Pergamon

Neural Networks, Vol. 7, No. 8, pp. 1285–1293, 1994
Copyright © 1994 Elsevier Science Ltd
Printed in the USA. All rights reserved
0893-6080/94 $6.00 + .00

0893-6080(94)E0018-G

CONTRIBUTED ARTICLE

How to Make Sigma-Pi Neural Networks Perform Perfectly on Regular Training Sets

BURKHARD LENZE

Fachbereich Informatik, Fachhochschule Dortmund

(*Received* 9 *September* 1992; *revised and accepted* 17 *January* 1994)

Abstract—*In this paper, we show how to design three-layer feedforward neural networks with sigma-pi units in the hidden layer in order to perform perfectly on regular training sets. We obtain real-time design schemes based on massively parallel sampling and induced by so-called hyperbolic cardinal translation-type interpolation operators. The real-time nature of our strategy is due to the fact that in the neural network language our approach is nothing else but a very general and efficient one-shot learning scheme. Moreover, because of the very special hyperbolic structure of our sigma-pi units we do not have the usual dramatic increase of parameters and weights that in general happens in case of higher order networks. The final networks are of manageable complexity and may be applied to multigroup discriminant problems, pattern recognition, and image processing. In detail, the XOR-problem and a special multigroup discriminant problem are discussed at the end of the paper.*

Keywords—Feedforward sigma-pi neural networks, Hyperbolic cardinal translation-type interpolation operators, Parallel sampling, Real-time update, One-shot learning scheme, B-splines, XOR-problem, Multigroup discriminant problems.

1. INTRODUCTION

Since Minsky and Papert's results (1969) it is well-known that usual feedforward neural networks with first-order units can implement only linearly separable mappings. One possibility to drop this limitation is to use multilayer networks where so-called hidden units can combine the outputs of previous units and so give rise to nonlinear mappings (cf. Rumelhart & McClelland, 1986; Irie & Miyake, 1988; Funahashi, 1989; Hornik, Stinchcombe, & White, 1989; Cybenko, 1989; Chui & Li, 1992, 1993; Light, 1992). The other way to overcome the restriction to linear maps is to introduce higher order units, sigma-pi units, to model nonlinear dependences (cf. Rumelhart & McClelland, 1986; Giles & Maxwell, 1987; Giles, Griffin, & Maxwell, 1988; Gorse & Taylor, 1991; Gorse, Taylor, & Clarkson, 1993; Lenze, 1992, 1993). Both modifications imply that the complexity of the networks increase dramatically and that standard training strategies (variants of back propagation, gradient method, steepest descent, etc.) even for small to medium sized problems often fail or are at least very time consuming (cf. Hecht–Nielsen, 1990; Saarinen, Bramley, & Cybenko, 1991a,b). Therefore, it should be of some interest to have real-time ad hoc strategies for adjusting the network parameters properly at least in the case of special situations. In the following, such a general one-shot learning scheme will be presented for three-layer feedforward neural networks with special hyperbolic sigma-pi units in the hidden layer and regular training sets.

Requests for reprints should be sent to Fachbereich Informatik, Fachhochschule Dortmund Postfach 105018, D-44047 Dortmund, Germany; E-mail: burkhard.lenze@fernuni-hagen.de

2. NOTATION AND RESULTS

Let $n \in \mathbb{N}$ be given and $\mathbf{a} = (a_1, a_2, \ldots, a_n)$, $\mathbf{b} = (b_1, b_2, \ldots, b_n) \in \mathbb{R}^n$ with $\mathbf{a} < \mathbf{b}$ (i.e., $a_k < b_k$, $1 \le k \le n$) the endpoints of the interval $[\mathbf{a}, \mathbf{b}] \subset \mathbb{R}^n$,

$$[\mathbf{a}, \mathbf{b}] := \{\mathbf{x} \in \mathbb{R}^n \mid a_k \le x_k \le b_k,\ 1 \le k \le n\}. \tag{1}$$

By means of a standard translation argument we may assume that the point $\mathbf{a} \in \mathbb{R}^n$ is always equal to the origin, that is, without loss of generality we have $\mathbf{a} = \mathbf{0}$. Going ahead, we choose $J_1, J_2, \ldots, J_n \in \mathbb{N}$ and define $\mathbf{h} \in \mathbb{R}^n$ componentwise by

$$h_k := \frac{b_k}{J_k}, \quad 1 \le k \le n. \tag{2}$$

Now, the interval $[\mathbf{0}, \mathbf{b}]$ has a regular grid with grid points

$$\mathbf{h_j} := (h_1 j_1, h_2 j_2, \dots, h_n j_n) \in [\mathbf{0}, \mathbf{b}], \qquad 0 \le j_k \le J_k, \quad 1 \le k \le n, \tag{3}$$

where $\mathbf{j} := (j_1, j_2, \dots, j_n)$ and $\mathbf{J} := (J_1, J_2, \dots, J_n)$. Next, at each grid point $\mathbf{h_j}$, $\mathbf{0} \le \mathbf{j} \le \mathbf{J}$, there may be given a real value $f(\mathbf{h_j}) \in \mathbb{R}$ that we may assume to come from some underlying function $f\colon [\mathbf{0}, \mathbf{b}] \to \mathbb{R}$. Now, the problem is to design a neural network that is able to associate with each grid point $\mathbf{h_j}$ the right value $f(\mathbf{h_j})$. To solve this problem we proceed in two steps: in a first step, we define so-called hyperbolic cardinal translation-type interpolation operators that formally interpolate the data set. In a second step, we show how these operators can be interpreted as realizations of usual three-layer feedforward sigma-pi neural networks.

To do the first step, we first of all need some further definitions that may already be found in Lenze (1989, 1992). For arbitrary $\mathbf{c}, \mathbf{d} \in \mathbb{R}^n$ with $\mathbf{c} < \mathbf{d}$ let

$$C \text{ or } [\mathbf{c}, \mathbf{d}] := \{\mathbf{x} \in \mathbb{R}^n \mid x_k = c_k \vee x_k = d_k,\ 1 \le k \le n\} \tag{4}$$

be the set of corners of the interval $[\mathbf{c}, \mathbf{d}]$. Moreover, let

$$\gamma(\mathbf{x}, \mathbf{c}) := \#\{k \in \{1, 2, \dots, n\} \mid x_k = c_k\}, \qquad \mathbf{x} \in C \text{ or } [\mathbf{c}, \mathbf{d}]. \tag{5}$$

