Efficient Estimation of Neural Weights by Polynomial Approximation

Gunter Ritter

***Abstract*—It has been known for some years that the uniform-density problem for forward neural networks has a positive answer: Any real-valued, continuous function on a compact subset of $\boldsymbol{R}^d$ can be uniformly approximated by a sigmoidal neural network with one hidden layer. We design here algorithms for *efficient* uniform approximation by a certain class of neural networks with one hidden layer which we call *nearly exponential*. This class contains, e.g., all networks with the activation functions $1/(1+e^{-t})$, $\tanh(t)$, or $e^t \wedge 1$ in their hidden layers. The algorithms flow from a theorem stating that such networks attain the order of approximation $O(N^{-1/d})$, d being dimension and N the number of hidden neurons. This theorem, in turn, is a consequence of a close relationship between neural networks of nearly exponential type and multivariate algebraic and exponential polynomials. The algorithms need neither a starting point nor learning parameters; they do not get stuck in local minima, and the gain in execution time relative to the backpropagation algorithm is enormous. The size of the hidden layer can be bounded analytically as a function of the precision required.**

***Index Terms*— Approximation algorithms, complexity problem for neural networks, nearly exponential activation function, neural network with one hidden layer, order of uniform approximation.**

I. Introduction

A. Background

Artificial forward neural networks are nonlinear parametric expressions representing multivariate numerical functions. In connection with such paradigms there arise mainly three problems: a *density* problem, a *complexity* problem, and an *algorithmic* problem. The *density problem* deals with the following question: which functions can be approximated and, in particular, can all members of a certain class of functions be approximated in a suitable sense. This problem was satisfactorily solved in the late 1980's [7], [15], and [16]. Any continuous function on any compact subset of $\boldsymbol{R}^d$ can be uniformly approximated arbitrarily closely by a neural network with one hidden layer. Moreover, the proof given in [16] provides an intimate connection between forward neural networks and polynomials. This observation suggested to consider the remaining two problems, too, from this point of view.

In fact, more recently the *complexity problem* which deals with the relationship between the size of an expression (i.e., the number of neurons) and its approximation capacity was solved [32], [26], [25] for two classes of activation functions, both including the popular logistic function and none of them the Heaviside step function. The constructions in these papers provide answers of Jackson's type to the question: How closely can a *continuous* or m-times *differentiable* function be *uniformly* approximated by a network with one hidden layer of a given size? or, in other words, how many hidden units are sufficient to approximate such a function uniformly with an error not exceeding ε? The order of approximation of a parametric functional expression describes the asymptotic behavior of the maximum approximation error with respect to a given class of functions as the number of parameters in the expression tends to infinity. It was shown that neural networks based on the logistic and many other activation functions approximate any m-times differentiable function with a Lipschitz-continuous mth derivative of the order $O(N^{-(m+1)/d})$, d being the size of the input layer and N the number of hidden neurons; cf. Comment III-C a).

It remains to exploit the efficiency promised in the solution of the complexity problem. The proofs given in [32], [26], and [25] are based on polynomial approximation. Moreover, they are constructive so that algorithms for approximation can be derived from them. It is the main purpose of this paper to turn the method of proof introduced in [32] into algorithms for direct and effective uniform approximation. The most popular method of neural-network training today is backpropagation, i.e., gradient search in combination with the chain rule of multivariate differential calculus. However, this method is slow since it proceeds *iteratively* although some progress has been made in the past; cf. [4], [19], [36], [1], and [30]. Moreover, it may get stuck in a local minimum, cf. [24]. The algorithms proposed here are *noniterative*, thereby avoiding these disadvantages. Furthermore, they need no starting point and no learning parameters. We note finally that Chen [5] designed another efficient, noniterative algorithm but for a different, more complex network architecture.

B. Outline

The organization of the paper is as follows. In Section II, we derive Proposition 2.2, which is a result of Jackson's type for approximation of continuous functions on the hypercube $[0,1]^d$ by real exponential polynomials. Proposition 2.2 is derived from a similar theorem for algebraic polynomials, cf. Feinerman and Newman [12] and Soardi [35]. This theorem

Manuscript received April 17, 1997; revised January 10, 1999.
The author is with the Fakultät für Mathematik und Informatik, Universität Passau, 94030 Passau, Germany.
Communicated by P. Moulin, Associate Editor for Nonparametric Estimation, Classification, and Neural Networks.
Publisher Item Identifier S 0018-9448(99)04736-7.

is based on the isotropic modulus of continuity, which is the appropriate measure of *uniform* approximability. Section III is devoted to neural networks of nearly exponential type [32]. A neural activation function $\sigma : \boldsymbol{R} \to \boldsymbol{R}$ is called here *nearly exponential* if, after suitable affine rescalings of abscissa and ordinate, σ approximates the exponential function arbitrarily closely on the negative half line. This class of functions contains the functions $1/(1+e^{-t})$, $\tanh(t)$, and $e^t \wedge 1$. A neural network is of *nearly exponential type* if all hidden neurons carry nearly exponential activation functions. The close and simple relationship between nearly exponential neural networks with one hidden layer and (multivariate) algebraic and exponential polynomials combined with Proposition 2.2 leads to a theorem on the order of approximation of a neural network of nearly exponential type, Theorem 3.1. In all cases, we also trace the constants in the estimates.

As a corollary one immediately obtains estimates of the number of hidden units necessary for uniform approximation in terms of the admissible error, see Corollary 3.2. In Comment III-C b), we discuss the size of the weights. Finally, in Section IV, we derive two algorithms for efficient approximation of continuous functions or finitely many scattered data by neural networks of nearly exponential type. The order of approximation of these algorithms is the same as that of the method of polynomial approximation used.

C. Notation

We use the following notation. The symbols $\boldsymbol{N}$, $\boldsymbol{R}$, and $\boldsymbol{R}_+$ stand for the sets of nonnegative integers, real numbers, and nonnegative real numbers, respectively. For $\rho, \tau \in \boldsymbol{R}$, $\lceil \rho \rceil$ (respectively, $\lfloor \rho \rfloor$) denotes the least (respectively, largest) integer $n \ge \rho$ (respectively, $n \le \rho$), and $\rho \vee \tau$ (respectively, $\rho \wedge \tau$) denotes the maximum (respectively, minimum) of ρ and τ. Vectors in $\boldsymbol{R}^d$ are denoted by lower case, boldface letters. The symbol Λ stands for a finite subset of $\boldsymbol{R}_+^d$. All operations on vectors are meant coordinatewise. Thus if

$$\boldsymbol{x} = (x_1, \cdots, x_d) \in \boldsymbol{R}^d, \quad \boldsymbol{y} = (y_1, \cdots, y_d) \in \boldsymbol{R}^d$$

then, e.g.,

$$e^{\boldsymbol{x}} := (e^{x_1}, \cdots, e^{x_d}), \quad \boldsymbol{x}^{\boldsymbol{y}} := (x_1^{y_1}, \cdots, x_d^{y_d})$$

if $x_k > 0$ for all k, and $\boldsymbol{x}\boldsymbol{y} = (x_1y_1, \cdots, x_dy_d)$. However, a dot denotes the inner product, $\boldsymbol{x}\cdot\boldsymbol{y} = \sum_k x_k y_k$. The Euclidean norm on $\boldsymbol{R}^d$ is denoted by $\|\cdot\|$, and $\|\cdot\|_\infty$ designates the supremum norm with metric d_∞ on a space of bounded, real- or complex-valued functions.

II. Uniform Approximation of Continuous Functions by Real Exponential Polynomials

A. Explanation

Jackson's [18] well-known theorems estimate the uniform distance of a continuous function f on the interval $[0,1]$ to the space $\mathcal{P}_n([0,1])$ of polynomial functions of degree $\le n$

$$p(t) = \sum_{\lambda=0}^{n} a_\lambda t^\lambda$$

on this interval. These estimates are in terms of n and a measure of smoothness of f—e.g., its modulus of continuity or the sizes of its derivatives. Multivariate extensions of this theorem are due to Feinerman and Newman [12], Nikolskii [29], Soardi [35], and refinements to Ditzian and Totik [10] who defined new moduli of continuity. Feinerman and Newman [12, corollaries on p. 102], also give constants involved in the estimates.

