PERGAMON

Neural Networks 15 (2002) 395–414

Neural Networks

www.elsevier.com/locate/neunet

Contributed article

On the capabilities of neural networks using limited precision weights

Sorin Draghici*

Department of Computer Science, Wayne State University, 431 State Hall, Detroit, MI 48202, USA

Received 8 November 2001; revised 22 February 2002; accepted 22 February 2002

Abstract

This paper analyzes some aspects of the computational power of neural networks using integer weights in a very restricted range. Using limited range integer values opens the road for efficient VLSI implementations because: (i) a limited range for the weights can be translated into reduced storage requirements and (ii) integer computation can be implemented in a more efficient way than the floating point one. The paper concentrates on classification problems and shows that, if the weights are restricted in a drastic way (both range and precision), the existence of a solution is not to be taken for granted anymore. The paper presents an existence result which relates the difficulty of the problem as characterized by the minimum distance between patterns of different classes to the weight range necessary to ensure that a solution exists. This result allows us to calculate a weight range for a given category of problems and be confident that the network has the capability to solve the given problems with integer weights in that range. Worst-case lower bounds are given for the number of entropy bits and weights necessary to solve a given problem. Various practical issues such as the relationship between the information entropy bits and storage bits are also discussed. The approach presented here uses a worst-case analysis. Therefore, the approach tends to overestimate the values obtained for the weight range, the number of bits and the number of weights. The paper also presents some statistical considerations that can be used to give up the absolute confidence of a successful training in exchange for values more appropriate for practical use. The approach presented is also discussed in the context of the VC-complexity. © 2002 Published by Elsevier Science Ltd.

Keywords: Integer weights; VLSI implementation; VC-complexity

1. Introduction

One of the strong arguments in favor of neural networks as a computational paradigm has been the fact that neural networks using real weights are universal approximators (Cybenko, 1989; Hecht-Nielsen, 1987; Hornik, 1991; Hornik, Stinchcombe, & White, 1989, 1990; Sprecher, 1965, 1993; Stinchcombe & White, 1989). Recently, it has even been suggested that neural networks using real weights have super Turing power (Siegelmann, 1995).

Such results have been crucial in building a solid theoretical foundation for neural network techniques. However, the ultimate success of any technique is determined by the success of its practical applications in real-world problems. Unfortunately, the enthusiasm and reassurance generated by important theoretical results as above cannot be extended in a straightforward manner to applications because real numbers are not easily available in the real world. Most software implementations use a finite precision for number representation. Digital hardware implementations use a finite number of bits for weight storage and this translates into a limited precision for the weights. Even analog implementations which are often cited for their ability to implement easily real numbers as analog quantities are limited in their precision by issues like dynamic range, noise, VLSI area and power dissipation problems. Thus, most implementations use rational numbers.

When rational numbers are used, the precision becomes a crucial issue. If a high precision is used, the rational numbers approximate well the real numbers and neural network implementations using such high precision rational approximations are powerful enough for most applications. Unfortunately, the precision of the implementation is directly proportional to its cost. This dependency is most evident for analog implementations. In analog implementations, increasing the precision requires the ability to decrease the smallest quantity which is distinguishable from noise. If values are stored as voltages in a capacitor for instance, a higher precision will require a larger VLSI area dedicated to the capacitor which in turn will be translated into a higher cost of the circuit. Other issues like the technological limits of the fabrication process, the presence of defects, dynamic range limitations, power dissipation problems, etc. contribute to a higher cost for a higher precision implementation. For digital implementations, a higher

* Tel.: +1-313-577-5484.
E-mail address: sod@cs.wayne.edu (S. Draghici).

0893-6080/02/$ - see front matter © 2002 Published by Elsevier Science Ltd.
PII: S0893-6080(02)00032-1

Table 1
The capabilities of neural networks diminish as the precision of the weights is reduced. In this table n is the number of dimensions of the input space

Units	Weights	Capabilities	Example
Infinite	Real	Universal approximation	Cybenko (1989), Hornik et al. (1989) and Stinchcombe and White (1989)
Finite but depending on n	Real	Exact representation	Hecht-Nielsen (1987)
Finite but depending on n	Rational but depending on required precision	Arbitrary approximation	Sprecher (1996, 1997)
Finite	Rational (high precision)	Good approximation for specific problems	Any existing application
Finite	Rational with decreasing precision	Decreasing quality of approximation and convergence	Hollis et al. (1990), Lundin et al. (1996) and Reyneri and Filippi (1991)
Finite	Integers in a limited interval	Not known	This paper

precision requires more memory for storing the weights and more expensive floating point units in order to perform the computations involved. Although the relative low cost of digital technology reduces somehow the importance of this issue for digital implementations, the phenomenon is the same for all types of implementations: a higher precision means a higher cost of the implementation.