In (5) $\#$ denotes the number of distinct elements of the set under consideration and $n - \gamma(\mathbf{x}, \mathbf{c})$ is nothing else but the well-known Hamming distance of $\mathbf{x}$ and $\mathbf{c}$ (cf. Hecht-Nielsen, 1990: 43, for example). Now, for a given function $f\colon [\mathbf{c}, \mathbf{d}] \to \mathbb{R}$ the so-called corresponding interval function or iterated difference Δ_f of f at $[\mathbf{c}, \mathbf{d}]$ is defined by

$$\Delta_f[\mathbf{c}, \mathbf{d}] := \sum_{\mathbf{x} \in C\text{or}[\mathbf{c},\mathbf{d}]} (-1)^{\gamma(\mathbf{x},\mathbf{c})} f(\mathbf{x}). \tag{6}$$

In case $n = 1$, definition (6) reduces to

$$\Delta_f[\mathbf{c}, \mathbf{d}] = f(\mathbf{d}) - f(\mathbf{c}) = f(d_1) - f(c_1) \tag{7}$$

and in case $n = 2$ to

$$\Delta_f[\mathbf{c}, \mathbf{d}] = f(d_1, d_2) - f(d_1, c_2) - f(c_1, d_2) + f(c_1, c_2). \tag{8}$$

Moreover, let us note that the interval function Δ_f is additive which means that

$$[\mathbf{c}, \mathbf{d}] = [\mathbf{c}^{(1)}, \mathbf{d}^{(1)}] \cup [\mathbf{c}^{(2)}, \mathbf{d}^{(2)}] \quad \text{and} \quad (\mathbf{c}^{(1)}, \mathbf{d}^{(1)}) \cap (\mathbf{c}^{(2)}, \mathbf{d}^{(2)}) = \varnothing \tag{9}$$

implies

$$\Delta_f[\mathbf{c}, \mathbf{d}] = \Delta_f[\mathbf{c}^{(1)}, \mathbf{d}^{(1)}] + \Delta_f[\mathbf{c}^{(2)}, \mathbf{d}^{(2)}] \tag{10}$$

(cf. Saks, 1937, for example). We finish this part of basic definitions by introducing the notation of a generalized sigmoidal function.

DEFINITION 1. *A bounded function $\sigma\colon \mathbb{R} \to \mathbb{R}$ is called a generalized sigmoidal function, if*

$$\lim_{\xi \to -\infty} \sigma(\xi) = 0 \quad \textit{and} \quad \lim_{\xi \to \infty} \sigma(\xi) = 1. \tag{11}$$

REMARK. The term "generalized" means that we allow σ to be even discontinuous in order to include perceptron-type networks, too. In these cases, the sigmoidal function is equal to the step function $1\colon \mathbb{R} \to \{0, 1\}$, namely,

$$1(\xi) := \begin{cases} 0, & \xi < 0, \\ 1, & \xi \ge 0. \end{cases} \tag{12}$$

Now, we are prepared to introduce the hyperbolic cardinal translation-type operators $\Omega^{(\mathbf{h})}$. For σ a generalized sigmoidal function, $\mathbf{e} := (1, 1, \dots, 1) \in \mathbb{Z}^n$, $\mathbf{h} \in \mathbb{R}^n$ given by (2), and $f(\mathbf{h_j}) \in \mathbb{R}$, $\mathbf{0} \le \mathbf{j} \le \mathbf{J}$, the given data set the operator $\Omega^{(\mathbf{h})}$ is defined for all $\mathbf{x} \in [\mathbf{0}, \mathbf{b}]$ as

$$\Omega^{(\mathbf{h})}(f)(\mathbf{x}) := (-1)^n 2^{1-n} \sum_{\substack{\mathbf{j} \in \mathbb{Z}^n \\ -\mathbf{e} \le \mathbf{j} \le \mathbf{J}}} \sigma\left(\prod_{k=1}^{n} \left(j_k + \frac{1}{2} - \frac{x_k}{h_k}\right)\right) \Delta_f[\mathbf{h_j}, \mathbf{h_{j+e}}] \tag{13}$$

where we agree to set

$$f(\mathbf{h_j}) := 0, \quad \mathbf{h_j} \notin [\mathbf{0}, \mathbf{b}], \tag{14}$$

in order to obtain a compact notation of the n finite sums appearing in (13). Obviously, the operator is basically induced by the given sigmoidal function σ evaluated at component-wise products of shifted and scaled arguments, so-called hyperbolic-type arguments, and the interval function Δ_f that makes use of all underlying discrete information. By nature, the operator depends linearly on the given information because Δ_f is a linear functional. To get a more precise idea of the behaviour of the operator, especially with respect to the data it should reproduce, we need the following lemma.

LEMMA 1. *Using the notations and definitions given above we have for all grid points $\mathbf{h_i} \in [\mathbf{0}, \mathbf{b}]$, $\mathbf{0} \le \mathbf{i} \le \mathbf{J}$,*

$$f(\mathbf{h_i}) = (-1)^n 2^{1-n} \sum_{\substack{\mathbf{j} \in \mathbb{Z}^n \\ -\mathbf{e} \le \mathbf{j} \le \mathbf{J}}} 1\left(\prod_{k=1}^{n} \left(j_k + \tfrac{1}{2} - i_k\right)\right) \Delta_f[\mathbf{h_j}, \mathbf{h_{j+e}}]. \tag{15}$$

Proof. We only sketch the proof for the two-dimensional case; the general ideas and notations to handle the arbitrary n-dimensional case may be found in Lenze (1989, Lemmas 3.1 and 3.2).

Let $n = 2$ and $\mathbf{h_i} \in [\mathbf{0}, \mathbf{b}] = [0, b_1] \times [0, b_2]$ be given arbitrarily. Because of the special behaviour of the step function 1 and the additivity of the interval function Δ_f, we obtain by splitting the sum in (15)

$$\tfrac{1}{2} \sum_{\substack{\mathbf{j} \in \mathbb{Z}^2 \\ -\mathbf{e} \le \mathbf{j} \le \mathbf{J}}} 1\left(\prod_{k=1}^{2} \left(j_k + \tfrac{1}{2} - i_k\right)\right) \Delta_f[\mathbf{h_j}, \mathbf{h_{j+e}}]$$

$$= \tfrac{1}{2} \sum_{\substack{-1 \le j_1 < i_1 \\ -1 \le j_2 < i_2}} \underbrace{1\left(\prod_{k=1}^{2} \left(j_k + \tfrac{1}{2} - i_k\right)\right)}_{=1} \Delta_f[\mathbf{h_j}, \mathbf{h_{j+e}}]$$

$$+ \tfrac{1}{2} \sum_{\substack{i_1 \le j_1 \le J_1 \\ -1 \le j_2 < i_2}} \underbrace{1\left(\prod_{k=1}^{2} \left(j_k + \tfrac{1}{2} - i_k\right)\right)}_{=0} \Delta_f[\mathbf{h_j}, \mathbf{h_{j+e}}]$$

