Acta Numerica (2021), pp. 327–444 Printed in the United Kingdom
doi:10.1017/S0962492921000052

Neural network approximation

Ronald DeVore
Department of Mathematics, Texas A&M University,
College Station, TX 77843, USA
E-mail: rdevore@math.tamu.edu

Boris Hanin
Department of Operations Research and Financial Engineering,
Princeton University, NJ 08544, USA
E-mail: bhanin@princeton.edu

Guergana Petrova
Department of Mathematics, Texas A&M University,
College Station, TX 77843, USA
E-mail: gpetrova@math.tamu.edu

Neural networks (NNs) are the method of choice for building learning algorithms. They are now being investigated for other numerical tasks such as solving high-dimensional partial differential equations. Their popularity stems from their empirical success on several challenging learning problems (computer chess/Go, autonomous navigation, face recognition). However, most scholars agree that a convincing theoretical explanation for this success is still lacking. Since these applications revolve around approximating an unknown function from data observations, part of the answer must involve the ability of NNs to produce accurate approximations.

This article surveys the known approximation properties of the outputs of NNs with the aim of uncovering the properties that are not present in the more traditional methods of approximation used in numerical analysis, such as approximations using polynomials, wavelets, rational functions and splines. Comparisons are made with traditional approximation methods from the viewpoint of rate distortion, *i.e.* error versus the number of parameters used to create the approximant. Another major component in the analysis of numerical approximation is the computational time needed to construct the approximation, and this in turn is intimately connected with the stability of the approximation algorithm. So the stability of numerical approximation using NNs is a large part of the analysis put forward.

The survey, for the most part, is concerned with NNs using the popular ReLU activation function. In this case the outputs of the NNs are piecewise linear functions on rather complicated partitions of the domain of f into cells that are convex polytopes.

When the architecture of the NN is fixed and the parameters are allowed to vary, the set of output functions of the NN is a parametrized nonlinear manifold. It is shown that this manifold has certain space-filling properties leading to an increased ability to approximate (better rate distortion) but at the expense of numerical stability. The space filling creates the challenge to the numerical method of finding best or good parameter choices when trying to approximate.

CONTENTS

1. Introduction

Approximation using neural networks (NNs) is the method of choice for building numerical algorithms in machine learning (ML) and artificial intelligence (AI). It is now being looked at as a possible platform for computation in many other areas. Although NNs have been around for over 70 years, starting with the work of Hebb (1949) and Rosenblatt (1958), it is only recently that their popularity has surged as they have achieved state-of-the-art performance in a striking variety of machine learning domains. Examples of these are computer vision (Krizhevsky, Sutskever and Hinton 2012), employed for instance in self-driving cars, natural language processing (Wu *et al.* 2016), used in Google Translate, or reinforcement learning, such as superhuman performance at Go (Silver *et al.* 2016, 2017), to name a few.

Nevertheless, it is generally agreed upon that there is still a lack of solid mathematical analysis to explain the reasons behind these empirical successes. For a start, the understanding of the approximation properties of NNs is of vital importance since approximation is one of the main components of any algorithmic design. A rigorous analysis of what special properties NNs hold as a method of approximation could lead to both significant practical improvements (Bronstein *et al.* 2017, LeCun, Bengio and Hinton 2015) and *a priori* performance guarantees for computational algorithms based on NNs.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

At the heart of providing such a rigorous theory is understanding the benefits of using NNs as an approximation tool when compared with other more classical methods of approximation such as polynomials, wavelets, splines and sparse approximation from bases, frames and dictionaries. Indeed, most applications of NNs are built on some form of function approximation. This includes not only learning theory and statistical estimation, but also the new forays of NNs into other application domains such as numerical methods for solving partial differential equations (PDEs).

An often cited theoretical feature of neural networks is that they produce universal function approximants (Cybenko 1989, Hornik, Stinchcombe, White *et al.* 1989) in the sense that, given any continuous target function f and a target accuracy $\epsilon > 0$, neural networks with enough judiciously chosen parameters produce an approximation to f within an error of size ϵ. This universal approximation capacity has been known since the 1980s. But surely this cannot be the main reason why neural networks are so effective in practice. Indeed, all families of functions used in numerical approximation such as polynomials, splines, wavelets, *etc.*, produce universal approximants. What we need to understand is in what way NNs are more effective than other methods as an approximation tool.

The purpose of this article is to describe the approximation properties of NNs as we presently understand them, and to compare their performance with other methods of approximation. To accomplish such a comparative analysis, we introduce, starting in Section 5, the tools by which various methods of approximation are evaluated. These include approximation rates on model classes, n-widths, metric entropy and approximation classes. Since NN approximation is a form of nonlinear manifold approximation, we make this particular form of approximation the focal point of our exposition. The ensuing sections of the paper examine the specific approximation properties of NNs. After making some remarks that apply to general activation functions σ, we turn our attention to the performance of *rectified linear unit* (ReLU) networks. These are the most heavily used in numerical settings and fortunately also the NNs most amenable to analysis.

Since the output of a ReLU network is a continuous piecewise linear function (CPwL), it is important to understand what the class of outputs of a ReLU network depending on n parameters looks like in terms of their allowable partitions and the correlation between the linear pieces. This topic is addressed in Section 3. This structure increases in complexity with the depth of the network. It turns out that deeper NNs give a richer set of outputs than shallow networks. Therefore much of our analysis centres on deep ReLU networks.

The key takeaways from this paper are as follows. For a fixed value of n, the outputs of ReLU networks depending on n parameters form a rich parametric family of CPwL functions. This manifold exhibits certain space-filling properties (in the Banach space where we measure performance error), which are both a boon and a bottleneck. On one hand, space filling provides the possibility to approximate larger classes of functions with relatively few parameters, compared with the classes that

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

are currently approximated by classical methods. On the other hand, this flexibility comes at the expense of both the stability of the algorithm by which one selects the right parameters, and the *a priori* performance guarantees and uncertainty quantifications of performance when using NNs in numerical algorithms. This points to the need for a comprehensive study of the trade-offs between stability of numerical algorithms based on NNs and their numerical efficiency.

This exposition is far from providing a satisfactory theory for approximation by NNs, even when we restrict ourselves to ReLU networks. We highlight several fundamental questions that remain unanswered. Their solution would not only lead to a better understanding of NN approximation but would most likely guarantee better performance in numerical algorithms. These issues include:

- matching upper and lower bounds for the rate of approximation of standard model classes when using ReLU networks,
- how to precisely describe the types of function classes that benefit from NN approximation,
- how to numerically impose stability in parameter selection,
- how the imposition of stability limits the performance of the network.

2. What is a neural network?

This section begins by introducing feed-forward neural networks and their elementary properties. We begin with a general setting and then specialize to the case of fully connected networks. While the latter networks are generally not the architecture of choice in most targeted applications, their architecture provides the most convenient way to understand the trade-offs between approximation efficiency and the complexity of the network. They also allow for a clearer picture of the balance between width and depth in the assignment of parameters.

In its most general formulation, a *feed-forward neural network* $\mathcal{N}$ is associated with a directed acyclic graph (DAG),

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}),$$

called the *architecture* of $\mathcal{N}$, determined by a finite set $\mathcal{V}$ of vertices and a finite set of directed edges $\mathcal{E}$, in which every vertex $v \in \mathcal{V}$ must belong to at least one edge $e \in \mathcal{E}$. The set $\mathcal{V}$ consists of three distinguished subsets. The first is the set $\mathcal{I}$ of input vertices. These vertices have no incoming edges and are placeholders for independent variables (*i.e.* network inputs). The second is the set $\mathcal{O}$ of output vertices. These vertices have no outgoing edges and will store, for given inputs, the corresponding value of the dependent variables (*i.e.* the network output). The third is the set of hidden vertices $\mathcal{H} = \mathcal{V}\backslash\{\mathcal{I}, \mathcal{O}\}$. For a given input, hidden vertices store certain intermediate values used to compute the corresponding output. The vertices and edges also have the following adornments:

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

(1) With every $v \in \mathcal{V} \setminus \mathcal{I}$, there is an associated function $\sigma_v : \mathbb{R} \to \mathbb{R}$, called an *activation function*, and a scalar $b_v \in \mathbb{R}$, called a bias.

(2) For every $e \in \mathcal{E}$, there is a scalar $w_e \in \mathbb{R}$, called a *weight*.

In going forward, we often refer to the vertices as nodes. The weights and biases are referred to as the *trainable parameters* of $\mathcal{N}$. For a fixed network architecture, varying the values of these trainable parameters produces a family of output functions. The key to describing how these functions are constructed is that, to each vertex $v \in \mathcal{V} \setminus \mathcal{I}$, we associate a computational unit called a neuron. This unit takes as inputs the scalar outputs $x_{v'}$ from vertices $v' \in \mathcal{V} \setminus \mathcal{O}$ with an edge $e = (v', v) \in \mathcal{E}$ terminating at v, and outputs the scalar

$$x_v := \sigma_v \left(b_v + \sum_{e=(v',v)\in\mathcal{E}} w_e x_{v'} \right). \tag{2.1}$$

The word *neuron* comes from the fact that (2.1) can be viewed as a simple computational model for a single biological neuron. A neuron associated to a vertex $v \in \mathcal{V} \setminus \mathcal{I}$ observes signals $x_{v'}$ computed by upstream neurons associated to v', takes a superposition of these signals, mediated by synaptic weights w_e, $e = (v', v)$, and outputs x_v, which is then seen by the downstream neurons. For all neurons associated to vertices $v \in \mathcal{O}$, the activation function σ_v is the identity. The neuron associated to the ith input vertex $v \in \mathcal{I}$, $i = 1, \ldots, d$, where $d := |\mathcal{I}|$, observes a scalar incoming (*i.e.* externally provided) signal x_i and outputs x_i, which is then seen by the downstream neurons.

We view the network scalar inputs x_i, $i = 1, \ldots, d$, as an independent variable $x = (x_1, \ldots, x_d) \in \Omega \subset \mathbb{R}^d$, and define the output function $S_{\mathcal{N}} : \Omega \to \mathbb{R}^{d'}$ of the network $\mathcal{N}$ by

$$S_{\mathcal{N}}(x) := (x_v,\ v \in \mathcal{O}), \quad d' := |\mathcal{O}|. \tag{2.2}$$

Thus $S_{\mathcal{N}}$ is a function mapping $\Omega \subset \mathbb{R}^d$ into $\mathbb{R}^{d'}$, called the *output* of $\mathcal{N}$. Note that for a fixed network architecture $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the outputs $S_{\mathcal{N}}$ form a family of functions, determined by the trainable parameters $\{w_e, b_v\}$, $e \in \mathcal{E}$, $v \in \mathcal{V} \setminus \mathcal{I}$.

2.1. *Fully connected networks*

The preceding is a very general definition of neural networks and encompasses virtually all network architectures used in practice. In this article, however, we restrict our study to rather special examples of such networks, the so-called *fully connected networks*. The architecture of such a network is given by a directed acyclic graph whose vertices are organized into layers.

Each vertex of every layer is connected via outgoing edges to all vertices from the next layer and to no other vertices from any other layer; see Figure 2.1. The zeroth layer, called the *input layer*, consists of all $d := n_0$ input vertices $\mathcal{I}$, called inputs, where the ith input receives a scalar signal x_i from outside the network. The combined input $x := (x_1, \ldots, x_d)$ forms the independent variable of the function

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

$S_{\mathcal{N}}$. The input layer is followed by the hidden vertices $\mathcal{H}$, organized in L *hidden layers*, with the jth layer consisting of n_j hidden vertices, $j = 1, \dots, L$. The integer n_j is called the *width* of the jth layer. Finally, the $(L+1)$th layer, called the *output layer*, consists of all $d' := n_{L+1}$ output vertices $\mathcal{O}$, called outputs. The output vector of such a fully connected network is the value $S_{\mathcal{N}}(x) \in \mathbb{R}^{d'}$ of the function $S_{\mathcal{N}}$ for the input x.

As is customary, we specify that there is a single activation function σ that is used at each hidden vertex $v \in \mathcal{H}$, *i.e.* $\sigma_v = \sigma$ for all $v \in \mathcal{H}$. Recall that we always take the activation σ_v at the output vertices $v \in \mathcal{O}$ to be the identity. In this way, each coordinate of $S_{\mathcal{N}}(x)$ is a linear combination of the x_v's at layer L plus a bias term, which is a constant.

Thus, for a fully connected network $\mathcal{N}$, the output function $S_{\mathcal{N}}$ can be succinctly described by *weight matrices* and *bias vectors*

$$W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}, \quad b^{(\ell)} \in \mathbb{R}^{n_\ell}, \quad \ell = 1, \dots, L+1,$$

associated to layer ℓ as follows. If $X^{(\ell)} \in \mathbb{R}^{n_\ell}$ is the vector of outputs x_v corresponding to nodes v in layer ℓ, $\ell = 0, \dots, L+1$, then the output $S_{\mathcal{N}}$ is given by

$$S_{\mathcal{N}}(x) = X^{(L+1)} = W^{(L+1)} X^{(L)} + b^{(L+1)}, \tag{2.3}$$

where the vectors $X^{(\ell)}$ satisfy the recursion

$$X^{(\ell)} = \sigma(W^{(\ell)} X^{(\ell-1)} + b^{(\ell)}), \quad \ell = 1, \dots L, \quad X^{(0)} = x. \tag{2.4}$$

Here and throughout this paper, we use the convention that the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is defined to act on any vector $z = (z_1, \dots, z_{n_\ell}) \in \mathbb{R}^{n_\ell}$, $n_\ell \geq 1$ coordinate-wise, that is,

$$\sigma(z) = \sigma(z_1, \dots, z_{n_\ell}) := (\sigma(z_1), \dots, \sigma(z_{n_\ell})).$$

2.2. *The set* $\Upsilon^{W,L}(\sigma; d, d')$

We will almost always consider only fully connected feed-forward NNs whose hidden layer widths are all the same, namely $n_1 = \cdots = n_L = W$. Note that we can embed any fully connected feed-forward NN into a network with constant width $W := \max_{j=1,\dots,L} n_j$ by inserting $(W - n_j)$ additional zero bias vertices into layer j and adding new edges with weights set to 0 between these vertices and those in the next layer (if these vertices are in the first layer, we also add new edges with weights set to 0 between them and the input vertices). We use this fact frequently in what follows, sometimes without mentioning it.

We refer to W as the *width* of the network and to L as its *depth*. In such networks, each vertex v from a hidden layer can be associated with a pair of indices (i, j), where j is the layer index and i is the row index of the location of v. We commonly refer to all vertices from a fixed row as a *channel*, and to those from a fixed column as a *layer*. For every vertex v from the hidden layers, it is useful to introduce the

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

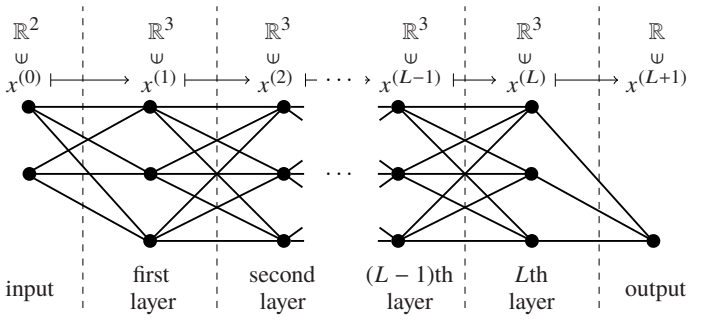

Figure 2.1. The graph associated to the outputs $\Upsilon^{3,L}(\sigma; 2, 1)$ of a fully connected network with input dimension 2, width 3, L hidden layers and output dimension 1.

function $z_v := z_{i,j}$, which records how the value at this neuron depends on the original input $x = (x_1, \ldots, x_d)$ before the activation σ is applied. It follows that

$$\sigma(z_v(x_1, \ldots, x_d)) := \sigma(z_{i,j}(x)) := X_i^{(j)}, \quad i = 1, \ldots, W, \; j = 1, \ldots, L, \tag{2.5}$$

which is the value of the ith coordinate of the vector $X^{(j)}$ defined in (2.4).

For a fully connected feed-forward network $\mathcal{N}$ with width W, depth L, activation function σ, input dimension d and output dimension d', we define the set

$$\Upsilon^{W,L} := \Upsilon^{W,L}(\sigma; d, d')$$

as the collection of all output functions $S_{\mathcal{N}}$ that we obtain by varying the choice of the trainable parameters of $\mathcal{N}$. Recall that $S_{\mathcal{N}}$ is a mapping from $\mathbb{R}^d$ (or $\Omega \subset \mathbb{R}^d$) to $\mathbb{R}^{d'}$. For notational simplicity, we often omit the dependence of $\Upsilon^{W,L}$ on σ, d and d' when these are understood from the context. Figure 2.1 shows the graph associated to a typical network that outputs functions from $\Upsilon^{W,L}(\sigma; d, d')$ with $d = 2$, $W = 3$, $d' = 1$.

Note that $\Upsilon^{W,L}$ is closed under addition of weights and biases *in the output layer*. This follows immediately from (2.3). However, it is *not closed* under addition of functions because we can find two outputs S_1, S_2 from $\Upsilon^{W,L}$ with $S_1 + S_2 \notin \Upsilon^{W,L}$. This will become apparent even in our discussion of one-layer ReLU networks; see Section 3. Therefore $\Upsilon^{W,L}$ is not a linear space. Each function $S_{\mathcal{N}} \in \Upsilon^{W,L}$ is determined by

$$n(W, L) := (d + 1)W + W(W + 1)(L - 1) + d'(W + 1) \tag{2.6}$$

parameters consisting of the entries of its weight matrices $W^{(1)}, \ldots, W^{(L+1)}$ and bias vectors $b^{(1)}, \ldots, b^{(L+1)}$. We note in passing that it is possible that distinct choices of trainable parameters end up describing the same outputs.

We order the parameters of $\mathcal{N}$ and organize them into a vector θ, where $\theta = \theta_{\mathcal{N}} \in \mathbb{R}^n$, $n = n(W, L)$. In the case $d' = 1$, the output $S_{\mathcal{N}}$ is then given by

$$S_{\mathcal{N}} = S(\cdot, \theta_{\mathcal{N}}) =: M(\theta_{\mathcal{N}}), \tag{2.7}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

where $M\colon \mathbb{R}^n \to \Upsilon^{W,L}(\sigma; d, 1)$ is the map from network weights and biases to output functions. In this way, we view $\Upsilon^{W,L}$ as a parametric manifold. Here we are using the term 'manifold' in a very loose sense because we are not attributing any of the topological or differential properties usually associated with this term.

The mapping M is completely determined once we have fixed the architecture and the activation function σ. Therefore, having made these choices, designing approximation methods for a target function f boils down to choosing parameters $\theta =: a(f)$ when f or information about f is given. In this way, any NN-based approximation method consists of determining a mapping $a\colon f \mapsto a(f)$ that assigns to each potential target function f a sequence of parameters $a(f) \in \mathbb{R}^n$. The approximation to f is then given by

$$A(f) := M(a(f)). \tag{2.8}$$

Our main focus in this paper is to understand the approximation power of NNs, and thus we work under the assumption that we have full access to the target function f. However, in the last two sections we do make forays into the more realistic (numerical) settings where we are only provided (partial) information about f in terms of data observations, or we are only allowed to query f to gain information. This separation between the approximation setting and the numerical setting is important since it may be that we could approximate f well if we had unlimited access to f, but in reality we are limited by the information provided to us.

Fully connected feed-forward NNs are an important approximation tool that is amenable to theoretical analysis. In practice, the most common choice of activation function σ is the so-called rectified linear unit

$$\sigma(t) = \mathrm{ReLU}(t) := t_+ := \max\{0, t\}.$$

This will constitute the main example of activation function studied in this article.

2.3. *Fundamental operations with neural networks*

In this section we discuss some fundamental operations that one can implement with NNs. Recall that $\Upsilon^{W,L} = \Upsilon^{W,L}(\sigma; d, d')$ is the set of functions that are outputs of an NN with the activation function σ, input dimension d, output dimension d' and L hidden layers, each of fixed width W.

Let us begin by pointing out that deep neural networks naturally allow for two fundamental operations – parallelization and concatenation – which we will often use.

Parallelization. If the NNs $\mathcal{N}_j$ have width W_j, depth L, input dimension d, output dimension d' and an activation function σ_j, $j = 1, \ldots, m$, then the parallelization of these networks is a new network $\mathrm{PAR}(\mathcal{N}_1, \ldots, \mathcal{N}_m)$ with width $W = W_1 + \cdots + W_m$, depth L, input dimension d and output dimension d'. Its graph is obtained by placing the hidden layers of $\mathcal{N}_j$ on top of each other. The parallelized network

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

can output any linear combination $S = \sum_{j=1}^m \alpha_j S_j$, where $S_j \in \Upsilon^{W_j,L}(\sigma_j; d, d')$, $j = 1, \dots, m$.

As described above, the network $\mathrm{PAR}(S_{\mathcal{N}_1}, \dots, S_{\mathcal{N}_m})$ does not have full connectivity since the nodes of $\mathcal{N}_j$ are not connected to the nodes of $\mathcal{N}_i$, $i \neq j$. However, we can view the resulting network as a fully connected network by completing it, namely by adding the missing edges and assigning to them zero weights.

To describe network concatenation, let us agree that given m functions $f_j : \mathbb{R}^{d_j} \to \mathbb{R}^{d_{j+1}}$, we will define their composition $f_m \circ \cdots \circ f_1 : \mathbb{R}^{d_1} \to \mathbb{R}^{d_{m+1}}$ by

$$(f_m \circ \cdots \circ f_1)(x) := f_m(f_{m-1}(\cdots f_1(x))). \tag{2.9}$$

If $f : \mathbb{R}^d \to \mathbb{R}^d$, we also introduce the notation

$$f^{\circ m} := f \circ f \circ \cdots \circ f, \tag{2.10}$$

where the composition is performed $m - 1$ times.

Concatenation. If the NNs $\mathcal{N}_j$ have width W_0, depth L_j, input dimension d_j, output dimension d_{j+1} and activation functions σ_j, $j = 1, \dots, m$, then the concatenation of these networks is a network $\mathrm{CONC}(\mathcal{N}_1, \dots, \mathcal{N}_m)$ with width W_0, depth $L = \sum_{j=1}^m L_j$, input dimension d_1 and output dimension d_{m+1}. Its graph is obtained by placing the hidden layers of these networks side by side with full connectivity between the hidden layers of $\mathcal{N}_j$ and $\mathcal{N}_{j+1}$. The concatenated NN can output any composition $S = S_m \circ S_{m-1} \circ \cdots \circ S_1$, where the functions $S_j \in \Upsilon^{W_0,L_j}(\sigma_j; d_j, d_{j+1})$, $j = 1, \dots, m$. It does this by assigning weights and biases, associated to edges connecting the last hidden layer of an $\mathcal{N}_j$ to a node of the first hidden layer of the neighbour $\mathcal{N}_{j+1}$, using the output weights and biases of $\mathcal{N}_j$ and input weights and biases of $\mathcal{N}_{j+1}$.

Parallelization and concatenation of neural networks allow us to perform the following operations between their outputs.

Addition by increasing width. It follows from **Parallelization** that for any $L \geq 1$ and $S_j \in \Upsilon^{W_j,L}(\sigma; d, d')$, $j = 1, \dots, m$, the linear combination satisfies

$$\sum_{j=1}^m \alpha_j S_j \in \Upsilon^{W,L}(\sigma; d, d'), \quad W := W_1 + \cdots + W_m.$$

Composition. It follows from **Concatenation** that for any $W \geq 1$ and $S \in \Upsilon^{W,L_1}(\sigma; m, d')$, $T \in \Upsilon^{W,L_2}(\sigma; d, m)$, the composition satisfies

$$S \circ T \in \Upsilon^{W,L}(\sigma; d, d'), \quad L := L_1 + L_2.$$

Remark 2.1. In particular, it follows from **Parallelization** and **Concatenation** that given the outputs $T_j \in \Upsilon^{W_j,L_1}(\sigma; d, 1)$, $j = 1, \dots, m$, and $S \in \Upsilon^{W,L_2}(\sigma; m, d')$,

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

with $W = \sum_{j=1}^{m} W_j$, then the function satisfies

$$S(T_1, \dots, T_m) \in \Upsilon^{W,L}(\sigma; d, d'), \quad L := L_1 + L_2.$$

Shifted dilates. If $S \in \Upsilon^{W,L}(\sigma; d, d')$, then for any $a \in \mathbb{R}$ and $c \in \mathbb{R}^d$, the shifted dilate satisfies

$$T(x) := S(ax + c) \in \Upsilon^{W,L}(\sigma; d, d').$$

To prove this, let $\mathcal{N}$ be the NN which outputs the function S. To output the function T, it is enough to alter the weights and biases of the first hidden layer of $\mathcal{N}$. Namely, if a neuron from this layer computes $\sigma(w \cdot x + b)$, $w \in \mathbb{R}^d$, $b \in \mathbb{R}$, we replace it with $\sigma(aw \cdot x + w \cdot c + b)$. Here $x \cdot x'$ denotes the inner product of two vectors x, x' of the same dimension. The remaining layers stay the same.

2.4. *One-layer neural networks*

The function $S_{\mathcal{N}} \in \Upsilon^{W,1}(\sigma; d, 1)$ produced by a single-hidden-layer fully connected feed-forward neural network $\mathcal{N}$ with activation function σ, d inputs and one output has the representation

$$S_{\mathcal{N}}(x) = b_0 + \sum_{j=1}^{W} a_j \sigma(w_j \cdot x + b_j), \quad a_j, b_j \in \mathbb{R},\ w_j \in \mathbb{R}^d, \tag{2.11}$$

where W is the width of the first (and only) hidden layer and b_0 is the bias of the output node. The above can equivalently be written as

$$S_{\mathcal{N}}(x) = b_0 + \int_{\mathbb{R}^{d+1}} \sigma(w \cdot x + b)\, \mathrm{d}\mu_{\mathcal{N}}(w, b), \quad \mu_{\mathcal{N}}(w, b) := \sum_{j=1}^{W} a_j \delta_{(w_j, b_j)},$$

where δ_z denotes the mass-one atomic measure at the point z. The correspondence between purely atomic Borel measures and $\Upsilon^{W,1}(\sigma; d, 1)$ is useful in addressing various structural properties of this set via functional analytic arguments. As an example of this, we briefly discuss the density question of whether, for each continuous function f, defined on a compact set $\Omega \subset \mathbb{R}^d$, we have

$$\operatorname{dist}(f, \Upsilon^{W,1}(\sigma; d, 1))_{C(\Omega)} \to 0, \quad W \to \infty, \tag{2.12}$$

where for the current discussion the distance is measured in the uniform norm $\|f\|_{C(\Omega)} := \sup_{x \in \Omega} |f(x)|$. This question is discussed in detail in Pinkus (1999); see also Cybenko (1989) and Petersen (2020). Here we only point out some key results.

Note that (2.12) does not hold for every activation function σ. For example, if $\sigma = P$ is a univariate polynomial of degree m, then $\sigma(w \cdot x + b)$ with $w \in \mathbb{R}^d$, $b \in \mathbb{R}$ is a multivariate polynomial in $x = (x_1, \dots, x_d)$ of total degree m and hence $\Upsilon^{W,1}(\sigma; d, 1) \subset X$, where X is a linear space of fixed finite dimension. Thus (2.12) does not hold.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Sufficient condition. If σ is a continuous function on $\mathbb{R}$ such that for each finite, signed regular Borel measure $\mu \neq 0$ on Ω, the function

$$F_\mu(w, b) := \int_\Omega \sigma(w \cdot x + b)\, \mathrm{d}\mu(x), \quad w \in \mathbb{R}^d,\ b \in \mathbb{R},$$

is not identically zero, then the density condition (2.12) holds. This condition can be used to prove the following examples of activation functions for which (2.12) holds:

- *Sigmoidal activation function.* A function σ, defined and continuous on $\mathbb{R}$, is called a sigmoidal function if

$$\lim_{t\to\infty} \sigma(t) = 1 \quad \text{and} \quad \lim_{t\to-\infty} \sigma(t) = 0.$$

  For each such σ the density statement (2.12) holds; see Cybenko (1989) for one of the first proofs in this case.
- *ReLU activation function.* If $\sigma(t) = t_+$, $t \in \mathbb{R}$, then the density condition (2.12) holds.

3. ReLU networks

In this section we summarize what is known about the outputs of NNs with ReLU activation (ReLU networks). We begin by making general remarks that hold for any ReLU network and then turn to special cases, especially those that form our main interest of study in this paper.

Perhaps the most important structural property of ReLU networks is that any output of such a network is a continuous piecewise linear function. To describe this precisely, we start with the following definitions.

Definition 3.1. A *polytope partition* of $\mathbb{R}^d$ is a finite collection $\mathcal{P} = \{P_j\}$ of convex closed d-dimensional polytopes (not necessarily bounded) which are exhaustive and have disjoint interiors P_j^o, that is,

$$\bigcup_j P_j = \mathbb{R}^d, \quad P_j^\circ \cap P_k^\circ = \emptyset \quad \text{for all } j \neq k.$$

Each such convex polytope is the intersection of a finite number of closed half-spaces. We refer to the polytopes P_j of such a partition as *cells*.

Definition 3.2. A function $S\colon \mathbb{R}^d \to \mathbb{R}$ is a continuous piecewise linear function (CPwL) if S is globally continuous and there is a polytope partition $\mathcal{P} = \{P_j\}$ on which S is locally affine, that is,

$$S|_{P_j} \quad \text{is affine} \quad \text{for all } j.$$

We then say that S is subordinate to the polytope partition $\mathcal{P}$.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

We let

$$\Sigma_{n,d} := \Sigma_{n,d}(\mathrm{CPwL})$$

denote the collection of all CPwL functions $S\colon \mathbb{R}^d \to \mathbb{R}$ that are subordinate to some polytope partition with at most n cells. This collection is a nonlinear set. For example, if S_1 and S_2 are subordinate to different partitions of size n then the sum $S_1 + S_2$ is typically not in $\Sigma_{n,d}$. Going forward in this paper, we do not study $\Sigma_{n,d}$ but only use it for comparison purposes. Note that if a CPwL function S is subordinate to $\mathcal{P}$ then it is also subordinate to any refinement of $\mathcal{P}$.

In the special case when $d = 1$, polytope partitions of $\mathbb{R}$ are simply decompositions of $\mathbb{R}$ into intervals with disjoint interiors, and thus $\Sigma_{n,1}$ is in fact the set of all univariate continuous linear free-knot splines with at most $n - 1$ breakpoints.

Theorem 3.3. Let $\mathcal{N}$ be a ReLU network with d inputs, one output node and m hidden neurons. Then the output $S_{\mathcal{N}}$ of $\mathcal{N}$ is a CPwL function subordinate to a partition $\mathcal{P}_{\mathcal{N}}$ with at most 3^m cells, *i.e.* $\#\mathcal{P}_{\mathcal{N}} \le 3^m$.

Proof. Let $z_1(x), \dots, z_m(x)$ denote the pre-activations of the network's neurons, *i.e.* the values stored at the neurons for input x before ReLU is applied. For every *activation pattern*

$$\nu = (\nu_1, \dots, \nu_m) \in \{-1, 0, 1\}^m,$$

functional (hyper-plane) is the dual formulation?

we define

$$\Omega_\nu := \{x \in \mathbb{R}^d \colon \operatorname{sign}(z_j(x)) = \nu_j,\ j = 1, \dots, m\}, \tag{3.1}$$

where for the purpose of this formula $\operatorname{sign}(0) := 0$. By construction, each Ω_ν is the (possibly empty) collection of all inputs $x \in \mathbb{R}^d$ at which the network neurons have a given pattern of being on, off or zero, prescribed by ν. A simple inductive argument (see Hanin and Rolnick 2019, Lemma 7) shows that Ω_ν is a convex polytope. Moreover, defining P_ν to be the closure of Ω_ν, we see that the collection

$$\mathcal{P}_{\mathcal{N}} := \{P_\nu \mid P_\nu^o \neq \emptyset\}$$

is a polytope partition of $\mathbb{R}^d$ and that $S_{\mathcal{N}}$ is a CPwL function subordinate to this partition. □

Having established that each output of a ReLU network is a CPwL function, it is of interest to give bounds for the number of cells in such a partition. The above theorem gives a bound 3^m. However, many of the cells Ω_ν, defined in (3.1), are either empty or have dimension smaller than d. We shall see as we proceed in this section that this bound can be improved in the cases of interest to us. At this stage let us just mention the following almost trivial result.

Claim. Consider a fixed architecture for neural networks with m neurons as above. Let $S(\cdot; \theta)$ be the outputs of a ReLU network with parameters $\theta \in \mathbb{R}^n$. Then outside a set of measure zero in $\mathbb{R}^n$, any selection of parameters results in an $S(\cdot, \theta)$ which is subordinate to a partition with at most 2^m cells.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

This claim is proved by showing that outside a set of measure zero in parameter space $\mathbb{R}^n$, all cells Ω_ν defined in (3.1) are empty or have dimension $< d$ whenever one of the components ν_j of ν is zero.

Our purpose in the remainder of this section is to explore the properties of both the polytope partitions created by ReLU networks and the complexity of the CPwL functions that they output. We start in Section 3.1 by studying in detail ReLU networks with input and output dimension 1, postponing a discussion of higher input dimensions to Section 3.2.

3.1. Univariate ReLU networks

In this section we consider ReLU networks with input and output dimensions both equal to one, *i.e.* $d = d' = 1$. In this case the polytope partitions of $\mathbb{R}$ are simply decompositions of $\mathbb{R}$ into a finite collection of intervals with disjoint interiors, and the CPwL functions subordinate to such partitions are customarily referred to as continuous linear free-knot splines.

3.1.1. Single-layer univariate ReLU networks

For the set $\Upsilon^{W,1} := \Upsilon^{W,1}(\mathrm{ReLU}; 1, 1)$, we have the simple inclusion (see Daubechies *et al.* 2019)

$$\Sigma_{W,1} \subsetneq \Upsilon^{W,1} \subsetneq \Sigma_{W+1,1}, \tag{3.2}$$

where we recall our notation $\Sigma_{W,1} := \Sigma_{W,1}(\mathrm{CPwL})$ for the set of CPwL functions subordinate to a partition of $\mathbb{R}$ into W intervals. This shows that $\Upsilon^{W,1}$ and the set of linear free-knot splines, determined by comparable number of parameters, essentially have the same approximation power. Recall that, according to (2.6), $\Upsilon^{W,1}$ is determined by $3W + 1$ parameters, while $\Sigma_{W,1}$ is determined by $2W$ parameters.

We point out a particularly important family of functions generated by ReLU networks, namely the hat functions $H_{\mathbf{p}}$, where $\mathbf{p} = (p_1, p_2, p_3) \in \mathbb{R}^3$, $p_1 < p_2 < p_3$, defined as

$$H_{\mathbf{p}}(t) = \begin{cases} 0 & t \notin [p_1, p_3], \\ \dfrac{t - p_1}{p_2 - p_1} & t \in [p_1, p_2], \\ -\dfrac{t - p_3}{p_3 - p_2} & t \in [p_2, p_3]. \end{cases} \tag{3.3}$$

Here $H_{\mathbf{p}}$ is a CPwL function that takes the value one at p_2, zero at p_1 and p_3, is linear on $[p_1, p_2]$ and $[p_2, p_3]$, and vanishes outside $[p_1, p_3]$. Note that since $H_{\mathbf{p}} \equiv 0$ outside $[p_1, p_3]$, we have

$$H_{\mathbf{p}}(t) = \frac{1}{p_2 - p_1}(t - p_1)_+ - \frac{p_3 - p_1}{(p_3 - p_2)(p_2 - p_1)}(t - p_2)_+ + \frac{1}{p_3 - p_2}(t - p_3)_+,$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

and hence $H_{\mathbf{p}} \in \Upsilon^{3,1}(\mathrm{ReLU}; 1, 1)$. In particular, the hat function H, defined as $H := H_{(0,1/2,1)}$ and viewed as a function on $[0, 1]$, has the representation

$$H(t) = 2(t - 0)_+ - 4\left(t - \frac{1}{2}\right)_+. \tag{3.4}$$

Thus $H \in \Upsilon^{2,1}(\mathrm{ReLU}; 1, 1)$ when considered only on $[0, 1]$.

3.1.2. Deep univariate ReLU networks

According to the discussion at the start of Section 3, any function from the set $\Upsilon^{W,L} := \Upsilon^{W,L}(\mathrm{ReLU}; 1, 1)$ is a CPwL function on $\mathbb{R}$. It is of interest to understand exactly which CPwL functions are in this set. We shall see that such a characterization is rather straightforward when $L = 1$, but the situation gets more complicated as L gets larger.

When $L = 1$, any selection of weights and biases produces as output a CPwL function S with at most W breakpoints. Indeed, S can be expressed as $S = b_0 + \sum_{j=1}^{W} a_j \eta_j(t)$, where the functions $\eta_j(t) = (\pm t + b_j)_+$. Obviously the bound W cannot be improved. Although $\Upsilon^{W,1}$ does not contain all of $\Sigma_{W+1,1}$, it does contain all of $\Sigma_{W,1}$; see (3.2).

When $L > 1$, the situation gets much more complicated. Even though there is no precise characterization of the set of outputs, we can provide some important insight. When L grows, two important things happen:

(i) the number of breakpoints of functions from $\Upsilon^{W,L}$ can be exponential in L,

(ii) not every CPwL function with this large number of breakpoints is in $\Upsilon^{W,L}$, in fact, far from it.

We first address (i). Fix W and let $S \in \Upsilon^{W,L} = \Upsilon^{W,L}(\mathrm{ReLU}; 1, 1)$. We define $m(L)$ as the largest number of breakpoints that any $S \in \Upsilon^{W,L}$ can have. We know that $m(1) = W$. Moreover, once the parameters are chosen for the first layer, any output S has breakpoints in a fixed set Λ of cardinality at most W. We can bound $m(L + 1)$ in terms of $m(L)$ as follows. Each $S \in \Upsilon^{W,L+1}$ can be expressed as

$$S = \sum_{k=1}^{W} a_k [S_k]_+ + b, \quad a_k, b \in \mathbb{R}, \quad S_k \in \Upsilon^{W,L}, \quad k = 1, \ldots, W. \tag{3.5}$$

There is a set Λ, $\#(\Lambda) \le m(L)$ such that each of the S_k have their breakpoints in Λ. Fix k and consider the function $[S_k]_+$. It has two types of breakpoints: those inherited from Λ, and the set Λ'_k of new breakpoints that arose after the application of ReLU. We have $\#(\Lambda'_k) \le \#(\Lambda) + 1 \le m(L) + 1$, $k = 1, \ldots, W$. Hence S has at most $m(L) + W(m(L) + 1)$ breakpoints. It follows that

$$m(L + 1) \le (W + 1)m(L) + W, \quad L \ge 1. \tag{3.6}$$

This recursion with the starting value $m(1) = W$ gives the bound

$$m(L) \le (W + 1)^L, \quad L = 1, 2, \ldots. \tag{3.7}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

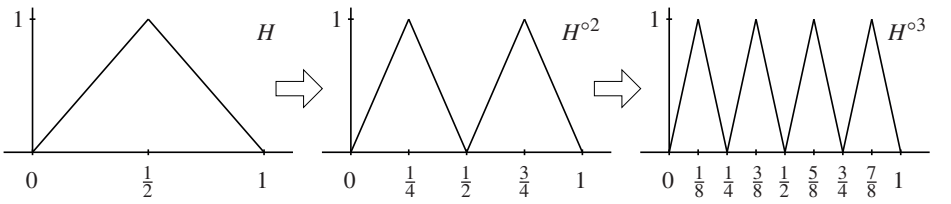

Figure 3.1. The sawtooth functions $H^{\circ L}$.

This bound can be improved somewhat at the expense of a more involved argument.

This potential exponential growth of the breakpoints as a function of the number of neurons can in fact be attained. A simple example, first noted by Telgarsky (2016), is to compose the hat function $H_{(0,1/2,1)}$ on $[0,1]$ (see (3.4)) with itself $(L-1)$ times. The resulting function $S := H^{\circ L}$ is the sawtooth function with 2^{L-1} teeth; see Figure 3.1. Since $H \in \Upsilon^{2,1}(\mathrm{ReLU}; 1, 1)$, it follows from **Composition** that $S \in \Upsilon^{2,L}(\mathrm{ReLU}; 1, 1)$.

While a function in $\Upsilon^{W,L}(\mathrm{ReLU}; 1, 1)$ can have an exponential (in L) number N_L of breakpoints, one should not get disillusioned into thinking that this set of functions is anywhere close to $\Sigma_{N_L+1,1}$, which was the point in (ii). The reason for this is that there are linear dependencies between the linear pieces in the case of a large number of breakpoints. There is not yet a good understanding of exactly which CPwL functions are in $\Upsilon^{W,L}(\mathrm{ReLU}; 1, 1)$ when L is large. A possible starting point to unravel this is to consider special cases such as breakpoints in $[0,1]$ at the dyadic integers $j2^{-n}$, $j = 0, 1, \ldots, 2^n$, with $n > 1$.

Problem 3.4. For $L > n$, characterize the CPwL functions in $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ which have breakpoints only at the dyadic integers $j2^{-n}$, $j = 0, \ldots, 2^n$.

3.2. Multivariate ReLU networks

We now turn to studying the properties of ReLU networks with input dimension $d > 1$, starting with those networks that have one hidden layer. Deeper multivariate ReLU networks are discussed in Section 3.2.2.

3.2.1. Multivariate ReLU networks with one hidden layer

A ReLU network $\mathcal{N}$ with input dimension $d > 1$, output dimension 1 and one hidden layer of width W outputs a function of the form

$$S_{\mathcal{N}}(x) = b_0 + \sum_{j=1}^{W} a_j \eta_j(x), \quad \eta_j(x) := (z_j(x))_+, \quad x \in \mathbb{R}^d, \tag{3.8}$$

where $b_0, a_j \in \mathbb{R}$, $j = 1, \ldots, W$, and z_j, $j = 1, \ldots, W$, is the function computed by the jth neuron before applying ReLU,

$$z_j(x) = z_j(x; w_j, b_j) = w_j \cdot x + b_j, \quad w_j \in \mathbb{R}^d,\ b_j \in \mathbb{R}.$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

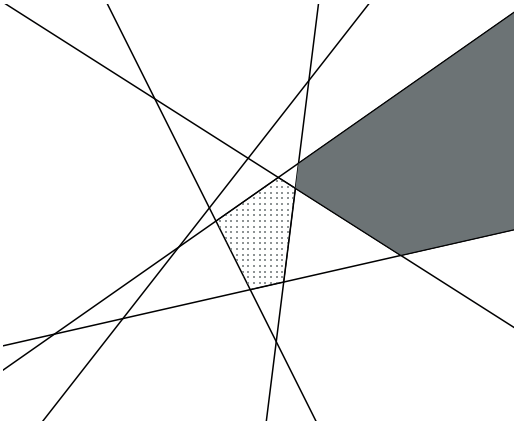

Figure 3.2. A hyperplane arrangement in $\mathbb{R}^2$ with two of its cells shaded.

Let us record the following useful fact.

Observation 3.5. Any function $S \in \Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$ has a representation (3.8) with the w_j, $j = 1, \dots, W$, unit norm vectors.

