Estimation of a function of low local dimensionality by deep neural networks

Michael Kohler, Adam Krzyżak, *Fellow, IEEE,* and Sophie Langer

Abstract—Deep neural networks (DNNs) achieve impressive results for complicated tasks like object detection on images and speech recognition. Motivated by this practical success, there is now a strong interest in showing good theoretical properties of DNNs. To describe for which tasks DNNs perform well and when they fail, it is a key challenge to understand their performance. The aim of this paper is to contribute to the current statistical theory of DNNs.
We apply DNNs on high dimensional data and we show that the least squares regression estimates using DNNs are able to achieve dimensionality reduction in case that the regression function has locally low dimensionality. Consequently, the rate of convergence of the estimate does not depend on its input dimension d, but on its local dimension d^* and the DNNs are able to circumvent the curse of dimensionality in case that d^* is much smaller than d. In our simulation study we provide numerical experiments to support our theoretical result and we compare our estimate with other conventional nonparametric regression estimates. The performance of our estimates is also validated in experiments with real data.

Index Terms—curse of dimensionality, deep neural networks, nonparametric regression, piecewise partitioning, rate of convergence.

1 Introduction

1.1 Nonparametric regression

Motivated by the huge success of deep neural networks in applications (cf., e.g., [1] and the literature cited therein) there is now keen interest in investigating theoretical properties of deep neural networks. In statistical research this is usually done in context of nonparametric regression (cf., [2], [3], [4], [5], [6] and [7]). Here, (X, Y) is an $\mathbb{R}^d \times \mathbb{R}$–valued random vector satisfying $\mathbf{E}\{Y^2\} < \infty$, and given a sample of (X, Y) of size n, i.e., given a data set

$$\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}, \tag{1}$$

where (X, Y), (X_1, Y_1), …, (X_n, Y_n) are i.i.d. random variables, the aim is to construct an estimate

$$m_n(\cdot) = m_n(\cdot, \mathcal{D}_n) : \mathbb{R}^d \to \mathbb{R}$$

of the regression function

$$m : \mathbb{R}^d \to \mathbb{R}, \quad m(x) = \mathbf{E}\{Y|X = x\},$$

such that the L_2 error

$$\int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx)$$

is "small" (see, e.g., [8] for a comprehensive study to nonparametric regression and motivation for the L_2 error).

- *M. Kohler is with the Department of Mathematics, Technical University of Darmstadt, 64289 Darmstadt, Germany E-mail: kohler@mathematik.tu-darmstadt.de*
- *A. Krzyżak is with the Department of Computer Science and Software Engineering, Concordia University, Montréal, QC H3G 1M8, Canada and during sabbatical leave in 2021 with the Department of Electrical Engineering, Westpomeranian University of Technology, Szczecin, Poland. Email: krzyzak@cs.concordia.ca*
- *S.Langer is with the Department of Applied Mathematics, Twente University, 7522 NB Enschede, Netherlands E-mail: s.langer@utwente.nl*

Manuscript received December 04, 2020; revised October 19, 2021. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Projektnummer 57157498 - SFB 805 and by the Natural Sciences and Engineering Research Council of Canada under grant RGPIN-2020-06793.

1.2 Rate of convergence

It is well–known that one needs smoothness assumptions on the regression function in order to derive non–trivial rates of convergence (cf., e.g., Theorem 7.2 and Problem 7.2 in [9] and Section 3 in [10]). Thus we introduce the following definition.

Definition 1. *Let $p = q + s$ for some $q \in \mathbb{N}_0$ and $0 < s \leq 1$, where $\mathbb{N}_0$ is the set of nonnegative integers. A function $f : \mathbb{R}^d \to \mathbb{R}$ is called (p, C)-**smooth**, if for every $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ with $\sum_{j=1}^d \alpha_j = q$ the partial derivative $\partial^q f / (\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d})$ exists and satisfies*

$$\left| \frac{\partial^q f}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}(x) - \frac{\partial^q f}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}(z) \right| \leq C \cdot \|x - z\|^s$$

for all $x, z \in \mathbb{R}^d$, where $\| \cdot \|$ denotes the Euclidean norm.

[11] showed that the optimal minimax rate of convergence in nonparametric regression for (p, C)-smooth functions is $n^{-2p/(2p+d)}$.

1.3 Curse of dimensionality

In case that d is large compared to p the above rate of convergence is rather slow which is a symptom of so-called curse of dimensionality. One way to circumvent it is to impose additional constraints on the structure of the regression function. Recently it was shown, that deep neural networks are able to circumvent the curse of dimensionality whenever suitable hierarchical composition assumptions on

the regression function hold. Here the regression function is contained in the following function class:

Definition 2. *Let $d \in \mathbb{N}$ and $m : \mathbb{R}^d \to \mathbb{R}$ and let $\mathcal{P}$ be a subset of $(0, \infty) \times \mathbb{N}$*
a) *We say that m satisfies a hierarchical composition model of level 0 with order and smoothness constraint $\mathcal{P}$, if there exists a $K \in \{1, \dots, d\}$ such that*

$$m(x) = x^{(K)} \quad \textit{for all } x = (x^{(1)}, \dots, x^{(d)})^\top \in \mathbb{R}^d.$$

b) *We say that m satisfies a hierarchical composition model of level $\ell + 1$ with order and smoothness constraint $\mathcal{P}$, if there exist $(p, K) \in \mathcal{P}$, $C > 0$, $g : \mathbb{R}^K \to \mathbb{R}$ and $f_1, \dots, f_K : \mathbb{R}^d \to \mathbb{R}$, such that g is (p, C)–smooth, $f_1, \dots, f_K$ satisfy a hierarchical composition model of level ℓ with order and smoothness constraint $\mathcal{P}$ and*

$$m(x) = g(f_1(x), \dots, f_K(x)) \quad \textit{for all } x \in \mathbb{R}^d.$$

In case that the order and smoothness constraint of g alternates between (p, d^*) and (∞, K) and g is a sum in every second level, this definition equals the definition of the so–called (p, C)*-smooth generalized hierarchical interaction models* of order d^*, which were introduced by [2]. They showed that for such models suitably defined multilayer neural networks (in which the number of hidden layers depends on the level of the generalized interaction model) achieve the rate of convergence $n^{-2p/(2p+d^*)}$ (up to some logarithmic factor) in case $p \leq 1$. [3] generalized this result for $p > 1$ provided the squashing function is suitably chosen. For the hierarchical composition model of Definition 2, where the smoothness and dimension is fixed within one level, [4] showed (up to some logarithmic factor) a rate of convergence

$$\max_{(p,K)\in\mathcal{P}} n^{-2p/(2p+K)}$$

for sparse neural networks with ReLU activation function. [5] showed that this rate holds even for simple fully connected neural networks and arbitrary hierarchical composition model of Definition 2. All the above mentioned results are optimal up to some logarithmic factor. [12] showed that some of these results hold even without the logarithmic factor. For regression functions with a form of common statistical models, i.e. multivariate adaptive regression splines (MARS), [13] showed that convergence rate by DNNs can also be improved. In case that the regression function is defined on a manifold, [14] showed, that the convergence rate by DNNs depends on the dimension of the manifold. [7] analyzed the performance of DNNs in case that the high-dimensional data have an intrinsic low dimensionality and showed that the convergence rate by DNNs depends only on the intrinsic dimension and not on the input dimension.