It follows from the above that a reduced precision can be translated into a reduced cost of the implementation. It is interesting to consider how the capabilities of the neural networks are affected when the number and precision of its weights are limited. At one extreme we have neural networks using an *unlimited* number of unlimited *real* weights. Such networks have been shown to be universal approximators. Following Kolmogorov's superpositions idea (Kolmogorov, 1957) and a later theorem by Sprecher (1965), Hecht-Nielsen gave in 1987 an existential proof that neural networks using a finite number of units and real weights can represent exactly an arbitrary function. This result has been followed by a series of fundamental theoretical proofs showing that neural networks using real weights are universal approximators in various mathematical frameworks (Cybenko, 1989; Hornik et al., 1989; Stinchcombe & White, 1989). One common feature between these results and their later developments is that they rely on properties of the set $\mathbb{R}$ of real numbers. It is not very clear how these results are affected when real numbers are substituted with rational numbers with a limited precision.

Following Kolmogorov's original approach, Sprecher has presented recently a numerical algorithm for arbitrary precision approximation of functions (Sprecher, 1993, 1996, 1997). This approach combines the generality of a theoretical result with the use of rational weights. However, the required precision is not known a priori since the required weights are obtained in an iterative process that depends on the function to be approximated and the desired degree of accuracy.

On the practical side, there exists a multitude of real-world applications that use a finite number of high-precision rational weights. Their very existence shows that such neural networks offer good approximations that are sufficient in many cases for practical use.

From the examples above it seems that the capabilities of neural networks diminish as more restrictions are placed on the number and the precision of the weights (see Table 1). This is also very intuitive: when moving from real to rational and further on to integer weights, some representational power is lost. However, there is no consensus as to *what* exactly it is lost. Several papers studied the effect of reducing the precision of the weights upon the learning process (Dundar & Rose, 1995; Hollis, Harper, & Paulos, 1990; Lundin, Fiesler, & Moerland, 1996; Moerland & Fiesler, 1997; Reyneri & Filippi, 1991) and many other papers have presented various adaptations aimed at an efficient hardware implementation (Beiu, 2000a,b, 2001c,d; Eberhardt, Duong, & Thakoor, 1989). The aim of this paper is to present some results regarding the capabilities of such neural networks in classification problems.

2. Existing work

Neural networks using threshold gates have become more interesting recently due to the fact that the technology has evolved to the point where it is possible to realize threshold gates having VLSI area and time characteristics comparable with classical logic gates (Graf & Henderson, 1990; Graf, Jackel, Howard, & Straughn, 1986; Graf, Jackel, & Hubbard, 1988; Hirai, 1993; Holler, 1991; Holler, Tam, Castro, & Benson, 1989).

Threshold gates have been studied extensively, starting from the 60s (Dertouzos, 1965; Fischler, 1962; Hu, 1965; Minnick, 1961; Redkin, 1970; Sheng, 1969; Winder, 1962). In principle, threshold gates are computationally more powerful than Boolean gates because Boolean gates can only compute one given function while a threshold logic gate is able to compute up to $2^{\alpha n^2}$ Boolean functions ($1/2 \leq \alpha \leq 1$) when their weights vary (Muroga, 1962, 1965, 1971; Winder, 1962, 1963). Several more recent articles support the same idea (Albrecht, 1992; Beiu, 1994; Beiu & Taylor, 1996; Beiu, Peperstraete, Vadewalle,

& Lauwereins, 1994; Bruck, 1990; Bruck & Goodman, 1990; Bruck & Smolensky, 1992; Mayoraz, 1991; Parberry & Schnitger, 1988; Razborov, 1992; Siu Bruck, 1990, 1991; Siu & Roychowdhury, 1993).

The trade-off between the *size* and the *depth* of a circuit has been discussed in various papers (Beiu, Peperstraete, & Lauewereins, 1992; Impagliazzo, Paturi, & Saks, 1993; Siu & Bruck, 1990; Siu, Roychowdhury, & Kailath, 1991). Numerous complexity results have been obtained by many researchers. One approach is to minimize either the size or the depth or both for circuits using threshold gates (Albrecht, 1995; Alon & Bruck, 1994; Beigel & Tarui, 1991; Chandra, Stockmeyer, & Vishkin, 1984; Hajnal, Maass, Pusl̀ak, Szegedy, & Turan, 1993; Håstad, 1986; Håstad & Goldmann, 1991; Håstad, Wegener, Wurm, & Yi, 1994; Hofmeister, Hohberg, & Köhling, 1991; Siu & Bruck, 1990; Siu & Roychowdhury, 1994; Siu, Roychowdhury, & Kailath, 1990, 1993; Wegener, 1993). The trade-off between the depth and the size was analyzed in several papers (Beiu, 1994; Beiu & Taylor, 1996; Impagliazzo et al., 1993; Siu et al., 1990, 1991). The fan-in issue is addressed (Akers, Walker, Ferry, & Grondin, 1988; Baker & Hammerstrom, 1988; Beiu, 1997; Beiu & Makaruk, 1998; Hammerstrom, 1988; Hong, 1987; Horne & Hush, 1994; Meir & Fontanari, 1993; Raghavan, 1988; Reyneri & Filippi, 1991; Sontag, 1990). The relationship between weights and area and time performances of a chip has been discussed (Alon & Bruck, 1994; Beiu, 1996a, 1998; Goldmann, Håstad, & Razborov, 1992; Goldmann & Karpinski, 1993; Siu & Bruck, 1990).

