ACADEMIC PRESS

Available at WWW.MATHEMATICSWEB.ORG POWERED BY SCIENCE DIRECT®

Journal of Approximation Theory 122 (2003) 260–266

JOURNAL OF Approximation Theory

http://www.elsevier.com/locate/jat

Approximation by neural networks with a bounded number of nodes at each level

G. Gripenberg

Institute of Mathematics, Helsinki University of Technology, P.O. Box 1100, FIN-02015 Espoo, Finland

Received 30 December 2002; accepted in revised form 10 April 2003

Communicated by Allan Pinkus

Abstract

It is shown that the general approximation property of feed-forward multilayer perceptron networks can be achieved in networks where the number of nodes in each layer is bounded, but the number of layers grows to infinity. This is the case provided the node function is twice continuously differentiable and not linear.

Keywords: Approximation; Neural; Network; Multilayer

1. Introduction and statement of results

The purpose of this note is to show that a multilayer perceptron (MLP) neural network with at most $d+d'+2$ nodes in each layer can approximate an arbitrary continuous $\mathbb{R}^{d'}$-valued function defined on $\mathbb{R}^d$ provided the node function is twice continuously differentiable and not linear. The standard way of proving that a given continuous function can be approximated by an MLP-network is to use one hidden layer and then consider the span of the ridge functions $\sigma(\langle \mathbf{w}, \underline{\mathbf{x}} \rangle - \tau)$, see e.g. [1,3,6,8], and in particular the survey [7] as well as the references mentioned there. But in this case the number of nodes in the hidden layer grows as one tries to achieve better and better approximations. Furthermore, one has to assume that the node

E-mail address: gustaf.gripenberg@hut.fi.

0021-9045/03/$ - see front matter
doi:10.1016/S0021-9045(03)00078-9

function is not a polynomial, because otherwise the output could only be a polynomial of at most the same degree as the node function.

In [5] it is shown that if one chooses a special node function then one obtains the approximation property for a network with two hidden layers and a fixed number of nodes in both layers. But since the proof is based on Kolmogorov Superposition Theorem (see e.g. [4]), the node function is rather pathological, although it can be chosen to be strictly increasing and real analytic.

Here we consider the case where one is given an arbitrary twice differentiable node function which is not linear and one wants to restrict the number of nodes in each layer, but one is willing to let the number of layers grow to infinity.

In [2] a related result is established, but there no restrictions are placed on the number of nodes in the hidden layers and hence it suffices to consider arbitrary continuous nonlinear node functions.

Apparently, not very much work has been done on the problem of what advantages, if any, there are in having several layers. Clearly the smaller the number of nodes needed, the better, but the difficulty is to evaluate the performance of a network structure. In [9] the authors conclude that network with two hidden layers ($L = 3$ in the notation below) is superior to a network with one hidden layer (with $L = 2$) whereas in [10] no such difference is found.

To simplify the statement of our result and to fix some notation we formally state what we mean by a multilayer perceptron network with at most k nodes in each layer. Here we assume (although it is, of course, not essential for these networks in general), that the node functions are the same at all nodes except that the output layer has a linear node function.

Definition 1. Let $\sigma : \mathbb{R} \to \mathbb{R}$ and suppose that d, d' and $k \geqslant 1$. Then one says that $\mathbf{f} \in MLP(d, d', \sigma, k, L)$ if and only if $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^{d'}$ is such that there are (weight) matrices $W_1 \in \mathbb{R}^{k\times d}$, $W_j \in \mathbb{R}^{k\times k}$, $1 < j < L$, $W_L \in \mathbb{R}^{d'\times k}$ and (threshold) vectors $\tau_j \in \mathbb{R}^{k\times 1}$, $1 \leqslant j < L$ and $\tau_L \in \mathbb{R}^{d',1}$ so that for every $\mathbf{x} \in \mathbb{R}^d$ one has $\mathbf{f}(\mathbf{x}) = \mathbf{a}_L$ when the vectors $\mathbf{a}_j$ (and $\mathbf{b}_j$) are defined by