1.4 Low local dimensionality

1.4.1 Empirical Motivation

In this article we consider regression functions with low local dimensionality. There exist several examples in the literature, where high dimensional problems can be treated locally in much lower dimension. [15] showed that the probability distribution of a natural scene is highly structured, since, for instance, the neighboring pixel of a natural scene have redundant informations. [16] and [17] analyzed in their research on human motor control some regularities in full-body movement of humans within and across individuals. These regularities also lead to locally low-dimensional data distributions. For instance, they showed that for estimating the inverse dynamics of an arm, a globally 21- dimensional space reduces, on average to 4-6 dimensions locally. And also in our own research it can be reasonably assumed, that the analyzed data set is of a locally low dimensional structure. The data set under study (which is part of the Machine Learning Repository: https://archive.ics.uci.edu/ml/machine-learning-databases/00275/) is related to 2–year usage log of a bike sharing system namely Captial Bike Sharing (CBS) at Washington, D.C., USA ([18]). The data show the hourly aggregated count of rental bikes and 12 attributes, namely the season (1: spring, 2: summer, 3: fall, 4: winter), the year (0: 2011, 1:2012), the month (1 to 12), the hour (0 to 23), holiday (whether the day is holiday (1) or not (0)), the day of the week (1 to 7), workingday (if day is neither weekend nor holiday is 1, otherwise is 0), the weather situation (1: Clear, Few clouds, Partly cloudy, 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist, 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds, 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog), the normalized temperature in Celsius, the normalized feeling temperature in Celsius, the normalized humidity and the normalized windspeed. For this data set we conjecture that depending on the season, the hour and the attribute working day the count of rental bikes depends on different subsets of the other attributes. E.g., in spring and fall during the rush hour on working days the weather is not important at all. But on days which are not working days, it depends mainly on the hour and the weather, where for different seasons different weather attributes are important (like temperature and humidity in summer or weather situation in the spring and in the fall). We evaluate this conjecture by analyzing the intrinsic dimensionality of different subsets of the data set. As a dimension estimator we apply the maximum likelihood estimator (MLE) (see [19]). We consider four different subsets of the data set: (I) Working day and rush hour time (7am - 10am and 4pm - 7pm), (II) holiday, (III) winter season (October - March) and humidity > 0.5 and (IV) summer season (April-September).

TABLE 1: Estimated intrinsic dimension of different subsets of the bike sharing data set.

d	Intrinsic dimensionality			
	(I)	(II)	(III)	(IV)
12	3.1475	2.4289	2.7003	2.1258

The results in Table 1 indicate that the estimated intrinsic dimensionality are significantly less than the real dimension of the data set and that depending on the subset the intrinsic dimensionality is different. In summary, that means that one can reduce dimension locally without losing much information for many high dimensional problems thus avoiding the curse of dimensionality. This finding motivates us to analyze regression functions with low local dimensionality.

1.4.2 Notion of low local dimensionality

We say a function $f : \mathbb{R}^d \to \mathbb{R}$ has a low local dimensionality, if it depends locally only on very few of its components, where in different areas these subsets of variables can be different. The simplest way to define this formally is to assume that there exist $d^* \in \{1, \dots, d\}$, $K \in \mathbb{N}$, disjoint sets $A_1, \dots, A_K \subset \mathbb{R}^d$, functions $f_1, \dots, f_K : \mathbb{R}^{d^*} \to \mathbb{R}$ and subsets of indices $J_1, \dots, J_K \subset \{1, \dots, d\}$ of cardinality at most d^* such that

$$f(x) = \sum_{k=1}^{K} f_k(x_{J_k}) \cdot \mathbb{1}_{A_k}(x) \tag{2}$$

holds for all $x \in \mathbb{R}^d$, where

$$x_{\{j_{k,1},\dots,j_{k,d^*}\}} = (x^{(j_{k,1})}, \dots, x^{(j_{k,d^*})})$$

for $1 \leq j_{k,1} < \cdots < j_{k,d^*} \leq d$. As a consequence of using the indicator function, assumption (2) implies that f in general is not globally smooth, in particular it is not even continuous. In view of many applications where it is intuitively expected that the dependent variable depends smoothly on the independent variables, this does not seem to be realistic.

To avoid this problem, we will allow in the sequel smooth transitions between the different areas $A_1, \dots, A_K$ in (2). To achieve this, we assume that the function f is squeezed between two functions of the form (2). In order to simplify the presentation, we use in the sequel d-dimensional polytopes for the sets $A_1, \dots, A_K$. Since polytopes can be described as the intersection of a finite number of half spaces, we define the local dimensionality as follows:

Definition 3. *A function $f : \mathbb{R}^d \to \mathbb{R}$ has local dimensionality $d^* \in \{1, \dots, d\}$ on $[-A, A]^d$ for $A > 0$ with order (K_1, K_2), $\mathbf{P}_X$-border $\epsilon > 0$ and borders $\delta_{i,k} > 0$ for $i = 1, \dots, K_1$, $k = 1, \dots, K_2$ if there exists $a_{i,k} \in \mathbb{R}^d$ with $\|a_{i,k}\| \leq 1$, $b_{i,k} \in \mathbb{R}$, $J_k \subseteq \{1, \dots, d\}$ with $|J_k| \leq d^*$ for $i = 1, \dots, K_1$, $k = 1, \dots, K_2$ and*

$$f_k : \mathbb{R}^{d^*} \to \mathbb{R}$$

such that for

$$(P_k)_{\delta_k} = \{x \in \mathbb{R}^d : a_{i,k}^T x \leq b_{i,k} - \delta_{i,k} \text{ for } i = 1, \dots, K_1\}$$

and

$$(P_k)^{\delta_k} = \{x \in \mathbb{R}^d : a_{i,k}^T x \leq b_{i,k} + \delta_{i,k} \text{ for } i = 1, \dots, K_1\}$$

with $\delta_k = (\delta_{1,k}, \dots, \delta_{K_1,k})$ we have

$$\sum_{k=1}^{K_2} f_k(x_{J_k}) \cdot 1_{(P_k)_{\delta_k}}(x) \leq f(x) \leq \sum_{k=1}^{K_2} f_k(x_{J_k}) \cdot 1_{(P_k)^{\delta_k}}(x)$$

for $x \in [-A, A]^d$ and

$$\mathbf{P}_X\left(\left(\bigcup_{k=1}^{K_2} (P_k)^{\delta_k} \backslash (P_k)_{\delta_k}\right) \cap [-A, A]^d\right) \leq \epsilon.$$

Fig. 1 shows a function $f(x)$ with $K_2 = 4$, polytopes $P_1 = [-2, 0] \times [-2, 0]$, $P_2 = [-2, 0] \times [0, 2]$, $P_3 = [0, 2] \times [0, 2]$ and $P_4 = [0, 2] \times [-2, 0]$ and functions $f_1(x_1) = \sin(4 \cdot x_1)$, $f_2(x_2) = \exp(x_2)$, $f_3(x_2) = \cos(4 \cdot x_2)$ and $f_4(x_2) = \exp(x_2)$ with smooth transitions between the polytopes.

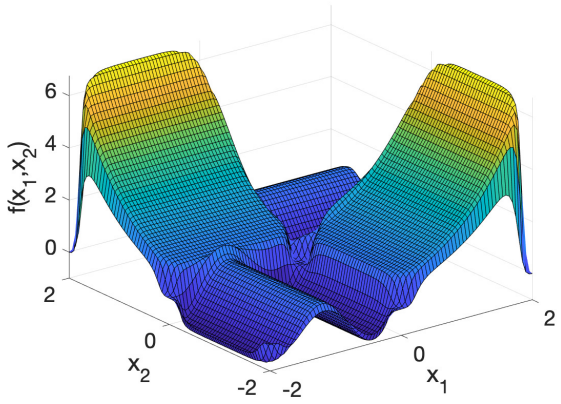

Fig. 1: An example of a function with low local dimensionality with a 2-dimensional support. The support is divided into four pieces, the function depends locally only on one variable of the input and is globally smooth.

1.5 Main results

In this paper we show that sparse neural network regression estimates are able to achieve a dimension reduction in case that the regression function has a low local dimensionality. We derive the rate of convergence which depends only on the local dimension d^* and not on the input dimension d (cf., Theorem 1). Thus our neural network regression estimates are able to circumvent the curse of dimensionality in case that d^* is much smaller than d. Finally, we verify the theoretical results using simulation studies and experiments on real data.

We point out another advantage of DNNs, namely that that neural networks are able to detect a locally low dimensional structure and therefore achieve a faster rate of convergence. As a technical contribution of this paper, we present a result concerning the connection between neural networks and so-called multivariate adapative regression splines (MARS). For instance, we show that the sparse neural network regression estimates, where the weights are chosen by the least squares, satisfy the expected error bound of MARS in case that this procedure works in an optimal way (cf., Theorem 2 in the Supplement).
Our results are based on a set of sparse neural networks instead of fully connected neural networks. On the one hand this network architecture leads to a better bound of the covering number, which is essential to show the convergence result. On the other hand they perform better with regard to the simulated and real data as shown in our simulation studies. In applying our estimates to a real-world data experiment we emphasize the practical relevance of our assumption on the regression function and show that our sparse neural network estimates outperform other nonparametric regression estimates, especially MARS, on this data set.

