Synthesis of Feedforward Networks in Supremum Error Bound

Krzysztof Ciesielski, Jarosław P. Sacha, and Krzysztof J. Cios

K. Ciesielski is with the Department of Mathematics, West Virginia University, Morgantown, USA; e-mail: K_Cies@math.wvu.edu

J. P. Sacha; e-mail: Jarek@IEEE.org

K. J. Cios is with the University of Colorado at Denver, USA; e-mail: KCios@carbon.cudenver.edu

Abstract

The main result of this paper is a constructive proof of a formula for the upper bound of the approximation error in L_∞ (supremum norm) of multi-dimensional functions by feedforward networks with one hidden layer of sigmoidal units and a linear output. This result is applied to formulate a new method of neural network synthesis. The result can also be used to estimate complexity of the maximum-error network and/or to initialize that network weights. An example of the network synthesis is given.

Keywords

Approximation error, supremum norm, synthesis, feedforward networks, sigmoidal functions, complexity, Radon transform.

I. Introduction

Universal approximation capabilities for a broad range of neural network topologies have been established by researchers like Cybenko [1], Funahashi [2], Hornik *et al.* [3], Ito [4], and Leshno *et al.* [5]. Their work focused on the existence of an approximating network and was fundamental for establishing validity of artificial neural networks.

Problem of complexity and realizability of networks with two hidden layers using step (threshold) units was addressed by Blum and Li [6]. The authors presented a method for construction of such networks and specified upper bound of the approximation error. Korain [7] extended approach of Funahashi [2] and established a method for constructing single hidden layer networks with step function neurons and a linear output; he also estimated complexity and the upper bound of the approximation error. A network using either step, ramp, or sigmoid functions was studied by Bulsari [8]. He gave the upper bound of approximation error for networks with one-dimensional input using step and ramp functions. Realizability of networks based on Kolmogorov's representation theorem was investigated by Kůrková [9] and Sprecher [10]. The latter presented a method for constructing transfer functions for hidden layers neurons. Sprecher [11] discussed numerical aspects of implementing Kolmogorov's representation approach.

More recently an important result has been established for approximation in the L_2 norm or mean squared error. Several authors (Jones [12], Barron [13], Girosi and Anzellotti [14], Murata [15]) have shown for various classes of sufficiently smooth functions that feedforward networks, with one hidden layer of sigmoidal nodes, achieve integrated square error of order $O(1/N)$, where N is the number of neurons in the hidden layer. This result is significant since it demonstrates that approximation with neural networks in L_2 norm can be free of the "curse of dimensionality," i.e., approximation rate is independent of the dimensionality of the space. However, the "curse" was avoided only in terms of the accuracy of approximation but not in terms of computational complexity [13].

Learning algorithms like backpropagation are based on minimization of the error in L_2 norm (mean square error). A number of applications exist, however, where approximation in the L_∞ norm (maximum error) is more advantageous then using L_2 norm. A typical example is manipulation of a robot arm, or other control applications, where one requires that the maximum deviation be within certain bounds, e.g., safety limits. To that end an interesting new learning algorithm that performs minimization of the error in L_∞ norm have been proposed by Meltser *et al.* [16]. The algorithm is used for networks with a hidden layer of sigmoidal nodes and a linear output layer.

The main result of this paper is a constructive proof of a formula for the upper bound of the approximation error in L_∞ (supremum norm) of multi-dimensional functions by feedforward networks with one hidden layer of sigmoidal units and a linear output. This result is used to formulate a new method of neural network synthesis. The method specifies network topology and all its weights and can be used to initialize learning algorithms based on the L_∞ norm, like the one proposed by Meltser *et al.* [16], or simply to estimate complexity of a maximum-error network. Note, that the synthesis method presented here does not provide for direct learning from examples, it assumes that full information about the function being approximated is given.

In what follows we shall consider a one-hidden-layer network:

$$\mathcal{N}_N(x) \;=\; c_0 + \sum_{i=1}^{N} \left(c_i \, \phi \left(w_{i0} + \sum_{j=1}^{n} w_{ij} x_j \right) \right), \quad \mathcal{N}_N : \mathbf{R}^n \mapsto \mathbf{R}^1. \tag{1}$$

For any bounded subset B of $\mathbf{R}^n$ we can find the one hidden layer network $\mathcal{N}$ for which

$$\sup_{x \in B} |f(x) - \mathcal{N}(x)| \; < \; \varepsilon,$$

where $f\colon \mathbf{R}^n \mapsto \mathbf{R}^1$ is the target function and ε is the desired approximation error. A constructive solution to this problem, in which the complexity of the approximating network (number of neurons in the hidden layer) is $O_\varepsilon(1/\varepsilon^n)$, will be presented.

The proposed network synthesis method consist of three stages. First, the problem of multi-dimensional approximation is replaced by several simpler one-dimensional problems. Second, each of the one-dimensional problems is solved by finding an approximation subnetwork. Third, the subnetworks are combined together to form the final approximation network.

In the first stage function f is approximated by a finite sum of plane waves, or ridge functions b_l:

$$f(x) \; \approx \; \nu \sum_{l=1}^{L} \mathsf{b}_l(x), \quad \mathsf{b}_l\colon \mathbf{R}^n \mapsto \mathbf{R}, \; \forall\, l,$$

where ν is a constant. A characteristic feature of a plain wave is that it can be represented by a one-dimensional function and a directional vector [17]. Let us denote this one-dimensional function by $b_l\colon \mathbf{R} \; \mapsto \; \mathbf{R}$ and the directional vector by $\omega^{(l)} \; \in \; \mathbf{R}^n$. Then

$$\mathsf{b}_l(x) \; = \; b_l(x \cdot \omega^{(l)}), \quad \forall\, x \in \mathbf{R}^n.$$

Figure 1 shows example of a plain wave:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \;\; \omega^{(l)} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \;\; b_l(x \cdot \omega^{(l)}) = \operatorname{sinc}(x \cdot \omega^{(l)}) = \operatorname{sinc}(2x_1 + x_2),$$

where

$$\operatorname{sinc}(t) \; = \; \frac{\sin(\pi t)}{\pi t}.$$

In the second stage of the method, one-dimensional functions b_l associated with plane waves are approximated be single-input single-output subnetworks $\aleph^{(l)}\colon \mathbf{R} \mapsto \mathbf{R}$. First, functions b_l are represented as a difference of two monotonic functions. Second, neural networks are synthesized to approximate each of these monotonic functions. The most significant implication of the

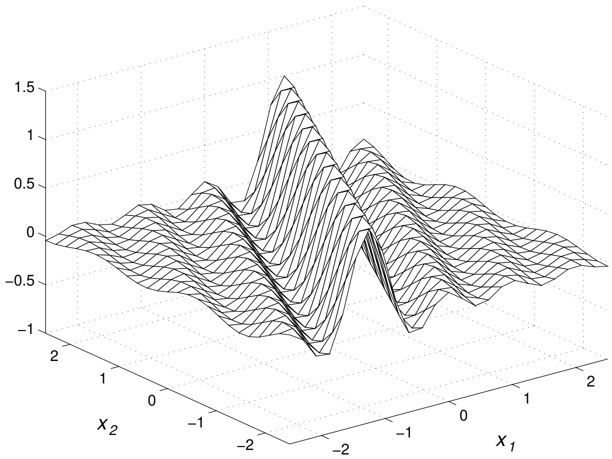

Fig. 1. A plain wave $\mathsf{b}_l(x_1, x_2) = \mathrm{sinc}(2x_1 + x_2)$.

second part of the method is constructive specification of the upper bound of an approximation error for networks with sigmoidal functions.