$$\mathbf{b}_0 = \mathbf{x},$$

$$\mathbf{a}_j = W_j \mathbf{b}_{j-1} - \tau_j, \quad 1 \leqslant j \leqslant L,$$

$$\mathbf{b}_j = \sigma(\mathbf{a}_j), \quad 1 \leqslant j < L. \tag{1}$$

Here the expression $\sigma(\mathbf{a}_j)$ is the vector obtained by applying the function σ to each component of the vector $\mathbf{a}_j$. The definition above is written so that each hidden layer has exactly k nodes, but by choosing some of the weights equal to zero, the effective number of nodes could be less than k.

When considering approximation properties it suffices, in principle, to consider the case of real-valued functions, i.e., $d' = 1$, but since we want to have a simple upper bound on the number of nodes, we treat the general case.

We can use the following metric in the space $\mathscr{C}(\mathbb{R}^d;\mathbb{R}^{d'})$ of continuous functions on $\mathbb{R}^d$:

$$d(\mathbf{f},\mathbf{g})=\sum_{j=1}^{\infty}2^{-j}\frac{\sup_{|\mathbf{x}|\leqslant j}|\mathbf{f}(\mathbf{x})-\mathbf{g}(\mathbf{x})|}{1+\sup_{|\mathbf{x}|\leqslant j}|\mathbf{f}(\mathbf{x})-\mathbf{g}(\mathbf{x})|}. \tag{2}$$

It is easy to see that convergence in this metric is the same as uniform convergence on compact sets and clearly the choice of norms $|\cdot|$ in $\mathbb{R}^d$ and $\mathbb{R}^{d'}$ is of no consequence.

Now we can state our result:

Theorem 2. *Assume that* $d,d'\geqslant 1$ *and that* $\sigma\in\mathscr{C}(\mathbb{R};\mathbb{R})$ *is twice continuously differentiable in the neighbourhood of a point* $\tau_*\in\mathbb{R}$ *and* $\sigma''(\tau_*)\neq 0$. *Then the set* $\bigcup_{L\geqslant 1} MLP(d,d',\sigma,d+d'+2,L)$ (*see Definition* 1) *is dense in* $\mathscr{C}(\mathbb{R}^d;\mathbb{R}^{d'})$.

2. Proof of Theorem 2

Since σ is assumed to be twice continuously differentiable in a neighbourhood of τ_* we may, without loss of generality, assume that $\sigma'(\tau_*)\neq 0$ in addition to the assumption that $\sigma''(\tau_*)\neq 0$.

Since every continuous scalar function can be approximated with arbitrary precision on compact sets by polynomials, the space of (vector) polynomials is dense in $\mathscr{C}(\mathbb{R}^d;\mathbb{R}^{d'})$ with the metric defined in (2). Thus, we can conclude that it suffices to show that every polynomial can be approximated with arbitrary precision on every bounded set of $\mathbb{R}^d$.

Let $T>0$ and $\varepsilon\in(0,1)$ be arbitrary. We will construct an MLP-network using one of two slightly different constructions each time a layer is added.

Suppose $p_1,p_2,\ldots,p_{d'}$ and q are some real-valued polynomials in d variables. Assume that there exists an MLP-network with d inputs, L layers, $d+d'+2$ nodes at each layer (including the output layer), i.e., a function $\mathbf{f}\in MLP(d,d+d'+2,\sigma,d+d'+2,L)$ such that if the inputs are $\mathbf{x}=(\mathbf{x}(1),\ldots,\mathbf{x}(d))$ and the output $\mathbf{y}=\mathbf{f}(\mathbf{x})$ then we have when $|\mathbf{x}|\leqslant T$,

$$\begin{aligned}
&|\mathbf{y}(j)-\mathbf{x}(j)|\leqslant L\varepsilon,\quad j=1,\ldots,d,\\
&|\mathbf{y}(d+k)-p_k(\mathbf{x})|\leqslant B_L\varepsilon,\quad k=1,\ldots,d',\\
&|\mathbf{y}(d+d'+1)-q(\mathbf{x})|\leqslant C_L\varepsilon,\\
&\mathbf{y}(d+d'+2)=0.
\end{aligned} \tag{3}$$