1.6 Discussion of related results

It is frequently observed by various researchers, that the true intrinsic dimensionality of high dimensional data is often

very low (e.g. [20], [21], [17], [7]). Several estimators like the kernel methods and the Gaussian process regression are able to exploit the intrinsic low dimensionality of covariates and achieve a fast rate of convergence depending only on the intrinsic dimensionality of the data set (e.g. [22], [23], [24], [25]). Recently, [7] also derived convergence rates by DNNs, which only depend on the intrinsic dimension and are free from the nominal dimension. [14] achieved approximation rates of DNNs that only depend on d^*, in case that the input lies on on a d^*-dimension manifold. To describe the intrinsic dimensionality, both articles used the notion of *Minkowski dimension*.

All the above mentioned results use the observation, that many high dimensional problems are contained in a low dimensional space. At this point we would like to highlight the difference with our assumption. While these studies focus on the behavior of the measure $\mathbf{P}_X$ of covariates, we analyze regression functions with some specific structure, i.e. regression functions with low local dimensionality. In our case the dimension of the regression function is locally of size $d^* \leq d$ and the regression function performs differently on different subsets. This does not imply, that there is some intrinsic low dimensionality in the measure $\mathbf{P}_X$ of covariates. This is also clearly visible from the example presented in Fig. 1: The function has a low local dimensionality independently from the choice of the design distribution. I.e., even when the design distribution is not concentrated on a d^*-dimensional manifold (as required e.g. in [4]) the corresponding distribution with this function as regression function has a low local dimensionality in the sense of our paper.

A similar structure of regression functions has been studied by [6]. They analyzed the performance of DNNs for a certain class of piecewise smooth functions. Here piecewise smooth regression functions where the partitions have smooth borders were considered. For instance, their partition consists of a finite number of pieces, where each piece is an intersection of so-called *basis pieces*. Each basis piece is defined with the help of a horizon function and is regarded as one side of surfaces by a Hölder-smooth function. Thus the pieces of the partition in this paper have smooth borders, which is a more flexible way to define piecewise smooth functions, but which does not contain the case of globally smooth functions. Since we also want to take into consideration smooth functions with low local dimensionality, i.e. functions which perform differently on different pieces (depending only on a few components of the input on each piece), but are nevertheless globally smooth, we define our pieces as d–dimensional polytopes and allow smooth transition between them.

As mentioned earlier, the proof of our main result is based on a result that analyzes the connection between DNNs and MARS. [13] already showed a similar result for the ReLU activation function. In particular, they showed that every function expressed as a function in MARS can also be approximated by a multilayer neural network (up to a sup-norm error ϵ). Using this result they derived a risk comparison inequality, that bounds the statistical risk of fitting a neural network by the statistical risk of spline-based methods. Due to the fact that the ReLU activation function and consequently the corresponding neural network are piecewise linear functions it is not that suprisingly to find connection to spline methods. This paper extends this result by showing connection between neural networks with smooth activation functions and MARS, which was not covered by the results in [13]. Additionally, we show our result for a more general basis of smooth piecewiese polynomials, i.e. a product of a truncated power basis of degree 1 and a B-spline basis. This leads to better approximation properties in case of very smooth regression function.

The approximation of B-Splines by DNNs has also been studied by [12]. They showed that a DNN with $const \cdot m$ hidden layers and a fixed number of neurons per layer achieves an approximation rate of size 4^{-2m} for a tensor product B-spline basis. In the Supplement we derive a related result for DNN with squashing activation function.

In all the abovementioned generalization results, it is assumed that the training algorithm finds a global minimum, which is not necessarily the case in practical applications and which, in general, cannot be guaranteed. Training a neural network requires minimizing a high-dimensional non-convex loss function. A series of theoretical studies have considered how *trainable* the parameter spaces of neural networks are. For instance, [26] analyzed the difference between local and global minima and [27] showed that in case of linear activation function every local minima is also a global minimum, but still some saddle points can exist. In our approach we assume that the optimization algorithm indeed finds the global minimum and therefore minimizes the empirical risks. While this simplified setting does not quite correspond to the practical applications, it is a starting point to explain the good performance of the networks from a theoretical point of view. A combination of our results with the literature analyzing the optimization algorithm should at some point enable us to analyze the networks with all three of its aspects, namely approximation, generalization and optimization.

1.7 Notation

Throughout the paper, the following notation is used: The sets of natural numbers, natural numbers including 0, integers, non-negative real numbers and real numbers are denoted by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Z}$, $\mathbb{R}_+$ and $\mathbb{R}$, respectively. For $z \in \mathbb{R}$, we denote the smallest integer greater than or equal to z by $\lceil z \rceil$, and $\lfloor z \rfloor$ denotes the largest integer that is less than or equal to z. Furthermore we set $z_+ = \max\{z, 0\}$. The Euclidean and the supremum norms of $x \in \mathbb{R}^d$ are denoted by $\|x\|_2$ and $\|x\|_\infty$, respectively. For $f : \mathbb{R}^d \to \mathbb{R}$

$$\|f\|_\infty = \sup_{x \in \mathbb{R}^d} |f(x)|$$

is its supremum norm, and the supremum norm of f on a set $A \subseteq \mathbb{R}^d$ is denoted by

$$\|f\|_{\infty, A} = \sup_{x \in A} |f(x)|.$$

A finite collection $f_1, \dots, f_N : \mathbb{R}^d \to \mathbb{R}$ is called an ε–$\|\cdot\|_{\infty,A}$– cover of $\mathcal{F}$ if for any $f \in \mathcal{F}$ there exists $i \in \{1, \dots, N\}$ such that

$$\|f - f_i\|_{\infty,A} = \sup_{x \in A} |f(x) - f_i(x)| < \varepsilon.$$

The ε–$\|\cdot\|_{\infty,A}$- covering number of $\mathcal{F}$ is the size N of the smallest ε–$\|\cdot\|_{\infty,A}$– cover of $\mathcal{F}$ and is denoted by $\mathcal{N}(\varepsilon, \mathcal{F}, \|\cdot\|_{\infty,A})$. We write $x = \arg\min_{z \in D} f(z)$ if $\min_{z \in \mathcal{D}} f(z)$ exists and if x satisfies

$$x \in D \quad \text{and} \quad f(x) = \min_{z \in \mathcal{D}} f(z).$$

If not otherwise stated, any c_i with $i \in \mathbb{N}$ symbolizes a real nonnegative constant, which is independent of the sample size n.

1.8 Outline

The outline of this paper is as follows: In Section 2 the sparse neural network regression estimates analyzed in this paper are introduced. The main result is presented in Section 3. The finite sample size behavior of our estimate is analyzed by applying it to simulated and real data in Section 4.

2 Sparse neural network regression estimates

The starting point in defining a neural network is the choice of an activation function $\sigma : \mathbb{R} \to \mathbb{R}$. We use in the sequel so–called squashing functions, which are nondecreasing and satisfy $\lim_{x \to -\infty} \sigma(x) = 0$ and $\lim_{x \to \infty} \sigma(x) = 1$. An example of a squashing function is the so-called sigmoidal or logistic squasher

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad (x \in \mathbb{R}). \tag{3}$$

A multilayer feedforward neural network with L hidden layers and k_1, k_2, …, k_L number of neurons in the first, second, …, L-th hidden layer and sigmoidal function σ is a real-valued function defined on $\mathbb{R}^d$ of the form

$$f(x) = \sum_{i=1}^{k_L} c_{1,i}^{(L)} \cdot f_i^{(L)}(x) + c_{1,0}^{(L)}, \tag{4}$$

for some $c_{1,0}^{(L)}, \dots, c_{1,k_L}^{(L)} \in \mathbb{R}$ and for $f_i^{(L)}$'s recursively defined by

$$f_i^{(r)}(x) = \sigma\left(\sum_{j=1}^{k_{r-1}} c_{i,j}^{(r-1)} \cdot f_j^{(r-1)}(x) + c_{i,0}^{(r-1)} \right) \tag{5}$$

for some $c_{i,0}^{(r-1)}, \dots, c_{i,k_{r-1}}^{(r-1)} \in \mathbb{R}$ $(r = 2, \dots, L)$ and

$$f_i^{(1)}(x) = \sigma\left(\sum_{j=1}^{d} c_{i,j}^{(0)} \cdot x^{(j)} + c_{i,0}^{(0)} \right) \tag{6}$$

for some $c_{i,0}^{(0)}, \dots, c_{i,d}^{(0)} \in \mathbb{R}$. We denote by $\mathcal{F}(L, r, \alpha)$ the set of all fully connected neural networks with L hidden layers, r neurons in each hidden layer and weights bounded in absolute value by α.