The third stage combines subnetworks $\aleph^{(l)}$ into the final approximation network $\mathcal{N}$:

$$\mathcal{N}(x) \;=\; \nu \sum_{l=1}^{L} \aleph^{(l)} \left(x \cdot \omega^{(l)} \right) .$$

The reduction of dimensionality (the first approximation stage) is done by means of the inverse Radon transform of function f. It is based on the proposed here revision of Carroll and Dickinson [18] method. Some of the constrains posed by these authors on the function being approximated will be eliminated and, at the same time, the upper bound of the approximation error, by a finite number of plain waves, will be lowered.

Carroll and Dickinson [18] used inverse Radon transform approach to provide an alternative proof of universal approximation properties of neural networks with mean square error norm. They first decompose a mutltidimensional function into scalar functions using inverse Radon transform. Then they approximate scalar functions with single layer neural networks using step-like approximation — weights of sigmoidal neurons are made to tend to infinity so that the neural transfer functions approach step functions. However, they do not provide actual values of network parameters needed to achieve the desired accuracy. (This part of their proof is existential.)

The method presented in this paper was inspired by the work of Carroll and Dickinson [18], but is similar to it only in the way it uses inverse Radon transform to decomposed multi-dimensional functions into scalar functions. In contrast to Carroll and Dickinson [18] we are interested in maximum approximation error rather than mean square error. We also perform approximation of scalar functions differently. We use finite neural weights and provide the actual values needed to achieve the desired error rate. The weights are assigned so that derivative of the neural transfer function near the center of the scaled-shifter sigmoid is finite and close to the derivative of the approximated function for the similar value of the argument.

Application of the inverse Radon transform for construction of representation networks was also studied by Ito [19]. He concentrated on delivering the results for uniform approximation on the entire space $\mathbf{R}^n$ (not on a compact subset) and use of a step and sigmoidal functions without scaling. However, his mainly constructive proof is existential in the part specifying partitioning for the approximation of the integral representation (not only of the unit sphere, as in our case, but of the entire space $\mathbf{R}^n$), and no estimation of the approximation error was given.

The method of network synthesis presented in this paper is non-iterative. The decision on the network size is made before the weights of the network are calculated. This decision is usually based on the required upper bound of the approximation error. The method can also be used to estimate number of neurons in the hidden layer given the approximation error limit and the target function, without the need of calculating network weights.

We start by presenting our initial result: the constructive proof that any one (real) variable monotone function can be approximated, with an arbitrary accuracy, by a neural network with sigmoidal transfer functions. Next, we show how this result can be used to approximate multi-dimension functions. We prove the upper bound of the maximum error of the multi-dimensional approximation and show an illustrative example of applying our method for network synthesis. Section V gives an example of applying the proposed method. Some remarks on practical use of the method are presented in Section VI. Proofs of the theorems and corollaries are given in the Appendix.

II. Approximation of One-dimensional Monotone Functions

We shall start with discussion of approximation with ramp transfer functions. It is included here because it illustrates well the intuition behind the more complex approach — approximation with sigmoidal transfer functions.

Let $\psi\colon \mathbf{R} \mapsto \mathbf{R}$ be continuous strictly increasing function defined on an interval $K = [a, b]$. Function ψ maps K onto interval $Q = [\psi(a), \psi(b)]$. Let us assume that the network used for approximation of ψ has N neurons in the hidden layer. Next, denote:

$$\begin{aligned} \Delta &= \psi(b) - \psi(a), \\ \delta_i &= \psi(a) + i\,\frac{\Delta}{N} \quad \text{for } \ i = 0, 1, \ldots, N, \end{aligned}$$

and divide interval Q into N equal subintervals $Q_i = [\delta_{i-1}, \delta_i]$, $i = 1, 2, \ldots, N$. Define:

$$\gamma_i = \psi^{-1}(\delta_i) \quad \text{for } \ i = 0, 1, \ldots, N.$$

A. *Ramp Transfer Functions*

Results presented in this section are similar to well-known results using polynomial and spline approximation. They are presented here in the context of neural networks. The following definition of the ramp function is used:

$$\phi(x) = \begin{cases} -1 & x < -1, \\ x & -1 \le x \le 1, \\ 1 & x > 1. \end{cases} \tag{2}$$

Let each of the neurons in the hidden layer correspond to one of the intervals Q_i defined above; they "approximate" function ψ in these intervals. Assume that the mapping of the network $\aleph$ and the function ψ take the same values at each γ_i:

$$\aleph(\gamma_i) = \psi(\gamma_i) = \delta_i \quad \text{for } \ i = 0, 1, \ldots, N.$$

Figure 2 illustrates this situation (the approximation network has $N = 5$ neurons in the hidden layer). Circles represent points (γ_i, δ_i) in which values of ψ and $\aleph$ are assumed to be the same.

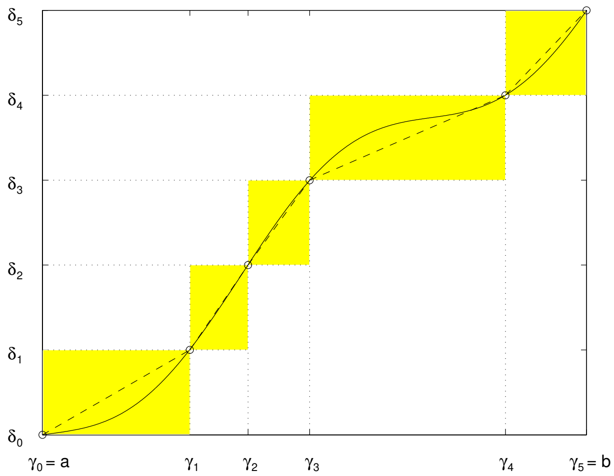

Fig. 2. Approximation of an increasing function with ramp transfer functions.

Shaded areas correspond to regions where the target function and its approximation are allowed to differ. Since both ψ and $\aleph$ are monotonically increasing the difference is less then $\delta_i - \delta_{i-1} = \frac{\Delta}{N}$. This value constitutes the upper error bound of approximating a monotonic function with ramp functions. It is summarized in the following proposition.

Proposition 1: Let function $\psi\colon \mathbf{R}^1 \mapsto \mathbf{R}^1$ be continuous strictly increasing on $[a, b]$, a neural network given by (1), the ramp transfer function ϕ given by (2), and the network weights be assigned as follows:

$$\begin{aligned} c_0 &= \frac{\psi(a) + \psi(b)}{2}, \\ c_i &= \frac{\psi(b) - \psi(a)}{2N}, \\ \gamma_i &= \psi^{-1}\left[\psi(a) + i\,\frac{\psi(b) - \psi(a)}{N}\right], \\ w_{i0} &= -\frac{\gamma_i + \gamma_{i-1}}{\gamma_i - \gamma_{i-1}}, \\ w_{i1} &= \frac{2}{\gamma_i - \gamma_{i-1}}. \end{aligned}$$

Then, the upper bound of the approximation error of function ψ by network $\aleph$ on interval $[a, b]$ is:

$$\sup_{x\in[a,b]} |\aleph(x) - \psi(x)| \;\leq\; \frac{\psi(b) - \psi(a)}{N}.$$

□

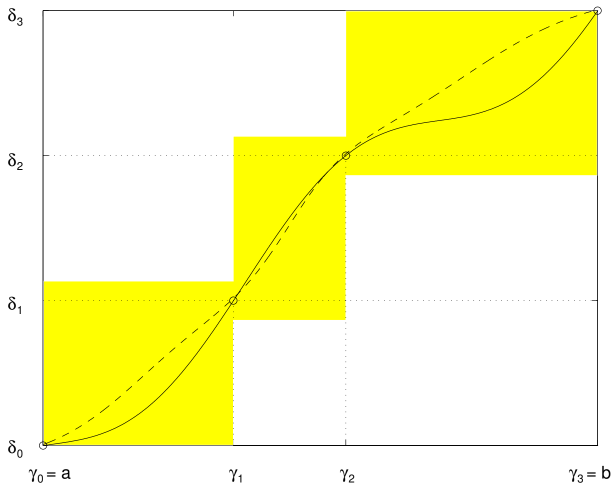

Fig. 3. Approximation of an increasing function with sigmoidal transfer functions.