This follows by removing the zero weight vectors and normalizing the remaining w_j's by adjusting the constants b_j and a_j. We use this representation of functions in $\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$ in going forward. Given the collection of W unit vectors $w_j \in \mathbb{R}^d$ and biases $b_j \in \mathbb{R}$ from (3.8), we define the hyperplanes

$$H_j := \{x \in \mathbb{R}^d : w_j \cdot x + b_j = 0\}, \quad j = 1, \dots, W,$$

and the collection $\mathcal{H} := \{H_1, \dots, H_W\}$, associated to the network $\mathcal{N}$. This collection is an example of a *hyperplane arrangement*, a classical subject in combinatorics (Stanley *et al.* 2004); see Figure 3.2, for example.

We now describe how the hyperplane arrangement $\mathcal{H}$ associated to $\mathcal{N}$ determines the polytope partition of $\mathbb{R}^d$ to which $S_{\mathcal{N}}$ is subordinate in the sense of Definition 3.2. Because of our assumption that each w_j has unit norm, only activation patterns with entries $\nu_j \in \{\pm 1\}$ lead to cells with non-empty interiors. For any such activation pattern $\nu = (\nu_1, \dots, \nu_W)$, we may write, in the notation of (3.1),

$$\Omega_\nu = \bigcap_{j=1}^{W} H_j^{\nu_j}, \quad H_j^{\nu_j} = \{x \in \mathbb{R}^d : \mathrm{sign}(z_j(x)) = \nu_j\},$$

as an intersection of half-spaces. Thus, in the special case of $\Upsilon^{W,1}$, the cells in partitions $\mathcal{P}$ of the output functions have a simple global description. The partition $\mathcal{P}$ associated to $\mathcal{H}$ is the collection of the closures $P_\nu := \overline{\Omega}_\nu$ for which $\Omega_\nu \neq \emptyset$. Each cell P_ν is a convex closed polytope. By a special case of a classical result of

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Zaslavsky (1975), the number of cells $\#\mathcal{P}$ in $\mathcal{P}$ satisfies

$$\#\mathcal{P} \leq \sum_{j=0}^{d} \binom{W}{j}. \tag{3.9}$$

In fact, Zaslavsky's theorem shows that away from a codimension 1 set of weights and biases (*i.e.* when the hyperplanes are in general position), this upper bound is attained.

In summary, any function $S \in \Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$ is a CPwL function subordinate to a partition $\mathcal{P}$, generated by an arrangement of W hyperplanes determined by the weights and biases of the network $\mathcal{N}$. However, it is important to note that, unlike the case of one hidden layer with input dimension 1, not every CPwL function subordinate to a polytope partition arising from a hyperplane arrangement is the output of a one-layer ReLU network. There are several ways to see this, as we now discuss.

First of all, let us show that $\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$ does not contain any non-zero compactly supported functions on $\mathbb{R}^d$ once $d > 1$. To see this, consider a function S of the form (3.8) and suppose that S has compact support on $\mathbb{R}^d$. For each $j = 1, \ldots, W$, there is a ball $B_j \subset \mathbb{R}^d$ outside the support of S that intersects the hyperplane H_j but none of the other hyperplanes H_i, $i \neq j$. We have

$$0 = S(x) = L(x) + a_j \eta_j(x), \quad x \in B_j,$$

where L is an affine function. Since L and η_j are linearly independent on B_j, this implies $a_j = 0$. Hence all a_j's are zero and S is a constant. Since S was assumed to have compact support, this constant is zero.

Another way to see that $\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$ does not contain all CPwL functions subordinate to a given hyperplane arrangement is the following. Once $\mathcal{H} = \{H_1, \ldots, H_W\}$ is chosen, thereby determining the partition $\mathcal{P}$, the outputs of $\Upsilon^{W,1}$ that are subordinate to $\mathcal{P}$ are all contained in a linear space of dimension $2W + 1$. This follows from the representation (3.8). Indeed, since we have assumed that $\|w_j\| = 1$ for each j, there are only two choices of (w_j, b_j) for the functions $z_j(x) = w_j \cdot x + b_j$, $j = 1, \ldots, W$, such that

$$H_j = \{x \in \mathbb{R}^d \colon z_j(x) = 0\}.$$

However, when $d > 1$, Zaslavsky's theorem shows that the number of cells in $\mathcal{P}$ can grow as fast as CW^d when $W \geq d$. Hence, in general, the set of all CPwL functions subordinate to $\mathcal{P}$ is a linear space with dimension much larger than $2W + 1$.

The following lemma gives a simple way of checking when a CPwL function subordinate to a partition generated by an arrangement of W hyperplanes is in $\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$. Before formulating the lemma, let us note that if T is a CPwL function subordinate to $\mathcal{P}$, then on any cell $\mathcal{P}_\nu$ of $\mathcal{P}$, the gradient ∇T is a constant vector. It follows that ∇T is a piecewise constant vector-valued function subordinate to $\mathcal{P}$.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Lemma 3.6. Let $\mathcal{P}$ be a partition of $\mathbb{R}^d$ generated by a hyperplane arrangement $\mathcal{H} = \{H_1, \ldots, H_W\}$, where $H_j := \{x \in \mathbb{R}^d : w_j \cdot x + b_j = 0\}$, and w_j is a unit vector, $j = 1, \ldots, W$. Let T be a CPwL function that is subordinate to $\mathcal{P}$. Then T has the representation

$$T = S + L, \quad S \in \Upsilon^{W,1}(\mathrm{ReLU}; d, 1), \quad L\text{-globally affine},$$

if and only if the following condition holds:

(A) For each $j = 1, \ldots, W$, there is a real number a_j such that for every $x \in \mathbb{R}^d$ on the hyperplane H_j, and on no other hyperplane, the jump in ∇T across H_j at x, equals $a_j w_j$.

Proof. First, let $T = S + L$ with S and L as above. We know that the function $S \in \Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$ has the representation (3.8) with the w_j's being unit vectors. Given $x \in \mathbb{R}^d$ that belongs to H_j and to no other hyperplane, the jump in ∇T at x is the same as that of $a_j \nabla \eta_j$ at x, which is $a_j w_j$. This shows that T satisfies (A).

For the converse, suppose that T is any CPwL that is subordinate to $\mathcal{P}$ and that T satisfies condition (A). We define $S := \sum_{j=1}^{W} a_j \eta_j \in \Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$, where the a_j's are given by (A) and $\eta_j(x) := (w_j \cdot x + b_j)_+$. Consider the function $(T - S)$ which is piecewise linear subordinate to the partition $\mathcal{P}$. We claim that this function is a globally affine function. Indeed, otherwise there would be two adjacent cells which share a $(d-1)$-dimensional boundary (which is part of some H_j) and the jump of $\nabla(T - S)$ across this boundary is not zero. But both T and S have the same jump $a_j w_j$ of their gradient across this boundary. This is a contradiction and proves the lemma. □

3.2.2. Deep multivariate ReLU networks

The discussion at the beginning of Section 3 showed that any $S \in \Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ is a CPwL function subordinate to a partition $\mathcal{P}$ of $\mathbb{R}^d$ into convex polytopes. The partition we produced to show this was not determined by a hyperplane arrangement. It turns out, as we shall see in Section 3.3.3, that S is always subordinate to some partition given by a hyperplane arrangement. However, the latter partition is not a minimal partition to which S is subordinate. In other words, unlike the case of $L = 1$, the minimal polytope partition of $\mathbb{R}^d$ to which S is subordinate is not simply given by the cells of a hyperplane arrangement.

Given $S \in \Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$, we do not know the best bound for the number of cells in a minimal convex polytope partition to which S is subordinate, but we can give some bounds. Recall that from (3.1) we have the bound 3^{WL}. We also stated that in the generic case this bound can be improved to 2^{WL}. Indeed, in the generic case this partition consists of the closures of those convex sets

$$\Omega_\nu = \{x \in \mathbb{R}^d : \mathrm{sign}(z_j(x)) = \nu_j\}, \quad \nu = (\nu_1, \ldots, \nu_{WL}) \in \{\pm 1\}^{WL},$$

which have a non-empty interior. We continue to write $z_j(x)$ for the CPwL function

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

computed by the jth neuron in $\mathcal{N}$ before ReLU is applied, and we have assumed for simplicity that for every neuron z_j the sets

$$H_j = \{x \in \mathbb{R}^d : z_j(x) = 0\}$$

have codimension at least 1. It is important to note that the H_j's are no longer hyperplanes since the functions $x \mapsto z_j(x)$ are not affine. Instead, H_j is the zero level set of z_j and, following the language in Hanin (2019), we refer to the H_j as *bent hyperplanes* and

$$\mathcal{H} =: \{H_1, \ldots, H_{WL}\}$$

as a bent hyperplane arrangement. We can now describe the cells in the partition $\mathcal{P}$,

$$\mathcal{P} = \{\overline{\Omega}_\nu,\ \nu \in \{\pm 1\}^{WL},\ \dim(\Omega_\nu) = d\},$$

or equivalently the cells that are the closures of the connected components of $\mathbb{R}^d \setminus \mathcal{H}$.

To understand this setting more clearly, let us consider a neuron z in the second hidden layer of $\mathcal{N}$. Note that the function $x \mapsto z(x)$ is the output of an element of $\Upsilon^{W,1}$. Hence it is CPwL subordinate to the partition defined by the hyperplane arrangement

$$\mathcal{H}^{(1)} = \{H_1, \ldots, H_W\}$$

created by the neurons $z_1, \ldots, z_W$ in the first hidden layer of $\mathcal{N}$. On each cell $\mathcal{C}$ of the arrangement $\mathcal{H}^{(1)}$, the function $x \mapsto z(x)$ is affine. Let H_z denote the bent hyperplane associated with this neuron z from the second layer. We see that $H_z \cap \mathcal{C}$ is given by the (possibly empty) intersection of a single hyperplane with $\mathcal{C}$. However, because $x \mapsto z(x)$ is a different affine function on different cells, its zero set H_z may 'bend' at the boundary between two cells and is not given globally by a single hyperplane. More is true: while in every cell H_z coincides with a single hyperplane, globally it may have several connected components.

Just as in the case of univariate ReLU networks considered in Section 3.1, the number of cells in a deep ReLU network with any input dimension can grow exponentially with depth. In fact (see Montufar, Pascanu, Cho and Bengio 2014, Theorem 5) there are ReLU networks of depth L and width $W \geq d$ giving rise to partitions with at least

$$\left\lfloor \frac{W}{d} \right\rfloor^{d(L-1)} \sum_{j=0}^{d} \binom{W}{j}$$

cells. Similarly to the univariate case, the exponential growth in the number of pieces of the CPwL functions produced by deep networks is a consequence of composition.

Let us summarize what we know about the sets $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$. Each $S \in \Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ is a CPwL function on a finite partition of $\mathbb{R}^d$ into convex polytopes. The number of cells in the partition for S can be very large compared to

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

the number of parameters $n(W, L)$. For example, when $L = 1$ the number of cells can be of order W^d, and as L grows the number of cells can grow exponentially with respect to L. Although the number of cells is large, not every CPwL function subordinate to such a partition is in $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ since there is linear dependency imposed on the affine pieces. However, every CPwL function subordinate to a partition into convex polytopes is eventually in the $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ spaces, provided we take L and W large enough; see CPwL1 and CPwL2 in Section 3.3.3. Let us also repeat the fact that every convex polytope is the intersection of a finite number of half-spaces given by a suitable hyperplane arrangement. Finally, by refining partitions, we have that every S that is in one of the spaces $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ is a CPwL function on a partition given by a hyperplane arrangement but the number of these hyperplanes may be huge.

3.3. *Properties of deep ReLU networks*

Deep ReLU networks have a variety of remarkable properties that make their outputs a powerful approximation tool. We describe some of these properties in the present section. We study the sets $\Upsilon^{W,L} := \Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$, where W is generally fixed and L is allowed to vary. We begin by introducing a special class of ReLU networks that are effective in the construction of numerical approximation methods.

3.3.1. *Special networks and their set of outputs* $\underline{\Upsilon}^{W,L}$

We describe in this section a set of NNs that we call *special networks*, following Daubechies *et al.* (2019), which designate certain channels for specific tasks. We introduce the notation for the function $x \mapsto \eta_{i,j}(x)$ of the initial input x at the (i, j)th node. Usually $\eta_{i,j} = [z_{i,j}]_+$, that is, $\eta_{i,j}$ is the CPwL function computed by the (i, j)th neuron after ReLU is applied, but in special networks we sometimes do not apply the activation ReLU at certain nodes. However, as we shall see, the outputs of a special network, when restricted to a bounded domain, are still functions in $\Upsilon^{W,L}$.

In a special network, we reserve the top d channels to simply push forward the input values of x. Namely, channel $i \in \{1, \dots, d\}$ has

$$\eta_{i,j}(x) := x_i, \quad j = 1, \dots, L,$$

where $x = (x_1, \dots, x_d)$ is the initial input. This allows us to use x as an input to the computation performed at any later layer of the network. We refer to such channels as *source channels* (SC).

In a special network, we also designate some channels, called *collation channels* (CC), to simply aggregate the value of certain intermediate computations. In a collation channel the ReLU activation may or may not be applied. The key point here is that nodes from both collation and source channels may be ReLU-free. Therefore such networks are not true ReLU networks.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

We let $\overline{\underline{\Upsilon}}^{W,L}$ denote the set of functions S which are the outputs of a special network of width W and depth L. A useful observation made in Daubechies *et al.* (2019) is that the functions that are outputs of a special network, when restricted to a bounded domain, are in $\Upsilon^{W,L}$, that is,

$$\overline{\underline{\Upsilon}}^{W,L} \subset \Upsilon^{W,L}, \quad L \geq 1. \tag{3.10}$$

This is proved using the following observations:

- Given any configuration of weights and biases in any collation channel of a special network and any fixed compact set K of inputs, we may choose a sufficiently large value $b_{i,j}$ associated to the (i, j)th node so that $z_{i,j}(x)+b_{i,j} > 0$ for all $x \in K$. Then we construct the true ReLU network by assigning to this node the function $\eta'_{i,j}$, given by $\eta'_{i,j}(x) = [z_{i,j}(x) + b_{i,j}]_+ = z_{i,j}(x) + b_{i,j}$. The effect of $b_{i,j}$ on any subsequent computation is then eliminated by adding an extra bias (to the bias present from the special network) for any neuron from the next layer to which the output $\eta'_{i,j}(x)$ is passed. We apply this procedure to every ReLU-free node from the collation channels.
- A similar treatment to that above is used for all nodes in all source channels.
- The ReLU network that has been constructed has the same output as that of the special network we started with.

This specific trick works only when K is compact. Alternatively, at the expense of increasing the width, we can create a true ReLU network of width $W = W_0+2d+2k$, where $W_0 + d + k$ is the width of the special network with d source and k collation channels by using the identity $t = t_+ - (-t)_+$. This approach works for arbitrary inputs but at the expense of increasing the width of the network.

In what follows, we make extensive use of special networks to derive some important properties of deep networks since they facilitate many constructions.

3.3.2. Some important properties of deep ReLU networks

As noted in Observation 3.5 in the case of one-layer networks, the vector w consisting of all incoming weights into any hidden node of a ReLU network, if non-zero, can be taken to be of Euclidean norm $\|w\|_2 = 1$. Indeed, this follows from the equality

$$(w \cdot x + b)_+ = \|w\|_2 \left(\frac{w}{\|w\|_2} \cdot x + \frac{b}{\|w\|_2} \right)_+,$$

and the fact that the factor $\|w\|_2$ can be absorbed by the outgoing weights.

Next, we return to the addition property. Earlier we showed that we can add functions in $\Upsilon^{W,L}(\sigma; d, d')$ by increasing the width of the network using the method of parallelization. Here we want to observe that addition can also be performed by increasing depth and not significantly enlarging width. Here is a statement to that effect.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Addition by increasing depth. If $S_j \in \Upsilon^{W,L_j}$, $j = 1, \dots, m$, then for any $\alpha_j \in \mathbb{R}$ we have

$$S := \sum_{j=1}^{m} \alpha_j S_j \in \Upsilon^{W+d+1,L},$$

where $L := L_1 + \cdots + L_m$. In this statement all S_j's are viewed as functions on $[0,1]^d$ (or any bounded rectangle $\mathcal{R} \subset \mathbb{R}^d$).

Indeed, if $\mathcal{N}_1, \dots, \mathcal{N}_m$, are the ReLU networks that produce the S_j's, then we create from these the following special network. First we augment each of the $\mathcal{N}_j$'s by adding d source channels and one collation channel. We denote this new augmented network by $\mathcal{N}_j'$. Next we place the hidden layers of the augmented networks side by side, connect the source channels of the $\mathcal{N}_j'$'s and place (with appropriate weights) the outputs of $\mathcal{N}_j'$, $j = 1, \dots, m-1$, in the collation channel. Finally the desired sum is the output of the concatenated network (with appropriate weights). As a result, we obtain a special network with width $W + d + 1$ and depth $L = L_1 + \cdots + L_m$. The result follows from the containment (3.10).

Another operation on the output functions of ReLU networks that can be easily performed with increasing depth is to take their minimum or maximum. To that end, let us first observe that given $t, t' \in \mathbb{R}$, we have

$$\begin{aligned} \max\{t, t'\} &= (t - t')_+ + (t')_+ - (-t')_+, \\ \min\{t, t'\} &= (t')_+ - (-t')_+ - (t' - t)_+. \end{aligned} \tag{3.11}$$

Hence $\min\{t, t'\}, \max\{t, t'\} \in \Upsilon^{3,1}(\mathrm{ReLU}; 2, 1)$. We can extend the above minimization to an arbitrary number of inputs.

Minimization/maximization 1 (MM1). Let

$$z_j(x) := w_j \cdot x + b_j, \quad w_j \in \mathbb{R}^d, \quad b_j \in \mathbb{R}, \quad j = 1, \dots, m, \quad x \in \mathbb{R}^d,$$

be m affine functions on $\mathbb{R}^d$. Then, for $W = 3 \cdot 2^{\lceil \log_2 m \rceil - 1}$ and $L = \lceil \log_2 m \rceil$, we have

$$\min\{z_1(x), \dots, z_m(x)\}, \max\{z_1(x), \dots, z_m(x)\} \in \Upsilon^{W,L}(\mathrm{ReLU}; d, 1). \tag{3.12}$$

In particular, if $x_i \in \mathbb{R}$, $i = 1, \dots, m$, we have

$$\min\{x_1, \dots, x_m\}, \max\{x_1, \dots, x_m\} \in \Upsilon^{W,L}(\mathrm{ReLU}; m, 1). \tag{3.13}$$

Moreover, $\mathrm{ReLU}(\min\{z_1(x), \dots, z_m(x)\})$ and $\mathrm{ReLU}(\max\{z_1(x), \dots, z_m(x)\})$ are elements of $\Upsilon^{W,L+1}(\mathrm{ReLU}; d, 1)$.

We discuss the case of the minimum only, since the case of the maximum is almost the same. We start by proving (3.13). In our construction, we will use the fact that

$$\min\{x_1, \dots, x_{2^k}\} = \min_{1 \le j < 2^k,\, j \text{ odd}} \min\{x_j, x_{j+1}\}.$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

We first use **Parallelization** to construct for each $k \geq 1$ a neural network $\mathcal{N}_k$ with 2^k inputs and 2^{k-1} outputs that creates a vector in $\mathbb{R}^{2^{k-1}}$ with components $\min\{x_j, x_{j+1}\}$, $j = 1, \ldots, 2^k - 1$, j-odd, by stacking on top of each other the networks that produce $\min\{x_j, x_{j+1}\}$. Since each of the networks in the stack has width 3, we end up with a network with width $W_k = 3 \cdot 2^{k-1}$ and depth $L_k = 1$. We concatenate the networks $\mathcal{N}_k, \ldots, \mathcal{N}_1$ in this order, by feeding the output of $\mathcal{N}_j$ as input to $\mathcal{N}_{j-1}$. It is easy to see that the concatenated network $\mathcal{N}^k$ outputs $\min\{x_1, \ldots, x_{2^k}\}$, and has depth $L = k$ and varying widths. We can augment the network by adding extra nodes and edges to each layer so that we end up with a network of width $W = 3 \cdot 2^{k-1}$.

For general m, we let $k := \lceil \log_2 m \rceil$ and define $\hat{x}_j = x_j$, $1 \leq j \leq m$ and $\hat{x}_j := x_m$, $m < j \leq 2^k$. Applying the above to this new sequence gives the result (3.13). To show (3.12), we feed $z_j(x)$ into the first hidden layer of $\mathcal{N}_k$ by assigning appropriate input weights and node biases.

At the end, if we want to output the ReLU of min/max, we just add another hidden layer to perform the ReLU.

Another way to compute the above min/max is by increasing the depth and keeping the width relatively small by means of a recursive formula, first used by Hanin (2019).

Minimization/maximization 2 (MM2). Let

$$z_j(x) := w_j \cdot x + b_j, \quad w_j \in \mathbb{R}^d, \quad b_j \in \mathbb{R}, \quad j = 1, \ldots, m, \quad x \in \mathbb{R}^d,$$

be $m \geq 2$ affine functions on $\mathbb{R}^d$. Then we have

$$\min\{z_1(x), \ldots, z_m(x)\}, \max\{z_1(x), \ldots, z_m(x)\} \in \Upsilon^{d+1,m-1}(\mathrm{ReLU}; d, 1).$$

In addition, we have that both functions

$$\mathrm{ReLU}(\min\{z_1(x), \ldots, z_m(x)\}), \quad \mathrm{ReLU}(\max\{z_1(x), \ldots, z_m(x)\})$$

are elements of $\Upsilon^{d+1,m}(\mathrm{ReLU}; d, 1)$. In this statement all z_j's are viewed as functions on $[0, 1]^d$ (or any bounded rectangle $\mathcal{R} \subset \mathbb{R}^d$).

We discuss the case of the maximum only (the case of the minimum is treated likewise). Let $\mu_1(x) := z_1(x)$ and $\mu_k(x) := \max\{z_1(x), \ldots, z_k(x)\}$, $k \geq 2$. We use the recursion formula

$$\mu_k(x) = (\mu_{k-1}(x) - z_k(x))_+ + z_k(x), \quad 2 \leq k \leq m,$$

and discuss the case $\mathcal{R} = [0, 1]^d$. For the case of a general rectangle $\mathcal{R}$ one needs to add appropriate biases. Our construction is as follows:

- The first d channels of the network push forward the variables $x_1, \ldots, x_d$. Their nodes can be viewed as ReLU nodes since $t_+ = t$ for $t \geq 0$.
- The $(d + 1)$th channel computes in its first node $(z_1(x) - z_2(x))_+$. Note that if we wanted to, we could stop and output $\mu_2(x)$ at this stage. The jth node

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

of this channel, $j = 2, \dots, m-2$, computes $(\mu_j(x) - z_{j+1}(x))_+$, which is then given as an input to the $(j+1)$th node. The final layer $L = m-1$ will hold $\mu_{m-1}(x)$ and hence can output $\mu_m(x)$.

To show the last statement, we add a hidden layer after the last hidden layer of the NN from the construction above to perform the ReLU of the max/min. Of course, we could augment the resulting network by adding nodes and connections so that we have a fully connected feed-forward NN.

Note that if we want to compute the min/max of m linear functions z_j, $j = 1, \dots, m$, viewed as functions on the whole $\mathbb{R}^d$, we can do this if we take $W = 2d+1$, since any channel i, $1 \le i \le d$, in the above construction doubles in order to be able to forward the input $t = t_+ - (-t)_+$.

More general statements hold when instead of computing the min/max of affine functions we have to find the min/max of outputs of neural networks.

Minimization/maximization 3 (MM3). Let $m \ge 2$ and let the functions S_j be in $\Upsilon^{W_j, L_0}(\mathrm{ReLU}; d, 1)$, $j = 1, \dots, m$. Then

$$S := \min\{S_1, \dots, S_m\} \in \Upsilon^{W,L}(\mathrm{ReLU}; d, 1), \tag{3.14}$$

where

$$W := \max\{W_1 + W_2 + \cdots + W_m, 3 \cdot 2^{\lceil \log_2 m \rceil - 1}\}, \quad L = L_0 + \lceil \log_2 m \rceil.$$

If $W_j \ge 3$, $j = 1, \dots, m$, we have $W = W_1 + W_2 + \cdots + W_m$. The same statement holds for $\max\{S_1, \dots, S_m\}$.

We use **Parallelization** to construct the first L_0 hidden layers of the network $\mathcal{N}$ that outputs S. Then from the L_0th layer we can output any of the S_j, $j = 1, \dots, m$. We concatenate this with the network in MM1 which has $\lceil \log_2 m \rceil$ hidden layers to complete the construction of $\mathcal{N}$. Clearly the resulting network has varying width, where the first L_0 layers have width $W_1 + W_2 + \cdots + W_m$ while the last $\lceil \log_2 m \rceil$ layers have width $3 \cdot 2^{\lceil \log_2 m \rceil - 1}$. We augment this network by adding extra nodes and edges. At the end, our network has width $W = \max\{W_1 + W_2 + \cdots + W_m, 3 \cdot 2^{\lceil \log_2 m \rceil - 1}\}$. In the case of $W_j \ge 3$, $W_1 + \cdots + W_m \ge 3 \cdot 2^{\lceil \log_2 m \rceil - 1}$, which gives that $W = W_1 + \cdots + W_m$.

It is also possible to do the minimization by increasing the depth of the network while keeping the width relatively the same.

Minimization/maximization 4 (MM4). Let $m \ge 2$ and let the functions S_j be in $\Upsilon^{W_0, L_j}(\mathrm{ReLU}; d, 1)$, $j = 1, \dots, m$. Then $S := \min\{S_1, \dots, S_m\}$ belongs to $\Upsilon^{W,L}(\mathrm{ReLU}; m, 1)$, where the width $W := \max\{W_0, 3\} + d + 1$ and the depth $L = \sum_{j=1}^m L_j + m - 1$. The same statement holds for $\max\{S_1, \dots, S_m\}$. Here all S_j's are viewed as functions on $[0,1]^d$ (or any bounded rectangle $\mathcal{R} \subset \mathbb{R}^d$).

In order to construct a neural network $\mathcal{N}$ which shows that $S \in \Upsilon^{W,L}$, we utilize **Concatenation** in place of **Parallelization** and we use special networks. Let $\mathcal{N}_j$ be a network of width W_0 and depth L_j which outputs S_j, $j = 1, \dots, m$. To each of the networks $\mathcal{N}_j$ we add d source channels to push forward the original

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

inputs $x_1, \dots, x_d$ and one collation channel that we will use to update computations towards outputting S. Let us denote these special networks by $\mathcal{N}'_j$. We now explain how to construct $\mathcal{N}$. The first L_1 hidden layers of $\mathcal{N}$ consist of those of $\mathcal{N}'_1$. The collation channel simply pushes forward zero for these layers. We concatenate $\mathcal{N}'_1$ with $\mathcal{N}'_2$ by placing S_1 in the collation channel of $\mathcal{N}'_2$ and then pushing it forward, and by placing the outputs of the source channels of $\mathcal{N}'_1$, multiplied by appropriate weights (those that enter the first layer of $\mathcal{N}_2$), into the first hidden layer of $\mathcal{N}'_2$. If $m = 2$, we can complete the construction by placing a last hidden layer which takes S_1 from the collation channel and S_2 as an output from $\mathcal{N}'_2$ and computes $(S_1 - S_2)_+$, $(S_2)_+$ and $(-S_2)_+$. We augment the resulting network with additional nodes, if necessary, so that we have a special network with width W. This network outputs S and has depth $L = L_1 + L_2 + 1$ and width W. If $m > 2$, we continue by concatenating with $\mathcal{N}'_3$. The collation channel is now occupied by $T_2 := \min\{S_1, S_2\}$. If $m = 3$, then we complete as before by adding a layer to compute $(T_2 - S_3)_+$, $(S_3)_+$ and $(-S_3)_+$. Continuing in this way, we obtain the desired network.

3.3.3. General CPwL functions

We observed earlier that if $\mathcal{P}$ is a partition into a finite number of cells obtained from a hyperplane arrangement, then not every CPwL function subordinate to this partition is in the set $\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$. In particular, the latter set does not include CPwL functions with compact support. This can be remedied by slightly increasing the depth of the network. More precisely, let Δ be any simplex in $\mathbb{R}^d$ and let x^* be any point in its interior. Since Δ is a convex polytope with $d + 1$ facets, there exist $d + 1$ affine functions $z_j : \mathbb{R}^d \to \mathbb{R}$ such that $z_j(x^*) = 1$ and

$$\Delta = \{x \in \mathbb{R}^d : z_j(x) \geq 0,\ j = 1, \dots, d + 1\}.$$

Thus the *tent function*

$$T := T_{\Delta, x^*} := \mathrm{ReLU}(\min\{z_1, \dots, z_{d+1}\}) \tag{3.15}$$

vanishes outside Δ and satisfies $T(x^*) = 1$. For example, when $d = 1$ this is the hat function on $\mathbb{R}$. The construction MM1 ensures the following:

Tent functions. For each d-dimensional simplex Δ and each x^* in its interior, the tent function T_{Δ, x^*} is in $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ with $W = 3 \cdot 2^{\lceil \log_2(d+1) \rceil - 1}$ and $L = 1 + \lceil \log_2(d + 1) \rceil$.

With these remarks in hand, let us now turn to the question of whether every CPwL function is in one of the spaces $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$. We make the following observations.

CPwL1. If S is a CPwL function on $\mathbb{R}^d$, then

$$S \in \Upsilon^{W',L}(\mathrm{ReLU}; d, 1), \quad \text{with } L = \lceil \log_2(d + 1) \rceil, \quad \text{for } W' \text{ sufficiently large.}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

This is proved in Arora, Basu, Mianjy and Mukherjee (2018*a*) by using the fact that any CPwL function S can be written as a linear combination of piecewise linear convex functions, each with at most $(d+1)$ affine pieces, that is,

$$S = \sum_{j=1}^{p} \varepsilon_j \Big(\max_{i \in S_j} z_i \Big), \quad \varepsilon_j \in \{-1, 1\}, \quad S_j \subset \{1, 2, \dots, k\}, \tag{3.16}$$

with $s_j := \#(S_j) \le d+1$, for some affine functions $z_1, \dots, z_k$.

To show that $S \in \Upsilon^{W',L}(\mathrm{ReLU}; d, 1)$, we use MM1 to show that for every $j = 1, \dots, p$, the function $\max_{i \in S_j} z_i$ is in $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ with

$$W = 3 \cdot 2^{\lceil \log_2 s_j \rceil - 1}, \quad L = \lceil \log_2 s_j \rceil.$$

For the proof, we can assume that $s_j = d+1$ for all j by artificially writing an index already in S_j several times, so that we end up with networks with the same depth $L = \lceil \log_2(d+1) \rceil$. Using **Parallelization**, we then stack these networks to produce $S \in \Upsilon^{W',L}(\mathrm{ReLU}; d, 1)$ with $L = \lceil \log_2(d+1) \rceil$ and $W' = 3p2^{\lceil \log_2(d+1) \rceil - 1}$.

At the other extreme, one may wish to keep the width W of the ReLU network as small as possible at the expense of letting the depth of the network grow. In this direction, we have the following result.

CPwL2. If $S \colon \mathcal{R} \to \mathbb{R}$ is a CPwL function defined on a rectangle $\mathcal{R} \subset \mathbb{R}^d$, then $S \in \Upsilon^{d+2,L}(\mathrm{ReLU}; d, 1)$ for L suitably large.

To show this, we use the representation (3.16) for S and utilize MM2 to view each of the functions $\max_{i \in S_j} z_i$ as an output of a network $\mathcal{N}_j$ that produces $\Upsilon^{d+1, s_j - 1}(\mathrm{ReLU}; d, 1)$. We concatenate the networks $\mathcal{N}_j$, $j = 1, \dots, p$, by placing them next to each other and connecting their source channels. We add a collation channel where we store the consecutive outputs $\varepsilon_j(\max_{i \in S_j} z_i)$ from $\mathcal{N}_j$. The resulting network computes S and has width $W = d+2$ and depth at most $L = pd$.

A result along these lines was proved by Hanin (2019), who showed that the output of any ReLU network can be generated by a sufficiently deep ReLU network with fixed width $W = d+3$.

3.3.4. Finite element spaces

One of the most popular methods of approximation used in numerical analysis is the finite element method (FEM). This method employs certain linear spaces of piecewise polynomials. In its simplest case, one partitions a given polyhedral domain $\Omega \subset \mathbb{R}^d$ into simplicial cells with some requirements on the cells to avoid hanging nodes and small angles. The linear space $X(\mathcal{P})$ of CPwL functions subordinate to such a partition $\mathcal{P}$ is used for the approximation. A natural question is if and how we can use ReLU NNs in place of $X(\mathcal{P})$.

Rather than address this question in its full generality, we consider only a very special setting that will be sufficient for our discussion of NN approximation given in later sections of this paper. The reader can consult He, Li, Xu and Zheng (2020)

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

and Opschoor, Petersen and Schwab (2019*a*) for a more far-reaching exposition of the relation between FEM and NNs.

For our special example, we begin with $\Omega = [0,1]^d$, and given $n \geq 1$, we consider the uniform partition $\mathcal{Q}_n$ of Ω into n^d cubes with side length $1/n$. We let V_n denote the set of vertices of the cubes in $\mathcal{Q}_n$. There are $(n+1)^d$ such vertices. Each cube $Q \in \mathcal{Q}_n$ can in turn be partitioned into $d!$ simplices using the so-called Kuhn triangulation with northwest diagonal. This gives a partition $\mathcal{K}$ of Ω into $n^d d!$ simplices. Let $X(\mathcal{K})$ be the space of all CPwL functions defined on Ω and subordinate to $\mathcal{K}$. This is a linear space of dimension $N = (n+1)^d$. A basis for $X(\mathcal{K})$ is given by the nodal functions $\{\phi_v,\ v \in V_n\}$, which are the CPwL functions defined on Ω, subordinate to $\mathcal{K}$, and satisfy

$$\phi_v(v') = \delta(v, v'), \quad v, v' \in V_n, \tag{3.17}$$

where δ is the usual Kronecker delta function. Each $S \in X(\mathcal{K})$ has the representation

$$S = \sum_{v \in V_n} S(v)\phi_v. \tag{3.18}$$

FEM spaces. Let $X(\mathcal{K})$ be the finite element space in d dimensions described above, and let $d^* := (d+1)!$. Then the following holds:

$$X(\mathcal{K}) \subset \Upsilon^{W,L}(\mathrm{ReLU}; d, 1), \qquad W = 3(n+1)^d 2^{\lceil \log_2 d^* \rceil - 1}, \quad L := 1 + \lceil \log_2 d^* \rceil,$$
$$X(\mathcal{K}) \subset \Upsilon^{d+2,L'}(\mathrm{ReLU}; d, 1), \quad L' = (n+1)^d d^*.$$

To prove these statements, we first observe that each nodal basis function ϕ_v can be expressed as

$$\phi_v = \mathrm{ReLU}(\min\{z_\Delta : \Delta \in D_v\}), \tag{3.19}$$

where D_v is the set of simplices in $\mathcal{K}$ that have v as one of their vertices. There are $(d+1)!$ such simplices when v is an internal vertex and less than that for vertices on the boundary of Ω. The function z_Δ is the linear function which is one at v and vanishes on the facet of Δ opposite to v. If $|D_v| < (d+1)!$, we add artificially some of the functions z_Δ that are already in D_v so that we end up with $(d+1)!$ not necessarily different functions, since later we will do parallelization that requires the depth of certain networks to be the same.

It follows from MM1 that $\phi_v \in \Upsilon^{\widetilde{W},L}(\mathrm{ReLU}; d, 1)$, $\widetilde{W} := 3 \cdot 2^{\lceil \log_2 d^* \rceil - 1}$, $L = 1 + \lceil \log_2 d^* \rceil$. Using **Parallelization**, we stack the networks $\mathcal{N}_v$ that output the ϕ_v's to obtain a network with width $W = (n+1)^d \widetilde{W}$ that can output any linear combination of the ϕ_v's. Hence $X(\mathcal{K})$ is contained in $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ for the advertised values of W and L. Note that since $X(\mathcal{K})$ is generated by parallelization of $(n+1)^d$ networks of width $\widetilde{W}$, the number of parameters used in the resulting network is $O((n+1)^d)$.

To prove the second containment, we first observe from MM2 that each of the nodal basis functions $\phi_v \in \Upsilon^{d+1,|D_v|}(\mathrm{ReLU}; d, 1)$. We then utilize **Concatenation**

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

of the networks $\mathcal{N}_v$ used to produce ϕ_v and add a collation channel for the computation of S; see (3.18) (as is done for the CPwL2 construction). The resulting network has width $W = d + 2$ and depth at most $(n + 1)^d d^*$.

Several remarks are in order concerning this result. First, note that the number of parameters used in both NNs is comparable (up to a factor depending on d) to the dimension $(n + 1)^d$ of $X(\mathcal{K})$. The most important point to stress is that when using the set $\Upsilon^{W,L}$ in place of a piecewise linear FEM space, we are using a much larger nonlinear family as an approximation tool. Indeed, the set $\Upsilon^{W,L}$ not only contains the FEM space $X(\mathcal{K})$ based on the initial choice of partitioning but also contains an infinite number of such FEM spaces corresponding to an infinite number of possible ways to partition Ω. In fact the NN approach is rather more than a simple generalization of the adaptive finite element method (AFEM), where one is allowed to adaptively choose partitions (from a restricted family of partitions). It will be shown in Section 8.7 that these NNs provide a provably better approximation rate to various Sobolev and Besov classes than that provided by the FEM spaces. While this seems like a tremendous advantage for NNs over FEMs, one must address (in the specific problem setting) how one (near-) optimally chooses the parameters of these NNs.

In the case when FEMs are used to numerically solve linear elliptic PDEs, one can employ the Galerkin method, which finds a (near-) best approximation to the solution to the PDE by projecting onto $X(\mathcal{K})$. This is well understood and quantified in both theory and practice via theorems that bound error and establish stable numerical implementation.

When we eventually discuss quantitative theorems for NN approximation, we shall see that the known results point to a tremendous potential increase in approximation efficiency (error versus number of parameters needed) when using NNs for the numerical solution of elliptic problems. Whether this advantage can be maintained in concrete stable numerical implementation is less clear.

3.4. Width versus depth

An underlying issue when choosing an NN architecture to be used in a numerical setting is whether to increase the width or the depth of the NN when one is willing to allocate more parameters to improve accuracy. Suppose we fix a bound n on the number of parameters to be used and ask which of the sets $\Upsilon^{W,L}$ depending on at most n parameters we should employ in designing a numerical algorithm. All other issues being the same, the general consensus is that in practice deeper networks are preferable. We make some comments to explain this preference from the point of view of the enhanced approximation capacities of deeper networks.

First, we have shown that addition of the output functions of an NN can be implemented by increasing either width (parallelization) or depth (concatenation) with a controlled increase in the number of parameters. However, certain operations such as composition and forming minima can only be implemented by increasing

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

depth. So, for example, if we fix a width $W = W_0$ sufficiently large to accommodate d source channels and a couple of collation channels, then we can seemingly implement as outputs from $\Upsilon^{W_0,L}$ all functions that occur as outputs of shallower networks with a comparable number of parameters. The only rigorous statement given to this effect was for ReLU networks with $d = 1$. In this case Daubechies *et al.* (2019) proved that for any fixed $W_0 \geq 4$ we have

$$\Upsilon^{n,1}(\mathrm{ReLU}; 1, 1) \subset \Upsilon^{W_0,L}(\mathrm{ReLU}; 1, 1), \tag{3.20}$$

provided the elements in these sets are viewed as functions on $[0, 1]$ (or any finite interval $[r, e]$), and $L \asymp n/W_0^2$, where the constants in $\asymp$ are absolute constants. Note that the number of parameters $n(W_0, L)$ determining the set $\Upsilon^{W_0,L}$ is $n(W_0, L) \asymp W_0^2 L \asymp n$, and therefore comparable to the number of parameters in $\Upsilon^{n,1}$. However, $\Upsilon^{W_0,L}$ is richer, since it contains compositions, for example. So in this special case depth beats width. This leads us to formulate the following general question.

Problem 3.7. Are shallow networks always contained in deep networks of fixed width W_0 with the same number of parameters? More precisely, is it true that if we fix the depth L and the width W_0, we have the inclusion $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1) \subset \Upsilon^{W_0,L_0}(\mathrm{ReLU}; d, 1)$ whenever L_0 and W satisfy $n(W_0, L_0) \asymp n(W, L)$, where the constants in $\asymp$ depend at most on d?

The results given above show that Problem 3.7 has a positive answer if we are not concerned about the number of parameters. Indeed, each function in $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ is a CPwL function, and therefore, according to CPwL2, is in $\Upsilon^{d+2,L}(\mathrm{ReLU}; d, 1)$ for L suitably large. So the key issue in Problem 3.7 is the control on the number of parameters.

3.5. Interpolation by neural network outputs

A common strategy for approximating a given target function f is to interpolate some of its point values. Although this is often not a good method for approximation, it is important to understand when we can interpolate a given set of data, and how stable the process is. A satisfactory understanding of interpolation using NNs is far from complete. The purpose of this section is to frame the interpolation problem and point out what is known. We begin by considering interpolation by NNs with an arbitrary activation function σ and later specialize to ReLU activations.

Let $\Upsilon^{W,L}(\sigma) = \Upsilon^{W,L}(\sigma; d, 1)$ be the set of outputs of NNs with activation σ, input dimension d, output dimension 1, width W and depth L. Given a finite set of data points $(x^{(i)}, y_i)$, with $x^{(i)} \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, $i = 1, \ldots, D$, a natural question is whether there is an $S \in \Upsilon^{W,L}(\sigma)$ which interpolates the given data in the sense that

$$S(x^{(i)}) = y_i, \quad i = 1, \ldots, D. \tag{3.21}$$

This is the *existence question* for data interpolation. In the case that interpolants

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

exist, let $\mathcal{S}_I := \mathcal{S}_I(W, L; \sigma, d)$ denote the set of functions $S \in \Upsilon^{W,L}(\sigma; d, 1)$ which satisfy the interpolation conditions (3.21).

Given that we are going to use the set $\Upsilon^{W,L}(\sigma; d, 1)$ for interpolation, the first question to ask is as follows.

Question. Determine the largest value $D^* := D^*(W, L; \sigma, d)$ such that the interpolation problem has a solution from $\Upsilon^{W,L}(\sigma; d, 1)$ for all data sets of size D^*.

One expects that D^* should be closely related to the number of parameters used to describe $\Upsilon^{W,L}(\sigma; d, 1)$.

There seems to be only one general theorem addressing the interpolation problem for general activation functions σ. It applies to the case of single-hidden-layer networks, *i.e.* $L = 1$, and is discussed in detail in the survey article by Pinkus (1999); see Theorem 5.1 in that paper.

Interpolation from $\Upsilon^{W,1}(\sigma; d, 1)$. If $\sigma \in C(\mathbb{R})$ is not a polynomial, then

$$D^*(W, 1; \sigma, d) \geq W, \quad W \geq 1. \tag{3.22}$$

The following sections discuss the interpolation problem for ReLU activation where more results are known.