In the sequel we propose sparse neural networks architectures, where the consecutive layers of neurons are not fully connected. The structure of our sparse neural networks depends on smaller neural networks that are fully connected. For $M^* \in \mathbb{N}$, $L \in \mathbb{N}$, $r \in \mathbb{N}$ and $\alpha > 0$, we denote the set of all functions $f : \mathbb{R}^d \to \mathbb{R}$ that satisfy

$$f(x) = \sum_{i=1}^{M^*} \mu_i \cdot f_i(x) \ (x \in \mathbb{R}^d)$$

for some $f_i \in \mathcal{F}(L, r, \alpha)$ and for some $\mu_i \in \mathbb{R}$, where $|\mu_i| \leq \alpha$, by $\mathcal{F}_{M^*,L,r,\alpha}^{(sparse)}$. An example of a network in class $\mathcal{F}_{3,L,r,\alpha}^{(sparse)}$ is shown in Figure 2 which gives a good idea of how the network structure looks like. In the sequel we want to use

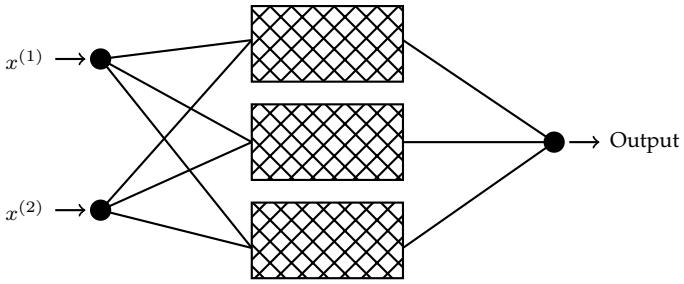

Fig. 2: A neural network with $M^* = 3$ boxes of fully connected neural networks

data (1) in order to choose a function from $\mathcal{F}_{M^*,L,r,\alpha}^{(sparse)}$ such that this function is a good regression estimate. In order to do this, we use the principle of least squares and define our regression estimate $\tilde{m}_n$ as a function

$$\tilde{m}_n(\cdot) = \tilde{m}_n(\cdot, \mathcal{D}_n) \in \mathcal{F}_{M^*,L,r,\alpha_n}^{(sparse)} \tag{7}$$

from $\mathcal{F}_{M^*,L,r,\alpha_n}^{(sparse)}$, which minimizes the so–called empirical L_2 risk over $\mathcal{F}_{M^*,L,r,\alpha_n}^{(sparse)}$, i.e., which satisfies

$$\frac{1}{n} \sum_{i=1}^{n} |Y_i - \tilde{m}_n(X_i)|^2 = \min_{f \in \mathcal{F}_{M^*,L,r,\alpha_n}^{(sparse)}} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(X_i)|^2. \tag{8}$$

Here we assume for notational simplicity that the minimum above does indeed exist. In case that it does not exist our results also hold for any function chosen from $\mathcal{F}_{M^*,L,r,\alpha_n}^{(sparse)}$ which minimizes the empirical L_2 risk in (8) up to some small additive term, e.g., up to $1/n$. For technical reasons in the analysis of our estimate we need to truncate it at some data–independent level β_n satisfying $\beta_n \to \infty$ for $n \to \infty$, i.e., we set

$$m_n(x) = T_{\beta_n} \tilde{m}_n(x) \quad (x \in \mathbb{R}^d), \tag{9}$$

where $T_{\beta_n} z = \max\{\min\{z, \beta_n\}, -\beta_n\}$ for $z \in \mathbb{R}$.

The number L of layers and the number r of parameters of each fully connected neural network f_i will be chosen as a large enough constant. For the bound α_n on the absolute value of the weights we will use a data–independent bound of the form $\alpha_n = c_1 \cdot n^{c_2}$ for some $c_1, c_2 > 0$. The main parameter left which controls the flexibility of the networks is then the number M^* of fully connected neural networks $f_i \in \mathcal{F}(L, r, \alpha_n)$ $(i = 1, \dots, M^*)$. To choose it, we will use the principle of splitting of the sample (cf., e.g., Chapter 7 in Györfi et al. (2002)). Here we split the sample into a learning sample of size n_l and a testing sample of size n_t,

where $n_l, n_t \geq 1$ satisfy $n = n_l + n_t$, e.g., $n_l = \lceil n/2 \rceil$ and $n_t = n - n_l$. We use the learning sample

$$\mathcal{D}_{n_l} = \{(X_1, Y_1), \ldots, (X_{n_l}, Y_{n_l})\}$$

to define for each M^* in $\mathcal{P}_n = \{2^l : l = 1, \ldots, \lceil \log n \rceil\}$ an estimate $\tilde{m}_{n_l, M^*}$ by

$$\tilde{m}_{n_l, M^*}(\cdot) = \tilde{m}_{n_l, M^*}(\cdot, \mathcal{D}_{n_l}) \in \mathcal{F}^{(sparse)}_{M^*, L, r, \alpha_n} \tag{10}$$

and

$$\frac{1}{n_l} \sum_{i=1}^{n_l} |Y_i - \tilde{m}_{n_l, M^*}(X_i)|^2 = \min_{f \in \mathcal{F}^{(sparse)}_{M^*, L, k, \alpha_n}} \frac{1}{n_l} \sum_{i=1}^{n_l} |Y_i - f(X_i)|^2, \tag{11}$$

and set

$$m_{n_l, M^*}(x) = T_{\beta_n} \tilde{m}_{n_l, M^*}(x) \quad (x \in \mathbb{R}^d). \tag{12}$$

Then we choose $M^* \in \mathcal{P}_n$ such that the empirical L_2 error of the estimate on the testing data is minimal, i.e., we define

$$m_n(x, \mathcal{D}_n) = m_{n_l, \hat{M}^*}(x, \mathcal{D}_{n_l}), \tag{13}$$

where $\hat{M}^* \in \mathcal{P}_n$ and

$$\frac{1}{n_t} \sum_{i=n_l+1}^{n} |Y_i - m_{n_l, \hat{M}^*}(X_i)|^2 = \min_{M^* \in \mathcal{P}_n} \frac{1}{n_t} \sum_{i=n_l+1}^{n} |Y_i - m_{n_l, M^*}(X_i)|^2. \tag{14}$$

3 MAIN RESULT

Our theoretical result will be valid for sigmoidal functions which are 2–admissible according to the following definition.

Definition 4. *Let $N \in \mathbb{N}_0$. A function $\sigma : \mathbb{R} \to [0, 1]$ is called **N-admissible**, if it is nondecreasing and Lipschitz continuous and if, in addition, the following three conditions are satisfied:*

(i) *The function σ is $N + 1$ times continuously differentiable with bounded derivatives.*
(ii) *A point $t_\sigma \in \mathbb{R}$ exists, where all derivatives up to order N of σ are nonzero.*
(iii) *If $y > 0$, the relation $|\sigma(y) - 1| \leq \frac{1}{y}$ holds. If $y < 0$, the relation $|\sigma(y)| \leq \frac{1}{|y|}$ holds.*

It is easy to see that the logistic squasher (3) is N–admissible for any $N \in \mathbb{N}$ (cf., e.g. [3]).

Our main result shows, that the sparse neural networks can achieve the d^*–dimensional rate of convergence in case that the regression function has local dimensionality d^*.

Theorem 1. *Let $\beta_n = c_3 \cdot \log(n)$ for some constant $c_3 > 0$. Assume that the distribution of (X, Y) satisfies*

$$\mathbf{E}\left(\exp(c_4 \cdot |Y|^2)\right) < \infty \tag{15}$$

for some constant $c_4 > 0$ and that the distribution of X has bounded support $supp(X) \subseteq [-A, A]^d$ for some $A \geq 1$. Let $M, K_1, K_2 \in \mathbb{N}$. Assume furthermore that m has local dimensionality d^* on $supp(X)$ with order (K_1, K_2), $\mathbf{P}_X$-border c_5/n and borders $\delta_{i,k}$, where $\delta_{i,k} \geq c_6/n^{c_7}$ holds for some constants $c_5, c_6, c_7 > 0$ ($i = 1, \ldots, K_1$, $k = 1, \ldots, K_2$) and where all functions f_k in Definition 3 are bounded and (p, C)–smooth for some $p = q + s$ with $0 < s \leq 1$ and $q \leq M$. Let the least squares neural network regression estimate m_n be defined as in Section 2 with parameters

(i) $L = 3K_1 + d \cdot (M + 2) - 1$
(ii) $r = 2^{M-1} \cdot 16 + \sum_{k=2}^{M} 2^{M-k+1} + d + 5$
(iii) $\alpha_n = c_1 \cdot n^{c_2}$
(iv) $n_l = \lceil n/2 \rceil$.