B. *Sigmoidal Transfer Functions*

Let us assume that a sigmoidal transfer function is the hyperbolic tangent,

$$\phi(x) = \tanh(x).$$

The results presented in this section work for any other increasing continuous function ϕ with bounded range, but parameters have to be slightly changed.[1] We derive the result for the hyperbolic tangent since it is commonly used in neural networks.

Since ramp functions vary only on a finite interval, it was reasonable to require that ψ and $\aleph$ take the same values at points (γ_i, δ_i), see Figure 2. However, this requirement is not practical for a transfer function like hyperbolic tangent or other sigmoids which change their values on an infinite interval. In order to find the solution and the error's upper bound for sigmoids we shall relax this matching requirement at a cost of a slightly higher value of the approximation error, but without compromising the approximation rate. The following theorem specifies the network weights and the the upper bound of the approximation error.

Theorem 2: Let $\psi\colon [a, b] \to \mathbf{R}$ be a strictly increasing continuous function, and for every

[1] The parameters remain the same for any such ϕ for which $\phi(-x) = -\phi(x)$ and $\lim_{x\to\infty} \psi(x) = 1$.

natural number N let $\aleph_N : [a, b] \to \mathbf{R}$ be defined by the formula (1):

$$\aleph_N(x) = c_0 + \sum_{i=1}^{N} c_i \phi(w_{i1} x + w_{i0})$$

where $\phi(x) = \tanh x$ and parameters c_i's and w_{jk}'s are defined as below. Then

$$\sup_{x \in [a,b]} |\aleph_N(x) - \psi(x)| \; < \; \frac{3\Delta}{N},$$

where

$$\Delta \; = \; \psi(b) - \psi(a).$$

In particular $\aleph_N$ converges to ψ (as $N \to \infty$) in sup-norm. Here we use, for $i = 1, \ldots, N$,

$$\begin{aligned}
d_N &= \phi^{-1}\left(1 - \frac{1}{2N}\right), \\
\gamma_0 &= a, \\
\gamma_i &= \psi^{-1}\left(\psi(a) + i\frac{\Delta}{N}\right), \\
w_{i0} &= d_N - \frac{2d_N}{\gamma_i - \gamma_{i-1}}\,\gamma_i \;=\; -d_N \frac{\gamma_i + \gamma_{i-1}}{\gamma_i - \gamma_{i-1}}, \qquad (3) \\
w_{i1} &= \frac{2d_N}{\gamma_i - \gamma_{i-1}}, \\
c_i &= \frac{\Delta}{2N}, \\
c_0 &= \psi(a) - \sum_{i=1}^{N} c_i \phi(w_{i1} a + w_{i0}).
\end{aligned}$$

□

III. Approximation of One-dimensional Functions

Any function of bounded variation can be represented as a difference of two monotonic increasing functions [20]. Thus we can approximate any continuous function of bounded variation with a neural network using the result presented in the previous section. Let g be a continuous function of bounded variation. Function g can be represented as difference of two monotone increasing functions ψ^+ and ψ^-

$$g(x) \; = \; \psi^+(x) - \psi^-(x) \qquad (4)$$

using Jordan decomposition[2] [20]:

$$\begin{aligned}\psi^+(x) &= \frac{1}{2}\left(T_g[a,x]+g(x)\right)\\ \psi^-(x) &= \frac{1}{2}\left(T_g[a,x]-g(x)\right)\end{aligned} \tag{5}$$

where $T_g[a,b]$ is the total variation of function g on interval $[a,b]$:

$$T_g[a,b] \;=\; \sup\sum_{i=0}^{k-1}|f(\alpha_{i+1})-f(\alpha_i)|, \quad a=\alpha_0<\alpha_1<\cdots<\alpha_k=b.$$

To ensure that ψ^+ and ψ^- are strictly increasing we can modify the above by the following assignment:

$$\begin{aligned}\psi^+(x) &= \frac{1}{2}\left(T_g[a,x]+g(x)+x\right),\\ \psi^-(x) &= \frac{1}{2}\left(T_g[a,x]-g(x)+x\right).\end{aligned} \tag{6}$$

Strictly increasing functions ψ^+ and ψ^- can be approximated by networks $\aleph^+_{N^+}$ and $\aleph^-_{N^-}$, respectively. The network approximating function g is given by

$$\aleph_N(x) \;=\; \aleph^+_{N^+}(x)-\aleph^-_{N^-}(x), \tag{7}$$

where $N=N^+ + N^-$.

Lemma 3: Let $g\colon [a,b]\to\mathbf{R}$ be of bounded variation. For every natural number N let $\aleph_N\colon [a,b]\to\mathbf{R}$ be defined by the formula (7). Then $\aleph_N$ is in the form of formula (1):

$$\aleph_N(x)=c_0+\sum_{i=1}^{N}c_i\phi(w_{i1}x+w_{i0})$$

where $\phi(x)=\tanh x$ and coefficients are chosen from those for $\aleph^+_{N^+}$ and $\aleph^-_{N^-}$. Moreover

$$\sup_{x\in[a,b]}|\aleph_N(x)-g(x)| \;<\; \frac{6}{N}\,T_g[a,b]. \tag{8}$$

□

[2]If ψ is continuously differentiable then the decomposition can be easier described by the formulas: $\psi^+(x) = g(a) + \int_a^x \max\{0,g'(t)\}\,dt$ and $\psi^-(x)=\int_a^x \min\{0,-g'(t)\}\,dt$.

IV. Approximation of Multi-dimensional Functions

Multi-dimensional functions are first decomposed into one-dimensional functions using plain waves. One-dimensional functions are approximated using results from the preceding sections.

A. Plain Wave Decomposition

Let function $f\colon \mathbf{R}^n \mapsto \mathbf{R}$ be infinitely differentiable rapidly decreasing.[3] Let $\check{f}$ denote the Radon transform of function f [21]:

$$\check{f}(\omega, t) = \int_{\omega\cdot x\, =\, t} f(x)\, \xi \tag{9}$$

where ω is a unit vector in $\mathbf{R}^n$, $\omega \cdot x$ is a scalar product, and ξ is a differential form used to integrate f over all hyperplanes $\omega \cdot x \ = \ t$.

The Radon transform inversion formula

$$f(x) = \int_{\Phi} b(\omega \cdot x, \omega)\, \xi(\omega) \tag{10}$$

is an integration of plain waves, $b(\omega \cdot x, \omega)$, over any closed surface Φ enclosing the origin of space $\mathbf{R}^n$. A plain wave can be computed as follows [21]:

$$b(t, \omega) \ = \ \begin{cases} \frac{(-1)^{\frac{1}{2}(n-1)}}{2(2\pi)^{n-1}} \left(\frac{\partial}{\partial t}\right)^{n-1} \check{f}(\omega, t) & \text{for odd } n, \\ \frac{(-1)^{\frac{1}{2}n}(n-1)!}{(2\pi)^n} \int_{\Phi} \int_{-\infty}^{+\infty} \left[\check{f}(\omega, t)\{t - \omega \cdot x\}^{-n} dt\right] \xi(\omega) & \text{for even } n, \end{cases} \tag{11}$$

where the integral over t is understood in terms of its regularization, and

$$\xi(\omega) = \sum_{k=1}^{n} (-1)^{k-1} \omega_k d\omega_1 \cdots d\omega_{k-1} d\omega_{k+1} \cdots d\omega_n.$$

In the context of the inverse Radon transform the plain waves $b(\omega \cdot x, \omega)$ are traditionally called the back projection data. We use integral representation (10) as a basis for reduction of dimensionality, utilizing the fact that $b(\omega \cdot x, \omega)$ is a one-dimensional function for any fixed ω.

