_{m,\gamma}^r \in \mathcal{F}\big((m+5)\lceil \log_2 r \rceil, (r, 6rC_{r,\gamma}, \ldots, 6rC_{r,\gamma}, C_{r,\gamma})\big),$$

such that $\mathrm{Mon}_{m,\gamma}^r \in [0, 1]^{C_{r,\gamma}}$ *and*

$$\Big| \mathrm{Mon}_{m,\gamma}^r(\mathbf{x}) - (\mathbf{x}^{\boldsymbol{\alpha}})_{0 < |\boldsymbol{\alpha}| < \gamma} \Big|_\infty \leq 3^r 2^{-m}, \quad \textit{for all } \mathbf{x} \in [0, 1]^r.$$

Proof. This follows from computing all monomials with degree $0 < |\boldsymbol{\alpha}| < \gamma$ in parallel. The claim follows from Lemma 3. □

5.3 Reconstruction of a multivariate function with a multilayer neural network

For a vector $\mathbf{a} \in [0, 1]^r$ define

$$P_{\mathbf{a}}^\beta f(\mathbf{x}) = \sum_{0 \leq |\boldsymbol{\alpha}| < \beta} (\partial^{\boldsymbol{\alpha}} f)(\mathbf{a}) \frac{(\mathbf{x} - \mathbf{a})^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}. \tag{5.4}$$

Since $\mathbf{a} \in [0,1]^r$ and $f \in \mathcal{C}_r^\beta([0,1]^r, K)$, $|(\mathbf{x}-\mathbf{a})^{\boldsymbol{\alpha}}| = \prod_i |x_i - a_i|^{\alpha_i} \leq |\mathbf{x}-\mathbf{a}|_\infty^{|\boldsymbol{\alpha}|}$ and by the definition of Hölder balls,

$$\left|P_{\mathbf{a}}^\beta f(\mathbf{x})\right| \leq K. \tag{5.5}$$

By Taylor's theorem for multivariate functions, we have for a suitable $\xi \in [0,1]$,

$$f(\mathbf{x}) = \sum_{\alpha:|\alpha|<\beta-1} (\partial^{\boldsymbol{\alpha}} f)(\mathbf{a}) \frac{(\mathbf{x}-\mathbf{a})^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} + \sum_{\beta-1\leq|\alpha|<\beta} (\partial^{\boldsymbol{\alpha}} f)(\mathbf{a}+\xi(\mathbf{x}-\mathbf{a})) \frac{(\mathbf{x}-\mathbf{a})^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}$$

and so, for $f \in \mathcal{C}_r^\beta([0,1]^r, K)$,

$$\begin{aligned} \left|f(\mathbf{x}) - P_{\mathbf{a}}^\beta f(\mathbf{x})\right| &= \sum_{\beta-1\leq|\alpha|<\beta} \frac{|(\mathbf{x}-\mathbf{a})^{\boldsymbol{\alpha}}|}{\boldsymbol{\alpha}!} \left|(\partial^{\boldsymbol{\alpha}} f)(\mathbf{a}+\xi(\mathbf{x}-\mathbf{a})) - (\partial^{\boldsymbol{\alpha}} f)(\mathbf{a})\right| \\ &\leq K|\mathbf{x}-\mathbf{a}|_\infty^\beta. \end{aligned} \tag{5.6}$$

We may also write (5.4) as a linear combination of monomials

$$P_{\mathbf{a}}^\beta f(\mathbf{x}) = \sum_{0\leq|\boldsymbol{\gamma}|<\beta} \mathbf{x}^{\boldsymbol{\gamma}} c_{\boldsymbol{\gamma}}, \tag{5.7}$$

for suitable coefficients $c_{\boldsymbol{\gamma}}$. Since $\partial^{\boldsymbol{\gamma}} P_a^\beta f(\mathbf{x})\,|_{\mathbf{x}=0} = \boldsymbol{\gamma}! c_{\boldsymbol{\gamma}}$, we must have

$$c_{\boldsymbol{\gamma}} = \sum_{\boldsymbol{\gamma}\leq\boldsymbol{\alpha}\&|\boldsymbol{\alpha}|<\beta} (\partial^{\boldsymbol{\alpha}} f)(\mathbf{a}) \frac{(-\mathbf{a})^{\boldsymbol{\alpha}-\boldsymbol{\gamma}}}{\boldsymbol{\gamma}!(\boldsymbol{\alpha}-\boldsymbol{\gamma})!}.$$

Notice that since $\mathbf{a} \in [0,1]^r$ and $f \in \mathcal{C}_r^\beta([0,1]^r, K)$,

$$|c_{\boldsymbol{\gamma}}| \leq K/\boldsymbol{\gamma}!. \tag{5.8}$$

Consider the set of grid points $\mathbf{D}(M) := \{\mathbf{x}_{\boldsymbol{\ell}} = (\ell_j/M)_{j=1,\ldots,r} : \boldsymbol{\ell} = (\ell_1,\ldots,\ell_r) \in \{0,1,\ldots,M\}^r\}$. The cardinality of this set is $(M+1)^r$. We write $\mathbf{x}_{\boldsymbol{\ell}} = (x_j^{\boldsymbol{\ell}})_j$ to denote the components of $\mathbf{x}_{\boldsymbol{\ell}}$. Define

$$P^\beta f(\mathbf{x}) := \sum_{\mathbf{x}_{\boldsymbol{\ell}}\in\mathbf{D}(M)} P_{\mathbf{x}_{\boldsymbol{\ell}}}^\beta f(\mathbf{x}) \prod_{j=1}^r (1 - M|x_j - x_j^{\boldsymbol{\ell}}|)_+.$$

Lemma 5. *If $f \in \mathcal{C}_r^\beta([0,1]^r, K)$, then $\|P^\beta f - f\|_{L^\infty[0,1]^r} \leq KM^{-\beta}$.*