Assume that the sigmoidal function σ is 2–admissible, and that $c_1, c_2 > 0$ are suitably large. Then we have for any $n > 7$:

$$\mathbf{E} \int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \leq c_8 \cdot (\log n)^3 \cdot 2^{K_1} \cdot K_2 \cdot n^{-\frac{2p}{2p+d^*}}.$$

The proof is available in the Supplement.

Remark 1. *The deep neural network estimate in the above theorem achieves a rate of convergence which is independent of the dimension d of X, hence it is able to circumvent the curse of dimensionality in case that the regression function has low local dimensionality.*

Remark 2. *Theorem 1 states a bound in dependence on K_1 and K_2. The proofs, however, are non-asymptotic although we did not make any attempt to minimize the constants depending on K_1 and K_2. While these large constants might spoil our bound for small sample sizes, we see this as an acceptable drawback, as deep learning methods especially outperform other methods for high sample sizes, where our result indicates a fast rate of convergence.*

Remark 3. *Theorem 1 holds for an estimator that minimizes the empirical risk. In practical applications, it is not clear that we will find this minimum or that the network trained by (stochastic) gradient descent lies even close to this minimum. At the present time optimization and generalization results of deep learning tend to be considered separately from each other. We hope that we can combine these findings in future work.*

Next we show a lower bound for the minimax estimation risk over the class of all regression functions with low local dimensionality d^* satisfying the assumptions of Theorem 1. This shows that the rate of Theorem 1 is optimal.

Theorem 2. *Let $\mathcal{D}$ be the class of all distributions of (X, Y) such that:*

(i) $supp(X) \subseteq [-A, A]^d$ *for some* $A \geq 1$
(ii) $\mathbf{E}\left(\exp(c_4 \cdot |Y|^2)\right) < \infty$
(iii) m *has low local dimensionality* d^* *as in Theorem 1.*

Then we have

$$\inf_{\hat{m}_n} \sup_{(X,Y) \in \mathcal{D}} \mathbf{E} \int |\hat{m}_n(x) - m(x)|^2 \mathbf{P}_X(dx) \geq const \cdot n^{-\frac{2p}{2p+d^*}}.$$

Proof. First we define a subclass of distributions of (X, Y) contained in $\mathcal{D}$. In this subclass

$$m(x) = f_1(x_{J_1}) \cdot 1_{P_1}(x)$$

with $J_1 \in \{1, \ldots, d\}$, $|J_1| = d^*$,

$$P_1 = \{x \in \mathbb{R}^d : a^T x \leq \sqrt{d \cdot A} + 1\}$$

with $a \in \mathbb{R}^d$ and $\|a\| \leq 1$ and f_1 being (p, C)-smooth. Now it is easy to see that all $x \in [-A, A]^d$ are contained in P_1.

In this case $m(x) = f_1(x_{J_1})$ and therefore equals a (p, C)-smooth function with d^*-dimensional input. Arguing as in the proof of Theorem 3.2 in [8] we get

$$\inf_{\hat{m}_n} \sup_{(X,Y)\in\mathcal{D}} \mathbf{E} \int |\hat{m}_n(x) - f_1(x_{J_1})|^2 \mathbf{P}_X(dx) \geq const \cdot n^{-\frac{2p}{2p+d^*}}$$

which implies the assertion. □

Outline of the proof of Theorem 1. In the proof of Theorem 1 the following bound on the expected L_2 error of our sparse neural network regression estimate is essential:

$$\begin{aligned}&\mathbf{E} \int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \\ &\leq (\log n)^3 \cdot \inf_{I\in\mathbb{N},\ B_1,\dots,B_I\in\mathcal{B}^*} \Bigg(c_9 \cdot \frac{I}{n} \\ &+ \min_{(a_i)_{i=1,\dots,I}\in[-c_{10}\cdot n, c_{10}\cdot n]^I} \int |\sum_{i=1}^{I} a_i \cdot B_i(x) \\ &\qquad - m(x)|^2 \mathbf{P}_X(dx) \Bigg). \end{aligned} \tag{16}$$

Here $\mathcal{B}^*$ is a basis consisting of functions representable as a product of a truncated power basis of degress 1, i.e. the MARS function class, and a tensor product B-spline basis. In particular $\mathcal{B}^*$ consists of functions of the form

$$\begin{aligned}B(x) = &\prod_{v\in J_1} B_{j_v,M,t_v}(x^{(i_v)}) \\ &\cdot \prod_{k\in J_2} \left(\sum_{j=1}^{d} \alpha_{k,j} \cdot (x^{(i_j)} - \gamma_{k,j}) \right)_+ \end{aligned} \tag{17}$$

where B_{j_v,M,t_v} represents an univariate B-Splines of degree M with $t_v = \{t_{v,k}\}_{k=-M,\dots,K+M}$ and $j_v \in \{-M, -M+1, \dots, K-1\}$ (see Definition 5 in the supplement) and $J_1 \subseteq \{1,\dots,d\}$, $J_2 \subseteq \{1,\dots,K_1\}$, $K \in \mathbb{N}$, $i_v, i_j \in \{1,\dots,d\}$, $t_{v,k}, \alpha_{k,j}, \gamma_{k,j} \in [-c_{11}\cdot n^{c_{12}}, c_{11}\cdot n^{c_{12}}]$ and $t_{v,k+1} - t_{v,k} \geq \frac{1}{n}$. The proof of (16) is given in the supplement. The idea is to first build networks that approximate the square function. Using polarization identity one can then approximate the product xy given (x, y). Using that $\max\{x, 0\} = x \cdot \mathbb{1}_{[0,\infty)}(x)$ one can then build networks approximating the ReLU function, which finally enables us to approximate $\left(\sum_{j=1}^{d} \alpha_{k,j} \cdot (x^{(i_j)} - \gamma_{k,j})\right)_+$ and the univariate B-splines $B_{j_v,M,t_v}(x^{(i_v)})$ in (17). A combination with a result approximating the product of functions by neural networks finally states networks approximating the basis functions $B(x)$. By computing I of these networks in parallel, we are finally able to approximate the linear combination of these basis functions, i.e.

$$B(x) = \sum_{i=1}^{I} a_i \cdot B_i(x), \quad B_i \in \mathcal{B}^*$$

by sparse neural networks.

Since every function with low local dimensionality d^* (according to Definition 3) can be expressed as a linear combination of functions of $\mathcal{B}^*$ in case that x is not contained in $\left(\left(\bigcup_{k=1}^{K_2} (P_k)^{\delta_k} \backslash (P_k)_{\delta_k}\right) \cap [-A, A]^d\right)$, we can use the bound (16) to show our main result. Here we proceed as follows: First we show that an indicator function of a polytope can be approximated by a linear truncated power basis. In the second step we prove that every (p, C)-smooth function can be approximated by a linear combination of a tensor product B-Spline basis. In the last step we show that every function of the form

$$\sum_{k=1}^{K_2} f_k(x_{J_k}) \cdot \mathbb{1}_{(P_k)^{\delta_k}}(x)$$

with notations according to Definition 3 can be expressed as a linear combination of functions of $\mathcal{B}^*$. Together with the assumption

$$\mathbf{P}_X \left(\left(\bigcup_{k=1}^{K_2} (P_k)^{\delta_k} \backslash (P_k)_{\delta_k} \right) \cap [-A, A]^d \right) \leq \frac{c_5}{n}$$

we conclude the assertion of the Theorem. □

4 Simulation study

To illustrate how the introduced nonparametric regression estimate based on our sparsely connected neural networks behaves in case of finite sample sizes, we apply it to simulated data using the MATLAB software. Due to the fact that our estimate contains some parameters that may influence their behavior, we will choose these parameters in a data-dependent way by splitting of the sample. Here we use $n_{train} = \lceil \frac{4}{5} \cdot n \rceil$ realizations to train the estimate several times with different choices for the parameters and $n_{test} = n - n_{train}$ realizations to test the estimate by comparing the empirical L_2 risk of different parameter settings and choosing the best estimate according to this criterion. The parameters L, r and M^* of the estimates in Section 2 are chosen in a data-dependent way. Here we choose $L = \{1, 3, 6\}$, $r \in \{3, 6, 10\}$ and $M^* \in \{1, 2, \dots, 10\}$. To solve the least squares problem in (8), we use the quasi-Newton method of the function *fminunc* in MATLAB to approximate the solution. All initial weights are chosen independent and uniformly distributed on $[0, 1]$.