3.5.1. Interpolation for $\Upsilon^{W,1}(\mathrm{ReLU}; 1, 1)$

The interpolation question is easiest to answer for ReLU networks with $d = L = 1$. In this case we know that the set $\Upsilon^{W,1}(\mathrm{ReLU}; 1, 1)$ is almost the same as the space $\Sigma_{W,1} = \Sigma_{W,1}(\mathrm{CPwL})$ of CPwL functions subordinate to a partition of $\mathbb{R}$ into W intervals ($W - 1$ breakpoints); see (3.2). Interpolation by functions in $\Sigma_{W,1}$ is well understood; see de Boor (1978). In the case of $\Upsilon^{W,1}(\mathrm{ReLU}; 1, 1)$, we claim that

$$D^*(W, 1; \mathrm{ReLU}, 1) = W + 1. \tag{3.23}$$

To show this, we will use the representation (3.8) for functions S from this set. Consider data points $\{(t^{(i)}, y_i)\}$, $i = 1, \dots, D$, with $t^{(1)} < \cdots < t^{(D)}$. We first show that when $D = W + 1$, interpolation is not only possible, but there are infinitely many $S \in \Upsilon^{W,1}(\mathrm{ReLU}; 1, 1)$ for which

$$S(t^{(i)}) = y_i, \quad i = 1, \dots, W + 1.$$

We take any points ξ_j that satisfy the interlacing property

$$t^{(1)} < \xi_1 < t^{(2)} < \cdots < \xi_W < t^{(W+1)},$$

and consider the function

$$S(t) := c + \sum_{j=1}^{W} a_j (t - \xi_j)_+ \in \Upsilon^{W,1}(\mathrm{ReLU}; 1, 1). \tag{3.24}$$

We establish that interpolation is possible by induction on W. When $W = 1$, we choose $c := y_1$ and a_1 so that $c + a_1(t^{(2)} - \xi_1) = y_2$. For the induction step, let

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

$S_0(t) := c + \sum_{j=1}^{W-1} a_j(t-\xi_j)_+$ satisfy the first W interpolation conditions. We define a_W so that we have

$$a_W(t^{(W+1)} - \xi_W) + S_0(t^{(W+1)}) = y_{W+1}.$$

Then

$$S(t) := S_0(t) + a_W(t - \xi_W)_+ \in \Upsilon^{W,1}(\text{ReLU}; 1, 1)$$

satisfies all of the interpolation conditions. This shows that the interpolation conditions can always be satisfied and that the set $\mathcal{S}_I$ is infinite since we have infinitely many choices for the ξ_i's.

Finally, we want to see that interpolation at $W+2$ points is generally not possible. To that end we use the following proposition, which will also be useful when we discuss the Vapnik–Chervonenkis (VC) dimension of NNs.

Proposition 3.8. Let $n \geq 3$ and let $t^{(1)} < t^{(2)} \cdots < t^{(n)}$ be n distinct arbitrary points. Let $y_1, \ldots, y_n$ be such that $y_j y_{j+1} < 0$, $j = 1, \ldots, n-1$. Then there is no $S \in \Upsilon^{n-2,1}(\text{ReLU}; 1, 1)$ such that $S(t^{(j)}) = y_j$, $j = 1, \ldots, n$.

Proof. Without loss of generality we can assume that $y_1 > 0$. We prove the proposition by induction on n. We first consider the case $n = 3$. If $S \in \Upsilon^{1,1}(\text{ReLU}; 1, 1)$, then we have the representation $S(t) = c + a(t - \xi)_+$ or $S(t) = c + a(\xi - t)_+$. We assume the first representation since the second one is treated in a similar way. In this case we note the following:

- If $\xi \leq t^{(1)}$, then S is linear on $[t^{(1)}, \infty)$ and therefore cannot satisfy the three interpolation conditions.
- If $\xi \in (t^{(1)}, t^{(2)})$, then in order for S to satisfy the first two interpolation conditions we would need $c > 0$ and $a < 0$. So the function S is then a non-increasing function of t and thus $S(t^{(3)}) \leq S(t^{(2)})$, which shows that S cannot satisfy the third interpolation condition.
- If $\xi \geq t^{(2)}$, then S cannot satisfy the first two interpolation conditions, since S is constant on $(-\infty, \xi]$.

Now we consider the induction step. Suppose that we have proved the proposition for a value of $n \geq 3$ and consider $n+1$ interpolation points. If S is any function in $\Upsilon^{n-1,1}(\text{ReLU}; 1, 1)$, then

$$S(t) = S_0(t) + a(t - \xi_{n-1})_+ \quad \text{or} \quad S(t) = S_0(t) + a(\xi_{n-1} - t)_+,$$

where $S_0 \in \Upsilon^{n-2,1}(\text{ReLU}; 1, 1)$, and has breakpoints $\xi_1 < \ldots < \xi_{n-2}$, with $\xi_{n-2} < \xi_{n-1}$. We again consider only the first possibility, since the other one is handled similarly. We show that S cannot satisfy the interpolation conditions by considering the following two cases:

- If $t^{(n)} < \xi_{n-1}$, then $S_0(t^{(j)}) = S(t^{(j)}) = y_j$, $j = 1, \ldots, n$, and S_0 would contradict the induction hypothesis. Hence this case is not possible.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

- If $\xi_{n-1} < t^{(n)} < t^{(n+1)}$, then S is a linear function on $[\xi_{n-1}, \infty)$. Note that $y^* := S_0(\xi_{n-1}) = S(\xi_{n-1})$ and because of $y_n y_{n+1} < 0$, $\operatorname{sign}(S(\xi_{n-1})) = \operatorname{sign}(y_n)$. Note that S cannot satisfy the last three interpolation conditions corresponding to $t^{(n-1)}, t^{(n)}, t^{(n+1)}$ unless we have $t^{(n-1)} < \xi_{n-1}$. Thus $S_0(t^{(j)}) = S(t^{(j)}) = y_j$, for $j = 1, \dots, n-1$, and $S_0(\xi_{n-1}) = y^*$, where the sign of y^* is the same as the sign of y_n. Therefore, according to the induction hypothesis, such S_0 cannot be an output of $\Upsilon^{n-2,1}(\mathrm{ReLU}; 1, 1)$.

This completes the proof of the proposition. □

3.5.2. Interpolation for $\Upsilon^{W_0,L}(\mathrm{ReLU}; 1, 1)$

It is also possible to produce an interpolant to given data by using deep networks with a fixed width. This of course follows from (3.20) together with what we have just proved. However, we wish to give a direct construction because it will be used later in this paper.

Proposition 3.9. Given D points $0 \le t^{(1)} < t^{(2)} < \cdots < t^{(D)} \le 1$ and values $y_j \in \mathbb{R}$, $j = 1, \dots, D$, there is an $S \in \Upsilon^{3,D-1}(\mathrm{ReLU}; 1, 1)$ which interpolates these data, namely

$$S(t^{(j)}) = y_j, \quad j = 1, \dots, D. \tag{3.25}$$

Therefore $D^*(3, L; \mathrm{ReLU}, 1) \ge L + 1$, where $L \ge 1$.

Proof. We have shown above that there is an S of the form (3.24) with $W := D - 1$, that satisfies the interpolation conditions (3.25). We view S as a function on $[0, 1]$ and construct a special network that outputs any such S. The first channel of this special network is a source channel that pushes forward the input t and the last channel is a collation channel. This last channel is initialized with 0 at layer 1 and then successively collects the sums $\sum_{i=1}^{j-1} a_i(t - \xi_i)_+$ at layers $2, \dots, D-1$, respectively, while the middle channel successively produces the terms $a_j(t - \xi_j)_+$, at layers $j = 1, \dots, D-1$, using the inputs t from the source channel. We can then output S from layer $D - 1$. □

3.5.3. Interpolation from $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$

We now turn to results that hold for general $d \ge 1$. There is a simple way to derive interpolation results for arbitrary $d > 1$ from those for $d = 1$. Let

$$\mathcal{X} := \{x^{(j)}, \ j = 1, \dots, D\} \subseteq \mathbb{R}^d$$

be any finite collection of data sites. A simple measure-theoretic argument shows that there exists a unit vector $v \in \mathbb{R}^d$ for which the points $t^{(j)} \in \mathbb{R}$, given by $t^{(j)} := v \cdot x^{(j)}$, $j = 1, \dots, D$, are all distinct. If g is any univariate function which satisfies

$$g(t^{(j)}) = y_j, \quad j = 1, \dots, d,$$

then the ridge function $f(x) := g(v \cdot x)$ satisfies $f(x^{(j)}) = y^j$, $j = 1, \dots, D$. We utilize this observation to prove the following.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Proposition 3.10. For any $W, L \geq 1$, we have

$$D^*(W, L; \mathrm{ReLU}, d) \geq D^*(W, L; \mathrm{ReLU}, 1).$$

Proof. Given a data set $\mathcal{X} \subset \mathbb{R}^d$ of size D, we choose v as above to arrive at the points $t^{(j)} \in \mathbb{R}$, $j = 1, \ldots, D$. If $D \leq D^*(W, L; \mathrm{ReLU}, 1)$, then there is an $S \in \Upsilon^{W,L}(\mathrm{ReLU}; 1, 1)$ which satisfies $S(t^{(j)}) = y_j$, $j = 1, \ldots, D$. Then the function $T(x) := S(v \cdot x) \in \Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ interpolates the multidimensional data set $\mathcal{X}$. □

While the above proposition is of theoretical interest, it is not used in practice because the ridge function interpolant does not reflect the local flavour of the data. A more common scenario is to construct via ReLU networks a dual basis $\{\phi_j\}$, $j = 1, \ldots, D$, for the data sites, *i.e.* a basis that satisfies the conditions

$$\phi_i(x^{(j)}) = \delta_{i,j}, \quad 1 \leq i,\ j \leq D.$$

The goal is to construct a locally supported dual basis. In that case the interpolation operator

$$P_{\mathcal{X}}(f) := \sum_{j=1}^{D} f(x^{(j)})\phi_j$$

is a bounded projection onto $\mathrm{span}\{\phi_j\}$ whenever the data sites are in Ω. The norm of this projector,

$$\|P_{\mathcal{X}}\|_{C(\Omega)\to C(\Omega)} = \max_{x\in\Omega} \sum_{j=1}^{D} |\phi_j(x)|,$$

to a large extent determines the approximation properties of interpolation at these sites.

We have already discussed such dual bases in the context of FEM, where for the Kuhn simplicial decomposition $\mathcal{K}$ of $\Omega = [0, 1]^d$ we showed that the space $X(\mathcal{K})$ spanned by the nodal basis $\{\phi_v\}$ (see (3.17)) is contained in $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ for certain choices of W and L with the number of parameters $n(W, L)$ comparable to the dimension of $X(\mathcal{K})$. In this case the nodal basis forms a partition of unity $\sum_v \phi_v \equiv 1$ on Ω and the projection operator $P_{\mathcal{X}}$ is of norm one. It thus follows that

$$\begin{aligned}
\mathrm{dist}(f, \Upsilon^{W,L}(\mathrm{ReLU}; d, 1))_{C(\Omega)} &\leq \|f - P_{\mathcal{X}} f\|_{C(\Omega)} \\
&\leq \|f - S\|_{C(\Omega)} + \|P_{\mathcal{X}}(f - S)\|_{C(\Omega)} \\
&\leq 2\ \mathrm{dist}(f, X(\mathcal{K}))_{C(\Omega)},
\end{aligned}$$

where we insert the best approximation S to f from $X(\mathcal{K})$ to obtain the last inequality. This allows us to deduce estimates for NN approximation from those known in FEM and also to exhibit simple linear operators that achieve these bounds.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

3.6. VC dimension of ReLU outputs

An important ingredient in understanding the approximation power of ReLU networks is the Vapnik–Chervonenkis (VC) dimension of the sets of their outputs $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$. This topic is by now well studied; see Bartlett, Harvey, Liaw and Mehrabian (2019) for a summary of the most recent results. Here we shall only discuss the results on the VC dimension that are important for approximation.

Let $\mathcal{F}$ be a collection of real-valued functions defined on $\Omega \subset \mathbb{R}^d$. We say that a set $\{x^{(1)}, \dots, x^{(n)}\} \subset \Omega$ is *shattered* by $\mathcal{F}$ if, for each subset $\Lambda \subset \{1, \dots, n\}$ there is a function $f = f_\Lambda \in \mathcal{F}$ such that

$$f(x^{(i)}) > 0 \quad \text{if and only if} \quad i \in \Lambda.$$

The maximum value of n for which there exists such a collection of n points that are shattered by $\mathcal{F}$ is called the Vapnik–Chervonenkis (VC) dimension of $\mathcal{F}$ and is denoted by $\mathrm{VC}(\mathcal{F})$; see Vapnik (1989).

In the case that $\mathcal{F}$ is one of the sets $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$, the VC dimension of $\mathcal{F}$ is the largest value of n for which there exist n points such that for any assignment of signs $\varepsilon_i \in \{-1, +1\}$, there is an $S \in \mathcal{F}$ such that

$$\varepsilon_i S(x^{(i)}) > 0, \quad i = 1, \dots, n. \tag{3.26}$$

This follows from the fact that whenever $S \in \mathcal{F}$, then $S + c$, $c \in \mathbb{R}$, is also in $\mathcal{F}$. We sometimes use property (3.26) instead of the original definition of shattering for the output of NNs in going forward.

Let us note that the definition of the VC dimension of $\mathcal{F}$ only requires the existence of one set of points where shattering takes place. When proving upper bounds on the error of approximation, it is useful to know precisely which collections of points can be shattered. The reader will see how this issue arises when we use the VC dimension in proving approximation results.

We are interested in describing the VC dimension of the set $\mathcal{F}$ of outputs of ReLU networks in terms of the number $n(W, L)$ of their parameters. Let us now consider what is known in the special cases of interest to us.

3.6.1. VC dimension of $\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$

We first consider the space $\Upsilon^{W,1}$ of function of d variables which is described by $n(W, 1) = (d + 2)W + 1$ parameters.

Lemma 3.11. We have the following upper and lower bounds for the VC dimension of $\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$, $W \geq 1$:

(i) If $d = 1$, then $\mathrm{VC}(\Upsilon^{W,1}(\mathrm{ReLU}; 1, 1)) = W + 1$.

(ii) If $d \geq 2$, then $\mathrm{VC}(\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)) \leq C_0 W \log_2 W$, where C_0 depends only on d.

(iii) If $d \geq 4$, then $\mathrm{VC}(\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)) \geq c_0 W \log_2 W$, where c_0 depends only on d.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

(iv) If $d = 2, 3$, then $\mathrm{VC}(\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)) \geq W + 1$.

Proof. (i) The VC dimension in this case is at least $W + 1$ because we can interpolate any $W + 1$ data by an element from $\Upsilon^{W,1}(\mathrm{ReLU}; 1, 1)$; see (3.23). On the other hand, the VC dimension is at most $W + 1$ because of Proposition 3.8.

(ii) This upper bound can be found in Bartlett *et al.* (2019).

(iii) The lower bounds in the case $d \geq 4$ can be derived from known lower bounds for the VC dimension of the collection $\mathcal{C} = \{R\}$ of sets R which are the union of W closed half-spaces. Indeed, whenever points $P_1, \ldots, P_m$ are shattered by $\mathcal{C}$, then they are shattered by $\Upsilon^{W,1}$. To see this, suppose that Λ is any subset of these points. We can construct $S \in \Upsilon^{W,1}$ that is positive on this set and zero on the remaining points as follows. Let R_j, $j = 1, \ldots, W$, be the closed half-spaces whose union contains only the points from Λ and none of the rest of the P_j's, $j = 1, \ldots, m$. Each of these half-spaces can be represented as $w_j \cdot x + b_j \geq 0$ for some $w_j \in \mathbb{R}^d$, $b_j \in \mathbb{R}$. Then, if $\varepsilon > 0$, the function

$$S := \sum_{j=1}^{W} (w_j \cdot x + b_j + \varepsilon)_+ \in \Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$$

will be positive on the points in Λ and zero on the rest of the points P_j, provided we take ε small enough. It follows that

$$\mathrm{VC}(\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)) \geq \mathrm{VC}(\mathcal{C}),$$

and hence the lower bounds stated in (iii), follow from the lower bounds on the VC dimension of $\mathcal{C}$ given in Csiskos, Kupavskii and Mustafa (2019).

(iv) The lower bounds in this case follow from the fact that we can interpolate any data at any $(W + 1)$ data sites; see Proposition 3.10 and (3.23). □

It appears that the VC dimension of $\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$ when $d = 2, 3$ is not completely determined because of the discrepancy between the upper and lower bounds in the above lemma.

3.6.2. *VC dimension of* $\Upsilon^{W_0,L}(\mathrm{ReLU}; d, 1)$

Next we consider the case where W_0 is fixed but sufficiently large, depending only on d, and L is allowed to vary. Note that in this case the number of parameters of the network $n(W_0, L) \asymp W_0^2 L$. The following theorem gives bounds on the VC dimension of such networks.

Theorem 3.12. Let W_0 be fixed and sufficiently large depending only on d. There are fixed constants c_1, C_1, depending only on d, such that

$$c_1 L^2 \leq \mathrm{VC}(\Upsilon^{W_0,L}(\mathrm{ReLU}; d, 1)) \leq C_1 L^2. \tag{3.27}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

The upper bound in this theorem follows from Theorem 8 of Bartlett *et al.* (2019). The remainder of this section will provide a proof of the lower bound in a form which will be used later in this paper to prove certain approximation results. Related lower bounds are stated in Theorem 3 of Bartlett *et al.* (2019).

3.6.3. Bit extraction using ReLU networks

We discuss in detail a very specific way to prove the lower bound in Theorem 3.12. This particular construction, called *bit extraction*, is useful in proving upper bounds for approximation using deep neural networks. For a fixed W_0 and C, depending only on d, the set $\Upsilon^{W_0,Cn}(\mathrm{ReLU}; d, 1)$ not only shatters $N = n^2$ equally spaced points $x^{(1)}, \dots, x^{(N)} \in \Omega := [0,1]^d$, but for certain bit data y_j, it contains an S such that $S(x^{(j)}) = y_j$, $j = 1, \dots, N$.

In order to avoid certain technicalities, we present this result only in the case $d = 1$. The full implementation for $d \geq 2$ can be found in Yarotsky (2018) and Shen, Yang and Zhang (2019).

Theorem 3.13. Let $N := n^2$ with $n \geq 4$ be an even integer. Define $t_i := i/N$, $i = 0, 1, \dots, N$, and consider any data y_i, $i = 0, \dots, N$, with the properties:

(i) $y_{jn} = 0$, $j = 0, \dots, n$,

(ii) $y_{i+1} = y_i + \varepsilon_i$, with $\varepsilon_i \in \{-1, 1\}$ for all $i = 0, \dots, N-1$.

Then there is an $S \in \Upsilon^{11,15n+2}(\mathrm{ReLU}; 1, 1)$, such that

$$S(t_i) = y_i, \quad i = 0, \dots, N.$$

Moreover, we have

$$|S(t) - S(t_i)| \leq 1, \quad t \in [t_i, t_{i+1}], \quad i = 0, 1, \dots, N-1. \tag{3.28}$$

The novelty in this theorem is that while the number of parameters in the NN is Cn, the number of data points is $N + 1 = n^2 + 1$. The theorem provides the lower bound in (3.27) for the VC dimension. Indeed, at any point t_{2i} not of the form t_{jn}, we can assign any data $y_{2i} \in \{0, 2\}$ because of property (ii). Thus we can shatter these points using the set $\Upsilon^{11,15n+2}(\mathrm{ReLU}; 1, 1)$. Since there are at least cn^2 such points, with c an absolute constant, we have the lower bound in (3.27) in the case $d = 1$ with $W_0 = 11$. The general case of $d > 1$ also easily follows from this by using the method of proof used in Proposition 3.10.

Before we present the proof of Theorem 3.13, which is a bit laborious, we introduce some notation, make several observations, and present the general idea of the proof.

First, note that for each $i = 0, 1, \dots, N-1$, there is a unique representation

$$t_i = \frac{i}{N} = \frac{j(i)}{n} + \frac{k(i)}{N}, \quad j(i), k(i) \in \{0, 1, \dots, n-1\}.$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Next, recall that any $t \in [-1, 1]$ can be represented as

$$t = \sum_{k=1}^{\infty} B_k(t) 2^{-k},$$

where the bits $B_k(t) \in \{-1, 1\}$ of t are found using the familiar quantizer function

$$Q := -\chi_{[-1,0]} + \chi_{(0,1]},$$

with χ_I denoting the characteristic function of a set I. The first bit of t satisfies $B_1(t) = Q(t)$ and has the residual $R_1(t) := 2t - B_1(t) \in [-1, 1]$. We find the later bits and residuals recursively as

$$B_j(t) = Q(R_{j-1}(t)), \quad R_j(t) := 2R_{j-1}(t) - B_j(t), \quad j = 2, 3, \dots. \tag{3.29}$$

Given our assigned bit sequence $\{\varepsilon_i\}$, $i = 0, \dots, N-1$, available to us from the values y_i, $i = 0, 1, \dots, N$, we define the numbers

$$Y_j := \sum_{k=0}^{n-1} \varepsilon_{jn+k} 2^{-k-1}, \quad j = 0, \dots, n-1. \tag{3.30}$$

Note that

$$Y_j \in [-1 + 2^{-n}, -2^{-n}] \cup [2^{-n}, 1 - 2^{-n}] \subset [-1, 1],$$

and the bits $B_\nu(Y_j) = \varepsilon_{jn+\nu-1}$, $\nu = 1, \dots, n$.

The idea of proving Theorem 3.13 is to produce a function S from the set $\Upsilon^{11,15n+2}(\mathrm{ReLU}; 1, 1)$, such that for each $i = 1, \dots N$, $i = j(i)n + k(i)$,

$$\begin{aligned} S(t_i) = y_i = \sum_{\nu=1}^{k(i)} \varepsilon_{j(i)n+\nu-1} = \sum_{\nu=1}^{k(i)} B_\nu(Y_{j(i)}), \quad k(i) = 1, \dots, n-1, \\ S(t_{jn}) = y_{jn} = 0, \quad j = 0, \dots, n, \end{aligned} \tag{3.31}$$

that in addition satisfies (3.28). We construct S by showing that each of the functions

$$t_i = \frac{i}{N} = \frac{j(i)}{n} + \frac{k(i)}{N} \mapsto j(i), k(i), Y_{j(i)}, \chi_{\{\nu:\, \nu \le k(i)\}}, \quad Y_j \mapsto B_\nu(Y_j),$$

are outputs of ReLU networks of an appropriate size.

To do this, let $\delta = 2^{-N}$ and define the following:

- The CPwL function $J = J_N$ which has breakpoints at each of the points

$$\xi_j := j/n, \quad \xi_j' := (j+1)/n - \delta, \quad j = 0, \dots, n-1, \tag{3.32}$$

and no other breakpoints, and takes the value j on the interval $[\xi_j, \xi_j']$. We also require $J(1) = n$. Note that J has the property

$$J(t_i) = j(i), \quad i = 0, 1, \dots, N-1.$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

- The CPwL function $K(t) = K_N(t) := J(nt - J(t))$. Observe that the key property of K is

$$K(t_i) = k(i), \quad i = 0, 1, \dots, N-1. \tag{3.33}$$

Next we would like to implement quantization by a neural network. However, the function Q is not continuous, so we cannot exactly reproduce Q. Instead, we use a surrogate

$$\hat{Q}(t) = -1 + \left(\frac{1}{\delta}t + 1\right)_+ - \left(\frac{1}{\delta}t - 1\right)_+, \tag{3.34}$$

where $\delta := 2^{-N}$. The surrogate $\hat{Q}$ is in $\Upsilon^{2,1}(\mathrm{ReLU}; 1, 1)$ and coincides with Q on $[-1, 1] \setminus [-\delta, \delta]$.

We define the surrogate bits $\hat{B}_\nu(t)$ for $t \in [-1, 1]$ by using $\hat{Q}$ in place of Q in the recursive definition of B_ν, described in (3.29). Because of the choice of δ, $\hat{B}_\nu$ can be used in place of B_ν to compute the bits of t whenever t has the representation

$$t = \sum_{\nu=1}^{k} B_\nu(t) 2^{-\nu}, \quad \text{with } k \le N - 1. \tag{3.35}$$

For such a t, we have $\hat{B}_\nu(t) = B_\nu(t)$, $\nu = 1, \dots, N-1$.

Finally, we introduce the following:

- The CPwL function Y, which has exactly the same breakpoints as J (see (3.32)) and satisfies

$$Y(\xi_j) = Y(\xi_j') = Y_j, \quad j = 0, 1, \dots, n-1, \tag{3.36}$$

with Y_j defined in (3.30) and $Y(1) = 0$.

The function S will be the output of a special neural network of width $W = 11$ and depth $L = 15n + 2$, which is a concatenation of five special networks that we describe below. The top channel of each of these networks is a source channel which simply passes forward the input t. Some of the other channels are collation channels and are occupied by zeros in their first layers so that they can be used later for passing forward certain function values.

We want to point out that our construction is probably not optimal in the sense that it does not provide an NN with the best possible minimal width and depth that outputs S. In addition, some of the channels in our NN are ReLU-free. We discussed earlier how we can construct a true ReLU network with the same outputs as a network that has ReLU-free nodes: see (3.10).

In going further, we note that any CPwL function T with k breakpoints is in $\Upsilon^{3,k}(\mathrm{ReLU}; 1, 1)$, where the network used to output T has one source channel, one computational channel and one collation channel that collects the successive terms we have computed; see the construction in Proposition 3.9.

Proof of Theorem 3.13. We can now give the proof of Theorem 3.13. The network $\mathcal{N}$ which outputs the function S of the theorem is a concatenation of five special

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

networks $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4, \mathcal{N}_5$. Each of them has a source channel as its first channel. It pushes forward the input $t \in [0, 1]$. The first of these networks outputs $K(t)$, the second outputs $Y(t)$ and the third takes input $K(t)$ and $Y(t)$ and outputs a CPwL function $\tilde{S}$ which almost satisfies the theorem. Namely, it satisfies the interpolation conditions and it also satisfies (3.28) except for a small set of t values. The last two networks make a technical correction to $\tilde{S}$ to obtain the desired S which satisfies (3.28) for all $t \in [0, 1]$. We now describe these five networks. All of them have width at most 11 and we make the width exactly 11 by adding zero channels. The depth of each network is also controlled so that the final network has depth $L = 15n + 2$.

First NN. This network, which we denote by $\mathcal{N}_1$, has depth $4n - 2$ and for any input $t \in [0, 1]$ outputs the function value $K(t)$. From our remarks on interpolation (see Proposition 3.9), we know that $J(t)$ is the output of a special ReLU network $\mathcal{N}_0$ of width $W = 3$ and depth $2n - 1$, where channel 3 is a collation channel. The CPwL function K is the output of a ReLU network $\mathcal{N}_1$ of width $W = 3$ and depth $4n - 2$, which is obtained by concatenating the network $\mathcal{N}_0$ for J with itself and using $nt - J(t)$ as the input to the second of these networks. Channel 3 is a collation channel, used first to build $J(t)$. Once $J(t)$ is computed, it sends this value as an input to the $(2n)$th layer. Then it is zeroed out by assigning a weight 0, and is subsequently used as a collation channel to build $K(t)$. It follows from (3.33) that the output of this network is $k(i)$ when the input is t_i. We add eight other channels with zero parameters. These channels will be used later.

Second NN. The second network $\mathcal{N}_2$ takes the input t from channel 1 and outputs the CPwL function $Y(t)$ which belongs to $\Upsilon^{3,2n-1}(\text{ReLU}; 1, 1)$. This network has depth $2n - 1$ and only needs three channels, but we augment it with eight more channels. After a concatenation with the existing network $\mathcal{N}_1$, it uses channel 2 to compute the terms involved in $Y(t)$, while channels 3 and 4 push forward the values $K(t)$ and the terms involved in $Y(t)$, respectively. Channels 5–11 have all parameters zero. Note that after this concatenation, we have available to us as outputs t, coming from the source channel, $K(t)$, kept in channel 3, and $Y(t)$ kept in channel 4.

Third NN. This network takes as inputs $K(t), Y(t)$ and outputs a function $\tilde{S}$ which satisfies the interpolation conditions and coincides with the desired S except for a small subset of $[0, 1]$. To describe this network, we shall use the CPwL function T with breakpoints $-1, 1, 2$, defined as

$$T(t) = \begin{cases} -1 & t \le -1, \\ t & -1 \le t \le 1, \\ 2 - t & 1 \le t \le 2, \\ 0 & t \ge 2. \end{cases}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Since $T(t) = -1 + (t+1)_+ - 2(t-1)_+ + (t-2)_+$, it belongs to $\Upsilon^{3,1}(\mathrm{ReLU}; 1, 1)$. Note that T is the identity on the interval $[-1, 1]$ and has the important property that for $t \in [-1, 1]$ and for each $1 \le k < n$ we have

$$T(B_\nu(t) + 3(\nu - k)_+) = B_\nu(t), \quad 1 \le \nu \le k, \tag{3.37}$$

and is zero otherwise, since $3(\nu - k)_+ = 0$ when $\nu \le k$ and $3(\nu - k)_+ \ge 3$ when $\nu > k$. It follows from (3.37) that

$$\sum_{\nu=1}^{n} T(B_\nu(t) + 3(\nu - k)_+) = \sum_{\nu=1}^{k} B_\nu(t).$$

Now, for $i = 0, \dots, N-1$, consider one of our points t_i which is not a multiple of n, *i.e.* $k(i) \ne 0$. Then $Y(t_i) = Y_{j(i)}$, $K(t_i) = k(i)$, and

$$\sum_{\nu=1}^{n} T(B_\nu(Y(t_i)) + 3(\nu - K(t_i))_+) = \sum_{\nu=1}^{k(i)} B_\nu(Y_{j(i)}) = \sum_{\nu=1}^{k(i)} \varepsilon_{j(i)n+\nu-1} = y_i.$$

Since we cannot produce B_ν with a ReLU network, we use the surrogate $\hat{B}_\nu$ in its place. This leads us to define the following function:

$$\tilde{S}(t) := \sum_{\nu=1}^{n} T(\hat{B}_\nu(Y(t)) + 3(\nu - K(t))_+), \quad t \in [0, 1]. \tag{3.38}$$

This function satisfies the interpolation conditions (3.31) since $\hat{B}_\nu(Y(t)) = B_\nu(Y(t))$, $\nu = 1, \dots, n$, whenever t is one of the points t_i, $i = 0, \dots, N$, where interpolation is to take place. In addition, since for $j = 0, \dots, n-1$, $K(t_{jn}) = 0$ and $K(1) = J(0) = 0$, we have

$$\tilde{S}(t_{jn}) = \sum_{\nu=1}^{n} T(\hat{B}_\nu(Y(t_{jn})) + 3\nu) = 0, \quad j = 0, \dots, n,$$

because of the definition of T.

Next we describe how $\tilde{S}$ is an output of a ReLU network $\mathcal{N}_3$ with inputs $Y(t), K(t)$. The network $\mathcal{N}_3$ is organized as follows. Channel 1 is left to be a source channel that forwards the value of t. Channels 2 and 3 are occupied with the values of $K(t)$ and $Y(t)$, forwarded to the next layers. Channel 4 computes $(\nu - K(t))_+$ in layer ν, for $\nu = 1, \dots, n$. Channel 5 computes the residual $R_{\nu-1}(Y(t))$ in layer ν, $\nu = 1, \dots, n$ (this is a ReLU-free channel), where $R_0(Y(t)) = Y(t)$, and $R_\nu(Y(t)) = 2R_{\nu-1}(Y(t)) - \hat{B}_\nu(Y(t))$, $\nu = 1, \dots, n$. Channels 6 and 7 implement the network for $\hat{Q}$ and compute consecutively $\hat{B}_1(Y(t)) = \hat{Q}(Y(t))$, $\hat{B}_\nu = \hat{Q}(2R_{\nu-2}(Y(t)) - \hat{B}_{\nu-1}(Y(t)))$, for $\nu = 2, \dots, n$. Channels 8, 9 and 10 implement T. Channel 11 successively adds the T values in the sum (3.38), and therefore $\mathcal{N}_3$ outputs $\tilde{S}$. In total, the entire network $\mathcal{N}_3$ has $(n+1)$ layers and width 11.

We have already observed that $\tilde{S}(t_i) = y_i$ and so the interpolation conditions are satisfied. The reader can imagine that we can take a max and min with an upper

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

and lower CPwL to obtain the control (3.28). The network $\mathcal{N}_4$ will do precisely that. So the remainder of the proof is to give one such construction.

We claim that the output $\tilde{S}$ of $\mathcal{N}_3$ already satisfies the inequalities

$$|\tilde{S}(t) - \tilde{S}(t_i)| \le 1, \quad t \in [t_i, t_{i+1}) \cap \Omega_N, \quad i = 0, 1, \dots, N-1, \tag{3.39}$$

where

$$\Omega_N := [0,1] \setminus \bigcup_{j=1}^{n} (t_{jn} - \delta, t_{jn}), \quad \delta := 2^{-N}.$$

We verify this property when $t \in [0, 1/n) \cap \Omega_N = [0, 1/n - \delta]$ since the verification on the intervals $[j/n, (j+1)/n) \cap \Omega_N$, $j = 1, \dots, n-1$, is the same. For $t \in [0, 1/n - \delta] = [0, t_n - \delta]$ we have $Y(t) = Y_0$ (see (3.36)), and therefore for $\nu = 1, \dots, n$, $\hat{B}_\nu(Y(t)) = \hat{B}_\nu(Y_0) = B_\nu(Y_0) = \varepsilon_{\nu-1}$. Thus (see (3.38)) we have

$$\tilde{S}(t) = \sum_{\nu=1}^{n} T(\varepsilon_{\nu-1} + 3(\nu - K(t))_+), \quad t \in [0, t_n - \delta].$$

We consider the following cases:

- If $t \in [0, t_1 - \delta]$, then $K(t) = 0$ and
$$\tilde{S}(t) = \sum_{\nu=1}^{n} T(\varepsilon_{\nu-1} + 3\nu) = 0, \tag{3.40}$$
and thus (3.39) is satisfied for these t.
- If $t \in [t_k, t_{k+1} - \delta] \subset [0, t_n - \delta]$, with $1 \le k < n$, then $K(t) = k$ and
$$\tilde{S}(t) = \sum_{\nu=1}^{k} \varepsilon_{\nu-1} = y_k, \tag{3.41}$$
and thus (3.39) is satisfied again.
- If $t \in (t_k - \delta, t_k)$, $1 \le k \le n-1$, then $k - 1 \le K(t) < k$ and
$$3(\nu - K(t))_+ \text{ is } \begin{cases} = 0 & \nu \le k-1, \\ \le 3 & \nu = k, \\ > 3 & \nu \ge k+1. \end{cases}$$
It follows from the definition of T that
$$\begin{aligned} \tilde{S}(t) &= \sum_{\nu=1}^{n} T(\varepsilon_{\nu-1} + 3(\nu - K(t))_+) \\ &= \sum_{\nu=1}^{k-1} T(\varepsilon_{\nu-1}) + T(\varepsilon_{k-1} + 3(k - K(t))_+) \\ &= y_{k-1} + \eta, \end{aligned}$$
with $|\eta| \le 1$, and therefore (3.39) is satisfied in this case as well.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

In summary, the function $\tilde{S}$ satisfies the properties we want except for control on the small intervals that make up the complement of Ω_N. We do not have a bound for $\tilde{S}$ on these small intervals. Our last construction will be to take care of these intervals while leaving $\tilde{S}$ unchanged elsewhere.

To see how to do this, we concentrate on the interval $[t_{(j-1)n}, t_{jn}]$, for $j = 1, \dots, n$, and let $I_j := (t_{jn} - \delta, t_{jn}]$. We know from our analysis that $\tilde{S}(t) = y_{jn-1} = \eta_j$ for $t \in [t_{jn-1}, t_{jn} - \delta]$, where $\eta_j = \pm 1$. Assume for now that $\eta_j = 1$. Also, recall that $\tilde{S}(t) = 0$ for $t \in [t_{(j-1)n}, t_{(j-1)n+1} - \delta]$. If $M := \|\tilde{S}\|_{C(\Omega_N)} \geq 1$, we consider the CPwL function U_j whose graph passes through the points $(t_{(j-1)n}, 0)$, $(t_{(j-1)n+1} - \delta, M)$, (t_{jn-1}, M), $(t_{jn} - \delta, 1)$ and $(t_{jn}, 0)$, and is otherwise linear between these points. Then $U_j(t) \geq \tilde{S}(t)$ on $[t_{(j-1)n}, t_{jn} - \delta]$, and thus on $[t_{(j-1)n}, t_{jn}] \setminus I_j$ the function $\min\{\tilde{S}, U_j\} = \tilde{S}$. In addition, $\min\{\tilde{S}, U_j\}$ will have values between 0 and 1 on I_j. This is the correction we want on I_j. We then define $U := \sum_{j \in \Lambda_+} U_j \chi_{[t_{(j-1)n}, t_{jn}]}$, where Λ_+ is the set of j's for which $\eta_j = +1$. In a similar way we define a lower envelope $\hat{U}$ for the j's such that $\eta_j = -1$. We can then take

$$S := \max\{\min\{\tilde{S}, U\}, \hat{U}\}.$$

The function S satisfies the conclusions of the theorem and we only have to see how it is output by a suitable neural network. Each of the functions $U, \hat{U}$ have at most $4n + 1$ breakpoints and hence are in $\Upsilon^{3,4n+1}(\mathrm{ReLU}; 1, 1)$.

Fourth and fifth NNs. These are the networks $\mathcal{N}_4$ and $\mathcal{N}_5$ outputting U and $\hat{U}$. We augment them with collation channels so that they have width 11. Since they already have a source channel (channel 1), there is no need to add such a channel.

The network $\mathcal{N}$. We use a construction similar to the one in MM4 to output S by concatenating the networks for $\tilde{S}$, U and $\hat{U}$. Following the construction in MM4, we end up with a network with width $W = 11$ (the same as the one for $\tilde{S}$) and depth $L = 15n + 2$ where we added the depth $7n - 2$ of the network for $\tilde{S}$, the depths of the network for U and $\hat{U}$, each of which is $4n + 1$, and two more layers to perform the min and max. This completes the proof of the theorem. □

Remark 3.1. We make some final remarks on the above construction. We have used the fact that the t_i's are $N = n^2 + 1$ equally spaced points. It would be interesting and useful to clarify for which other patterns of univariate points the construction can be done. We know that we cannot increase the number of points significantly because of the upper bound in (3.27). Note that in the case $d > 1$, we can construct a similar interpolant for points $x^{(i)} \in \mathbb{R}^d$ if there is a $v \in \mathbb{R}^d$ such that $v \cdot x^{(i)} = t_i$, $i = 1, \dots, N$. Again, it would be useful to know exactly when we can interpolate certain patterns of values like the y_i's with Cn^2 points from $\mathbb{R}^d$. The constructions in Yarotsky (2018) and Shen *et al.* (2019) show that this is possible on equally spaced tensor product grids.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

4. Classical model classes: smoothness spaces

In order to prove anything quantitative about the rate of approximation of a given target function f, one obviously needs to assume something about f. Such assumptions are referred to as *model class assumptions*. We say that a set K in a Banach space X is a *model class* of X if K is compact in X. The classical model classes for multivariate functions are the unit balls of smoothness spaces such as Lipschitz, Hölder, Sobolev and Besov spaces. We give a brief (mostly heuristic) review of these spaces in this section. A detailed development of these spaces can be found in the standard references; see *e.g.* Adams and Fournier (2003), Peetre (1976), Stein (1970), DeVore and Sharpley (1993) and DeVore (1998).

We consider these spaces on the domain $\Omega := [0,1]^d$. All definitions and properties extend to more general domains such as Lipschitz domains in $\mathbb{R}^d$. We use standard multivariate notation.

4.1. L_p spaces

As a starting point, we recall that the $L_p(\Omega)$ spaces consist of all Lebesgue measurable functions f for which $|f|^p$ is integrable. We define

$$\|f\|_{L_p(\Omega)} := \left(\int_\Omega |f(x)|^p \, \mathrm{d}x \right)^{1/p}, \quad 0 < p < \infty.$$

This is a norm when $1 \le p < \infty$ and a quasi-norm when $0 < p < 1$. When $p = \infty$, one usually takes $X = C(\Omega)$, the space of continuous functions on Ω with the uniform norm

$$\|f\|_{C(\Omega)} := \sup_{x \in \Omega} |f(x)|.$$

However, on occasion we need the space $L_\infty(\Omega)$ consisting of all functions that are essentially bounded on Ω with

$$\|f\|_{L_\infty(\Omega)} := \operatorname{ess\,sup}_{x \in \Omega} |f(x)|.$$

We assume throughout that the reader is familiar with the standard properties of these spaces.

4.2. Sobolev spaces

We begin by defining smoothness spaces of continuous functions. If r is a positive integer then $C^r := C^r(\Omega)$, $\Omega = [0,1]^d$, is the set of all continuous functions f defined on Ω, which have classical derivatives $D^\alpha f$ for all α with $|\alpha| = r$, where $|\alpha| := \sum_{j=1}^d |\alpha_j| = r$. We equip this space with the semi-norm

$$|f|_{C^r} := |f|_{C^r(\Omega)} := \max_{|\alpha|=r} \|D^\alpha f\|_{C(\Omega)}.$$

A norm on this space is given by $\|f\|_{C^r(\Omega)} := |f|_{C^r(\Omega)} + \|f\|_{C(\Omega)}$.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

The Sobolev spaces (of integer order) generalize the spaces C^r by imposing weaker assumptions on the derivatives $D^\alpha f$. First the notion of weak (or distributional) derivatives $D^\alpha f$ is introduced in place of classical derivatives. Then, for any $1 \le p \le \infty$, the Sobolev space $W^r(L_p(\Omega))$ is defined as the set of all $f \in L_p(\Omega)$ such that $D^\alpha f \in L_p(\Omega)$ for all $|\alpha| = r$. We equip this space with the semi-norm

$$|f|_{W^r(L_p(\Omega))} := \max_{|\alpha|=r} \|D^\alpha f\|_{L_p(\Omega)},$$

and obtain a norm on this space by $\|f\|_{W^r(L_p(\Omega))} := |f|_{W^r(L_p(\Omega))} + \|f\|_{L_p(\Omega)}$.

4.3. Besov spaces

The Sobolev spaces above are not sufficient because they only classify smoothness for integer values r. There is a long history of introducing smoothness spaces for any order $s > 0$. This began with Lipschitz and Hölder spaces and culminated with the Besov spaces that we define in this section.

Given a function $f \in L_p(\Omega)$, $0 < p \le \infty$, and any integer r, we define its modulus of smoothness of order r as

$$\omega_r(f,t)_p := \sup_{0<|h|\le t} \|\Delta_h^r(f,\cdot)\|_{L_p(\Omega)}, \quad t > 0,$$

where $h \in \mathbb{R}^d$ and $|h|$ is its Euclidean norm. Here Δ_h^r is the rth difference operator, defined by

$$\Delta_h^r(f,x) := \sum_{k=0}^{r} (-1)^{r-k} \binom{r}{k} f(x+kh), \quad x \in \Omega \subset \mathbb{R}^d,$$

where this difference is set to zero whenever one of the points $x + kh$ is not in Ω. It is easy to see that for any $f \in L_p(\Omega)$ we have $\omega_r(f,t)_p \to 0$ when $t \to 0$. How fast this modulus tends to zero with t measures the L_p smoothness of f.

For example, the Lipschitz space $\mathrm{Lip}(\alpha, p)$ for $0 < \alpha \le 1$ and $0 < p \le \infty$ consist of those functions $f \in L_p(\Omega)$ for which

$$\omega_1(f,t)_p \le Mt^\alpha, \quad t > 0,$$

and the smallest M for which this holds is the semi-norm $|f|_{\mathrm{Lip}(\alpha,p)}$. Again, we obtain a norm on this space by simply adding $\|f\|_{L_p(\Omega)}$ to the semi-norm.