Proof. Since for all $\mathbf{x} = (x_1,\ldots,x_r) \in [0,1]^r$,

$$\sum_{\mathbf{x}_{\boldsymbol{\ell}}\in\mathbf{D}(M)} \prod_{j=1}^r (1 - M|x_j - x_j^{\boldsymbol{\ell}}|)_+ = \prod_{j=1}^r \sum_{\ell=0}^M (1 - M|x_j - \ell/M|)_+ = 1, \tag{5.9}$$

we have $f(\mathbf{x}) = \sum_{x_{\boldsymbol{\ell}} \in \mathbf{D}(M): \|\mathbf{x}-\mathbf{x}_{\boldsymbol{\ell}}\|_\infty \le 1/M} f(\mathbf{x}) \prod_{j=1}^r (1 - M|x_j - x_j^{\boldsymbol{\ell}}|)_+$ and with (5.6),

$$\big|P^\beta f(\mathbf{x}) - f(\mathbf{x})\big| \le \max_{\mathbf{x}_{\boldsymbol{\ell}} \in \mathbf{D}(M): \|\mathbf{x}-\mathbf{x}_{\boldsymbol{\ell}}\|_\infty \le 1/M} \big|P^\beta_{\mathbf{x}_{\boldsymbol{\ell}}} f(\mathbf{x}) - f(\mathbf{x})\big| \le K M^{-\beta}.$$

□

In a next step, we describe how to build a network that approximates $P^\beta f$.

Lemma 6. *For $M \ge 1$, there exists a network*

$$\mathrm{Hat}^r \in \mathcal{F}\Big(2 + (m+5)\lceil \log_2 r\rceil, \big(r, 6r(M+1)^r, \ldots, 6r(M+1)^r, (M+1)^r\big), s, 1\Big)$$

with $s = 49r^2(M+1)^r(1 + (m+5)\lceil \log_2 r\rceil)$, such that $\mathrm{Hat}^r \in [0,1]^{(M+1)^r}$ and

$$\Big|\mathrm{Hat}^r(\mathbf{x}) - \Big(\prod_{j=1}^r (1/M - |x_j - x_j^{\boldsymbol{\ell}}|)_+\Big)_{\mathbf{x}_\ell \in \mathbf{D}(M)}\Big|_\infty \le 3^r 2^{-m}, \quad \textit{for all } \mathbf{x} = (x_1, \ldots, x_r) \in [0,1]^r.$$

Proof. The first hidden layer computes the functions $(x_j - \ell/M)_+$ and $(\ell/M - x_j)_+$ using $2r(M+1)$ units and $4r(M+1)$ parameters. The second hidden layer computes $(1/M - |x_j - \ell/M|)_+ = (1/M - (x_j - \ell/M)_+ - (\ell/M - x_j)_+)_+ = (1/M - |x_j - \ell/M|)_+$ using $r(M+1)$ units and $3r(M+1)$ parameters. These functions take values in the interval $[0,1]$ and we can apply Lemma 3 to construct a network which computes the products $\prod_{j=1}^r (1/M - |x_j - \ell/M|)_+$ approximately. Each of these Mult_m^r networks is of size $\mathcal{F}((m+5)\lceil \log_2 r\rceil, (r, 6r, 6r, \ldots, 6r, 1))$ and computes $\prod_{j=1}^r (1/M - |x_j - x_j^{\boldsymbol{\ell}}|)_+$ up to an error that is bounded by $3^r 2^{-m}$. By (5.2), the number of parameters of one Mult_m^r network is bounded by $42r^2(1+(m+5)\lceil \log_2 r\rceil)$. As there are $(M+1)^r$ of these networks in parallel, this requires $6r(M+1)^r$ units in each hidden layer and $42r^2(M+1)^r(1+(m+5)\lceil \log_2 r\rceil)$ parameters for the multiplications. Together with the $7r(M+1)$ parameters from the first two layers, the total number of parameters is thus bounded by $49r^2(M+1)^r(1+(m+5)\lceil \log_2 r\rceil)$. □

In the next step, we construct a network which approximately computes $P^\beta f$ and apply Lemma 5.

Theorem 3. *For any function $f \in \mathcal{C}_r^\beta([0,1]^r, K)$ and any integers $m \ge 1$ and $N \ge (\beta+1)^r \vee (K+1)$, there exists a network $\widetilde{f} \in \mathcal{F}\big(L, (r, 12rN, \ldots, 12rN, 1), s, \infty\big)$ with depth*

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

and number of parameters

$$s \le 94r^2(\beta+1)^{2r} N(m+6)(1 + \lceil \log_2 r\rceil),$$

such that

$$\|\widetilde{f} - f\|_{L^\infty([0,1]^r)} \le (2K+1)3^{r+1}N2^{-m} + K2^\beta N^{-\frac{\beta}{r}}.$$

Proof. All the constructed networks in this proof are of the form $\mathcal{F}(L, \mathbf{p}, s) = \mathcal{F}(L, \mathbf{p}, s, \infty)$ with $F = \infty$. Let M be the largest integer such that $(M+1)^r \leq N$ and define $L^* := (m+5)\lceil \log_2 r \rceil$. Thanks to (5.5), (5.8), and (5.7), we can add a layer to the network $\mathrm{Mon}^r_{m,\beta}$ to obtain a network $Q_1 \in \mathcal{F}(1 + L^*, (r, 6rC_{r,\beta}, \ldots, 6rC_{r,\beta}, C_{r,\beta}, (M+1)^r))$, such that $Q_1(\mathbf{x}) \in [0,1]^{(M+1)^r}$ and for any $\mathbf{x} \in [0,1]^r$,

$$\Big|Q_1(\mathbf{x}) - \Big(\frac{P^\beta_{\mathbf{x}_{\boldsymbol{\ell}}} f(\mathbf{x})}{B} + \frac{1}{2}\Big)_{\mathbf{x}_{\boldsymbol{\ell}} \in \mathbf{D}(M)}\Big|_\infty \leq 3^r 2^{-m} \tag{5.10}$$

with $B := \lceil 2K \rceil$. By (5.2), the number of non-zero parameters in the Q_1 network is bounded by $42r^2 C^2_{r,\beta} L^* + (C_{r,\beta}+1)(M+1)^r$.