Let f be a real-valued function defined on a convex subset $\mathcal{C} \subseteq \boldsymbol{R}^d$ and let $\delta > 0$. We will use the *isotropic modulus of continuity* of f, $\omega(f,\delta)$, defined by

$$\omega(f,\delta) := \sup_{\|\boldsymbol{y}-\boldsymbol{x}\| \le \delta} |f(\boldsymbol{y}) - f(\boldsymbol{x})|. \tag{1}$$

This is the modulus of continuity introduced in [12, p. 87], and in [35, p. 67]. By convexity of the domain of f and as a consequence of the triangle inequality, we have for any real number $\rho \ge 0$

$$\omega(f,\rho\delta) \le \lceil \rho \rceil \omega(f,\delta) \le (\rho+1)\omega(f,\delta). \tag{2}$$

The second inequality is strict if f is not constant.

The function f is called Lipschitz continuous if there exists a constant L such that $|f(\boldsymbol{y}) - f(\boldsymbol{x})| \le L\|\boldsymbol{y}-\boldsymbol{x}\|$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$. We then define

$$L(f) := \sup_{\boldsymbol{x} \ne \boldsymbol{y}} \frac{|f(\boldsymbol{y}) - f(\boldsymbol{x})|}{\|\boldsymbol{y}-\boldsymbol{x}\|}.$$

If f is Lipschitz continuous, then $\omega(f,\delta) \le L(f)\delta$. We denote the space of real, d-variate algebraic polynomials of *maximal* degree n by $\mathcal{P}_n(d)$. The following theorem can be derived from Feinerman and Newman [12] or Soardi [35].

Theorem 2.1: For any continuous function $f : [0,1]^d \to \boldsymbol{R}$ and all $n \in \boldsymbol{N}$, we have

$$d_\infty(f, \mathcal{P}_n(d)) \le c(d)\,\omega\left(f, \frac{1}{n+2}\right) \tag{3}$$

for some constant $c(d)$ depending on dimension d, only. □

There are two reasons for the denominator $n+2$ instead of n in (3). First, it leads to a nicer estimate in Corollary 3.2. Second, it can be shown, using tensor products of Korovkin's kernels [20, p. 75], that a possible constant in (3) is

$$c(d) = \frac{1}{2} + \frac{\pi^2}{4}\sqrt{d}.$$

If f is Lipschitz continuous then

$$d_\infty(f, \mathcal{P}_n(d)) \le \frac{\pi^2}{4}\sqrt{d}\,\frac{L(f)}{n+2}$$

cf. Ritter [32]. In the one-dimensional case, $d = 1$, a result of Korneicuk [21] implies the estimate

$$d_\infty(f, \mathcal{P}_n(1)) \le \omega\left(f, \frac{\pi}{2n+2}\right)$$

cf. Cheney [6, pp. 144 and 147].

B. Exponential Sums and Polynomials

A (real) *exponential sum* over a finite subset $\Lambda \subseteq \boldsymbol{R}_+^d$ is an expression of the form

$$\sum_{\boldsymbol{\lambda} \in \Lambda} a_{\boldsymbol{\lambda}} e_{\boldsymbol{\lambda}}. \tag{4}$$

In this context, the elements of Λ are called *multi-intensities*, and $e_{\boldsymbol{\lambda}}$ is the multivariate exponential function on $\boldsymbol{R}^d$ defined by the expression

$$e_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \prod_{k=1}^{d} e^{-\lambda_k x_k} = e^{-\boldsymbol{\lambda} \cdot \boldsymbol{x}} \tag{5}$$

$\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_d) \in \Lambda$, $\boldsymbol{x} = (x_1, \cdots, x_d) \in \boldsymbol{R}^d$. When the index set Λ in (4) is a subset of a lattice $\alpha \boldsymbol{N}^d$ for some $\alpha > 0$ we will speak of a (real) exponential *polynomial* over Λ. An exponential polynomial of *maximal degree* $n \in \boldsymbol{N}$ is of the form

$$\sum_{\boldsymbol{\lambda} \in \alpha(0 \cdots n)^d} a_{\boldsymbol{\lambda}} e^{-\boldsymbol{\lambda} \cdot \boldsymbol{x}}$$

for some $\alpha > 0$. The symbol $\mathcal{P}_n^E(d)$ stands for the set of all real, d-variate exponential polynomials of maximal degree n and arbitrary α.

We transfer Theorem 2.1 to uniform approximation by exponential polynomials. It is well known that the situation differs from the algebraic case: Whereas $\mathcal{P}_n(d)$ is a finite-dimensional vector space and there exists a best approximating polynomial in this space, $\mathcal{P}_n^E(d)$ is neither a vector space nor closed and an optimal polynomial does not exist in general. It is sufficient to consider the sequence $k(1 - e^{-x/k}) \in \mathcal{P}_1^E(1)$; it converges to the identity function as $k \to \infty$ uniformly for x in any bounded subset of the real line.

Proposition 2.2: For any continuous function $f : [0,1]^d \to \boldsymbol{R}$ and all $n \in \boldsymbol{N}$, we have

$$d_\infty\left(f, \mathcal{P}_n^E(d)\right) \le c(d)\omega\left(f, \frac{1}{n+2}\right)$$

with the constant $c(d)$ appearing in Theorem 2.1.

Proof: First use Theorem 2.1 in order to approximate the function f by an algebraic polynomial

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{\lambda} \in (0 \cdots n)^d} a_{\boldsymbol{\lambda}} \prod_{k=1}^{d} x_k^{\lambda_k}$$

with an error

$$\|p - f\|_\infty \le c(d)\omega\left(f, \frac{1}{n+2}\right) + \varepsilon. \tag{6}$$

Given $\alpha > 0$, define the univariate function

$$\eta_\alpha(x) := \frac{1 - e^{-\alpha x}}{1 - e^{-\alpha}}, \qquad x \in [0,1].$$

Its d-dimensional extension, $\boldsymbol{x} \to \eta_\alpha(\boldsymbol{x})$, is a topological isomorphism of $[0,1]^d$ and the family $(\eta_\alpha)_{\alpha > 0}$ converges to the identity function η_0 uniformly on $[0,1]^d$ as $\alpha \to 0$. It is, therefore, possible to choose α so small that

$$\|p \circ \eta_\alpha - p \circ \eta_0\|_\infty \le \varepsilon. \tag{7}$$

Note that $p \circ \eta_\alpha$ is an exponential polynomial in $\mathcal{P}_n^E(d)$. Using (6) and (7), we obtain the estimates

$$\begin{aligned} \|p \circ \eta_\alpha - f\|_\infty &\le \|p \circ \eta_\alpha - p \circ \eta_0\|_\infty + \|p \circ \eta_0 - f\|_\infty \\ &\le c(d)\omega(f, \frac{1}{n+2}) + 2\varepsilon. \end{aligned}$$

Since $\varepsilon > 0$ is arbitrary the proposition follows. □

III. Uniform Approximation of Continuous Functions by Neural Networks with One Hidden Layer

A. Neural Networks with One Hidden Layer

The functionality of a neural network with one hidden layer is represented by an expression of the form

$$\sum_{u \in U} a_u \sigma_u(-\boldsymbol{\lambda}_u \cdot \boldsymbol{x} + b_u). \tag{8}$$

Here, $\boldsymbol{x}$ is a variable taking values in $\boldsymbol{R}^d$ (the input variable), U is the index set of *units* (*neurons*) in the *hidden layer*, $\boldsymbol{\lambda}_u \in \boldsymbol{R}^d$ comprises all *input weights* of the hidden unit $u \in U$, b_u is its *offset*, σ_u its *activation function*, and a_u its *output weight*. The dot indicates the inner product. The activation function $\sigma_u : \boldsymbol{R} \to \boldsymbol{R}$ is a univariate nonlinearity, usually of *sigmoidal* form, i.e., bounded and increasing. A graphical representation of (8) is shown in Fig. 1.