The Besov spaces generalize the measure of smoothness in two ways. They allow for r to be replaced by any $s > 0$ and they introduce a finer way to measure decay of the modulus as t tends to zero. This finer decay is controlled by a new parameter $0 < q \le \infty$.

If $f \in L_p(\Omega)$, $0 < p, q \le \infty$ and $s > 0$, the space $B_q^s(L_p(\Omega))$ is defined as the set of functions f for which

$$|f|_{B_q^s(L_p(\Omega))} := \|t^{-s}\omega_r(f,t)_p\|_{L_q((0,\infty),\,\mathrm{d}t/t)} < \infty, \quad \text{where } r := \lfloor s \rfloor + 1. \tag{4.1}$$

Note here that the L_q norm is taken with respect to the Haar measure $\mathrm{d}t/t$. The

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

case $q = \infty$ is simply the supremum norm over $t > 0$. The norm on this space is

$$\|f\|_{B^s_q(L_p(\Omega))} := |f|_{B^s_q(L_p(\Omega))} + \|f\|_{L_p(\Omega)}.$$

The Besov spaces are now a standard way of measuring smoothness. Functions in this space are said to have smoothness of order s in L_p with q giving a finer gradation of this smoothness. We mention without a proof a few of the properties of these spaces that are frequently used in analysis.

First, note that when $s \in (0, 1)$ and $q = \infty$, these spaces are the $\mathrm{Lip}(s, p)$ spaces. However, the space $B^1_\infty(L_p(\Omega))$ is not $\mathrm{Lip}(1, p)$ since ω_2 is used in place of ω_1 in the definition (4.1), thereby resulting in a slightly larger space. A second useful remark is that in (4.1) we could have used any $r > s$ and obtained the same space and an equivalent norm. When we insert q into the picture, the requirement for f to be in the space $B^s_q(L_p(\Omega))$ gets stronger as q gets smaller, namely, we have the following embeddings.

BE1. Let $0 < p \le \infty$. If $s > s'$ and $0 < q, q' \le \infty$ or $s = s'$ and $q \le q'$, we have $|f|_{B^{s'}_{q'}(L_p(\Omega))} \le C|f|_{B^s_q(L_p(\Omega))}$ with the constant C independent of f.

BE2. If $0 < p < p' \le \infty$ and $0 < q' \le q \le \infty$ then $|f|_{B^s_q(L_p(\Omega))} \le |f|_{B^s_{q'}(L_{p'}(\Omega))}$.

We also have the well-known Sobolev embeddings for Besov spaces.

BE3. Let $0 < p \le \infty$. For any $s > 0$ and $0 < q \le \infty$, we have that the unit ball $U(B^s_q(L_\tau(\Omega)))$, $0 < q \le \infty$, is a compact subset of $L_p(\Omega)$ whenever $s > d/\tau - d/p$.

There is a simple graphical way to describe these embeddings that we shall refer to in this paper. We use the upper right quadrant of $\mathbb{R}^2$ to graphically represent smoothness spaces. We can write any point in this quadrant as $(1/p, s)$ with $0 < p \le \infty$ and $s \ge 0$. We think of any such point as corresponding to a smoothness space with smoothness of order s measured in L_p. For example, the space $\mathrm{Lip}(\alpha, L_p)$ can be thought of as corresponding to the point $(1/p, \alpha)$, and all Besov spaces $B^s_q(L_p(\Omega))$, $0 < q \le \infty$, are identified with the same point $(1/p, s)$. In terms of this graphical description, given an $L_p(\Omega)$ space, the smoothness spaces embedded into $L_p(\Omega)$ are the ones that correspond to points $(1/\tau, s)$ that lie on or above the line with equation $s = d(1/\tau - 1/p)$. Those corresponding to points strictly above this line are compactly embedded. These embedding results are summarized in Figure 4.1.

4.3.1. Atomic decompositions

An often used fact about Besov spaces is that functions in these spaces can be described by certain so-called *atomic decompositions*. Historically, this began with the Littlewood–Paley decompositions; see Frazier, Jawerth and Weiss (1991). In the case of approximation by ReLU networks, the two most relevant decompositions are those using splines or wavelets. We discuss the case of spline decompositions. Details and proofs can be found, for example, in DeVore and Popov (1988).

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

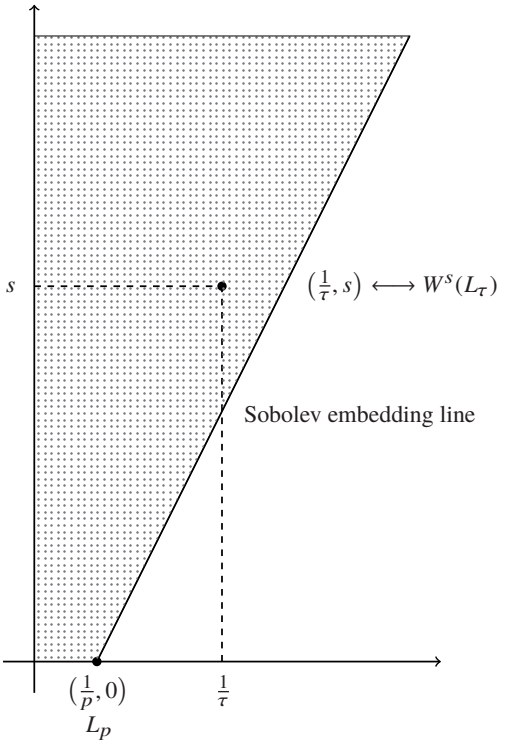

Figure 4.1. The Sobolev embedding theorem.

Let $r \geq 1$ be a positive integer and consider the univariate cardinal B-spline N_r of order r (degree $r-1$), which is defined by

$$N(t) := N_r(t) := \frac{r}{r!} \sum_{k=0}^{r} (-1)^{r-k} \binom{r}{k} (k-t)_+^{r-1}, \quad t \in \mathbb{R}. \tag{4.2}$$

The function N_r is a piecewise polynomial of degree $r-1$, is in $C^{r-2}(\mathbb{R})$, and is supported on $[0, r]$. With this normalization of the B-spline we have $\|N_r\|_{C(\mathbb{R})} \leq 1$.

The multivariate cardinal B-splines are defined as tensor products

$$N(x) := N_r(x_1, \ldots, x_d) := N_r(x_1) \cdots N_r(x_d), \quad x = (x_1, \ldots, x_d) \in \mathbb{R}^d. \tag{4.3}$$

We do not indicate the dependence on r when it is known from context. Let $\mathcal{D}$ denote the collection of dyadic cubes in $\mathbb{R}^d$ and let $\mathcal{D}_k$ denote the dyadic cubes of side length 2^{-k}. We also use the notation $\mathcal{D}_+ := \bigcup_{k \geq 0} \mathcal{D}_k$. If $I \in \mathcal{D}_k$ has smallest vertex $2^{-k} j$ with $j \in \mathbb{Z}^d$, we let

$$N_I(x) := N_{I,r}(x) := N(2^k x - j), \quad x \in \mathbb{R}^d. \tag{4.4}$$

The splines N_I provide an atomic decomposition for many function spaces and, in particular, the L_p, Sobolev and Besov spaces. Consider, for example, $\Omega = [0, 1]^d$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

and let $\mathcal{D}_k(\Omega)$ denote the set of those $I \in \mathcal{D}_k$ for which the support of N_I non-trivially intersects Ω. Then each $f \in L_1(\Omega)$ has a representation

$$f = \sum_{I \in \mathcal{D}_+(\Omega)} c_I(f) N_I, \tag{4.5}$$

where the c_I's are linear functionals on L_1, and $\mathcal{D}_+(\Omega) = \bigcup_{k \geq 0} \mathcal{D}_k(\Omega)$. The representation (4.5) is not unique since the N_I's are not linearly independent. However, we can fix the c_I's so that all properties stated below in this section are valid.

We can characterize membership of f in a Besov space $B^s_q(L_p(\Omega))$ in terms of the decomposition (4.5); see Corollary 5.3 in DeVore and Popov (1988). Namely, $f \in B^s_q(L_p(\Omega))$, $0 < s < \min\{r, r - 1 + 1/p\}$, and $0 < q, p \leq \infty$ if and only if f has the representation (4.5) with coefficients $c_I(f)$ satisfying

$$\|f\|'_{B^s_q(L_p(\Omega))} := \left\{ \sum_{k=0}^{\infty} 2^{skq} \left(\sum_{I \in \mathcal{D}_k(\Omega)} |c_I(f)|^p |I| \right)^{q/p} \right\}^{1/q} < \infty, \tag{4.6}$$

for $0 < q, p < \infty$, with the obvious modifications when either p or q is infinity. Moreover, $\|\cdot\|'$ is equivalent to the usual Besov norm. This fact is the starting point for proving many approximation theorems for functions in Besov spaces.

4.4. *Interpolation of operators*

Next we mention how, from known upper bounds for the approximation error on a model class, we can derive new upper bounds on a spectrum of new model classes by using results from the theory of interpolation of operators. We assume the reader is familiar with the rudiments of the theory of interpolation spaces via the real method of interpolation; see either Bergh and Lofstrom (1976) or Bennett and Sharpley (1990).

Given two Banach spaces X, Y with (for convenience) Y continuously embedded in X, the real method of interpolation generates a family of new Banach spaces $(X, Y)_{\theta,q}$, $0 < \theta < 1$, $0 < q \leq \infty$, which interpolate between them. These spaces are defined via what is called the K functional for the pair

$$K(f, t) := K(f, t; X, Y) := \inf_{g \in Y} \|f - g\|_X + t|g|_Y, \quad t > 0,$$

where $\|\cdot\|_X$ is the norm on X and $|\cdot|_Y$ is a semi-norm on Y.[1] The space $(X, Y)_{\theta,q}$, $0 < \theta < 1$, $0 < q \leq \infty$, then consists of all $f \in X$ such that

$$\|f\|_{(X,Y)_{\theta,q}} := \|t^{-\theta} K(f, t)\|_{L_q((0,\infty), \mathrm{d}t/t)} < \infty, \tag{4.7}$$

where the L_q norm is taken with respect to the Haar measure $\mathrm{d}t/t$. The important fact for us is that for classical pairs (X, Y) of spaces such as L_p and Besov/Sobolev spaces, the interpolation spaces are known and can be used to easily extend known

[1] When Y is not continuously embedded in X, we use $\|\cdot\|_Y$ in the definition of K.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

error estimates for approximation. We mention two typical approximation results. By $U(Y)$ we mean the unit ball of the space Y. In the following two statements, we let $\Sigma_n \subset X$ be any set with the property that if $s \in \Sigma_n$, then $cs \in \Sigma_n$ for every $c \in \mathbb{R}$.

Extend 1. If $\Sigma_n \subset X$ is a set that provides the approximation error

$$\inf_{S \in \Sigma_n} \sup_{f \in U(Y)} \|f - S\|_X =: E(U(Y), \Sigma_n)_X = \varepsilon_n,$$

then for the space $Z = (X, Y)_{\theta,q}$, $0 < \theta < 1$ and $0 < q \le \infty$ we have

$$E(U(Z), \Sigma_n)_X \le \varepsilon_n^\theta.$$

Extend 2. If, for the Banach spaces Y_0, Y_1 continuously embedded in X and the set $\Sigma_n \subset X$, we know that

$$E(U(Y_0), \Sigma_n)_X \le \varepsilon_n, \quad E(U(Y_1), \Sigma_n)_X \le \tilde{\varepsilon}_n,$$

then it follows that for $Z := (Y_0, Y_1)_{\theta,q}$, $0 < \theta < 1$ and $0 < q \le \infty$, we have

$$E(U(Z), \bar{\Sigma}_n)_X \le C \varepsilon_n^{1-\theta} \tilde{\varepsilon}_n^\theta,$$

where

$$\bar{\Sigma}_n := \{aS + bT : a, b \in \mathbb{R};\ S, T \in \Sigma_n\},$$

and C depends only on θ.

We prove only Extend 1 since the proof of Extend 2 is similar. We can take $q = \infty$ because this is the largest space Z for the given θ. If $f \in U(Z)$, for $t = \varepsilon_n$, there is a $g \in Y$ (if the infimum is not achieved, the proof follows from some limiting arguments) that satisfies

$$\|f - g\|_X + \varepsilon_n \|g\|_Y \le K(f, \varepsilon_n; X, Y) \le \varepsilon_n^\theta. \tag{4.8}$$

We know that there is an $S \in \Sigma_n$ which approximates g to accuracy $\varepsilon_n \|g\|_Y$. For this S, we have

$$\|f - S\|_X \le \|f - g\|_X + \|g - S\|_X \le K(f, \varepsilon_n; X, Y) \le \varepsilon_n^\theta. \tag{4.9}$$

Here is a simple but typical example of Extend 1. If we establish a bound ε_n for approximation of functions in $U(\mathrm{Lip}\,1)$ with error measured in $X = C(\Omega)$, then we automatically get the bound ϵ_n^α for approximating functions from $U(\mathrm{Lip}\,\alpha)$, $0 < \alpha < 1$, because $\mathrm{Lip}\,\alpha = (C(\Omega), \mathrm{Lip}\,1)_{\alpha,\infty}$, where $\mathrm{Lip}\,1 = \mathrm{Lip}(1, C(\Omega))$, and $\mathrm{Lip}\,\alpha = \mathrm{Lip}(\alpha, C(\Omega))$.

5. Evaluation of nonlinear methods of approximation

Before embarking on an analysis of the approximation performance of ReLU networks, we wish to place this type of approximation into the usual setting of approximation theory, and thereby draw out the type of questions that should be

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

answered. As we have noted, approximation using the outputs of neural networks with a fixed architecture is a form of nonlinear approximation known as manifold approximation. Given a target function f in a Banach space X, the approximation is given by $A_n(f) := M_n(a_n(f))$, where the two maps

$$a = a_n : X \to \mathbb{R}^n, \quad M = M_n : \mathbb{R}^n \to X,$$

select the n parameters of the network and output the approximation, respectively.

Of course, there are many methods of approximation. The question we address in this section is how we could possibly determine if approximation by NNs is in some quantifiable sense superior to other more traditional methods of approximation. Also, what are the inherent limits on the capacity of NNs to approximate, once the number n of parameters allocated to the approximation is fixed? To answer such questions, we introduce various traditional ways to compare approximation methods and say with certainty whether an approximation method is optimal among all methods of approximation, or perhaps among all approximation methods with a specified structure. How NNs do under such methods of comparison is not the subject of this section. That topic is dealt with in later sections of this paper.

To begin the discussion, we take the view that an *approximation method* is a sequence

$$\{0\} =: \Sigma_0 \subset \Sigma_1 \subset \cdots \subset \Sigma_n \subset \cdots \subset X$$

of nested sets to be used in approximating functions f from the Banach space X in the norm $\|\cdot\|_X$. Here n, in some sense, measures the complexity of Σ_n. The typical spaces X used in practice are the spaces $L_p(\Omega)$. However, at this point we let X be any Banach space of functions on Ω with a norm $\|\cdot\|_X$.

The various methods of approximation are divided into two general categories: linear and nonlinear. A method is said to be linear if, for each n, the set Σ_n is a linear space of dimension n, *i.e.* Σ_n is the linear span of n elements from X. The standard examples are spaces of polynomials, splines and wavelets. Note that the term linear does not refer to how the approximation depends on $f \in X$. It only refers to the structure of each Σ_n, $n \geq 0$. All other methods of approximation are referred to as nonlinear. For nonlinear methods, a linear combination of elements from Σ_n may not lie in Σ_n. There are three prominent examples of nonlinear approximation we wish to mention.

In the first, Σ_n consists of piecewise polynomials (of a fixed and generally small degree r) on a domain $\Omega \subset \mathbb{R}^d$. Let $\mathcal{P}_r$ denote the linear space of polynomials of degree r. Here we can use any notion of degree in d variables, such as coordinate degree or total degree. Given n, an element $S \in \Sigma_n$ is obtained by partitioning the domain Ω into n disjoint cells $C_j \subset \Omega$, $j = 1, \ldots, n$, and assigning a polynomial $P_j \in \mathcal{P}_r$ to each cell. Thus we have

$$S = \sum_{j=1}^{n} P_j \chi_{C_j},$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

where χ_{C_j} is the characteristic function of the cell C_j. The partitions are not fixed but allowed to vary within a class of partitions that can be described by n parameters. We have already seen an example of this in the case of free-knot CPwL functions of one variable, in which case the partition was allowed to consist of any n intervals. In the multivariate case, the allowable partitions are more structured and usually generated adaptively. The general idea of this form of approximation is to use small cells where the target function is rough and large cells where the function is smooth.

From our description of the sets $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$, we see that NN approximation fits into the above framework of piecewise polynomial approximation in the sense that each element in one of these sets is a CPwL function on a polytope partition; see Section 3. However, several notable distinctions arise. First of all, we have many fewer restrictions on the partitions that arise when compared to other piecewise polynomial methods of approximation such as FEMs, adaptive methods, free-knot splines, *etc.* Another important point is that $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ is not the collection of all CPwL functions subordinate to a fixed class of partitions. Here, choosing the parameters of the network specifies in tandem the partition and the CPwL. One view of how this is done is nicely explained in Balestriero and Baraniuk (2021).

Another widely used example of nonlinear approximation is *n-term approximation*. Let $B := \{\phi_j,\ j \geq 1\}$ be an unconditional basis for X. The set $\Sigma_n := \Sigma_n(B)$ in this case consists of all functions $S \in X$ which are a linear combination of at most n of these basis elements. Thus each $S \in \Sigma_n(B)$ takes the form

$$S = \sum_{k \in \Lambda} \alpha_k \phi_k, \quad \#(\Lambda) = n, \quad \alpha_k \in \mathbb{R},$$

where $\Lambda \subset \mathbb{N}$ is a subset of n indices. The index set Λ is allowed to change at each occurrence with the only restriction being that $\#(\Lambda) = n$. One can generalize this setting if B is replaced by a frame or, more generally, a dictionary. Typical examples used in numerical analysis and signal/image processing are dictionaries of wavelets, curvelets, ridge functions, shearlets and other families of waveforms. In this generality, n-term approximation is not numerically implementable because the dictionary is infinite. To circumvent this in practice, one uses a large but finite dictionary that is sufficiently rich for the problem at hand.

Neural network approximation fits most naturally into a third type of nonlinear approximation known as *manifold approximation*. In manifold approximation, the elements of $S \in \Sigma_n$ take the form

$$S = M_n(y), \quad y \in \mathbb{R}^n,$$

where $M_n \colon \mathbb{R}^n \to X$, *i.e.* $\Sigma_n = \{M_n(y) \colon y \in \mathbb{R}^n\}$. As noted earlier, a numerical implementation of manifold approximation is made by specifying a mapping $a_n \colon K \to \mathbb{R}^n$ which, when presented with $f \in K$, describes the parameters of the point on the manifold used to approximate f.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Given an approximation method $\Sigma := (\Sigma_n)_{n\geq 0}$ and $f \in X$, we let

$$E_n(f)_X := E(f, \Sigma_n)_X := E_n(f, \Sigma)_X := \inf_{g\in\Sigma_n} \|f - g\|_X$$

denote the *error of approximation of f by elements from Σ_n*. Note that $E_n(f)_X$ gives the smallest error we can achieve using Σ_n to do the approximation, but it does not address the question of how to find such a best or near-best approximation. This is an important issue, especially for NN approximation, that we address later in this section.

An oft-quoted property of NNs is their *universality*, which means that $E_n(f)_X \to 0$ as $n \to \infty$, for all $f \in X$. This is a property possessed by all approximation methods used in numerical analysis. Universality is not at all special and certainly cannot be used to explain the success of NNs.

5.1. *Approximation of model classes*

We do not measure the performance of an approximation method on a single function f but rather on a class $K \subset X$ of functions. In this case we have the *class error*

$$E_n(K)_X := E(K, \Sigma_n)_X := E_n(K, \Sigma)_X := \sup_{f\in K} E_n(f, \Sigma)_X, \quad n \geq 0. \tag{5.1}$$

Here K incorporates the knowledge we have about the function or potential functions f that we are trying to capture. For example, when numerically solving a PDE, K is typically provided by a regularity theorem for the PDE. In the case of signal processing, K summarizes what is known or assumed about the underlying signal, such as band limits or sparsity.

Note that $E_n(K)_X$ represents the worst-case error. It is also possible to measure error in some averaged sense. This would be meaningful, for example, when the set K is given by a stochastic process with some underlying probability measure. For now, we discuss only the worst-case error.

A set K on which we wish to measure the performance of an approximation method is called a *model class*. We always assume that K is a compact subset of X. If the approximation process is universal, then $E_n(K)_X \to 0$ as $n \to \infty$ for every model class K. How fast it tends to zero represents how good the sets $(\Sigma_n)_{n\geq 0}$ are for approximating the elements of K.

If we are presented with approximation processes given by $\Sigma = (\Sigma_n)_{n\geq 0}$ and $\Sigma' = (\Sigma'_n)_{n\geq 0}$ respectively, then given a model class K, we can compare the performance of these methods on K by checking the decay of $E_n(K, \Sigma)_X$ and $E_n(K, \Sigma')_X$ as $n \to \infty$. If the decay rate of $E_n(K, \Sigma)_X$ is faster than that of $E_n(K, \Sigma')_X$ as $n \to \infty$, we are tempted to say that Σ is superior to Σ' at least on this model class. However, the question of the computability of the approximant is an important issue and has to be taken into consideration.

To drive home this latter point, the following example is germane. Given a compact set $K \subset X$ and $\varepsilon > 0$, let $S_1 = S_1(\varepsilon)$ be a finite subset of K such that

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

$\operatorname{dist}(f, S_1)_X \leq \varepsilon$ for all $f \in K$. For example, S_1 could be the set of centres of an ε covering of K. Going further, we can find a one-dimensional manifold Σ_1 that is parametrized by $t \in [0,1]$ and passes through each point in S_1 as t runs through $[0,1]$, and thus $E(K,\Sigma_1)_X \leq \varepsilon$. The point of this simple observation is to emphasize that we must place some further restrictions on what we allow as an approximation method $(\Sigma_n)_{n\geq 0}$ so that we can have a meaningful theory. We next address the subject of what such restrictions should look like and what their implications are.

5.2. *Widths for measuring approximation error*

The concept of widths was introduced to quantify the best possible performance of approximation methods on a given model class K. The best known width is the Kolmogorov width, which was introduced to quantify the best possible approximation when using linear spaces. If X_n is a linear subspace of X of dimension n, then its performance in approximating the elements of the model class K is given by the error $E(K,X_n)_X$ defined in (5.1). If we fix the value of $n \geq 0$, the Kolmogorov n-width of K is defined as

$$d_0(K)_X := \sup_{f\in K} \|f\|_X, \quad d_n(K)_X := \inf_{\dim(Y)=n} E(K,Y)_X, \quad n \geq 1, \tag{5.2}$$

where the infimum is taken over all linear spaces $Y \subset X$ of dimension n. An n-dimensional space which achieves the infimum in (5.2) is called a Kolmogorov space for K if it exists.

The Kolmogorov n-width of a model class K tells us the optimal performance possible for approximating K using linear spaces of dimension n for the approximation. It does not tell us how to select a (near-) optimal space Y of dimension n for this purpose nor how to find a good/best approximation from Y once it is chosen. In recent years, discrete optimization methods have been discovered for finding optimal subspaces. They go by the name of greedy algorithms; see Buffa *et al.* (2012), Binev *et al.* (2011) and DeVore, Petrova and Wojtaszczyk (2013). If X is a Hilbert space and Y is a finite-dimensional subspace, then we can always find the best approximation from Y to a given $f \in X$ by orthogonal projections. This becomes a problem when X is a general Banach space because linear projections onto a general n-dimensional space Y may have large norm when n is large. Although a famous theorem of Kadec and Snobar says that there is always a projection with norm at most $\sqrt{n}$, finding such a projection is a numerical challenge. Also, projecting onto such a linear space does not give the best approximation from the space because the norm of the projection is large.

For classical model classes such as the finite ball in smoothness spaces such as Lipschitz, Sobolev or Besov spaces, the Kolmogorov widths are known asymptotically when X is an L_p space. Furthermore, it is often known that specific linear spaces of dimension n such as polynomials, splines on uniform partition, *etc.*,

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

achieve this optimal asymptotic performance (at least within reasonable constants). This can then be used to show that for such K, certain numerical methods, such as spectral methods or FEMs, are also asymptotically optimal among all possible choices of numerical methods built on using linear spaces of dimension n for the approximation.

Let us note that in the definition of Kolmogorov widths we do not require the mapping that sends $f \in K$ into the approximation to f to be a linear map. There is a concept of *linear width* which requires the linearity of the approximation map. Namely, given $n \geq 0$ and a model class $K \subset X$, its *linear width* $d_n^L(K)_X$ is defined as

$$d_0^L(K)_X := \sup_{f \in K} \|f\|_X, \quad d_n^L(K)_X := \inf_{L \in \mathcal{L}_n} \sup_{f \in K} \|f - L(f)\|_X, \quad n \geq 1, \tag{5.3}$$

where the infimum is taken over the class $\mathcal{L}_n$ of all linear maps from X into itself with rank at most n. The asymptotic decay of linear widths for classical smoothness classes is also known. We refer the reader to the books by Pinkus (2012) and Lorenz, Makovoz and von Golitschek (1996) for the fundamental results for Kolmogorov and linear widths. When X is not a Hilbert space, the linear width of K can decay worse than the Kolmogorov width.

Now we want to make a very important point. There is a general lower bound on the decay of Kolmogorov widths that was given by Carl (1981). This lower bound can be very useful in showing that a linear method of approximation is nearly optimal. To state this lower bound, we need to introduce the Kolmogorov *entropy* of a compact set K. Given $\varepsilon > 0$, compactness says that K can be covered by a finite number of balls of radius ε; see Figure 5.1. We define the covering number $N_\varepsilon(K)_X$ to be the smallest number of balls of radius ε that cover K, and we define the entropy $H_\varepsilon(K)_X$ of K to be the logarithm of this number

$$H_\varepsilon(K)_X := \log_2(N_\varepsilon(K)_X).$$

The entropy of K measures how compact the set K is, and is often used to give lower bounds on how well we can approximate the elements in K and also how well we can learn an element from K given data observations. The Kolmogorov entropy of a compact set is an important quantity for measuring optimality, not only in approximation theory and numerical analysis, but also in statistical estimation and encoding of signals and images.

To formulate the lower bounds for widths in terms of entropy, we introduce the related concept of entropy numbers. Given $n \geq 0$, we define the *entropy number* $\varepsilon_n(K)_X$ to be the infimum of all $\varepsilon > 0$ for which 2^n balls of radius ε cover K, that is,

$$\varepsilon_n(K)_X := \inf\{\varepsilon \colon N_\varepsilon(K)_X \leq 2^n\}.$$

The decay rate of entropy numbers for all classical smoothness spaces in $L_p(\Omega)$ are known.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

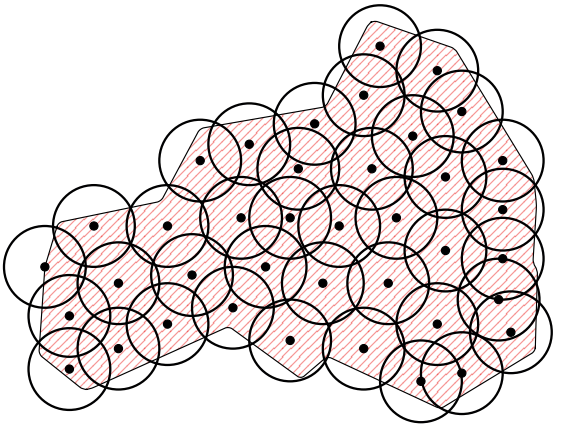

Figure 5.1. Kolmogorov covering of K.

Carl proved that for each $r > 0$, there is a constant C_r, depending only on r, such that

$$\Lambda := \sup_{m\geq 0}(m+1)^r d_m(K)_X < \infty \;\Rightarrow\; \varepsilon_n(K)_X \leq C_r\Lambda(n+1)^{-r}, \quad n \geq 0. \tag{5.4}$$

Thus, for polynomial decay rates for approximation by n-dimensional linear spaces, this decay rate cannot be better than the decay rate for the entropy numbers of K. Let us note that for many standard model classes K, such as finite balls in Sobolev and Besov spaces, the decay rate of $d_n(K)_X$ is much worse than $\varepsilon_n(K)_X$. A version of Carl's inequality holds for other decay rates, even exponential, and can be found in Cohen, DeVore, Petrova and Wojtaszczyk (2020).

5.3. Nonlinear widths

Since NN approximation is a nonlinear method of approximation, the Kolmogorov widths are not an appropriate measure of performance. Many different notions of nonlinear widths (see the discussion in DeVore, Kyriazis, Leviatan and Tikhomirov 1993) have been introduced to match the various forms of nonlinear approximation used in numerical computations. We shall discuss only nonlinear widths that match the form of approximation provided by NNs.

Recall that NN approximation is a form of manifold approximation, where the approximation set Σ_n consists of the outputs of a neural network with n parameters. Thus $\Sigma_n = M_n(\mathbb{R}^n)$, with M_n being the mapping that describes how the output function is constructed once the parameters and architecture of the NN are set. Note that Σ_n is a nonlinear set in the sense that the sum of two elements from Σ_n is generally not in Σ_n.

There are by now numerous papers that discuss the approximation by NNs. They typically provide estimates for $E(K,\Sigma_n)_X$ for certain model classes K. We will discuss such estimates subsequently in Sections 7 and 8. We have cautioned that

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

such results must be taken with a grain of salt since they do not typically discuss how the approximation would be found or numerically constructed. Our point of view is that it is not just an issue of how well Σ_n approximates K, although this is indeed an interesting question, but also how a good approximation would be found. In other words, the parameter selection mapping a_n is equally important.

When presented with an $f \in K$, one chooses the parameters of the NN to be used to construct the approximation to f. Typical algorithms in learning base this selection of parameters on some form of optimization, executed through gradient descent methods. For our analysis, we denote this selection procedure by the mapping $a_n : K \to \mathbb{R}^n$. So the approximation procedure is given by $A_n(f) := M_n(a_n(f))$. If we wish to establish some form of optimality of NNs, we should compare NN approximation with other approximation methods of this form.

Given any pair of mappings (not necessarily using NNs) $a : X \to \mathbb{R}^n$ and $M : \mathbb{R}^n \to X$, we define the *error for approximating* $f \in X$ by

$$E_{a,M}(f)_X := \|f - M(a(f))\|_X,$$

and the approximation error on a model class $K \subset X$ by

$$E_{a,M}(K)_X := \sup_{f \in K} E_{a,M}(f).$$

For any such a, M we have

$$E(K, \Sigma_n)_X \le E_{a,M}(K)_X, \quad \text{where } \Sigma_n := M(\mathbb{R}^n), \tag{5.5}$$

and we have equality when we choose $a(f)$ so that $M(a(f))$ is a best approximation to f (assuming such a best approximation exists) from Σ_n.

A first possibility for defining optimal performance of such methods of manifold approximation on a model class K would be to simply find the minimum of $E_{a,M}(K)_X$ over all such pairs of mappings. However, we have already pointed out that this minimum would always be zero (even when $n = 1$) because of the existence of space-filling manifolds. On the other hand, these space-filling manifolds are useless in numerical analysis. Consider, for example, a one-parameter space-filling manifold. By necessity, a small perturbation of the parameter will generally result in a large change in the output, which makes parameter selection for fitting f impossible. The natural question that arises is what restrictions need to be imposed on the mappings a, M so that we have a theory which corresponds to reasonable numerical methods. We discuss this next.

5.4. *Restrictions on a, M in manifold approximation*

The first suggestion, given in DeVore, Howard and Micchelli (1989), for the possible restrictions to place on the mappings a, M, was to require them to be continuous. This led to the following definition of *manifold* widths $\delta_n(K)_X$,

$$\delta_n(K)_X := \inf_{a_n, M_n} E_{a_n, M_n}(K)_X,$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

with the infimum taken over all maps $a_n : K \to \mathbb{R}^n$ and $M_n : \mathbb{R}^n \to X$, where a_n is continuous on K and M_n is continuous on $\mathbb{R}^n$. Manifold widths are closely connected to other definitions of nonlinear widths; see the discussion in DeVore *et al.* (1993).

It turns out that even with these very modest assumptions on the mappings a, M, one can prove lower bounds for $\delta_n(K)_X$ when K is a unit ball of a classical smoothness space, *e.g.* Besov, Sobolev or Lipschitz, and these lower bounds show that manifold approximation is no better than other methods of nonlinear approximation such as n-term wavelet approximation or adaptive finite element approximation for these model classes. For example, if we approximate in $L_p(\Omega)$, with $\Omega = [0,1]^d$, and K is a unit ball of any Besov space $B^s_q(L_p(\Omega))$ that embeds compactly into $L_p(\Omega)$, then it was shown in DeVore *et al.* (1993) that

$$\delta_n(K)_{L_p(\Omega)} \geq Cn^{-s/d}, \quad n \geq 1. \tag{5.6}$$

This should not be used to deduce that manifold approximation, in general, and NN approximation, in particular, offer nothing new in terms of their ability to approximate. It may be that their power to approximate lies in their ability to handle non-traditional model classes. Nevertheless, this should make us proceed with caution.

A stark criticism of manifold widths is that its requirement of continuity of the mappings is too minimal and does not correspond to the notions of numerical stability used in practice. In other words, manifold approximation based on just assuming that a, M are continuous may also not be implementable in a numerical setting. We next discuss what may be more viable restrictions on a, M that match numerical practice.

5.5. *Stable manifold widths*

A major issue in the implementation of a method of approximation is its stability, *i.e.* its sensitivity to computational error or noisy inputs. The stability we want can be summarized in the following two properties:

S1. When we input f into the algorithm, we often input a noisy discretization of f, which can be viewed as a perturbation of f. So we would like to have the property that when $\|f - g\|_X$ is small, the algorithm outputs $M(a(g))$ which is close to $M(a(f))$. A standard quantification of this is to require the mapping $A := M \circ a$ to be a Lipschitz mapping from K to X. Note that in this formulation the perturbation g should also be in K.

S2. In the numerical implementation of the algorithm, the parameters $a(f)$ are not computed exactly, so when $a, b \in \mathbb{R}^n$ are close to one another we would like $M(a)$ and $M(b)$ to be close as well. Again, the usual quantification of this observation is to impose that $M : \mathbb{R}^n \to X$ is a Lipschitz map. This property requires the specification of a norm on $\mathbb{R}^n$ which controls the size of the perturbation of a.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

The simplest way to guarantee S1–S2 is to require both mappings a, M to be Lipschitz, which means that there is a norm $\|\cdot\|_Y$ on $\mathbb{R}^n$ and a number $\gamma \geq 1$, such that

$$\begin{aligned} \|a(f) - a(g)\|_Y &\leq \gamma \|f - g\|_X, \qquad f, g \in K, \\ \|M(x) - M(x')\|_X &\leq \gamma \|x - x'\|_Y, \qquad x, x' \in \mathbb{R}^n. \end{aligned} \tag{5.7}$$

If a, M satisfy (5.7), then obviously S1 and S2 hold, where the Lipschitz constant in S1 is γ^2.

Imposing Lipschitz stability on a, M leads to the following definition of *stable manifold* widths,

$$\delta_n^*(K)_X := \delta_{n,\gamma}^*(K)_X := \inf_{a,M} E_{a,M}(K)_X,$$

where now the infimum is taken over all maps $a\colon K \to \mathbb{R}^n$ and $M\colon \mathbb{R}^n \to X$ that are Lipschitz-continuous with constant γ.

5.6. *Bounds for stable manifold widths*

Both upper and lower bounds for stable manifold widths of a compact set K are given in Cohen *et al.* (2020). These bounds are tight in the case when the approximation takes place in a Hilbert space. Approximation in a Hilbert space is often used in applications of NNs.

Lower bounds for the decay of stable manifold widths in a general Banach space X are given by the following Carl's type inequality (see (5.4)), which compares $\delta_{n,\gamma}^*(K)_X$ with the entropy numbers $\varepsilon_n(K)_X$. Specifically, for any $r > 0$ we have

$$\varepsilon_n(K)_X \leq C(r,\gamma)(n+1)^{-r} \sup_{m \geq 0} (m+1)^r \delta_{m,\gamma}^*(K)_X, \quad n \geq 0. \tag{5.8}$$

This shows that whenever the stable manifold widths $\delta_{n,\gamma}^*(K)_X$ of a model class K tend to zero like $O(n^{-r})$, $n \to \infty$, then the entropy numbers of K must have the same or faster rate of decay. Similar bounds are known when the decay rate n^{-r}, $n \to \infty$, is replaced by other decays; see Cohen *et al.* (2020). In this sense, the stable manifold widths $\delta_{n,\gamma}^*(K)_X$ cannot tend to zero faster than the entropy numbers of K.

The inequalities (5.8) give a bound for how well manifold approximation can perform on a model class K once Lipschitz stability of the maps a, M is imposed. One might speculate, however, that in general $\varepsilon_n(K)_X$ may go to zero faster than $\delta_{n,\gamma}^*(K)_X$. This is not the case when $X = H$ is a Hilbert space, since in that case, for any compact set $K \subset H$, we have the estimate

$$\delta_{26n,2}^*(K)_H \leq 3\varepsilon_n(K)_H, \quad n \geq 1, \tag{5.9}$$

proved in Cohen *et al.* (2020). This is very useful information since it is often relatively easy to compute the entropy numbers of a model class K. In addition, it is also a very useful result for our forthcoming analysis of NN approximation.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

The upper bound (5.9) is proved via three fundamental steps. The first is to select 2^n points $S_n := \{f_i\}_{i=1}^n$ from H such that the balls of radius $\varepsilon := \varepsilon_n(K)_H$ centred at these points cover K. The next step is to use the Johnson–Lindenstrauss dimension reduction lemma to find a (linear) mapping $a\colon S_n \to \mathbb{R}^{26n}$, for which

$$\frac{1}{2}\|f_i - f_j\|_H \le \|a(f_i) - a(f_j)\|_{\ell_2(\mathbb{R}^{26n})} \le \|f_i - f_j\|_H, \quad i, j = 1, \dots, 2^n.$$

According to the Kirszbraun extension theorem (see Theorem 1.12 from Benyamini and Lindenstrauss 2000), the mapping a can be extended from S_n to the whole H, preserving the Lipschitz constant 1. The last step is to define M on $a(f_j)$, $j = 1, \dots, 2^n$, as

$$M(a(f_j)) = f_j, \quad j = 1, \dots, 2^n.$$

Clearly

$$\|M(a(f_i)) - M(a(f_j))\|_H = \|f_i - f_j\|_H \le 2\|a(f_i) - a(f_j)\|_{\ell_2(\mathbb{R}^{26n})},$$

and therefore M is a Lipschitz map with a Lipschitz constant 2 when restricted to the finite set $a(S_n)$. Again, according to the Kirszbraun extension theorem, we can extend M to a Lipschitz map on the whole $\mathbb{R}^{26n}$ with the same constant 2.

It is now easy to see that the approximation operator $A := M \circ a$ gives the desired approximation performance since with a suitable choice of j we have

$$\begin{aligned}\|f - A(f)\|_H &\le \|f - f_j\|_H + \|M(a(f_j)) - M(a(f))\|_H \\ &\le \varepsilon_n(K)_H + 2\|a(f) - a(f_j)\|_{\ell_2(\mathbb{R}^{26n})} \\ &\le \varepsilon_n(K)_H + 2\|f - f_j\|_H \\ &\le 3\varepsilon_n(K)_H.\end{aligned}$$

Therefore we have proved (5.9). Let us remark, however, that A is not very constructive and that it is generally difficult to create Lipschitz mappings a, M that achieve the optimal performance in stable nonlinear widths.

Remark 5.1. It is shown in Cohen *et al.* (2020) that the conditions SP2 and SP3 are sufficient to guarantee the validity of Carl inequalities of the form (5.12). However, note that they are not sufficient to guarantee the stability we want for perturbations of the input f.

5.7. Weaker measures of stability

It may be argued that requiring Lipschitz stability is too strong a requirement. Recall that Lipschitz stability is just a sufficient condition to guarantee the stability properties S1–S2 that we want. In this direction, we mention that S1–S2 will hold if a, M satisfy the following weaker properties with $\|\cdot\|_Y$ some fixed norm on $\mathbb{R}^n$.

SP1. The mapping $A := M \circ a$, $A\colon K \to X$ is Lipschitz. We can even weaken this further to requiring only $\|A(f) - A(g)\|_X \le C\|f - g\|_X^\alpha$, $f, g \in K$, for some $\alpha \in (0, 1]$. This is known as Lip α stability.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

SP2. The mapping $M : \mathbb{R}^n \to X$ is Lipschitz, or more generally, Lip α with respect to $\|\cdot\|_Y$ on $\mathbb{R}^n$.

While not directly needed for stability, the following property will play a role in our further discussions.

SP3. The mapping $a : K \to \mathbb{R}^n$ is bounded with respect to $\|\cdot\|_Y$ on $\mathbb{R}^n$. This property limits the search over parameter space.

It turns out that if the mappings a, M satisfy the weaker assumptions SP1–SP3, then one can still prove a version of Carl's inequality, and thus we still have limitations on the performance of these approximation methods in terms of entropy number lower bounds. Let us briefly indicate how lower bounds for performance are proved when the mappings a_n, M_n satisfy SP1–SP3. For notational simplicity only, we take $\alpha = 1$, the Lipschitz constant of both M_n and $M_n \circ a_n$, to be γ, and the image of K under a_n to be contained in the unit ball of $\mathbb{R}^n$ with respect to $\|\cdot\|_Y$.

We fix $\varepsilon > 0$ and let $a_n : K \to \mathbb{R}^n$, $M_n : \mathbb{R}^n \to X$, satisfy SP1–SP3 and approximate the elements of K with the accuracy

$$E_{a_n, M_n}(K)_X \le \varepsilon/3 \quad \text{for some } \varepsilon > 0. \tag{5.10}$$

We now show that (5.10) implies a bound on the entropy of K. Let $\mathrm{Pack}_\varepsilon := \{f_1, \dots, f_{P_\varepsilon(K)}\}$ denote a maximal ε-packing of K, *i.e.* a collection of points $\{f_i\} \in K$, with $\min_{i \ne j} \|f_i - f_j\|_X > \varepsilon$, whose size is maximal among all such collections. Now define

$$y_i := a_n(f_i), \quad g_i := (M_n \circ a_n)(f_i) = M_n(y_i), \quad i = 1, \dots, P_\varepsilon(K).$$

It follows that for $i, j = 1, \dots, P_\varepsilon(K)$,

$$\|g_i - g_j\|_X \ge \|f_i - f_j\|_X - \|f_i - g_i\|_X - \|f_j - g_j\|_X > \varepsilon/3, \quad i \ne j,$$

where we used (5.10). Since M_n is γ Lipschitz, we obtain

$$\|y_i - y_j\|_Y \ge \frac{1}{\gamma}\|M_n(y_i) - M_n(y_j)\|_X = \frac{1}{\gamma}\|g_i - g_j\|_X > \frac{\varepsilon}{3\gamma}, \quad i \ne j.$$

In other words, since $\|y_i\|_Y = \|a_n(f_i)\|_Y \le 1$, the collection $\{y_1, \dots, y_{P_\varepsilon(K)}\}$ is an $\varepsilon/(3\gamma)$-packing of the unit ball in $\mathbb{R}^n$. Well-known volumetric considerations show that a maximal such packing can have at most $(1 + 6\gamma\varepsilon^{-1})^n$ elements, and therefore $P_\varepsilon(K) \le (1 + 6\gamma\varepsilon^{-1})^n$.