Assume some suitable partitioning of the integration surface

$$\Phi = \{\Phi_l\}_{l=1}^{L}.$$

[3]Function f defined on $\mathbf{R}^n$ is rapidly decreasing when $\lim_{x\to\infty} \left| x_1^{k_1} \cdots x_n^{k_n} \right| f(x) \ = \ 0$ for any nonnegative k_i's.

Now, approximate integral (10) by the sum:

$$f(x) \;\approx\; \bar{f}(x) \;=\; \sum_{l=1}^{L} \mu(\Phi_l)\, b(\omega^{(l)} \cdot x, \omega^{(l)}). \tag{12}$$

where

$$\mu(\Phi_l) \;=\; \int_{\Phi_l} \xi(\omega).$$

The upper bound of the approximation error of (12) is given by the following theorem. This theorem is a correction of the result presented in [18].

Theorem 4: Let $f\colon \mathbf{R}^n \mapsto \mathbf{R}$ be infinitely differentiable rapidly decreasing. Let $\check{f}$ be the Radon transform of f and $b(t,\omega)$ be the corresponding back-projection data. Set the number of partitions to $L = k^{n-1}$ where k is a positive integer. Then the error of approximation (12) is bounded by:

$$\left| f(x) - \bar{f}(x) \right| \;\;\le\;\; \frac{\alpha}{k}, \tag{13}$$

$$\alpha \;=\; \frac{2\pi^{\frac{n+2}{2}}\sqrt{n}}{\Gamma\left(\frac{n}{2}\right)}\; \max_{t} \max_{\omega} \left|\nabla_{\omega} b(t,\omega)\right|,$$

where the gamma function Γ is given by:

$$\Gamma(x) \;=\; \int_0^{\infty} z^{x-1} e^{-z} dz.$$

□

Notice, that α is constant for fixed f, hence the error can be arbitrarily small as the number of partitions increases.

B. Multi-dimensional Representation by Neural Networks

For a fixed $\omega^{(l)} = (\omega_1^{(l)}, \ldots, \omega_n^{(l)})$, $l = 1, \ldots, k^{n-1}$, the function $b(\omega^{(l)} \cdot x, \omega^{(l)})$ is one-dimensional. To simplify notation denote $t = \omega^{(l)} \cdot x$, and $b_l(t) = b(\omega^{(l)} \cdot x, \omega^{(l)})$. Each function b_l can be approximated using decomposition discussed in Section II and Theorem 2 by appropriate $\aleph_\eta^{(l)}\colon \mathbf{R} \mapsto \mathbf{R}$. We shall assume that each of the subnetworks $\aleph_\eta^{(l)}$ has the same number of neurons in the hidden layer, η. To represent n-dimensional back-projection data mappings $b(\omega^{(l)} \cdot x, \omega^{(l)})$ there is a need to implement the inner product $t = \omega^{(l)} \cdot x$. This can be done by extending the number of inputs of the subnetwork from 1 to n, and identifying the weights w_{ij}, $i = 1, \ldots, \eta k^{n-1}$, with $w_{(i,l)j}$, $i = 1, \ldots, \eta$ and $l = 1, \ldots, k^{n-1}$, defined as follows

$$w_{(i,l)j} \doteq \omega_j^{(l)} \, w_{i1}^{(l)}, \quad j = 1, \ldots, n. \tag{14}$$

Weights c_i and w_{i0}, for $i = 1, \ldots, \eta k^{n-1}$, are identified with appropriate $c_i^{(l)}$ and $w_{i0}^{(l)}$, $i = 1, \ldots, \eta$ and $l = 1, \ldots, k^{n-1}$, and remain unchanged. The mapping of network $\mathcal{N}_N \colon \mathbf{R}^n \mapsto \mathbf{R}$, the final approximated representation of function $f \colon \mathbf{R}^n \mapsto \mathbf{R}$, is the superposition of the mappings $\aleph_\eta^{(l)}$

$$\mathcal{N}_N(x) = \sum_{l=1}^{k^{n-1}} \mu(\Phi_l) \, \aleph_\eta^{(l)}(x) = c_0 + \sum_{i=1}^{N} \left(c_i \, \phi \left(w_{i0} + \sum_{j=1}^{n} w_{ij} x_j \right) \right), \tag{15}$$

where $c_0 = \sum_{l=1}^{k^{n-1}} \mu(\Phi_l) \, c_0^{(l)}$.

Theorem 5: Let $f \colon \mathbf{R}^n \mapsto \mathbf{R}$ be infinitely differentiable rapidly decreasing. Let $\mathcal{N}_N$ be a single hidden layer network, given by (1), with $N = \eta k^{n-1}$ neurons in the hidden layer, $\psi(x) = \tanh(x)$. Then there exists an approximated representation of function f by network $\mathcal{N}_N$ such that

$$|f(x) - \mathcal{N}_N(x)| \leq \frac{\alpha}{k} + \frac{\beta}{\eta}$$

where

$$\alpha = \frac{2\pi^{\frac{n+2}{2}} \sqrt{n}}{\Gamma(\frac{n}{2})} \max_t \max_\omega |\nabla_\omega b(t, \omega)|,$$

$$\beta = \frac{12\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}\right)} \max_l T_{b_l}[K_l],$$

and K_l is the respective domain of function $b_l(t)$. □

Observe that for a given function f parameters α and β are constant. Theorem 5 specifies the upper bound of the multi-dimensional approximation error using single hidden-layer network. On the other hand, it can be used to find the number of hidden layer neurons needed to guarantee the required approximation error.

Corollary 6: The upper bound of the approximation error of infinitely differentiable rapidly decreasing function $f \colon \mathbf{R}^n \mapsto \mathbf{R}$ by a single hidden layer network using sigmoidal (hyperbolic tangent) function neurons is proportional to the inverse of the n^{th}–root of the number of neurons in the hidden layer:

$$\varepsilon \ \sim \ \frac{1}{\sqrt[n]{N}}$$

In other words function f can be realized with accuracy ε by a single hidden-layer network using $N = O_\varepsilon(1/\varepsilon^n)$ neurons in the hidden layer. □

V. EXAMPLE

This section shows an example of the network synthesis method proposed in the paper. The method performs three steps:

STEP 1: Find approximation of function f by a finite number of plain waves $b(t, \omega^{(l)})$ using formulas (9), (11), and (12).

STEP 2: Decompose one-dimensional functions $b_l(t)$, associated with plain waves $b(t, \omega^{(l)})$, into monotonic functions $b_l^+(t)$ and $b_l^-(t)$ using formula (6). Then find subnetworks approximating each of these functions using Theorem 2.

STEP 3: Find final approximation network by combining subnetworks found in STEP 2 using formulas (14) and (15).

Let us consider the following two-dimensional function $f\colon \mathbf{R}^2 \mapsto \mathbf{R}$

$$\begin{aligned} f(x_1, x_2) \quad = \quad & 3\,(1 - x_1)^2 \, \exp[-x_1^2 - (x_2 + 1)^2] \\ & - 10\,(\frac{1}{5}x_1 - x_1^3 - x_2^5) \, \exp[-x_1^2 - x_2^2] \\ & - \frac{1}{3} \, \exp[-(x_1 + 1)^2 - x_1^2], \end{aligned} \tag{16}$$

A plot of the function f is shown in Figure 4. A two-dimensional example is chosen so that all of the steps of the method can be graphically illustrated. The method was numerically implemented using MATLAB.