For simplicity, we assume that all hidden units in (8) carry the same nearly exponential activation function σ, i.e., $\sigma_u = \sigma$ for all $u \in U$. A neural network of the form

$$\sum_{u \in U} a_u \sigma(-\boldsymbol{\lambda}_u \cdot \boldsymbol{x} + b_u). \tag{9}$$

will be called of *type* σ.

By the exponential sigmoidal function we mean the function $\sigma^E = \exp \wedge 1$. Any exponential sum $p = \sum_{\boldsymbol{\lambda} \in \Lambda} a_{\boldsymbol{\lambda}} e_{\boldsymbol{\lambda}}$, restricted to the positive hyperquadrant $\boldsymbol{R}_+^d$, is itself a neural network of type σ^E, namely,

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{\lambda} \in \Lambda} a_{\boldsymbol{\lambda}} \sigma^E(-\boldsymbol{\lambda} \cdot \boldsymbol{x}). \tag{10}$$

Looking at a neural network from this point of view, the weights and offsets have a neat interpretation: The input weights correspond to the intensities of an exponential sum, the output weights and offsets to its coefficients. However, the close relationship between neural networks and exponential sums is not restricted to the exponential sigmoidal activation function but can be extended. This is the purpose of the following definition.

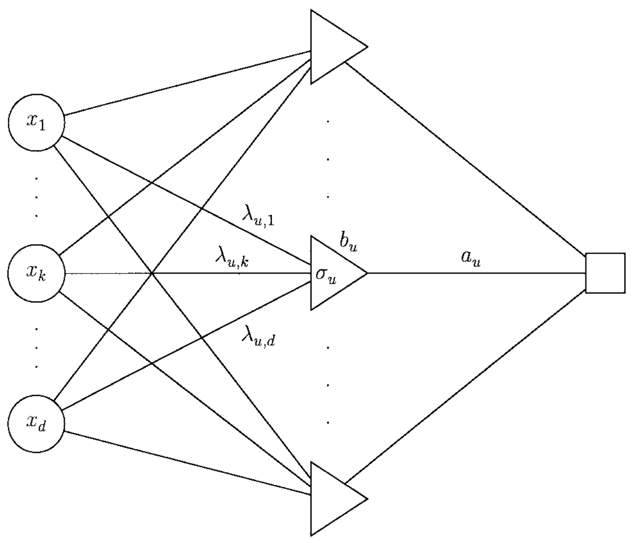


Fig. 1. Graphical representation of a neural network with one hidden layer.

B. Nearly Exponential Functions

We call a function $\sigma : \boldsymbol{R} \to \boldsymbol{R}$ *nearly exponential* if, for all $\varepsilon > 0$, there exist real numbers γ, ζ, β, ρ such that

$$|\gamma\sigma(\zeta t + \beta) + \rho - e^t| \le \varepsilon$$

for all $t \le 0$. This means that, after suitable linear rescalings and shiftings of abscissa and ordinate, the function σ approximates the exponential function arbitrarily closely on the negative half line. In the sequel, we restrict ourselves to nearly exponential activation functions and speak of a neural network of *nearly exponential type*.

Both the logistic sigmoidal function $\sigma^L(t) = 1/(1+e^{-t})$ and the hyperbolic tangent are nearly exponential. To see this in the former case it is sufficient to put $\zeta = 1$, $\rho = 0$, $\gamma = 1/\sigma^L(\beta)$, and to observe that

$$\left|\frac{\sigma^L(t+\beta)}{\sigma^L(\beta)} - e^t\right| = \frac{e^{t+\beta}}{e^{t+\beta}+1}|1-e^t| \le e^{\beta}|e^t - e^{2t}|$$

converges to 0 uniformly in $t \le 0$ as $\beta \to -\infty$. The latter function is just $\tanh t = 2\sigma^L(2t) - 1$. Of course, the exponential sigmoidal function σ^E is nearly exponential and Heaviside's step function $1_{\boldsymbol{R}_+}$ is not.

Given some activation function $\sigma : \boldsymbol{R} \to \boldsymbol{R}$, $\mathcal{R}_n^\sigma(d)$ will denote the set of all sums of the form

$$\sum_{\boldsymbol{\lambda}\in\Lambda} \pm a\sigma(-\boldsymbol{\lambda}\cdot\boldsymbol{x} + b_{\boldsymbol{\lambda}}) \tag{11}$$

with $\Lambda \subseteq \alpha(0\cdots n)^d$ for some $\alpha > 0$ and with $a > 0$ independent of $\boldsymbol{\lambda}$. Note that (10) may be put in the form

$$p(\boldsymbol{x}) = \sum_{a_{\boldsymbol{\lambda}}\ne 0} a\,\mathrm{sign}(a_{\boldsymbol{\lambda}})\sigma^E(-\boldsymbol{\lambda}\cdot\boldsymbol{x} + b_{\boldsymbol{\lambda}}) \tag{12}$$

$a = \max_{\boldsymbol{\lambda}\in\Lambda}|a_{\boldsymbol{\lambda}}|$, $b_{\boldsymbol{\lambda}} = \ln(|a_{\boldsymbol{\lambda}}|/a) \le 0$, a neural network with output weights $\pm a$; in particular, if $\Lambda \subseteq \alpha(0\cdots n)^d$ then (10) can be turned into a member of $\mathcal{R}_n^{\sigma^E}(d)$ for some n.

The following theorem approximates continuous functions f of d variables by neural networks of nearly exponential type. Its proof, being based on Proposition 2.2, exploits the intimate relationship between neural networks of nearly exponential type and exponential sums, cf.(10), and provides deeper insight into the properties of neural networks. The function f will be defined on a compact, convex subset $\mathcal{C} \subseteq [0,1]^d$, and we wish to control the error of uniform approximation by its isotropic modulus of continuity. In order to be able to apply Proposition 2.2, we have to extend f to a hypercube contained in $\boldsymbol{R}^d$ without violating its modulus of continuity. This is the reason why we assume $\mathcal{C}$ to be *convex*. We abbreviate

$$c_d(f,n) := c(d)\omega\left(f, \frac{1}{n+2}\right). \tag{13}$$

If f is Lipschitz continuous with constant $L(f)$ then

$$c_d(f,n) \le c(d)\frac{L(f)}{n+2}.$$

Theorem 3.1: Let σ be nearly exponential, let $\mathcal{C} \subseteq [0,1]^d$ be compact and convex, and let $n \in \boldsymbol{N}$. For any continuous function $f : \mathcal{C} \to \boldsymbol{R}$, we have $d_\infty(f, \mathcal{R}_n^\sigma(d)) \le c_d(f,n)$.

Proof: Let π denote the Euclidean projection $[0,1]^d \to \mathcal{C}$. The function $f \circ \pi$ is a continuous extension of f to the unit hypercube $[0,1]^d$ with the same modulus of continuity as that of f. Proposition 2.2 produces an exponential polynomial

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{\lambda}\in\alpha(0\cdots n)^d} a_{\boldsymbol{\lambda}} e^{-\boldsymbol{\lambda}\cdot\boldsymbol{x}} \tag{14}$$

such that

$$\|p_{|\mathcal{C}} - f\|_\infty \le \|p_{|[0,1]^d} - f\circ\pi\|_\infty \le c_d(f,n) + \varepsilon.$$

Let

$$\Lambda = \{\boldsymbol{\lambda} \in \alpha(0\cdots n)^d / a_{\boldsymbol{\lambda}} \ne 0\}.$$

Another representation of p is

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{\lambda}\in\Lambda} \pm a e^{-\boldsymbol{\lambda}\cdot\boldsymbol{x} + b_{\boldsymbol{\lambda}}}$$

with $b_{\boldsymbol{\lambda}} \le 0$ and $a > 0$ independent of $\boldsymbol{\lambda}$; cf. (12).