Now, since the balls of radius ε centred at the f_i are a covering of K, we have that

$$N_\varepsilon(K) \le P_\varepsilon(K) \le (1 + 6\gamma\varepsilon^{-1})^n = 2^{n \log_2(1 + 6\gamma\varepsilon^{-1})}. \tag{5.11}$$

For example, the above derivation shows that whenever there are mappings a_n, M_n satisfying SP1–SP3, then we have the Carl type inequality

$$E_{a_n, M_n}(K)_X \le Cn^{-r}, \ n \ge 1 \implies \varepsilon_n(K)_X \le C' n^{-r} [\log_2 n]^r, \quad n \ge 1. \tag{5.12}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Indeed, we take $\varepsilon = 3Cn^{-r}$ and use (5.11) to find $\varepsilon_{cn\log_2 n} \le Cn^{-r}$, which gives (5.12).

5.8. *Optimal performance for classical model classes described by smoothness*

Although the definition of manifold widths places very mild conditions on the mappings a, M, it still turns out that these conditions are sufficiently strong to restrict how fast $\delta_n(K)_X$ tends to zero for model classes built on classical notions of smoothness described by smoothness conditions such as Sobolev or Besov regularity. For example, if $B^s_q(L_\tau(\Omega))$, with $\Omega = [0,1]^d$, is any Besov space that lies above the Sobolev embedding line for $L_p(\Omega)$, then DeVore *et al.* (1993) proved that

$$\delta_n(U(B^s_q(L_\tau(\Omega))))_{L_p(\Omega)} \asymp \varepsilon_n(U(B^s_q(L_\tau(\Omega))))_{L_p(\Omega)} \asymp n^{-s/d}, \quad n > 0,$$

with the constants in this equivalence depending only on d.

It turns out that the decay rate $O(n^{-s/d})$ can be obtained by many methods of nonlinear approximation such as adaptive finite elements or n-term wavelet approximation. The main message for us is that even with this mild condition of imposing only continuity on the maps a, M, we cannot do better than the rate $O(n^{-s/d})$ for these classical smoothness classes when using manifold approximation. In particular, this holds for NN approximation with the restriction of continuity on the mappings a, M associated to the NNs.

5.9. *VC dimension also limits approximation rates for model classes*

The results we have given above provide lower bounds on how well a model class K can be approximated by a stable manifold approximation. If we remove the requirement of stability, it is still possible to give lower bounds on approximation rates for model classes if the approximation method $(\Sigma_n)_{n\ge 0}$ is made up of sets Σ_n which have limited VC dimension. For such results, one needs some additional assumptions on the model class K. We describe results of this type in this section.

Suppose K is a model class in $L_p(\Omega)$ with $1 \le p \le \infty$. A common technique in proving lower bounds on the Kolmogorov entropy or widths of K is to exhibit a function $\phi \in L_p(\Omega)$ with compact support for which the normalized dilate

$$\Phi(x) := A\phi(\lambda x), \quad x \in \Omega, \tag{5.13}$$

is in K, provided A and λ are chosen appropriately. The function Φ is called a *bump* function. By choosing λ large, one concentrates the support of Φ but of course this is at the expense of making A small in order to guarantee that the resulting ϕ is in K. The small support of Φ guarantees that the shifted functions $\Phi_i(\cdot) = \Phi(\cdot - x^{(i)})$, $i = 1, \dots, N$, are also in K and these functions have disjoint supports, provided N is not too large and the $x^{(i)}$'s are suitably spaced out in Ω. It then follows that for

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

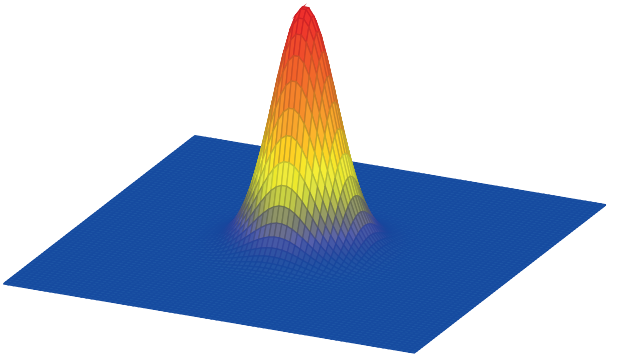

Figure 5.2. A bump function in 3D.

any assignment of signs $\Lambda := (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N)$, $\varepsilon_i = \pm 1$, $i = 1, \ldots, N$, the function

$$f_\Lambda := B \sum_{i=1}^{N} \varepsilon_i \Phi_i \tag{5.14}$$

is also in K for a proper choice of B. One then uses the rich family of functions f_Λ as Λ runs over the 2^N sign patterns to show that the Kolmogorov entropy of K must be suitably large.

This strategy can be used to bound from below how well a model class can be approximated by sets with limited VC dimension. For illustration, we consider the simplest example where $K = U(C^r(\Omega))$, with r being a positive integer, and measure approximation error in the norm $\|\cdot\|_{C(\Omega)}$. If we approximate the functions in K by using a set $\mathcal{F}$ with $\mathrm{VC}(\mathcal{F}) \le m$, then we claim that there is a constant $C = C(r, d) > 0$ such that

$$\delta := \mathrm{dist}(K, \mathcal{F})_{C(\Omega)} \ge C m^{-r/d}. \tag{5.15}$$

We prove this claim in the case $\Omega = [-1, 1]^d$. Consider a non-negative bump function $\phi \in C^r(\mathbb{R}^d)$ which vanishes outside $[-1/2, 1/2]^d$ and has norm $\|\phi\|_{C(\Omega)} = \phi(0)$; see Figure 5.2. The dilated function Φ of (5.13) is in K if we choose A so that $A\phi(0) + A|\phi|_{C^r(\Omega)}\lambda^r = 1$. The support of Φ is contained in a d-dimensional cube centred at 0 with side length λ^{-1}. So if $N = \lfloor \lambda^d \rfloor$, we can make the Φ_i's appearing in (5.14) have disjoint support by taking the $x^{(i)}$ suitably separated. Moreover, if $B = 1$, we have $f_\Lambda \in K$.

Now, to prove (5.15), we take $\lambda = \lceil (m+1)^{1/d} \rceil$ and obtain $N \ge m + 1$ functions $\Phi_i(\cdot) := \Phi(\cdot - x^{(i)}) \in K$, $i = 1, \ldots, N$, with disjoint supports. Then, for each choice of sign patterns the function f_Λ from (5.14) is in K and $f_\Lambda(x^{(i)}) = A\phi(0)\varepsilon_i$, $i = 1, \ldots, N$. Now f_Λ is approximated by an $S_\Lambda \in \mathcal{F}$ to accuracy δ. If δ were smaller than $A\phi(0)$, then the function S_Λ would carry the sign pattern of the ε_i at each $x^{(i)}$. Hence the points $x^{(i)}$, $i = 1, \ldots, N$, would be shattered by $\mathcal{F}$. Since by

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

assumption $\mathrm{VC}(\mathcal{F}) \le m$, this is not possible, and we must have $\delta \ge A\phi(0)$. Since we have that $\phi(0)A = \phi(0)(\phi(0) + \lambda^r |\phi|_{C^r(\Omega)})^{-1} \ge Cm^{-r/d}$, this proves (5.15).

This argument can also be used to prove that there is an absolute constant $C > 0$ depending only on s and d, such that for $K = U(B^s_q(L_\infty(\Omega)))$, with $s > 0$, $0 < q \le \infty$, we have

$$\operatorname{dist}(K, \mathcal{F})_{C(\Omega)} \ge Cm^{-s/d}, \tag{5.16}$$

whenever the VC dimension of $\mathcal{F}$ is at most m. We leave the proof to the reader.

Let us now specialize to the case where $\mathcal{F}$ is the output of a ReLU network. With an eye towards our bounds on the VC dimension for the spaces $\Upsilon^{W,1}(\mathrm{ReLU}; d, 1)$ (see Lemma 3.11) and $\Upsilon^{W_0,L}(\mathrm{ReLU}; d, 1)$ (see Theorem 3.12), we obtain the following lower bounds for NN approximation for $W \ge 1$ and $d \ge 2$:

$$\operatorname{dist}(U(B^s_q(L_\infty(\Omega))), \Upsilon^{W,1}(\mathrm{ReLU}; d, 1))_{C(\Omega)} \ge C(s, d)[W \cdot \log_2 W]^{-s/d}, \tag{5.17}$$

and

$$\operatorname{dist}(U(B^s_q(L_\infty(\Omega))), \Upsilon^{W_0,L}(\mathrm{ReLU}; d, 1))_{C(\Omega)} \ge C(s, d)L^{-2s/d}. \tag{5.18}$$

The logarithm in (5.17) can be removed when $d = 1$.

In the case of (5.17), the lower bound can be stated as $C(s, d)[n \log_2 n]^{-s/d}$, where $n = n(W, 1)$ is the number of parameters used to describe $\Upsilon^{W,1}$. Thus, in this case, save for the logarithm, we cannot achieve any better approximation rates than that obtained by traditional linear methods of approximation. We discuss later in Section 7 what rates have been proved in the literature for one-layer networks.

In the case of (5.18), the lower bound is of the form $C(s, d)n^{-2s/d}$, where $n = n(W_0, L)$ is the number of parameters used to describe the space $\Upsilon^{W_0,L}$. The factor 2 in the exponent leaves open the possibility of much improved approximation rates (when compared with classical methods) when using deep networks. We shall show in Section 8.7 that these rates of approximation are attained.

We close this section by mentioning that use of the VC dimension to bound approximation rates from below seems to be restricted to the case when approximation error is measured in the norm $\|\cdot\|_{C(\Omega)}$. This makes one wonder if there is a concept analogous to the VC dimension suitable for L_p approximation when $p \ne \infty$.

5.10. Another measure of optimal performance: approximation classes

There is another important way to measure the performance of an approximation method $\Sigma = (\Sigma_n)_{n\ge 0}$ by looking at the set of all functions which have a given approximation rate as $n \to \infty$. Let $\lambda = (\lambda_n)_{n\ge 0}$ be a sequence of positive real numbers which decrease monotonically to zero. We define

$$\mathcal{A}(\lambda) := \mathcal{A}(\lambda, \Sigma) := \{f \in X : \exists \Lambda > 0 \text{ such that } E(f, \Sigma_n)_X \le \Lambda\lambda_n, \ \forall n \ge 0\}, \tag{5.19}$$

and further define $\|f\|_{\mathcal{A}(\lambda)}$ as the smallest number Λ for which (5.19) holds. The larger this set is, the better the approximation method Σ is.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

The case when $\lambda_n := (n+1)^{-r}$ is the most often studied since it corresponds to the rates most often encountered in numerical scenarios. In this case $\mathcal{A}(\lambda)$ is usually denoted by

$$\mathcal{A}^r = \mathcal{A}^r(\Sigma) = \mathcal{A}^r(\Sigma, X).$$

A major chapter in approximation theory is to characterize the approximation classes $\mathcal{A}^r$ for a given approximation method. The main theorems of approximation theory characterize $\mathcal{A}^r$ for polynomial and spline approximation. Such characterizations are also known for some methods of nonlinear approximation.

As we shall see, we are far from understanding the approximation classes $\mathcal{A}^r$ for NN approximation. However, some useful results on the structure of these classes can be found in Gribonval, Kutyniok, Nielsen and Voigtlaender (2019).

6. Approximation using ReLU networks: overview

As we have already noted, the collection $\Upsilon^{W,L} = \Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ of outputs of a ReLU network with width W, depth L and input dimension d is a nonlinear set of CPwL functions determined by $n(W, L)$ parameters. Our interest in the next few sections is to summarize the approximation power of $\Upsilon^{W,L}$. In the process of analysing this, we shall not address the question of whether there is a practical stable algorithm to produce the approximation, an issue we will discuss in Section 9.

One of the impediments to giving a coherent presentation of the approximation properties of the outputs of neural networks, as the number of parameters increases, is the great variety of possible architectures of the networks. Namely, when examining the approximation efficiency, we can fix W and let L change, or fix L and let W change, or let both change simultaneously. We can also vary the architecture by allowing full connectivity or sparse connectivity between layers. We may also impose further structure on the weight matrices, leading, for example, to convolution networks. Moreover, we can also consider a variety of activation functions σ.

While each such setting is of interest, we primarily concentrate on two cases of ReLU networks. The first is the case that most closely matches classical approximation, the set $\Upsilon^{W,1}$ as $W \to \infty$. We shall see that even this case is not completely understood. At the other extreme is the case when we take the width W to be some fixed constant W_0 and let $L \to \infty$. This is a most illuminating setting in that we shall see a dramatic gain in approximation efficiency when the depth L is allowed to grow. This is commonly referred to as the *power of depth*.

To provide a unified notational platform, we use Σ_n for the set $\Upsilon^{W,L}$ under consideration, where n is equivalent to the number of parameters being used. For example, we can take $\Sigma_n = \Upsilon^{n,1}$ or $\Sigma_n = \Upsilon^{W_0,n}$ since both of these sets depend on a number of parameters proportional to n. Our goal is to understand how the family $\Sigma := (\Sigma_n)_{n\geq 0}$ performs as an approximation tool.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

In what follows in this section, we consider the set $\Upsilon^{W,L}$ restricted to the domain $\Omega := [0,1]^d$. Recall that each function $S \in \Upsilon^{W,L}$ is the output of a neural network with at most $n(W,L) = (d+1)W + W(W+1)(L-1) + (W+1)$ parameters. We consider the error of approximation to be measured in an $L_p(\Omega)$ norm, $1 \le p \le \infty$. Therefore, for $f \in L_p(\Omega)$, we are interested in the error of approximation

$$E(f,\Sigma_n)_{L_p(\Omega)} := E_n(f,\Sigma)_{L_p(\Omega)} := \inf_{S\in\Sigma_n} \|f - S\|_{L_p(\Omega)},$$

when Σ_n is one of the nonlinear sets $\Upsilon^{W,L}$ and $n \asymp n(W,L)$. In the case $p = \infty$, we assume that f is continuous and the error is measured in the $\|\cdot\|_{C(\Omega)}$ norm, so the results hold uniformly in $x \in \Omega$.

Note that using $L_p(\Omega)$, $1 \le p \le \infty$, norms to measure error does not match the usual measures of performance of classification algorithms, where the main criteria are the probability or expectation of misclassification; see Bousquet, Boucheron and Lugosi (2005). This is an important distinction that we will unfortunately not address because of a lack of definitive results. It may be that this distinction is in fact behind the success of NNs in the learning environment.

The results we prove can be extended to approximation in $L_p(\Omega)$ for $0 < p < 1$, but this requires some technical effort we want to avoid. We concentrate on the three most important cases, namely $p = \infty$ (the case of uniform approximation), the case $p = 2$ which is prevalent in stochastic estimates, and the case $p = 1$ which monitors average error. We always take the $L_p(\Omega)$ spaces with Lebesgue measure. Let us also remark that the results we derive hold equally well for general Lipschitz domains taken in place of $\Omega = [0,1]^d$. If we fix the value of p, the results we seek are of the following two types.

Model class performance. For a model class $K \subset L_p(\Omega)$, we previously defined

$$E(K,\Sigma_n)_p := E_n(K,\Sigma)_{L_p(\Omega)} := \sup_{f\in K} E(f,\Sigma_n)_p.$$

Our interest is to describe the decay of this error (with estimates from above and below) as $n \to \infty$.

There are two types of model classes K that are of interest. The first are classical smoothness classes such as the unit ball of a Lipschitz, Hölder, Sobolev or Besov space; see Section 4. In this way we can compare the approximation properties of NNs with more standard methods of approximation and see whether NNs offer better performance on these classical model classes.

A second type of result of interest is to uncover new model classes K for which NNs perform well and classical methods of approximation do not. Such new model classes would help clarify exactly when NN approximation is beneficial. Motivation for these new model classes should come from the intended application of NN approximation. Such results might explain why NNs perform well in these applications.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Characterization of approximation classes. A second category of results that is of interest would be to understand the approximation classes $\mathcal{A}^r(\Sigma, L_p(\Omega))$ for NN approximation. Recall that these classes (see Section 5.10 for their definition) consist of all functions f whose approximation error satisfies

$$E(f, \Sigma_n)_{L_p(\Omega)} \le M(n+1)^{-r}, \quad n \ge 0, \tag{6.1}$$

with the smallest M defining $\|f\|_{\mathcal{A}^r}$.

We would like to know which functions are in $\mathcal{A}^r$. While a precise characterization of these classes is beyond our current understanding of NN approximation, the results that follow give sufficient conditions for a function f to be in such a class. In contrast, for many types of classical approximation, both linear and nonlinear, there are characterizations of their corresponding approximation classes. Such characterizations require what are called *inverse theorems* in approximation theory. An inverse theorem is a statement that whenever $f \in \mathcal{A}^r$, we can prove that f is in a certain Banach space Y_r.

Consider, for example, the case of approximation in $L_p(\Omega)$. An inverse theorem is proved by showing an inequality of the form

$$|S|_{Y_r} \le C(n+1)^r \|S\|_{L_p(\Omega)}, \quad S \in \Sigma_n, \quad n \ge 0.$$

For example, if we consider approximation by trigonometric polynomials of degree n in one variable, in the metric $L_p([-\pi, \pi])$, one inequality of this type is the famous Bernstein inequality for trigonometric polynomials,

$$\|T'\|_{L_p(\Omega)} \le n\|T\|_{L_p(\Omega)},$$

which holds for any trigonometric polynomial of degree n. So $r = 1$ in this example, and $Y_1 = W^1(L_p([-\pi, \pi]))$.

Such inverse theorems are not known for NN approximation save for the case of $\Sigma = (\Upsilon^{n,1}(\sigma; 1, 1))_{n\ge 0}$ for certain activation functions σ, including ReLU. Thus there is quite a large gap in our understanding of NN approximation as compared to these more classical methods. It is of major interest to establish inverse inequalities for the elements in Σ_n when Σ_n is a set of outputs of an NN.

7. Approximation using single-layer ReLU networks

In this section we study approximation on the domain $\Omega := [0, 1]^d$ by the family $\Sigma := (\Sigma_n)_{n\ge 0}$, where $\Sigma_0 := \{0\}$ and $\Sigma_n := \Upsilon^{n,1}(\mathrm{ReLU}; d, 1)$, $n = 1, 2 \ldots$, with input dimension $d \ge 1$. These sets are rarely used in numerical settings since deeper networks are preferred. However, for theoretical reasons, it is important to understand their approximation properties in order to see the advantages of the deeper networks studied later in this paper. Much of the activity on NN approximation has been directed at understanding the approximation properties of these single-hidden-layer networks. Surprisingly, we shall see that most fundamental questions about approximation using Σ are not yet answered.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

We have discussed in Section 3.2.1 the structure of Σ_n. Each function $S \in \Sigma_n$ is a CPwL function in d variables $x = (x_1, \dots, x_d)$ of the form

$$S(x) := b_0 + \sum_{j=1}^{n} a_j \eta_j(x), \quad \eta_j(x) := (w_j \cdot x + b_j)_+, \; w_j \in \mathbb{R}^d, \; b_j, a_j \in \mathbb{R}, \quad (7.1)$$

where η_j is linear on the half-space $H_j^+ := \{x : w_j \cdot x + b_j > 0\}$ and is zero otherwise. Note that $S \in \Sigma_n$ is a CPwL function subordinate to a hyperplane partition $\mathcal{P}$ of $\mathbb{R}^d$ into cells which are convex polytopes.

In spite of the simplicity of the representation (7.1), the set Σ_n is quite complicated save for the case $d = 1$; see Section 3.1.1. First of all, the possible partitions $\mathcal{P}$ that arise from hyperplane arrangements are complex in the sense that the cells are not isotropic, the number of cells can be quite large, and there is not a simple characterization of these partitions. This is compounded by the fact that, as we have previously discussed, not every CPwL function subordinate to a partition given by an arrangement of n hyperplanes is in Σ_n. For example, this set does not contain any compactly supported functions when $d > 1$. This is in contrast to the typical applications of CPwL functions in numerical PDEs. Thus Σ_n is a complex but possibly rich nonlinear family. We shall see that this complexity inhibits our understanding of its approximation properties.

Keeping in mind the discussion in the previous section, there are three types of results that we would like to prove in order to understand the approximation power of $\Sigma := (\Sigma_n)_{n \geq 0}$, measured in the $\| \cdot \|_{L_p(\Omega)}$ norm, $1 \leq p \leq \infty$.

Problem 7.1. Give matching upper and lower bounds for $E_n(K, \Sigma)_{L_p(\Omega)}$ when K is one of the classical model classes such as unit balls of Lipschitz, Hölder, Sobolev and Besov spaces.

We shall see that, save for the case $d = 1$, this problem is far from being solved.

As we have previously stressed, the partitions generated by hyperplane arrangements are complex and not well understood, with cells that are possibly highly anisotropic. This suggests the possibility of being able to approximate functions which are not described by classical isotropic smoothness and leads us to expect new model classes that are well approximated by Σ.

Problem 7.2. Describe new model classes K of functions that are guaranteed to be well approximated by Σ.

Some advances on Problem 7.2 have been made, centering on the so-called Barron classes that we discuss in Section 7.2.3.

Finally, the most ambitious approximation problem for $\Sigma = (\Sigma_n)_{n \geq 0}$ is the following.

Problem 7.3. For each $r > 0$ and $1 \leq p \leq \infty$, characterize the approximation class $\mathcal{A}^r(\Sigma, L_p(\Omega))$ consisting of all functions $f \in L_p(\Omega)$ for which

$$E_n(f, \Sigma)_{L_p(\Omega)} = O((n+1)^{-r}), \quad n \geq 0.$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Nothing is known on this last problem when $d > 1$, and we are sceptical that any definitive result is around the corner for the case of general d.

In order to orient us to the type of results we might strive to obtain on these problems for general d, we begin in the next section by discussing the case $d = 1$, where we have the most extensive results and the best understanding of approximation from these spaces.

7.1. Approximation by single-layer networks when $d = 1$

We begin by discussing the case $d = 1$, not only because it is the best understood but also because it can orient the reader as to what we can possibly expect when engaging the case $d > 1$. Because approximation by $\Upsilon^{n,1}(\mathrm{ReLU}; 1, 1)$ is essentially the same as free-knot linear spline approximation, results for the NN approximation are derived from the known results on free-knot splines. The latter are well explained in DeVore (1998) and the literature cited therein, and summarized below.

7.1.1. Approximation of classical model classes when $d = 1$

Here we measure approximation error in $L_p(\Omega)$ with $1 \le p \le \infty$ and domain $\Omega = [0, 1]$. The classical model classes for $L_p(\Omega)$ are finite balls in the Lipschitz, Hölder, Sobolev and Besov spaces. The latter spaces are the most flexible for measuring smoothness and approximation properties, as all of the other smoothness classes can be derived from them. So we restrict our discussion to the model classes $K = U(B^s_q(L_\tau(\Omega)))$, $0 < q, \tau \le \infty$, which have smoothness of order $s > 0$. These spaces were introduced and discussed in Section 4.3, where we noted that these spaces are compactly embedded in $L_p(\Omega)$ when $s > 1/\tau - 1/p$, *i.e.* when these spaces lie above the Sobolev embedding line; see Figure 4.1. They are not embedded in $L_p(\Omega)$ if they lie below the embedding line.

The following theorem summarizes the results known about approximating Besov classes in the case $d = 1$.

Theorem 7.4. Let $K = U(B^s_q(L_\tau(\Omega)))$ be the unit ball of the Besov space $B^s_q(L_\tau(\Omega))$. If $0 < s \le 2$ and this space lies above the Sobolev embedding line for $L_p(\Omega)$, $1 \le p \le \infty$, then

$$E_n(K, \Sigma)_p \le C(s, p, \tau)(n + 1)^{-s}, \quad n \ge 0. \tag{7.2}$$

Let us elaborate a little on what this theorem is saying. First, note that the sets K for which we obtain the approximation rate $O((n + 1)^{-s})$ allow the smoothness describing K to be measured in $L_\tau(\Omega)$, where $\tau \ne p$. When $\tau \ge p$, the result does not need to exploit the nonlinearity of Σ_n in the sense that the approximation rate can be obtained already by using linear spaces corresponding to fixing the breakpoints in Σ_n to be equally spaced on $[0, 1]$. It is only when $\tau < p$ that we need to exploit nonlinearity.

A couple of simple examples may be in order. Consider approximation in $C(\Omega)$ and smoothness of order $s = 1$. Obviously the space Lip 1 is compactly embedded

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

in $C(\Omega)$ and the approximation rate is $O((n+1)^{-1})$, $n \to \infty$, when $K = U(\text{Lip}\,1)$. Note that Lip 1 is not a Besov space but is continuously embedded in $B^1_\infty(L_\infty(\Omega))$ and the latter space is covered by the theorem. Hence Lip 1 is also covered by the theorem. We can obtain the approximation rate $O((n+1)^{-1})$ by taking the breakpoints equally spaced and thereby using a linear subspace of Σ_n. The Sobolev space $W^1(L_1(\Omega))$ is also contained in $C(\Omega)$ but not compactly. Nevertheless, its unit ball has the approximation rate $O((n+1)^{-1})$. The Sobolev spaces $W^1(L_p(\Omega))$, $p > 1$, have unit balls that are compact in $C(\Omega)$ and the theorem gives that they also have the approximation rate $O((n+1)^{-1})$, $n \to \infty$. Recall that for f to be in Lip 1 requires that it has bounded derivative $\|f'\|_{L_\infty(\Omega)} < \infty$, while $f \in W^1(L_p(\Omega))$ only requires $f' \in L_p(\Omega)$. For example, the function $f(t) = t^\alpha$, $0 < \alpha < 1$, is in $W^1(L_p(\Omega))$ if $p > 1$ is small enough, but this function is not in Lip 1. The way to get good approximation of t^α by Σ_n is to put half of the breakpoints of the output $S \in \Sigma_n$ near 0 and the remaining half equally spaced in Ω. Thus, for these Sobolev spaces one truly needs the nonlinearity of Σ_n. To achieve the optimal approximation rate, we need to choose the breakpoints to depend on f, and thus we cannot choose them in advance.

Finally, let us remark why we have the restriction $s \le 2$. We are approximating locally by linear functions. A function f with smoothness of order $s > 2$ would need to use locally polynomials of degree higher than one to improve its local error of approximation (think of Taylor expansions). Hence, when f has smoothness of order $s > 2$, we do not improve on the rate $O((n+1)^{-2})$, $n \to \infty$, which we already have for functions with smoothness of order 2.

7.1.2. Approximation classes for $d = 1$

One of the crowning achievements of nonlinear approximation at the end of the last century was the characterization of the approximation classes for several classical methods of nonlinear approximation, including free-knot spline, n-term wavelet and adaptive piecewise polynomial approximation. The key to establishing these results was not only to give upper bounds for the error in approximating functions from Besov spaces but also to prove certain inverse theorems saying that if a function f can be approximated with a certain rate $O((n+1)^{-r})$, $n \to \infty$, then f must possess a certain Besov smoothness. These inverse theorems should not be underestimated since they allow precise characterization of approximation classes.

In the case of approximation using CPwL functions, the inverse theorems were provided by the seminal theorems of Pencho Petrushev; see Petrushev (1988). The approximation space $\mathcal{A}^r = \mathcal{A}^r(\Sigma, L_p(\Omega))$ is precisely characterized, provided $0 < r < 2$ and $1 \le p \le \infty$, with $C(\Omega)$ used in place of $L_\infty(\Omega)$ when $p = \infty$. In this case $\mathcal{A}^r$ is a certain interpolation space; see DeVore (1998). Since we do not want to go too deeply into interpolation space theory here, we simply mention that $\mathcal{A}^r$ is sandwiched between two Besov spaces of smoothness order r. More precisely, if $0 < r < 2$ and $1 \le p \le \infty$ are fixed, and $\tau^* := (r + 1/p)^{-1}$, then for all $0 < q \le \infty$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

we have

$$B_q^r(L_\tau(\Omega)) \subset \mathcal{A}^r \subset B_\infty^r(L_{\tau^*}(\Omega)), \quad \text{whenever } \tau > \tau^*. \tag{7.3}$$

Since this result may be difficult to digest at first glance, we make some comments to explain what these embeddings say. First, recall the relation of Besov spaces to the Sobolev embedding line; see Figure 4.1. For a fixed value of r, all spaces $B_q^r(L_\tau(\Omega))$ appearing on the left-hand side of the embedding (7.3) are compactly embedded in the space $L_p(\Omega)$, where we are measuring error. The left embedding says that any function in one of these spaces is in $\mathcal{A}^r$, and hence has approximation error decaying at the rate $O((n+1)^{-r})$. Note that these spaces get larger as we approach the embedding line. The right embedding says that we cannot allow τ to be smaller than τ^*; in fact if τ is smaller than τ^* we do not even embed into $L_p(\Omega)$. Besov spaces that appear on the embedding line itself may or may not be compactly embedded in $L_p(\Omega)$, depending on q. They are compactly embedded if q is small enough.

Finally, we remark that we have the characterization of $\mathcal{A}^r$ only if $r < 2$ for the same reason we had the restriction $s \le 2$ when discussing approximation of classical model classes in the previous section. Going a little further, note that if $S_n \in \Sigma_n$, $n \ge 1$, then $S_n \in \mathcal{A}^r$ for all $r > 0$, but S_n is not in any smoothness space of order $s > 2$. Moreover, any function $f = \sum_{k\ge 1} \alpha_k S_k$, $\alpha_k \in \mathbb{R}$, will be in $\mathcal{A}^r$, $0 < r < \infty$, if $(\alpha_k)_{k\ge 1}$ tends to zero sufficiently fast. However, f will not have any classical smoothness of order $s > 2$. So $\mathcal{A}^r$, $r > 2$, cannot be characterized by classical smoothness such as membership in a Besov space.

7.2. *Results for $d \ge 2$*

Continuing with single-hidden-layer networks, let us now consider the case $d \ge 2$. The difficulty in constructing effective approximations in this case is the fact that when $d \ge 2$, the set of NN outputs $\Sigma_n = \Upsilon^{n,1}(\mathrm{ReLU}; d, 1)$ does not have locally supported nodal functions that are commonly used to build approximants. So approximation methods are built on global constructions. It is not surprising, therefore, that the strongest results are known in the case where the approximation is measured in the $L_2(\Omega)$ norm, where orthogonality can be employed in the constructions. We discuss the L_2 approximation first.

7.2.1. *Approximation in $L_2(\Omega)$*

For approximation in $X = L_2(\Omega)$, it is known that when $f \in W^s(L_2(\Omega))$ we have

$$E_n(f, \Sigma)_{L_2(\Omega)} \le C n^{-s/d} \|f\|_{W^s(L_2(\Omega))}, \quad n \ge 1, \tag{7.4}$$

provided $s \le 2 + (d-1)/2$. The case $d = 2$ is given in DeVore, Oskolkov and Petrushev (1997) and the general case is considered in Petrushev (1998).

We give a very coarse description of the ideas behind proving (7.4). In this discussion it is useful to work with functions defined on the unit Euclidean ball $\Omega^* \subset \mathbb{R}^d$ rather than on the cube $[0,1]^d$. One can move between these different domains

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

via restriction and extension operators which are known to preserve Sobolev and Besov regularity. When g is a univariate function, then $g(a \cdot x)$, $a \in \mathbb{R}^d$ is a *ridge function* of d variables (sometimes called a planar wave). If X_n is a linear space of dimension n of univariate functions and Λ is a fixed subset of m unit vectors in $\mathbb{R}^d$, then the set of functions $Y_{n,m} := \operatorname{span}\{g(a \cdot x)\colon\ g \in X_n,\ a \in \Lambda\}$ is a linear space of dimension at most mn.

The core of the proof of (7.4) is to show that if X_n is effective in approximating univariate functions in L_2, and if the set Λ is 'uniformly distributed' on the unit sphere in $\mathbb{R}^d$ (the boundary of the unit ball of $\mathbb{R}^d$), then $Y_{m,n}$ will provide an approximation to $W^s(L_2(\Omega^*))$ functions when choosing $m = O(n^{d-1})$. We can take X_n to be the space of univariate CPwL functions on an equally spaced partition. The resulting space $Y_{n,m}$ is contained in $\Upsilon^{mn,1}(\mathrm{ReLU}; d, 1)$, and thereby proves (7.4). The proofs of these results are quite elaborate and technical.

If we wish to characterize the approximation performance of Σ on the model class $K := U(W^s(L_2(\Omega^*))$, then we would need to establish lower bounds for the approximation error $E_n(K)_{L_2(\Omega^*)}$ that match those of (7.4). Such bounds are plausible but seem not to be known. However, there are lower bounds for approximating K by general ridge functions given in Maiorov (1999), which for our setting and $d \geq 2$ give the lower bound

$$E_n(K)_{L_2(\Omega^*)} \geq Cn^{-s/(d-1)}, \quad n \geq 1,$$

where C depends only on d.

While the results given above are less than satisfactory, because of the lack of matching upper and lower bounds, the situation becomes even worse when we seek results that show the benefits of the nonlinear structure of the sets Σ_n, $n \geq 1$. As we know from the case $d = 1$, nonlinear methods of approximation should allow smoothness to be measured in the weaker $L_\tau(\Omega)$ norms while retaining the same approximation order. Namely, the question is what the approximation rates are when K is the unit ball of a Besov space $B^s_q(L_\tau(\Omega))$ that is above the Sobolev embedding line for $L_2(\Omega)$. In contrast to the case $d = 1$, we do not know results that quantify the performance of Σ, for the Besov spaces that compactly embed into L_2.

7.2.2. Approximation in L_p, $p \neq 2$

When we consider approximation in $L_p(\Omega)$, $p \neq 2$, we are only aware of results for $p = \infty$ given in Bach (2017). These are only stated for the unit ball K of Lip 1 with approximation error measured in the norm of $C(\Omega)$, and take the form

$$E_n(K, \Sigma)_{C(\Omega)} \leq C\frac{\log_2 n}{n^{1/d}}, \quad n \geq 1.$$

Since we are now considering approximation in the space $C(\Omega)$, we can use the known upper bounds for the VC dimension of $\Upsilon^{n,1}(\mathrm{ReLU}; d, 1)$ to derive lower bounds on the approximation error for K. If we apply Lemma 3.11 and employ

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

arguments similar to those we used to prove (5.15), we obtain

$$E_n(K,\Sigma)_{C(\Omega)} \geq C[n\log_2 n]^{-1/d}, \quad n \geq 1.$$

In other words, modulo logarithms, the approximation rate of K is $n^{-1/d}$. It is of interest to remove these log terms.

We can derive bounds on the approximation rates for the model classes $K_\alpha := U(\mathrm{Lip}\,\alpha)$, $0 < \alpha < 1$, from the known Lip 1 bound by using interpolation theory (see Extend 1 in Section 4.4), which gives the bound

$$E_n(K_\alpha,\Sigma)_{C(\Omega)} \leq C\left[\frac{\log_2 n}{n^{1/d}}\right]^\alpha, \quad n \geq 1.$$

One expects that these results also extend to error estimates for approximation by Σ_n of the unit balls of the smoothness spaces $B^s_q(L_\infty(\Omega))$ for some range of s larger than one. However, these do not seem to be found in the literature. Equally missing are results for approximation in $L_p(\Omega)$ when $p \neq 2, \infty$. Moreover, none of the known results reflect the expected gain from the fact that Σ is a nonlinear method of approximation.

7.2.3. Novel model classes for single-layer approximation

As we noted earlier, there is much interest in identifying new model classes for which NN approximation is particularly effective. One celebrated model class of this type was introduced by Andrew Barron (1994). This model class and its corresponding approximation results are nicely explained in the exposition by Pinkus (1999). The most recent results on NN approximation of this class of functions can be found in Siegel and Xu (2020) and Klusowski and Barron (2018). We limit ourselves to describing how these model classes fit into the themes of this article.

Given any domain $\Omega \subset \mathbb{R}^d$, Barron introduced the model class $K = K_\Omega$ consisting of all functions $f \in L_2(\Omega)$ which have an extension to all of $\mathbb{R}^d$ (still denoted by f) whose Fourier transform $\hat{f}$ satisfies

$$\int_{\mathbb{R}^d} \|\omega\|_{\ell_1(\mathbb{R}^d)} |\hat{f}(\omega)|\, \mathrm{d}\omega \leq 1. \tag{7.5}$$

Note that (7.5) imposes additional conditions over just requiring that f is square-integrable. Namely, (7.5) requires the decay of $\hat{f}(\omega)$ as the frequency ω gets large. It is easy to check that this is equivalent to requiring that f has a gradient (in the weak sense) whose Fourier transform is in L_1.

Barron initially showed that for any sigmoidal activation function σ the approximation family $\Sigma := (\Upsilon^{n,1}(\sigma; d, 1))_{n\geq 1}$ approximates the model class K_Ω in the norm of $L_2(\Omega)$ with the following accuracy:

$$E_n(K_\Omega,\Sigma)_{L_2(\Omega)} \leq C_\Omega n^{-1/2}, \quad n \geq 1. \tag{7.6}$$

This result was then shown to hold for ReLU activation as well, by using the fact that $(\mathrm{ReLU}(t) - \mathrm{ReLU}(t-1))$ is a sigmoidal function.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Barron's result has inspired a lot of generalizations and applications, and even the introduction of new Banach spaces; see E, Ma and Wu (2019). Important generalizations of (7.6) were given by Makovoz (1996), who showed that the above result for the class K_Ω holds for approximation in L_q, $1 \le q < \infty$, and moreover, the rate of approximation can be improved to $O(n^{-1/2-1/(q^* d)})$, where q^* is the smallest even integer $\ge q$. Further improvements on approximation rates for Barron classes and their generalizations have been given through the years. We refer the reader to Siegel and Xu (2020) for the latest information.

We will not dig too deeply into the known approximation rates for Barron classes and their generalizations here. Rather, we confine ourselves to some comments to properly frame these results in the context of nonlinear approximation. Let H be a Hilbert space. We say that a collection $\mathcal{D} := \{\phi\}$ of functions from H is a dictionary if each ϕ has norm one, and whenever $\phi \in \mathcal{D}$ then so is $-\phi$. Given such a dictionary $\mathcal{D}$, we consider the closed convex hull $\mathrm{co}(\mathcal{D})$ of $\mathcal{D}$. A fundamental result in approximation theory is that whenever $f \in \mathrm{co}(\mathcal{D})$, then there exists $g = \sum_{k=1}^n c_k \phi_k$ with the $\phi_k \in \mathcal{D}$, such that

$$\|f - g\|_H \le C n^{-1/2}, \quad n \ge 1. \tag{7.7}$$

There is a constructive method to find such a g, known as the *orthogonal greedy algorithm*; see DeVore and Temlyakov (1996).

To derive (7.6) from this, it is enough to show that K_Ω is contained in the convex hull of the dictionary of all functions $c\sigma(w \cdot x + b)$ with $w \in \mathbb{R}^d$, $b \in \mathbb{R}$ and c a suitable normalizing constant. The proof of this fact can be found in Barron (1993) and Pinkus (1999). The improvements of Makovoz rest on the fact that in the case of sigmoidal functions, the dictionary elements $\sigma(w \cdot x + b)$ are very close to one another when the parameters w and b change slightly and so one can reduce the number of terms needed in the approximation when seeking an error ε.

Note that neither the constant C_Ω nor the form of the decay $n^{-1/2}$ in (7.6) depend on d. This should be compared with approximation for Sobolev classes where the rates decrease and the constant explodes in size as d grows. However, this must be viewed in the light that the condition for membership in K gets much stronger as d gets large. This class is analogous to requiring f to have a Fourier series (in d variables) whose coefficients are absolutely summable. Another important point is that the proof of (7.6) exploits nonlinear approximation since the n terms from the dictionary $\mathcal{D}$ used to approximate f are chosen to depend on f.

8. Approximation using deep ReLU networks

We now study in detail the approximation by the family $\Sigma = (\Sigma_n)_{n\ge 0}$ of deep networks, with $\Sigma_0 := \{0\}$ and $\Sigma_n := \Upsilon^{W_0,n}(\mathrm{ReLU}; d, 1)$, $n \ge 1$, where W_0 is fixed depending on d. The three main conclusions we uncover, following the order of our exposition, are as follows:

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

- When error is measured in an L_p norm, $1 \le p \le \infty$, deep NNs approximate functions in the classical model classes (such as Lipschitz, Hölder, Sobolev and Besov classes) at least as well as all of the known methods of nonlinear approximation; see Section 8.6.
- For all classical model classes, deep NN approximation gives error rates dramatically better than all other standard methods of nonlinear approximation; see Section 8.7.
- There are novel model classes, built on the ideas of self-similarity, where NNs provide approximation rates not available by standard approximation methods; see Section 8.10.

8.1. Results obtained from basic decompositions

In this section we describe what is perhaps the most common method of obtaining estimates for deep NN approximation. It is based on two principles. The first is to show that the target function has a decomposition in terms of fundamental building blocks with a control on the coefficients in the decomposition. These building blocks could be wavelets or some of their mathematical cousins, such as shearlets or ridgelets, or they could be global representations such as power series or Fourier decompositions. For functions f in classical smoothness spaces, we often know the existence of such decompositions with quantifiable bounds on the coefficients of f. The second step is then to show that each of these building blocks can be captured very efficiently (usually with exponential accuracy) by deep networks.

These two principles can then be put together in order to give quantifiable performance for approximation using deep NNs. This technique appears often in the literature. A partial list of prominent papers using this method are those of Yarotsky (2017), Opschoor, Schwab and Zech (2019*b*), Bölcskei, Grohs, Kutyniok and Petersen (2019), Gühring, Raslan and Kutyniok (2020), Elbrächter, Perekrestenko, Grohs and Bölcskei (2019), Petersen and Voigtlaender (2018), Petersen (2020), E and Wang (2018), Shen *et al.* (2019) and Lu, Shen, Yang and Zhang (2020).

We formalize the above-mentioned procedure by considering any Banach space X and representing $f \in X$ as $f = \sum_{k\ge1} \alpha_k g_k$, where the α_k's are scalars, $g_k \in X$, and $\|g_k\|_X = 1$. Then we can bound the error in approximating f by its partial sum by

$$\left\| f - \sum_{k\le n} \alpha_k g_k \right\|_X \le \sum_{k>n} |\alpha_k|.$$

We can exploit this simple observation in the context of neural networks as follows. If $g_k \in \Upsilon^{W_0-(d+1),n}(\mathrm{ReLU}; d, 1)$, $1 \le k \le n$, then

$$E_{n^2}(f, \Sigma)_X \le \sum_{k>n} |\alpha_k|, \tag{8.1}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

since the partial sum satisfies

$$\sum_{k=1}^{n} \alpha_k g_k \in \Sigma_{n^2} = \Upsilon^{W_0, n^2}(\mathrm{ReLU}; d, 1)$$

(see **Addition by increasing depth** in Section 3.3.2).

The bound (8.1) is quite crude and can be improved in many ways. For example, we can give a better control on depth needed, when each g_k is a composition of the same univariate function T. We shall use this fact below, so we formulate it in the following proposition.