$$+ \tfrac{1}{2} \sum_{\substack{-1 \le j_1 < i_1 \\ i_2 \le j_2 \le J_2}} \underbrace{1\left(\prod_{k=1}^{2} \left(j_k + \tfrac{1}{2} - i_k\right)\right)}_{=0} \Delta_f[\mathbf{h_j}, \mathbf{h_{j+e}}]$$

$$+ \tfrac{1}{2} \sum_{\substack{i_1 \le j_1 \le J_1 \\ i_2 \le j_2 \le J_2}} \underbrace{1\left(\prod_{k=1}^{2} \left(j_k + \tfrac{1}{2} - i_k\right)\right)}_{=1} \Delta_f[\mathbf{h_j}, \mathbf{h_{j+e}}]$$

$$= \tfrac{1}{2}(\Delta_f[\mathbf{h_{-e}}, \mathbf{h_i}] + \Delta_f[\mathbf{h_i}, \mathbf{h_{J+e}}])$$

$$= \tfrac{1}{2}\left(f(\mathbf{h_i}) + f(\mathbf{h_i})\right)$$

$$= f(\mathbf{h_i}). \tag{16}$$

Here, the last but one identity follows from the definition of Δ_f and the fact that $f(\mathbf{h_j}) = 0$ for $\mathbf{h_j} \notin [\mathbf{0}, \mathbf{b}]$, respectively, for $\mathbf{j} \notin [\mathbf{0}, \mathbf{J}]$. ■

With the above lemma we now can formulate our main result.

THEOREM 1. *Using the notations and definitions given above we additionally assume that* $\sigma: \mathbb{R} \to \mathbb{R}$ *is a generalized sigmoidal function satisfying*

$$\sigma(\xi) = 1(\xi), \quad |\xi| \ge 2^{-n}. \tag{17}$$

Then for all grid points $\mathbf{h_i} \in [\mathbf{0}, \mathbf{b}]$, $\mathbf{0} \le \mathbf{i} \le \mathbf{J}$, *the operators* $\Omega^{(\mathbf{h})}$ *defined in* (13) *yield*

$$\Omega^{(\mathbf{h})}(f)(\mathbf{h_i}) = f(\mathbf{h_i}), \tag{18}$$

that is, the operators interpolate the given data on the regular grid $\mathbf{h_i}$, $\mathbf{0} \le \mathbf{i} \le \mathbf{J}$.

Proof. Let $\mathbf{h_i} \in [\mathbf{0}, \mathbf{b}]$ be given arbitrarily. Because σ satisfies (17) and since

$$\left|\prod_{k=1}^{n} \left(j_k + \tfrac{1}{2} - i_k\right)\right| \ge 2^{-n} \tag{19}$$

is true for all $\mathbf{j} \in \mathbb{Z}^n$ we obtain by means of Lemma 1

$$\Omega^{(\mathbf{h})}(f)(\mathbf{h_i}) = (-1)^n 2^{1-n} \sum_{\substack{\mathbf{j} \in \mathbb{Z}^n \\ -\mathbf{e} \le \mathbf{j} \le \mathbf{J}}} \sigma\left(\prod_{k=1}^{n} \left(j_k + \tfrac{1}{2} - i_k\right)\right) \Delta_f[\mathbf{h_j}, \mathbf{h_{j+e}}]$$

$$= (-1)^n 2^{1-n} \sum_{\substack{\mathbf{j} \in \mathbb{Z}^n \\ -\mathbf{e} \le \mathbf{j} \le \mathbf{J}}} 1\left(\prod_{k=1}^{n} \left(j_k + \tfrac{1}{2} - i_k\right)\right) \Delta_f[\mathbf{h_j}, \mathbf{h_{j+e}}]$$

$$= f(\mathbf{h_i}). \tag{20}$$

■

With the above theorem we have finished the first part of our initial program, namely, we have defined some abstract operators that proved to interpolate the underlying regular gridded data. It remains to show how these operators are connected with three-layer feedforward sigma-pi neural networks. We start with a formal definition of these networks.

A prototype of a three-layer feedforward sigma-pi neural network consists of an input layer with n fixed input units, a hidden layer with N so-called sigma-pi (or higher order) units, and an output layer with one single output unit. Each input unit is connected with all hidden units and each input signal may be multiplied with each possible collection of other input signals before weighting and further processing (weights $w_R^{(m)}$, $\varnothing \ne R \subset \{1, 2, \ldots, n\}$, $1 \le m \le N$). Moreover, each hidden unit is connected with the output unit (weights $\alpha^{(m)}$, $1 \le m \le N$) and has a fixed given threshold $\Theta^{(m)}$, $1 \le m \le N$. Summing up, at the output unit the network answers with

$$\mathrm{OUT}(\mathbf{x}) = \sum_{m=1}^{N} \alpha^{(m)} \sigma\left(\sum_{\substack{R \ne \varnothing \\ R \subset \{1,2,\ldots,n\}}} w_R^{(m)} \prod_{k \in R} x_k - \Theta^{(m)} \right), \tag{21}$$

where σ is any fixed generalized sigmoidal function (see also Figure 1).

Now, in order to see that our operators (13) are really of type (21), we have to evaluate the product in the argument of σ and reorganize it in terms of products of x_k, $1 \le k \le n$. More precisely, we have

$$\prod_{k=1}^{n} \left(j_k + \frac{1}{2} - \frac{x_k}{h_k}\right)$$

$$= \sum_{\substack{R \cap S = \varnothing \\ R \cup S = \{1,2,\ldots,n\}}} \prod_{k \in S} \left(j_k + \frac{1}{2}\right) \prod_{k \in R} (-h_k)^{-1} x_k \tag{22}$$

with

$$\Theta^{(\mathbf{j})} := -\prod_{k=1}^{n} \left(j_k + \tfrac{1}{2}\right) \tag{23}$$

the threshold given by the summation term corresponding to $R = \varnothing$, except for sign. Moreover, we have to enumerate our hidden units not in terms of multiindices but in terms of natural numbers, that is, we have to identify the set $\{\mathbf{j} \in \mathbb{Z}^n : -\mathbf{e} \le \mathbf{j} \le \mathbf{J}\}$ with

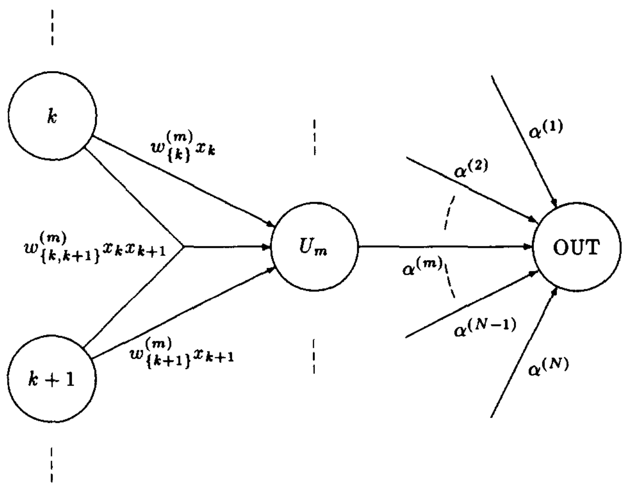

FIGURE 1. Input layer (left), hidden layer (middle), and output layer (right).