Now, since σ is nearly exponential, we may approximate the exponential function e^t by an expression $\gamma\sigma(\zeta t + \beta) + \rho$

uniformly on the negative half line up to the error $\varepsilon/(a\#\Lambda)$. The sum

$$\sum_{\boldsymbol{\lambda}\in\Lambda\setminus\{\mathbf{0}\}} \pm a\gamma\sigma(-\zeta\boldsymbol{\lambda}\cdot\boldsymbol{x}+\zeta b_{\boldsymbol{\lambda}}+\beta) + \left(\pm a\gamma\sigma(\zeta b_{\mathbf{0}}+\beta)+\rho\sum_{\boldsymbol{\lambda}\in\Lambda}\pm a\right)$$

belongs to $\mathcal{R}_n^\sigma(d)$ and its uniform distance to f does not exceed the value $c_d(f,n)+2\varepsilon$. (Here, we have assumed $0\in\Lambda$; the contrary case is similar.) This proves the theorem. □

The following corollary of Theorem 3.1 determines a number of hidden neurons sufficient for uniform approximation of a continuous function f up to an error ε. It says in particular that $O(\varepsilon^{-d})$ neurons are sufficient if f is Lipschitz continuous.

Corollary 3.2: Let σ be nearly exponential and let $\mathcal{C}\subseteq[0,1]^d$ be compact and convex.

a) For any continuous function $f:\mathcal{C}\to\boldsymbol{R}$ and all $\varepsilon>0$ there is a neural network (11) with at most $\min\{(n+1)^d/c_d(f,n)<\varepsilon\}$ hidden neurons uniformly approximating f up to an error not exceeding ε.

b) f is Lipschitz continuous then

$$\max\left\{\left\lfloor c(d)\frac{L(f)}{\varepsilon}\right\rfloor^d,1\right\}$$

neurons suffice.

Proof:

a) If n is such that $c_d(f,n)<\varepsilon$ then, according to Theorem 3.1, there exists a neural network $g\in\mathcal{R}_n^\sigma(d)$ such that $d_\infty(g,f)<\varepsilon$. This neural network contains at most $(n+1)^d$ hidden neurons.

b) Let us distinguish two cases. If $c(d)L(f)/\varepsilon<1$ then

$$c_d(f,n)\le c(d)\frac{L(f)}{n+2}<\frac{\varepsilon}{n+2}<\varepsilon$$

for all $n\ge 0$. Hence, the minimum in a) is 1 in this case. If $c(d)L(f)/\varepsilon\ge 1$ we put $n=\lfloor c(d)L(f)/\varepsilon\rfloor-1$. We have $n\ge 0$ and $n+2>c(d)L(f)/\varepsilon$, hence

$$c_d(f,n)\le c(d)\frac{L(f)}{n+2}<\varepsilon.$$

Thus the minimum in a) is at most $(n+1)^d=\lfloor c(d)L(f)/\varepsilon\rfloor^d$ in this case. □

C. Comments

a) Since a neural network of class $\mathcal{R}_n^\sigma(d)$ has at most $(n+1)^d$ hidden neurons, Theorem 3.1 says, among other things, that the optimal order of approximation of any Lipschitz-continuous function with respect to $\mathcal{R}_n^\sigma(d)$ is at least $O(N^{-1/d})$, N being the number of hidden neurons. The exponent and constant involved in this asymptotic statement depend on dimension d. If f is m times differentiable with a Lipschitz-continuous mth derivative $(m\ge 0)$ then similar arguments, again based on Jackson's theorems, show that the order of approximation is at least $O(N^{-(m+1)/d})$; cf. Soardi [35, p. 94], concerning approximation by polynomials. The approximation operators underlying the present methods are continuous. Since a network of N hidden neurons has $(d+2)N$ real-valued parameters, a theorem due to DeVore, Howard, and Micchelli [9] implies that no *continuous* (linear or nonlinear) approximation operator can improve these asymptotic estimates. (For a discussion of the classical *linear* case, cf. also Schumaker [33, p. 96], and Feinerman and Newman [12, pp. 97, 84, 85].) However, it seems to be unknown whether there exist (discontinuous) approximation operators with a better rate of approximation.

Barron [2] proves a rate-of-approximation result related to Theorem 3.1. He uses an integral representation by means of indicator functions of half spaces in order to show that the order of uniform approximation of any function $f:[-1,1]^d\to\boldsymbol{R}$ with Fourier representation $\tilde{f}$

$$f(\boldsymbol{x})=\int_{\boldsymbol{R}^d} e^{i\boldsymbol{x}\cdot\boldsymbol{y}}\tilde{f}(\boldsymbol{y})\,d\boldsymbol{y}$$

such that $\|\boldsymbol{y}\|\tilde{f}(\boldsymbol{y})$ is integrable is at least $O(N^{-1/2})$. This describes neatly a class of functions for which a dimension-independent approximation order holds (the *constants* involved in the estimates depend on dimension). However, the interpretation of this result and its comparison to the present result need some care. Barron's class is described in terms of a *global* property of its Fourier transforms. The largest space defined by *differentiability* properties and contained in it is $\mathcal{C}^{\lceil(d+3)/2\rceil}$, a space depending on dimension d. As stated above, the order of uniform approximation of a function $f\in\mathcal{C}^{\lceil d/2-1\rceil}$ is also at least $O(N^{-1/2})$.

Barron [2] shows also that the order of *mean-square* approximation (with respect to any probability measure with bounded support) of a member of the said class is at least $O(N^{-1/2})$; cf. also [3], where a geometric argument in Hilbert space is used. An extension to spaces of p-times integrable functions, $1\le p<\infty$, as well as a treatment of incremental approximants appear in Darken *et al.* [8]. These theorems are particularly useful in the context of statistical classification problems where the inputs come as independent realizations of a distribution. Mhaskar and Micchelli [28, Theorem 2.4], show that this describes again the optimal order attainable by any *continuous* mean-square approximation operator on Barron's class. A result related to [3] for activation functions with bounded "supports" but Lipschitz-continuous target functions appears in [17, Theorem 3].

b) There is a tradeoff between size of hidden layer and sizes of weights. The approach taken in Proposition 2.2 and Theorem 3.1 is chosen so as to achieve a neural network with a small number of neurons. The weights a_u in (8) may, however, be very large and have alternating signs, possibly making this network numerically unstable. Even if the coefficients $a_{\boldsymbol{\lambda}}$ of the polynomial p in the proof of Proposition 2.2 are small or of moderate size, insertion of the function η_α into this polynomial increases the output weights if a very small α has to be chosen for close approximation or if the degree of p is large.

The problem of large weights is intrinsic. It is plain that approximation of a large function needs a neural network with large weights if a small number of neurons is to be used. But there are small functions that also need large weights. The simple example of the identity function on the interval $[-1, 1]$ suffices to expose this. It can be approximated by a neural network with one hidden neuron of type $\tanh$ to arbitrary precision. However, approximation with uniform error δ needs an output weight of the order of at least $\delta^{-1/2}$, as an elementary computation shows. This is the price for reducing the number of neurons. In contrast, Mhaskar and Micchelli [27, Theorem 5.1 and Lemma 5.3(b)] show that $O(\varepsilon^{-(2d+2)})$ summands of size $O(1)$ can approximate a Lipschitz-continuous function with error ε (put $k = 0$ in the notation of [27]). This means that close approximation by neural networks with small weights is possible with about the square of the minimum number of neurons given in Corollary 3.2.

There is an intermediate way which is sketched as Algorithm 4.2 below; it still guarantees the optimal order of continuous approximation while just needing a larger constant in (13) because of the appearance of the logarithm in 4.2 i). This method does not need small values of α (and hence a small denominator in η_α) for precise approximation.

c) It is striking that the method presented here uses the convex part of the sigmoidal shape of σ, only.

