Neural Networks 94 (2017) 103–114

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Error bounds for approximations with deep ReLU networks

Dmitry Yarotsky [*]

Skolkovo Institute of Science and Technology, Skolkovo Innovation Center, Building 3, Moscow 143026, Russia
Institute for Information Transmission Problems, Bolshoy Karetny per. 19, Building 1, Moscow 127051, Russia

ARTICLE INFO

Article history:
Received 7 November 2016
Received in revised form 26 June 2017
Accepted 2 July 2017
Available online 13 July 2017

Keywords:
Deep ReLU networks
Approximation complexity

ABSTRACT

We study expressive power of shallow and deep neural networks with piece-wise linear activation functions. We establish new rigorous upper and lower bounds for the network complexity in the setting of approximations in Sobolev spaces. In particular, we prove that deep ReLU networks more efficiently approximate smooth functions than shallow networks. In the case of approximations of 1D Lipschitz functions we describe adaptive depth-6 network architectures more efficient than the standard shallow architecture.

1. Introduction

Recently, multiple successful applications of deep neural networks to pattern recognition problems (LeCun, Bengio, & Hinton, 2015; Schmidhuber, 2015) have revived active interest in theoretical properties of such networks, in particular their expressive power. It has been argued that deep networks may be more expressive than shallow ones of comparable size (see, e.g., Bianchini & Scarselli, 2014; Delalleau & Bengio, 2011; Montufar, Pascanu, Cho, & Bengio, 2014; Raghu, Poole, Kleinberg, Ganguli, & Sohl-Dickstein, 2016; Telgarsky, 2015). In contrast to a shallow network, a deep one can be viewed as a long sequence of non-commutative transformations, which is a natural setting for high expressiveness (cf. the well-known Solovay–Kitaev theorem on fast approximation of arbitrary quantum operations by sequences of non-commutative gates, see Dawson and Nielsen (2006); Kitaev, Shen, and Vyalyi (2002)).

There are various ways to characterize expressive power of networks. Delalleau and Bengio (2011) consider sum–product networks and prove for certain classes of polynomials that they are much more easily represented by deep networks than by shallow networks. Montufar et al. (2014) estimate the number of linear regions in the network's landscape. Bianchini and Scarselli (2014) give bounds for Betti numbers characterizing topological properties of functions represented by networks. Telgarsky (2015, 2016) provides specific examples of classification problems where deep networks are provably more efficient than shallow ones.

In the context of classification problems, a general and standard approach to characterizing expressiveness is based on the notion of the Vapnik–Chervonenkis dimension (Vapnik & Chervonenkis, 2015). There exist several bounds for VC-dimension of deep networks with piece-wise polynomial activation functions that go back to geometric techniques of Goldberg and Jerrum (1995) and earlier results of Warren (1968); see Bartlett, Maiorov, and Meir (1998); Sakurai (1999) and the book (Anthony & Bartlett, 2009). There is a related concept, the fat-shattering dimension, for real-valued approximation problems (Anthony & Bartlett, 2009; Kearns & Schapire, 1990).

A very general approach to expressiveness in the context of approximation is the method of nonlinear widths by DeVore, Howard, and Micchelli (1989) that concerns approximation of a family of functions under assumption of a continuous dependence of the model on the approximated function.

In this paper we examine the problem of shallow-vs-deep expressiveness from the perspective of approximation theory and general spaces of functions having derivatives up to certain order (Sobolev-type spaces). In this framework, the problem of expressiveness is very well studied in the case of shallow networks with a single hidden layer, where it is known, in particular, that to approximate a C^n-function on a d-dimensional set with infinitesimal error ϵ one needs a network of size about $\epsilon^{-d/n}$, assuming a smooth activation function (see, e.g., Mhaskar (1996), Pinkus (1999) for a number of related rigorous upper and lower bounds and further qualifications of this result). Much less seems to be known about deep networks in this setting, though Mhaskar, Liao, and Poggio (2016), Mhaskar and Poggio (2016) have recently introduced functional spaces constructed using deep dependency graphs and obtained expressiveness bounds for related deep networks.

We will focus our attention on networks with the ReLU activation function $\sigma(x) = \max(0, x)$, which, despite its utter simplicity, seems to be the most popular choice in practical applications (LeCun et al., 2015). We will consider L^∞-error of approximation of

[*] Correspondence to: Skolkovo Institute of Science and Technology, Skolkovo Innovation Center, Building 3, Moscow 143026, Russia.
E-mail address: d.yarotsky@skoltech.ru.

http://dx.doi.org/10.1016/j.neunet.2017.07.002
0893-6080/© 2017 Elsevier Ltd. All rights reserved.

functions belonging to the Sobolev spaces $\mathcal{W}^{n,\infty}([0,1]^d)$ (without any assumptions of hierarchical structure). We will often consider families of approximations, as the approximated function runs over the unit ball $F_{d,n}$ in $\mathcal{W}^{n,\infty}([0,1]^d)$. In such cases we will distinguish scenarios of fixed and adaptive network architectures. Our goal is to obtain lower and upper bounds on the expressiveness of deep and shallow networks in different scenarios. We measure complexity of networks in a conventional way, by counting the number of their weights and computation units (cf. Anthony and Bartlett (2009)).

The main body of the paper consists of Sections 2–4.

In Section 2 we describe our ReLU network model and show that the ReLU function is replaceable by any other continuous piece-wise linear activation function, up to constant factors in complexity asymptotics (Proposition 1).

In Section 3 we establish several upper bounds on the complexity of approximating by ReLU networks, in particular showing that deep networks are quite efficient for approximating smooth functions. Specifically:

- In Section 3.1 we show that the function $f(x) = x^2$ can be ϵ-approximated by a network of depth and complexity $O(\ln(1/\epsilon))$ (Proposition 2). This also leads to similar upper bounds on the depth and complexity that are sufficient to implement an approximate multiplication in a ReLU network (Proposition 3).
- In Section 3.2 we describe a ReLU network architecture of depth $O(\ln(1/\epsilon))$ and complexity $O(\epsilon^{-d/n}\ln(1/\epsilon))$ that is capable of approximating with error ϵ any function from $F_{d,n}$ (Theorem 1).
- In Section 3.3 we show that, even with fixed-depth networks, one can further decrease the approximation complexity if the network architecture is allowed to depend on the approximated function. Specifically, we prove that one can ϵ-approximate a given Lipschitz function on the segment $[0, 1]$ by a depth-6 ReLU network with $O(\frac{1}{\epsilon\ln(1/\epsilon)})$ connections and activation units (Theorem 2). This upper bound is of interest since it lies below the lower bound provided by the method of nonlinear widths under assumption of continuous model selection (see Section 4.1).

In Section 4 we obtain several lower bounds on the complexity of approximation by deep and shallow ReLU networks, using different approaches and assumptions.

- In Section 4.1 we recall the general lower bound provided by the method of continuous nonlinear widths. This method assumes that parameters of the approximation continuously depend on the approximated function, but does not assume anything about how the approximation depends on its parameters. In this setup, at least $\sim\epsilon^{-d/n}$ connections and weights are required for an ϵ-approximation on $F_{d,n}$ (Theorem 3). As already mentioned, for $d = n = 1$ this lower bound is above the upper bound provided by Theorem 2.
- In Section 4.2 we consider the setup where the same network architecture is used to approximate all functions in $F_{d,n}$, but the weights are not assumed to continuously depend on the function. In this case, application of existing results on VC-dimension of deep piece-wise polynomial networks yields a $\sim \epsilon^{d/(2n)}$ lower bound in general and a $\sim \epsilon^{-d/n}\ln^{-2p-1}(1/\epsilon)$ lower bound if the network depth grows as $O(\ln^p(1/\epsilon))$ (Theorem 4).
- In Section 4.3 we consider an individual approximation, without any assumptions regarding it as an element of a family as in Sections 4.1 and 4.2. We prove that for any d, n there exists a function in $\mathcal{W}^{n,\infty}([0,1]^d)$ such that its approximation complexity is not $o(\epsilon^{-d/(9n)})$ as $\epsilon \to 0$ (Theorem 5).

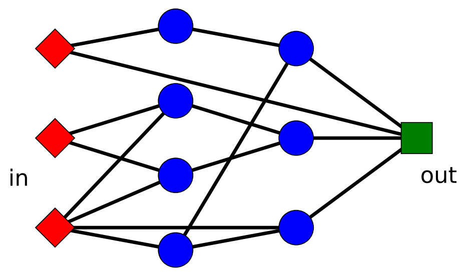

Fig. 1. A feedforward neural network having 3 input units (diamonds), 1 output unit (square), and 7 computation units with nonlinear activation (circles). The network has 4 layers and $16 + 8 = 24$ weights.

- In Section 4.4 we prove that ϵ-approximation of any nonlinear C^2-function by a network of fixed depth L requires at least $\sim \epsilon^{-1/(2(L-2))}$ computation units (Theorem 6). By comparison with Theorem 1, this shows that for sufficiently smooth functions approximation by fixed-depth ReLU networks is less efficient than by unbounded-depth networks.

In Section 5 we discuss the obtained bounds and summarize their implications, in particular comparing deep vs. shallow networks and fixed vs. adaptive architectures.

The arXiv preprint of the first version of the present work appeared almost simultaneously with the work of Liang and Srikant (2016) containing results partly overlapping with our results in Sections 3.1, 3.2 and 4.4. Liang and Srikant consider networks equipped with both ReLU and threshold activation functions. They prove a logarithmic upper bound for the complexity of approximating the function $f(x) = x^2$, which is analogous to our Proposition 2. Then, they extend this upper bound to polynomials and smooth functions. In contrast to our treatment of generic smooth functions based on standard Sobolev spaces, they impose more complex assumptions on the function (including, in particular, how many derivatives it has) that depend on the required approximation accuracy ϵ. As a consequence, they obtain strong $O(\ln^c(1/\epsilon))$ complexity bounds rather different from our bound in Theorem 1 (in fact, our lower bound proved in Theorem 5 rules out, in general, such strong upper bounds for functions having only finitely many derivatives). Also, Liang and Srikant prove a lower bound for the complexity of approximating convex functions by shallow networks. Our version of this result, given in Section 4.4, is different in that we assume smoothness and nonlinearity instead of global convexity.

2. The ReLU network model

Throughout the paper, we consider feedforward neural networks with the ReLU (Rectified Linear Unit) activation function

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

The network consists of several input units, one output unit, and a number of "hidden" computation units. Each hidden unit performs an operation of the form

$$y = \sigma\left(\sum_{k=1}^{N} w_k x_k + b\right) \tag{1}$$

with some weights (adjustable parameters) $(w_k)_{k=1}^N$ and b depending on the unit. The output unit is also a computation unit, but without the nonlinearity, i.e., it computes $y = \sum_{k=1}^N w_k x_k + b$. The units are grouped in layers, and the inputs $(x_k)_{k=1}^N$ of a computation unit in a certain layer are outputs of some units belonging to any of the preceding layers (see Fig. 1). Note that we allow connections between units in non-neighboring layers. Occasionally, when this cannot cause confusion, we may denote the network and the function it implements by the same symbol.

The depth of the network, the number of units and the total number of weights are standard measures of network complexity (Anthony & Bartlett, 2009). We will use these measures throughout the paper. The number of weights is, clearly, the sum of the total number of connections and the number of computation units. We identify the depth with the number of layers (in particular, the most common type of neural networks – shallow networks having a single hidden layer – are depth-3 networks according to this convention).

We finish this subsection with a proposition showing that, given our complexity measures, using the ReLU activation function is not much different from using any other piece-wise linear activation function with finitely many breakpoints: one can replace one network by an equivalent one but having another activation function while only increasing the number of units and weights by constant factors. This justifies our restricted attention to the ReLU networks (which could otherwise have been perceived as an excessively particular example of networks).