Proposition 8.1. If $T \in \Upsilon^{W_0-1, L_0}(\mathrm{ReLU}; 1, 1)$, then any linear combination S satisfies

$$S = \sum_{i=1}^{m} \alpha_i T^{\circ i} \in \Upsilon^{W_0, mL_0}(\mathrm{ReLU}; 1, 1).$$

Proof. Let $\mathcal{N}$ be a neural network with width $W_0 - 1$ and depth L_0, with input and output dimension 1, whose output function is T. We concatenate $\mathcal{N}$ with itself $(m-1)$ times to obtain the network $\mathcal{N}^*$ of width W_0 and depth mL_0. Note that the (kL_0)th layer of $\mathcal{N}^*$ can output $T^{\circ k}$. We add one collation channel to $\mathcal{N}^*$, whose nodes pass value zero until layer (L_0+1), where its node collects $\alpha_1 T$. This value is then passed forward until layer $2L_0+1$, where $\alpha_2 T^{\circ 2}$ is added, so that $\alpha_1 T + \alpha_2 T^{\circ 2}$ is now held in the node of this channel for layers, $2L_0+1, \dots 3L_0$. We continue in this way. Then we output S from the (mL_0)th layer. □

In deriving an estimate like (8.1), it is not necessary to assume that the functions $g_k \in \Sigma_n$, $k = 1, \dots, n$, but merely that the g_k's are approximated sufficiently well by Σ_n, as we see in the next proposition.

Proposition 8.2. If g_k, $k = 1, \dots, n$, can be approximated by outputs $\hat{g}_k$ from $\Upsilon^{W_0-(d+1), n}(\mathrm{ReLU}; d, 1)$ with error $\|g_k - \hat{g}_k\|_X \le \varepsilon$, then the function

$$\hat{S} := \sum_{k=1}^{n} \alpha_k \hat{g}_k$$

is in $\Upsilon^{W_0, n^2}(\mathrm{ReLU}; d, 1)$, and

$$E_{n^2}(f, \Sigma)_X \le \|f - \hat{S}\|_X \le \varepsilon \sum_{k=1}^{n} |\alpha_k| + \sum_{k>n} |\alpha_k|. \tag{8.2}$$

Proof. The error estimate (8.2) follows from the fact that

$$\left\| f - \sum_{k=1}^{n} \alpha_k \hat{g}_k \right\|_X \le \left\| f - \sum_{k=1}^{n} \alpha_k g_k \right\|_X + \sum_{k=1}^{n} |\alpha_k| \|g_k - \hat{g}_k\|_X.$$

The network that outputs $\hat{S}$ is obtained in the same way as described above. □

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

In this section we shall use the following theorem.

Theorem 8.3. Let $\varphi \in \Upsilon^{W_0,L_0}(\mathrm{ReLU}; d, 1)$, A_j be a $d \times d$ matrix, and $b_j \in \mathbb{R}^d$, $j = 1, \dots, n$. Then the function

$$S = \sum_{j=1}^{n} c_j \varphi(A_j x + b_j), \quad c_j \in \mathbb{R}, \tag{8.3}$$

is in $\Upsilon^{d+1+W_0, nL_0}(\mathrm{ReLU}; d, 1)$.

Proof. Let $\mathcal{N}_0$ be the network which outputs φ, and let A^* be the $W_0 \times d$ matrix of input weights of $\mathcal{N}_0$ and b^* the biases of its first layer.

We build a special network $\mathcal{N}$ with width $W = d + 1 + W_0$ and depth $L = nL_0$ to output S. Its first d channels are source channels to push forward $x_1, \dots, x_d$. The next W_0 channels will be the channels of $\mathcal{N}_0$, and the final channel will be a collation channel to form the sum defining S.

The network $\mathcal{N}$ consists of n copies of $\mathcal{N}_0$ placed next to each other. We feed the source channels to the jth copy of $\mathcal{N}_0$, $j = 1, \dots, n$. For this copy we use input matrix A^*A_j and bias $(b^* + A^*b_j)$ for its first layer. The nodes of the collation channel forward zeros up to layer $L_0 + 1$, where the output $c_1\varphi(A_1x + b_1)$ of the first copy of $\mathcal{N}_0$ is entered and then forwarded. The output of the jth copy is multiplied by c_j through modification of the output weights of $\mathcal{N}_0$, and forwarded to the $(jL_0 + 1)$th node of the collation channel if $j < n$, where it is added to the current sum in that channel and then the result is forwarded. When $j = n$ the output of the nth copy is output together with the content of the collation channel to produce S. □

8.2. *Approximation of products*

We now turn to showing how to approximate certain simple building blocks with exponential accuracy using deep ReLU networks. These building blocks include monomials, polynomials, tensor products and B-splines. An important tool in establishing such results is to show how one can approximate products of functions, which is our next item of interest.

Let H be the hat function introduced in (3.4). We begin with the well-known formula[2]

$$t(1 - t) = \sum_{k \geq 1} 4^{-k} H^{\circ k}(t), \quad t \in [0, 1]. \tag{8.4}$$

We define

$$S(t) := t^2 \quad \text{and} \quad S_n(t) := t - \sum_{k=1}^{n} 4^{-k} H^{\circ k}(t), \quad n \geq 1, \quad t \in [0, 1].$$

[2] It is not clear who was the first to observe this formula, but it appears already in Hata (1986).

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Let us note that we can also represent S_n by

$$S_n(t) := t^2 + \sum_{k=n+1}^{\infty} 4^{-k} H^{\circ k}(t), \quad n \geq 1. \tag{8.5}$$

These two representations of S_n show that

$$t^2 \leq S_n(t) \leq t, \quad t \in [0,1], \tag{8.6}$$

and so $S_n : [0,1] \to [0,1]$.

We now prove the following univariate result.

Proposition 8.4. For each $n \geq 1$, the function $S_n \in \Upsilon^{4,n}(\mathrm{ReLU}; 1, 1)$ satisfies

$$\|S - S_n\|_{C([0,1])} \leq \frac{1}{3} \cdot 4^{-n}, \quad n \geq 1, \tag{8.7}$$

and

$$\|S' - S_n'\|_{L_\infty([0,1])} \leq 2^{-n}, \quad n \geq 1. \tag{8.8}$$

Proof. Since $H \in \Upsilon^{2,1}(\mathrm{ReLU}; 1, 1)$, in view of Proposition 8.1, there is a ReLU network of width 3 and depth $L = n$ that outputs $\sum_{k=1}^{n} 4^{-k} H^{\circ k}$. If we add one more channel to push forward t, then we can also output $S_n(t) := t - \sum_{k=1}^{n} 4^{-k} H^{\circ k}(t)$.

Since

$$S(t) - S_n(t) = -\sum_{k=n+1}^{\infty} 4^{-k} H^{\circ k}(t),$$

the bound (8.7) follows from

$$|S(t) - S_n(t)| \leq \sum_{k=n+1}^{\infty} 4^{-k} \leq \frac{1}{3} \cdot 4^{-n}, \quad t \in [0,1], \tag{8.9}$$

whereas (8.8) follows from the fact that each $H^{\circ k}$ has Lipschitz norm 2^k. □

Let us mention that there are many functions other than t^2 for which explicit formulas like (8.4) hold. These will be discussed in Section 8.10. For now, we want to examine how we can capture higher-order monomials from the above results. First we begin by showing how we can implement multiplication using deep ReLU networks. We start with the simple formula

$$\Pi(x_1, x_2) := x_1 x_2 = 2S\left(\frac{x_1 + x_2}{2}\right) - \frac{1}{2}\{S(x_1) + S(x_2)\}, \quad x_1, x_2 \in [0,1]. \tag{8.10}$$

We can construct a neural network with input dimension 2 which outputs the function $\Pi(\cdot, \cdot)$ with high accuracy.

For $n \geq 1$, we define for $x_1, x_2 \in [0,1]$ the function

$$\Pi_n(x_1, x_2) := 2S_n\left(\frac{x_1 + x_2}{2}\right) - \frac{1}{2}\{S_n(x_1) + S_n(x_2)\}, \tag{8.11}$$

and prove the following properties of Π_n.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Proposition 8.5. For each $n \geq 1$, $\Pi_n(x_1, x_2) \in [0, 1]$ for $(x_1, x_2) \in [0, 1]^2$.

Proof. First we show that $\Pi_n(x_1, x_2) \leq 1$ for $(x_1, x_2) \in [0, 1]^2$. Indeed, this follows from (8.6) since for $(x_1, x_2) \in [0, 1]^2$

$$\begin{aligned}\Pi_n(x_1, x_2) &= 2S_n\left(\frac{x_1 + x_2}{2}\right) - \frac{1}{2}\{S_n(x_1) + S_n(x_2)\} \\ &\leq (x_1 + x_2) - \frac{1}{2}(x_1^2 + x_2^2) \\ &= \frac{1}{2}[x_1(2 - x_1) + x_2(2 - x_2)] \\ &\leq 1.\end{aligned}$$

To show that $\Pi_n \geq 0$, we start with

$$2\Pi_n(x_1, x_2) = x_1 + x_2 + \sum_{k=1}^{n} 4^{-k}\left[H^{\circ k}(x_1) + H^{\circ k}(x_2) - 4H^{\circ k}\left(\frac{x_1 + x_2}{2}\right)\right]. \quad (8.12)$$

We introduce the function

$$\zeta(t) := 2\min\{|t - m| : m \in \mathbb{Z}\}, \quad t \in \mathbb{R}.$$

Then for $t \in [0, 1]$ we have

$$H(t) = \zeta(t) \quad \text{and} \quad H^{\circ k}(t) = \zeta(2^{k-1}t), \quad k \geq 2.$$

Since ζ is subadditive, *i.e.* $\zeta(t + t') \leq \zeta(t) + \zeta(t')$, we have

$$H^{\circ k}\left(\frac{x_1 + x_2}{2}\right) \leq H^{\circ k}\left(\frac{x_1}{2}\right) + H^{\circ k}\left(\frac{x_2}{2}\right) = H^{\circ(k-1)}(x_1) + H^{\circ(k-1)}(x_2). \quad (8.13)$$

We now replace each term $H^{\circ k}((x_1 + x_2)/2)$ appearing in (8.12) with the right-hand side of (8.13). The result is a telescoping sum. Since $H(t/2) = t$, $t \in [0, 1]$, this telescoping sum gives

$$2\Pi_n(x_1, x_2) \geq 4^{-n}[H^{\circ n}(x_1) + H^{\circ n}(x_2)] \geq 0,$$

as desired. □

Next we observe that Π_n approximates Π with exponential accuracy.

Proposition 8.6. For each $n \geq 1$, the function $\Pi_n \in \Upsilon^{5,3n}(\mathrm{ReLU}; 2, 1)$ satisfies the inequalities

$$\|\Pi - \Pi_n\|_{C([0,1]^2)} \leq 4^{-n}$$

and

$$\|\partial_i\Pi - \partial_i\Pi_n\|_{L_\infty([0,1]^2)} \leq 2 \cdot 2^{-n}, \quad \text{where } \partial_i := \partial_{x_i},\ i = 1, 2. \quad (8.14)$$

Proof. Let $\mathcal{N}$ be the network of width $W = 4$ and depth n which outputs S_n; see Proposition 8.4. We now construct a network $\mathcal{N}'$ which inputs (x_1, x_2) and

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

outputs Π_n. First we add a source channel to $\mathcal{N}$ to push forward x_2 ($\mathcal{N}$ already has a source channel to push x_1). Then we place three copies of this extended network next to each other. We output the three terms from (8.10) in the collation channel of $\mathcal{N}$, and produce $\Pi_n(x_1, x_2)$. The new network has width $W = 5$ and depth $L = 3n$. From (8.7), we have $\|\Pi - \Pi_n\|_{C([0,1]^2)} \le 4^{-n}$.

Finally we check (8.14) for $i = 1$. The case $i = 2$ is the same. We have $\partial_1 \Pi(x_1, x_2) = x_2$, and modulo a set of measure zero,

$$\begin{aligned}\partial_1 \Pi_n(x_1, x_2) &= S_n'\left(\frac{x_1 + x_2}{2}\right) - \frac{1}{2} S_n'(x_1) \\ &= x_1 + x_2 + \varepsilon_1 - \frac{1}{2}(2x_1 + \varepsilon_2) \\ &= x_2 + \varepsilon_1 - \varepsilon_2 \\ &= \partial_1 \Pi(x_1, x_2) + \varepsilon_1 - \varepsilon_2 \\ &\le 2^{-n+1},\end{aligned}$$

where $|\varepsilon_1|, |\varepsilon_2| \le 2^{-n}$ because of (8.8). The proof is completed. □

In general, we can approximate any product

$$\Pi^k(x_1, \dots, x_k) := x_1 x_2 \cdots x_k, \quad x_1, \dots, x_k \in [0, 1],$$

up to exponential accuracy, using outputs of ReLU neural networks. We write

$$\Pi^{k+1}(x_1, \dots, x_{k+1}) = \Pi(x_{k+1}, \Pi^k(x_1, \dots, x_k)),$$

denote $\Pi_n^2 := \Pi_n$ (see (8.11)) and recursively define

$$\Pi_n^{k+1}(x_1, \dots, x_{k+1}) = \Pi_n(x_{k+1}, \Pi_n^k(x_1, \dots, x_k)), \quad k = 2, 3, \dots.$$

It follows by induction, using Proposition 8.5, that $\Pi^k(x_1, \dots, x_k) \in [0, 1]$, and therefore Π^{k+1} is well-defined. Then the following theorem holds.

Theorem 8.7. For each $k \ge 2$, the function $\Pi_n^k \in \Upsilon^{3+k, 3(k-1)n}(\mathrm{ReLU}; k, 1)$ satisfies

$$|\Pi^k(x_1, \dots, x_k) - \Pi_n^k(x_1, \dots, x_k)| \le C_k \cdot 4^{-n}, \quad x_1, \dots, x_k \in [0, 1], \tag{8.15}$$

where, for $k \ge 2$,

$$C_k \le (k - 1)\alpha_n^{k-2}, \quad \alpha_n := 1 + 2^{-n+1}.$$

In particular, $C_k \le ek$, as long as $n \ge 1 + \log_2 k$.

Proof. For $k \ge 3$, we construct a network of width $k + 2$ which takes the inputs $x_1, \dots, x_{k-1}$ and outputs Π_n^{k-1} (when $k = 3$, this is the network for Π_n^2). Its first $3n(k - 2)$ layers are the same as the network that inputs $x_1, \dots, x_{k-1}$ and outputs Π_n^{k-1}, except that we add an additional channel to push forward x_k. We then follow this with the network for Π_n^2 using as inputs x_k and $\Pi_n^{k-1}(x_1, \dots, x_{k-1})$. This network will have width $W = k + 3$ and depth $L = 3(k - 1)n$, as desired.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Next, we fix n and prove (8.15) by induction on k. The case $k = 2$ is covered by Proposition 8.6 with $C_2 = 1$. To advance the induction hypothesis, we assume that we have proved the result for some $k - 1$ with constant C_{k-1}. We write (with the obvious abbreviation of notation)

$$\Pi^k - \Pi_n^k = \Pi(x_k, \Pi^{k-1}) - \Pi_n(x_k, \Pi^{k-1}) + \Pi_n(x_k, \Pi^{k-1}) - \Pi_n(x_k, \Pi_n^{k-1}),$$

and use Proposition 8.6 to obtain

$$\|\Pi^k - \Pi_n^k\|_{C([0,1]^k)} \le 4^{-n} + \|\partial_2 \Pi_n\|_{L_\infty([0,1]^2)} \|\Pi^{k-1} - \Pi_n^{k-1}\|_{C([0,1]^{k-1})}. \tag{8.16}$$

We now use (8.14) to conclude that $\|\partial_2 \Pi_n\|_{L_\infty([0,1]^2)} \le 1 + 2 \cdot 2^{-n}$. Inserting this into (8.16) gives

$$\|\Pi^k - \Pi_n^k\|_{C([0,1]^k)} \le 4^{-n} + (1 + 2^{-n+1})\, C_{k-1} 4^{-n} = (1 + \alpha_n C_{k-1})\, 4^{-n}.$$

The recurrence formula $C_k = 1 + \alpha_n C_{k-1}$, $k \ge 3$, with initial value $C_2 = 1$, has the solution

$$C_k = \sum_{j=0}^{k-2} \alpha_n^j \le (k-1)\alpha_n^{k-2}.$$

Moreover, if $k \le 2^{n-1}$, we have

$$C_k \le (k-1)\left(1 + \frac{1}{2^{n-1}}\right)^{k-2} < k\left(1 + \frac{1}{k}\right)^k < ek.$$

This completes the proof of the theorem. □

Remark 8.1. If $a > 1$, then a simple change of variables gives that the function Π^k, $k \ge 2$, now considered as a function in $C([0,a]^k)$, is approximated by $\tilde{\Pi}_n^k(x_1, \dots, x_k) := a^k \Pi_n^k(x_1/a, \dots, x_k/a)$ with accuracy

$$\|\Pi^k - \tilde{\Pi}_n^k\|_{C([0,a]^k)} \le C_k a^k \cdot 4^{-n},$$

with C_k as in Theorem 8.7. Moreover $\tilde{\Pi}_n^k \in \Upsilon^{3+k, 3(k-1)n}(\mathrm{ReLU}; k, 1)$.

8.3. Approximation of polynomials

In the following, we show that polynomials can be well approximated by the outputs of deep ReLU networks. We begin with monomials.

Approximation of monomials. For $\nu \in \mathbb{N}^d$, let $\phi_\nu(x) := x^\nu$, $x \in [0,1]^d$. We use the standard notation $|\nu| = \sum_{j=1}^d \nu_j$ for the length of ν. Theorem 8.7 shows that any monomial ϕ_ν, with $|\nu| = m$, is well approximated by deep ReLU networks. Namely, for each $n \ge 1$, there is an $S_\nu \in \Upsilon^{3+d, 3(m-1)n}(\mathrm{ReLU}; d, 1)$ such that

$$\|\phi_\nu - S_\nu\|_{C([0,1]^d)} \le em \cdot 4^{-n}, \quad n \ge 1 + \log_2 m. \tag{8.17}$$

Note that here we can keep the width of the network bounded by $3 + d$ rather than $3 + m$ because the $x_1, \dots, x_d$ are repeated; we leave the details to the reader.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Approximation of polynomials. If $P(x) = \sum_{\nu\in\Lambda} c_\nu x^\nu$, $x \in [0,1]^d$, where all of the indices $\nu \in \Lambda$ satisfy $|\nu| \le m$, then we can approximate P by the function $S := \sum_{\nu\in\Lambda} c_\nu S_\nu$. Note that S is the output of the concatenation of the networks that output the S_ν's; see **Addition by increasing depth** in Section 3.3.2. Since all networks producing the S_ν's already have source channels that forward the values $x_1, \dots, x_d$, we need to add only a collation channel to collect the terms in the sum defining S, and therefore $S \in \Upsilon^{4+d,3(m-1)n\#(\Lambda)}(\mathrm{ReLU}; d, 1)$. We have the following estimate for the approximation error:

$$\|P - S\|_{C([0,1]^d)} \le em \cdot 4^{-n} \sum_{\nu\in\Lambda} |c_\nu|, \quad n \ge 1 + \log_2 m, \tag{8.18}$$

obtained from (8.17). There are several savings that can be made in the size of the network in such constructions by balancing the size of c_ν with the size of the networks for the S_ν when given a desired target accuracy.

Constructions of NN approximations to polynomial sums have been employed to prove results on approximating real analytic functions using deep ReLU networks. We do not formulate those results here but rather refer the reader to the papers by Opschoor *et al.* (2019*b*) and E and Wang (2018) for statements and proofs.

8.4. Approximation of tensor products

Tensor structures are a very effective method for approximation in high dimensions. It is beyond the scope of this article to lay this subject out in its full detail. We simply wish to point out here that a rank-one tensor product

$$f(x_1, \dots, x_d) = f_1(x_1) \cdots f_d(x_d), \quad x_1, \dots, x_d \in [0,1], \tag{8.19}$$

is well approximated in $C(\Omega)$, $\Omega = [0,1]^d$, whenever the univariate components f_j are well approximated. The starting point for this is the following simple proposition.

Proposition 8.8. If $g_j : [0,1] \to [0,1]$, $g_j \in \Upsilon^{W_0,L_0}(\mathrm{ReLU}; 1, 1)$, $W_0 \ge 3$, for $j = 1, \dots, d$, then the rank-one tensor

$$g(x_1, \dots, x_d) = g_1(x_1) \cdots g_d(x_d) \tag{8.20}$$

can be approximated by $S \in \Upsilon^{dW_0, L_0+3(d-1)n}(\mathrm{ReLU}; d, 1)$ to accuracy

$$\|g - S\|_{C(\Omega)} \le ed \cdot 4^{-n}, \quad n \ge 1 + \log_2 d. \tag{8.21}$$

Proof. We can take $S := \Pi_n^d(g_1, \dots, g_d)$. We claim that S is an element of $\Upsilon^{dW_0, L_0+3(d-1)n}(\mathrm{ReLU}; d, 1)$. Indeed, we stack the networks producing the g_j's, $j = 1, \dots, d$ on top of each other and end up with a network $\mathcal{N}_1$ with width dW_0 and depth L_0. Then we concatenate it with the network $\mathcal{N}_2$ producing $\Pi_n^d(y_1, \dots, y_d)$. The latter has depth $3(d-1)n$ and width $3 + d \le dW_0$. The concatenation is done by forwarding the output of the network producing g_j as an input to the jth channel

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

of $\mathcal{N}_2$ (the first d channels of $\mathcal{N}_2$ are source channels). We end up with a network with the desired width and depth. Inequality (8.21) follows from Theorem 8.7. □

Remark 8.2. We make two remarks on the above proposition:

- If $0 \le g_j(t) \le M$ instead of $0 \le g_j(t) \le 1$, then by using Remark 8.1 we obtain an S in the same ReLU space but the accuracy of approximation is now lessened by the factor M^d.
- If the g_j's are not in the designated ReLU space but are only approximated by $\hat{g}_j : [0,1] \to [0,1]$, $j = 1, \dots, d$, from the designated ReLU space to an accuracy ε, then the function $S := \Pi_n^d(\hat{g}_1, \dots, \hat{g}_d)$ is in the designated ReLU space and we can write

$$\begin{aligned} \|g - S\|_{C(\Omega)} &\le \|g - \Pi(\hat{g}_1 \cdots \hat{g}_d)\|_{C(\Omega)} + \|[\Pi - \Pi_n](\hat{g}_1 \cdots \hat{g}_d)\|_{C(\Omega)} \\ &\le d\varepsilon + ed \cdot 4^{-n}, \quad n \ge 1 + \log_2 d, \end{aligned} \tag{8.22}$$

  where the first term does not exceed the sum of the d errors

$$\|\hat{g}_1 \cdots \hat{g}_j \cdot g_{j+1} \cdots g_d - \hat{g}_1 \cdots \hat{g}_{j+1} \cdot g_{j+2} \cdots g_d\|_{C(\Omega)} \le \varepsilon. \tag{8.23}$$

  In particular, the result also holds for the univariate product

$$g(t) = g_1(t) g_2(t) \cdots g_d(t).$$

8.5. Approximation of B-splines

In our presentation of classical smoothness classes K of functions given in Section 4, we have stressed that the elements in K have certain atomic decompositions and their membership in K is characterized by the decay of their coefficients in such representations. Thus, if we can show that the atoms in such a decomposition are well approximated by NNs, then we can obtain bounds for NN approximation of K. The aim of the present section is to show how this unfolds when we choose B-splines as the atomic representation system.

Let N_r be the univariate B-spline defined in (4.2). Let us recall that N_r is supported on $[0, r]$ and is normalized so that $\|N_r\|_{C(\mathbb{R})} = 1$.

Proposition 8.9. Let $r \ge 2$ and consider the B-spline

$$N(x) := N_r(x_1) \cdots N_r(x_d)$$

of d variables. There is a function $\hat{N} \in \Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ with width $W = 6d$ and depth $L = Cn$, with C depending only on r and d, which satisfies

$$\|N - \hat{N}\|_{C(\mathbb{R}^d)} \le C'(r,d) 4^{-n}, \quad n \ge 1, \tag{8.24}$$

with the constant $C'(r,d)$ depending only on r and d. Moreover, the support of $\hat{N}$ is contained in that of N.

Proof. This is proved by approximating in succession the functions

$$t^{r-1}, \quad \rho_{r-1}(t) := t_+^{r-1}, \quad N_r(t), \quad N(x_1, \dots, x_d) = N_r(x_1) \cdots N_r(x_d), \tag{8.25}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

where N_r is the univariate B-spline. Our results of the previous sections on approximating products were stated for approximation on $[0,1]^d$ and now we want approximation on $[0,r]^d$. This is done by using Remark 8.1, and changes the estimates by a constant depending only on r and d. We assume such changes without further elaboration in what follows. All constants C appearing in the proof depend at most on r and d.

Because of (8.17), we can approximate the function t^{r-1} by an element of $\Upsilon^{4,3(r-2)n}(\mathrm{ReLU};1,1)$ with an error that does not exceed $C4^{-n}$. By adding an extra layer for ReLU, we can approximate the function ρ_{r-1} by an S_r from $\Upsilon^{4,3(r-2)n+1}(\mathrm{ReLU};1,1)$ with accuracy

$$\|\rho_{r-1} - S_r\|_{C([0,r])} \le C4^{-n}. \tag{8.26}$$

Next, we use S_r to approximate the univariate B-spline N_r by replacing $\rho_{r-1}(k-t)$ with $S_r(k-t)$ in formula (4.2). The resulting function T_r (see Theorem 8.3) is in $\Upsilon^{6,3(r+1)(r-2)n+r+1}(\mathrm{ReLU};1,1)$ (note that the network producing the latter set has one source channel for t and one collation channel). In addition, we have

$$\|N_r - T_r\|_{C([0,r])} \le C4^{-n} \quad \text{and} \quad \|T_r\|_{C([0,r])} \le 1 + C4^{-n}. \tag{8.27}$$

We next consider $T_r^+ := \mathrm{ReLU}(T_r)$ which also satisfies (8.27), with the additional property $T_r^+ \ge 0$.

Remark 8.2 with $\varepsilon = C4^{-n}$ and $M \le (1 + C/4)$ gives that N can be approximated by $\tilde{N} = \Pi_n^d(T_r^+, \dots, T_r^+)$ with accuracy

$$\|N - \tilde{N}\|_{C([0,r]^d)} \le C4^{-n}, \quad n \ge 1 + \log_2 d. \tag{8.28}$$

The approximant $\tilde{N} \in \Upsilon^{W,L}(\mathrm{ReLU};d,1)$ with width $W = 6d$ and depth $L = 3(r+1)(r-2)n + r + 2 + 3(d-1)n$. The network producing $\tilde{N}$ has d source channels for each of the variables x_i, $i = 1, \dots, d$, and d collation channels.

Finally, we modify the function $\tilde{N}$ of (8.28) so that it vanishes outside $[0,r]^d$. This is done by what should by now be a familiar technique to the reader. We construct a function S with support $[0,r]^d$ and $S \ge N_r$, using the method for the construction of nodal functions; see (3.19). More precisely,

$$S := \mathrm{ReLU}(\min\{\ell_1, \dots, \ell_{2^d}\}) \in \Upsilon^{d+1,2^d}(\mathrm{ReLU};d,1)$$

(see MM2 in Section 3.3.2), where ℓ_j, $j = 1, \dots, 2^d$, are affine functions, each of which vanishes on one of the 2^d facets of the cube $[0,r]^d$ and is above the graph of the B-spline N. Then the function $\hat{N} := \mathrm{ReLU}(\min\{\tilde{N}, S\})$ agrees with $\tilde{N}$ when $\tilde{N}$ is non-negative and vanishes outside $[0,r]^d$. Therefore it satisfies all the properties of the theorem. Since both networks producing $\tilde{N}$ and S have source and collation channels, it follows that $\hat{N} \in \Upsilon^{6d,Cn}(\mathrm{ReLU};d,1)$ with C depending only on r and d. □

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

8.6. *Approximation of Besov classes with deep ReLU networks*

With the results of the previous section on B-spline approximation in hand, we can now show that deep neural networks are at least as effective as standard nonlinear methods (modulo logarithms), such as adaptive FEMs or n-term wavelets, when approximating the classical smoothness spaces (Sobolev and Besov). The ideas used in the presentation below were put forward by Ali and Nouy (2020), Bölcskei *et al.* (2019) and Gribonval *et al.* (2019).

We fix $\Omega = [0,1]^d$ and measure the approximation error in some $L_p(\Omega)$ norm, $1 \le p \le \infty$. Let $K = U(B^s_q(L_\tau(\Omega)))$ be the unit ball of a Besov space that lies above the Sobolev embedding line for $L_p(\Omega)$, and therefore is a compact subset of $L_p(\Omega)$. Classical methods of nonlinear approximation show that K can be approximated in $L_p(\Omega)$ to accuracy $O(n^{-s/d})$, where n is the number of parameters used in the approximation. We now show how to achieve these rates when using $\Sigma_n := \Upsilon^{W_0,n}(\mathrm{ReLU}; d, 1)$ with W_0 fixed and depending only on s and d. Note that the number of parameters needed to describe the elements in Σ_n is at most $C(s,d)n$. Here and later in this section, the constant $C(s,d)$ changes at each occurrence.

Theorem 8.10. Let $s > 0$ and $\Omega = [0,1]^d$. Suppose

$$K = U(B^s_q(L_\tau(\Omega))), \quad 0 < q, \tau \le \infty,$$

is the unit ball of a Besov space lying above the Sobolev embedding line for $L_p(\Omega)$ with $1 \le p \le \infty$, that is,

$$\delta := s - \frac{d}{\tau} + \frac{d}{p} > 0.$$

Then we have

$$E(K, \Sigma_{n[\log_2 n]^\beta})_{L_p(\Omega)} \le C(s,d,\delta) n^{-s/d}, \quad n \ge 1, \tag{8.29}$$

where $\Sigma_n := \Upsilon^{6d, Cn}(\mathrm{ReLU}; d, 1)$, $n \ge 1$, with $C = C(s,d,\delta)$ fixed, depending only on s, d and δ, and where $\beta := \max\{1, 2d/(s-\delta)\}$.

Proof. We only treat the case $1 \le p < \infty$ and leave it to the reader to make the necessary changes for $p = \infty$. We fix p and $s > 0$. We can assume $q = \infty$ since this is the largest unit ball for the given τ, and $\tau < p$. We can further assume that $\delta > 0$ is arbitrarily small since the Besov spaces of order s get larger as we approach the Sobolev embedding line which corresponds to $\delta = 0$.

To prove the theorem, it is sufficient to prove that it holds for $n = 2^L$ with L a sufficiently large positive integer. We take $r = \lceil s \rceil + 1$ and let N denote the multivariate tensor product B-spline of order r. We recall the notation $\mathcal{D}(\Omega)$ for the collection of all dyadic cubes I such that N_I is non-zero on Ω, $\mathcal{D}_k(\Omega)$ for these cubes at dyadic level k (they have measure 2^{-kd}), and $\mathcal{D}_+(\Omega) := \cup_{k \ge 0} \mathcal{D}_k(\Omega)$.

From (4.5) and (4.6), we know that any $f \in K$ has the representation

$$f = \sum_{I \in \mathcal{D}_+(\Omega)} c_I(f) N_I, \tag{8.30}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

with

$$\sum_{I\in\mathcal{D}_k(\Omega)} |c_I(f)|^\tau |I| \le 2^{-ks\tau}, \quad k = 0, 1, 2, \dots. \tag{8.31}$$

Here we use (4.6) for the definition of the norm in the Besov space. For each $j \in \mathbb{Z}$ and $k \ge 0$, we define

$$\Lambda(j,k) := \{I \in \mathcal{D}_k(\Omega)\colon 2^{-j} \le |c_I(f)| < 2^{-j+1}\}, \tag{8.32}$$

and estimate its cardinality from (8.31). We derive that

$$\sum_{j=-\infty}^{\infty} 2^{-j\tau} \#(\Lambda(j,k)) \le 2^{k(d-s\tau)}, \quad k = 0, 1, \dots. \tag{8.33}$$

It follows from (8.33) that if $\Lambda(j,k) \ne \emptyset$, then $2^{-j\tau} \le 2^{k(d-s\tau)}$, and therefore

$$j \ge \left(s - \frac{d}{\tau}\right)k = \left(\delta - \frac{d}{p}\right)k =: J_k. \tag{8.34}$$

In other words,

$$\Lambda(j,k) = \emptyset, \quad \text{when } j < J_k. \tag{8.35}$$

We will now replace some of the N_I's from (8.30) with approximants $\hat{N}_I$ from $\Sigma_{m(I)}$, where the non-negative integers $m(I) := m(j,k) \in \{1, 2, \dots\}$ will be chosen the same for each $I \in \Lambda(j,k)$ (as we shall see below). The N_I's that are not approximated are associated with $m(I) = 0$.

It follows from (8.24) that

$$\|N_I - \hat{N}_I\|_{L_p(\Omega)} \le C|I|^{1/p} 4^{-m(I)}, \tag{8.36}$$

where here and later in this proof all constants C depend only on s, d and δ. According to Proposition 8.9, we can also assume that $\hat{N}_I$ is zero outside the support of N_I.

Next we define the functions

$$\hat{S} := \sum_{I\in\mathcal{D}_+(\Omega), m(I)>0} c_I(f)\hat{N}_I, \quad \hat{S}_k := \sum_{I\in\mathcal{D}_k(\Omega), m(I)>0} c_I(f)\hat{N}_I, \quad k \ge 0, \tag{8.37}$$

and proceed to show that $\hat{S}$ provides the needed approximation if we choose $m(I)$ appropriately.

In preparation for the choice of the $m(I)$, we first estimate how well $\hat{S}$ approximates f. If we let

$$S_k := \sum_{I\in\mathcal{D}_k(\Omega)} c_I(f)N_I, \quad k \ge 0,$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

using (8.36) and the fact that $\|N_I\|_{L_p(\Omega)} \le C|I|^{1/p}$, we obtain

$$\begin{aligned}\|S_k - \hat{S}_k\|_{L_p(\Omega)}^p &\le C^p \sum_{I\in\mathcal{D}_k(\Omega)} |c_I(f)|^p |I| 4^{-m(I)p} \\ &\le C^p 2^{-kd} \sum_{j\ge J_k} 2^{-jp} \#(\Lambda(j,k)) 4^{-m(j,k)p} \\ &= C^p 2^{-kd} \sum_{j\ge J_k} 2^{-j\tau} \#(\Lambda(j,k)) 2^{-2m(j,k)p - jp + j\tau}.\end{aligned}$$

For the definition of $m(j,k)$, let us introduce the notation

$$\varepsilon_t := 2\log_2(t+1), \quad t \ge 0. \tag{8.38}$$

For every $j \ge J_k$, $k \ge 0$, we choose $m(j,k)$ to be the smallest non-negative integer such that

$$\varepsilon_k + Lsp/d \le ks\tau + [2m(j,k)+2]p + j(p-\tau).$$

This choice satisfies

$$0 \le 2m(j,k)p \le [\varepsilon_k p + Lsp/d - jp - (ks-j)\tau]_+. \tag{8.39}$$

Then we obtain

$$\begin{aligned}\|S_k - \hat{S}_k\|_{L_p(\Omega)}^p &\le C^p 2^{-kd} 2^{-\varepsilon_k p - Lsp/d + ks\tau} \sum_{j\ge J_k} 2^{-j\tau} \#(\Lambda(j,k)) \\ &\le C^p 2^{-Lsp/d} (k+1)^{-2p},\end{aligned}$$

where we used (8.33) for the last inequality. It then follows from (8.30) that

$$\|f - \hat{S}\|_{L_p(\Omega)} \le \sum_{k=0}^{\infty} \|S_k - \hat{S}_k\|_{L_p(\Omega)} \le C 2^{-Ls/d} \sum_{k=0}^{\infty} (k+1)^{-2} = C n^{-s/d}.$$

We are left to show that $\hat{S} \in \Sigma_{L^\beta 2^L}$, which in turn proves the theorem. We know that $\hat{N}_I \in \Upsilon^{6d, Cm(I)}(\mathrm{ReLU}; d, 1)$. Since the network producing $\hat{N}_I$ already has d source channels and a collation channel, our **Addition by increasing depth** in Section 3.3.2 gives that $\hat{S} \in \Upsilon^{6d, CA}(\mathrm{ReLU}; d, 1)$, where

$$A := \sum_{k=0}^{\infty} \sum_{j=J_k}^{J_k^+} m(j,k) \#(\Lambda(j,k)). \tag{8.40}$$

The index in the second sum in (8.40) has upper bound J_k^+, where J_k^+ is defined by the equation

$$J_k^+(1-\tau/p) + ks\tau/p = \varepsilon_k + Ls/d, \tag{8.41}$$

because $m(j,k) = 0$ when $j \ge J_k^+$; see (8.39). Later we shall use the fact that

$$\tau J_k^+ = \lambda(\varepsilon_k + Ls/d - ks\tau/p), \quad \text{with } \lambda := (1/\tau - 1/p)^{-1} = \frac{d}{s-\delta}. \tag{8.42}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

For $j \in [J_k, J_k^+]$, $m(j,k)$ takes its maximum value at $j = J_k$, which is

$$\begin{aligned} m(J_k,k) &\le \frac{1}{2}(\varepsilon_k + Ls/d - ks\tau/p - J_k(1-\tau/p)) \\ &= \frac{1}{2}(2\log_2(k+1) - k\delta + Ls/d) \\ &\le CL, \end{aligned}$$

where we used the definition of J_k and (8.39). Therefore we have the estimate

$$A \le CL \sum_{k=0}^{\infty} \sum_{j=J_k}^{J_k^+} \#(\Lambda(j,k)) \le CL \sum_{k=0}^{L/d-1} 2^{kd} + CL \sum_{k=L/d}^{\infty} 2^{kd - ks\tau + J_k^+\tau}, \tag{8.43}$$

where in the first sum we used the fact that

$$\sum_{j=J_k}^{J_k^+} \#(\Lambda(j,k)) \le C2^{kd},$$

because $\Lambda(j,k) \subset \mathcal{D}_k(\Omega)$, and the second sum used that

$$\begin{aligned} \sum_{j=J_k}^{J_k^+} \#(\Lambda(j,k)) &\le 2^{J_k^+\tau} \sum_{j=J_k}^{J_k^+} 2^{-j\tau} \#(\Lambda(j,k)) \\ &\le 2^{J_k^+\tau} \sum_{j=J_k}^{\infty} 2^{-j\tau} \#(\Lambda(j,k)) \\ &\le 2^{kd - ks\tau + J_k^+\tau}. \end{aligned}$$

Obviously the first sum on the right does not exceed $CL2^L$, so we concentrate on the second sum. We first want to see what the exponent is in that sum. From (8.42), we have

$$kd - ks\tau + J_k^+\tau = \lambda(Ls/d + \varepsilon_k) + k\{d - s\tau(1+\lambda/p)\}. \tag{8.44}$$

Going further, we find

$$\begin{aligned} d - s\tau(1+\lambda/p) &= d - s\tau\left(1 + \frac{1}{p(1/\tau - 1/p)}\right) \\ &= d - s\tau\left(1 + \frac{1}{p/\tau - 1}\right) \\ &= d - \frac{sp}{p/\tau - 1} \\ &= d - \frac{s}{1/\tau - 1/p} \end{aligned}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

$$= \frac{d(1/\tau - 1/p) - s}{1/\tau - 1/p}$$
$$= \frac{-\delta}{1/\tau - 1/p}$$
$$= -\delta\lambda,$$

and thus

$$kd - ks\tau + J_k^+\tau = \lambda(Ls/d + \varepsilon_k - \delta k).$$

We substitute the latter relation into (8.43) and obtain, after change of index $i = k - L/d$ and using (8.42),

$$\begin{aligned} A &\le CL2^L + CL \sum_{k=L/d}^{\infty} 2^{\lambda(Ls/d+\varepsilon_k-\delta k)} \\ &= CL2^L + \sum_{i=0}^{\infty} 2^{\lambda(s-\delta)L/d} 2^{2\lambda \log_2(i+L/d+1)-i\lambda\delta} \\ &= CL2^L + 2^L \sum_{i=0}^{\infty} (i + L/d + 1)^{2\lambda} 2^{-i\lambda\delta} \\ &< CL2^L + CL^{2\lambda}2^L \\ &< CL^{\beta}2^L. \end{aligned}$$

This gives the bound we want and proves the theorem. □

Before proceeding, we make the following remarks concerning the above theorem and its proof.

Remark 8.3. The above result is not quite as good as the results for approximating unit balls of Besov classes when using other methods of nonlinear approximation (see DeVore 1998) because of the appearance of the logarithm. We should mention that Ali and Nouy (2020) have proved a result similar to the above theorem by using spline wavelets rather than B-splines as the main vehicle. In the next section we show that when $p = \infty$ this logarithm does not appear, and in fact we can prove much better rates of approximation. These can in turn be used to improve the above results for all Besov classes, as we shall discuss at the end of the next section. The determination of the best approximation rates for L_p approximation when using the outputs of deep networks such as $\Sigma_n = \Upsilon^{W_0,n}(\mathrm{ReLU}; d, 1)$ remains unsettled.

8.7. *Super-convergence for deep ReLU networks*

In this section we present some very intriguing results on approximation by NNs that show quite unexpected rates of approximation for certain classical model classes K described by smoothness. The initial results were given in Yarotsky (2018) for the model classes $U(\mathrm{Lip}\,\alpha)$, $0 < \alpha \le 1$ on $\Omega = [0, 1]^d$, and were later extended

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

to more general model classes $U(C^s(\Omega))$, $s > 0$, in Lu *et al.* (2020). Our aim in this section is to show how these super rates are established in the simple case of univariate functions in Lip 1, and leave the reader to consult the above references for the treatment for functions of d variables and higher smoothness. At the end of this section we put these results into perspective and formulate some related questions.

Theorem 8.11. If $K = U(\text{Lip}\,1)$ on $\Omega := [0,1]$ and $\Sigma := (\Sigma_n)_{n\geq 1}$ with $\Sigma_n := \Upsilon^{11,16n+2}(\text{ReLU}; 1, 1)$, $n \geq 1$, then we have

$$E(K, \Sigma_n)_{C(\Omega)} \leq 6n^{-2}, \quad n \geq 1. \tag{8.45}$$

Proof. We use the same notation as in Section 3.6.3. Namely, we define $N := n^2$, with $n \geq 4$ an even positive integer and set $t_i := i/N$, $0 \leq i \leq N$, and $\xi_j := j/n$, $j = 0, \ldots, n$. Given $f \in K$, as a first step, we take S_0 to be the CPwL function with breakpoints precisely the ξ_j's, $j = 1, \ldots, n-1$, which interpolates f at the ξ_j, $j = 0, \ldots, n$. Then S_0 has the following three properties:

(i) $\|f - S_0\|_{C(\Omega)} \leq 1/n, \quad n \geq 4.$

(ii) $\|S_0\|_{\text{Lip}\,1} \leq 1, \quad n \geq 4.$

(iii) $S_0 \in \Upsilon^{3,n}(\text{ReLU}; 1, 1)$; see Proposition 3.9.