To get the process started, we take $L=0$, add $d'+2$ zeroes to the input layer and thus take $p_1=\cdots=p_{d'}=q=0$, but we leave the details of this construction to the reader.

In the first alternative we want, assuming some numbers $c_1,c_2,\ldots,c_{d'}$ are given, to construct a network with $L+1$ layers, that is a function $\mathbf{g}\in MLP(d,d+d'+$

$2, \sigma, d+d'+2, L+1)$ such that if $\mathbf{z} = \mathbf{g}(\mathbf{x})$ and $|\mathbf{x}| \leqslant T$, then

$$
\begin{aligned}
&|\mathbf{z}(j) - \mathbf{x}(j)| \leqslant (L+1)\varepsilon, \quad j = 1, \dots, d,\\
&|\mathbf{z}(d+k) - p_k(\mathbf{x}) - c_k q(\mathbf{x})| \leqslant \left(B_L + \max_{1 \leqslant k \leqslant d'} |c_k| C_L + 1\right)\varepsilon, \quad k = 1, \dots, d',\\
&|\mathbf{z}(d+d'+1) - 1| = 0,\\
&\mathbf{z}(d+d'+2) = 0.
\end{aligned} \tag{4}
$$

In the second alternative we want, assuming m is one of the numbers $1, 2, \dots, d$, to construct a network with $L+1$ layers, that is a function $\mathbf{g} \in MLP(d, d+d'+2, \sigma, d+d'+2, L+1)$ such that if $\mathbf{z} = \mathbf{g}(\mathbf{x})$ and $|\mathbf{x}| \leqslant T$, then

$$
\begin{aligned}
&|\mathbf{z}(j) - x(j)| \leqslant (L+1)\varepsilon, \quad j = 1, \dots, d,\\
&|\mathbf{z}(d+k) - p_k(\mathbf{x})| \leqslant (B_L + 1)\varepsilon, \quad k = 1, \dots, d',\\
&|\mathbf{z}(d+d'+1) - \mathbf{x}(m) q(\mathbf{x})| \leqslant (C_L(T+L) + L \max_{|\mathbf{x}| \leqslant T} |q(\mathbf{x})| + 1)\varepsilon,\\
&\mathbf{z}(d+d'+2) = 0.
\end{aligned} \tag{5}
$$

In both cases we have to take the network defining the function $\mathbf{f}$, modify the matrix W_L and the vector τ_L in a way that will be described below and add a new layer.

First, we note that for the output terms that one wants to be unchanged we can use the following estimate when $\eta > 0$:

$$
\left| \frac{1}{\eta \sigma'(\tau_*)} (\sigma(\eta a + \tau_*) - \sigma(\tau_*)) - a \right| \leqslant \frac{\sup_{|s - \tau_*| \leqslant \eta |a|} |\sigma''(s)| \eta a^2}{2 |\sigma'(\tau_*)|}. \tag{6}
$$

For the first case we let $\eta > 0$ and define a $(d+d'+2) \times (d+d'+2)$ matrix V_L that will be used to modify W_L:

$$
V_L(i,j) = \begin{cases} \eta & \text{if } 1 \leqslant i = j \leqslant d+d'+1, \\ 0 & \text{otherwise.} \end{cases}
$$

We take ρ_L to be a $(d+d'+2) \times 1$ matrix with each element equal to $-\tau_*$. Next, we replace the matrix W_L by the new weight matrix $\tilde{W}_L = V_L W_L$ and the new threshold vector will be $\tilde{\tau}_L = V_L \tau_L + \rho_L$.