Proposition 1. *Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous piece-wise linear function with M breakpoints, where $1 \le M < \infty$.*

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

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

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

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

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

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

can be equivalently represented by a linear combination of a constant function and computations of $M+1$ σ-units,

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

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

(b) Let a be any breakpoint of ρ, so that $\rho'(a+) \ne \rho'(a-)$. Let r_0 be the distance separating a from the nearest other breakpoint, so that ρ is linear on $[a, a + r_0]$ and on $[a - r_0, a]$ (if ρ has only one node, any $r_0 > 0$ will do). Then, for any $r > 0$, we can express the ReLU function σ via ρ in the r-neighborhood of 0:

$$\sigma(x) = \frac{\rho\big(a + \frac{r_0}{2r}x\big) - \rho\big(a - \frac{r_0}{2} + \frac{r_0}{2r}x\big) - \rho(a) + \rho\big(a - \frac{r_0}{2}\big)}{\big(\rho'(a+) - \rho'(a-)\big)\frac{r_0}{2r}},$$
$$x \in [-r, r].$$

It follows that a computation performed by a single σ-unit,

$$x_1, \dots, x_N \mapsto \sigma\Big(\sum_{k=1}^{N} w_k x_k + b\Big),$$

can be equivalently represented by a linear combination of a constant function and two ρ-units,

$$x_1, \dots, x_N \mapsto \begin{cases} \rho\Big(a + \frac{r_0}{2r}b + \frac{r_0}{2r}\sum_{k=1}^{N} w_k x_k\Big), \\ \rho\Big(a - \frac{r_0}{2} + \frac{r_0}{2r}b + \frac{r_0}{2r}\sum_{k=1}^{N} w_k x_k\Big), \end{cases}$$

provided the condition

$$\sum_{k=1}^{N} w_k x_k + b \in [-r, r] \qquad (2)$$

holds. Since $\mathcal{D}$ is a bounded set, we can choose r at each unit of the initial network η sufficiently large so as to satisfy condition (2) for all network inputs from $\mathcal{D}$. Then, like in (a), we replace each σ-unit with two ρ-units, which produces the desired ρ-network. □

3. Upper bounds

Throughout the paper, we will be interested in approximating functions $f : [0, 1]^d \to \mathbb{R}$ by ReLU networks. Given a function $f : [0, 1]^d \to \mathbb{R}$ and its approximation $\widetilde{f}$, by the *approximation error* we will always mean the uniform maximum error

$$\|f - \widetilde{f}\|_\infty = \max_{\mathbf{x} \in [0,1]^d} |f(\mathbf{x}) - \widetilde{f}(\mathbf{x})|.$$

3.1. Fast deep approximation of squaring and multiplication

Our first key result shows that ReLU networks with unconstrained depth can very efficiently approximate the function $f(x) = x^2$ (more efficiently than any fixed-depth network, as we will see in Section 4.4). Our construction uses the "sawtooth" function that has previously appeared in the paper by Telgarsky (2015).

Proposition 2. *The function $f(x) = x^2$ on the segment $[0, 1]$ can be approximated with any error $\epsilon > 0$ by a ReLU network having the depth and the number of weights and computation units $O(\ln(1/\epsilon))$.*

Proof. Consider the "tooth" (or "mirror") function $g : [0, 1] \to [0, 1]$,

$$g(x) = \begin{cases} 2x, & x < \frac{1}{2}, \\ 2(1 - x), & x \ge \frac{1}{2}, \end{cases}$$

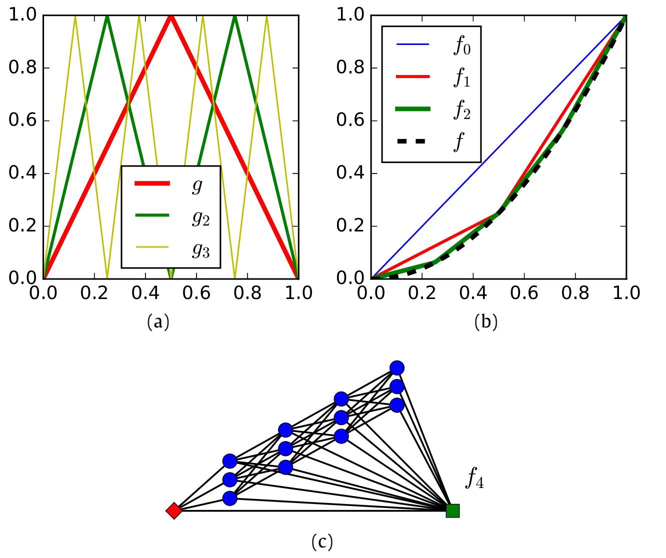

Fig. 2. Fast approximation of the function $f(x) = x^2$ from Proposition 2: (a) the "tooth" function g and the iterated "sawtooth" functions g_2, g_3; (b) the approximating functions f_m; (c) the network architecture for f_4.

and the iterated functions

$$g_s(x) = \underbrace{g \circ g \circ \cdots \circ g}_{s}(x).$$

Telgarsky has shown (see Lemma 2.4 in Telgarsky (2015)) that g_s is a "sawtooth" function with 2^{s-1} uniformly distributed "teeth" (each application of g doubles the number of teeth):

$$g_s(x) = \begin{cases} 2^s\left(x - \frac{2k}{2^s}\right), & x \in \left[\frac{2k}{2^s}, \frac{2k+1}{2^s}\right], k = 0, 1, \ldots, 2^{s-1} - 1 \\ 2^s\left(\frac{2k}{2^s} - x\right), & x \in \left[\frac{2k-1}{2^s}, \frac{2k}{2^s}\right], k = 1, 2, \ldots, 2^{s-1}, \end{cases}$$

(see Fig. 2(a)). Our key observation now is that the function $f(x) = x^2$ can be approximated by linear combinations of the functions g_s. Namely, let f_m be the piece-wise linear interpolation of f with $2^m + 1$ uniformly distributed breakpoints $\frac{k}{2^m}$, $k = 0, \ldots, 2^m$:

$$f_m\left(\frac{k}{2^m}\right) = \left(\frac{k}{2^m}\right)^2, \quad k = 0, \ldots, 2^m$$

(see Fig. 2(b)). The function f_m approximates f with the error $\epsilon_m = 2^{-2m-2}$. Now note that refining the interpolation from f_{m-1} to f_m amounts to adjusting it by a function proportional to a sawtooth function:

$$f_{m-1}(x) - f_m(x) = \frac{g_m(x)}{2^{2m}}.$$

Hence

$$f_m(x) = x - \sum_{s=1}^{m} \frac{g_s(x)}{2^{2s}}.$$

Since g can be implemented by a finite ReLU network (as $g(x) = 2\sigma(x) - 4\sigma\left(x - \frac{1}{2}\right) + 2\sigma(x - 1)$) and since construction of f_m only involves $O(m)$ linear operations and compositions of g, we can implement f_m by a ReLU network having depth and the number of weights and computation units all being $O(m)$ (see Fig. 2(c)). This implies the claim of the proposition. □

Since

$$xy = \frac{1}{2}((x+y)^2 - x^2 - y^2), \tag{3}$$

we can use Proposition 2 to efficiently implement accurate multiplication in a ReLU network. The implementation will depend on the required accuracy and the magnitude of the multiplied quantities.

Proposition 3. *Given $M > 0$ and $\epsilon \in (0, 1)$, there is a ReLU network η with two input units that implements a function $\widetilde{\times} : \mathbb{R}^2 \to \mathbb{R}$ so that*

(a) *for any inputs x, y, if $|x| \le M$ and $|y| \le M$, then $|\widetilde{\times}(x, y) - xy| \le \epsilon$;*
(b) *if $x = 0$ or $y = 0$, then $\widetilde{\times}(x, y) = 0$;*
(c) *the depth and the number of weights and computation units in η are not greater than $c_1 \ln(1/\epsilon) + c_2$ with an absolute constant c_1 and a constant $c_2 = c_2(M)$.*

Proof. Let $\widetilde{f}_{\mathrm{sq},\delta}$ be the approximate squaring function from Proposition 2 such that $\widetilde{f}_{\mathrm{sq},\delta}(0) = 0$ and $|\widetilde{f}_{\mathrm{sq},\delta}(x) - x^2| < \delta$ for $x \in [0, 1]$. Assume without loss of generality that $M \ge 1$ and set

$$\widetilde{\times}(x, y) = \frac{M^2}{8}\left(\widetilde{f}_{\mathrm{sq},\delta}\left(\frac{|x+y|}{2M}\right) - \widetilde{f}_{\mathrm{sq},\delta}\left(\frac{|x|}{2M}\right) - \widetilde{f}_{\mathrm{sq},\delta}\left(\frac{|y|}{2M}\right)\right), \tag{4}$$

where $\delta = \frac{8\epsilon}{3M^2}$. Then property (b) is immediate and (a) follows easily using expansion (3). To conclude (c), observe that computation (4) consists of three instances of $\widetilde{f}_{\mathrm{sq},\delta}$ and finitely many linear and ReLU operations, so, using Proposition 2, we can implement $\widetilde{\times}$ by a ReLU network such that its depth and the number of computation units and weights are $O(\ln(1/\delta))$, i.e. are $O(\ln(1/\epsilon) + \ln M)$. □

3.2. Fast deep approximation of general smooth functions

In order to formulate our general result, Theorem 1, we consider the Sobolev spaces $\mathcal{W}^{n,\infty}([0, 1]^d)$ with $n = 1, 2, \ldots$. Recall that $\mathcal{W}^{n,\infty}([0, 1]^d)$ is defined as the space of functions on $[0, 1]^d$ lying in L^∞ along with their weak derivatives up to order n. The norm in $\mathcal{W}^{n,\infty}([0, 1]^d)$ can be defined by

$$\|f\|_{\mathcal{W}^{n,\infty}([0,1]^d)} = \max_{\mathbf{n}:|\mathbf{n}| \le n} \operatorname*{ess\,sup}_{\mathbf{x}\in[0,1]^d} |D^{\mathbf{n}} f(\mathbf{x})|,$$

where $\mathbf{n} = (n_1, \ldots, n_d) \in \{0, 1, \ldots\}^d$, $|\mathbf{n}| = n_1 + \cdots + n_d$, and $D^{\mathbf{n}} f$ is the respective weak derivative. Here and in the sequel we denote vectors by boldface characters. The space $\mathcal{W}^{n,\infty}([0, 1]^d)$ can be equivalently described as consisting of the functions from $C^{n-1}([0, 1]^d)$ such that all their derivatives of order $n - 1$ are Lipschitz continuous.

Throughout the paper, we denote by $F_{n,d}$ the unit ball in $\mathcal{W}^{n,\infty}([0, 1]^d)$:

$$F_{n,d} = \{f \in \mathcal{W}^{n,\infty}([0, 1]^d) : \|f\|_{\mathcal{W}^{n,\infty}([0,1]^d)} \le 1\}.$$

Also, it will now be convenient to make a distinction between *networks* and *network architectures*: we define the latter as the former with unspecified weights. We say that a network architecture *is capable of expressing any function from $F_{d,n}$ with error ϵ* meaning that this can be achieved by some weight assignment.