Now consider the function $R := f - S_0$. It vanishes at each of the ξ_j, $j = 0, \ldots, n$, and $R \in \text{Lip}\,1$ with $\|R\|_{\text{Lip}\,1} \leq 2$. We next show that there is a sequence of $\varepsilon_i \in \{-1, +1\}$, $i = 0, \ldots, N-1$, such that $(y_i)_{i=0}^N$, defined recursively by $y_0 := 0$ and

$$y_{i+1} := y_i + \varepsilon_i, \quad i = 0, \ldots, N-1, \tag{8.46}$$

satisfy

$$y_{jn} = 0, \quad j = 0, \ldots, n \quad \text{and} \quad \left| R(t_i) - \frac{2y_i}{N} \right| \leq \frac{2}{N}, \quad 0 \leq i \leq N. \tag{8.47}$$

Let us for the moment assume we have found such a sequence $(\varepsilon_i)_{i=0}^{N-1}$ and show how to complete the proof of the theorem. We apply Theorem 3.13 to the $(y_i)_{i=0}^N$ defined in (8.46) and obtain a function $S_1 \in \Upsilon^{11,15n+2}(\text{ReLU}; 1, 1)$ with the properties guaranteed by this theorem. Next we consider the function

$$S := S_0 + \frac{2}{N} S_1.$$

Note that $S \in \Upsilon^{11,16n+2}(\text{ReLU}; 1, 1)$ because of **Addition by increasing depth** in Section 3.3.2, taking into account that the network from Theorem 3.13 producing S_1 already has a source and collation channel. Moreover,

$$\|f - S\|_{C([0,1])} = \left\| R - \frac{2}{N} S_1 \right\|_{C([0,1])} \leq 6/N. \tag{8.48}$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Indeed, if $t \in [t_i, t_{i+1}]$, $i = 0, \dots, N-1$, then $S_1(t_i) = y_i$, and we have

$$\left| R(t) - \frac{2}{N} S_1(t) \right| \le |R(t) - R(t_i)| + \left| R(t_i) - \frac{2}{N} S_1(t_i) \right| + \frac{2}{N} |S_1(t) - S_1(t_i)| \le 6/N,$$

because of the Lipschitz properties of R, the properties of S_1, and (8.47). This in turn would prove the theorem.

So we are left with finding a sequence $(\varepsilon_i)_{i=0}^{N-1}$ such that (8.47) is valid. It is enough to show how to define this sequence for $i = 0, \dots, n-1$ since for $i = jn, \dots, (j+1)n - 1$ it is defined similarly. We choose the sequence $\varepsilon_0, \varepsilon_1, \dots$ and the corresponding $y_{j+1} := y_j + \varepsilon_j$ and verify (8.47) recursively. We first choose $\varepsilon_0 \in \{-1, 1\}$ so that $2\varepsilon_0/N$ is closest to $R(t_1)$ for this choice of the two possible values ± 1. Clearly, since $|R(t_1)| \le 2/N$, for $y_1 := \varepsilon_0$ we have the inequality $|R(t_1) - 2y_1/N| \le 2/N$. In other words we have verified (8.47) for $i = 1$.

Assume now that $\varepsilon_0, \dots, \varepsilon_{j-1}$ have been chosen and the corresponding $y_1, \dots, y_j$ have been shown to satisfy (8.47). We now choose ε_j so that

$$\frac{2y_{j+1}}{N} = \frac{2(y_j + \varepsilon_j)}{N}$$

is closest to $R(t_{j+1})$. Since R changes by at most $2/N$ in moving from t_j to t_{j+1}, this choice will also satisfy (8.47). So we are left to verify that $y_n = 0$. Since n is even, $y_n = \varepsilon_0 + \cdots + \varepsilon_{n-1} = 2m$ for some integer m. In addition, we have $|2y_n/N - 0| \le 2/N$, and therefore we must have $m = 0$. Thus we have shown the existence of a sequence $(\varepsilon_i)_{i=0}^{N-1}$ with the required properties. The proof of the theorem is completed. □

8.7.1. Remarks on Theorem 8.11

We make some remarks on this theorem in order to put into perspective what it is saying. In this section we take $\Sigma := (\Sigma_n)$, where the sets Σ_n used for approximation are $\Sigma_n = \Upsilon^{W_0, Cn}(\mathrm{ReLU}; d, 1)$ with W_0 and C fixed and depending only on d.

At first glance this theorem is very surprising to numerical analysts and approximation theorists since it is giving a rate of approximation $O(n^{-2})$, $n \ge 1$, whose exponent is twice that given by standard approximation methods based on n parameters. This indicates that the nonlinear manifold $\Sigma_n := \Upsilon^{W_0, Cn}$ has certain space-filling properties in $X = C(\Omega)$. While this seems like a great advantage of this manifold, recall that there are always one-parameter manifolds which are dense in X, albeit not as neatly described as Σ_n. But then we must throw in some caution. The theorem says that given $f \in K$, there is a mapping $a \colon K \to \mathbb{R}^n$ which selects the parameters $a(f)$ of the approximant that produces this exceptional approximation performance. From our remarks in Section 5 on manifold width, the mapping a cannot be continuous (note that the mapping M is always continuous, as will be discussed in more detail in the next section). This shows a lack of numerical stability in the approximation process which yields Theorem 8.11. This means that we can expect that it will be very difficult to numerically find the parameters that

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

attain the super-convergence rate via a search over parameter domain. On the other hand, if we are willing to allow a long enough search time with an *a posteriori* error estimator, we might be able to find such parameters.

In spite of the negative comments just put forward, the theorem is intriguing and brings up several questions that we now discuss. The first natural question is: In what generality does this super-convergence hold? We have already mentioned that Yarotsky proved it for multivariate functions of d variables. He also proved a general result which gives that the theorem holds for $\mathrm{Lip}\,\alpha$ spaces, $0 < \alpha \le 1$. A generalization of this theorem is provided in Lu *et al.* (2020). It shows that the set $K = U(\mathrm{Lip}\,1)$ can be replaced by the unit ball K of $C^s(\Omega)$, $\Omega = [0,1]^d$, for any $s > 0$. However, in the latter presentation there is a loss of logarithm in that the proved approximation rate is

$$E(K, \Sigma_n)_{C(\Omega)} \le \left(\frac{\log n}{n}\right)^{2s/d}, \quad n \ge 1.$$

Next, let us remark that the results of Section 5.9 and Theorem 3.12 give that for the model classes $K = U(C^s(\Omega))$ we have the lower bound

$$E(K, \Sigma_n)_{C(\Omega)} \ge c_0 n^{-2s/d}, \quad n \ge 1. \tag{8.49}$$

So, at least for the Lipschitz spaces, we have matching upper and lower bounds, and therefore a satisfactory understanding of the approximation properties of deep NNs for these classes.

8.8. Super-convergence for approximation in L_p

The above results were limited to approximation in $C(\Omega)$, $\Omega = [0,1]^d$ and the Sobolev spaces $W^s(L_\infty(\Omega))$. What happens when the approximation takes place in $L_p(\Omega)$, $1 \le p < \infty$, and what happens for general Besov spaces that compactly embed in L_p? We show in this section that we can obtain super-convergence results in this case as well by using results from the theory of interpolation spaces.

Theorem 8.12. We consider approximation in $L_p(\Omega)$, $1 \le p \le \infty$, with domain $\Omega = [0,1]^d$. Let $\Sigma := (\Sigma_n)_{n\ge 1}$, where $\Sigma_n := \Upsilon^{W_0, Cn}(\mathrm{ReLU}; d, 1)$, $n \ge 1$, W_0 sufficiently large depending only on d, $C = C(s, d, \tau, p)$. If $K := U(B^s_q(L_\tau(\Omega)))$ is the unit ball of a Besov space above the Sobolev embedding line, then

$$E_n(K, \Sigma)_{L_p(\Omega)} \le C[\log n]^\beta n^{-\theta s/d}, \quad n \ge 1, \tag{8.50}$$

for any $1 \le \theta < 2 - \tau^*/\tau$, with $\tau^* := (s/d + 1/p)^{-1}$, $\tau > \tau^*$, and β depending only on s, d and θ.

Proof. This is proved by using the K-functionals of interpolation theory. To keep the presentation simple and to just sketch how this is done, we limit ourselves to proving one result of the above form when $d = 1$ and $s = 1$ with the approximation taking place in L_∞. Instead of Besov balls, we use the unit balls

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

$K_\tau := U(W^1(L_\tau(\Omega)))$, $1 \le \tau \le \infty$ of the Sobolev spaces. After presenting this example, we give in Remark 8.4 an outline of the proof of the general result stated in the theorem.

We know the two estimates

$$E_n(K_1, \Sigma)_{L_\infty(\Omega)} \le Cn^{-1}, \quad E_n(K_\infty, \Sigma)_{L_\infty(\Omega)} \le Cn^{-2}, \quad n \ge 1, \tag{8.51}$$

of which the first was given in Section 7.1.1 (in this case the result holds on the Sobolev embedding line and the approximant in this theorem can be viewed as element from $\Upsilon^{3,n}(\mathrm{ReLU}; 1, 1)$) and the second is the super-convergence result of Yarotsky (see Theorem 8.11) with $\Sigma_n := \Upsilon^{11,16n+2}(\mathrm{ReLU}; 1, 1)$. From interpolation between the pair $W^1(L_1(\Omega))$ and $W^1(L_\infty(\Omega))$ (this is where K-functionals are used; see DeVore and Scherer 1979), we know that whenever $f \in K_\tau$, for any $t > 0$ there is a function $g \in W^1(L_\infty(\Omega))$ such that

$$\|f - g\|_{W^1(L_1(\Omega))} + t\|g\|_{W^1(L_\infty(\Omega))} \le Mt^{1-1/\tau}, \tag{8.52}$$

with M an absolute constant. We take $t = 1/n$ in going further. Now let S approximate $(f - g)$ in $L_\infty(\Omega)$ with the accuracy of the first statement in (8.51), and let T approximate g with the accuracy of the second statement. Then $S + T \in \Upsilon^{11,17n+2}$ and

$$\begin{aligned} \|f - (S + T)\|_{L_\infty(\Omega)} &\le \|f - g - S\|_{L_\infty(\Omega)} + \|g - T\|_{L_\infty(\Omega)} \\ &\le C\{n^{-1}\|f - g\|_{W^1(L_1(\Omega))} + n^{-2}\|g\|_{W^1(L_\infty(\Omega))}\} \\ &\le Cn^{-1}n^{-1+1/\tau} \\ &= Cn^{-2+1/\tau}. \end{aligned} \tag{8.53}$$

In this case $\tau^* = 1$, so this is the desired inequality. Moreover, since

$$\|f - (S + T)\|_{L_p(\Omega)} \le \|f - (S + T)\|_{L_\infty(\Omega)},$$

we also have

$$E_n(K_\tau, \Sigma)_{L_p(\Omega)} \le Cn^{-2+1/\tau}, \quad 1 \le p \le \infty. \qquad \square$$

Remark 8.4. We outline the changes necessary to prove the general case in the statement of the theorem. Now we want to measure approximation error in $L_p(\Omega)$, $1 \le p < \infty$, not just $C(\Omega)$. Of course, the error of approximation in $L_p(\Omega)$ of a function f is smaller than that in $C(\Omega)$. We use analogues of (8.51) for approximation in $L_p(\Omega)$ and two Besov balls. The first is $K_0 = U(Z_0)$, $Z_0 = B^s_\infty(L_{\tau_0}(\Omega))$, where we use Theorem 8.10 to get the approximation rate $[\log_2 n]^{\beta_0} n^{-s/d}$. Here we can choose $\tau_0 > \tau^*$ so that we are as close to the Sobolev embedding line as we want (but not on it). The second inequality is the super-convergence result for $K_1 = U(Z_1)$, $Z_1 = C^s(\Omega)$. To that end, we use the generalization of Theorem 8.11, as given in Lu *et al.* (2020), which gives the super approximation rate $[\log_2 n]^{\beta_1} n^{-2s/d}$. We now interpolate between Z_0 and Z_1 to obtain the theorem for approximation in the fixed $L_p(\Omega)$ space. The reason we

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

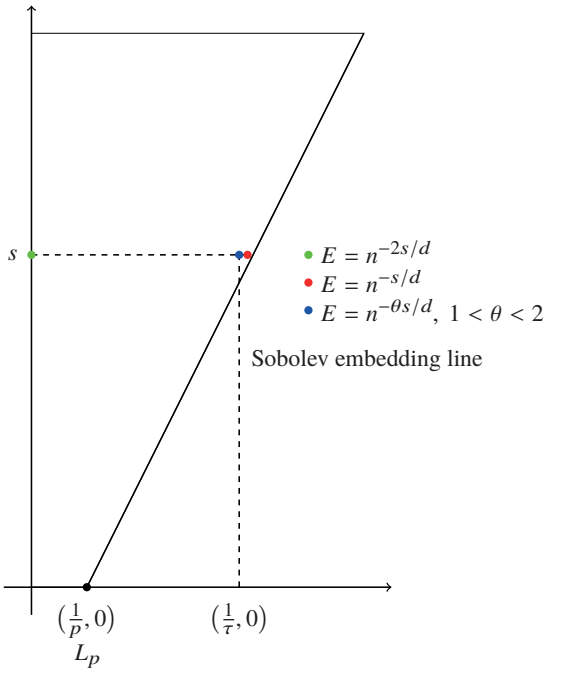

Figure 8.1. Why we get general super-convergence by using interpolation theory. All error rates E are modulo powers of logarithms when $s > 1$.

have the given restriction on θ is because we cannot take Z_0 directly on the Sobolev embedding line. Figure 8.1 may be useful in helping the reader to understand this theorem.

8.9. A summary of known approximation rates for classical smoothness spaces

Let us summarize what we know about the optimal approximation rates when approximating functions from Besov (and Sobolev) model classes using the outputs Σ_n of deep neural networks, $\Sigma_n := \Upsilon^{W_0, Cn}(\mathrm{ReLU}; d, 1)$, where W_0 is fixed, large enough, and depending only on d, and C depends on d and the model class. Given a value of p with $1 \le p \le \infty$, recall that

$$E_n(K, \Sigma)_{L_p(\Omega)} := \sup_{f \in K} \operatorname{dist}(f, \Sigma_n)_{L_p(\Omega)}, \quad \Omega := [0,1]^d. \tag{8.54}$$

We want to address what we know regarding the following problem.

Problem 8.13. For each model class K which is the unit ball of a Besov space $B^s_q(L_\tau(\Omega))$ which lies above the Sobolev embedding line for $L_p(\Omega)$, determine asymptotically matching upper and lower bounds for $E_n(K, \Sigma)_{L_p(\Omega)}$, $n \ge 1$.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Even for the most favourable case $p = \infty$, we only have a satisfactory answer to this question when $0 < s \leq 1$ and $\tau = \infty$, in which case the optimal rate is $n^{-2s/d}$, $n \geq 1$. The above results on super-convergence provide the upper bounds. The lower bounds follow from the derivation of lower bounds on approximation rates using the VC dimension, given in Section 5.9. Going further with the case $p = \infty$, the above results only provide a complete description of approximation rates when $s \leq 1$ because of the appearance of a logarithm in the extension of Yarotsky's results given in Lu *et al.* (2020).

When we move to the case $p < \infty$, the situation is even less clear. First, Theorem 8.12 does give a super rate. However, we have no corresponding lower bounds that come close to matching this rate because we cannot use the VC dimension theory for L_p approximation. In summary, for all Besov spaces that compactly embed into $L_p(\Omega)$, we obtain error bounds for approximation in $L_p(\Omega)$ strictly better than classical methods. What is missing vis-à-vis Problem 8.13 is what the best bounds are and how we prove lower bounds for approximation rates in $L_p(\Omega)$, $p \neq \infty$.

8.10. Novel model classes

While the performance of NN approximation on the classical smoothness spaces is an intriguing question that deserves a full and complete answer, we must stress the fact that such an answer will not provide an explanation for the success and popularity of NNs in their current domains of application, especially in deep learning. Indeed, the problems addressed via deep learning typically have the feature that the functions to be captured are very high-dimensional, *i.e.* the input dimension d is very large. Since all of the classical model classes built on smoothness have large entropy and suffer the curse of dimensionality as d gets large, they are not appropriate model classes for such learning problems. This amplifies the need to uncover new model classes that do not suffer the curse of dimensionality, that are well approximated by outputs of NNs, and are a good match for the targeted application. We must say that little is formally known in terms of rigorously defining new model classes in high dimensions, showing that they have reasonable entropy bounds, and then analysing their approximation properties by NNs. However, several ideas have emerged as to how such model classes may be defined. We mention some of those ideas here with the intention of outlining a road map of how to proceed with defining model classes in high dimensions.

8.10.1. Comments on the curse of dimensionality

First let us say a few words about the curse of dimensionality. One frequently hears the claim that a certain numerical method 'breaks the curse of dimensionality'. There are two components to such a statement. The first is that the specific numerical problem is such that it can be solved in high dimensions without suffering adversely from dimensionality. The second is that a particular numerical method has been found that actually does the job.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

In the setting of numerical methods for function approximation, the first statement has to do with the model class assumption on f, or the model class information that can be derived about f from the context of the problem. For example, when solving a PDE numerically, the model class information is usually given by a regularity theorem for the solution to the PDE. In other words, it is the model class K that determines whether or not the problem is solvable by a numerical method that avoids the curse of dimensionality.

Heuristically, it is thought that the crucial factor on whether or not a given model class K suffers from the curse of dimensionality is its Kolmogorov entropy in the metric where the error is to be measured; see Section 5.2 for the definition of this entropy and the entropy numbers $\varepsilon_n(K)_X$. There is not always a clear-cut mathematical proof that entropy is indeed the deciding factor. This lack of clarity stems from our vagueness in describing what is an allowable numerical method. This takes us back to the use of space-filling manifolds in approximation. We have already noted that such manifolds have the capacity to approximate arbitrarily well. But are they a fair method of approximation? Implementing such a manifold numerically as an approximation tool requires an inordinate amount of computation. So really, the computational time to implement the numerical method is an issue. This is well known in the numerical analysis community but does not seem to be treated sufficiently well in the learning community. The latter would involve statements about how many steps of a descent algorithm are necessary to guarantee a prescribed accuracy.

We have touched on this subject in Section 5.6, where we introduced stable methods of approximation. The introduction of stability was made precisely to quantify when a numerical method could be implemented within a reasonable computational budget. Under the imposition of stability in manifold approximation, we have shown that indeed the entropy of K governs optimal approximation rates.

Regarding the second factor, the question is whether we can put forward a concrete numerical scheme which can approximate the target function with a computational budget that does not grow inordinately with the dimensionality d. In this sense, it is not only an issue of how well we can approximate a given f using a specific tool $\Sigma := (\Sigma_n)_{n\geq 1}$, but whether we can find an approximant within a reasonable computational budget.

8.10.2. Model classes in high dimension

With these remarks in hand, our quest is to find appropriate model classes for high-dimensional functions which have reasonable entropy when d is large and yet match intended applications. In this context, it is allowable for the entropy of the model class to grow polynomially with d but not exponentially.

The search for appropriate high-dimensional model classes has carried on independently of deep learning or NN approximation, since it has always been a driving issue whenever we are dealing with high-dimensional approximation. We next mention some of the ideas that have emerged over recent decades on how

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

to define high-dimensional model classes and how these ideas intersect with NN approximation.

Model classes built on sparsity. The idea of using sparsity to describe high-dimensional model classes appeared largely in the context of signal/image processing. The simplest example is the following. Assume $\{\phi_j\}_{j\geq 1}$, with $\|\phi_j\|_X = 1$, is an unconditional basis in a Banach space X of functions of d variables. So, every $f \in X$ has a unique representation

$$f = \sum_{j=1}^{\infty} \lambda_j(f)\phi_j, \tag{8.55}$$

where λ_j's are linear functionals on X and the convergence in (8.55) is absolute. Here, the reader may assume that X is an L_p space to fix ideas. The space X defines the norm where we will measure performance (error of approximation). Given any $q \leq 1$, let K_q consist of all functions $f \in X$ such that

$$f = \sum_{j=1}^{\infty} \lambda_j(f)\phi_j, \quad \sum_{j=1}^{\infty} |\lambda_j(f)|^q \leq 1. \tag{8.56}$$

If one wishes to approximate functions from K_q, the most natural candidate is n-term approximation using the basis $(\phi_j)_{j\geq 1}$. Let Σ_n be the (nonlinear) set consisting of all functions $S = \sum_{j\in\Lambda} a_j\phi_j$, $\#(\Lambda) \leq n$. It is a simple exercise to show that

$$E(K_q, \Sigma_n)_X \leq C_q n^{-1/q+1}, \quad n \geq 1. \tag{8.57}$$

Note that the Besov model classes take a form similar to (8.56) because of their characterization by atomic decompositions using splines or wavelets; see Section 4.3.1. There are numerous generalizations of this notion of sparsity. For example, one can replace the unconditional basis with a more general set of functions, which form a frame or a dictionary.

Even though they give approximation rates that do not depend on the number of variables d, model classes built on sparsity are not necessarily immune to the curse of dimensionality because the basis or dictionary is infinite. To avoid this, we must impose other conditions on the sequence of coefficients $(\lambda_j(f))_{j\geq 1}$ that allow us to truncate the sum to a finite set of indices when seeking an n-term approximation. This is often imposed by putting mild decay assumptions on these coefficients. The other central issue is whether the model class built on sparsity matches the intended application. That is, there should be some justification that the sparsity class is a natural assumption in the application area.

We have already seen an example of using sparsity in terms of a dictionary in discussing NN approximation when we introduced the Barron class. The Barron class appears as a natural model class when using shallow neural networks as an approximation tool. The neat thing about the Barron class is that its definition was not made in terms of a dictionary but rather classical notions such as Fourier

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

transforms. Generalizations of Barron classes to deeper networks are given in E *et al.* (2019), where it was shown to be a sparsity class for a suitable dictionary of waveforms.

Model classes built on composition. Since NNs are built on the composition of functions, it is natural to try to define model classes based on such compositions. The basic idea is that the model class should consist of functions f with the representation $f = g_1 \circ g_2 \circ \cdots \circ g_m$, where g_k, $k = 1, \ldots, m$, are simple component functions. This approach is studied, for example, in Mhaskar and Poggio (2020), Shen *et al.* (2019) and Schmidt-Hieber (2020).

The key question in such an approach is what assumptions should be placed on the component functions. One expects to build the model class in a hierarchical fashion by showing that when g_1 and g_2 are well approximated then so is their composition. Let us consider for a moment the simple setting of approximating in the univariate uniform norm $\|\cdot\|_{C(\Omega)}$, $\Omega = [0, 1]$. Given g_1, g_2 and approximants $\hat{g}_1$ and $\hat{g}_2$, the simplest inequality for how well $\hat{g}_1 \circ \hat{g}_2$ approximates $g_1 \circ g_2$ is

$$\begin{aligned} \|g_1 \circ g_2 - \hat{g}_1 \circ \hat{g}_2\|_{C(\Omega)} &\le \|g_1 \circ g_2 - \hat{g}_1 \circ g_2\|_{C(\Omega)} + \|\hat{g}_1 \circ g_2 - \hat{g}_1 \circ \hat{g}_2\|_{C(\Omega)} \\ &\le \|g_1 - \hat{g}_1\|_{C(\Omega)} + \|\hat{g}_1\|_{\mathrm{Lip}\,1} \|g_2 - \hat{g}_2\|_{C(\Omega)}, \end{aligned} \tag{8.58}$$

which points to the observation that formulations of such model classes will probably involve mixed norms.

Model classes built on self-similarity. Let us continue with the last example of the composition $g_1 \circ g_2$. If g_2 is a CPwL function (as is the case for outputs of ReLU NNs), then as the input variable t traverses $[0, 1]$, the composition traces out scaled copies of g_1 or parts of it. For example, if g_2 is the sawtooth function $H^{\circ L}$ of Figure 3.1, then we trace out multiple copies of g_1. The composition is therefore a self-similar function. This self-similarity is prevalent in outputs of deep NNs and has been used to show that certain functions such as the Weierstrass nowhere differentiable function are well approximated by outputs of deep NNs. There are even classes of functions, generated by dynamical systems, which are efficiently approximated by outputs of deep NNs. So it is natural to try and build model classes using self-similarity or fractal-like structures, and then show that its members are well approximated by deep NNs. Examples of such univariate function classes are given in Daubechies *et al.* (2019), including the so-called Tagaki class. In higher dimensions, Dym, Sober and Daubechies (2020) showed that the characteristic functions χ_S of certain fractal sets are also efficiently approximated by the outputs of deep networks. This may relate to the success of deep learning in classification problems.

Model classes built on dimension reduction. A common high-dimensional model class with reasonable entropy is the set of functions with anisotropic smoothness. These functions depend non-democratically on their variables, that is, certain variables are more important than others; see *e.g.* DeVore, Petrova and Wojtaszczyk

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

(2011). This is a dominant theme in numerical methods for PDEs, where notions of hyperbolic smoothness classes and numerical methods built on sparse grids or tensor structures arise.

Another prominent example is a model class viewed as low-dimensional manifolds in a high-dimensional ambient space. Since our approximation tool is itself a parametrized manifold, these model classes seem like a good fit for NN approximation. This is related to the viewpoint that the NN output is an adaptive partition/filter design as expressed in Balestriero and Baraniuk (2021).

9. Stable approximation

Up to this point, we have mainly been interested in how well we can approximate a target function f by the elements of the sets $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$. The results we have obtained do not usually provide an actual procedure that could be implemented numerically. In this section we discuss in more detail issues surrounding the construction of numerical approximation procedures and whether we can guarantee their stability. This section builds on the general discussion in Section 5 which the reader needs to keep in view.

Here, we measure error in the norm of $X = L_p(\Omega)$, $\Omega = [0,1]^d$, $1 \le p \le \infty$. We let Σ_n, $n \ge 1$, be the ReLU sets $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ with the number of parameters $n(W, L) \asymp n$. As usual, the two main examples that we have in mind are when $W = n$ and $L = 1$, and secondly when $W = W_0$ is fixed (depending on d) and $L = n$. Let K be a model class in the chosen $L_p(\Omega)$.

We have mentioned before that any approximation method is described by two mappings

$$a_n \colon K \to \mathbb{R}^n, \quad M_n \colon \mathbb{R}^n \to \Sigma_n,$$

where a_n chooses the parameters of the network for a given $f \in K$, and M_n describes how the neural network takes a vector $y \in \mathbb{R}^n$ of parameters and assigns the output $M_n(y) \in \Sigma_n$. Thus the approximation to f is the function $A_n(f) = M_n(a_n(f))$. Note that once we have decided to use NN with a specific architecture for the method of approximation, the mapping M_n is fixed and we do not get to choose it.

We now wish to understand two main issues.

Stability issue 1. How does imposing stability restrictions on the mappings a_n and M_n affect the approximation rates we can obtain?

Stability issue 2. How can we construct stable numerical algorithms for approximation?

We have already discussed the first in some detail; see Section 5.5. We have seen that imposing stability limits the achievable approximation rates for NN approximation of a model class K in the sense that the decay rate cannot be better than the entropy numbers $\varepsilon_n(K)_X$ of K. Of course, this does not say we can necessarily achieve (with NNs) an approximation rate equivalent to $\varepsilon_n(K)_X$. However, this

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

does give a benchmark for optimal performance. This leads us to the following problem.

Problem 9.1. What are the optimal stable approximation rates for classical model classes such as Sobolev and Besov balls when using $\Sigma := (\Sigma_n)_{n\geq 1}$ with $\Sigma_n := \Upsilon^{W_0,Cn}(\mathrm{ReLU}; d, 1)$ as the approximation tool? In other words, we want matching upper and lower bounds for stable approximation of these model classes. A more modest question would be to replace stability by simply asking for continuity of these mappings.

Consider, for example, approximation in $L_p(\Omega)$ with $\Omega = [0,1]^d$ of the Besov balls $B^s_q(L_\tau(\Omega))$ that embed into $L_p(\Omega)$. The entropy of such a ball is known and gives the lower bounds $O(n^{-s/d})$ for the best approximation rate by a stable method of approximation. However, we have not provided stable mappings for NNs that achieve this approximation rate. A similar situation holds when we assume only continuity of these maps.

9.1. Stability of M_n

As we have noted, when using NN approximation, the mapping M_n is determined by the architecture of the NN. In this section we discuss the stability of this mapping. We always take M_n, $n = n(W, L)$ to be the natural mapping which identifies the output $S \in \Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ with the parameters that are the entries of the matrices and bias vectors of the NN; see Section 2.1. We identify these parameters with a point in $\mathbb{R}^n$ in such a way that the parameters at layer ℓ appear before those for the next layer and the ordering for each hidden layer is done in the same way.

It is easy to see that the mapping $M_n : \mathbb{R}^n \to C(\Omega)$, $\Omega = [0,1]^d$, is continuous. In fact, as we shall now show, it is a Lipschitz map on any bounded set of parameters, *i.e.* M_n is locally Lipschitz. To describe this, we need to specify a norm to be used for $\mathbb{R}^n$. We take this norm to be the $\ell_\infty(\mathbb{R}^n)$ norm, *i.e.* $\|y\|_{\ell^n_\infty} := \max_{1\leq i\leq n} |y_i|$. This choice is not optimal for obtaining the best constants in estimates but it will simplify the exposition that follows.

Theorem 9.2. If B is any finite ball in $\ell_\infty(\mathbb{R}^n)$, then $M_n : B \to C(\Omega)$ is a Lipschitz mapping, that is,

$$\|M_n(y) - M_n(y')\|_{C(\Omega)} \leq C\|y - y'\|_{\ell^n_\infty}, \quad y, y' \in B, \tag{9.1}$$

with the constant C depending only on B, W, L and d.

Sketch of proof. We will be a bit brutal and not search for the best constant in (9.1). In what follows in this proof, C denotes a constant depending only on B, W, L and d, and may change from line to line. For $y, y' \in B$ we wish to bound $\|M(y) - M(y')\|_{C(\Omega)}$ by $\delta := \|y - y'\|_{\ell^n_\infty}$. For a vector-valued continuous function g, we use $\|g\|$ to denote its $C(\Omega)$ norm, which is the maximum of the $C(\Omega)$ norm of its components.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

We let $\eta^{(j)}$, $j = 1, \dots, L$, denote the vector-valued function of $x \in \mathbb{R}^d$ computed at layer j by the network with parameters y, and we let $\eta'^{(j)}$ denote the corresponding vector of functions computed with parameters y'. We can write

$$\eta^{(1)} = \mathrm{ReLU}(A_0 x + b^{(0)}), \qquad \eta'^{(1)} = \mathrm{ReLU}(A'_0 x + b'^{(0)}),$$
$$\eta^{(j+1)} = \mathrm{ReLU}(A_j \eta^{(j)} + b^{(j)}), \quad \eta'^{(j+1)} = \mathrm{ReLU}(A'_j \eta'^{(j)} + b'^{(j)}),$$

where A_j is the matrix determined by y to go from layer j to layer $j+1$, and $b^{(j)}$ is the bias vector, $j = 0, \dots, L-1$. Similarly, A'_j and $b'^{(j)}$ correspond to the parameter y'.

Since $y, y' \in B$, all entries in the $A_j, A'_j, b^{(j)}, b'^{(j)}$ are bounded. Likewise the matrix norms of A_j, A'_j as mappings from ℓ^∞ to ℓ^∞ are bounded. Also, we have

$$\|A_0 - A'_0\|_{\ell^d_\infty \to \ell^W_\infty} \le C\delta, \quad \|A_j - A'_j\|_{\ell^W_\infty \to \ell^W_\infty} \le C\delta \quad \text{for } j = 1, \dots, L-1,$$

and

$$\|b^{(j)} - b'^{(j)}\|_{\ell^W_\infty} \le \delta \quad \text{for } j = 0, \dots, L-1.$$

Using

$$\eta^{(j+1)} = \mathrm{ReLU}[A_j(\eta^{(j)} - \eta'^{(j)}) + A_j(\eta'^{(j)}) + b^{(j)}]$$

and the fact that $\mathrm{ReLU}(\cdot)$ is a Lip 1 function, we derive

$$\begin{aligned}\|\eta^{(j+1)} - \eta'^{(j+1)}\| &\le \|A_j\|\|\eta^{(j)} - \eta'^{(j)}\| + \|A_j - A'_j\|\|\eta'^{(j)}\| + \|b^{(j)} - b'^{(j)}\| \\ &\le C\{\|\eta^{(j)} - \eta'^{(j)}\| + \delta\|\eta'^{(j)}\| + \delta\}.\end{aligned}$$

We then prove by induction that $\|\eta'^{(j)}\| \le C$, $j = 0, 1, \dots, L$, and that

$$\|\eta^{(j)} - \eta'^{(j)}\| \le C\delta, \quad j = 1, 2, \dots, L.$$

The final step is that

$$\|M_n(y) - M_n(y')\|_{C(\Omega)} \le C(\|\eta^{(L)} - \eta'^{(L)}\| + \delta),$$

which gives the theorem. □

Remark 9.1. A closer look at the above estimates shows that the Lipschitz constant for M_n can be controlled if we take B as a small ball around the origin. The size of the ball is chosen so that each of the matrices A_j, A'_j have small norm. To do this, the required size of the ball gets smaller as W gets larger.

9.2. Stability of a_n

With the above analysis of M_n in hand, we see that the stability of an NN approximation method rests on the properties of the parameter selection a_n. It is of interest to understand whether the most common methods of parameter selection based on gradient descent provide any stability. We discuss this issue in Section 11.3.1. For now, we limit ourselves to recalling our discussion on how imposing stability limits

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

approximation rates and when we know methods are stable. For the discussion that follows, we limit ourselves to approximation using $\Sigma_n = \Upsilon^{W_0,n}(\mathrm{ReLU}; d, 1)$ with W_0 fixed. Many of the same issues we raise concerning stability appear in approximation using shallow networks.

Let us first observe that the parameter selection procedures that generate the super rates of convergence for Besov and Sobolev classes cannot be continuous because of (5.6). If we require that the mappings a_n are only continuous and consider approximation in $L_p(\Omega)$, $\Omega = [0,1]^d$, then we can never attain rates of approximation better than $O(n^{-s/d})$ for the unit ball of any Besov space $B^s_q(L_\tau(\Omega))$ that embeds compactly into $L_p(\Omega)$. The only cases where we know that we can actually attain this rate is when $\tau \geq p$. In these cases there are linear spaces, such as FEM spaces, contained in Σ_n that provide this rate and the approximation can be done by a linear operator. So the following problem is not solved except for very special cases.

Problem 9.3. Consider the approximation of the unit ball of a Besov space $B^s_q(L_\tau(\Omega))$ compactly embedded in $L_p(\Omega)$ using the manifold Σ_n. Give matching upper and lower bounds for the approximation rate when a_n and M_n are Lipschitz mappings. Similarly, determine upper and lower bounds when the parameter selection mapping a_n is continuous.

A question closely related to stability is whether one can approximate well under the very modest restriction that a_n is bounded. Recall that boundedness helps us with M_n as well (see the above discussion). The issue of what approximation rates are possible when one imposes boundedness on a_n was studied in detail by Bölcskei *et al.* (2019). Their motivation was different from ours in that they were interested in NN approximation from the viewpoint of encoding. However, there is an intimate connection with stability, as we have just discussed.

10. Approximation from data

Thus far we have limited ourselves to understanding the approximation power of neural networks. The approximation rates we have obtained assumed full access to the target function f. This scenario does not match the typical application of NN approximation to the tasks of learning. In problems of learning, the only information we have is data observations of f. Such data observations alone do not allow any rigorous quantitative guarantee of how well f can be recovered, *i.e.* how accurately the behaviour of f at new points can be predicted. What is needed for the latter is additional information about f, which we have referred to as model class information. The model class information is an assumption about f that is often not provable but based more on heuristics about the application area.

Learning from data is a vast area of research that cannot be covered in any detail in this exposition. So we limit ourselves to pointing out some aspects of this problem and how they interface with the theory of NN approximation that

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

we have discussed so far. Obviously, any performance guarantees derived in the learning setting must necessarily be worse than those for approximation, where full information about f is assumed. Thus an important issue is to quantify this loss in performance.

The most common setting for the learning problem is a stochastic one, where it is assumed that the data are given by random draws from an underlying probability distribution. However, it is useful to consider the deterministic setting as well since it sheds some light on the stochastic formulation and the type of results that we can expect.

10.1. Deterministic learning; optimal recovery

In this section we wish to learn a function f which is an element of a Banach space X. Our goal is to recover f from some finite set of data observations. We assume that the data observations are in the form of bounded linearly independent linear functionals applied to f. Thus our data take the form

$$(\lambda_1(f), \ldots, \lambda_m(f)) \in \mathbb{R}^m, \quad \lambda_j \in X^*, \quad j = 1, \ldots, m, \tag{10.1}$$

where X^* is the dual space of X. As we have pointed out numerous times, to give quantitative results on how well f can be recovered requires more information about f, which we call model class information, *i.e.* information of the form $f \in K$, where K is a compact set in X. When we inject the model class assumption that $f \in K$, we have the question of how accurately we can recover f from the two pieces of information: the data and the model class. We shall present the functional analytic view of this problem which is known as *optimal recovery*. It will turn out that the optimal recovery problem is not always amenable to a simple numerical method for the recovery of f. Nevertheless, this viewpoint will be useful in motivating specific numerical methods and analysing how well they do when compared with the optimal solution.

10.2. Optimal recovery in a Hilbert space

We shall restrict our development here to the most popular setting where $X = H$ is a Hilbert space. The reader interested in the more general Banach space setting can consult DeVore *et al.* (2013). In the Hilbert space setting, each of the functionals λ_j has a representation

$$\lambda_j(f) = \langle f, \omega_j \rangle, \quad \omega_j \in H, \ j = 1, \ldots, m,$$

which is referred to as the Riesz representation of λ_j. The functions ω_j span an m-dimensional subspace

$$W := \operatorname{span}\{\omega_j\}_{j=1}^m$$

of H. We can assume without loss of generality that the ω_j's are an orthonormal system. From the given data, we can find the projection

$$w := P_W f$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

of f onto W. We think of w as the given data.

Now let us assume in addition that f is in a certain model class K, and ask what is the best approximation (with error measured in the norm of H) that we can give to f based on this information, *i.e.* the data and the model class information. One may think that the best we can do is to take $P_W f$ as the approximation. However, this is not the case since the information that $f \in K$ allows us to say something about the projection of f onto the orthogonal complement $W^\perp$ of W.

Indeed, the model class information will allow us to give a best approximation to f from the available information (model class and data w) as follows. Let

$$K_w := \{g \in K : P_W g = w\}.$$

Then the membership of f in K_w is the totality of information we have about f. The best approximation to f is now given by the centre of the set K_w. Namely, let $B := B(K_w)$ be the smallest ball in H which contains K_w. This ball is referred to as the *Chebyshev ball*, its centre $b_w \in H$ is called the *Chebyshev centre*, and its radius R_w is the *Chebyshev radius*. The best approximation we can give to f is to take b_w as the approximation and the error that will ensue is R_w. The function b_w is the *optimal recovery* and R_w is its *error of optimal recovery*.

Let us reflect a bit on the above optimal solution. Every function in K_w is a possibility for f. From the information presented to us (model class plus data), we do not know which of these functions is the desired f. So we do the best we can to approximate all of the possible f's, which turns out to be the Chebyshev centre. Each $g \in K_w$ (the possibilities for approximants of f) is of the form $w + \eta$, where η is in the null space $\mathcal{N} = W^\perp$. So, in essence, we are trying to find the $\eta \in W^\perp$ that we can add to w so that the sum $w + \eta \in K$.

Remark 10.1. Note that if we find any $\eta \in \mathcal{N}$ such that $w + \eta$ is in K, then we have essentially solved the problem, since $\hat{f} := w + \eta$ approximates f to accuracy at worst $2R_w$. Such an $\hat{f}$ is called a near-best solution.

The above description of optimal recovery, despite being elegant and optimal, is not very useful in constructing a numerical procedure since the Chebyshev ball is difficult to find numerically. Also, in practice, we are often not sure what is the appropriate model class K in a given setting. However, optimal recovery is still a good guide for the development of numerical procedures.

There are two standard approaches to developing numerical algorithms for optimal recovery. The first one is to numerically generate a recovery through least-squares minimization with a constraint that enforces the model class assumption. We will not engage this approach here but simply mention that several elegant results show that for certain model classes these optimization problems have an exact solution in NN spaces, especially those with a single hidden layer. We refer the reader to Unser (2020), Parhi and Nowak (2020), Ongie, Willett, Soudry and Srebro (2020) and Savarese, Evron, Soudry and Srebro (2019) for the most recent results using this approach.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

The second approach, which is more closely tied to approximation, is to replace K with a simpler set $\hat{K}$ which is less complex than K, and yet accurate. One then solves the optimal recovery problem on the simpler surrogate model class $\hat{K}$. We discuss this approach in the following two sections.

10.3. Optimal recovery by linear space surrogates

The usual approach to finding a surrogate $\hat{K}$ for K is to approximate K by a linear space of dimension n, or more generally, a nonlinear manifold Σ_n, with n the number of parameters needed for its description. If we know that Σ_n approximates K to accuracy ε_n (here is where our error estimates for approximation are useful), we can then replace K with

$$\hat{K} := \{h \in H\colon\ \operatorname{dist}(h, \Sigma_n)_H \le \varepsilon_n\}. \tag{10.2}$$

Clearly $K \subset \hat{K}$. Usually we also have some knowledge on the norm $\|f\|_H$ for functions $f \in K$ and this can be used to trim the set $\hat{K}$ even further.

Once a surrogate $\hat{K}$ has been chosen, we solve the optimal recovery problem for $\hat{K}$ in place of K by using Chebyshev balls for $\hat{K}_w$ as described above. As we shall now see, we can often solve the optimal recovery problem for the surrogate exactly by a numerical procedure.

We assume for the time being that $\hat{K}$ is given by (10.2) with Σ_n a linear space of dimension $n \le m$. In this case the problem is a much simpler recovery problem than the one for K, and optimal recovery has an exact solution that we now describe; see Binev *et al.* (2017). Let us define $H_w := \{h \in H\colon P_W h = w\}$, that is, H_w is the set of all functions in H which satisfy the data. Since $f \in K$, $K \subset \hat{K}$, and $P_W f = w$, we see that $\hat{K}_w := H_w \cap \hat{K}$ is non-empty. The centre of the Chebyshev ball $B(\hat{K}_w)$ for $\hat{K}_w$ is the point $u^*(w) \in H_w$ which is closest to Σ_n, that is,

$$u^*(w) := \operatorname*{argmin}_{h \in H_w} \operatorname{dist}(h, \Sigma_n)_H.$$

The function $u^*(w)$ is found as follows. One solves the least-squares problem

$$v^*(w) := \operatorname*{argmin}_{v \in \Sigma_n} \|P_W v - w\|_H,$$

and then $u^*(w) = w + P_{W^\perp} v^*(w)$, where $W^\perp$ is the orthogonal complement of W in H (the null space of P_W). One can also compute the Chebyshev radius $\hat{R}_w$ of $B(\hat{K}_w)$ as

$$\hat{R}(w) = \mu(\Sigma_n, W)_H (\varepsilon_n^2 - \|u^*(w) - v^*(w)\|_H^2)^{1/2},$$

where

$$\mu(\Sigma_n, W)_H := \sup_{\eta \in W^\perp} \frac{\|\eta\|_H}{\operatorname{dist}(\eta, \Sigma_n)_H}.$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

Here are a few remarks to put the above results into context.

Remark 10.2. The quantity $\mu(\Sigma_n, W)$ is the reciprocal of the cosine of the angle between the two spaces Σ_n and W. It reflects the quality of the data relative to Σ_n. This number will be large when the data are not well positioned relative to the linear space Σ_n. In particular, it will always be infinite whenever the dimension n of Σ_n is larger than m. This is because there will always be elements from Σ_n in the null space of P_W and hence there will be points in $\hat{K}_w$ that are arbitrarily far apart in this case.