STEP 1. Function f is approximated by a finite number of plain waves, using formula (12). Let as assume initially that the number of plain waves used is $L = 6$. Figures 5 through 7 show some of the plain waves $b(t, \omega)|_{t=x\cdot\omega}$ obtained from f. Note, that in $\mathbf{R}^2$ a unit vector can be

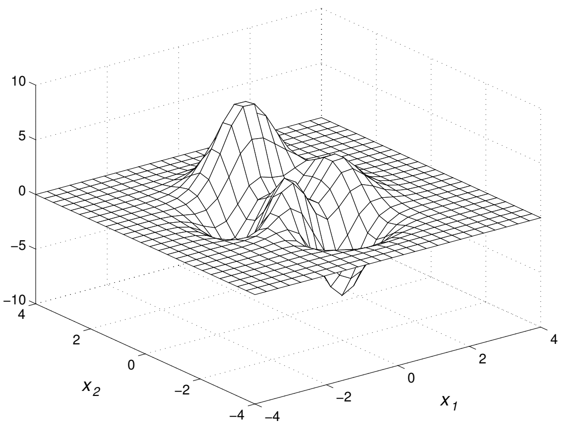

Fig. 4. Function $f(x_1, x_2)$.

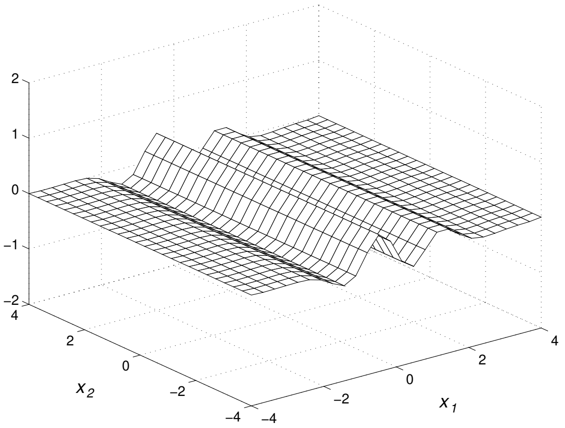

Fig. 5. Plain wave $b(x_1, x_2)|_{\theta_1=0^\circ}$.

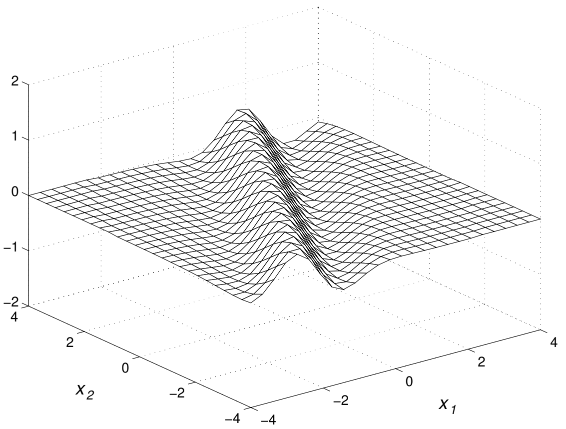

Fig. 6. Plain wave $b(x_1, x_2)|_{\theta_1=-30^\circ}$.

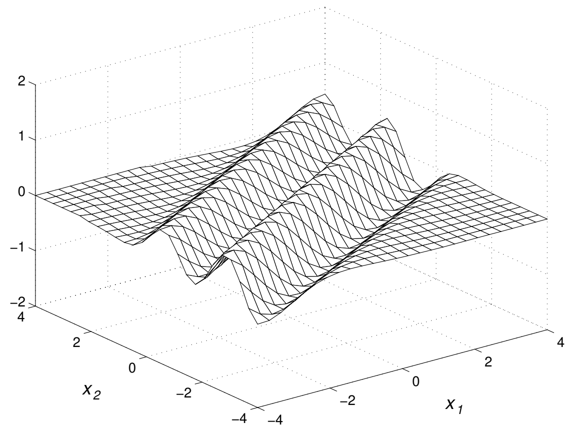

Fig. 7. Plain wave $b(x_1, x_2)|_{\theta_1=-60^\circ}$.

parameterized by a single parameter θ, which is the angle of rotation of the unit vector. We have:

$$\omega = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}.$$

Each of the plain waves is represented by a one-dimensional function $b_l(t)$ and a unit vector $\omega^{(l)}$ or, equivalently, by a rotation angle θ. For $L = 6$

$$\theta \in \{0^o, 30^o, 60^o, 90^o, 120^o, 150^o\}.$$

Figures 8 and 9 show approximation of function $f(x_1, x_2)$ using 6 and 12 plain waves, respectively.

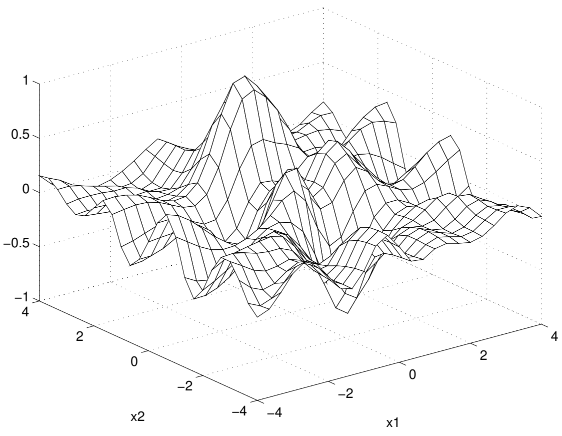

Fig. 8. Representation with 6 plain waves.

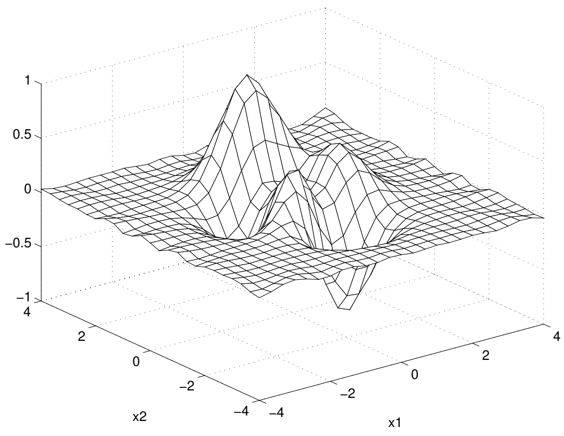

Fig. 9. Representation with 12 plain waves.

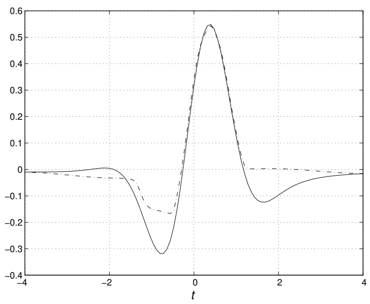

Fig. 10. $b_l(t)|_{\theta_1=-30^o}$.

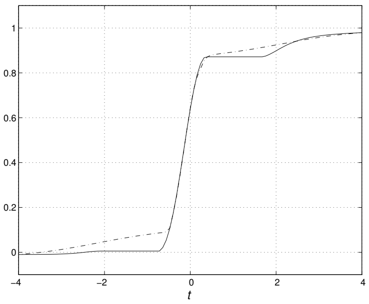

Fig. 11. $b_{l+}(t)|_{\theta_1=-30^o}$.

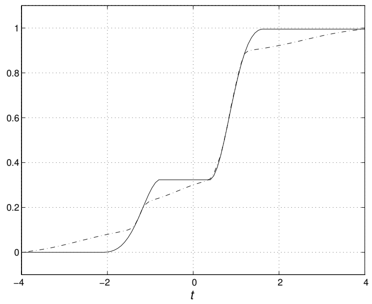

Fig. 12. $b_{l-}(t)|_{\theta_1=-30^o}$.