Theorem 1. *For any d, n and $\epsilon \in (0, 1)$, there is a ReLU network architecture that*

1. *is capable of expressing any function from $F_{d,n}$ with error ϵ;*
2. *has the depth at most $c(\ln(1/\epsilon) + 1)$ and at most $c\epsilon^{-d/n}(\ln(1/\epsilon) + 1)$ weights and computation units, with some constant $c = c(d, n)$.*

Proof. The proof will consist of two steps. We start with approximating f by a sum–product combination f_1 of local Taylor polynomials and one-dimensional piecewise-linear functions. After that,

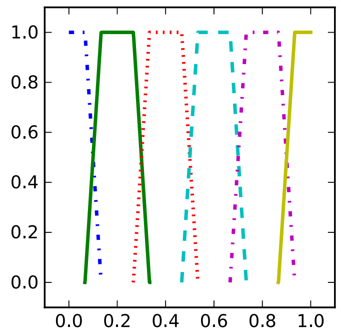

Fig. 3. Functions $(\phi_m)_{m=0}^5$ forming a partition of unity for $d = 1, N = 5$ in the proof of Theorem 1.

we will use results of the previous section to approximate f_1 by a neural network.

Let N be a positive integer. Consider a partition of unity formed by a grid of $(N+1)^d$ functions $\phi_{\mathbf{m}}$ on the domain $[0,1]^d$:

$$\sum_{\mathbf{m}} \phi_{\mathbf{m}}(\mathbf{x}) \equiv 1, \quad \mathbf{x} \in [0,1]^d.$$

Here $\mathbf{m} = (m_1, \dots, m_d) \in \{0, 1, \dots, N\}^d$, and the function $\phi_{\mathbf{m}}$ is defined as the product

$$\phi_{\mathbf{m}}(\mathbf{x}) = \prod_{k=1}^{d} \psi\left(3N\left(x_k - \frac{m_k}{N}\right)\right), \tag{5}$$

where

$$\psi(x) = \begin{cases} 1, & |x| < 1, \\ 0, & 2 < |x|, \\ 2 - |x|, & 1 \le |x| \le 2 \end{cases}$$

(see Fig. 3). Note that

$$\|\psi\|_\infty = 1 \text{ and } \|\phi_{\mathbf{m}}\|_\infty = 1 \quad \forall \mathbf{m} \tag{6}$$

and

$$\operatorname{supp} \phi_{\mathbf{m}} \subset \left\{\mathbf{x} : \left|x_k - \frac{m_k}{N}\right| < \frac{1}{N} \quad \forall k\right\}. \tag{7}$$

For any $\mathbf{m} \in \{0, \dots, N\}^d$, consider the degree-$(n-1)$ Taylor polynomial for the function f at $\mathbf{x} = \frac{\mathbf{m}}{N}$:

$$P_{\mathbf{m}}(\mathbf{x}) = \sum_{\mathbf{n}:|\mathbf{n}|<n} \frac{D^{\mathbf{n}} f}{\mathbf{n}!}\bigg|_{\mathbf{x}=\frac{\mathbf{m}}{N}} \left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{\mathbf{n}}, \tag{8}$$

with the usual conventions $\mathbf{n}! = \prod_{k=1}^d n_k!$ and $(\mathbf{x} - \frac{\mathbf{m}}{N})^{\mathbf{n}} = \prod_{k=1}^d (x_k - \frac{m_k}{N})^{n_k}$. Now define an approximation to f by

$$f_1 = \sum_{\mathbf{m} \in \{0,\dots,N\}^d} \phi_{\mathbf{m}} P_{\mathbf{m}}. \tag{9}$$

We bound the approximation error using the Taylor expansion of f:

$$\begin{aligned} |f(\mathbf{x}) - f_1(\mathbf{x})| &= \left|\sum_{\mathbf{m}} \phi_{\mathbf{m}}(\mathbf{x})(f(\mathbf{x}) - P_{\mathbf{m}}(\mathbf{x}))\right| \\ &\le \sum_{\mathbf{m}:|x_k - \frac{m_k}{N}| < \frac{1}{N} \forall k} |f(\mathbf{x}) - P_{\mathbf{m}}(\mathbf{x})| \\ &\le 2^d \max_{\mathbf{m}:|x_k - \frac{m_k}{N}| < \frac{1}{N} \forall k} |f(\mathbf{x}) - P_{\mathbf{m}}(\mathbf{x})| \\ &\le \frac{2^d d^n}{n!}\left(\frac{1}{N}\right)^n \max_{\mathbf{n}:|\mathbf{n}|=n} \operatorname*{ess\,sup}_{\mathbf{x}\in[0,1]^d} |D^{\mathbf{n}} f(\mathbf{x})| \\ &\le \frac{2^d d^n}{n!}\left(\frac{1}{N}\right)^n. \end{aligned}$$

Here in the second step we used the support property (7) and the bound (6), in the third the observation that any $\mathbf{x} \in [0,1]^d$ belongs to the support of at most 2^d functions $\phi_{\mathbf{m}}$, in the fourth a standard bound for the Taylor remainder, and in the fifth the property $\|f\|_{\mathcal{W}^{n,\infty}([0,1]^d)} \le 1$.

It follows that if we choose

$$N = \left\lceil \left(\frac{n!}{2^d d^n} \frac{\epsilon}{2}\right)^{-1/n} \right\rceil \tag{10}$$

(where $\lceil \cdot \rceil$ is the ceiling function), then

$$\|f - f_1\|_\infty \le \frac{\epsilon}{2}. \tag{11}$$

Note that, by (8) the coefficients of the polynomials $P_{\mathbf{m}}$ are uniformly bounded for all $f \in F_{d,n}$:

$$P_{\mathbf{m}}(\mathbf{x}) = \sum_{\mathbf{n}:|\mathbf{n}|<n} a_{\mathbf{m},\mathbf{n}} \left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{\mathbf{n}}, \quad |a_{\mathbf{m},\mathbf{n}}| \le 1. \tag{12}$$

We have therefore reduced our task to the following: construct a network architecture capable of approximating with uniform error $\frac{\epsilon}{2}$ any function of the form (9), assuming that N is given by (10) and the polynomials $P_{\mathbf{m}}$ are of the form (12).

Expand f_1 as

$$f_1(\mathbf{x}) = \sum_{\mathbf{m} \in \{0,\dots,N\}^d} \sum_{\mathbf{n}:|\mathbf{n}|<n} a_{\mathbf{m},\mathbf{n}} \phi_{\mathbf{m}}(\mathbf{x}) \left(\mathbf{x} - \tfrac{\mathbf{m}}{N}\right)^{\mathbf{n}}. \tag{13}$$

The expansion is a linear combination of not more than $d^n(N+1)^d$ terms $\phi_{\mathbf{m}}(\mathbf{x})(\mathbf{x} - \frac{\mathbf{m}}{N})^{\mathbf{n}}$. Each of these terms is a product of at most $d + n - 1$ piece-wise linear univariate factors: d functions $\psi(3Nx_k - 3m_k)$ (see (5)) and at most $n - 1$ linear expressions $x_k - \frac{m_k}{N}$. We can implement an approximation of this product by a neural network with the help of Proposition 3. Specifically, let $\widetilde{\times}$ be the approximate multiplication from Proposition 3 for $M = d + n$ and some accuracy δ to be chosen later, and consider the approximation of the product $\phi_{\mathbf{m}}(\mathbf{x})(\mathbf{x} - \frac{\mathbf{m}}{N})^{\mathbf{n}}$ obtained by the chained application of $\widetilde{\times}$:

$$\begin{aligned} \widetilde{f}_{\mathbf{m},\mathbf{n}}(\mathbf{x}) = {}& \widetilde{\times}\big(\psi(3Nx_1 - 3m_1), \widetilde{\times}\big(\psi(3Nx_2 - 3m_2), \dots, \\ & \widetilde{\times}\big(x_k - \tfrac{m_k}{N}, \dots\big)\dots\big)\big). \end{aligned} \tag{14}$$

Using statement (c) of Proposition 3, we see that $\widetilde{f}_{\mathbf{m},\mathbf{n}}$ can be implemented by a ReLU network with the depth and the number of weights and computation units not larger than $(d+n)c_1 \ln(1/\delta)$, for some constant $c_1 = c_1(d, n)$.

Now we estimate the error of this approximation. Note that we have $|\psi(3Nx_k - 3m_k)| \le 1$ and $|x_k - \frac{m_k}{N}| \le 1$ for all k and all $\mathbf{x} \in [0,1]^d$. By statement (a) of Proposition 3, if $|a| \le 1$ and $|b| \le M$, then $|\widetilde{\times}(a,b)| \le |b| + \delta$. Repeatedly applying this observation to all approximate multiplications in (14) while assuming $\delta < 1$, we see that the arguments of all these multiplications are bounded by our M (equal to $d + n$) and the statement (a) of Proposition 3 holds for each of them. We then have

$$\begin{aligned} &\left|\widetilde{f}_{\mathbf{m},\mathbf{n}}(\mathbf{x}) - \phi_{\mathbf{m}}(\mathbf{x})\left(\mathbf{x} - \tfrac{\mathbf{m}}{N}\right)^{\mathbf{n}}\right| \\ &\quad = \Big|\widetilde{\times}\big(\psi(3Nx_1 - 3m_1), \widetilde{\times}\big(\psi(3Nx_2 - 3m_2), \\ &\qquad \widetilde{\times}\big(\psi(3Nx_3 - 3m_3), \dots\big)\big)\big) \\ &\qquad - \psi(3Nx_1 - 3m_1)\psi(3Nx_2 - 3m_2)\psi(3Nx_3 - 3m_3)\dots\Big| \\ &\quad \le \Big|\widetilde{\times}\big(\psi(3Nx_1 - 3m_1), \widetilde{\times}\big(\psi(3Nx_2 - 3m_2), \\ &\qquad \widetilde{\times}\big(\psi(3Nx_3 - 3m_3), \dots\big)\big)\big) \\ &\qquad - \psi(3Nx_1 - 3m_1) \cdot \widetilde{\times}\big(\psi(3Nx_2 - 3m_2), \\ &\qquad \widetilde{\times}\big(\psi(3Nx_3 - 3m_3), \dots\big)\big)\Big| \\ &\qquad + |\psi(3Nx_1 - 3m_1)| \cdot \Big|\widetilde{\times}\big(\psi(3Nx_2 - 3m_2), \\ &\qquad \widetilde{\times}\big(\psi(3Nx_3 - 3m_3), \dots\big)\big) \end{aligned}$$

$$- \psi(3Nx_2 - 3m_2)\cdot \widetilde{\times}\big(\psi(3Nx_3-3m_3),\dots\big)\big| + \cdots \le (d+n)\delta. \tag{15}$$

Moreover, by statement (b) of Proposition 3,

$$\widetilde{f}_{\mathbf{m},\mathbf{n}}(\mathbf{x}) = \phi_{\mathbf{m}}(\mathbf{x})\big(\mathbf{x}-\tfrac{\mathbf{m}}{N}\big)^{\mathbf{n}}, \quad \mathbf{x}\notin \operatorname{supp}\phi_{\mathbf{m}}. \tag{16}$$

Now we define the full approximation by

$$\widetilde{f} = \sum_{\mathbf{m}\in\{0,\dots,N\}^d}\ \sum_{\mathbf{n}:|\mathbf{n}|<n} a_{\mathbf{m},\mathbf{n}}\widetilde{f}_{\mathbf{m},\mathbf{n}}. \tag{17}$$