IV. Algorithmic Implications

The method of proof leading to Theorem 3.1 and its corollary is constructive. It can be combined with algorithms for approximation or interpolation by algebraic polynomials for efficient, noniterative estimation of weights and offsets of neural networks. We consider algorithmic solutions of two related tasks, viz.,

a) approximation of a continuous function on a hyperrectangle in $\boldsymbol{R}^d$ specified by some functional expression by a sigmoidal neural network, and
b) approximation of finitely many data $(\boldsymbol{x}_i, y_i) \in \boldsymbol{R}^d \times \boldsymbol{R}$, $1 \le i \le l$, by a sigmoidal neural network.

The latter problem is also called *fitting a d-dimensional surface to given data* and is related to the *regression problem* of statistics. Sometimes, the points $\boldsymbol{x}_i$ lie on a regular (e.g., rectangular) grid, but the most important case is that of data points coming from irregularly scattered statistical measurements.

It is possible to reduce Task a) to Task b) by sampling data $(\boldsymbol{x}_i, y_i)$ from the functional expression given in a). This, however, means loss of information. The data points $\boldsymbol{x}_i$ may be chosen in such a way as to catch the characteristics of the given function. Reciprocally, Task b) can be reduced to solving Task a) by preliminary approximation or interpolation. Here, any method yielding a reasonable functional expression, such as triangulation, Shepard's method, radial basis functions, or other multivariate splines may be used. The interested reader is referred to Schumaker [34], Powell [31], Franke [14], or Lancaster and Šalkauskas [22].

The method developed in Sections II and III requires polynomial approximation of the function a) or of the data points b) as a first step. This is the content of Subsection A.

A. Algorithms for Polynomial Approximationand Interpolation

α) *Approximation of continuous functions:* Approximation of a continuous function f defined on a hyperrectangle in $\boldsymbol{R}^d$ by polynomials can be based on the fast Fourier transform (FFT) algorithm. After an affine substitution we may assume that the domain of definition of f is the hypercube $[-1, 1]^d$. The details are as follows.

i) Define an even, 2π-periodic function $h : \boldsymbol{R}^d \to \boldsymbol{R}$ by putting

$$h(\boldsymbol{\varphi}) := f(\cos(\boldsymbol{\varphi})).$$

ii) Choose $n \in \boldsymbol{N}$ (according to the precision required) and compute the coefficients

$$a'_{\boldsymbol{\lambda}} := \hat{h}(\boldsymbol{\lambda}) \prod_{k=1}^{d} \cos\left(\frac{\pi}{2}\frac{\lambda_k}{n+1}\right), \qquad \boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_d) \in (-n \cdots n)^d \tag{15}$$

where the hat denotes the Fourier transform

$$\hat{h}(\boldsymbol{\lambda}) = \frac{1}{(2\pi)^d}\int_{[-\pi,\pi]^d} e^{-i\boldsymbol{\lambda}\cdot\boldsymbol{\varphi}}\, h(\boldsymbol{\varphi})\, d\boldsymbol{\varphi} = \frac{1}{(2\pi)^d}\int_{[-\pi,\pi]^d} \cos(\boldsymbol{\lambda}\cdot\boldsymbol{\varphi}) h(\boldsymbol{\varphi})\, d\boldsymbol{\varphi}.$$

Computation of $\hat{h}$ is most efficiently performed by means of the FFT algorithm applied to discretely sampled values of h. The multipliers $\cos\left(\frac{\pi}{2}\frac{\lambda_k}{n+1}\right)$ in (15) improve the approximation; other multipliers such as Fejér's $1 - |\lambda_k|/(n+1)$, de la Vallée-Poussin's, or Korovkin's [20] could be used. The real coefficients $a'_{\boldsymbol{\lambda}}$ enjoy the property $a'_{(\pm\lambda_1,\ldots,\pm\lambda_d)} = a'_{\boldsymbol{\lambda}}$ for all sign combinations. Similarly, the trigonometric polynomial

$$q(\boldsymbol{\varphi}) := \sum_{\boldsymbol{\lambda}\in(-n\cdots n)^d} a'_{\boldsymbol{\lambda}} e^{i\boldsymbol{\lambda}\cdot\boldsymbol{\varphi}} = \sum_{\boldsymbol{\lambda}\in(-n\cdots n)^d} a'_{\boldsymbol{\lambda}} \prod_{k=1}^{d} e^{i\lambda_k\varphi_k} = \sum_{\boldsymbol{\lambda}\in(0\cdots n)^d} a''_{\boldsymbol{\lambda}} \prod_{k=1}^{d} \cos(\lambda_k\varphi_k) \tag{16}$$

uniformly approximates the function h and has the property $q(\pm\varphi_1, \ldots, \pm\varphi_d) = q(\boldsymbol{\varphi})$, $\boldsymbol{\varphi} = (\varphi_1, \ldots, \varphi_d)$. The coefficients $a''_{\boldsymbol{\lambda}}$ in (16) have the representation

$$a''_{\boldsymbol{\lambda}} = 2^{\eta_{\boldsymbol{\lambda}}} a'_{\boldsymbol{\lambda}}$$

where $\eta_{\boldsymbol{\lambda}}$ is the number of nonzero entries in the multi-frequency $\boldsymbol{\lambda}$.

iii) Using the expansion

$$\cos(\lambda\varphi) = \sum_{l\le\lambda/2} \binom{\lambda}{2l} (\cos^{\lambda-2l}\varphi)(1-\cos^2\varphi)^l$$

$\lambda \in \boldsymbol{N}$, $\varphi \in \boldsymbol{R}$, we obtain from (16) the representation

$$q(\boldsymbol{\varphi}) = \sum_{\boldsymbol{\lambda}\in(0\cdots n)^d} a_{\boldsymbol{\lambda}} \prod_{k=1}^{d} \cos^{\lambda_k}\varphi_k \tag{17}$$

of q as a polynomial in $\cos\varphi_k$ with certain coefficients $a_{\boldsymbol{\lambda}}$.

iv) The substitution $\varphi_k = \arccos(x_k)$ in (17) yields an algebraic polynomial $p \in \mathcal{P}_n(d)$

$$p(\boldsymbol{x}) := \sum_{\boldsymbol{\lambda}\in(0\cdots n)^d} a_{\boldsymbol{\lambda}} x_1^{\lambda_1}\cdots x_d^{\lambda_d}.$$

This procedure is optimal in the sense that p realizes the order of uniform approximation to f claimed by Jackson's [18] theorems, cf. Comment III-C a).

β) *Interpolation and approximation of data given on a rectangular grid:* If the data $(\boldsymbol{x_m}, y_{\boldsymbol{m}})$ are given on the rectangular grid $\boldsymbol{x_m} = \boldsymbol{m}/n$, $\boldsymbol{m} \in (0\cdots n)^d$, say, then there exists a unique *interpolating* multivariate polynomial L_n of maximum degree n. However, it is well known that the resulting surface may be too undulating to be acceptable, cf. [34].

This shortcoming is not observed if Bernstein polynomials are used for *approximation*. The Bernstein polynomial B_n associated with the data points $(\boldsymbol{x_m}, y_{\boldsymbol{m}})$, $\boldsymbol{m} \in (0\cdots n)^d$, is specified by the following formulas:

$$\begin{aligned} P_n(m,x) &= \binom{n}{m}(1-x)^{n-m}x^m \\ &= \sum_{l=m}^{n} (-1)^{l-m}\binom{n}{m}\binom{n-m}{l-m}x^l, \\ &\qquad 0\le m\le n, \quad 0\le x\le 1 \end{aligned} \tag{18}$$

$$\begin{aligned} \bar{P}_n(\boldsymbol{m},\boldsymbol{x}) &= \prod_{k=1}^{d} P_n(m_k, x_k), \\ \boldsymbol{m} &= (m_1,\ldots,m_d) \in (0\cdots n)^d, \\ \boldsymbol{x} &= (x_1,\ldots,x_d) \in [0,1]^d \end{aligned} \tag{19}$$

$$B_n(\boldsymbol{x}) = \sum_{\boldsymbol{m}\in(0\cdots n)^d} y_{\boldsymbol{m}} \bar{P}_n(\boldsymbol{m},\boldsymbol{x}). \tag{20}$$

In general, this method does not attain the optimal rate of continuous approximation possible for continuous and differentiable functions. The class of functions with order of approximation $O(n^{-1})$ was determined by Ditzian and Zhou [11].