STEP 2. Each of the one-dimensional functions $b_l(t)$ is decomposed into two monotonic increasing functions, $b_l^+(t)$ and $b_l^-(t)$, using formula (5). Then, these monotonic functions are approximated by single hidden-layer networks, as is illustrated in Figures 10 through 12. Solid lines represent functions $b_l(t)$, $b_l^+(t)$ and $b_l^-(t)$, respectively. Dotted lines represent their approximation with neural networks based on Theorem 2. In this example nine neurons in the hidden layer are used. Figure 13 compares the upper bound of the approximation error obtained from formula (8) with the actual approximation error for different numbers of neurons in the hidden layer of the approximating network. Note, that the scale on the vertical axis is logarithmic.

STEP 3. The networks approximating functions $b_l(t)$ are combined together, using formulas (14) and (15), to form the final approximation network. Examples of approximation of the function (16) with $L = 12, N = 9$ and $L = 12, N = 18$ are shown in Figures 14 and 15, respectively.

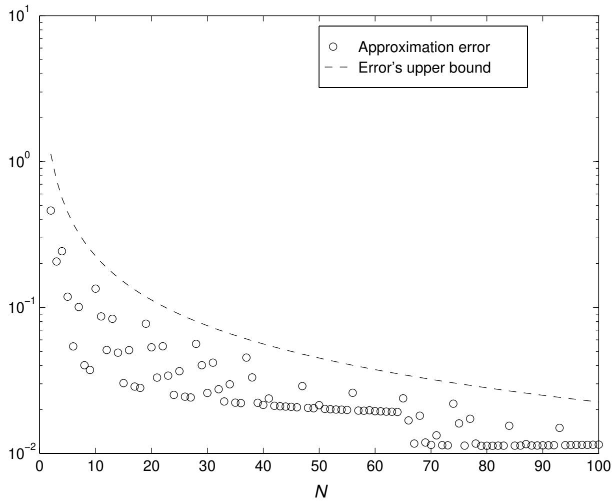

Fig. 13. Approximation error and its upper bound.

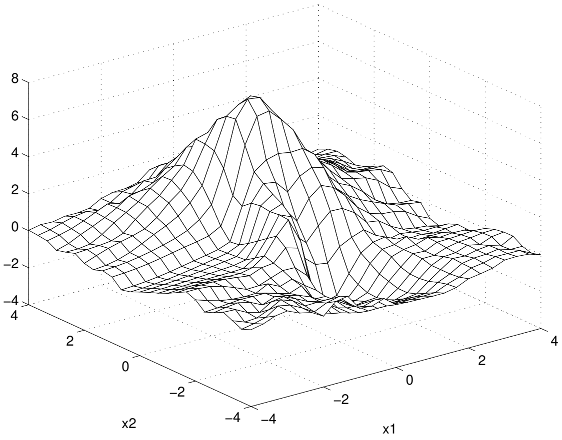

Fig. 14. Approximation with 12 plain waves and 9 neurons.

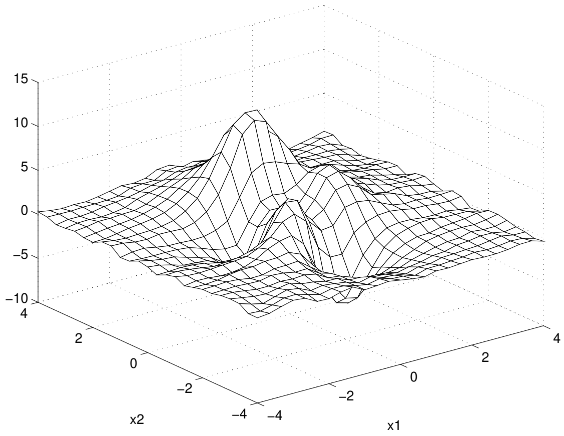

Fig. 15. Approximation with 12 plain waves and 18 neurons.

VI. Remarks

Although the emphasis of the paper is on giving analytical solution of the problem of neural network synthesis assuming a perfect information about the function being approximation, we would like to comment on what happens when this is not the case. We addressed assumptions on the function, computational error, and complexity of the method. It should be stress that we just touch here the tip of the iceberg and that more precise treatment of these issues calls for a separate study.

A. *Information about a function and computational error*

In general, to apply our method for estimating function $f\colon \mathbf{R}^n \mapsto \mathbf{R}^1$ on a bounded subset B of $\mathbf{R}^n$ we need to assume that our function is rapidly decreasing[4] infinitely many times differentiable on some open subset U containing the closure of B. This requirement comes from the use of Radon transform for reduction of the multi-dimensional case to one dimensional problem. This assumption serves well the current discussion, however, it can be relaxed in more general treatments of the Radon transform [21], [17], [22].

Consider case when our original function f is one dimensional. It can be equal to a function $g\colon \mathbf{R}^1 \mapsto \mathbf{R}^1$. This reduces our requirements on f essentially, since from g we require only that it is continuous and of bounded variation (see Section III). The situation becomes especially easy when the number $M = \sup_{x,y\in B,\ x\neq y} \left|\frac{g(y)-g(x)}{y-x}\right|$ is finite, for example, when g has a continuous derivative. In this case for any $M < m < \infty$ the representation $g(x) = \psi^+(x) - \psi^-(x)$ of g as a difference of two strictly monotone increasing functions can be achieved just by putting $\psi^+(x) = \frac{1}{2}(mx + g(x))$ and $\psi^-(x) = \frac{1}{2}(mx - g(x))$. Note, that an error ε of an estimation of M has no impact on this representation as long as $M + \varepsilon < m$. It is also interesting to realize that our estimation of a strictly increasing function $\psi\colon [a,b] \mapsto \mathbf{R}$ obtained this way (i.e., ψ equals ψ^+ or ψ^-) has very clear computational complexity and does not depend much on error of calculating function ψ. More precisely, in order to find reasonable estimations of γ_i's (see Section IIB) used in calculating coefficients in our representation we need to calculate the values of ψ only at points $a+j\delta$ for $j = 0,\ldots,k$, where k is the smallest number for which $\frac{m(b-a)}{k} < \Delta$, and $\delta = \frac{b-a}{k}$.[5] Thus the number of computations needed for finding an approximation of ψ is of order: constant (≤ 10) times $\frac{m(b-a)}{\Delta}$. Finally note that if the values of ψ are calculated within an $\frac{\Delta}{N}$ error, this can increase the error of the $\aleph$ estimation of ψ just by $\frac{\Delta}{N}$. (This is the case since our

[4] In practice, we do not need assume the "rapidly decreasing" part, since we estimate our function only on the bounded set B. More precisely, instead of estimating f we can always apply our method to the function $f_0 = f \cdot g$, where $g\colon \mathbf{R}^n \mapsto \mathbf{R}^1$ is infinitely many times differentiable, is equal 1 on B, and is equal 0 on the complement of U. Then f_0 is equal to f on B, while it satisfies the "rapidly decreasing" requirement.

[5] Estimate γ_i by the first $a + j\delta$ for which $\left|\psi(a+j\delta) - \left(\psi(a) + i\frac{\Delta}{N}\right)\right| < \frac{\Delta}{N}$. This will increase the error estimate in Theorem 2 from $\frac{3\Delta}{N}$ to $\frac{4\Delta}{N}$.

method for finding the parameters is non-iterative; and in calculating the values of $\aleph_N(x)$ the method essentially relies only to on the information to which interval $[\gamma_{i-1}, \gamma_i]$ point x belongs.)