We estimate the approximation error of $\widetilde{f}$:

$$\begin{aligned}
&|\widetilde{f}(\mathbf{x}) - f_1(\mathbf{x})|\\
&= \Big|\sum_{\mathbf{m}\in\{0,\dots,N\}^d}\ \sum_{\mathbf{n}:|\mathbf{n}|<n} a_{\mathbf{m},\mathbf{n}}\Big(\widetilde{f}_{\mathbf{m},\mathbf{n}}(\mathbf{x}) - \phi_{\mathbf{m}}(\mathbf{x})\big(\mathbf{x}-\tfrac{\mathbf{m}}{N}\big)^{\mathbf{n}}\Big)\Big|\\
&= \Big|\sum_{\mathbf{m}:\mathbf{x}\in\operatorname{supp}\phi_{\mathbf{m}}}\ \sum_{\mathbf{n}:|\mathbf{n}|<n} a_{\mathbf{m},\mathbf{n}}\Big(\widetilde{f}_{\mathbf{m},\mathbf{n}}(\mathbf{x}) - \phi_{\mathbf{m}}(\mathbf{x})\big(\mathbf{x}-\tfrac{\mathbf{m}}{N}\big)^{\mathbf{n}}\Big)\Big|\\
&\le 2^d \max_{\mathbf{m}:\mathbf{x}\in\operatorname{supp}\phi_{\mathbf{m}}} \sum_{\mathbf{n}:|\mathbf{n}|<n}\Big|\widetilde{f}_{\mathbf{m},\mathbf{n}}(\mathbf{x}) - \phi_{\mathbf{m}}(\mathbf{x})\big(\mathbf{x}-\tfrac{\mathbf{m}}{N}\big)^{\mathbf{n}}\Big|\\
&\le 2^d d^n (d+n)\delta,
\end{aligned}$$

where in the first step we use expansion (13), in the second the identity (16), in the third the bound $|a_{\mathbf{m},\mathbf{n}}| \le 1$ and the fact that $\mathbf{x} \in \operatorname{supp}\phi_{\mathbf{m}}$ for at most 2^d functions $\phi_{\mathbf{m}}$, and in the fourth the bound (15). It follows that if we choose

$$\delta = \frac{\epsilon}{2^{d+1}d^n(d+n)}, \tag{18}$$

then $\|\widetilde{f} - f_1\|_\infty \le \frac{\epsilon}{2}$ and hence, by (11),

$$\|\widetilde{f} - f\|_\infty \le \|\widetilde{f} - f_1\|_\infty + \|f_1 - f\|_\infty \le \frac{\epsilon}{2} + \frac{\epsilon}{2} \le \epsilon.$$

On the other hand, note that by (17), $\widetilde{f}$ can be implemented by a network consisting of parallel subnetworks that compute each of $\widetilde{f}_{\mathbf{m},\mathbf{n}}$; the final output is obtained by weighting the outputs of the subnetworks with the weights $a_{\mathbf{m},\mathbf{n}}$. The architecture of the full network does not depend on f; only the weights $a_{\mathbf{m},\mathbf{n}}$ do. As already shown, each of these subnetworks has not more than $c_1 \ln(1/\delta)$ layers, weights and computation units, with some constant $c_1 = c_1(d, n)$. There are not more than $d^n(N+1)^d$ such subnetworks. Therefore, the full network for $\widetilde{f}$ has not more than $c_1 \ln(1/\delta) + 1$ layers and $d^n(N+1)^d(c_1\ln(1/\delta)+1)$ weights and computation units. With δ given by (18) and N given by (10), we obtain the claimed complexity bounds. □

We remark that an analog of Theorem 1 for so-called k'th order activation functions with $k \ge 2$ can be found in Mhaskar (1993).

3.3. *Faster approximations using adaptive network architectures*

Theorem 1 provides an upper bound for the approximation complexity in the case when the same network architecture is used to approximate all functions in $F_{d,n}$. We can consider an alternative, "adaptive architecture" scenario where not only the weights, but also the architecture is adjusted to the approximated function. We expect, of course, that this would decrease the complexity of the resulting architectures, in general (at the price of needing to find the appropriate architecture). In this section we show that we can indeed obtain better upper bounds in this scenario.

For simplicity, we will only consider the case $d = n = 1$. Then, $\mathcal{W}^{n,\infty}([0,1]^d)$ is the space of Lipschitz functions on the segment $[0, 1]$. The set $F_{1,1}$ consists of functions f having both $\|f\|_\infty$ and the Lipschitz constant bounded by 1. Theorem 1 provides an upper bound $O(\frac{\ln(1/\epsilon)}{\epsilon})$ for the number of weights and computation units, but in this special case there is in fact a better bound $O(\frac{1}{\epsilon})$ obtained simply by piece-wise interpolation.

Namely, given $f \in F_{1,1}$ and $\epsilon > 0$, set $T = \lceil\frac{1}{\epsilon}\rceil$ and let $\widetilde{f}$ be the piece-wise interpolation of f with $T+1$ uniformly spaced breakpoints $(\frac{t}{T})_{t=0}^T$ (i.e., $\widetilde{f}(\frac{t}{T}) = f(\frac{t}{T}), t = 0,\dots,T$). The function $\widetilde{f}$ is also Lipschitz with constant 1 and hence $\|f - \widetilde{f}\|_\infty \le \frac{1}{T} \le \epsilon$ (since for any $x \in [0,1]$ we can find t such that $|x - \frac{t}{T}| \le \frac{1}{2T}$ and then $|f(x) - \widetilde{f}(x)| \le |f(x) - f(\frac{t}{T})| + |\widetilde{f}(\frac{t}{T}) - \widetilde{f}(x)| \le 2\cdot\frac{1}{2T} = \frac{1}{T}$). At the same time, the function $\widetilde{f}$ can be expressed in terms of the ReLU function σ by

$$\widetilde{f}(x) = b + \sum_{t=0}^{T-1} w_t\sigma\Big(x - \frac{t}{T}\Big)$$

with some coefficients b and $(w_t)_{t=0}^{T-1}$. This expression can be viewed as a special case of the depth-3 ReLU network with $O(\frac{1}{\epsilon})$ weights and computation units.

We show now how the bound $O(\frac{1}{\epsilon})$ can be improved by using adaptive architectures.

Theorem 2. *For any $f \in F_{1,1}$ and $\epsilon \in (0, \frac{1}{2})$, there exists a depth-6 ReLU network η (with architecture depending on f) that provides an ϵ-approximation of f while having not more than $\frac{c}{\epsilon\ln(1/\epsilon)}$ weights, connections and computation units. Here c is an absolute constant.*

Proof. We first explain the idea of the proof. We start with interpolating f by a piece-wise linear function, but not on the length scale ϵ—instead, we do it on a coarser length scale $m\epsilon$, with some $m = m(\epsilon) > 1$. We then create a "cache" of auxiliary subnetworks that we use to fill in the details and go down to the scale ϵ, in each of the $m\epsilon$-subintervals. This allows us to reduce the amount of computations for small ϵ because the complexity of the cache only depends on m. The assignment of cached subnetworks to the subintervals is encoded in the network architecture and depends on the function f. We optimize m by balancing the complexity of the cache with that of the initial coarse approximation. This leads to $m \sim \ln(1/\epsilon)$ and hence to the reduction of the total complexity of the network by a factor $\sim \ln(1/\epsilon)$ compared to the simple piece-wise linear approximation on the scale ϵ. This construction is inspired by a similar argument used to prove the $O(2^n/n)$ upper bound for the complexity of Boolean circuits implementing n-ary functions (Shannon, 1949).

The proof becomes simpler if, in addition to the ReLU function σ, we are allowed to use the activation function

$$\rho(x) = \begin{cases} x, & x \in [0,1), \\ 0, & x \notin [0,1) \end{cases} \tag{19}$$

in our neural network. Since ρ is discontinuous, we cannot just use Proposition 1 to replace ρ-units by σ-units. We will first prove the analog of the claimed result for the model including ρ-units, and then we will show how to construct a purely ReLU network.

Lemma 1. *For any $f \in F_{1,1}$ and $\epsilon \in (0, \frac{1}{2})$, there exists a depth-5 network including σ-units and ρ-units, that provides an ϵ-approximation of f while having not more than $\frac{c}{\epsilon\ln(1/\epsilon)}$ weights, where c is an absolute constant.*

Proof. Given $f \in F_{1,1}$, we will construct an approximation $\widetilde{f}$ to f in the form

$$\widetilde{f} = \widetilde{f}_1 + \widetilde{f}_2.$$

Here, $\widetilde{f}_1$ is the piece-wise linear interpolation of f with the breakpoints $\{\frac{t}{T}\}_{t=0}^T$, for some positive integer T to be chosen later. Since

f is Lipschitz with constant 1, $\widetilde{f}_1$ is also Lipschitz with constant 1. We will denote by I_t the intervals between the breakpoints:

$$I_t = \Big[\frac{t}{T}, \frac{t+1}{T}\Big), \quad t = 0, \dots, T-1.$$

We will now construct $\widetilde{f}_2$ as an approximation to the difference

$$f_2 = f - \widetilde{f}_1. \tag{20}$$

Note that f_2 vanishes at the endpoints of the intervals I_t:

$$f_2\Big(\frac{t}{T}\Big) = 0, \quad t = 0, \dots, T, \tag{21}$$

and f_2 is Lipschitz with constant 2:

$$|f_2(x_1) - f_2(x_2)| \le 2|x_1 - x_2|, \tag{22}$$

since f and $\widetilde{f}_1$ are Lipschitz with constant 1.

To define $\widetilde{f}_2$, we first construct a set Γ of cached functions. Let m be a positive integer to be chosen later. Let Γ be the set of piecewise linear functions $\gamma : [0, 1] \to \mathbb{R}$ with the breakpoints $\{\frac{r}{m}\}_{r=0}^m$ and the properties

$$\gamma(0) = \gamma(1) = 0$$

and

$$\gamma\Big(\frac{r}{m}\Big) - \gamma\Big(\frac{r-1}{m}\Big) \in \Big\{-\frac{2}{m}, 0, \frac{2}{m}\Big\}, \quad r = 1, \dots, m.$$

Note that the size $|\Gamma|$ of Γ is not larger than 3^m.

If $g : [0, 1] \to \mathbb{R}$ is any Lipschitz function with constant 2 and $g(0) = g(1) = 0$, then g can be approximated by some $\gamma \in \Gamma$ with error not larger than $\frac{2}{m}$: namely, take $\gamma(\frac{r}{m}) = \frac{2}{m}\lfloor g(\frac{r}{m})/\frac{2}{m}\rfloor$.

Moreover, if f_2 is defined by (20), then, using (21), (22), on each interval I_t the function f_2 can be approximated with error not larger than $\frac{2}{Tm}$ by a properly rescaled function $\gamma \in \Gamma$. Namely, for each $t = 0, \dots, T-1$ we can define the function g by $g(y) = Tf_2(\frac{t+y}{T})$. Then it is Lipschitz with constant 2 and $g(0) = g(1) = 0$, so we can find $\gamma_t \in \Gamma$ such that

$$\sup_{y\in[0,1)} \Big| Tf_2\Big(\frac{t+y}{T}\Big) - \gamma_t(y)\Big| \le \frac{2}{m}.$$

This can be equivalently written as

$$\sup_{x\in I_t} \Big| f_2(x) - \frac{1}{T}\gamma_t(Tx - t)\Big| \le \frac{2}{Tm}.$$

Note that the obtained assignment $t \mapsto \gamma_t$ is not injective, in general (T will be larger than $|\Gamma|$).

We can then define $\widetilde{f}_2$ on the whole $[0, 1)$ by

$$\widetilde{f}_2(x) = \frac{1}{T}\gamma_t(Tx - t), \quad x \in I_t, \quad t = 0, \dots, T-1. \tag{23}$$

This $\widetilde{f}_2$ approximates f_2 with error $\frac{2}{Tm}$ on $[0, 1)$:

$$\sup_{x\in[0,1)} |f_2(x) - \widetilde{f}_2(x)| \le \frac{2}{Tm}, \tag{24}$$

and hence, by (20), for the full approximation $\widetilde{f} = \widetilde{f}_1 + \widetilde{f}_2$ we will also have

$$\sup_{x\in[0,1)} |f(x) - \widetilde{f}(x)| \le \frac{2}{Tm}. \tag{25}$$

Note that the approximation $\widetilde{f}_2$ has properties analogous to (21), (22):

$$\widetilde{f}_2\Big(\frac{t}{T}\Big) = 0, \quad t = 0, \dots, T, \tag{26}$$

$$|\widetilde{f}_2(x_1) - \widetilde{f}_2(x_2)| \le 2|x_1 - x_2|, \tag{27}$$

in particular, $\widetilde{f}_2$ is continuous on $[0, 1)$.