Recall that the network Hat^r computes the products of hat functions $\prod_{j=1}^r (1/M - |x_j - x_j^{\boldsymbol{\ell}}|)_+$ up to an error that is bounded by $3^r 2^{-m}$. It requires at most $49r^2 N(1+L^*)$ active parameters. Consider now the parallel network (Q_1, Hat^r). Observe that $C_{r,\beta} + 1 \leq (\beta+1)^r \leq N$ by the definition of $C_{r,\beta}$ and the assumptions on N. By Lemma 6, the networks Q_1 and Hat^r can be embedded into a joint network (Q_1, Hat^r) with $2 + L^*$ hidden layers, weight vector $(r, 12rN, \ldots, 12rN, 2(M+1)^r)$ and all parameters bounded by one. The number of non-zero parameters in the combined network (Q_1, Hat^r) is bounded by

$$r + 42r^2 C^2_{r,\beta} L^* + (C_{r,\beta}+1)(M+1)^r + 49r^2 N(1+L^*) \leq 51r^2 (C_{r,\beta}+1)^2 N(1+L^*). \tag{5.11}$$

Next, we pair the $\mathbf{x}_{\boldsymbol{\ell}}$-th entry of the output of Q_1 and Hat^r and apply to each of the $(M+1)^r$ pairs the Mult_m network described in Lemma 2. In the last layer, we add all entries. By Lemma 2 this requires at most $42(m+5)r^2(M+1)^r + (M+1)^r$ active parameters for the $(M+1)^r$ multiplications and the sum. Using Lemma 2, Lemma 6, (5.10) and triangle inequality, there is a network $Q_2 \in \mathcal{F}(3 + (m+5)(1 + \lceil \log_2 r \rceil), (r, 12rN, \ldots, 12rN, 1))$ such that for any $\mathbf{x} \in [0,1]^r$,

$$\begin{aligned} \Big|Q_2 - \sum_{\mathbf{x}_{\boldsymbol{\ell}} \in \mathbf{D}(M)} \Big(\frac{P^\beta_{\mathbf{x}_{\boldsymbol{\ell}}} f(\mathbf{x})}{B} + \frac{1}{2}\Big) \prod_{j=1}^r \Big(\frac{1}{M} - |x_j - x_j^{\boldsymbol{\ell}}|\Big)_+\Big| &\leq 2^{-m} + 3^r 2^{-m} + 3^r 2^{-m} \\ &\leq 3^{r+1} 2^{-m}. \end{aligned} \tag{5.12}$$

Because of (5.11), the network Q_2 has at most

$$94r^2 (C_{r,\beta}+1)^2 N(m+5)(1 + \lceil \log_2 r \rceil) \leq 94r^2 (\beta+1)^{2r} N(m+5)(1 + \lceil \log_2 r \rceil) \tag{5.13}$$

active parameters.

To obtain a network reconstruction of the function f, it remains to scale and shift the output entries. This is not entirely trivial because of the bounded parameter weights in the network. Recall that $B = \lceil 2K \rceil$. Notice that the network $x \mapsto BM^r x$ is in the class $\mathcal{F}(3, (1, M^r, 1, \lceil 2K \rceil, 1))$ with shift vectors $\mathbf{v}_j$ are all equal to zero and weight matrices W_j having all entries equal to one. Because of $N \geq (K+1)$, the number of parameters of this network is bounded by $2M^r + 2\lceil 2K \rceil \leq 6N$. It shows that there is a network in the class $\mathcal{F}(4, (1, 2, 2M^r, 2, 2\lceil 2K \rceil, 1))$ computing $a \mapsto BM^r(a - c)$ with $c := 1/(2M^r)$. This network computes in the first hidden layer $(a - c)_+$ and $(c - a)_+$ and then applies the network $x \mapsto BM^r x$ to both units. In the output layer both entries are added. This requires at most $6 + 12N$ active parameters.

Because of (5.12) and (5.9), there exists a network

$$Q_3 \in \mathcal{F}\big(8 + (m+5)(1 + \lceil \log_2 r \rceil), (r, 12rN, \ldots, 12rN, 1)\big)$$

such that

$$\Big| Q_3(\mathbf{x}) - \sum_{\mathbf{x}_{\boldsymbol{\ell}} \in \mathbf{D}(M)} P^{\beta}_{\mathbf{x}_{\boldsymbol{\ell}}} f(\mathbf{x}) \prod_{j=1}^{r} \Big(1 - M|x_j - x_j^{\boldsymbol{\ell}}|\Big)_+ \Big| \leq (2K+1) M^r 3^{r+1} 2^{-m},$$

for all $\mathbf{x} \in [0,1]^r$. With (5.13), the number of non-zero parameters of Q_3 is bounded by

$$94 r^2 (\beta + 1)^{2r} N (m + 6)(1 + \lceil \log_2 r \rceil).$$

Observe that by construction $(M+1)^r \leq N \leq (M+2)^r \leq (2M)^r$ and hence $M^{-\beta} \leq N^{-\beta/r} 2^{\beta}$. Together with Lemma 5, the result follows. □

Based on the previous result, we can now construct a network that approximates $f = g_1 \circ \ldots \circ g_0$. In a first step, we show that f can always be written as composition of functions defined on hypercubes $[0,1]^{t_i}$. As in the previous theorem, let $g_{ij} \in \mathcal{C}^{\beta_i}_{t_i}([a_i, b_i]^{t_i}, K_i)$ and assume that $K_i \geq 1$. For $i = 1, \ldots, q-1$, 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, $2K_{i-1}\mathbf{x} - K_{i-1}$ means that we transform the entries componentwise by $(2K_{i-1}x_i - K_{i-1})_i$. Clearly,

$$f = g_q \circ \ldots g_0 = h_q \circ \ldots \circ h_0. \tag{5.14}$$

Using the definition of the Hölder balls $C^{\beta}_r(D, K)$, it follows that h_{0j} takes values in $[0,1]$, $h_{0j} \in \mathcal{C}^{\beta_0}_{t_0}([0,1]^{t_0}, 1)$, $h_{ij} \in \mathcal{C}^{\beta_i}_{t_i}([0,1]^{t_i}, (2K_{i-1})^{\beta_i})$ for $i = 1, \ldots, q-1$, and $h_{qj} \in \mathcal{C}^{\beta_q}_{t_q}([0,1]^{t_q}, K_q (2K_{q-1})^{\beta_q})$. Notice that without loss of generality, we can always assume that the radii of the Hölder balls are at least one, that is, $K_i \geq 1$.