The results of our estimate are compared to other conventional estimates. In particular we compare the sparsely connected neural network estimate (abbr. *neural-sc*) to a fully connected neural network (abbr. *neural-fc*) with adaptively chosen number of hidden layers and number of neurons per layer. The selected values of these two parameters to be tested were $\{1, 2, 4, 6, 8, 10, 12\}$ for L and $\{1, 2, \dots, 6, 8, 10\}$ for r. Beside this, we compare our neural network estimate to another sparsely connected neural network estimate, namely the network *neural-x* defined in [3]. The parameters l, K, d^*, M^* of this estimate are chosen in a data-dependent way as described in [3]. For instance, we select these parameters out of the set $\{0, 1, 2\}$ for l, out of $\{1, \dots, 5\}$ for K, out of $\{1, \dots, d^*\}$ for d^*, and out of $\{1, \dots, 5, 6, 11, 16, 21, \dots, 46\}$ for M^*.

Furthermore, we consider a nearest neighbor estimate (abbr. *neighbor*). This means that the function value at a given point x is approximated by the average of the values $Y_1, \dots, Y_{k_n}$ observed for the data points $X_1, \dots, X_{k_n}$, which are closest to x with respect to the Euclidean norm (breaking the ties by indices). Here the parameter $k_n \in \mathbb{N}$ denoting

the involved neighbors is chosen adaptively from the set $\{1,2,3\}\cup\{4,8,12,16,\dots,4\cdot\lceil\frac{n_{train}}{4}\rceil\}$. Another competitive approach is the interpolation with radial basis function (abbr. *RBF*). Here we use Wendland's compactly supported radial basis function $\phi(r)=(1-r)_+^6\cdot(35r^2+18r+3)$, which can be found in [28]. The radius r that scales the basis functions is also selected adaptively from the set $\{0.1,0.5,1,5,30,60,100\}$. Additionally, we consider of course MARS. Here we used the ARESLab MATLAB toolbox provided by [29]. As a last competitive approach, we consider a regression tree (abbr. *tree*), which is implemented by MATLAB's function *fitrtree*. Here we choose the hyperparameters that minimize five-fold cross-validation loss by using automatic hyperparameter optimization.
The n observations (for $n\in\{100,200\}$) $(X,Y),(X_1,Y_1),(X_2,Y_2),\dots,(X_n,Y_n)$ are chosen as independent and identically distributed random vectors with X uniformly distributed on $[0,1]^{10}$ (in particular, the dimension of X is $d=10$) and Y generated by

$$Y=m_i(X)+\sigma_j\cdot\lambda_i\cdot\epsilon\ (i\in\{1,2,3\},j\in\{1,2\})$$

for $\sigma_j\geq 0$, $\lambda_i\geq 0$ and ϵ standard normally distributed and independent of X. The λ_i is chosen in way that respects the range covered by m_i on the distribution of X. Since our regression functions perform differently on different polytopes we determine the interquartile range of 10^5 realizations of $m_i(X)$ (additionally stabilized by taking the median of hundred repetitions of this procedure) not for the whole regression function, but on each set seperately and use the average of those values. For the regression functions below we got $\lambda_1=2.72$, $\lambda_2=6.28$, $\lambda_3=12.2$, $\lambda_4=13.97$ and $\lambda_5=0.04$. The parameters scaling the noise are chosen as $\sigma_1=5\%$ and $\sigma_2=20\%$.

The regression functions which were used to compare the different approaches are listed below.

$$\begin{aligned}
m_1(x)=&\left(\frac{10}{1+x_1^2}+5\cdot\sin(x_3\cdot x_4)+2\cdot x_5\right)\cdot 1_{H_1}(x)\\
&+\left(\exp(x_1)+x_2^2+\sin(x_3\cdot x_4)-3\right)\cdot 1_{\mathbb{R}^{10}\setminus H_1}(x),\\
m_2(x)=&\left(\cot\left(\frac{\pi}{1+\exp(x_1^2+2\cdot x_2+\sin(6\cdot x_4^3)-3)}\right)\right)\\
&\cdot 1_{H_1}(x)\\
&+\left(\cot\left(\frac{\pi}{1+\exp(x_1^2+2\cdot x_2+\sin(6\cdot x_4^3)-3)}\right)\right.\\
&+\exp\left(3\cdot x_3+2\cdot x_4-5\cdot x_1\right.\\
&\left.\left.+\sqrt{x_3+0.9\cdot x_4+0.1}\right)\right)\cdot 1_{\mathbb{R}^{10}\setminus H_1}(x),\\
m_3(x)=&\left(2\cdot\log(x_1\cdot x_2+4\cdot x_3+|\tan(x_4)|\right)\cdot 1_{H_2\cup H_3}(x)\\
&+\left(x_3^4\cdot x_5^2\cdot x_6-x_4\cdot x_7\right)\cdot 1_{H_2^C\cup H_3}(x),\\
&+\left(3\cdot x_8^2+x_9+2\right)^{0.1+4\cdot x_{10}^2}\cdot 1_{H_3^C}(x),\\
m_4(x)=&2\cdot\log(x_1\cdot x_2+4\cdot x_3+|\tan(x_4)|+0.1)\\
&+x_3^4\cdot x_5^2\cdot x_6-x_4\cdot x_7\\
&+\left(3\cdot x_8^2+x_9+2\right)^{0.1+4\cdot x_{10}^2}\\
m_5(x)=&\tanh\left(0.2x_1+0.9x_2+x_3+x_4+0.2x_5+0.6x_6\right)
\end{aligned}$$

with

$$\begin{aligned}
H_1=&\{x\in\mathbb{R}^{10}:0.1\cdot x_1+0.4\cdot x_2+0.3\cdot x_3\\
&+0.1\cdot x_4+0.2\cdot x_5+0.3\cdot x_6+0.6\cdot x_7\\
&+0.02\cdot x_8+0.7\cdot x_9+0.6\cdot x_{10}\leq 1.63\}\\
H_2=&\{x\in\mathbb{R}^{10}:0.1\cdot x_1+0.4\cdot x_2+0.3\cdot x_3\\
&+0.1\cdot x_4+0.2\cdot x_5+0.3\cdot x_6+0.6\cdot x_7\\
&+0.02\cdot x_8+0.7\cdot x_9+0.6\cdot x_{10}\leq 1.6\}\\
H_3=&\{x\in\mathbb{R}^{10}:4\cdot x_1+2\cdot x_2+x_3\\
&+4\cdot x_4+x_5+x_6\leq 7.5\}.
\end{aligned}$$

The quality of each of the estimates is determined by the empirical L_2-error, i.e. we calculate

$$\epsilon_{L_2,N}(m_{n,i})=\frac{1}{N}\sum_{k=1}^{N}\left(m_{n,i}(X_{n+k})-m_i(X_{n+k})\right)^2),$$

where $m_{n,i}$ $(i=1,\dots,4)$ is one of our estimates based on the n observations and m_i is our regression function. The input vectors $X_{n+1},X_{n+2},\dots,X_{n+N}$ are newly generated independent realizations of the random variable X, i.e. different from the n input vectors for the estimate. We choose $N=10^5$. We normalize our error by the error of the simplest estimate of m_i, i.e. the error of a constant function, calculated by the average of the observed data. Thus the errors given in our tables below are normalized error measures of the form $\epsilon_{L_2,N}(m_{n,i})/\bar{\epsilon}_{L_2,N}(avg)$. Here $\bar{\epsilon}_{L_2,N}(avg)$ is the median of 50 independent realizations you obtain if you plug the average of n observations into $\epsilon_{L_2,N}(\cdot)$. Since our simulation results depend on randomly chosen data points we repeat our estimation 50 times by using differently generated random realizations of X in each run. In Table 2, 3, 4, 5, 6 and 7 we listed the median (plus interquartile range IQR) of $\epsilon_{L_2,N}(m_{n,i})/\bar{\epsilon}_{L_2,N}(avg)$.