an appropriate set $\{m \in \mathbb{N}\colon 1 \leq m \leq N\}$. With this one-to-one correspondence in mind each factor $(-1)^n 2^{1-n} \Delta_f[\mathbf{h}_\mathbf{j}, \mathbf{h}_{\mathbf{j}+\mathbf{e}}]$ is nothing else but one easily computable output weight $\alpha^{(m)}$. Especially, all output weights together may be computed massively parallel by means of simultaneous sampling on the underlying data grid and the representation of (13) in terms of (21) is obvious. However, in view of concrete implementations and for reasons of complexity control we naturally prefer the compact hyperbolic representation (13) instead of the formal sigma-pi representation given by (21)–(23).

In conclusion, we have shown that our operators may be interpreted as explicit ad hoc realizations of perfectly trained three-layer feedforward sigma-pi neural networks with real-time update behaviour and manageable complexity. Before we now apply them to some concrete examples we have to add some final remarks.

REMARKS. First, networks of sigma-pi-type have been considered by various authors and have proved to be useful in order to recognize translation or rotation invariances and higher order correlations. Without a claim of completeness we mention the PDP Research Group around Rumelhart and McClelland (1986), Giles et al. (1987, 1988), and the contributions of Taylor and coworkers (Gorse & Taylor, 1991; Clarkson et al., 1993; Gorse et al., 1993). In the special case of exact representation of Boolean functions in terms of sigma-pi-type networks our results are closely related to those of Taylor et al. More precisely, in the deterministic case the pRAM-model of Taylor can exactly simulate any Boolean function, in our terminology, it can interpolate the data in a similar way as our networks do. Second, in view of our definition of the network operators given in (13) it should be also of interest to have some information on the nature of their behaviour between the given grid points or, in other words, on the generalization powers of the resulting approximation. These aspects are discussed in detail in Lenze (1992, 1993) where even error estimates are given in case that some quite weak assumptions on the structure of the information to be simulated can be presupposed. We do not want to go into further details here, but would like to encourage the reader to take a look at the following examples in view of the generalization capabilities of our strategy.

3. EXAMPLES

As already stated, we will now apply our network realizations to the so-called XOR-problem and to a special multigroup discriminant problem. Therefore, we first of all have to face the question as to which sigmoidal function(s) we should use in order to generate our operators, respectively, networks. Because of the essential condition (17) a nice way to get sigmoidal functions of the requested type is to start with the well-known B-splines (cf. Schumaker, 1981, for example) and integrate them. In the simplest case, let $B_1\colon \mathbb{R} \to \mathbb{R}$ be the cardinal B-spline of degree 1 centered at the origin with integral equal to 1,

$$B_1(\tau) := \begin{cases} 1 - |\tau|, & |\tau| \leq 1, \\ 0, & |\tau| > 1, \end{cases} \tag{24}$$

also often called hat- or roof-function. Integrating B_1 leads to

$$\sigma(\xi) := \int_{-\infty}^{\xi} B_1(\tau)d\tau = \begin{cases} 0, & \xi \leq -1, \\ \frac{1}{2}\xi^2 + \xi + \frac{1}{2}, & -1 < \xi \leq 0, \\ -\frac{1}{2}\xi^2 + \xi + \frac{1}{2}, & 0 < \xi \leq 1, \\ 1, & \xi > 1. \end{cases} \tag{25}$$

Now, it may be easily checked that any scaled version $\sigma^{(\beta)}\colon \mathbb{R} \to \mathbb{R}$ of σ,

$$\sigma^{(\beta)}(\xi) := \sigma(\beta\xi), \quad \xi \in \mathbb{R}, \quad \beta > 0, \tag{26}$$

induces a differentiable sigmoidal function that satisfies (17) in case of β sufficiently large and that generates smooth operators defined by (13). In the following, we will consider the special case $n = 2$ and use scaling factors $\beta = 4$ (smallest β to guarantee interpolation in the two-dimensional setting), $\beta = 10$ (medium sized β with less smooth interpolation surfaces on the one hand side but already larger regions of almost proper decision on the other hand side), and $\beta = \infty$ (nonsmooth perceptron-type network with $\sigma^{(\infty)} := 1$ and with largest regions of proper decision). In detail, our operators are

$$\Omega^{(\mathbf{h},\beta)}(f)(\mathbf{x}) := \frac{1}{2} \sum_{\substack{\mathbf{j}\in\mathbb{Z}^2 \\ -\mathbf{e}\leq\mathbf{j}\leq\mathbf{J}}} \sigma^{(\beta)}\left(\prod_{k=1}^{2}\left(j_k + \frac{1}{2} - \frac{x_k}{h_k}\right)\right)\Delta_f[\mathbf{h}_\mathbf{j}, \mathbf{h}_{\mathbf{j}+\mathbf{e}}] \tag{27}$$

with $\mathbf{e} := (1, 1) \in \mathbb{Z}^2$, $\beta = 4, 10, \infty$, and $\mathbf{x} \in [\mathbf{0}, \mathbf{b}]$.

As a first example, we apply our operators defined in (27) to the well-known XOR-problem that deals with one of the simplest nonlinear separable mappings (cf. Rumelhart & McClelland, 1986; and Hecht-Nielsen, 1990, for the history and details of the XOR-problem). In the standard setting, the XOR-function maps from $\{0, 1\} \times \{0, 1\}$ to $\{0, 1\}$ and is defined as

$$\mathrm{XOR}(x_1, x_2) := \begin{cases} 0, & x_1 = x_2 = 0 \quad \text{or} \quad x_1 = x_2 = 1, \\ 1, & x_1 = 1, x_2 = 0 \quad \text{or} \quad x_1 = 0, x_2 = 1. \end{cases} \tag{28}$$

If we now want to simulate or better interpolate this function with our sigma-pi neural network operators we have to make the identifications $[\mathbf{0}, \mathbf{b}] := [0, 1] \times [0, 1]$, $\mathbf{J} := (1, 1)$, and $\mathbf{h} := (1, 1) = \mathbf{e}$, and obtain