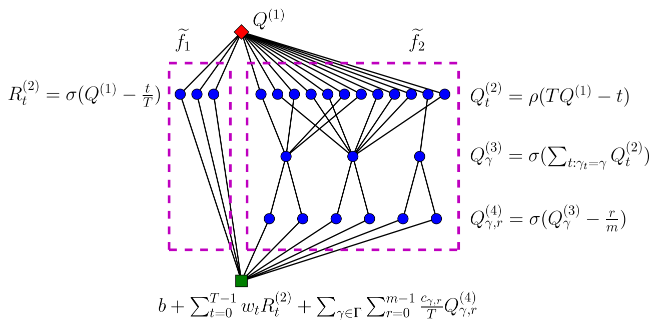
$\widetilde{f}_1$
$Q^{(1)}$
$\widetilde{f}_2$
$R_t^{(2)} = \sigma(Q^{(1)} - \frac{t}{T})$
$Q_t^{(2)} = \rho(TQ^{(1)} - t)$
$Q_\gamma^{(3)} = \sigma(\sum_{t:\gamma_t=\gamma} Q_t^{(2)})$
$Q_{\gamma,r}^{(4)} = \sigma(Q_\gamma^{(3)} - \frac{r}{m})$
$b + \sum_{t=0}^{T-1} w_t R_t^{(2)} + \sum_{\gamma\in\Gamma}\sum_{r=0}^{m-1} \frac{c_{\gamma,r}}{T} Q_{\gamma,r}^{(4)}$

Fig. 4. Architecture of the network implementing the function $\widetilde{f} = \widetilde{f}_1 + \widetilde{f}_2$ from Lemma 1.

We will now rewrite $\widetilde{f}_2$ in a different form interpretable as a computation by a neural network. Specifically, using our additional activation function ρ given by (19), we can express $\widetilde{f}_2$ as

$$\widetilde{f}_2(x) = \frac{1}{T}\sum_{\gamma\in\Gamma}\gamma\Big(\sum_{t:\gamma_t=\gamma}\rho(Tx - t)\Big). \tag{28}$$

Indeed, given $x \in [0, 1)$, observe that all the terms in the inner sum vanish except for the one corresponding to the t determined by the condition $x \in I_t$. For this particular t we have $\rho(Tx - t) = Tx - t$. Since $\gamma(0) = 0$, we conclude that (28) agrees with (23).

Let us also expand $\gamma \in \Gamma$ over the basis of shifted ReLU functions:

$$\gamma(x) = \sum_{r=0}^{m-1} c_{\gamma,r}\sigma\Big(x - \frac{r}{m}\Big), \quad x \in [0, 1].$$

Substituting this expansion in (28), we finally obtain

$$\widetilde{f}_2(x) = \frac{1}{T}\sum_{\gamma\in\Gamma}\sum_{r=0}^{m-1} c_{\gamma,r}\sigma\Big(\sum_{t:\gamma_t=\gamma}\rho(Tx - t) - \frac{r}{m}\Big). \tag{29}$$

Now consider the implementation of $\widetilde{f}$ by a neural network. The term $\widetilde{f}_1$ can clearly be implemented by a depth-3 ReLU network using $O(T)$ connections and computation units. The term $\widetilde{f}_2$ can be implemented by a depth-5 network with ρ- and σ-units as follows (we denote a computation unit by Q with a superscript indexing the layer and a subscript indexing the unit within the layer).

1. The first layer contains the single input unit $Q^{(1)}$.
2. The second layer contains T units $(Q_t^{(2)})_{t=1}^T$ computing $Q_t^{(2)} = \rho(TQ^{(1)} - t)$.
3. The third layer contains $|\Gamma|$ units $(Q_\gamma^{(3)})_{\gamma\in\Gamma}$ computing $Q_\gamma^{(3)} = \sigma(\sum_{t:\gamma_t=\gamma} Q_t^{(2)})$. This is equivalent to $Q_\gamma^{(3)} = \sum_{t:\gamma_t=\gamma} Q_t^{(2)}$, because $Q_t^{(2)} \ge 0$.
4. The fourth layer contains $m|\Gamma|$ units $(Q_{\gamma,r}^{(4)})_{\substack{\gamma\in\Gamma \\ r=0,\dots,m-1}}$ computing $Q_{\gamma,r}^{(4)} = \sigma(Q_\gamma^{(3)} - \frac{r}{m})$.
5. The final layer consists of a single output unit $Q^{(5)} = \sum_{\gamma\in\Gamma}\sum_{r=0}^{m-1}\frac{c_{\gamma,r}}{T}Q_{\gamma,r}^{(4)}$.

Examining this network, we see that the total number of connections and units in it is $O(T + m|\Gamma|)$ and hence is $O(T + m3^m)$. This also holds for the full network implementing $\widetilde{f} = \widetilde{f}_1 + \widetilde{f}_2$, since the term $\widetilde{f}_1$ requires even fewer layers, connections and units. The output units of the subnetworks for $\widetilde{f}_1$ and $\widetilde{f}_2$ can be merged into the output unit for $\widetilde{f}_1 + \widetilde{f}_2$, so the depth of the full network is the maximum of the depths of the networks implementing $\widetilde{f}_1$ and $\widetilde{f}_2$, i.e., is 5 (see Fig. 4).

Now, given $\epsilon \in (0, \frac{1}{2})$, take $m = \lceil\frac{1}{2}\log_3(1/\epsilon)\rceil$ and $T = \lceil\frac{2}{m\epsilon}\rceil$. Then, by (25), the approximation error $\max_{x\in[0,1]}|f(x) - \widetilde{f}(x)| \le$

$\frac{2}{Tm} \le \epsilon$, while $T + m3^m = O(\frac{1}{\epsilon \ln(1/\epsilon)})$, which implies the claimed complexity bound. □

We show now how to modify the constructed network so as to remove ρ-units. We only need to modify the $\widetilde{f}_2$ part of the network. We will show that for any $\delta > 0$ we can replace $\widetilde{f}_2$ with a function $\widetilde{f}_{3,\delta}$ (defined below) that

(a) obeys the following analog of approximation bound (24):

$$\sup_{x\in[0,1]} |f_2(x) - \widetilde{f}_{3,\delta}(x)| \le \frac{8\delta}{T} + \frac{2}{Tm}, \tag{30}$$

(b) and is implementable by a depth-6 ReLU network having complexity $c(T + m3^m)$ with an absolute constant c independent of δ.

Since δ can be taken arbitrarily small, the theorem then follows by arguing as in Lemma 1, only with $\widetilde{f}_2$ replaced by $\widetilde{f}_{3,\delta}$.

As a first step, we approximate ρ by a continuous piece-wise linear function ρ_δ, with a small $\delta > 0$:

$$\rho(y) = \begin{cases} y, & y \in [0, 1-\delta), \\ \frac{1-\delta}{\delta}(1-y), & y \in [1-\delta, 1), \\ 0, & y \notin [0, 1). \end{cases}$$

Let $\widetilde{f}_{2,\delta}$ be defined as $\widetilde{f}_2$ in (29), but with ρ replaced by ρ_δ:

$$\widetilde{f}_{2,\delta}(x) = \frac{1}{T}\sum_{\gamma\in\Gamma}\sum_{r=0}^{m-1} c_{\gamma,r}\sigma\Big(\sum_{t:\gamma_t=\gamma}\rho_\delta(Tx - t) - \frac{r}{m}\Big).$$

Since ρ_δ is a continuous piece-wise linear function with three breakpoints, we can express it via the ReLU function, and hence implement $\widetilde{f}_{2,\delta}$ by a purely ReLU network, as in Proposition 1, and the complexity of the implementation does not depend on δ. However, replacing ρ with ρ_δ affects the function $\widetilde{f}_2$ on the intervals $(\frac{t-\delta}{T}, \frac{t}{T}]$, $t = 1, \dots, T$, introducing there a large error (of magnitude $O(\frac{1}{T})$). But recall that both f_2 and $\widetilde{f}_2$ vanish at the points $\frac{t}{T}$, $t = 0, \dots, T$, by (21), (26). We can then largely remove this newly introduced error by simply suppressing $\widetilde{f}_{2,\delta}$ near the points $\frac{t}{T}$.

Precisely, consider the continuous piece-wise linear function

$$\phi_\delta(y) = \begin{cases} 0, & y \notin [0, 1-\delta), \\ \frac{y}{\delta}, & y \in [0, \delta), \\ 1, & y \in [\delta, 1-2\delta), \\ \frac{1-\delta-y}{\delta}, & y \in [1-2\delta, 1-\delta) \end{cases}$$

and the full comb-like filtering function

$$\Phi_\delta(x) = \sum_{t=0}^{T-1}\phi_\delta(Tx - t).$$

Note that Φ_δ is continuous piece-wise linear with $4T$ breakpoints, and $0 \le \Phi_\delta(x) \le 1$. We then define our final modification of $\widetilde{f}_2$ as

$$\widetilde{f}_{3,\delta}(x) = \sigma\Big(\widetilde{f}_{2,\delta}(x) + 2\Phi_\delta(x) - 1\Big) - \sigma\Big(2\Phi_\delta(x) - 1\Big). \tag{31}$$

Lemma 2. *The function $\widetilde{f}_{3,\delta}$ obeys the bound (30).*

Proof. Given $x \in [0, 1)$, let $t \in \{0, \dots, T-1\}$ and $y \in [0, 1)$ be determined from the representation $x = \frac{t+y}{T}$ (i.e., y is the relative position of x in the respective interval I_t). Consider several possibilities for y:

1. $y \in [1-\delta, 1]$. In this case $\Phi_\delta(x) = 0$. Note that

$$\sup_{x\in[0,1]} |\widetilde{f}_{2,\delta}(x)| \le 1, \tag{32}$$

because, by construction, $\sup_{x\in[0,1]}|\widetilde{f}_{2,\delta}(x)| \le \sup_{x\in[0,1]}|\widetilde{f}_2(x)|$, and $\sup_{x\in[0,1]}|\widetilde{f}_2(x)| \le 1$ by (26), (27). It follows that both terms in (31) vanish, i.e., $\widetilde{f}_{3,\delta}(x) = 0$. But, since f_2 is Lipschitz with constant 2 by (22) and $f_2(\frac{t+1}{T}) = 0$, we have $|f_2(x)| \le |f_2(x) - f_2(\frac{t+1}{T})| \le \frac{2|y-1|}{T} \le \frac{2\delta}{T}$. This implies $|f_2(x) - \widetilde{f}_{3,\delta}(x)| \le \frac{2\delta}{T}$.

2. $y \in [\delta, 1-2\delta]$. In this case $\Phi_\delta(x) = 1$ and $\widetilde{f}_{2,\delta}(x) = \widetilde{f}_2(x)$. Using (32), we find that $\widetilde{f}_{3,\delta}(x) = \widetilde{f}_{2,\delta}(x) = \widetilde{f}_2(x)$. It follows that $|f_2(x) - \widetilde{f}_{3,\delta}(x)| = |f_2(x) - \widetilde{f}_2(x)| \le \frac{2}{Tm}$.