TABLE 2: Median of the normalized empirical L_2-error for each estimate and regression function m_1

m_1		
noise	5%	
sample size	$n=100$	$n=200$
$\bar{\epsilon}_{L_2,\bar{N}}(avg)$	29.5445	29.4330
neural-sc	**0.3809**(0.1902)	**0.1926**(0.1568)
neural-x	0.4412(0.2653)	0.2035(0.2178)
neural-fc	0.5040(0.3988)	0.2220(0.1568)
RBF	0.6856(0.1205)	0.6064(0.0670)
neighbor	0.6387(0.0785)	0.5610(0.0489)
MARS	0.6747(0.1433)	0.5091(0.0567)
tree	0.8469(0.1237)	0.8533(0.1248)

m_1		
noise	20%	
sample size	$n=100$	$n=200$
$\bar{\epsilon}_{L_2,\bar{N}}(avg)$	29.4970	29.4375
neural-sc	0.5113(0.3604)	0.2971(0.2546)
neural-x	**0.4674**(0.4427)	**0.2218**(0.3167)
neural-fc	0.4958(0.4742)	0.3016(0.1928)
RBF	0.7044(0.1150)	0.6173(0.0754)
neighbor	0.6411(0.0776)	0.5589(0.0500)
MARS	0.6949(0.1787)	0.5149(0.0519)
tree	0.7481(0.0940)	0.7653(0.1021)

TABLE 3: Median of the normalized empirical L_2-error for each estimate and regression function m_2

m_2		
noise	5%	
sample size	$n = 100$	$n = 200$
$\bar{\epsilon}_{L_2,\bar{N}}(avg)$	671.83	670.77
neural-sc	**0.8108**(0.6736)	**0.5468**(0.6812)
neural-x	0.8296(0.3139)	0.5543(0.3884)
neural-fc	1.0668(0.6779)	0.7792(0.4642)
RBF	1.0172(0.2613)	0.6896(0.3906)
neighbor	0.8640(0.1086)	0.7990(0.1476)
MARS	1.6299(1.5082)	3.4815(16.9055)
tree	0.9467(0.0950)	0.9016(0.1661)

m_2		
noise	20%	
sample size	$n = 100$	$n = 200$
$\bar{\epsilon}_{L_2,\bar{N}}(avg)$	669.82	672.04
neural-sc	**0.7453**(0.5348)	**0.5146**(0.4298)
neural-x	0.8788(0.5053)	0.5488(0.4127)
neural-fc	0.9678(0.4276)	0.8476(0.6150)
RBF	1.0179(0.2517)	0.6582(0.3297)
neighbor	0.8657(0.0884)	0.7469(0.1156)
MARS	1.6363(2.4886)	2.3530(10.0750)
tree	0.9483(0.0776)	0.9053(0.1489)

TABLE 4: Median of the normalized empirical L_2-error for each estimate and regression functions m_3

m_3		
noise	5%	
sample size	$n = 100$	$n = 200$
$\bar{\epsilon}_{L_2,\bar{N}}(avg)$	9023.9	9018.4
neural-sc	0.5983(0.6832)	**0.2006**(0.3523)
neural-x	**0.5168**(0.6809)	0.3156(0.2091)
neural-fc	0.7337(0.6276)	0.3657(0.4543)
RBF	0.6764(0.4601)	0.5527(0.3601)
neighbor	0.8188(0.1170)	0.7137(0.0985)
MARS	0.9925(1.7966)	0.6596(0.7020)
tree	0.7017(0.2901)	0.4624(0.1447)

m_3		
noise	20%	
sample size	$n = 100$	$n = 200$
$\bar{\epsilon}_{L_2,\bar{N}}(avg)$	9117.1	9017.4
neural-sc	**0.5521**(0.3977)	0.3223(0.3143)
neural-x	0.5555(0.6642)	**0.3147**(0.2386)
neural-fc	0.8311(0.4058)	0.3397(0.4208)
RBF	0.6580(0.4698)	0.5312(0.3780)
neighbor	0.8024(0.1117)	0.7191(0.0987)
MARS	1.1440(5.5270)	0.6445(0.7419)
tree	0.6873(0.2788)	0.4697(0.1501)

TABLE 5: Median of the normalized empirical L_2-error for each estimate and regression functions m_4 with noise 5%

m_4		
noise	5%	
sample size	$n = 100$	$n = 200$
$\bar{\epsilon}_{L_2,\bar{N}}(avg)$	5485.1	5468.7
neural-sc	0.3672(0.2972)	0.1571(0.1581)
neural-x	0.5667(0.8250)	0.2723(0.3540)
neural-fc	**0.1170**(0.5696)	**0.0394**(0.1405)
RBF	0.8612(0.3638)	0.8024(0.4667)
neighbor	0.8655(0.1006)	0.8105(0.1014)
MARS	1.4588(3.3845)	0.7384(4.6870)
tree	0.6215(0.2619)	0.3622(0.2567)

TABLE 6: Median of the normalized empirical L_2-error for each estimate and regression function m_4 with noise 20%

m_4		
noise	20%	
sample size	$n = 100$	$n = 200$
$\bar{\epsilon}_{L_2,\bar{N}}(avg)$	5485.3	5456.6
neural-sc	0.4295(0.3284)	0.2026(0.6585)
neural-x	0.5266(0.4642)	0.3663(0.5245)
neural-fc	**0.1551**(0.4228)	**0.0399**(0.0756)
RBF	0.9983(0.3678)	0.8046(0.4744)
neighbor	0.8648(0.0912)	0.8141(0.0831)
MARS	1.6776(17.3311)	0.7137(2.6324)
tree	0.6218(0.2553)	0.3461(0.2620)

TABLE 7: Median of the normalized empirical L_2-error for each estimate and regression function m_5

m_5		
noise	5%	
sample size	$n = 100$	$n = 200$
$\bar{\epsilon}_{L_2,\bar{N}}(avg)$	0.0049	
neural-sc	0.0037(0.0048)	0.0011(0.0012)
neural-x	10.0835(100.7)	7.0749(226.1946)
neural-fc	**0.0010**(0.0065)	**0.0006**(0.0014)
RBF	0.0204(0.0089)	0.0051(0.0023)
neighbor	0.3875(0.1197)	0.2679(0.0835)
MARS	0.1799(0.1099)	0.1107(0.0775)
tree	0.6215(0.2619)	0.3622(0.2567)

m_5		
noise	20%	
sample size	$n = 100$	$n = 200$
$\bar{\epsilon}_{L_2,\bar{N}}(avg)$	0.0049	0.0049
neural-sc	0.0967(0.1671)	0.0237(0.0278)
neural-x	0.7286(161.174)	0.1686(108.228)
neural-fc	**0.0098**(0.0906)	**0.0077**(0.0347)
RBF	0.0649(0.0284)	0.0361(0.0112)
neighbor	0.3814(0.1259)	0.2724(0.0922)
MARS	0.1993(0.1036)	0.1122(0.0577)
tree	0.6218(0.2553)	0.3461(0.2620)

We observe that our estimate outperforms the other approaches in 8 out of 12 examples in the three examples m_1, m_2 and m_3 of regression functions with low local dimensionality. Especially in cases m_1 and m_3, the error of our estimate is about half the error in each of the other approaches for $n = 200$ and $\sigma = 0.05$, except for the error of the other neural networks. We also observe, that the relative improvement of our estimate (and of the other networks) with an increasing sample size is much larger than the improvements for most of the other approaches (except for m_2 for the RBF and for m_3 for MARS). This could be a plausible indicator of a better rate of convergence.

It makes sense that we also get good approximations for the fully connected neural networks, since some of the sparse networks can be expressed by fully connected ones (e.g., choosing some weights as zero). The estimate *neural-x* of [3] was originally constructed to estimate regression functions with some composition assumption, for instance (p, C)-smooth generalized hierarchical interaction models. Since our regression functions represent a (p, C)-smooth generalized hierarchical interaction model on each polytope, it is plausible that this estimate also performs well for those regression functions. Nevertheless, with regard to our simulation results we see, that (with four exceptions) our sparse neural networks perform better than the other neural

network estimates.
With regards to the regression functions m_4 and m_5 the error of our sparse estimator is at least twice as large as the error of the fully connected networks. In these cases the regression functions do not fulfill Definition 3 of low local dimensionality. This indicates, that the architecture of our sparse networks was specially tailored to functions with low local dimensionality and shows no advantages for other function classes.