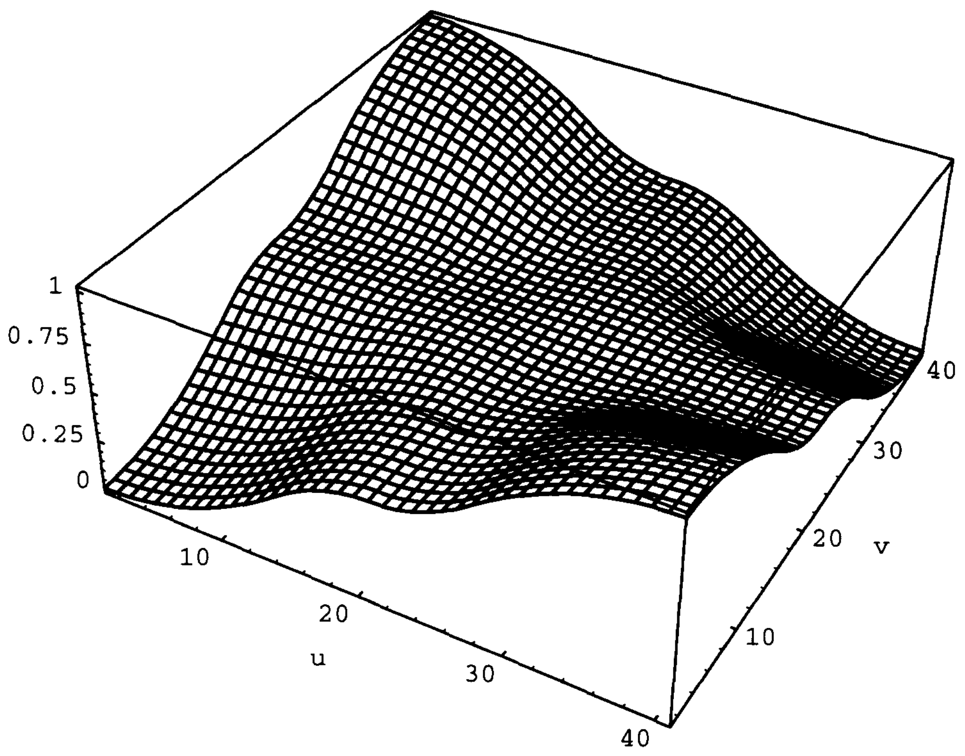

FIGURE 2. Interpolating XOR-surface $\Omega^{(\mathbf{e},4)}(\mathrm{XOR})(p(u, v))$.

$$\Omega^{(\mathbf{e},\beta)}(\mathrm{XOR})(\mathbf{x}) = \frac{1}{2} \sum_{\substack{-1 \le j_1 \le 1 \\ -1 \le j_2 \le 1}} \sigma^{(\beta)}\left(\prod_{k=1}^{2}\left(j_k + \frac{1}{2} - x_k\right)\right)\Delta_{\mathrm{XOR}}[\mathbf{j}, \mathbf{j} + \mathbf{e}]. \tag{29}$$

The plots of (29) for $\beta = 4, 10, \infty$ are shown in Figures 2–4 where we have parameterized the region of interest $[0, 1] \times [0, 1]$ with respect to the one-to-one correspondence p,

$$p\colon [0, 40]^2 \to [0, 1]^2, \quad (u, v) \mapsto \left(\frac{u}{40}, \frac{v}{40}\right). \tag{30}$$

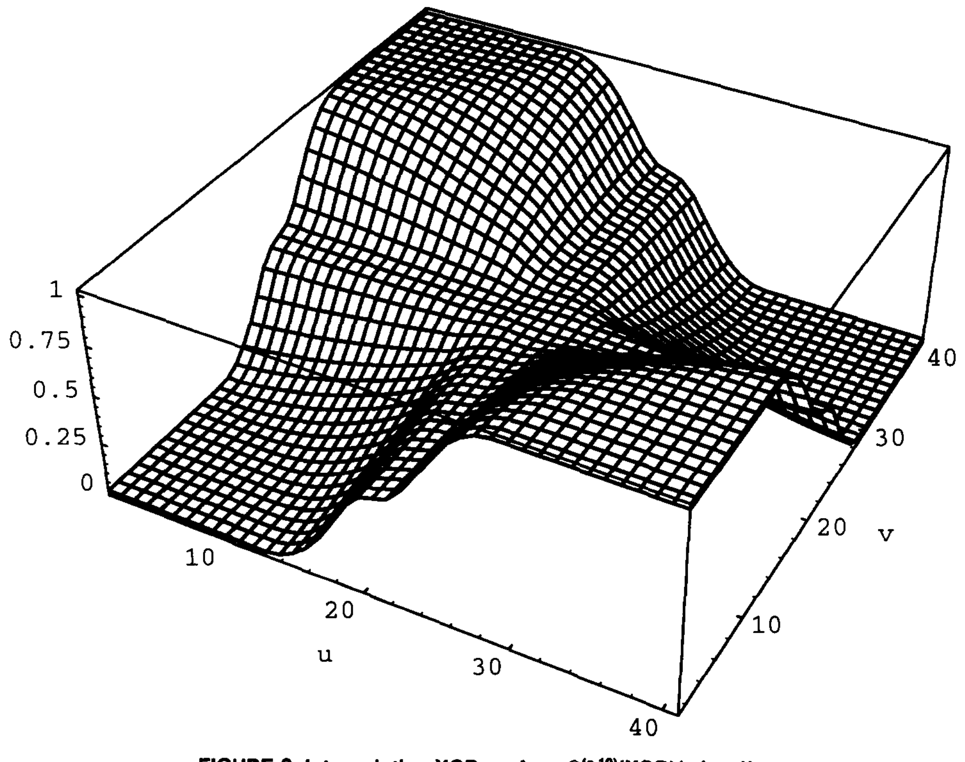

FIGURE 3. Interpolating XOR-surface $\Omega^{(\mathbf{e},10)}(\mathrm{XOR})(p(u, v))$.