Lemma 7. *Let h_{ij} be as above with $K_i \geq 1$. Then, for any functions $\widetilde{h}_i = (\widetilde{h}_{ij})_j^t$ with $\widetilde{h}_{ij} : [0,1]^{t_i} \to [0,1]$,*

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

Proof. Define $H_i = h_i \circ \ldots \circ h_0$ and $\widetilde{H}_i = \widetilde{h}_i \circ \ldots \circ \widetilde{h}_0$. Using triangle inequality,

$$\begin{aligned}\big|H_i(\mathbf{x}) - \widetilde{H}_i(\mathbf{x})\big|_\infty &\leq \big|h_i \circ H_{i-1}(\mathbf{x}) - h_i \circ \widetilde{H}_{i-1}(\mathbf{x})\big|_\infty + \big|h_i \circ \widetilde{H}_{i-1}(\mathbf{x}) - \widetilde{h}_i \circ \widetilde{H}_{i-1}(\mathbf{x})\big|_\infty \\ &\leq K_i \big|H_{i-1}(\mathbf{x}) - \widetilde{H}_{i-1}(\mathbf{x})\big|_\infty^{\beta_i \wedge 1} + \||h_i - \widetilde{h}_i|_\infty\|_{L^\infty[0,1]^{d_i}}.\end{aligned}$$

Together with the inequality $(y+z)^\alpha \leq y^\alpha + z^\alpha$ which holds for all $y, z \geq 0$ and all $\alpha \in [0,1]$, the result follows. $\square$

Proof of Theorem 1. It is enough to prove the result for all sufficiently large n. By Theorem 2 with $\varepsilon = 1$ and the upper bounds on L, $|\mathbf{p}|_\infty$ and s, it follows that there exists a constant C^*, such that for $n \geq 3$,

$$R(\widehat{f}, f) \leq 4 \inf_{f^* \in \mathcal{F}(L,\mathbf{p},s)} \big\|f^* - f\big\|_\infty^2 + C^* \log^2 n \max_{i=0,\ldots,q} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}}.$$

It therefore remains to bound the approximation error. To do this, we rewrite f as in (5.14), that is, $f = h_q \circ \ldots h_0$ with $h_i = (h_{ij})_j^t$ and h_{ij} taking values in $[0,1]$.

Now, we apply Theorem 3 to each $h_{ij} : [0,1]^{t_i} \to [0,1]$ separately. Take $m = \lceil \log_2 n \rceil$ and let $L_i' := 8 + (\lceil \log_2 n \rceil + 5)(1 + \lceil \log_2 t_i \rceil)$. This means there exists a network

$$\widetilde{h}_{ij} \in \mathcal{F}(L_i', (r, 12t_iN, \ldots, 12t_iN, 1), s_i)$$

with $s_i \leq 94r^2(\beta+1)^{2r}N(\lceil \log_2 n \rceil + 6)(1 + \lceil \log_2 t_i \rceil)$, such that

$$\|\widetilde{h}_{ij} - h_{ij}\|_{L^\infty([0,1]^{t_i})} \leq (2Q_i + 1)3^{t_i+1}Nn^{-1} + Q_i 2^\beta N^{-\frac{\beta}{t_i}},$$

where Q_i is any upper bound of the Hölder constants of $\widetilde{h}_{ij}$. If $i < q$, then we apply to the output the two additional layers $1-(1-x)_+$. This requires four additional parameters. Call the resulting network $h_{ij}^* \in \mathcal{F}(L_i'+2, (r, 12t_iN, \ldots, 12t_iN, 1), s+4)$ and observe that $\sigma(h_{ij}^*) = (x \vee 0) \wedge 1$. Since $h_{ij}(\mathbf{x}) \in [0,1]$, we must have $\|\sigma(h_{ij}^*) - h_{ij}\|_{L^\infty([0,1]^r)} \leq \|\widetilde{h}_{ij} - h_{ij}\|_{L^\infty([0,1]^r)}$.

Next, we compute the network

$$h_i^* = (h_{ij}^*)_{j=1,\ldots,d_i} \in \mathcal{F}\big(L_i' + 2, (r, 12d_it_iN, \ldots, 12d_it_iN, d_i), d_i(s+4)\big).$$

Finally, we construct the composite network $f^* = \widetilde{h}_{q1} \circ \sigma(h^*_{q-1}) \circ \ldots \circ \sigma(h^*_0)$ which can be realized in the class

$$\mathcal{F}\Big(E, (r, 12\max_i d_i t_i N, \ldots, 12\max_i d_i t_i N, 1), \sum_{i=0}^q d_i(s_i+4)\Big), \tag{5.15}$$

with $E := -1+\sum_{i=0}^q (L_i'+2)$. Observe that there is an A_n that is bounded in n such that $E = A_n + \log_2 n (\sum_{i=0}^q \lceil \log_2 t_i \rceil + 1)$. For sufficiently large n, $E \leq \sum_{i=0}^q (2+\log_2 t_i)\log_2 n \leq L$. By (5.1) and for sufficiently large n, the space (5.15) can be embedded into $\mathcal{F}(L, \mathbf{p}, s+(L-E)d)$ with $L, \mathbf{p}, s$ satisfying the assumptions of the theorem by choosing $N = \lceil c \max_{i=0,\ldots,q} n^{\frac{t_i}{2\beta_i^*+t_i}} \rceil$ for a sufficiently small constant $c > 0$. The result follows now from Lemma 7 and Theorem 3 noting that $F \geq \|f\|_\infty + 1$ and

$$\big\|f^* - f\big\|_\infty^2 \lesssim \max_{i=0,\ldots,q} N^{-\frac{2\beta_i^*}{t_i}} \asymp \max_{i=0,\ldots,q} n^{-\frac{2\beta_i^*}{2\beta_i^*+t_i}} \to 0.$$

□

5.4 Proof of Theorem 2

Several oracle inequalities for the least-squares estimator are know, cf. [13, 25, 10, 14, 29]. The common assumption of bounded response variables is, however, violated in the nonparametric regression model with Gaussian measurement noise. We therefore include the following result and provide a proof that can be easily generalized to arbitrary noise distributions. Let $\mathcal{N}(\delta, \mathcal{F}, \|\cdot\|_\infty)$ be the covering number, that is, the minimal number of $\|\cdot\|_\infty$-balls with radius δ that covers $\mathcal{F}$ (the centers do not need to be in $\mathcal{F}$).