Of course, all methods suitable for approximating *scattered* data may be applied to the present situation as well. However, their rate of approximation is not covered by the theory in Sections II and III. Two of them are briefly addressed next since they yield good results.

γ) *Polynomial approximation of scattered data:* There are two popular approaches to least-squares approximation of given scattered data $(\boldsymbol{x}_i, y_i) \in \boldsymbol{R}^d \times \boldsymbol{R}$, $1 \le i \le l$, by linear combinations of d-variate monomials $\mu_1,\ldots,\mu_N$.

– Construction of an *orthonormal basis* in the linear span of $\mu_1,\cdots,\mu_N$ with respect to the inner product

$$(f,g) := \sum_{i=1}^{N} f(\boldsymbol{x}_i)g(\boldsymbol{x}_i)$$

e.g., by the Gram–Schmidt procedure.

– The *normal equations* $A^TA\boldsymbol{a} = A^T\boldsymbol{y}$ specified by

$$A = (\mu_j(\boldsymbol{x}_i))_{\substack{1\le i\le l\\ 1\le j\le N}},$$

$\boldsymbol{y} = (y_i)_{1\le i\le l}$. Extensions of the method of normal equations that are also applicable in the presence of rank deficiencies and constraints are the *matrix methods*; cf. Lawson and Hanson [23], Forsythe *et al.* [13], and Lancaster and Šalkauskas [22].

These methods have the advantage over the methods β) and γ) that not *all* monomials up to a certain maximum degree have to be used.

B. Algorithms for Weight Estimation

Here are two algorithms for efficient, noniterative estimation of weights and offsets of a nearly exponential neural network for approximation of continuous functions f defined on the hypercube $[0,1]^d$ or of scattered data $(\boldsymbol{x}_i, y_i), \boldsymbol{x}_i \in [0,1]^d$. The resulting overall algorithms, including the polynomial approximation, contain numerical operations such as integration and matrix inversion and symbolic (algebraic) operations such as substitutions and expansions of trigonometric expressions. The simplest method is a least-squares technique. All procedures can be easily implemented numerically. Both algorithms conserve the order of approximation by polynomials given in Section IV-A, cf. also Theorem 3.1.

In order to gain some flexibility, we extend the homeomorphism $\boldsymbol{x} \to \sigma_\alpha(\boldsymbol{x})$ appearing in the proof of Proposition 2.2 to different α's in the d coordinate directions. Thus let

$$\eta_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \frac{\mathbf{1} - e^{-\boldsymbol{\alpha x}}}{\mathbf{1} - e^{-\boldsymbol{\alpha}}}, \qquad \boldsymbol{\alpha} \in \boldsymbol{R}_+^d,\ \boldsymbol{x} \in [0,1]^d.$$

It follows

$$\eta_{\boldsymbol{\alpha}}^{-1}(\boldsymbol{t}) = \frac{\ln(\mathbf{1} - (\mathbf{1} - e^{-\boldsymbol{\alpha}})\boldsymbol{t})}{-\boldsymbol{\alpha}}, \qquad \boldsymbol{t} \in [0,1]^d.$$

Algorithm 4.1:

i) Approximate f uniformly or approximate the given data $(\boldsymbol{x}_i, y_i)$ by an algebraic polynomial $p \in \mathcal{P}_n(d)$; cf. Section IV-A.

ii) Choose some vector $\boldsymbol{\alpha} = (\alpha_1,\cdots,\alpha_d) \in \boldsymbol{R}^d$ with $\alpha_k > 0$ so small that $\eta_{\boldsymbol{\alpha}}$ approximates the identity function well enough on the unit hypercube $[0,1]^d$. The expression $p(\eta_{\boldsymbol{\alpha}}(\boldsymbol{x}))$ is a multivariate exponential polynomial and a neural network of type σ^E approximating f on $[0,1]^d$ or the data $(\boldsymbol{x}_i, y_i)$; cf. Proposition 2.2.

iii) If necessary and suitable, a pruning method may be applied in order to reduce the network size, e.g., by canceling all units with small output weights.

TABLE I

$\sqrt{\text{MSE}}$/(Execution Time in Seconds) for Approximation of the Functions f and g Displayed in Fig. 2 by Means of the Method Indicated in the First Row. Empty Entries Are Due to Excessive Runtime Requirements (Backpropagation) or to Numerical Instabilities (Least Squares). Training Samples Are Evenly Spaced

	size N of hidden layer	number l of training samples for BP and LS	backpropagation logistic activation	Bernstein Alg. 4.1	least squares Alg. 4.1
f	11 neurons	31	0.028/15	0.158/0.01	0.024/0.03
	21 neurons	61	0.027/20	0.107/0.03	0.022/0.13
	31 neurons	91	0.030/80	0.082/0.05	0.019/0.22
g	25 neurons	169	0.067/400	0.127/0.02	0.075/0.17
	81 neurons	625	0.045/60000	0.106/0.04	0.026/4.3
	441 neurons	3721	———	0.073/0.06	———

iv) If some other nearly exponential activation function σ is to be used in the hidden layer then the exponentials $e_{\boldsymbol{\lambda}}(\boldsymbol{x}) = e^{-\boldsymbol{\lambda}\cdot\boldsymbol{x}}$ appearing in $p(\eta_{\boldsymbol{\alpha}}(\boldsymbol{x}))$ must subsequently be approximated by a suitable transform of σ; cf. Section III-B and Theorem 3.1. In the logistic case, e.g., they are replaced by the function $\sigma^L(-\boldsymbol{\lambda}\cdot\boldsymbol{x}+\beta)/\sigma^L(\beta)$ for some fixed number $\beta < 0$ depending on the precision required.

Algorithm 4.2:

i) Choose some vector $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_d) \in \boldsymbol{R}^d$, $\alpha_k > 0$, and approximate the function $\boldsymbol{t} \rightarrow f(\eta_{\boldsymbol{\alpha}}^{-1}(\boldsymbol{t}))$ uniformly for $\boldsymbol{t} \in [0,1]^d$ or approximate the transformed data $(\eta_{\boldsymbol{\alpha}}(\boldsymbol{x}_i), y_i)$ by an algebraic polynomial $p \in \mathcal{P}_n(d)$; cf. Section IV-A. The approximation improves if the vector $\boldsymbol{\alpha}$ is chosen such that the modulus of continuity

$$\omega\left(f \circ \eta_{\boldsymbol{\alpha}}^{-1}, \frac{1}{n+2}\right)$$

becomes as small as possible.

ii) The exponential polynomial and neural network of type σ^E, $p(\eta_{\boldsymbol{\alpha}}(\boldsymbol{x}))$, approximates f on $[0,1]^d$ or the data $(\boldsymbol{x}_i, y_i)$; cf. also Comment III-C b).

Steps iii) and iv) are the same as in Algorithm 4.1.

The following procedure can sometimes improve the effectiveness of Algorithm 4.2. Suppose the target function f has small modulus of continuity except near one corner of the hypercube. The transformation $x \rightarrow 1-x$ applied to some of the coordinates maps f into a function $\bar{f}$ with small modulus of continuity off the origin. Now apply Algorithm 4.2 to $\bar{f}$. The modulus of continuity of the function $\bar{f} \circ \eta_{\boldsymbol{\alpha}}^{-1}$ appearing in Step i) of this algorithm can be reduced by choosing large α_k's. Finally, the resulting neural network approximating $\bar{f}$ is transformed into a neural network approximating f by inverting the transformation above.