5 Real-world data experiment

The different approaches of the simulation study were further tested on a real–world data set to emphasize the practical relevance of our estimate. The data set under study was the earlier mentioned 2–year usage log of a bike sharing system named Captial Bike Sharing (CBS) at Washington, D.C., USA ([18]), where we conjecture some low local dimensionality in the data set, which fits our assumption on the regression function. The data set consists of 17379 data points, where each of them represents one hour of a day between 2011 and 2012; 500 were used for training and testing and the rest was used to compute the errors contained in Table 8. We used the same parameter sets as in the simulation study for all of our estimates and normalized the results again with the simplest estimate i.e. the average of the observed data . Table 8 summarizes the results. Again

TABLE 8: Normalized empirical L_2-risk for each estimate for the bike sharing data

neural-sc	*neural-x*	*neural-fc*	*RBF*
0.1680	0.3706	0.5924	0.8121

neighbor	*MARS*	*tree*
0.6829	0.3970	0.6522

we observe that our estimate outperforms the others i.e. the error of our estimate is about half the error of the second best approach (MARS). Hence our assumption of low local dimensionality seems plausible, at least for this real data set, since the estimate following this assumption outperforms all other estimates.

Acknowledgments

The authors would like to thank the Assciate Editor Dr. Alekh Agarwal and the two anonymous referees for their useful suggestions.

References

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw*, vol. 61, pp. 85–117, 2015.
[2] M. Kohler and A. Krzyżak, "Nonparametric regression based on hierarchical interaction models," *IEEE Trans. Inf. Theory*, vol. 63, no. 3, pp. 1620–1630, 2017.
[3] B. Bauer and M. Kohler, "On deep learning as a remedy for the curse of dimensionality in nonparametric regression," *Ann. Stat.*, vol. 47, no. 4, pp. 2261–2285, 2019.
[4] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with relu activation function," *Ann. Stat.*, vol. 48, no. 4, pp. 1875–1897, 2020.
[5] M. Kohler and S. Langer, "On the rate of convergence of fully connected deep neural network regression estimates," *Ann. Stat.*, vol. 49, no. 4, pp. 2231 – 2249, 2021.
[6] M. Imaizumi and K. Fukumizu, "Deep neural networks learn non-smooth functions effectively," in *AISTATS*, 2019.
[7] R. Nakada and M. Imaizumi, "Adaptive approximation and estimation of deep neural network to intrinsic dimensionality," *J. Mach. Learn. Res.*, vol. 21, no. 1-38, 2020.
[8] L. Györfi, M. Kohler, A. Krzyżak, and H. Walk, *A Distribution-Free Theory of Nonparametric Regression.*, ser. Springer Series in Statistics. Springer, 2002.
[9] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.
[10] L. P. Devroye and T. J. Wagner, "Distribution-free consistency results in nonparametric discrimination and regression function estimation," *Ann. Stat.*, vol. 8, no. 2, pp. 231–239, 1980.
[11] C. J. Stone, "Optimal global rates of convergence for nonparametric regression," *Ann. Stat.*, vol. 10, no. 4, pp. 1040–1053, 1982.
[12] R. Liu, B. Boukai, and Z. Shang, "Optimal nonparametric inference via deep neural network," *J. Math. Anal. Appl.*, vol. 505, no. 125561, 2022.
[13] K. Eckle and J. Schmidt-Hieber, "A comparison of deep networks with relu activation function and linear spline-type methods," *Neural Netw.*, vol. 110, pp. 232 – 242, 2019.
[14] J. Schmidt-Hieber, "Deep relu network approximation of functions on a manifold," *arXiv:1908.00695*, 2019.
[15] A. J. Bell and T. J. Sejnowski, "The "independent components" of natural scenes are edge filters," *Vision Res.*, vol. 37, no. 23, pp. 3327–3338, 1997.
[16] S. Schaal and S. Vijayakumar, "Local dimensionality reduction for locally weighted learning," in *CIRA*. IEEE Computer Society, 1997, pp. 220–225.
[17] H. Hoffmann, S. Schaal, and S. Vijayakumar, "Local dimensionality reduction for non-parametric regression," *Neural Process. Lett.*, vol. 29, no. 2, p. 109, 2009.
[18] H. Fanaee-T and J. Gama, "Event labeling combining ensemble detectors and background knowledge," *Prog. Artif.*, vol. 2, pp. 113–127, 2014.
[19] E. Levina and P. Bickel, "Maximum likelihood estimation of intrinsic dimension," in *Adv. Neural Inf. Process. Syst.*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2005.
[20] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.
[21] J. B. Tenenbaum, V. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[22] P. J. Bickel and B. Li, "Local polynomial regression on unknown manifolds," *Institute of Mathematical Statistics Lecture Notes - Monograph Series*, vol. 54, pp. 177–186, 2007.
[23] S. Kpotufe, "k-nn regression adapts to local intrinsic dimension," in *Adv. Neural Inf. Process. Syst.*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, vol. 24, pp. 729–737.
[24] S. Kpotufe and V. Garg, "Adaptivity to local smoothness and dimension in kernel regression," in *Adv. Neural Inf. Process. Syst.*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, vol. 26, pp. 3075–3083. [Online]. Available: h
[25] Y. Yang and A. Barron, "Information-theoretic determination of minimax rates of convergence," *Ann. Stat.*, vol. 27, no. 5, pp. 1564–1599, 1999.
[26] A. Choromanska, M. Henaff, M. Mathieu, G. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," *JMLR*, vol. 38, pp. 192–204, 2015.
[27] K. Kawaguchi, "Deep learning without poor local minima," in *Adv. Neural Inf. Process. Syst.*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, vol. 29, pp. 586–594.
[28] D. Lazzaro and L. B. Montefusco, "Radial basis functions for the multivariate interpolation of large scattered data sets," *J. Comput. Appl. Math.*, vol. 140, no. 1-2, pp. 521–536, 2002.
[29] G. Jekabsons. (2016) Areslab: Adaptive regression splines toolbox for matlab/octave. [Online]. Available: http://www.cs.rtu.lv/jekabsons/

Michael Kohler was born in Esslingen, Germany, on July 17, 1969. He received diploma degrees in computer science and mathematics from the University of Stuttgart in 1995, and a Ph.D degree in mathematics from the University of Stuttgart in 1997. In 1998 he worked as a Visiting Scientist at the Stanford University, Stanford, USA. From 2005 till 2007 he was a Professor of Applied Mathematics at the University of Saarbrücken, Germany, since 2007 he is a Professor of Mathematical Statistics at the Technische Universität Darmstadt, Germany. He coauthored with L. Györfi, A. Krzyżak and H. Walk the book *A Distribution-Free Theory of Nonparametric Regression* (New York:Springer, 2002). His main research interests are in the area of nonparametric statistics, especially curve estimation and deep learning.

Adam Krzyżak received the M.Sc. and Ph.D. degrees in computer engineering from the Wrocław University of Science and Technology, Poland, in 1977 and 1980, respectively, D.Sc. degree (habilitation) in computer engineering from the Warsaw University of Technology, Poland in 1998 and the Title of Professor from the President of Poland in 2003. Since 1983, he has been with the Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, where he is currently a Professor. In 1983, he held an International Scientific Exchange Award in the School of Computer Science, McGill University, Montreal, Canada, in 1991, the Vineberg Memorial Fellowship at the Technion-Israel Institute of Technology and, in 1992, Humboldt Research Fellowship at the University of Erlangen-NÃ¼rnberg, Germany. He visited the University of California Irvine, Information Systems Laboratory at Stanford University, Riken Frontiers Research Laboratory, Japan, Stuttgart University, Technical University of Berlin, University of Saarlandes and Technical University Darmstadt. His current research inter- ests include nonparametric statistics, deep learning theory and applications and classification. He has been an associate editor of *IEEE Transactions on Neural Networks* and *IEEE Transactions on Information Theory* and is presently an Associate Editor-in-Chief of Pattern Recognition Journal. He is a co-author of the book *A Distribution-Free Theory of Nonparametric Regression*, New York: Springer, 2002. He has been a General Co-chair of the IAPR Workshop on Statistical, Syntactic and Structural Pattern Recognition 2022, Co-chair of the Program Committee of the 10-th IEEE International Conference on Advanced Video and Signal-Based Surveillance 2013 and has served on program committees of numerous international conferences. He is a Fellow of the IEEE and a Fellow of the IAPR.

Sophie Langer received her B.Sc. , M.Sc. and Ph.D. degrees from the Technical University of Darmstadt, Germany in 2015, 2017 and 2020, respectively. From April 2020 to September 2021 she worked as a Post-Doctoral Research Scientist at Technical University of Darmstadt. Since then, she is working as a Post-Doctoral Research Scientist at Twente University, Netherlands. Her current research include the areas of deep learning theory, image classification and statistical learning theory.