3. $y \in [0, \delta] \cup [1-2\delta, 1-\delta]$. In this case $\widetilde{f}_{2,\delta}(x) = \widetilde{f}_2(x)$. Since σ is Lipschitz with constant 1, $|\widetilde{f}_{3,\delta}(x)| \le |\widetilde{f}_{2,\delta}(x)| = |\widetilde{f}_2(x)|$. Both f_2 and $\widetilde{f}_2$ are Lipschitz with constant 2 (by (22), (27)) and vanish at $\frac{t}{T}$ and $\frac{t+1}{T}$ (by (21), (26)). It follows that

$$|f_2(x) - \widetilde{f}_{3,\delta}(x)| \le |f_2(x)| + |\widetilde{f}_2(x)| \le 2\begin{cases} 2|x - \frac{t}{T}|, & y \in [0, \delta] \\ 2|x - \frac{t+1}{T}|, & y \in [1-2\delta, 1-\delta] \end{cases} \le \frac{8\delta}{T}. \quad \square$$

It remains to verify the complexity property (b) of the function $\widetilde{f}_{3,\delta}$. As already mentioned, $\widetilde{f}_{2,\delta}$ can be implemented by a depth-5 purely ReLU network with not more than $c(T + m3^m)$ weights, connections and computation units, where c is an absolute constant independent of δ. The function Φ_δ can be implemented by a shallow, depth-3 network with $O(T)$ units and connection. Then, computation of $\widetilde{f}_{3,\delta}$ can be implemented by a network including two subnetworks for computing $\widetilde{f}_{2,\delta}$ and Ψ_δ, and an additional layer containing two σ-units as written in (31). We thus obtain 6 layers in the resulting full network and, choosing T and m in the same way as in Lemma 1, obtain the bound $\frac{c}{\epsilon \ln(1/\epsilon)}$ for the number of its connections, weights, and computation units. □

4. Lower bounds

4.1. Continuous nonlinear widths

The method of continuous nonlinear widths (DeVore et al., 1989) is a very general approach to the analysis of parameterized nonlinear approximations, based on the assumption of continuous selection of their parameters. We are interested in the following lower bound for the complexity of approximations in $\mathcal{W}^{n,\infty}([0, 1]^d)$.

Theorem 3 (DeVore et al., (1989), Theorem 4.2)**.** *Fix d, n. Let W be a positive integer and $\eta : \mathbb{R}^W \to C([0, 1]^d)$ be any mapping between the space $\mathbb{R}^W$ and the space $C([0, 1]^d)$. Suppose that there is a continuous map $\mathcal{M} : F_{d,n} \to \mathbb{R}^W$ such that $\|f - \eta(\mathcal{M}(f))\|_\infty \le \epsilon$ for all $f \in F_{d,n}$. Then $W \ge c\epsilon^{-d/n}$, with some constant c depending only on n.*

We apply this theorem by taking η to be some ReLU network architecture, and $\mathbb{R}^W$ the corresponding weight space. It follows that if a ReLU network architecture is capable of expressing any function from $F_{d,n}$ with error ϵ, then, under the hypothesis of continuous weight selection, the network must have at least $c\epsilon^{-d/n}$ weights. The number of connections is then lower bounded by $\frac{c}{2}\epsilon^{-d/n}$ (since the number of weights is not larger than the sum of the number of computation units and the number of connections, and there are at least as many connections as units).

The hypothesis of continuous weight selection is crucial in Theorem 3. By examining our proof of the counterpart upper bound $O(\epsilon^{-d/n}\ln(1/\epsilon))$ in Theorem 1, the weights are selected there in a continuous manner, so this upper bound asymptotically lies above $c\epsilon^{-d/n}$ in agreement with Theorem 3. We remark, however, that the optimal choice of the network weights (minimizing the error) is known to be discontinuous in general, even for shallow networks (Kainen, Kůrková, & Vogt, 1999).

We also compare the bounds of Theorems 2 and 3. In the case $d = n = 1$, Theorem 3 provides a lower bound $\frac{c}{\epsilon}$ for the number of weights and connections. On the other hand, in the adaptive architecture scenario, Theorem 2 provides the upper bound $\frac{c}{\epsilon \ln(1/\epsilon)}$ for the number of weights, connections and computation units. The fact that this latter bound is asymptotically below the bound of Theorem 3 reflects the extra expressiveness associated with variable network architecture.

4.2. Bounds based on VC-dimension

In this section we consider the setup where the same network architecture is used to approximate all functions $f \in F_{d,n}$, but the dependence of the weights on f is not assumed to be necessarily continuous. In this setup, some lower bounds on the network complexity can be obtained as a consequence of existing upper bounds on VC-dimension of networks with piece-wise polynomial activation functions and Boolean outputs (Anthony & Bartlett, 2009). In the next theorem, part (a) is a more general but weaker bound, while part (b) is a stronger bound assuming a constrained growth of the network depth.

Theorem 4. *Fix d, n.*

(a) *For any $\epsilon \in (0, 1)$, a ReLU network architecture capable of approximating any function $f \in F_{d,n}$ with error ϵ must have at least $c\epsilon^{-d/(2n)}$ weights, with some constant $c = c(d, n) > 0$.*
(b) *Let $p \geq 0, c_1 > 0$ be some constants. For any $\epsilon \in (0, \frac{1}{2})$, if a ReLU network architecture of depth $L \leq c_1 \ln^p(1/\epsilon)$ is capable of approximating any function $f \in F_{d,n}$ with error ϵ, then the network must have at least $c_2\epsilon^{-d/n}\ln^{-2p-1}(1/\epsilon)$ weights, with some constant $c_2 = c_2(d, n, p, c_1) > 0$.*[1]

Proof. Recall that given a class H of Boolean functions on $[0, 1]^d$, the VC-dimension of H is defined as the size of the largest shattered subset $S \subset [0, 1]^d$, i.e. the largest subset on which H can compute any dichotomy (see, e.g., (Anthony & Bartlett, 2009), Section 3.3). We are interested in the case when H is the family of functions obtained by applying thresholds $\mathbb{1}(x > a)$ to a ReLU network with fixed architecture but variable weights. In this case Theorem 8.7 of Anthony and Bartlett (2009) implies that

$$\mathrm{VCdim}(H) \leq c_3 W^2, \tag{33}$$

and Theorem 8.8 implies that

$$\mathrm{VCdim}(H) \leq c_3 L^2 W \ln W, \tag{34}$$

where W is the number of weights, L is the network depth, and c_3 is an absolute constant.

Given a positive integer N to be chosen later, choose S as a set of N^d points $\mathbf{x}_1, \ldots, \mathbf{x}_{N^d}$ in the cube $[0, 1]^d$ such that the distance between any two of them is not less than $\frac{1}{N}$. Given any assignment of values $y_1, \ldots, y_{N^d} \in \mathbb{R}$, we can construct a smooth function f satisfying $f(\mathbf{x}_m) = y_m$ for all m by setting

$$f(\mathbf{x}) = \sum_{m=1}^{N^d} y_m \phi(N(\mathbf{x} - \mathbf{x}_m)), \tag{35}$$

with some C^∞ function $\phi : \mathbb{R}^d \to \mathbb{R}$ such that $\phi(\mathbf{0}) = 1$ and $\phi(\mathbf{x}) = 0$ if $|\mathbf{x}| > \frac{1}{2}$.

[1] The author thanks Matus Telgarsky for suggesting this part of the theorem.

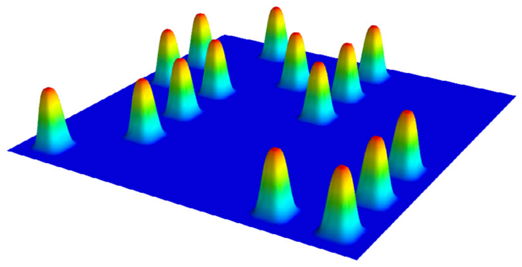

Fig. 5. A function f considered in the proof of Theorem 2 (for $d = 2$).

Let us obtain a condition ensuring that such $f \in F_{d,n}$. For any multi-index $\mathbf{n}$,

$$\max_{\mathbf{x}} |D^{\mathbf{n}} f(\mathbf{x})| = N^{|\mathbf{n}|} \max_m |y_m| \max_{\mathbf{x}} |D^{\mathbf{n}} \phi(\mathbf{x})|,$$

so if

$$\max_m |y_m| \leq c_4 N^{-n}, \tag{36}$$

with the constant $c_4 = (\max_{\mathbf{n}:|\mathbf{n}|\leq n} \max_{\mathbf{x}} |D^{\mathbf{n}} \phi(\mathbf{x})|)^{-1}$, then $f \in F_{d,n}$.

Now set

$$\epsilon = \frac{c_4}{3} N^{-n}. \tag{37}$$

Suppose that there is a ReLU network architecture η that can approximate, by adjusting its weights, any $f \in F_{d,n}$ with error less than ϵ. Denote by $\eta(\mathbf{x}, \mathbf{w})$ the output of the network for the input vector $\mathbf{x}$ and the vector of weights $\mathbf{w}$.

Consider any assignment $\mathbf{z}$ of Boolean values $z_1, \ldots, z_{N^d} \in \{0, 1\}$. Set

$$y_m = z_m c_4 N^{-n}, \quad m = 1, \ldots, N^d,$$

and let f be given by (35) (see Fig. 5); then (36) holds and hence $f \in F_{d,n}$.

By assumption, there is then a vector of weights, $\mathbf{w} = \mathbf{w_z}$, such that for all m we have $|\eta(\mathbf{x}_m, \mathbf{w_z}) - y_m| \leq \epsilon$, and in particular

$$\eta(\mathbf{x}_m, \mathbf{w_z}) \begin{cases} \geq c_4 N^{-n} - \epsilon > c_4 N^{-n}/2, & \text{if } z_m = 1, \\ \leq \epsilon < c_4 N^{-n}/2, & \text{if } z_m = 0, \end{cases}$$

so the thresholded network $\eta_1 = \mathbb{1}(\eta > c_4 N^{-n}/2)$ has outputs

$$\eta_1(\mathbf{x}_m, \mathbf{w_z}) = z_m, \quad m = 1, \ldots, N^d.$$

Since the Boolean values z_m were arbitrary, we conclude that the subset S is shattered and hence

$$\mathrm{VCdim}(\eta_1) \geq N^d.$$

Expressing N through ϵ with (37), we obtain

$$\mathrm{VCdim}(\eta_1) \geq \left(\frac{3\epsilon}{c_4}\right)^{-d/n}. \tag{38}$$

To establish part (a) of the Theorem, we apply bound (33) to the network η_1:

$$\mathrm{VCdim}(\eta_1) \leq c_3 W^2, \tag{39}$$

where W is the number of weights in η_1, which is the same as in η if we do not count the threshold parameter. Combining (38) with (39), we obtain the desired lower bound $W \geq c\epsilon^{-d/(2n)}$ with $c = (c_4/3)^{d/(2n)} c_3^{-1/2}$.

To establish part (b) of the theorem, we use bound (34) and the hypothesis $L \leq c_1 \ln^p(1/\epsilon)$:

$$\mathrm{VCdim}(\eta_1) \leq c_3 c_1^2 \ln^{2p}(1/\epsilon) W \ln W. \tag{40}$$

Combining (38) with (40), we obtain

$$W \ln W \geq \frac{1}{c_3 c_1^2}\left(\frac{3\epsilon}{c_4}\right)^{-d/n} \ln^{-2p}(1/\epsilon). \quad (41)$$

Trying a W of the form $W_{c_2} = c_2\epsilon^{-d/n}\ln^{-2p-1}(1/\epsilon)$ with a constant c_2, we get

$$\begin{aligned} W_{c_2} \ln W_{c_2} &= c_2\epsilon^{-d/n}\ln^{-2p-1}(1/\epsilon) \\ &\quad\times\left(\frac{d}{n}\ln(1/\epsilon) + \ln c_2 - (2p+1)\ln\ln(1/\epsilon)\right) \\ &= \left(c_2\frac{d}{n} + o(1)\right)\epsilon^{-d/n}\ln^{-2p}(1/\epsilon). \end{aligned}$$

Comparing this with (41), we see that if we choose $c_2 < (c_4/3)^{d/n} n/(dc_3c_1^2)$, then for sufficiently small ϵ we have $W \ln W \geq W_{c_2} \ln W_{c_2}$ and hence $W \geq W_{c_2}$, as claimed. We can ensure that $W \geq W_{c_2}$ for all $\epsilon \in (0, \frac{1}{2})$ by further decreasing c_2. $\square$

We remark that the constrained depth hypothesis of part (b) is satisfied, with $p = 1$, by the architecture used for the upper bound in Theorem 1. The bound stated in part (b) of Theorem 4 matches the upper bound of Theorem 1 and the lower bound of Theorem 3 up to a power of $\ln(1/\epsilon)$.