The next weight matrix W_{L+1} is defined by

$$
W_{L+1}(i,j) = \begin{cases} \dfrac{1}{\eta \sigma'(\tau_*)} & \text{if } 1 \leqslant i = j \leqslant d+d', \\ \dfrac{c_k}{\eta \sigma'(\tau_*)} & \text{if } i = d+k,\ j = 1+d+d',\ 1 \leqslant k \leqslant d', \\ 0 & \text{otherwise.} \end{cases}
$$

Similarly, the next threshold vector τ_{L+1} is defined by

$$\tau_{L+1}(i) = \begin{cases} \dfrac{\sigma(\tau_*)}{\eta\sigma'(\tau_*)} & \text{if } 1 \leqslant i \leqslant d, \\ (1+c_k)\dfrac{\sigma(\tau_*)}{\eta\sigma'(\tau_*)} & \text{if } i = d+k, 1 \leqslant k \leqslant d', \\ -1 & \text{if } i = d+d'+1, \\ 0 & \text{if } i = d+d'+2. \end{cases}$$

With these definitions we see that

$$\mathbf{z}(j) = \frac{\sigma(\eta y(j) + \tau_*) - \sigma(\tau_*)}{\eta\sigma'(\tau_*)}, \quad j = 1, \ldots, d,$$

$$\begin{aligned} \mathbf{z}(d+k) = &\frac{\sigma(\eta y(d+k) + \tau_*) - \sigma(\tau_*)}{\eta\sigma'(\tau_*)} \\ &+ c_k \frac{\sigma(\eta y(d+d'+1) + \tau_*) - \sigma(\tau_*)}{\eta\sigma'(\tau_*)}, \quad k = 1, \ldots, d', \end{aligned}$$

$$\mathbf{z}(d+d'+1) = 1,$$

$$\mathbf{z}(d+d'+2) = 0.$$

Thus we conclude from (3) and (6) that if η is sufficiently small, then (4) holds (when $|\mathbf{x}| \leqslant T$).

In the second case we proceed in the same manner and we shall again utilize (6). But in order to get an approximation for the term $\mathbf{x}(m)q(\mathbf{x})$ we must also use the observation that

$$\begin{aligned} &\left| \frac{\sigma(\eta a + \eta b + \tau_*) - \sigma(\eta a + \tau_*) - \sigma(\eta b + \tau_*) + \sigma(\tau_*)}{\eta^2\sigma''(\tau_*)} - ab \right| \\ &\quad \leqslant \frac{|a||b|}{\sigma''(\tau_*)} \sup_{|s| \leqslant \eta|a| + \eta|b|} |\sigma''(s + \tau_*) - \sigma''(\tau_*)|. \end{aligned} \tag{7}$$

In order to get the modified network in this case we define a matrix used to modify W_L:

$$V_L(i,j) = \begin{cases} \eta & \text{if } 1 \leqslant i = j \leqslant d + d' + 1, \\ \eta & \text{if } i = d + d' + 2,\ j = m \text{ or } j = d + d' + 1, \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, we choose $\rho_L(j) = -\tau_*$ for all $j = 1, \ldots, d + d' + 2$. Again we replace the old weight matrix W_L by a new one, $\tilde{W}_L = V_L W_L$ and the new threshold vector will be $\tilde{\tau}_L = V_L \tau_L + \rho_L$.