B. Complexity of the method

Let assume that computation of a plain wave b_l from a function f has complexity O_{b_l}, and the complexity evaluation of function b_l for a given x is negligible. Then complexity of one-dimensional approximation, O_{1D} is proportional to the number of neurons in the hidden layer η

$$O_{1D} = O(\eta).$$

Complexity of multi-dimensional approximation is proportional to complexity O_{b_l}, number of one dimensional approximations k^{n-1} and complexity of each of these approximations $O(\eta)$

$$O_{nD} = \eta \cdot k^{n-1} \cdot O_{b_l} = N \cdot O_{b_l},$$

where N is the total number of neurons in a hidden layer of a neural network. Thus complexity of approximating multi-dimensional function f is directly proportional to the complexity of computing a plain wave b_l. Applications of Radon transform are gaining in popularity in recent years. Initially there were restricted to subjects related to computational medical imaging. Recently the transform is used in many new fields like optics and holographic interferometry, geophysics, radio astronomy, and pure mathematics. A good discussion of relatively recent advances in Radon transform and its applications can be found in [23], [22]. In general, computation of plain waves and inverse Radon transform is a challenging problem. However, there have been a significant progress in addressing practical issues of inverse Radon transform computations, for instance wavelet transform based methods introduced in [24], [25].

VII. Conclusions

The goal of this paper was twofold. First, it was to determine the upper bound of the approximation error for single hidden-layer feedforward networks using sigmoidal transfer function neurons. Second, it was to develop the method for synthesis of networks to approximate functions of several variables.

A constructive proof have been presented that the upper bound of the maximum approximation error using sigmoidal (hyperbolic tangent) function neurons is proportional to $1/\sqrt[n]{N}$, where n is dimension of the space and N is the number of neurons in the hidden layer. The proof of the error's upper bound was used to introduce a new method of network synthesis.

The presented method for network synthesis consists of three main parts. In part one, the dimensionality of the approximation problem is reduced by applying the inverse Radon transform. In the second part, one-dimensional problems are approximately represented by single hidden-layer subnetworks. Then these subnetworks are combined into the final approximation network. The method fully determines the network topology, including the neural weights.

REFERENCES

[1] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[2] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[3] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of unknown mapping and its derivatives using multilayer feedforward networks," *Neural Networks*, vol. 3, pp. 551–560, 1990.

[4] Y. Ito, "Approximation of continuous functions on $\mathbf{R}^d$ by linear combination of of shifted rotations of sigmoid function with and without scaling," *Neural Networks*, vol. 5, pp. 105–115, 1992.

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

[6] E. K. Blum and L. K. Li, "Approximation theory and feedforward networks," *Neural Networks*, vol. 4, pp. 511–515, 1991.

[7] P. Korain, "On the complexity of approximating mappings using neural networks," *Neural Networks*, vol. 6, pp. 649–653, 1993.

[8] A. Bulsari, "Some analytical solutions to the general approximation problem for feedforward neural networks," *Neural Networks*, vol. 6, pp. 991–996, 1993.

[9] V. Kůrková, "Kolmogorov's theorem and multilayer neural networks," *Neural Networks*, vol. 5, pp. 501–506, 1992.

[10] D. A. Sprecher, "A universal mapping for Kolmogorov's superposition theorem," *Neural networks*, vol. 6, pp. 1089–1094, 1993.

[11] D. A. Sprecher, "A numerical implementation of Kolmogorov's superpositions," *Neural Networks*, vol. 9, no. 5, pp. 765–772, 1996.

[12] L. K. Jones, "A simple lemma on greedy approximation in hilbert space and convergence rates for projection pursuit regression and neural network training.," *Annals of Statistics*, vol. 20, no. 1, pp. 608–613, 1992.

[13] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 930–945, May 1993.

[14] F. Girosi and G. Anzellotti, "Convergence rates of approximation by translates," Tech. Rep. A.I. Memo No. 1288, MIT, Artificial Intelligence Laboratory, April 1995.

[15] N. Murata, "An integral representation of functions using three-layered networks and their approximation bounds," *Neural Networks*, vol. 9, no. 6, pp. 947–956, 1996.

[16] M. Meltser, M. Shoham, and L. M. Manevitz, "Approximating functions by neural networks: A constructive solution in the uniform norm," *Neural Networks*, vol. 9, no. 6, pp. 965–978, 1996.

[17] S. Helgason, *The Radon Transform*, Brikhäuser, 1980.

[18] S. M. Carroll and B. W. Dickinson, "Construction of neural nets using the Radon transform," in *'89 IJCNN Proceedings*, 1989, vol. I, pp. 607–611.

[19] Y. Ito, "Representation of functions by superposition of a step or sigmoidal functions and their applications to neural network theory," *Neural Networks*, vol. 4, pp. 385–394, 1991.

[20] E. J. McShane, *Integration*, Princeton University Press, Princeton, 1944.

[21] I. M. Gel'fand, M. I. Graev, and N. Y. Vilenkin, *Generalized Functions*, vol. 5, Academic Press, 1966.

[22] S. R. Deans, "Radon and Abel transforms," in *The Transforms and Applications Handbook*, A. D. Poularikas, Ed., pp. 631–717. CRC Press, 1996.

[23] S. R. Deans, *The Radon Transform and Some of Its Applications*, Krieger Publishing Company, Malabar, Florida, 1993.

[24] G. Kaiser and R. F. Streater, "Windowed Radon transform, analytic signals, and thw wave function," in *Wavelets: A Tutorial in Theory and Applications*, C. K. Chui, Ed., pp. 399–441. Academic Press, San Diego, CA, 1992.

[25] M. Holschneider, "Inverse Radon transform through inverse wevelet transform," *Inverse Problems*, vol. 7, pp. 853–861, 1991.

APPENDIX

Proof of Proposition 1: The proof is identical to that of Theorem 2 if you define $d_N = 1$. This also makes respective weights the same. □

Proof of Theorem 2: Note that the choice of c_0 guarantees that $\aleph_N(a) = \psi(a)$ and the choice of w_{jk}'s is made to ensure that $L_i(x) = w_{i1}x + w_{i0}$ maps $[\gamma_{i-1}, \gamma_i]$ onto $[-d_N, d_N]$, that is that $F_i(x) = \phi(L_i(x)) = \phi(w_{i1}x + w_{i0})$ maps $[\gamma_{i-1}, \gamma_i]$ onto $\left[-1 + \frac{1}{2N}, 1 - \frac{1}{2N}\right]$.

To see the estimation fix N, $0 < j \leq N$, and $x \in [\gamma_{j-1}, \gamma_j]$ and note that

$$\begin{aligned}
\aleph_N(x) &= \psi(a) + \sum_{i=1}^{N} \frac{\Delta}{2N}[\phi(w_{i1}x + w_{i0}) - \phi(w_{i1}a + w_{i0})] \\
&= \psi(a) + \frac{\Delta}{2N}\sum_{i=1}^{N}[F_i(x) - F_i(a)] \\
&= \psi(a) + \frac{\Delta}{2N}\left(\sum_{i=1}^{j-1}[F_i(x) - F_i(a)] + [F_j(x) - F_j(a)] + \sum_{i=j+1}^{N}[F_i(x) - F_i(a)]\right)
\end{aligned}$$

Now, for $i > j$ we have $x \leq \gamma_j \leq \gamma_{i-1}$ and so

$$0 < F_i(x) - F_i(a) < F_i(\gamma_{i-1}) - (-1) = \left(-1 + \frac{1}{2N}\right) + 1 = \frac{1}{2N}.$$

Similarly for $i < j$ we have $x \geq \gamma_{j-1} \geq \gamma_i$ and so

$$2 > F_i(x) - F_i(a) > F_i(\gamma_i) - F_i(\gamma_{i-1}) = 2 - \frac{1}{N}.$$

Thus,

$$\begin{aligned}
\left|\aleph_N(x) - \psi(a) - \Delta\frac{2(j-1)}{2N}\right| &\leq \frac{\Delta}{2N}\left(\sum_{i=1}^{j-1} |[F_i(x) - F_i(0)] - 2| + |F_j(x) - F_j(0)|\right. \\
&\qquad \left. + \sum_{i=j+1}^{N} |F_i(x) - F_i(0)|\right) \\
&\leq \frac{\Delta}{2N}\left(\frac{j-1}{N} + 2 + \frac{N-(j+1)}{N}\right) \\
&\leq \frac{3\Delta}{2N} \leq \frac{2\Delta}{N}
\end{aligned}$$