Remark 10.3. The above results give a bound for the performance of least-squares; see Binev *et al.* (2017). Namely, given data $w_j^* = \lambda_j(f)$, $j = 1, \dots, m$, for some $f \in H$, let

$$S^* := \operatorname*{argmin}_{S \in \Sigma_n} \sum_{j=1}^{m} [w_j^* - \lambda_j(S)]^2.$$

Then, for any $f \in H$ which satisfies the data, we have

$$\|f - S^*\|_H \le \mu(\Sigma_n, W)_H \operatorname{dist}(f, \Sigma_n)_H,$$

and this bound cannot be improved in the sense that there are always $f \in H$ for which we have equality.

The above analysis and remarks only apply to the case that Σ_n is a linear space and $X = H$ is a Hilbert space. In the spirit of this paper, we would take Σ_n to be $\Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$, the outputs of a ReLU network which depends on roughly n parameters. We should choose the architecture to match K as closely as possible, given the budget n of parameters.

Let us, for example, consider the case where $\Sigma_n := \Upsilon^{W_0,n}$, $n \ge 1$, with W_0 fixed, *i.e.* the case of a deep network with constant width, and continue to assume that $X = H$ is a Hilbert space. We suppose that Σ_n provides an approximation with error

$$\operatorname{dist}(K, \Sigma_n)_H = \varepsilon_n.$$

We view $\hat{K} := \{h \in H : \operatorname{dist}(h, \Sigma_n)_H \le \varepsilon_n\}$ as a surrogate for K. Note that $K \subset \hat{K}$. If $f, g \in \hat{K}_w := \{h \in \hat{K} : P_W h = w\}$ then $\eta := f - g \in W^\perp$, and

$$\operatorname{dist}(\eta, \bar{\Sigma}_n)_H \le 2\varepsilon_n,$$

where $\bar{\Sigma}_n := \Upsilon^{W_0+d+1,2n}(\mathrm{ReLU}; d, 1)$. It follows that

$$\|f - g\|_H \le 2\mu_n \varepsilon_n, \quad \text{where } \mu_n := \sup_{\eta \in W^\perp} \frac{\|\eta\|_H}{\operatorname{dist}(\eta, \bar{\Sigma}_n)_H}, \ n \ge 1. \tag{10.3}$$

This tells us that the Chebyshev radius $\hat{R}_w$ of $\hat{K}_w$ (and thereby the Chebyshev radius R_w of K_w) satisfies

$$R_w \le \hat{R}_w \le \mu_n \varepsilon_n.$$

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

This is the same estimate as in the case when Σ_n is a linear space, except that now we have to expand Σ_n to $\bar{\Sigma}_n$ because of the nonlinearity of Σ_n.

We are left with finding an approximation to the Chebyshev centre of $\hat{K}_w$ (and thereby K_w). For this we take any $S^* \in \Sigma_n$ which satisfies

$$\|w - P_W S^*\|_H = \inf_{S \in \Sigma_n} \|w - P_W S\|_H \leq \varepsilon_n, \tag{10.4}$$

where the last inequality follows because

$$\|w - P_W S\|_H = \|P_W f - P_W S\|_H \leq \|f - S\|_H,$$

and we know $\operatorname{dist}(f, \Sigma_n)_H \leq \varepsilon_n$. This is a least-squares problem which does not necessarily have a unique solution. However, we now show that any solution S^* provides a good estimate for the Chebyshev centre of $\hat{K}_w$.

Indeed, let us take any of its solutions $S^* \in \Sigma_n$ and consider

$$h^* := w + P_{W^\perp} S^* \in H_w.$$

With an eye towards (10.4), we see that

$$\|h^* - S^*\|_H = \|w - P_W S^*\|_H \leq \varepsilon_n,$$

and thus $h^* \in \hat{K}_w$. Moreover, it follows from (10.3) that for every $f \in \hat{K}_w$ we have

$$\|f - h^*\|_H \leq 2\mu_n \varepsilon_n,$$

and therefore the ball of radius $2\mu_n\varepsilon_n$ with centre h^* contains $\hat{K}_w$. Thus h^* can be taken as an approximation to the Chebyshev centre of $\hat{K}_w$ (and thus to the Chebyshev centre of K_w). A cruder but less laborious approximation to f is provided by S^*, since

$$\|f - S^*\|_H \leq \|f - h^*\|_H + \|h^* - S^*\|_H \leq 2\mu_n\varepsilon_n + \varepsilon_n, \quad f \in K_w, \tag{10.5}$$

and therefore

$$\operatorname{dist}(K_w, S_n^*)_H \leq (2\mu_n + 1)\varepsilon_n.$$

Inequality (10.5) can be reformulated in the following way. For any $f \in H$, the least-squares solution S_n^* for $w := P_W f$ provides an approximation to f of accuracy

$$\|f - S_n^*\|_H \leq (2\mu_n + 1) \operatorname{dist}(f, \Sigma_n)_H, \tag{10.6}$$

since the above argument can be repeated with $\varepsilon_n = \operatorname{dist}(f, \Sigma_n)_H$.

Finally, note again that if $n > m$, then there will be non-trivial elements of Σ_n that interpolate zero data and hence μ_n is infinite, which renders the bound (10.6) useless. Yet this is the case of overparametrized learning, which is often used in practice. So something must be added to least-squares minimization in order to have viable results in the overparametrized case. What this additional ingredient should be is the subject of the next section.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

11. Using neural networks for data fitting

The typical setting for supervised learning is to find an approximation of an unknown function f, given a training data set of its point values

$$\{(x^{(i)}, f(x^{(i)})\}, \quad x^{(i)} \in \mathbb{R}^d, \quad f(x^{(i)}) \in \mathbb{R}, \quad i = 1, \dots, m. \tag{11.1}$$

We refer to the points $x^{(i)}$, $i = 1, \dots, m$, as the *data sites*. Thus the data observation functionals are point evaluations (delta functionals). In many applications the dimension d is very large. For example, in classification problems for images, d is the number of pixels in the images, typically somewhere in the range of 10^3 to 10^6, and for videos it is even higher. The learning problem is then to numerically produce a function $\hat{f}$ from these data that is in some sense a good predictor of f on new unseen draws $x \in \mathbb{R}^d$.

In the preceding section we described a systematic approach to learning from data, called optimal recovery. It begins with two vital requirements: (i) a known model class K to which f is assumed to belong, and (ii) a specific norm or metric in which the recovery of f by $\hat{f}$ is measured. In the optimal recovery formulation of the problem, a solid theory exists to describe the optimal solution via the Chebyshev ball. The deficiency in this approach is that the construction of numerical algorithms to generate a surrogate $\hat{f}$ may be a significant computational challenge.

Optimal recovery is not the viewpoint taken in the general literature on learning. Rather, in the learning community, the data-fitting task is formulated in a stochastic setting, where one assumes that the data come from random draws of the data sites $x^{(i)}$ with respect to a probability distribution, and the $f(x^{(i)})$'s are noisy observations of some unknown function f. Performance is then evaluated on new draws of data in the sense of probability or expectation of accuracy on these draws. This is commonly referred to as *generalization error*. Note that in this setting there is no model class assumption on the function f giving rise to the data, so there can be no provable bound for the generalization error. What is done in practice is to give an empirical bound based on checking performance on a lot of new (random) draws which are referred to as validation data.

Traditionally, model class assumptions on the unknown function f played a dominant role in the classical formulation and proof of *a priori* performance guarantees; see Bousquet *et al.* (2005). However, as noted in the previous paragraph, in the now dominant field of deep learning, where neural network approximation is an important technique, one deviates from the classical setting of model class assumptions. Our goal in the sections that follow is to understand what role approximation using neural networks plays in this new setting.

11.1. Deep learning

Deep learning is characterized by its ability to successfully treat very high-dimensional problems, beginning with inordinately large data sets and employing intens-

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

ive computation for generating surrogates. Its success in handling high-dimensional problems is provided only by empirical verification that the numerically created surrogate performs well on new draws of x. *A priori* guarantees of performance are generally lacking. In fact, performance is not typically formulated under model class assumptions, which in turn prevents such *a priori* analysis. The lack of a specific model class assumption is probably due, at least in part, to the high dimensionality d, since in this case it is often unclear what appropriate model classes should be. Note, however, that since the data observations are point evaluations, a minimal assumption is that f is in a reproducing kernel Hilbert space (RKHS), although the specific RKHS is not known or postulated.

Another important feature of deep learning is its use of overparametrization in the search for a surrogate. This runs in the face of classical learning, which warns against overfitting the data because it leads to fitting the noise.

11.2. Possible model class assumption in high dimension

Before turning to the overparametrized setting, we wish to make a few remarks on possible viable model class assumptions that could be used towards providing *a priori* guarantees in deep learning. One valid viewpoint is that the functions we are trying to recover do possess some special properties; we just do not know what they are.

The fact that neural networks are used quite successfully suggests that the functions we are trying to learn are well approximated by neural networks. If this is the case, then a natural model class assumption would be that f is in an approximation class $\mathcal{A}^r((\Sigma_n)_{n\geq 1}, X)$, which we recall consists of the functions f for which

$$\operatorname{dist}(f, \Sigma_n)_X \leq Mn^{-r}, \quad n \geq 1, \tag{11.2}$$

where again there is the question: What is the appropriate space X in which to measure error? Here $(\Sigma_n)_{n\geq 1}$ would be the family of spaces output by the chosen NNs and n would represent the number of their parameters. This underlines the importance of understanding the approximation performance of neural networks in a rate/distortion sense, and, in particular, which functions are well approximated by neural network outputs.

11.3. Overparametrization

We turn now to learning from data using overparametrized models. When searching for an approximation to f from a set of outputs of a neural network with a given architecture, say $\Sigma_n = \Upsilon^{W,L}(\sigma; d; 1)$, it is usually the case in practice that the number of trainable parameters, *i.e.* the number $n = n(W, L)$ of weights and biases, exceeds the number m of data sites $x^{(k)}$:

$$\#\text{data points} = m \ll n = \#\text{parameters}.$$

In other words, neural networks are usually *overparametrized*. This means that there are generally infinitely many choices of the parameter vector θ (of network

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

weights and biases) so that the network with these parameters outputs a function $S(\cdot;\theta)$ that fits (interpolates) the data, that is,

$$S(x^{(i)};\theta) = f(x^{(i)}), \quad i = 1,\dots,m.$$

Characterizing exactly which interpolant is chosen by the numerical method is at the heart of learning via overparametrized neural networks. In this section we want to understand how this selection is done in practice and whether the selection has an analytic interpretation. In particular, there is the question of whether the numerical method itself is in a certain sense specifying a model class assumption. If so, it would be important to unravel what this hidden assumption is.

11.3.1. Selecting the interpolant by gradient descent

The standard way of selecting an approximant $\hat{f}$ to f in the practice of overparametrized deep learning using neural networks is to begin with a random starting guess $\theta^{(0)}$ for the parameters and thereby specify the first guess $S(\cdot;\theta_0)$ for a surrogate. Successive approximations $S(\cdot,\theta^{(k)})$, for $k = 1,2,\dots$, are then generated by applying a gradient descent (or stochastic gradient descent) algorithm to find an approximate global minimum of a loss function $\mathcal{L}$, which usually takes the form of an empirical risk, such as the mean squared error

$$\mathcal{L}(\theta) := \sum_{i=1}^{m} (f(x^{(i)}) - S(x^{(i)};\theta))^2. \tag{11.3}$$

If the step sizes are appropriately chosen in the descent algorithm, then this procedure seems to work well in practice in that $S(\cdot,\theta^{(k)})$ with k large is an approximation to f which generalizes well. Here $\theta^{(k)}$ is the output parameter of the gradient descent algorithm at the kth step.

The above method for selecting a surrogate does not employ the traditional remedy for working with approximation methods that have the capacity to overfit the data, which is to add a regularizer such as an ℓ_1 or ℓ_2 penalty function on the parameter vector in the iteration. The effect of incorporating such regularizers is studied, for example, in Savarese *et al.* (2019), Ongie *et al.* (2020) and Parhi and Nowak (2020). The main conclusion of the above papers is that one can view the addition of a constraint as a model class assumption. However, it is important to note that adding such a regularizer is not usually done in NN practice. While a weak regularization is sometimes employed, empirical evidence seems to indicate that it is not necessary for good generalization performance; see Zhang *et al.* (2017).

A number of attempts have been made to understand why descent algorithms, employed to train overparametrized neural networks, provide a surrogate that generalizes well; see Jacot, Gabriel and Hongler (2018), Dziugaite and Roy (2017), Arora, Ge, Neyshabur and Zhang (2018*b*) and Bartlett, Foster and Telgarsky (2017). However, the resulting *a priori* performance guarantees are often vacuous in practice in the sense that the probability of misclassification of a new sample is bounded from above by a number greater than one. Of course, no such guarantee can hold

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

in the absence of a model class assumption on the underlying function f which provided the data.

On the other hand, some heuristic explanations have been put forward to explain the success of this approach. One of the most popular is that the descent algorithm itself provides a form of *implicit regularization* that biases learning towards selecting parameter values θ^* that correspond in some sense to low complexity functions $S(\cdot;\theta^*)$. The idea is that the starting guess $S(\cdot;\theta^{(0)})$ has relatively low complexity with high probability. Then, since the model is overparametrized, there are many values of θ for which $S(\cdot;\theta)$ interpolates the data. In particular, there is often such a value θ^* near $\theta^{(0)}$. Since gradient descent is essentially a greedy local search, it is reasonable to expect that it will converge to such a θ^* that is near $\theta^{(0)}$.

These heuristics would match a model class assumption that f itself is well approximated by the output of neural networks depending on relatively few parameters, that is, f is in a model class $\mathcal{A}^r$ with a large value of r. Or, more generally, that f is well approximated by networks depending on many parameters, but with some additional constraints on the size or complexity of these parameters. The purpose of the next section is to provide some support for this idea in the simple case of overparametrized regression.

11.3.2. Gradient descent for linear regression

As we have seen, the outputs of a neural network form a complicated nonlinear family which is difficult to analyse. It could therefore be useful to understand what the above numerical approach based on gradient descent yields in the simpler case of linear regression. We briefly describe this in the present section.

We seek to model a data set

$$\{(x^{(i)}, f(x^{(i)}))\}, \quad x^{(i)} \in \mathbb{R}^d,\ f(x^{(i)}) \in \mathbb{R},\ i = 1, \dots, m,$$

by using a function from a linear space

$$V_n = \operatorname{span}\{\phi_j,\ j = 1, \dots, n\}.$$

The key assumption we make is that the model is overparametrized, meaning that $m < n$. If $A := (a_{ij})$ is the $m \times n$ matrix with entries

$$a_{i,j} := \phi_j(x^{(i)}), \quad i = 1, \dots, m,\ j = 1, \dots, n,$$

the coefficients $\theta = (\theta_j)_{j=1}^n$ of any interpolant

$$S(\cdot, \theta) = \sum_{j=1}^{n} \theta_j \phi_j(\cdot) \in V_n \tag{11.4}$$

to the data satisfy the underdetermined system of equations

$$A\theta = y, \quad y := (f(x^{(1)}), \dots, f(x^{(m)})) \in \mathbb{R}^m. \tag{11.5}$$

The standard way of choosing a solution to (11.5) is to choose the Moore–Penrose

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

pseudoinverse $\theta^* \in \mathbb{R}^n$, which we recall is the solution which has minimum $\ell_2(\mathbb{R}^n)$ norm.

Let $W^\perp$ denote the null space, which corresponds to all $\theta \in \mathbb{R}^n$ which are solutions to (11.5) with the zero vector on the right-hand side. Further, we let W denote the orthogonal complement of $W^\perp$ in $\mathbb{R}^n$. Note that $\theta^* \in W$ since θ^* is itself a solution to (11.5).

Claim. Suppose that we apply the gradient descent algorithm, with appropriate step size restrictions, to find a minimum of the loss function (11.3), where S is the interpolant (11.4). Then this procedure determines parameter selections $\theta^{(k)} \in \mathbb{R}^n$, $k = 1, 2, \ldots$, which have a limit

$$\hat{\theta} = \lim_{k\to\infty} \theta^{(k)} = \theta^* + P_{W^\perp}\theta^{(0)}, \tag{11.6}$$

where $\theta^{(0)}$ is the initial guess.

We do not provide a full detailed proof of this claim, but make the following remarks, which will allow the reader to fill in the details. The iterative procedure chooses step sizes η_k and defines an optimization trajectory as follows,

$$\theta^{(k+1)} := \theta^{(k)} - \eta_k \nabla\mathcal{L}(\theta^{(k)}), \quad k = 0, 1, \ldots,$$

where $\nabla\mathcal{L}$ is the gradient of the loss function $\mathcal{L}$. Note that $\nabla\mathcal{L}(\theta)\cdot\theta' = 0$ for any $\theta \in \mathbb{R}^n$ and $\theta' \in W^\perp$ since the function $h(t) := \mathcal{L}(\theta + t\theta')$ is a constant function of $t \in \mathbb{R}$. Thus the vector $\nabla\mathcal{L}(\theta^{(k)})$, $k = 0, 1, \ldots$, does not have components in $W^\perp$. It follows that $P_{W^\perp}(\theta^{(k+1)}) = P_{W^\perp}(\theta^{(k)})$, $k = 0, 1, \ldots$, which gives

$$\theta^{(k)} = P_W\theta^{(k)} + P_{W^\perp}(\theta^{(0)}), \quad k = 0, 1, \ldots, \tag{11.7}$$

and

$$\mathcal{L}(\theta^{(k)}) = \mathcal{L}(P_W\theta^{(k)}). \tag{11.8}$$

The function $\mathcal{L}$ is strictly convex on W with minimizer θ^*. Since the iterations of gradient descent converge under restriction on the step size provided by the eigenvalues of A^TA, we obtain the claim.

In summary, we find that optimization by gradient descent from a random initialization has at least two important effects. First, the choice of initialization determines the value of the component $P_{W^\perp}(\theta^{(0)})$ not 'seen' by the data. Its norm is precisely the distance between the θ^* and $\hat{\theta}$, which suggests that it is important to properly initialize the optimization. Second, the gradient descent was greedy, leaving $P_{W^\perp}(\theta^{(0)})$ unchanged during the optimization. This can be viewed as a form of implicit regularization, since at least it does not increase this component. In addition, it implies an implicit model class assumption that the function f underlying the data $\{(x^{(i)}, f(x^{(i)}))\}$ is of low complexity, which means that it is well approximated by V_n.

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

11.3.3. Gradient descent selection for neural networks

The above discussion does not carry over directly to overparametrized data fitting with neural networks because the set of NN outputs is not a linear space. However, a recent line of work has shown that for sufficiently *wide* networks such considerations are still approximately valid: see Jacot *et al.* (2018), Du, Zhai, Poczos and Singh (2019*b*), Allen-Zhu, Li and Song (2019), Du *et al.* (2019*a*) and Liu, Zhu and Belkin (2020). In short, as we sketch immediately below, a number of rigorous results show that, as $W \to \infty$, gradient descent on the mean squared error loss $\mathcal{L}$ using neural networks $\Upsilon^{W,L}(\sigma; d, 1)$ can be recast as overparametrized regression in a RKHS $H_{\sigma,L}$, determined by σ and L. The reproducing kernel of $H_{\sigma,L}$ is called the neural tangent kernel and is fixed throughout training in the limit when $W \to \infty$. These results hold under certain restrictions on the initialization scheme and the learning rate.

To explain this point, suppose we are given a data set as in (11.1). Let us fix L and solve the learning problem for this data set using a class of neural networks $\Upsilon^{W,L}(\sigma; d, 1)$ in which W is large. Starting from a random guess $\theta^{(0)}$, the trajectory of the gradient descent on the loss $\mathcal{L}$ (see (11.3)) for the network parameters is given by

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t \nabla_\theta \mathcal{L}(\theta^{(t)}). \tag{11.9}$$

Varying W changes the number of components of θ. It is convenient to introduce the functions

$$v_i = v_i(\theta) := S(x^{(i)}; \theta), \quad i = 1, \dots, m,$$

which record the values of S on the data set. A simple calculus exercise (Taylor's formula) shows that the trajectory of v_i induced by (11.9) is

$$v_i^{(t+1)} = v_i^{(t)} - \eta_t \sum_{j=1}^{m} K_{\theta^{(t)}}(x^{(j)}, x^{(i)})(v_j^{(t)} - y_j) + O(\eta_t^2), \tag{11.10}$$

where $v_i^{(t+1)} := S(x^{(i)}; \theta^{(t+1)})$, $y_j = f(x^{(j)})$ and K_θ is the so-called *neural tangent kernel*

$$K_\theta(x^{(j)}, x^{(i)}) := 2 \sum_{l=1}^{n} \frac{\partial S(x^{(j)}; \theta)}{\partial \theta_l} \frac{\partial S(x^{(i)}; \theta)}{\partial \theta_l}.$$

Note that $K_{\theta^{(t)}}$ depends on the current setting $\theta^{(t)}$ of trainable parameters. However, it turns out that in the limit when W, and hence n, tends to infinity, $K_{\theta^{(t)}}$ is given for all t by the average

$$K_{\sigma,L}(x, x') := \mathbb{E}[K_{\theta^{(0)}}(x, x')], \quad x, x' \in \mathbb{R}^d,$$

of $K_{\theta^{(0)}}$ over the randomness in $\theta^{(0)}$. The notation $K_{\sigma,L}$ is meant to emphasize that this limiting kernel depends on the network depth L and the activation function σ; see Jacot *et al.* (2018) and subsequent work. Thus the training dynamics are

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

summarized by

$$v_i^{(t+1)} = v_i^{(t)} - \eta_t \sum_{j=1}^{m} K_{\sigma,L}(x^{(j)}, x^{(i)})(v_j^{(t)} - y_j). \tag{11.11}$$

The term multiplied by η_t on the right-hand side is precisely the derivative with respect to v_i of

$$\|v^{(t)} - y\|^2_{K_{\sigma,L}} := \sum_{j,i=1}^{m} K_{\sigma,L}(x^{(j)}, x^{(i)})(v_j^{(t)} - y_j)(v_i^{(t)} - y_i),$$

where $v^{(t)} - y := (v_1^{(t)} - y_1, \dots, v_m^{(t)} - y_m)$ and the norm is with respect to the RKHS structure determined by $K_{\sigma,L}$.

This derivation shows that in the case of small step sizes and large widths, using gradient descent on the loss function $\mathcal{L}$ (see (11.3)) for neural networks of fixed depth is similar to using gradient descent for the least-squares regression problem in the RKHS determined by $K_{\sigma,L}$.

While the discussion above gives some view of what gradient descent minimization is doing, a satisfactory understanding of why overparametrized learning generalizes well remains elusive. This is an important but poorly understood topic with a rapidly growing literature; see Ghorbani, Mei, Misiakiewicz and Montanari (2021), Bartlett, Long, Lugosi and Tsigler (2020) and Chizat, Oyallon and Bach (2019).

11.3.4. Stability of gradient descent

A natural question when applying gradient descent to find an approximant to the underlying function is its stability as a numerical algorithm. That is, if we slightly change the input data (the data sites and the values assigned to these points), how does this affect the output of the numerical algorithm? In this section we ask some natural questions that would aid our understanding of stability.

In our earlier treatment of stability (see Section 5.5) we assumed full access to the target function f in the formulation of what stability meant and what was an optimal performance of a stable recovery when using nonlinear manifolds. Recall that the optimal recovery rate on a model class K was given by the stable widths $\delta^*_{n,\gamma}(K)_X$, and these were connected to the entropy of K.

Let us denote the data provided to us by

$$D := \{(x^{(i)}, f(x^{(i)})\}, \quad x^{(i)} \in \mathbb{R}^d, \ f(x^{(i)}) \in \mathbb{R}, \ i = 1, \dots, m.$$

So D is a collection of m points in $\mathbb{R}^{d+1}$ and D itself can be viewed as a point in $\mathbb{R}^{(d+1)m}$. We let $\Sigma_n = \Upsilon^{W,L} := \Upsilon^{W,L}(\mathrm{ReLU}; d, 1)$ be the output set of the neural network architecture that has been chosen. Here n is the total number of parameters used to describe the elements of Σ_n, *i.e.* $n = n(W, L)$. We let $a_n \colon D \mapsto \theta(D)$ denote the mapping of the data into the parameters $\theta(D)$ chosen by the numerical

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

algorithm which, for the time being, we assume is based on gradient descent. Then $A_n(D) = M_n(a_n(D)) \in \Sigma_n$ is the output of the algorithm and the learned surrogate.

Question 1. What are the regularity properties of A_n? Is it continuous or perhaps even smoother?

Of course, the answer will depend on the step size restrictions imposed during the steps of gradient descent and, in addition, on the stopping criteria for the iterations. Recall that we know from Section 9.1 that M_n is locally Lipschitz, that is, on each bounded set B in $\mathbb{R}^n$ it is Lipschitz with Lipschitz constant γ_B. This leads us to ask the next question.

Question 2. On which compact sets in $\mathbb{R}^n$ does M_n have a reasonable Lipschitz constant?

Some information about this question can be extracted from our discussion in Section 9.1, but the analysis there was quite crude. Given an answer to Question 2, we would like a_n to map into such a ball, which leads us to the next question.

Question 3. What can be said about the range of a_n as it relates to the initial parameter guess and subsequent step size restrictions?

Our next questions centre on whether $A_n(D)$ is a good surrogate. Although model classes do not appear in the construction of A_n, there is a belief that $A_n(D)$ provides a good surrogate for the target function f that gave rise to the data. If this is indeed the case, then this statement needs an analytic formulation. One such possible answer is that A_n is good for a universal collection of model classes. To try to formulate this, let us now introduce a model class K into the picture, where $K \subset X$ is a compact subset of X. We take the view that K exists but is unknown to us.

Given such a model class K, the data sets given to us are now of the form $D = D(f)$, $f \in K$, where $f(x^{(i)})$ are the observed values at the data sites $x^{(i)}$, $i = 1, \dots, m$. We can further add in variability of the data sites by introducing $\mathcal{X} := (x^{(i)})_{i=1}^m$. In this way, we can view the data provided to depend on both the selection of sites and the $f \in K$, and write $D(\mathcal{X}, f)$. One can then revisit Questions 1–3 in this setting.

We can now view A_n as a map $A_n \colon \mathcal{X} \times K \to \Sigma_n$ and treat it as a random variable. This would allow us to measure its performance in expectation or with high probability. At this point there would be no need to require the mapping a_n to be given by gradient descent but rather put gradient descent into competition with more general mappings. This would lead to various notions of optimal performance similar to those considered in information-based complexity; see *e.g.* Traub and Wozniakowski (1980). One of these is

$$E_{m,n}(K)_X := \inf_{\#(\mathcal{X})=m; A_n \in \mathcal{A}} \sup_{f \in K} \|f - A_n(\mathcal{X}, f)\|_X, \tag{11.12}$$

where the infimum is taken over a class $\mathcal{A}$ of algorithms A_n, perhaps imposing

https://doi.org/10.1017/S0962492921000052 Published online by Cambridge University Press

some stability on A_n. Another meaningful measure of optimality would involve expected performance over random draws $\mathcal{X}$.

Whatever measure of performance is chosen, one can introduce a corresponding concept of width. Now the width $\delta_{m,n}(K)_X$ for a model class K would depend on both m and n, and the properties imposed on the algorithms in $\mathcal{A}$ such as Lipschitz mappings. With such a width in hand, one can now ask for lower and upper bounds for these widths.

Acknowledgement

All three authors were supported by MURI grant N00014-20-1-2787, administered through the US Office of Naval Research. RD and GP were supported by NSF grant DMS-1817603, NSF TRIPODS grant CCF-1934904, and BH was supported by NSF grant DMS-1855684.

References

R. A. Adams and J. J. F. Fournier (2003), *Sobolev Spaces*, Elsevier.

M. Ali and A. Nouy (2020), Approximation of smoothness classes by deep ReLU networks. Available at arXiv:2007.15645v1.

Z. Allen-Zhu, Y. Li and Z. Song (2019), A convergence theory for deep learning via over-parameterization, in *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)* (K. Chaudhuri and R. Salakhutdinov, eds), Vol. 97 of Proceedings of Machine Learning Research, PMLR, pp. 242–252.

R. Arora, A. Basu, P. Mianjy and A. Mukherjee (2018*a*), Understanding deep neural networks with rectified linear units, in *6th International Conference on Learning Representations (ICLR 2018)*. Available at https://openreview.net/forum?id=B1J_rgWRW.

S. Arora, R. Ge, B. Neyshabur and Y. Zhang (2018*b*), Stronger generalization bounds for deep nets via a compression approach, in *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)* (J. Dy and A. Krause, eds), Vol. 80 of Proceedings of Machine Learning Research, PMLR, pp. 254–263.

F. Bach (2017), Breaking the curse of dimensionality with convex neural networks, *J. Mach. Learn. Res.* **18**, 1–53.

R. Balestriero and R. Baraniuk (2021), Mad Max: Affine spline insights into deep learning, *Proc. IEEE* **109**, 704–727.

A. R. Barron (1993), Universal approximation bounds for superpositions of a sigmoidal function, *IEEE Trans. Inform. Theory* **39**, 930–945.

A. R. Barron (1994), Approximation and estimation bounds for artificial neural networks, *Mach. Learn.* **14**, 115–133.

P. L. Bartlett, D. J. Foster and M. Telgarsky (2017), Spectrally-normalized margin bounds for neural networks, in *Advances in Neural Information Processing Systems 30 (NIPS 2017)* (I. Guyon *et al.*, eds), Curran Associates, pp. 6240–6249.

P. L. Bartlett, N. Harvey, C. Liaw and A. Mehrabian (2019), Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks, *J. Mach. Learn. Res.* **20**, 1–17.

P. L. Bartlett, P. M. Long, G. Lugosi and A. Tsigler (2020), Benign overfitting in linear regression, *Proc. Natl Acad. Sci.* **117**, 30063–30070.
C. Bennett and R. Sharpley (1990), *Interpolation of Operators*, Academic Press.
Y. Benyamini and J. Lindenstrauss (2000), *Geometric Nonlinear Functional Analysis 1*, Vol. 48 of Colloquium Publications, American Mathematical Society.
J. Bergh and Lofstrom (1976), *Interpolation Spaces: An Introduction*, Springer.
P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova and P. Wojtaszczyk (2011), Convergence rates for greedy algorithms in reduced basis methods, *SIAM J. Math. Anal.* **43**, 1457–1472.
P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova and P. Wojtaszczyk (2017), Data assimilation in reduced modeling, *SIAM/ASA J. Uncertain. Quantif.* **5**, 1–29.
H. Bölcskei, P. Grohs, G. Kutyniok and P. Petersen (2019), Optimal approximation with sparsely connected deep neural networks, *SIAM J. Math. Data Sci.* **1**, 8–45.
O. Bousquet, S. Boucheron and G. Lugosi (2005), Theory of classification: A survey of some recent advances, *ESAIM Probab. Statist.* **9**, 323–375.
M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam and P. Vandergheynst (2017), Geometric deep learning: Going beyond Euclidean data, *IEEE Signal Process. Mag.* **34**, 18–42.
A. Buffa, Y. Maday, A. T. Patera, C. Prud'homme and G. Turinici (2012), *A priori* convergence of the greedy algorithm for the parameterized reduced basis, *Math. Model. Numer. Anal.* **46**, 595–603.
B. Carl (1981), Entropy numbers, *s*-numbers, and eigenvalue problems, *J. Funct. Anal.* **41**, 290–306.
L. Chizat, E. Oyallon and F. Bach (2019), On lazy training in differentiable programming, in *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)* (H. Wallach *et al.*, eds), Curran Associates, pp. 2937–2947.
A. Cohen, R. DeVore, G. Petrova and P. Wojtaszczyk (2020), Optimal stable nonlinear approximation. Available at arXiv:2009.09907.
M. Csiskos, A. Kupavskii and N. Mustafa (2019), Tight lower bounds on the VC-dimension of geometric set systems, *J. Mach. Learn. Res.* **20**, 1–8.
G. Cybenko (1989), Approximation by superpositions of a sigmoidal function, *Math. Control Signals Systems* **2**, 303–314.
I. Daubechies, R. DeVore, S. Foucart, B. Hanin and G. Petrova (2019), Nonlinear approximation and (deep) ReLU networks. Available at arXiv:1905.02199 (to appear in *Constr. Approx.*).
C. de Boor (1978), *A Practical Guide to Splines*, Vol. 27 of Applied Mathematical Sciences, Springer.
R. A. DeVore (1998), Nonlinear approximation, in *Acta Numerica*, Vol. 7, Cambridge University Press, pp. 51–150.
R. A. DeVore and V. A. Popov (1988), Interpolation of Besov spaces, *Trans. Amer. Math. Soc.* **305**, 397–414.
R. A. DeVore and K. Scherer (1979), Interpolation of linear operators on Sobolev spaces, *Ann. of Math.* **189**, 583–599.
R. A. DeVore and R. C. Sharpley (1993), Besov spaces on domains in RD, *Trans. Amer. Math. Soc.* **335**, 843–864.
R. A. DeVore and V. N. Temlyakov (1996), Some remarks on greedy algorithms, *Adv. Comput. Math.* **5**, 173–187.

R. A. DeVore, R. Howard and C. Micchelli (1989), Optimal non-linear approximation, *Manuscripta Math.* **4**, 469–478.
R. A. DeVore, G. Kyriazis, D. Leviatan and V. Tikhomirov (1993), Wavelet compression and nonlinear n-widths, *Adv. Comput. Math.* **1**, 197–214.
R. A. DeVore, K. I. Oskolkov and P. P. Petrushev (1997), Approximation by feed-forward neural networks, *Ann. Numer. Math.* **4**, 261–287.
R. A. DeVore, G. Petrova and P. Wojtaszczyk (2011), Approximation of functions of few variables in high dimensions, *Constr. Approx.* **33**, 125–143.
R. A. DeVore, G. Petrova and P. Wojtaszczyk (2013), Greedy algorithms for reduced basis in Banach spaces, *Constr. Approx.* **37**, 455–466.
S. S. Du, J. Lee, H. Li, L. Wang and X. Zhai (2019*a*), Gradient descent finds global minima of deep neural networks, in *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)* (K. Chaudhuri and R. Salakhutdinov, eds), Vol. 97 of Proceedings of Machine Learning Research, PMLR, pp. 1675–1685.
S. S. Du, X. Zhai, B. Poczos and A. Singh (2019*b*), Gradient descent provably optimizes over-parameterized neural networks, in *7th International Conference on Learning Representations (ICLR 2019)*. Available at https://openreview.net/forum?id=S1eK3i09YQ.
N. Dym, B. Sober and I. Daubechies (2020), Expression of fractals through neural network functions, *IEEE J. Selected Areas Inform. Theory* **1**, 57–66.
G. K. Dziugaite and D. M. Roy (2017), Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data, in *Workshop on Principled Approaches to Deep Learning (ICML 2017)*.
W. E and Q. Wang (2018), Exponential convergence of the deep neural network approximation for analytic functions, *Sci. China Math.* **61**, 1733–1740.
W. E, C. Ma and L. Wu (2019), The Barron space and the flow-induced function spaces for neural network models. Available at arXiv:1906.08039.
D. Elbrächter, D. Perekrestenko, P. Grohs and H. Bölcskei (2019), Deep neural network approximation theory. Available at arXiv:1901.02220.
M. W. Frazier, B. Jawerth and G. Weiss (1991), *Littlewood–Paley Theory and the Study of Function Spaces*, Vol. 79 of CBMS Regional Conference Series in Mathematics, American Mathematical Society.
B. Ghorbani, S. Mei, T. Misiakiewicz and A. Montanari (2021), Linearized two-layers neural networks in high dimension, *Ann. Statist.* **49**, 1029–1054.
R. Gribonval, G. Kutyniok, M. Nielsen and F. Voigtlaender (2019), Approximation spaces of deep neural networks. Available at arXiv:1905.01208.
I. Gühring, M. Raslan and G. Kutyniok (2020), Expressivity of deep neural networks. Available at arXiv:2007.04759.
B. Hanin (2019), Universal function approximation by deep neural nets with bounded width and ReLU activations, *Mathematics* **7**, 992.
B. Hanin and D. Rolnick (2019), Deep ReLU networks have surprisingly few activation patterns, in *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)* (H. Wallach *et al.*, eds), Curran Associates, pp. 361–370.
M. Hata (1986), Fractals in mathematics, in *Patterns and Waves: Qualitative Analysis of Nonlinear Differential Equations* (T. Nishida, M. Mimura and H. Fujii, eds), Vol. 18 of Studies in Mathematics and Its Applications, Elsevier, pp. 259–278.
J. He, L. Li, J. Xu and C. Zheng (2020), ReLU deep neural networks and linear finite elements, *Comput. Math* **38**, 502–527.

D. O. Hebb (1949), *The Organization of Behavior: A Neuropsychological Theory*, Wiley, Chapman & Hall.
K. Hornik, M. Stinchcombe, H. White *et al.* (1989), Multilayer feedforward networks are universal approximators, *Neural Networks* **2**, 359–366.
A. Jacot, F. Gabriel and C. Hongler (2018), Neural tangent kernel: Convergence and generalization in neural networks, in *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)* (S. Bengio *et al.*, eds), Curran Associates, pp. 8571–8580.
J. Klusowski and Barron (2018), Approximation by combinations of ReLU and squared ReLU ridge functions with l1 and l0 controls, *IEEE Trans. Inform. Theory* **64**, 7649–7656.
A. Krizhevsky, I. Sutskever and G. E. Hinton (2012), ImageNet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems 25 (NIPS 2012)* (F. Pereira *et al.*, eds), Curran Associates, pp. 1097–1105.
Y. LeCun, Y. Bengio and G. Hinton (2015), Deep learning, *Nature* **521** (7553), 436–444.
C. Liu, L. Zhu and M. Belkin (2020), Toward a theory of optimization for over-parameterized systems of non-linear equations: The lessons of deep learning. Available at arXiv:2003.00307.
G. G. Lorenz, Y. Makovoz and M. von Golitschek (1996), *Constructive Approximation: Advanced Problems*, first edition, Springer.
J. Lu, Z. Shen, H. Yang and S. Zhang (2020), Deep network approximation for smooth functions. Available at arXiv:2001.03040.
V. Maiorov (1999), On best approximation by ridge functions, *J. Approx. Theory* **99**, 68–94.
Y. Makovoz (1996), Random approximants and neural networks, *J. Approx. Theory* **85**, 98–109.
H. N. Mhaskar and T. Poggio (2020), Function approximation by deep networks, *Commun. Pure Appl. Anal.* **19**, 4085–4095.
G. F. Montufar, R. Pascanu, K. Cho and Y. Bengio (2014), On the number of linear regions of deep neural networks, in *Advances in Neural Information Processing Systems 27 (NIPS 2014)* (Z. Ghahramani *et al.*, eds), Curran Associates, pp. 2924–2932.
G. Ongie, R. Willett, D. Soudry and N. Srebro (2020), A function space view of bounded norm infinite width ReLU nets: The multivariate case, in *8th International Conference on Learning Representations (ICLR 2020)*. Available at https://openreview.net/forum?id=H1lNPxHKDH.
J. Opschoor, P. Petersen and C. Schwab (2019*a*), Deep ReLU networks and high-order finite element methods, *SAM, ETH Zürich*.
J. Opschoor, C. Schwab and J. Zech (2019*b*), Exponential ReLU DNN expression of holomorphic maps in high dimension. SAM Research Report, ETH Zürich.
R. Parhi and R. D. Nowak (2020), Banach space representer theorems for neural networks and ridge splines. Available at arXiv:2006.05626v2.
J. Peetre (1976), *New Thoughts on Besov Spaces*, Vol. 1 of Duke University Mathematics Series, Mathematics Department, Duke University.
P. Petersen (2020), Neural network theory. Available at http://pc-petersen.eu/Neural_Network_Theory.pdf.
P. Petersen and F. Voigtlaender (2018), Optimal approximation of piecewise smooth functions using deep ReLU neural networks, *Neural Networks* **108**, 296–330.
P. Petrushev (1988), Direct and converse theorems for spline and rational approximation and Besov spaces, in *Function Spaces and Applications*, Vol. 1302 of Lecture Notes in Mathematics, Springer, pp. 363–377.

P. Petrushev (1998), Approximation by ridge functions and neural networks, *SIAM J. Math. Anal.* **30**, 155–189.
A. Pinkus (1999), Approximation theory of the MLP model in neural networks, in *Acta Numerica*, Vol. 8, Cambridge University Press, pp. 143–195.
A. Pinkus (2012), *N-widths in Approximation Theory*, Vol. 7 of A Series of Modern Surveys in Mathematics, Springer Science & Business Media.
F. Rosenblatt (1958), The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review* **65**, 386.
P. Savarese, I. Evron, D. Soudry and N. Srebro (2019), How do infinite width bounded norm networks look in function space?, in *Proceedings of the 32nd Conference on Learning Theory (COLT 2019)* (A. Beygelzimer and D. Hsu, eds), Vol. 99 of Proceedings of Machine Learning Research, PMLR, pp. 2667–2690.
J. Schmidt-Hieber (2020), Nonparametric regression using deep neural networks with ReLU activation, *Ann. Statist.* **48**, 1875–1897.
Z. Shen, H. Yang and S. Zhang (2019), Nonlinear approximation via compositions, *Neural Networks* **119**, 74–84.
J. W. Siegel and J. Xu (2020), High order approximation rates for neural networks with ReLU^k activation functions. Available at arXiv:2012.07205.
D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.* (2016), Mastering the game of Go with deep neural networks and tree search, *Nature* **529** (7587), 484–489.
D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.* (2017), Mastering the game of Go without human knowledge, *Nature* **550** (7676), 354–359.
R. P. Stanley *et al.* (2004), An introduction to hyperplane arrangements, *Geometric Combinatorics* **13**, 389–496.
E. M. Stein (1970), *Singular Integrals and Differentiability Properties of Functions*, Princeton University Press.
M. Telgarsky (2016), Representation benefits of deep feedforward networks, *JMLR: Workshop and Conference Proceedings* **49**, 1–23.
J. Traub and H. Wozniakowski (1980), *A General Theory of Optimal Algorithms*, Academic Press.
M. Unser (2020), A unifying representer theorem for inverse problems and machine learning, *Found. Comput. Math*. doi:10.1007/s10208-020-09472-x.
V. Vapnik (1989), *Statistical Learning Theory*, Wiley-Interscience.
Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.* (2016), Google's neural machine translation system: Bridging the gap between human and machine translation. Available at arXiv:1609.08144.
D. Yarotsky (2017), Error bounds for approximations with deep ReLU networks, *Neural Networks* **94**, 103–114.
D. Yarotsky (2018), Optimal approximation of continuous functions by very deep ReLU networks, in *31st Conference on Learning Theory (COLT 2018)* (S. Bubeck *et al.*, eds), Vol. 75 of Proceedings of Machine Learning Research, PMLR, pp. 639–649.
T. Zaslavsky (1975), *Facing Up To Arrangements: Face-Count Formulas for Partitions of Space by Hyper-Planes*, American Mathematical Society.
C. Zhang, S. Bengio, M. Hardt, B. Recht and O. Vinyals (2017), Understanding deep learning requires rethinking generalization, in *5th International Conference on Learning Representations (ICLR 2017)*. Available at https://openreview.net/forum?id=Sy8gdB9xx.