Lemma 8. *Consider the d-variate nonparametric regression model* (1.1) *with unknown regression function f_0. Let $\widehat{f} \in \arg\min_{f\in\mathcal{F}} \sum_{i=1}^n (Y_i - f(\mathbf{X}_i))^2$ be the least-squares estimator and $\{f_0\} \cup \mathcal{F} \subset \{f : [0,1]^d \to [0,F]\}$ for some $F \geq 1$. If $\mathcal{N}_n := \mathcal{N}(\delta, \mathcal{F}, \|\cdot\|_\infty) \geq 3$, then,*

$$R(\widehat{f}, f) \leq (1+\varepsilon)^2 \Big[\inf_{f\in\mathcal{F}} E_f\big[\big(f(\mathbf{X}) - f_0(\mathbf{X})\big)^2\big] + F^2 \frac{14\log\mathcal{N}_n + 20}{n\varepsilon} + 26\delta F\Big],$$

for all $\delta, \varepsilon \in (0,1]$.

Proof. Define $\|g\|_n^2 := \frac{1}{n}\sum_{i=1}^n g(\mathbf{X}_i)^2$. If $\log \mathcal{N}_n \geq n$ then there is nothing to prove since $R(\widehat{f}, f) \leq 4F^2$. It is therefore enough to show the inequality for $1 \leq \log\mathcal{N}_n \leq n$. For convenience we write $E = E_f$. The proof is divided into two parts. In a first part of the proof, we relate the risk $R(\widehat{f}, f) = E[(\widehat{f}(\mathbf{X}) - f_0(\mathbf{X}))^2]$ to its empirical counterpart $E[\|\widehat{f} - f_0\|_n^2]$ via

$$R(\widehat{f}, f) \leq (1+\varepsilon)\Big(E\big[\|\widehat{f} - f_0\|_n^2\big] + (1+\varepsilon)\frac{F^2}{n\varepsilon}\big(8\log\mathcal{N}_n + 18\big) + 20\delta F\Big). \tag{5.16}$$

In a second part of the proof, we derive

$$E\big[\|\widehat{f}-f_0\|_n^2\big] \le (1+\varepsilon)\Big[\inf_{f\in\mathcal{F}} E_f\big[\big(f(\mathbf{X})-f_0(\mathbf{X})\big)^2\big] + 6\delta + \frac{6\log\mathcal{N}_n+2}{n\varepsilon}\Big]. \tag{5.17}$$

Together, these inequalities yield the assertion since $F \ge 1$.

Let us now establish (5.16). Given a minimal δ-covering of $\mathcal{F}$, denote the centers of the balls by f_j. By construction there exists a (random) j^* such that $\|\widehat{f}-f_{j^*}\|_\infty \le \delta$. Without loss of generality, we can assume that $f_j \le F$. Generate i.i.d. random variables $\mathbf{X}_i'$, $i=1,\ldots,n$ with the same distribution as $\mathbf{X}$ and independent of $(\mathbf{X}_i)_{i=1,\ldots,n}$. Using that $\|f_j\|_\infty, \|f_0\|_\infty, \delta \le F$,

$$\begin{aligned} R(\widehat{f},f) &= E\Big[\|\widehat{f}-f_0\|_n^2 + \frac{1}{n}\sum_{i=1}^n \big(\widehat{f}(\mathbf{X}_i')-f_0(\mathbf{X}_i')\big)^2 - \frac{1}{n}\sum_{i=1}^n \big(\widehat{f}(\mathbf{X}_i)-f_0(\mathbf{X}_i)\big)^2\Big] \\ &\le E\Big[\|\widehat{f}-f_0\|_n^2 + \frac{1}{n}\sum_{i=1}^n g_{j^*}(\mathbf{X}_i,\mathbf{X}_i')\Big] + 10\delta F, \end{aligned}$$

with $g_{j^*}(\mathbf{X}_i,\mathbf{X}_i') := \big(f_{j^*}(\mathbf{X}_i')-f_0(\mathbf{X}_i')\big)^2 - (f_{j^*}(\mathbf{X}_i)-f_0(\mathbf{X}_i))^2$. Define g_j in the same way with f_{j^*} replaced by f_j. Similarly, define $r_j := F\sqrt{56n^{-1}\log\mathcal{N}_n} \vee E^{1/2}[(f_j(\mathbf{X})-f_0(\mathbf{X}))^2]$ and r_{j^*}. Notice that r_{j^*} is a deterministic quantity. If $T := \max_j |\sum_{i=1}^n \frac{g_j(\mathbf{X}_i,\mathbf{X}_i')}{r_j}|$, then

$$R(\widehat{f},f) \le E\big[\|\widehat{f}-f_0\|_n^2\big] + \frac{r_{j^*}}{n}E[T] + 10\delta F. \tag{5.18}$$

Observe that $E[g_j(\mathbf{X}_i,\mathbf{X}_i')]=0$, $|g_j(\mathbf{X}_i,\mathbf{X}_i')| \le 4F^2$ and

$$\operatorname{Var}\big(g_j(\mathbf{X}_i,\mathbf{X}_i')\big) = 2\operatorname{Var}\big((f_j(\mathbf{X}_i')-f_0(\mathbf{X}_i'))^2\big) \le E\big[\big(f_j(\mathbf{X}_i')-f_0(\mathbf{X}_i')\big)^4\big] \le 4F^2 r_j^2.$$

Bernstein's inequality states that for independent and centered random variables $U_1,\ldots,U_n$, satisfying $|U_i| \le M$, $P(|\sum_{i=1}^n U_i| \ge t) \le 2\exp(-t^2/[2Mt/3 + 2\sum_{i=1}^n \operatorname{Var}(U_i)])$, cf. [39]. Combining Bernstein's inequality with a union bound argument yields

$$P(T \ge t) \le 1 \wedge 2\mathcal{N}_n \exp\Big(-\frac{t^2}{8F^2(t/(3r_j)+n)}\Big).$$