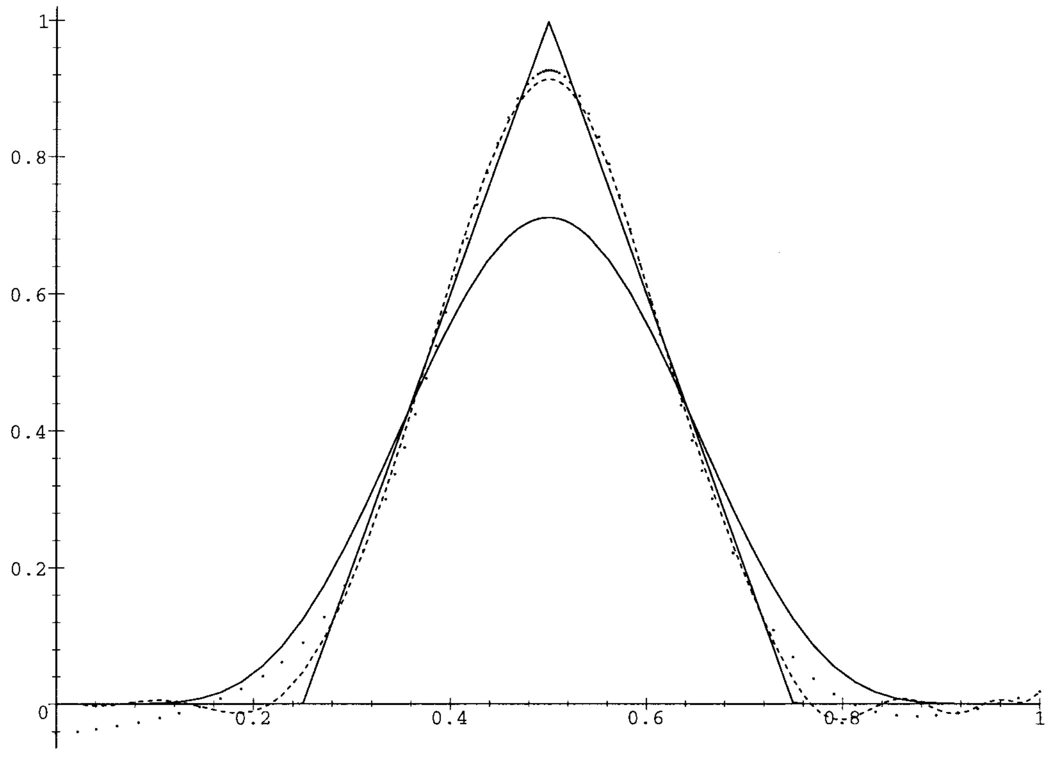
In theorems of Jackson's type for algebraic and exponential polynomials, cf. Theorem 2.1 and Proposition 2.2, the exponents and intensities are essentially fixed. In neural networks, the corresponding weights of hidden units are considered to be variable parameters. In both algorithms we do not fully exploit this flexibility, but gain some freedom by considering not only integral weights but also weights in a rectangular lattice of the form $\alpha_1(0\cdots n) \times \cdots \times \alpha_d(0\cdots n)$ with $n \in \boldsymbol{N}$ and $\alpha_k > 0$.

C. Examples

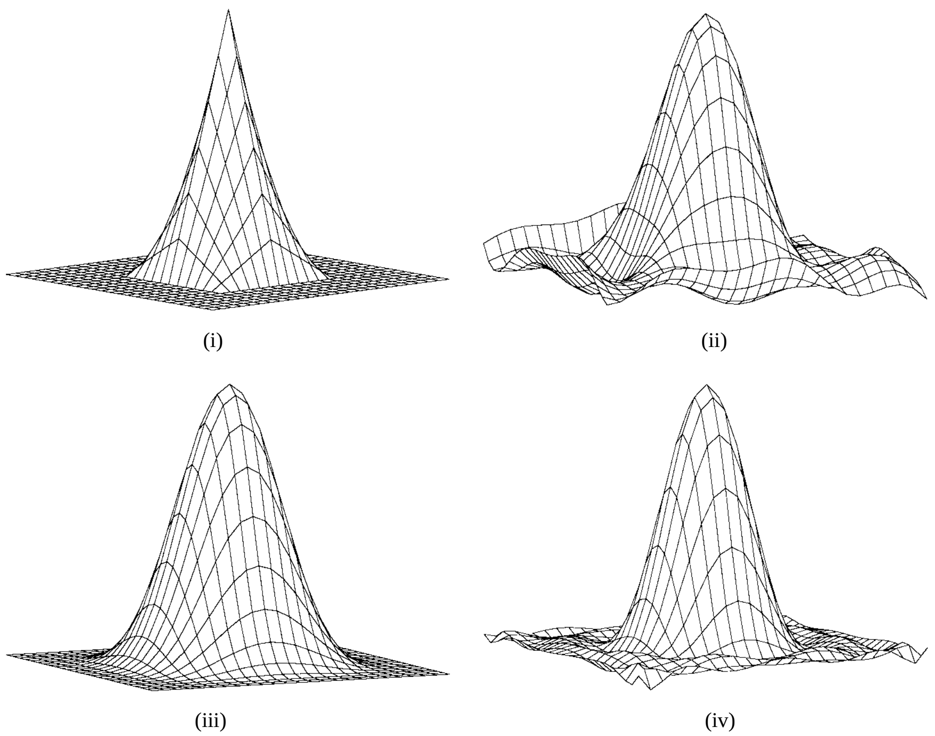
For the sake of illustration, we compare in Table I mean-square errors (MSE) and execution times for different methods applied to the univariate function f and the bivariate function g displayed in Fig. 2. The adequate error measure here would be the *maximum* error; however, since backpropagation minimizes the MSE this was chosen. The methods are

i) backpropagation,
ii) approximation by Bernstein polynomials, cf. Sections IV-A β) and IV-B,
iii) least-squares polynomial approximation, cf. Sections IV-A γ) and IV-B.

The quality of the approximation by the method of Fourier transformation, cf. Section IV-A α), is similar to that of the least-squares method; the results are not reproduced here. The results for the backpropagation algorithm are typical values obtained from the "Stuttgart Neural Network Simulator" (SNNS) and the other algorithms were implemented in the programming language C. All tests were run on a workstation SUN SPARC 10.



(a)



(b)

Fig. 2. (a) Approximations of the univariate function $f(x) = (1 - |4x - 2|)^+$, $0 \leq x \leq 1$, by networks with one hidden layer of 31 logistic neurons. Solid line: Original function f; dotted line: backpropagation, 91 evenly spaced training data, runtime 80 s; solid line: Bernstein's method, 31 evenly spaced training data, runtime 0.05 s; dashed line: least squares method, 91 evenly spaced training data, runtime 0.22 s. Algorithm 4.1 with $\alpha = 0.01$ was used for Bernstein's and least squares methods. (b) Approximations of the bivariate function $g(x, y) = (1 - |4x - 2|)^+(1 - |4y - 2|)^+$, $0 \leq x, y \leq 1$, by neural networks with one hidden layer. (i) Original function g; (ii) backpropagation, 81 logistic neurons in hidden layer, 625 evenly spaced training data, runtime 60 000 s; (iii) Bernstein's method, 441 logistic neurons in hidden layer, runtime 0.073 s; (iv) least squares method, 81 logistic neurons in hidden layer, 625 evenly spaced training data, runtime 4.3 s. Algorithm 4.1 with $\alpha = 0.01$ was used in Cases (iii) and (iv).