For the output layer we define the weight matrix as follows:

$$W_{L+1}(i,j) = \begin{cases} \dfrac{1}{\eta\sigma'(\tau_*)} & \text{if } 1 \leqslant i = j \leqslant d + d', \\ \dfrac{1}{\eta^2\sigma''(\tau_*)} & \text{if } i = d + d' + 1, j = d + d' + 2, \\ -\dfrac{1}{\eta^2\sigma''(\tau_*)} & \text{if } i = d + d' + 1,\ j = m \text{ or } j = d + d' + 1, \\ 0 & \text{otherwise.} \end{cases}$$

For the threshold vector we take

$$\tau_{L+1}(j) = \begin{cases} \dfrac{\sigma(\tau_*)}{\eta\sigma'(\tau_*)} & \text{if } j = 1, \ldots, d + d', \\ -\dfrac{\sigma(\tau_*)}{\eta^2\sigma''(\tau_*)} & \text{if } j = d + d' + 1, \\ 0 & \text{if } j = d + d' + 2. \end{cases}$$

We conclude that with these definitions we have

$$\mathbf{z}(j) = \frac{\sigma(\eta y(j) + \tau_*) - \sigma(\tau_*)}{\eta\sigma'(\tau_*)}, \quad j = 1, \ldots, d + d',$$

$$\begin{aligned} \mathbf{z}(d + d' + 1) = &\frac{1}{\eta^2\sigma''(\tau_*)}(\sigma(\eta y(d + d' + 1) + \eta y(m) + \tau_*) \\ &- \sigma(\eta y(d + d' + 1) + \tau_*) - \sigma(\eta y(m) + \tau_*) + \sigma(\tau_*)), \end{aligned}$$

$$\mathbf{z}(d + d' + 2) = 0.$$

In this case we see that if η is sufficiently small, then by (3), (6), and by (7) we get (5) (when we use the assumption that $\varepsilon < 1$).

Now the procedure works so that we start out with a step of the first type, then we build up (an approximation of) a monomial $\prod_{i=1}^{d} \mathbf{x}(i)^{\alpha_i}$ using $\sum_{i=1}^{d} \alpha_i$ steps of the second type, and this monomial is then added to the polynomials p_k, $k = 1, \ldots, d'$ in a step of the first type. Then the procedure is repeated for the next monomial until the polynomials (or more precisely, their approximations) are completed. (Clearly, it is not necessary to proceed in exactly this way, but the two steps could be combined.) At this point the last weight matrix and threshold vector must be modified so that the only outputs of the network are approximations of the desired polynomials, that is one should only keep the components numbered $d + 1, d + 2, \ldots, d + d'$ in the output vector. We see that the errors in the approximations are then some constant, that only depends on the polynomials and T times ε and by choosing ε sufficiently small we get the desired conclusion and the proof is completed.

References

[1] M. Burger, A. Neubauer, Error bounds for approximation with neural networks, J. Approx. Theory 112 (2) (2001) 235–250.
[2] A.N. Gorban′, Approximation of continuous functions of several variables by an arbitrary nonlinear continuous function of one variable, linear functions, and their superpositions, Appl. Math. Lett. 11 (3) (1998) 45–49.
[3] M. Leshno, V.Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Networks 6 (1993) 861–867.
[4] G.G. Lorentz, M.V. Golitschek, Y. Makovoz, Constructive Approximation, Advanced Problems, Springer, Berlin, 1996.
[5] V. Maiorov, A. Pinkus, Lower bounds for approximation by MLP neural networks, Neurocomputing 25 (1–3) (1999) 81–91.
[6] Y. Makovoz, Uniform approximation by neural networks, J. Approx. Theory 95 (2) (1998) 215–228.
[7] A. Pinkus, Approximation theory of the MLP model in neural networks, in: Acta Numerica, Vol. 8, Cambridge University Press, Cambridge, 1999, pp. 143–195.
[8] F. Scarselli, A.C. Tsoi, Universal approximation using feedforward neural networks: a survey of some existing methods and some new results, Neural Networks 11 (1998) 15–37.
[9] S. Tamura, M. Tateish, Capabilities of four-layered feedforward neural network: four layers versus three, IEEE Trans. Neural Networks 8 (2) (1997) 251–255.
[10] J. de Villiers, D. Barnard, Backpropagation neural nets with one and two hidden layers, IEEE Trans. Neural Networks 4 (12) (1992) 136–141.