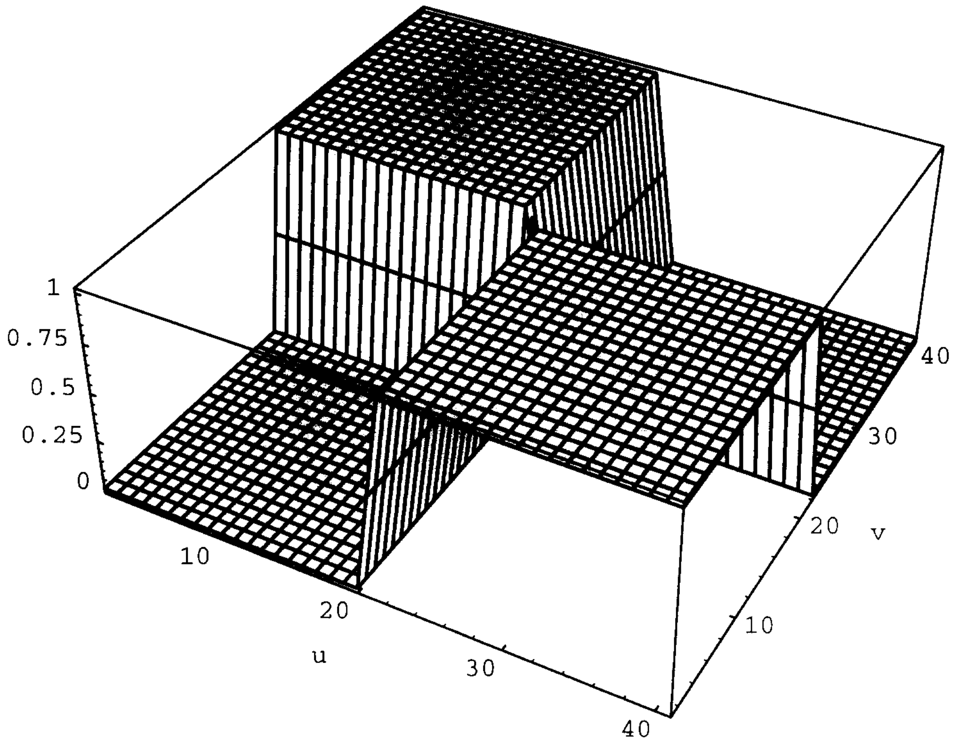

FIGURE 4. Interpolating XOR-surface $\Omega^{(\bullet,\infty)}(\mathrm{XOR})(p(u, v))$.

The XOR-surface shown in Figure 2 ($\beta = 4$) is the smoothest one and should be compared with the one presented in Lapedes and Farber (1988, fig 10). In Figure 3 ($\beta = 10$) the shape of the surface is already partially quite steep and the regions of the almost proper decision have become larger. Finally, Figure 4 shows the sigma-pi perceptron XOR-surface ($\beta = \infty$) where sharp edges of discontinuity appear and all points of the interval under consideration belong to a certain region of proper decision. At this point, it is perhaps convenient to say a few words about the solution of a problem closely related to the XOR-problem, namely, the so-called NXOR-problem,

$$\mathrm{NXOR}(x_1, x_2) := \begin{cases} 1, & x_1 = x_2 = 0 \quad \text{or} \quad x_1 = x_2 = 1, \\ 0, & x_1 = 1, x_2 = 0 \quad \text{or} \quad x_1 = 0, x_2 = 1. \end{cases} \tag{31}$$

Obviously, for $\beta \geq 4$ all functions

$$\sigma^{(\beta)}\left(\prod_{k=1}^{2}\left(\tfrac{1}{2} - x_k\right)\right), \quad (x_1, x_2) \in [0, 1]^2, \tag{32}$$

give rise to proper interpolating NXOR-surfaces. In other words, taking linear combinations of translations and dilations of the fundamental solution of the NXOR-problem can be seen as the underlying basic mechanism that makes our general sigma-pi neural network operators work.

At the end, we take a look at a special multigroup discriminant problem that should be seen in loose connection with a similar one discussed in Ishibuchi, Fujioka, and Tanaka (1992). Let the discriminant function f: $[0, 4]^2 \to \{0, 0.5, 1, 1.5, 2\}$ be defined as

$$f(x_1, x_2) := \begin{cases} 2, & x_1 \geq 3 \quad \text{and} \quad x_2 \geq 3, \\ 1.5, & (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5, \\ 1, & |x_1 - 1.5| + |x_2 - 2| \leq 1, \\ 0.5, & |x_1 - 2.5| + |x_2 - 2| \leq 1 \quad \text{and} \\ & |x_1 - 1.5| + |x_2 - 2| > 1, \\ 0, & \text{elsewhere in } [0, 4]^2 \text{ and outside } [0, 4]^2. \end{cases} \tag{33}$$

We set $[\mathbf{0}, \mathbf{b}] := [0, 4] \times [0, 4]$, $\mathbf{J} := (20, 20)$, and $\mathbf{h} := (4/20, 4/20) = 0.2\mathbf{e}$, and obtain

$$\Omega^{(0.2\mathbf{e},\beta)}(f)(\mathbf{x}) = \frac{1}{2} \sum_{\substack{-1 \leq j_1 \leq 20 \\ -1 \leq j_2 \leq 20}} \sigma^{(\beta)}\left(\prod_{k=1}^{2}\left(j_k + \frac{1}{2} - \frac{x_k}{0.2}\right)\right)\Delta_f[0.2\mathbf{j}, 0.2(\mathbf{j} + \mathbf{e})]. \tag{34}$$

The plots of the multigroup discriminant function f and its interpolation-type network realizations (34) for $\beta = 4, 10, \infty$ are shown in Figures 5–8 where in these cases we have used the parameterization q,

$$q\colon [0, 60]^2 \to [0, 4]^2, \quad (u, v) \mapsto \left(\frac{u}{15}, \frac{v}{15}\right). \tag{35}$$

Because the surface parameterization has been chosen three times finer than the interpolation grid $\mathbf{h_j} = \frac{1}{5}(j_1, j_2)$, $0 \leq j_1, j_2 \leq 20$, the plots give a quite complete idea

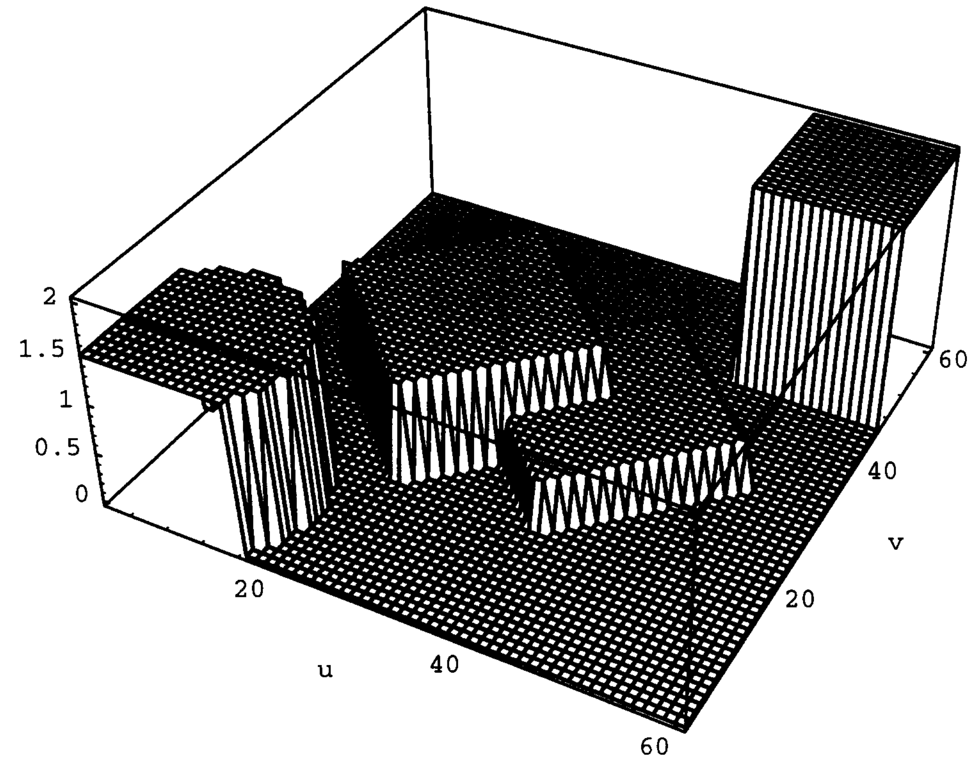

FIGURE 5. Discriminant surface $f(q(u, v))$.