4.3. Adaptive network architectures

Our goal in this section is to obtain a lower bound for the approximation complexity in the scenario where the network architecture may depend on the approximated function. This lower bound is thus a counterpart to the upper bound of Section 3.3.

To state this result we define the complexity $\mathcal{N}(f, \epsilon)$ of approximating the function f with error ϵ as the minimal number of hidden computation units in a ReLU network that provides such an approximation.

Theorem 5. *For any d, n, there exists $f \in \mathcal{W}^{n,\infty}([0, 1]^d)$ such that $\mathcal{N}(f, \epsilon)$ is not $o(\epsilon^{-d/(9n)})$ as $\epsilon \to 0$.*

The proof relies on the following lemma.

Lemma 3. *Fix d, n. For any sufficiently small $\epsilon > 0$ there exists $f_\epsilon \in F_{d,n}$ such that $\mathcal{N}(f_\epsilon, \epsilon) \geq c_1\epsilon^{-d/(8n)}$, with some constant $c_1 = c_1(d, n) > 0$.*

Proof. Observe that all the networks with not more than m hidden computation units can be embedded in the single "enveloping" network that has m hidden layers, each consisting of m units, and that includes all the connections between units not in the same layer (see Fig. 6(a)). The number of weights in this enveloping network is $O(m^4)$. On the other hand, Theorem 4(a) states that at least $c\epsilon^{-d/(2n)}$ weights are needed for an architecture capable of ϵ-approximating any function in $F_{d,n}$. It follows that there is a function $f_\epsilon \in F_{d,n}$ that cannot be ϵ-approximated by networks with fewer than $c_1\epsilon^{-d/(8n)}$ computation units. $\square$

Before proceeding to the proof of Theorem 5, note that $\mathcal{N}(f, \epsilon)$ is a monotone decreasing function of ϵ with a few obvious properties:

$$\mathcal{N}(af, |a|\epsilon) = \mathcal{N}(f, \epsilon), \text{ for any } a \in \mathbb{R} \setminus \{0\} \quad (42)$$

(follows by multiplying the weights of the output unit of the approximating network by a constant);

$$\mathcal{N}(f \pm g, \epsilon + \|g\|_\infty) \leq \mathcal{N}(f, \epsilon) \quad (43)$$

(follows by approximating $f \pm g$ by an approximation of f);

$$\mathcal{N}(f_1 \pm f_2, \epsilon_1 + \epsilon_2) \leq \mathcal{N}(f_1, \epsilon_1) + \mathcal{N}(f_2, \epsilon_2) \quad (44)$$

(follows by combining approximating networks for f_1 and f_2 as in Fig. 6(b)).

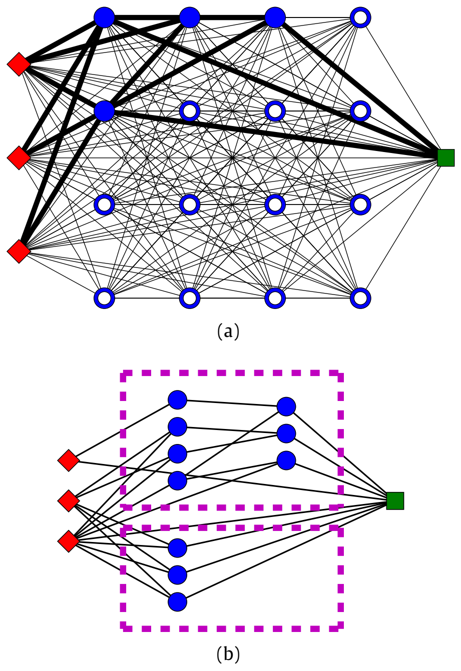

Fig. 6. (a) Embedding a network with $m = 4$ hidden units into an "enveloping" network (see Lemma 3). (b) Putting sub-networks in parallel to form an approximation for the sum or difference of two functions, see Eq. (44).

Proof (*Proof of Theorem 5*)**.** The claim of Theorem 5 is similar to the claim of Lemma 3, but is about a single function f satisfying a slightly weaker complexity bound at multiple values of $\epsilon \to 0$. We will assume that Theorem 5 is false, i.e.,

$$\mathcal{N}(f, \epsilon) = o(\epsilon^{-d/(9n)}) \quad (45)$$

for all $f \in \mathcal{W}^{n,\infty}([0, 1]^d)$, and we will reach contradiction by presenting f violating this assumption. Specifically, we construct this f as

$$f = \sum_{k=1}^{\infty} a_k f_k, \quad (46)$$

with some $a_k \in \mathbb{R}, f_k \in F_{d,n}$, and we will make sure that

$$\mathcal{N}(f, \epsilon_k) \geq \epsilon_k^{-d/(9n)} \quad (47)$$

for a sequence of $\epsilon_k \to 0$.

We determine a_k, f_k, ϵ_k sequentially. Suppose we have already found $\{a_s, f_s, \epsilon_s\}_{s=1}^{k-1}$; let us describe how we define a_k, f_k, ϵ_k.

First, we set

$$a_k = \min_{s=1,\ldots,k-1} \frac{\epsilon_s}{2^{k-s}}. \quad (48)$$

In particular, this ensures that

$$a_k \leq \epsilon_1 2^{1-k},$$

so that the function f defined by the series (46) will be in $\mathcal{W}^{n,\infty}([0, 1]^d)$, because $\|f_k\|_{\mathcal{W}^{n,\infty}([0,1]^d)} \leq 1$.

Next, using Lemma 3 and Eq. (42), observe that if ϵ_k is sufficiently small, then we can find $f_k \in F_{d,n}$ such that

$$\mathcal{N}(a_k f_k, 3\epsilon_k) = \mathcal{N}\left(f_k, \frac{3\epsilon_k}{a_k}\right) \geq c_1\left(\frac{3\epsilon_k}{a_k}\right)^{-d/(8n)} \geq 2\epsilon_k^{-d/(9n)}. \quad (49)$$

In addition, by assumption (45), if ϵ_k is small enough then

$$\mathcal{N}\left(\sum_{s=1}^{k-1} a_s f_s, \epsilon_k\right) \leq \epsilon_k^{-d/(9n)}. \quad (50)$$

Let us choose ϵ_k and f_k so that both (49) and (50) hold. Obviously, we can also make sure that $\epsilon_k \to 0$ as $k \to \infty$.

Let us check that the above choice of $\{a_k, f_k, \epsilon_k\}_{k=1}^\infty$ ensures that inequality (47) holds for all k:

$$\begin{aligned}\mathcal{N}\Big(\sum_{s=1}^{\infty} a_s f_s, \epsilon_k\Big) &\geq \mathcal{N}\Big(\sum_{s=1}^{k} a_s f_s, \epsilon_k + \Big\|\sum_{s=k+1}^{\infty} a_s f_s\Big\|_\infty\Big)\\ &\geq \mathcal{N}\Big(\sum_{s=1}^{k} a_s f_s, \epsilon_k + \sum_{s=k+1}^{\infty} a_s\Big)\\ &\geq \mathcal{N}\Big(\sum_{s=1}^{k} a_s f_s, 2\epsilon_k\Big)\\ &\geq \mathcal{N}(a_k f_k, 3\epsilon_k) - \mathcal{N}\Big(\sum_{s=1}^{k-1} a_s f_s, \epsilon_k\Big)\\ &\geq \epsilon^{-d/(9n)}.\end{aligned}$$

Here in the first step we use inequality (43), in the second the monotonicity of $\mathcal{N}(f, \epsilon)$, in the third the monotonicity of $\mathcal{N}(f, \epsilon)$ and the setting (48), in the fourth the inequality (44), and in the fifth the conditions (49) and (50). □

4.4. Slow approximation of smooth functions by shallow networks

In this section we show that, in contrast to deep ReLU networks, shallow ReLU networks relatively inefficiently approximate sufficiently smooth (C^2) nonlinear functions. We remark that Liang and Srikant (2016) prove a similar result assuming global convexity instead of smoothness and nonlinearity.

Theorem 6. *Let $f \in C^2([0,1]^d)$ be a nonlinear function (i.e., not of the form $f(x_1,\ldots,x_d) \equiv a_0 + \sum_{k=1}^d a_k x_k$ on the whole $[0,1]^d$). Then, for any fixed L, a depth-L ReLU network approximating f with error $\epsilon \in (0,1)$ must have at least $c\epsilon^{-1/(2(L-2))}$ weights and computation units, with some constant $c = c(f, L) > 0$.*

Proof. Since $f \in C^2([0,1]^d$ and f is nonlinear, we can find $\mathbf{x}_0 \in [0,1]^d$ and $\mathbf{v} \in \mathbb{R}^d$ such that $\mathbf{x}_0 + x\mathbf{v} \in [0,1]^d$ for all $x \in [-1,1]$ and the function $f_1 : x \mapsto f(\mathbf{x}_0 + x\mathbf{v})$ is strictly convex or concave on $[-1,1]$. Suppose without loss of generality that it is strictly convex:

$$\min_{x\in[-1,1]} f_1''(x) = c_1 > 0. \tag{51}$$

Suppose that $\widetilde{f}$ is an ϵ-approximation of function f, and let $\widetilde{f}$ be implemented by a ReLU network η of depth L. Let $\widetilde{f}_1 : x \mapsto \widetilde{f}(\mathbf{x}_0 + x\mathbf{v})$. Then $\widetilde{f}_1$ also approximates f_1 with error not larger than ϵ. Moreover, since $\widetilde{f}_1$ is obtained from $\widetilde{f}$ by a linear substitution $\mathbf{x} = \mathbf{x}_0 + x\mathbf{v}$, $\widetilde{f}_1$ can be implemented by a ReLU network η_1 of the same depth L and with the number of units and weights not larger than in η (we can obtain η_1 from η by replacing the input layer in η with a single unit, accordingly modifying the input connections, and adjusting the weights associated with these connections). It is thus sufficient to establish the claimed bounds for η_1.

By construction, $\widetilde{f}_1$ is a continuous piece-wise linear function of x. Denote by M the number of linear pieces in $\widetilde{f}_1$. We will use the following counting lemma.