Since for any non-negative random variable with finite expectation, $E[X] = \int_0^\infty P(X \ge t)dt$, we have for any $t_0 > 0$,

$$\begin{aligned} E[T] &\le t_0 + \int_{t_0}^\infty 2\mathcal{N}_n e^{-\frac{t^2}{16F^2 n}}dt + \int_{3r_j n}^\infty 2\mathcal{N}_n e^{-\frac{3r_j t}{16F^2}}dt \\ &\le t_0 + \frac{16nF^2}{t_0}\mathcal{N}_n e^{-\frac{t_0^2}{16nF^2}} + \frac{32F^2}{3r_j}\mathcal{N}_n e^{-\frac{9r_j^2 n}{16F^2}}, \end{aligned}$$

where we used for the second inequality the tail bound $\int_a^\infty e^{-x^2/2}dx \le \int_a^\infty a^{-1}xe^{-x^2/2}dx = e^{-a^2/2}/a$ which holds for all $a>0$. Recall that $\log\mathcal{N}_n \ge 1$. With $t_0 := 4F\sqrt{n\log\mathcal{N}_n}$

and the definition of r_j, it follows that $E[T] \leq 4F\sqrt{n \log \mathcal{N}_n} + 6F\sqrt{n}$. Using the triangle inequality with respect to the weighted L^2-norm $E^{1/2}[(f(\mathbf{X}) - g(\mathbf{X}))^2]$, we find $r_{j^*} \leq (F\sqrt{56n^{-1}\log \mathcal{N}_n}) \vee (R(\widehat{f}, f)^{1/2} + \delta)$. If $R(\widehat{f}, f)^{1/2} \leq F\sqrt{56n^{-1}\log \mathcal{N}_n}$ the assertion of the lemma is true since $4 \leq (1+\varepsilon)^2/\varepsilon$ for $\varepsilon > 0$. It is therefore sufficient to consider the case $r_{j^*} \leq R(\widehat{f}, f)^{1/2} + \delta$. Because of $\log \mathcal{N}_n \leq n$, we can conclude that

$$\frac{r_{j^*}}{n} E[T] \leq R(\widehat{f}, f)^{1/2}\Big(4F\sqrt{\frac{\log \mathcal{N}_n}{n}} + \frac{6F}{\sqrt{n}}\Big) + 10F\delta. \tag{5.19}$$

Let a, b, c be positive real numbers, such that $a^2 \leq 2ab + c$. Then, for any $\varepsilon \in (0, 1]$, $a^2 \leq \varepsilon a^2/(1+\varepsilon) + (1+\varepsilon)b^2/\varepsilon + c$ and

$$a^2 \leq (1+\varepsilon)^2 \frac{b^2}{\varepsilon} + (1+\varepsilon)c. \tag{5.20}$$

With $a = R(\widehat{f}, f)^{1/2}$, $b = 2F\sqrt{\log \mathcal{N}_n/n} + 3Fn^{-1/2}$, $c = E[\|\widehat{f} - f_0\|_n^2] + 20\delta F$, we can thus conclude from (5.18) and (5.19) the inequality in (5.16).

Let us now prove (5.17). For any $f \in \mathcal{F}$, we have $\sum_{i=1}^n (Y_i - \widehat{f}(\mathbf{X}_i))^2 \leq \sum_{i=1}^n (Y_i - f(\mathbf{X}_i))^2$. This can be rewritten as

$$\|\widehat{f} - f_0\|_n^2 \leq \|f_0 - f\|_n^2 + \frac{2}{n}\sum_{i=1}^n \varepsilon_i\big(\widehat{f}(\mathbf{X}_i) - f(\mathbf{X}_i)\big). \tag{5.21}$$

Because of $\mathbf{X}_i \overset{\mathcal{D}}{=} \mathbf{X}$ and f being deterministic, we have $E[\|f - f_0\|_n^2] = E[(f(\mathbf{X}) - f_0(\mathbf{X}))^2]$. Since $E[\varepsilon_i f(\mathbf{X}_i)] = E[E[\varepsilon_i f(\mathbf{X}_i) \,|\, \mathbf{X}_i]] = 0$, we also find

$$E\Big[\sum_{i=1}^n \varepsilon_i\big(\widehat{f}(\mathbf{X}_i) - f(\mathbf{X}_i)\big)\Big] = E\Big[\sum_{i=1}^n \varepsilon_i\big(\widehat{f}(\mathbf{X}_i) - f_0(\mathbf{X}_i)\big)\Big]. \tag{5.22}$$

By Cauchy-Schwarz inequality,

$$\begin{aligned} E\Big[\sum_{i=1}^n \varepsilon_i\big(\widehat{f}(\mathbf{X}_i) - f_{j^*}(\mathbf{X}_i)\big)\Big] &\leq \sqrt{n}E\Big[\big(\sum_{i=1}^n \varepsilon_i^2\big)^{1/2}\|\widehat{f} - f_{j^*}\|_n\Big] \\ &\leq \sqrt{n}E^{1/2}\Big[\sum_{i=1}^n \varepsilon_i^2\Big]E^{1/2}\Big[\|\widehat{f} - f_{j^*}\|_n^2\Big] \leq n\delta. \end{aligned} \tag{5.23}$$

Taking expectations on both sides in (5.21) and using (5.22), (5.23) and triangle inequality, we find that with $\widetilde{R}_n(\widehat{f}, f_0) := E\big[\|\widehat{f} - f_0\|_n^2\big]$,

$$\widetilde{R}_n(\widehat{f}, f_0) \leq E\big[\big(f(\mathbf{X}) - f_0(\mathbf{X})\big)^2\big] + 2\delta + \frac{2}{\sqrt{n}}E\Big[(\|\widehat{f} - f_0\|_n + \delta)|\xi_{j^*}|\Big] \tag{5.24}$$

with

$$\xi_j := \frac{\sum_{i=1}^n \epsilon_i(f_j(\mathbf{X}_i) - f_0(\mathbf{X}_i))}{\sqrt{n}\|f_j - f_0\|_n}.$$

Conditionally on $(\mathbf{X}_i)_i$, we have $\xi_j \sim \mathcal{N}(0,1)$. With Lemma 9, we obtain $E[\xi_{j^*}^2] \leq E[\max_j \xi_j^2] \leq 3\log \mathcal{N}_n + 1$. Using Cauchy-Schwarz,

$$E\Big[(\|\widehat{f} - f_0\|_n + \delta)|\xi_{j^*}|\Big] \leq \Big(\widetilde{R}_n(\widehat{f}, f_0)^{1/2} + \delta\Big)\sqrt{3\log \mathcal{N}_n + 1}. \tag{5.25}$$