on how the networks generalize, respectively, extrapolate on noninterpolation points. Especially, in view of the shape of proper decision regions, Figures 6–8 may be discussed in the same way as has already been done previously for Figures 2–4.

4. CONCLUSIONS

In this paper, we presented a real-time design strategy for three-layer feedforward sigma-pi neural networks with manageable complexity in the case of regular

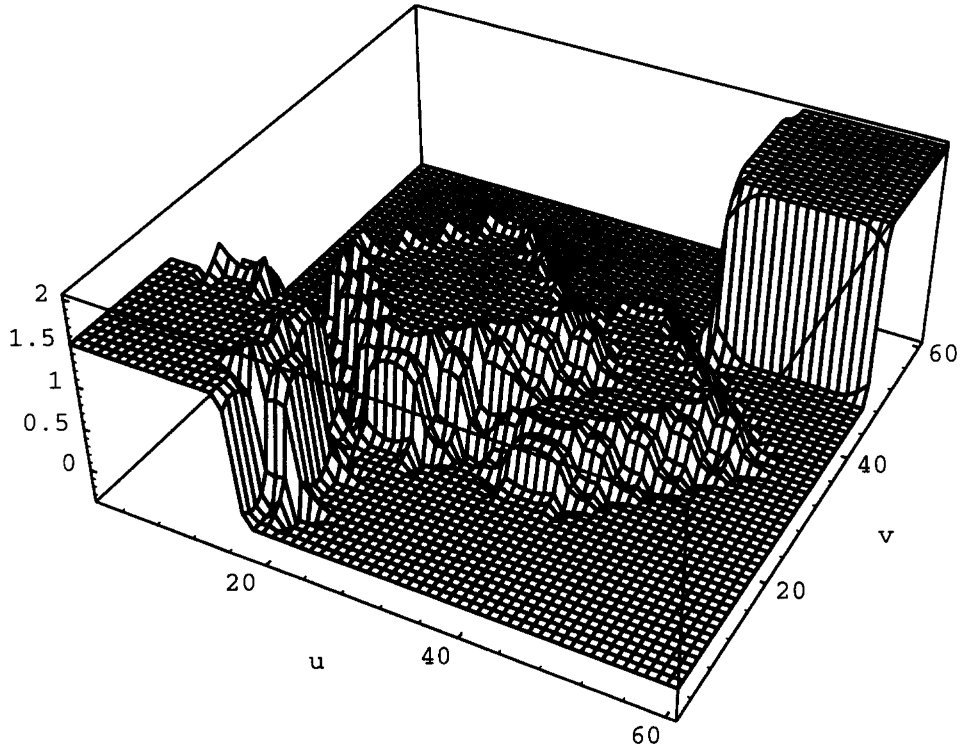

FIGURE 6. Interpolating discriminant surface $\Omega^{(0.2e,4)}(f)(q(u, v))$.

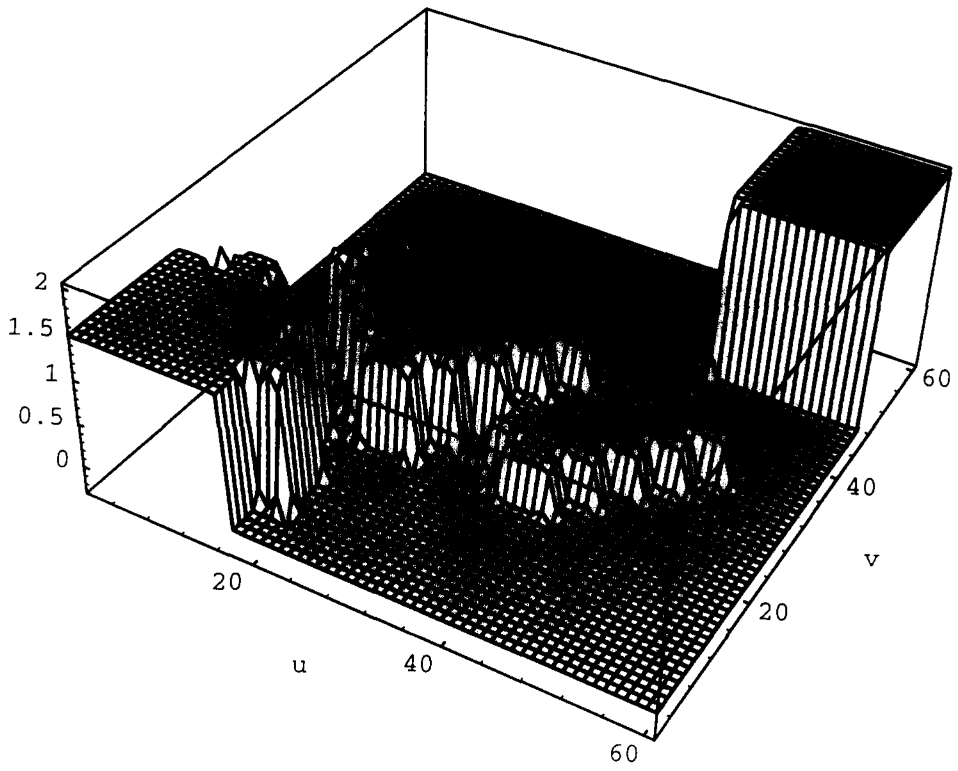

FIGURE 7. Interpolating discriminant surface $\Omega^{(0.2e,10)}(f)(q(u, v))$.

training sets. We proved that the networks obtained by our general one-shot learning scheme interpolate the training data, in other words, that they perform perfectly on the underlying discrete (partial) information. The basic idea to get the interpolation networks was to introduce so-called hyperbolic cardinal translation-type