V. Discussion

Methods for direct, noniterative estimation of neural weights for certain activation functions have been proposed. They are based on polynomial approximation. The most efficient of them uses the linear least-squares method. Runtimes of the methods are minimal. The practitioner, used to gradient search, may be astonished by the fact that the input weights (and offsets) are essentially set in advance. However, the simulations presented in Section IV.C indicate that the results may often be better than those obtained by gradient search. Moreover, this design allows application of the theories of linear least-squares approximation and normal equations which have a long tradition in approximation theory.

If gradient search finds an optimal solution then the approximation will, of course, be better than that of the proposed methods, but we cannot easily predict its runtime. On the other hand, we cannot predict *whether* it will find an optimal solution and, if it does not, *what* we will get.

From experience, a solution found by backpropagation usually has small weights. However, gradient search may not be able to find an optimal solution with large weights. This is not a problem with the proposed methods. But this is also their weakness: they do not control the sizes of output weights, so unnecessarily large weights and numerical instabilities may arise. Their strengths are short and predictable runtimes and, as was shown in Sections II and III, predictability of accuracy from elementary properties of the target function.

Acknowledgment

The author thanks Herr Christoph Pesch for his help with the SNNS and a referee for pointing out [2] and [8] and a weakness of Algorithm 4.1 that led to the formulation of Algorithm 4.2.

References

[1] H. Adeli and S. L. Hung, “An adaptive conjugate gradient learning algorithm for efficient training of neural networks,” *Appl. Math. and Comput.*, vol. 62, pp. 81–102, 1994.
[2] A. R. Barron, “Neural net approximation,” in *Proc. 7th Yale Workshop on Adaptive and Learning Systems*, 1992, pp. 69–72.
[3] ——, “Universal approximation bounds for superpositions of a sigmoidal function,” *IEEE Trans. Inform. Theory*, vol. 39, pp. 930–945, 1993.
[4] C. Charalambous, “Conjugate gradient algorithm for efficient training of artificial neural networks,” *Proc. Inst. Elec. Eng.–Pt. G*, vol. 139, pp. 301–310, 1992.
[5] C. L. P. Chen, “A rapid supervised learning neural network for function interpolation and approximation,” *IEEE Trans. Neural Networks*, vol. 7, pp. 1220–1230, 1996.
[6] E. W. Cheney, *Introduction to Approximation Theory*, 2nd ed. New York: Chelsea, 1982.
[7] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Math. Contr. Signals Syst.*, vol. 2, pp. 303–314, 1989.
[8] Ch. Darken, M. Donahue, L. Gurvits, and E. Sontag, “Rate of approximation results motivated by robust neural network learning,” in *Proc. 6th Ann. Conf. Computational Learning Theory (COLT ’93)*, 1993, pp. 303–309.
[9] R. DeVore, R. Howard, and C. A. Micchelli, “Optimal nonlinear approximation,” *Manuscr. Math.*, vol. 63, pp. 469–478, 1989.
[10] Z. Ditzian and V. Totik, *Moduli of Smoothness*. New York, Berlin, Heidelberg, London, Paris, Tokyo: Springer, 1987.
[11] Z. Ditzian and X. Zhou, “Optimal approximation class for multivariate Bernstein operators,” *Pacific J. Math.*, vol. 158, no. 1, pp. 93–120, 1993.
[12] R. P. Feinerman and D. J. Newman, *Polynomial Approximation*. Baltimore, MD: Williams & Wilkins, 1974.
[13] G. E. Forsythe, M. A. Malcolm, and C. B. Moler, *Computer Methods for Mathematical Computations*. Englewood Cliffs, NJ: Prentice Hall, 1977.
[14] R. Franke, “Scattered data interpolation: Tests of some methods,” *Math. Comput.*, vol. 38, pp. 181–200, 1982.
[15] K.-I. Funahashi, “On the the approximate realization of continuous mappings by neural networks,” *Neural Networks*, vol. 2, pp. 183–192, 1989.
[16] K. Hornik, M. Stinchcombe, and H. White, “Multilayer-feedforward networks are universal approximators,” *Neural Networks*, vol. 2, pp. 359–366, 1989.
[17] B. Igelnik and Y.-H. Pao, “Stochastic choice of basis functions in adaptive function approximation and the functional-link net,” *IEEE Trans. Neural Networks*, vol. 6, pp. 1320–1329, 1995.
[18] D. Jackson, “On the approximation by trigonometric sums and polynomials,” *Trans. Amer. Math. Soc.*, vol. 13, pp. 491–515, 1912.
[19] N. B. Karaiyannis and A. N. Venetsanopoulos, “Efficient learning algorithms for neural networks (ELEANNE),” *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 1372–1383, 1993.
[20] P. P. Korovkin, *Linear Operators and Approximation Theory*. Delhi, India: Hind. Publ., 1960.
[21] N. P. Korneicuk, “The exact constant in D. Jackson’s theorem on best uniform approximation of continuous periodic functions,” *Sov. Math.–Dokl.*, vol. 3, pp. 1040–1041, 1962.
[22] P. Lancaster and K. Šalkauskas, *Curve and Surface Fitting*. London, Orlando, San Diego, New York, Austin, Boston, Sydney, Tokyo, Toronto: Academic, 1986.
[23] C. Lawson and R. Hanson, *Solving Least Squares Problems*. Englewood Cliffs, NJ: Prentice-Hall, 1974.
[24] J. M. McInerney, K. G. Haines, S. Biafore, and R. Hecht-Nielsen, “Back propagation error surfaces can have local minima,” in *Int. Joint Conf. Neural Networks*, Vol. II–627 (Washington DC, 1989).
[25] H. N. Mhaskar, “Neural networks for optimal approximation for smooth and analytic functions,” *Neural Comput.*, vol. 8, pp. 164–177, 1996.
[26] H. N. Mhaskar and L. Khachikyan, “Neural networks for function approximation,” in *Neural Networks for Signal processing, Proc. 1995 IEEE Workshop*, vol. V, F. Girosi, J. Makhoul, E. Manolakos, and E. Wilson, Eds., (Cambridge, MA). New York: IEEE Press, 1995, pp. 21–29.
[27] H. N. Mhaskar and C. A. Micchelli, “Approximation by superposition of sigmoidal and radial basis functions,” *Adv. Appl. Math.*, vol. 13, pp. 350–373, 1992.
[28] ——, “Dimension-independent bounds on the degree of approximation by neural networks,” *IBM J. Res. Develop.*, vol. 38, pp. 277–284, 1994.
[29] S. M. Nikol’skii, *Approximation of Functions of Several Variables and Imbedding Theorems*. Berlin, Heidelberg, New York: Springer, 1975.
[30] S. Osowski, M. Stodolski, and P. Bojarczak, “Efficient supervised learning of multilayer feedforward neural networks,” in *Int. Symp. Speech, Image Processing and Neural Networks* (Hong Kong, 1994).
[31] M. J. D. Powell, “The theory of radial basis function approximation in 1990,” in *Advances in Numerical Analysis*, W. Light Ed., *Wavelets, Subdivision Algorithms, and Radial Basis Functions*, vol. II. Oxford, U.K.: Clarendon, 1992.
[32] G. Ritter, “Jackson’s theorems and the number of hidden units in neural networks for uniform approximation,” Univ. Passau, Fak. Math. Inform., Tech. Rep. MIP–9415, 1994.
[33] L. L. Schumaker, *Spline Functions*. New York, Chichester, Brisbane, Toronto: Wiley, 1981.
[34] ——, “Fitting surfaces to scattered data,” in *Approximation Theory II*, G. G. Lorenz, C. K. Chui, and L. L. Schumaker, Eds. New York, San Francisco, London: Academic, 1976, pp. 203–268.
[35] P. M. Soardi, “Serie di Fourier in piú variabili,” *Quad. dell’Unione Mat. Italiana*, vol. 26. Bologna, Italy: Pitagora Editrice, 1984.
[36] A. Sperduti and A. Sarita, “Speed up learning and network optimization with extended back propagation,” *Neural Networks*, vol. 6, pp. 365–383, 1993.