Because of $\log \mathcal{N}_n \leq n$, we have $2n^{-1/2}\delta\sqrt{3\log \mathcal{N}_n + 1} \leq 4\delta$. Thus, with $a := \widetilde{R}_n(\widehat{f}, f_0)$, $b := \sqrt{(3\log \mathcal{N}_n + 1)/n}$, $c := E[(\widehat{f}(\mathbf{X}) - f_0(\mathbf{X}))^2] + 6\delta$, (5.24), (5.25) and (5.20) yield (5.17). This completes the proof. $\square$

Lemma 9. *Let $\eta_j \sim \mathcal{N}(0,1)$, then $E[\max_{j=1,\dots,M} \eta_j^2] \leq 3\log M + 1$.*

Proof. For $M = 1$ the result is trivial. Let therefore be $M \geq 2$. For $\eta \sim \mathcal{N}(0,1)$ and $T > 0$, we find using integration by parts and $P(\eta > T) \leq e^{-T^2/2}$,

$$\begin{aligned} E[\eta^2 \mathbf{1}(|\eta| > T)] &= (2\pi)^{-1/2}\int_T^\infty x^2 e^{-x^2/2}dx = (2\pi)^{-1/2} T e^{-T^2} + (2\pi)^{-1/2}\int_T^\infty e^{-x^2/2}dx \\ &\leq (1 + (2\pi)^{-1/2}T)e^{-T^2/2}. \end{aligned}$$

Therefore, $E[\max_{j=1,\dots,M} \eta_j^2] \leq T^2 + M(1 + (2\pi)^{-1/2}T)e^{-T^2/2}$. Now, choosing $T = \sqrt{2\log M - 1}$ and using that $(2\pi)^{-1/2}T \leq \sqrt{e}/(4\pi) + T^2 e^{-1/2}/2$ gives $E[\max_{j=1,\dots,M} \eta_j^2] \leq 3T^2/2 + \sqrt{e} + e/(4\pi)^{-1}$. With $\sqrt{e} \leq 2$ and $e \leq \pi$, the result follows. $\square$

Next, we prove a covering entropy bound. Recall the definition of the network function class $\mathcal{F}(L, \mathbf{p}, s, F)$ in (2.3).

Lemma 10. *If $V := \prod_{\ell=0}^{L+1}(p_\ell + 1)$, then, for any $\delta > 0$,*

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

Results of this type have been derived earlier, cf. Theorem 14.5 in [1]. We nevertheless give a full proof of the lemma as we could not find the statement in this form elsewhere.

Proof. Given a network function $f(\mathbf{x}) = W_{L+1}\sigma_{\mathbf{v}_L} W_L \sigma_{\mathbf{v}_{L-1}} \cdots W_2 \sigma_{\mathbf{v}_1} W_1 \mathbf{x}$ we define for $k \in \{1, \dots, L\}$, $A_k^+ f : \mathbb{R}^d \to \mathbb{R}^{p_k}$,

$$A_k^+ f(\mathbf{x}) = \sigma_{\mathbf{v}_k} W_k \sigma_{\mathbf{v}_{k-1}} \cdots W_2 \sigma_{\mathbf{v}_1} W_1 \mathbf{x}$$

and $A_k^- f : \mathbb{R}^{p_{k-1}} \to \mathbb{R}^{p_{L+1}}$,

$$A_k^- f(\mathbf{y}) = W_{L+1}\sigma_{\mathbf{v}_L} W_L \sigma_{\mathbf{v}_{L-1}} \cdots W_{k+1}\sigma_{\mathbf{v}_k} W_k \mathbf{y}.$$

Set $A_0^+ f(\mathbf{x}) = A_{L+2}^+ f(\mathbf{x}) = \mathbf{x}$ and notice that for $f \in \mathcal{F}(L, \mathbf{p})$, $|A_k^+ f(\mathbf{x})|_\infty \leq \prod_{\ell=0}^{k-1}(p_\ell + 1)$. Composition of two Lipschitz functions with Lipschitz constants L_1 and L_2 gives again a

Lipschitz function with Lipschitz constant $L_1 L_2$. Therefore, the Lipschitz constant of $A_k^- f$ is bounded by $\prod_{\ell=k-1}^{L} p_\ell$. Fix $\varepsilon > 0$. Let $f, f^* \in \mathcal{F}(L, \mathbf{p}, s)$ be two network functions, such that all parameters are at most ε away from each other. Then, we can bound the absolute value of the difference by

$$\begin{aligned}
\big|f(\mathbf{x}) - f^*(\mathbf{x})\big| &\leq \sum_{k=1}^{L+1} \Big|A_{k+1}^- f \sigma_{\mathbf{v}_k} W_k A_{k-1}^+ f^*(\mathbf{x}) - A_{k+1}^- f \sigma_{\mathbf{v}_k^*} W_k^* A_{k-1}^+ f^*(\mathbf{x})\Big| \\
&\leq \prod_{\ell=k}^{L} p_\ell \sum_{k=1}^{L+1} \big|\sigma_{\mathbf{v}_k} W_k A_{k-1}^+ f^*(\mathbf{x}) - \sigma_{\mathbf{v}_k^*} W_k^* A_{k-1}^+ f^*(\mathbf{x})\big|_\infty \\
&\leq \prod_{\ell=k}^{L} p_\ell \sum_{k=1}^{L+1} \Big(\big|(W_k - W_k^*) A_{k-1}^+ f^*(\mathbf{x})\big|_\infty + |\mathbf{v}_k - \mathbf{v}_k^*|_\infty\Big) \\
&\leq \varepsilon \prod_{\ell=k}^{L} p_\ell \sum_{k=1}^{L+1} \Big(p_{k-1} \big|A_{k-1}^+ f^*(\mathbf{x})\big|_\infty + 1\Big) \\
&\leq \varepsilon V(L+1).
\end{aligned}$$

By (5.2) the total number of parameters is therefore bounded by $T := \sum_{\ell=0}^{L} (p_\ell + 1) p_{\ell+1} \leq (L+1) 2^{-L} \prod_{\ell=0}^{L+1} (p_\ell + 1) \leq V$ and there are $\binom{T}{s} \leq V^s$ combinations to pick s non-zero parameters. Since all the parameters are bounded in absolute value by one, we can discretize the non-zero parameters with grid size $\delta/(2(L+1)V)$ and obtain for the covering number

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

Taking logarithms yields the result. □

Proof of Theorem 2. The assertion follows from Lemma 10 with $\delta = 1/n$ and Lemma 8 since $F \geq 1$. □

Acknowledgments

The author is grateful to Roy Han for pointing out a mistake in the first version.