Since we also have

$$\left|\psi(x) - \psi(a) - \Delta\frac{2(j-1)}{2N}\right| = |\psi(x) - \psi(\gamma_{j-1})| \leq \frac{\Delta}{N}$$

we conclude that indeed $|\aleph_N(x) - \psi(x)| \leq \frac{\Delta}{N} + \frac{2\Delta}{N} = \frac{3\Delta}{N}$ finishing the argument. □

Proof of Lemma 3: Using Eq. (4), (6), Theorem 2, and assuming $N^+ = N^-$ (N^+ and N^- are number of neurons in approximating networks, $N^+ + N^- = N$) we have:

$$\begin{aligned}
\sup_{x\in[a,b]} |\aleph_N(x) - g(x)| \;&\le\; \sup_{x\in[a,b]} \left|\aleph_N^+(x) - \psi^+(x)\right| + \sup_{x\in[a,b]} \left|\aleph_N^-(x) - \psi^-(x)\right| \\
&<\; 3\,\frac{\psi^+(b) - \psi^+(a)}{N^+} + 3\,\frac{\psi^-(b) - \psi^-(a)}{N^-} \\
&=\; \frac{6}{N}\frac{1}{2}\,(T_g[a,b] + g(b) - T_g[a,a] - g(a) \\
&\qquad + T_g[a,b] - g(b) - T_g[a,a] + g(a)]) \\
&=\; \frac{6}{N}\,T_g[a,b].
\end{aligned}$$

□

Proof of Theorem 4: Assume fixed x. Let Φ be the unit sphere and assume uniform partitioning

$$\begin{aligned}
\left|f(x) - \bar{f}(x)\right| &= \left|\sum_{l=1}^{L}\int_{\Phi_l} b(\omega\cdot x,\omega)\,\xi(\omega) - \sum_{l=1}^{L}\mu(\Phi_l) b(\omega^{(l)}\cdot x,\omega^{(l)})\right| \\
&\le \sum_{l=1}^{L}\int_{\Phi_l}\left|b(\omega\cdot x,\omega) - b(\omega^{(l)}\cdot x,\omega^{(l)})\right|\,\xi(\omega) \\
&\le \sum_{l=1}^{L}\,\mu(\Phi_l)\,\max_{\omega\in\Phi_l} d_\mu(\omega,\omega^{(l)})\,\max_{\omega\in\Phi_l}|\nabla_\omega b(\omega\cdot x,\omega)| \\
&\le \mu(\Phi)\,\max_{l=1\ldots L}\max_{\omega\in\Phi_l} d_\mu(\omega,\omega^{(l)})\,\max_{\omega}|\nabla_\omega b(\omega\cdot x,\omega)|\,,
\end{aligned}$$

where $d_\mu(\omega,\omega^{(l)})$ is a distance between points ω and $\omega^{(l)}$ on the unit sphere. Since the partitioning is uniform we have

$$\max_{l=1\ldots L}\max_{\omega\in\Phi_l} d_\mu(\omega,\omega^{(l)}) \;=\; \frac{\pi\sqrt{n}}{k}.$$

The surface of the unit sphere is

$$\Omega_n \;=\; \mu(\Phi) \;=\; \frac{2\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2})}.$$

Hence

$$\left|f(x) - \bar{f}(x)\right| \;\le\; \frac{2}{k}\,\frac{\pi^{\frac{n+2}{2}}\sqrt{n}}{\Gamma(\frac{n}{2})}\,\max_{\omega}|\nabla_\omega b(\omega\cdot x,\omega)|\,.$$

Maximizing over x we obtain the upper bound of approximation error:

$$\begin{aligned}\left|f(x)-\bar{f}(x)\right| &\leq \frac{2}{k}\,\frac{\pi^{\frac{n+2}{2}}\sqrt{n}}{\Gamma(\frac{n}{2})}\,\max_x\max_\omega\left|\nabla_\omega b(\omega\cdot x,\omega)\right| \\ &= \frac{2}{k}\,\frac{\pi^{\frac{n+2}{2}}\sqrt{n}}{\Gamma(\frac{n}{2})}\,\max_t\max_\omega\left|\nabla_\omega b(t,\omega)\right|.\end{aligned}$$

□

Proof of Theorem 5: Form Lemma 3 we have:

$$|b_l(t)\ -\ \aleph^{(l)}(t)|\ <\ \frac{6}{N_l}T_{b_l}[K_l].$$

The error of approximation of function $\bar{f}$ by network $\mathcal{N}$, formula (12), is the sum of approximation errors of functions b_l by networks $\aleph^{(l)}$. Assume that each network $\aleph^{(l)}$ has the same number of neurons $N_l = \eta$, hence

$$|\bar{f}(x)\ -\ \mathcal{N}_N(x)|\ \leq\ k^{n-1}\,\mu(\Phi_l)\,\frac{6}{\eta}\,\max_l\,T_{b_l}[K_l]\ =\ \frac{12}{\eta}\,\frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}\right)}\,\max_l\,T_{b_l}[K_l]. \tag{17}$$

Thus from inequalities (13) and (17)

$$|f(x)\ -\ \mathcal{N}_N(x)|\ \leq\ \frac{2}{k}\,\frac{\pi^{\frac{n+2}{2}}\sqrt{n}}{\Gamma(\frac{n}{2})}\,\max_t\max_\omega\left|\nabla_\omega b(t,\omega)\right|\ +\ \frac{12}{\eta}\,\frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}\right)}\,\max_l\,T_{b_l}[K_l].$$

□

Proof of Corollary 6: From Theorem 5 we have that approximation's error upper bound is

$$\varepsilon\ =\ \alpha\,\frac{1}{k}+\beta\,\frac{1}{\eta}, \tag{18}$$

Let us relax conditions on k and η and assume that they can take any positive value, not only integer. Further assume that number of units in the hidden layer, $N = k^{n-1}\eta$, is constant and then find minimum value of ε.

Substitute $\eta = \frac{N}{k^{n-1}}$ into (18)

$$\varepsilon\ =\ \alpha\,\frac{1}{k}+\beta\,\frac{k^{n-1}}{N}, \tag{19}$$

Differentiate ε in respect to k

$$\frac{d\varepsilon}{dk} \;=\; -\alpha\,\frac{1}{k^2} + \beta\,\frac{(n-1)\,k^{n-2}}{N}$$

$$\frac{d\varepsilon}{dk} \;=\; 0 \quad\Longrightarrow\quad k \;=\; \sqrt[n]{\frac{\alpha N}{\beta(n-1)}}. \tag{20}$$

Check if it is a minium of function $\varepsilon(k)$

$$\frac{d^2\varepsilon}{dk^2} \;=\; 2\,\alpha\,\frac{1}{k^3} + \beta\,\frac{(n-1)(n-2)\,k^{n-3}}{N}. \tag{21}$$

Formula (21) is positive for any positive value of k, hence we have the minimum. Now, substitute (20) into (19)

$$\varepsilon \;=\; \left[\alpha^{\frac{n-1}{n}}(\beta(n-1))^{\frac{1}{n}} + \beta(n-1)\left(\frac{\alpha}{\beta(n-1)}\right)^{\frac{n-1}{n}}\right]\frac{1}{\sqrt[n]{N}}. \tag{22}$$

Thus, we have proportional relation

$$\varepsilon \;\sim\; \frac{1}{\sqrt[n]{N}}$$

since other components of formula (22) are constant for given function f. And

$$\varepsilon \;\sim\; \frac{1}{\sqrt[n]{N}} \quad\Longrightarrow\quad N \;\sim\; \frac{1}{\varepsilon^n}.$$

□