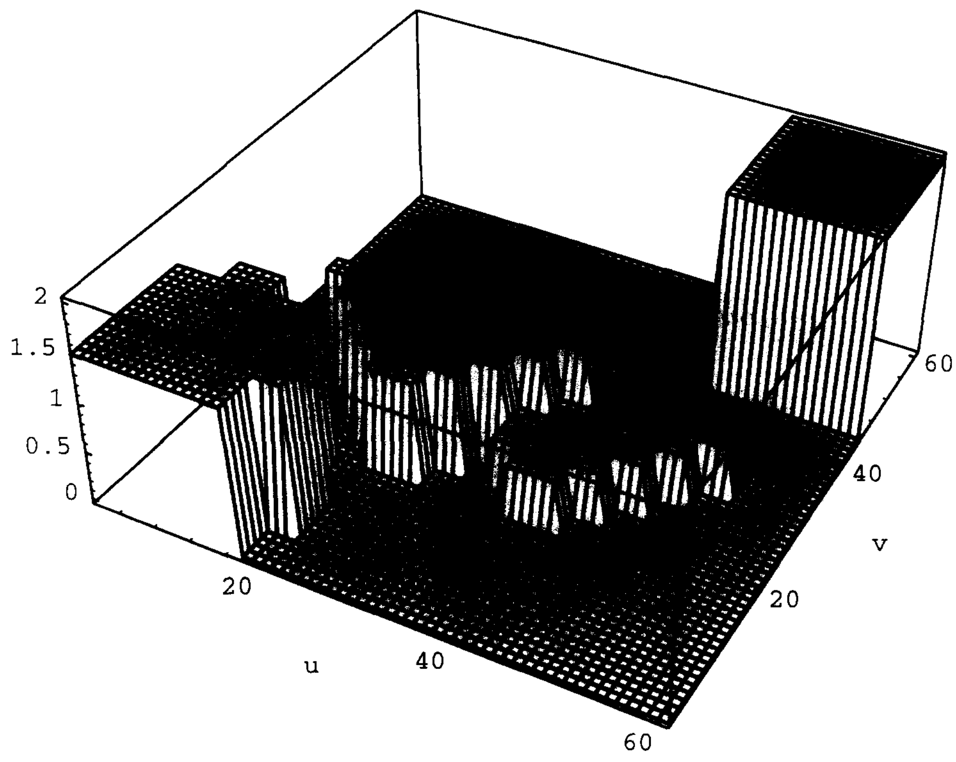

FIGURE 8. Interpolating discriminant surface $\Omega^{(0.2e,\infty)}(f)(q(u, v))$.

interpolation operators that were induced by special sigmoidal functions (key word: integrated B-splines) and by simple linear functionals operating on the given discrete information. Moreover, the special structure of the generating functionals allow the network output weights to be computed and set simultaneously and the other network parameters are data-independent and fixed in advance. Finally, the networks should be of interest for all applications where sigma-pi units make sense and regular training sets can be made available, for example, all aspects of image recognition and processing.

REFERENCES

Chui, C. K., & Li, X. (1992). Approximation by ridge functions and neural networks with one hidden layer. *Journal of Approximation Theory,* **70,** 131–141.

Chui, C. K., & Li, X. (1993). Realization of neural networks with one hidden layer. In K. Jetter & F. Utreras (Eds.), *Multivariate approximation: From CAGD to wavelets* (pp. 77–89). Singapore: World Scientific.

Clarkson, T. G., Guan, Y., Taylor, J. G., & Gorse, D. (1993). Generalization in probabilistic RAM nets. *IEEE Transactions on Neural Networks,* **4,** 360–363.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems,* **2,** 303–314.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks,* **2,** 183–192.

Giles, C. L., & Maxwell, T. (1987). Learning, invariance, and generalization in high-order neural networks. *Applied Optics,* **26,** 4972–4978.

Giles, C. L., Griffin, R. D., & Maxwell, T. (1988). Encoding geometric invariances in higher-order neural networks. In D. Z. Anderson (Ed.), *Neural information processing systems—Natural and synthetic* (pp. 301–309). New York: American Institute of Physics.

Gorse, D., & Taylor, J. G. (1991). A continuous input RAM-based stochastic neural model. *Neural Networks,* **4,** 657–665.

Gorse, D., Taylor, J. G., & Clarkson, T. G. (1993). Learning real-valued functions using a hardware-implementable stochastic reinforcement algorithm. Contribution to *The International Joint Conference on Neural Networks* (IJCNN'93), Nagoya, Japan.

Hecht-Nielsen, R. (1990). *Neurocomputing.* Reading, MA: Addison-Wesley.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks,* **2,** 359–366.

Irie, B., & Miyake, S. (1988). Capabilities of three-layered perceptrons. *IEEE International Conference on Neural Networks I* (pp. 641–648). New York: IEEE Press.

Ishibuchi, H., Fujioka, R., & Tanaka, H. (1992). Possibility and necessity pattern classification using neural networks. *Fuzzy Sets and Systems,* **48,** 331–340.

Lapedes, A., & Farber, R. (1988). How neural nets work. In D. Z. Anderson (Ed.), *Neural information processing systems—Natural and synthetic* (pp. 442–456). New York: American Institute of Physics.

Lenze, B. (1989). On multidimensional Lebesgue–Stieltjes convolution operators. In C. K. Chui, W. Schempp, & K. Zeller, (Eds.), *Multivariate approximation theory IV* (pp. 225–232). ISNM 90, Basel: Birkhäuser Verlag.

Lenze, B. (1992). Constructive multivariate approximation with sigmoidal functions and applications to neural networks. In D. Braess & L. L. Schumaker (Eds.), *Numerical methods of approximation theory* (pp. 155–175). ISNM 105, Basel: Birkhäuser Verlag.

Lenze, B. (1993). Quantitative approximation results for sigma-pi-type neural network operators. In K. Jetter & F. Utreras (Eds.), *Multivariate approximation: From CAGD to wavelets* (pp. 193–209). Singapore: World Scientific.

Light, W. A. (1992). Ridge functions, sigmoidal functions, and neural networks. In E. W. Cheney, C. K. Chui, & L. L. Schumaker (Eds.), *Approximation theory VII* (pp. 163–206). New York: Academic Press.

Minsky, M. L., & Papert, S. (1969). *Perceptrons.* Cambridge, MA: MIT Press.

Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition.* Volumes I, II, Cambridge, MA: MIT Press.

Saarinen, S., Bramley, R., & Cybenko, G. (1991a). Ill-conditioning in neural network training problems. *CSRD Report,* No. **1089.**

Saarinen, S., Bramley, R., & Cybenko, G. (1991b). Neural networks, backpropagation, and automatic differentiation. Preprint.

Saks, S. (1937). *Theory of the integral* (2nd ed.). New York: Hafner Publishing Company.

Schumaker, L. L. (1981). *Spline functions: Basic theory.* New York: John Wiley & Sons.