References

[1] Anthony, M., and Bartlett, P. L. *Neural network learning: theoretical foundations.* Cambridge University Press, Cambridge, 1999.

[2] Bach, F. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research 18*, 19 (2017), 1–53.

[3] Baraud, Y., and Birgé, L. Estimating composite functions by model selection. *Ann. Inst. Henri Poincaré Probab. Stat. 50*, 1 (2014), 285–314.

[4] Barron, A. R. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory 39*, 3 (1993), 930–945.

[5] Barron, A. R. Approximation and estimation bounds for artificial neural networks. *Machine Learning 14*, 1 (1994), 115–133.

[6] Bauer, B., and Kohler, M. On deep learning as a remedy for the curse of dimensionality in nonparametric regression. available on `http://www3.mathematik.tu-darmstadt.de/hp/ag-stochastik/kohler-michael/publikationen.html`.

[7] Bölcskei, H., Grohs, P., Kutyniok, G., and Petersen, P. Optimal approximation with sparsely connected deep neural networks. *ArXiv e-prints* (May 2017).

[8] Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. The loss surface of multilayer networks. In *Aistats* (2015), pp. 192–204.

[9] Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems 2*, 4 (1989), 303–314.

[10] Giné, E., and Koltchinskii, V. Concentration inequalities and asymptotic results for ratio type empirical processes. *Ann. Probab. 34*, 3 (2006), 1143–1216.

[11] Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *Aistats* (2011), vol. 15, pp. 315–323.

[12] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning.* MIT Press, 2016.

[13] Györfi, L., Kohler, M., Krzyżak, A., and Walk, H. *A distribution-free theory of nonparametric regression.* Springer Series in Statistics. Springer-Verlag, New York, 2002.

[14] Hamers, M., and Kohler, M. Nonasymptotic bounds on the L^2-error of neural network regression estimates. *Annals of the Institute of Statistical Mathematics 58*, 1 (2006), 131–151.

[15] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

[16] Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks 2*, 5 (1989), 359 – 366.

[17] Hornik, K., Stinchcombe, M., and White, H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks 3*, 5 (1990), 551 – 560.

[18] Horowitz, J. L., and Mammen, E. Rate-optimal estimation for a general class of nonparametric regression models with unknown link functions. *Ann. Statist. 35*, 6 (2007), 2589–2619.

[19] Juditsky, A. B., Lepski, O. V., and Tsybakov, A. B. Nonparametric estimation of composite functions. *Ann. Statist. 37*, 3 (2009), 1360–1404.

[20] Klusowski, J. M., and Barron, A. R. Risk bounds for high-dimensional ridge function combinations including neural networks. *ArXiv e-prints* (2016).

[21] Klusowski, J. M., and Barron, A. R. Uniform approximation by neural networks activated by first and second order ridge splines. *ArXiv e-prints* (2016).

[22] Kohler, M., and Krzyzak, A. Adaptive regression estimation with multilayer feedforward neural networks. *Journal of Nonparametric Statistics 17*, 8 (2005), 891–913.

[23] Kohler, M., and Krzyżak, A. Nonparametric regression based on hierarchical interaction models. *IEEE Trans. Inform. Theory 63*, 3 (2017), 1620–1630.

[24] Kohler, M., and Mehnert, J. Analysis of the rate of convergence of least squares neural network regression estimates in case of measurement errors. *Neural Networks 24*, 3 (2011), 273 – 279.

[25] Koltchinskii, V. Local Rademacher complexities and oracle inequalities in risk minimization. *Ann. Statist. 34*, 6 (2006), 2593–2656.

[26] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[27] Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks 6*, 6 (1993), 861 – 867.

[28] Liang, S., and Srikant, R. Why deep neural networks for function approximation? *ArXiv e-prints* (Oct. 2016).

[29] Massart, P., and Picard, J. *Concentration Inequalities and Model Selection*. Lecture notes in mathematics. Springer, 2007.

[30] McCaffrey, D. F., and Gallant, A. R. Convergence rates for single hidden layer feedforward networks. *Neural Networks 7*, 1 (1994), 147 – 158.

[31] Mhaskar, H. N. Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics 1*, 1 (1993), 61–80.

[32] Montúfar, G. F. Universal approximation depth and errors of narrow belief networks with discrete units. *ArXiv e-prints* (Mar. 2013).

[33] Pascanu, R., Montúfar, G., and Bengio, Y. On the number of response regions of deep feed forward networks with piece-wise linear activations. *ArXiv e-prints* (Dec. 2013).

[34] Ray, K., and Schmidt-Hieber, J. A regularity class for the roots of nonnegative functions. *Annali di Matematica Pura ed Applicata* (2017), 1–13.

[35] Stinchcombe, M. Neural network approximation of continuous functionals and continuous functions on compactifications. *Neural Networks 12*, 3 (1999), 467 – 477.

[36] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *ICLR 2016 Workshop* (2016).

[37] Telgarsky, M. Benefits of depth in neural networks. *ArXiv e-prints* (Feb. 2016).

[38] Tsybakov, A. B. *Introduction to nonparametric estimation*. Springer Series in Statistics. Springer, New York, 2009. Revised and extended from the 2004 French original, Translated by Vladimir Zaiats.

[39] van der Vaart, A. W., and Wellner, J. A. *Weak convergence and empirical processes*. Springer Series in Statistics. Springer-Verlag, New York, 1996. With applications to statistics.

[40] Wasserman, L. *All of nonparametric statistics*. Springer Texts in Statistics. Springer, New York, 2006.

[41] Yarotsky, D. Error bounds for approximations with deep ReLU networks. *Neural Networks 94* (2017), 103 – 114.