Lemma 4. *$M \leq (2U)^{L-2}$, where U is the number of computation units in η_1.*

Proof. This bound, up to minor details, is proved in Lemma 2.1 of Telgarsky (2015). Precisely, Telgarsky's lemma states that if a network has a single input, connections only between neighboring layers, at most m units in a layer, and a piece-wise linear activation function with t pieces, then the number of linear pieces in the output of the network is not greater than $(tm)^L$. By examining the proof of the lemma we see that it will remain valid for networks with connections not necessarily between neighboring layers, if we replace m by U in the expression $(tm)^L$. Moreover, we can slightly strengthen the bound by noting that in the present paper the input and output units are counted as separate layers, only units of layers 3 to L have multiple incoming connections, and the activation function is applied only in layers 2 to $L-1$. By following Telgarsky's arguments, this gives the slightly more accurate bound $(tU)^{L-2}$ (i.e., with the power $L-2$ instead of L). It remains to note that the ReLU activation function corresponds to $t = 2$. □

Lemma 4 implies that there is an interval $[a,b] \subset [-1,1]$ of length not less than $2(2U)^{-(L-2)}$ on which the function $\widetilde{f}_1$ is linear. Let $g = f_1 - \widetilde{f}_1$. Then, by the approximation accuracy assumption, $\sup_{x\in[a,b]}|g(x)| \leq \epsilon$, while by (51) and by the linearity of $\widetilde{f}_1$ on $[a,b]$, $\max_{x\in[a,b]} g''(x) \geq c_1 > 0$. It follows that $\max(g(a), g(b)) \geq g(\frac{a+b}{2}) + \frac{c_1}{2}(\frac{b-a}{2})^2$ and hence

$$\epsilon \geq \frac{1}{2}\Big(\max(g(a), g(b)) - g(\tfrac{a+b}{2})\Big) \geq \frac{c_1}{4}\Big(\frac{b-a}{2}\Big)^2 \geq \frac{c_1}{4}(2U)^{-2(L-2)},$$

which implies the claimed bound $U \geq \frac{1}{2}(\frac{4\epsilon}{c_1})^{-1/(2(L-2))}$. Since there are at least as many weights as computation units in a network, a similar bound holds for the number of weights. □

5. Discussion

We discuss some implications of the obtained bounds.

Deep vs. shallow ReLU approximations of smooth functions. Our results clearly show that deep ReLU networks more efficiently express smooth functions than shallow ReLU networks. By Theorem 1, functions from the Sobolev space $\mathcal{W}^{n,\infty}([0,1]^d)$ can be ϵ-approximated by ReLU networks with depth $O(\ln(1/\epsilon))$ and the number of computation units $O(\epsilon^{-d/n}\ln(1/\epsilon))$. In contrast, by Theorem 6, a nonlinear function from $C^2([0,1]^d)$ cannot be ϵ-approximated by a ReLU network of fixed depth L with the number of units less than $c\epsilon^{-1/(2(L-2))}$. In particular, it follows that in terms of the required number of computation units, unbounded-depth approximations of functions from $\mathcal{W}^{n,\infty}([0,1]^d)$ are asymptotically strictly more efficient than approximations with a fixed depth L at least when

$$\frac{d}{n} < \frac{1}{2(L-2)}$$

(assuming also $n > 2$, so that $\mathcal{W}^{n,\infty}([0,1]^d) \subset C^2([0,1]^d)$). The efficiency of depth is even more pronounced for very smooth functions such as polynomials, which can be implemented by deep networks using only $O(\ln(1/\epsilon))$ units (cf. Propositions 2 and 3 and the proof of Theorem 1). Liang and Srikant describe in Liang and Srikant (2016) some conditions on the approximated function (resembling conditions of local analyticity) under which complexity of deep ϵ-approximation is $O(\ln^c(1/\epsilon))$ with a constant power c.

Continuous model selection vs. function-dependent network architectures. When approximating a function by a neural network, one can either view the network architecture as fixed and only tune the weights, or optimize the architecture as well. Moreover, when tuning the weights, one can either require them to continuously depend on the approximated function or not. We naturally expect that more freedom in the choice of the approximation should lead to higher expressiveness.

Our bounds confirm this expectation to a certain extent. Specifically, the complexity of ϵ-approximation of functions from the unit ball $F_{1,1}$ in $\mathcal{W}^{1,\infty}([0,1])$ is lower bounded by $\frac{c}{\epsilon}$ in the scenario with a fixed architecture and continuously selected weights (see

Theorem 3). On the other hand, we show in Theorem 2 that this complexity is upper bounded by $O(\frac{1}{\epsilon \ln(1/\epsilon)})$ if we are allowed to adjust the network architecture. This bound is achieved by finite-depth (depth-6) ReLU networks using the idea of reused subnetworks familiar from the theory of Boolean circuits (Shannon, 1949).

In the case of fixed architecture, we have not established any evidence of complexity improvement for unconstrained weight selection compared to continuous weight selection. We remark however that, already for approximations with depth-3 networks, the optimal weights are known to discontinuously depend, in general, on the approximated function (Kainen et al., 1999). On the other hand, part b) of Theorem 4 shows that if the network depth scales as $O(\ln^p(1/\epsilon))$, discontinuous weight selection cannot improve the continuous-case complexity more than by a factor being some power of $\ln(1/\epsilon)$.

Upper vs. lower complexity bounds. We indicate the gaps between respective upper and lower bounds in the three scenarios mentioned above: fixed architectures with continuous selection of weights, fixed architectures with unconstrained selection of weights, or adaptive architectures.

For fixed architectures with continuous selection the lower bound $c\epsilon^{-d/n}$ is provided by Theorem 3, and the upper bound $O(\epsilon^{-d/n}\ln(1/\epsilon))$ by Theorem 1, so these bounds are tight up to a factor $O(\ln(1/\epsilon))$.

In the case of fixed architecture but unconstrained selection, part b) of Theorem 4 gives a lower bound $c\epsilon^{-d/n}\ln^{-2p-1}(1/\epsilon)$ under assumption that the depth is constrained by $O(\ln^p(1/\epsilon))$. This is only different by a factor of $O(\ln^{2p+2}(1/\epsilon))$ from the upper bound of Theorem 1. Without this depth constraint we only have the significantly weaker bound $c\epsilon^{-d/(2n)}$ (part a) of Theorem 4).

In the case of adaptive architectures, there is a big gap between our upper and lower bounds. The upper bound $O(\frac{1}{\epsilon \ln(1/\epsilon)})$ is given by Theorem 2 for $d = n = 1$. The lower bound, proved for general d, n in Theorem 5, only states that there are $f \in \mathcal{W}^{n,\infty}([0,1]^d)$ for which the complexity is not $o(\epsilon^{-d/(9n)})$.

ReLU vs. smooth activation functions. A popular general-purpose method of approximation is shallow (depth-3) networks with smooth activation functions (e.g., logistic sigmoid). Upper and lower approximation complexity bounds for these networks (Maiorov & Meir, 2000; Mhaskar, 1996) show that complexity scales as $\sim \epsilon^{-d/n}$ up to some $\ln(1/\epsilon)$ factors. Comparing this with our bounds in Theorems 1, 2, 4, it appears that deep ReLU networks are roughly (up to $\ln(1/\epsilon)$ factors) as expressive as shallow networks with smooth activation functions.

6. Conclusion

We have established several upper and lower bounds for the expressive power of deep ReLU networks in the context of approximation in Sobolev spaces. We should note, however, that this setting may not quite reflect typical real world applications, which usually possess symmetries and hierarchical and other structural properties substantially narrowing the actually interesting classes of approximated functions (LeCun et al., 2015). Some recent publications introduce and study expressive power of deep networks in frameworks bridging this gap, in particular, graph-based hierarchical approximations are studied in Mhaskar et al. (2016); Mhaskar and Poggio (2016) and convolutional arithmetic circuits in Cohen, Sharir, and Shashua (2015). Theoretical analysis of expressiveness of deep networks taking into account such properties of real data seems to be an important and promising direction of future research.

Acknowledgments

The author thanks Matus Telgarsky and the anonymous referees for multiple helpful comments on the preliminary versions of the paper. The research was funded by Skolkovo Institute of Science and Technology (project "Recognition of sparse geometric data", 2016).

References

Anthony, Martin, & Bartlett, Peter L. (2009). *Neural network learning: Theoretical foundations*. Cambridge University Press.

Bartlett, Peter L., Maiorov, Vitaly, & Meir, Ron (1998). Almost linear VC-dimension bounds for piecewise polynomial networks. *Neural Computation*, *10*(8), 2159–2173.

Bianchini, Monica, & Scarselli, Franco (2014). On the complexity of neural network classifiers: a comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(8), 1553–1565.

Cohen, Nadav, Sharir, Or, & Shashua, Amnon (2015). On the Expressive Power of Deep Learning: A Tensor Analysis. ArXiv preprint ArXiv:1509.05009.

Dawson, C. M., & Nielsen, M. A. (2006). The Solovay-Kitaev algorithm. *Quantum Information & Computation*, *6*(1), 81–95.

Delalleau, Olivier, & Bengio, Yoshua (2011). Shallow vs. deep sum-product networks. *Advances in Neural Information Processing Systems*. (pp. 666–674).

DeVore, Ronald A., Howard, Ralph, & Micchelli, Charles (1989). Optimal nonlinear approximation. *Manuscripta Mathematica*, *63*(4), 469–478.

Goldberg, Paul W., & Jerrum, Mark R. (1995). Bounding the Vapnik-Chervonenkis dimension of concept classes parameterized by real numbers. *Machine Learning*, *18*(2–3), 131–148.

Kainen, Paul C., Kůrková, Věra, & Vogt, Andrew (1999). Approximation by neural networks is not continuous. *Neurocomputing*, *29*(1), 47–56.

Kearns, Michael J., & Schapire, Robert E. (1990). Efficient distribution-free learning of probabilistic concepts. In *Foundations of computer science, 1990. Proceedings., 31st annual symposium on* (pp. 382–391). IEEE.

Kitaev, A. Yu., Shen, A. H., & Vyalyi, M. N. (2002). *Classical and quantum computation*. ISBN: 0821832298, Boston, MA, USA: American Mathematical Society.

LeCun, Yann, Bengio, Yoshua, & Hinton, Geoffrey (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Liang, Shiyu, & Srikant, R. (2016). Why Deep Neural Networks? ArXiv Preprint ArXiv:1610.04161.

Maiorov, V. E., & Meir, Ron (2000). On the near optimality of the stochastic approximation of smooth functions by neural networks. *Advances in Computational Mathematics*, *13*(1), 79–103.

Mhaskar, H. N. (1996). Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, *8*(1), 164–177.

Mhaskar, Hrushikesh Narhar (1993). Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics*, *1*(1), 61–80.

Mhaskar, Hrushikesh, Liao, Qianli, & Poggio, Tomaso (2016). Learning Real and Boolean Functions: When Is Deep Better Than Shallow. ArXiv Preprint ArXiv:1603.00988.

Mhaskar, Hrushikesh, & Poggio, Tomaso (2016). Deep vs. shallow networks: An approximation theory perspective. ArXiv Preprint ArXiv:1608.03287.

Montufar, Guido F., Pascanu, Razvan, Cho, Kyunghyun, & Bengio, Yoshua (2014). On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems* (pp. 2924–2932).

Pinkus, Allan (1999). Approximation theory of the MLP model in neural networks. *Acta Numerica*, *8*, 143–195.

Raghu, Maithra, Poole, Ben, Kleinberg, Jon, Ganguli, Surya, & Sohl-Dickstein, Jascha (2016). On the expressive power of deep neural networks. ArXiv Preprint ArXiv:1606.05336.

Sakurai, Akito (1999). Tight Bounds for the VC-Dimension of Piecewise Polynomial Networks. In *Advances in Neural Information Processing Systems* (pp. 323–329).

Schmidhuber, Jürgen (2015). Deep learning in neural networks: an overview. *Neural Networks*, *61*, 85–117.

Shannon, Claude (1949). The synthesis of two-terminal switching circuits. *Bell Labs Technical Journal*, *28*(1), 59–98.

Telgarsky, Matus (2015). Representation Benefits of Deep Feedforward Networks. ArXiv Preprint ArXiv:1509.08101.

Telgarsky, Matus (2016). Benefits of depth in neural networks. ArXiv Preprint ArXiv:1602.04485.

Vapnik, Vladimir N., & Chervonenkis, A. Ya (2015). On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity* (pp. 11–30). Springer.

Warren, Hugh E. (1968). Lower bounds for approximation by nonlinear manifolds. *Transactions of the American Mathematical Society*, *133*(1), 167–178.