Various issues relating the precision of the weights to various cost parameters are discussed in several hardware implementation review papers (Draghici, 2000a; König, 1995; Lande, 1997; Sánchez-Sinencio and Newcomb, 1992), chapters (Cauwenberghs, 1998) or books (Fiesler & Beale, 1996; Glessner & Pöchmuller, 1994; Jabri, Coggins, & Flower, 1996; Lande, 1998; Mead, 1989; Morgan, 1990; Sánchez-Sinencio & Lau, 1992). Of particular interest to the topics discussed here are also Parberry (1994) and Siu (1995).

The interest in neural networks using reduced precision weights has manifested itself since the late 80s. The efforts in this direction can be divided into two major trends. The first trend was to modify existing algorithms, adapting them to a reduced precision. Successful examples of such algorithms include the continuous-discrete learning method (Fiesler, Choudry, & Caulfield, 1990) which backpropagates the errors obtained from the discrete network through the continuous network. This technique works well with 5–7 bits/weight and sometimes it can even achieve 2 bits/weight. Another technique in the same category reduces the number of weights in a sequence of training sessions (Nakayama, Inomata, & Takeuchi, 1990). The technique approximates the sigmoid with a linear function and can achieve convergence with 4–7 bits/weight in general and as low as 2 bits, occasionally. Another continuous discrete learning method is presented in Takahashi, Tomita, Kawabata and Kyuma (1991). Here, local minima are avoided using two additional heuristics: (a) all values are restricted to a given interval in the early stages of learning and (b) the number of training epochs is increased for those training signals that cause errors larger than the average. This technique can also achieve a very low precision of up to 1–2 bits/weight.

The second trend is represented by novel techniques oriented towards low precision. Probabilistic rounding algorithms (Höhfeld & Fahlman, 1992; Vincent & Myers, 1992) use a minimal weight update. When a proposed update Δw_{ij} is smaller than this minimal update, the algorithms use the minimal one with a probability proportional to the magnitude of the proposed update Δw_{ij}. The probability is increased with a constant of 1/fan-in (where fan-in is the number of incoming connections) in order to increase the probability of at least one weight change per epoch. A second category groups together algorithms using dynamic rescaling of the weights and adapting the gain of the activation function. The gain can be adapted using the average value of the incoming connections to a neuron (Höhfeld & Fahlman, 1992; Xie & Jabri, 1991) or some form of gradient descent (Coggins & Jabri, 1994; Tang & Kwan, 1993). Another approach to obtaining solutions with reduced weight precision is to treat the quantization error like white noise and use a dynamical quantization step calculated from the error. This particular approach has been used with a cascade-correlation architecture (Duong & Daud, 1999; Duong, Eberhardt, Daud, & Thakoor, 1996). Model-free algorithms use a fixed architecture but no specific classical learning paradigm. Examples include the Tarana chip (Jabri et al., 1996, chapter 8) and the whole approach of weight and node perturbation or stochastic learning (Alspector, Gupta, & Allen, 1989; Cauwenberghs, 1993; Flower & Jabri, 1993; Jabri & Flower, 1992; Widrow & Lehr, 1990). Perturbation algorithms rely on calculating an ad-hoc approximation of the gradient $\partial E/\partial x$ of the error with respect to a specific architectural element x by injecting a perturbation Δx and measuring the error ΔE. Finally, multiplierless algorithms use power-of-two weights which eliminate the need for multiplications in digital implementations.

Such ideas focused on using low precision weights as the means to obtain high performance circuits have materialized recently in combinatorial circuits with low power dissipation and high noise immunity (Beiu, 2001a,b).

A particularly interesting category of theoretical algorithms consists of the algorithms using limited precision integer weights (LPIW). This class of algorithms is interesting because integer weights and integer computation circuitry can be implemented more efficiently in hardware, both in terms of chip space (VLSI area) and cost. If the integer numbers are powers of two, multiplications and divisions reduce to shifts, thus becoming even more attractive for hardware implementation.

The idea of using integer weights has encountered a strong initial resistance. Early experiments have shown that the convergence process is not a simple matter if the weights are truncated to integer values. However, more recent techniques have shown that it is possible to train such neural networks using integers

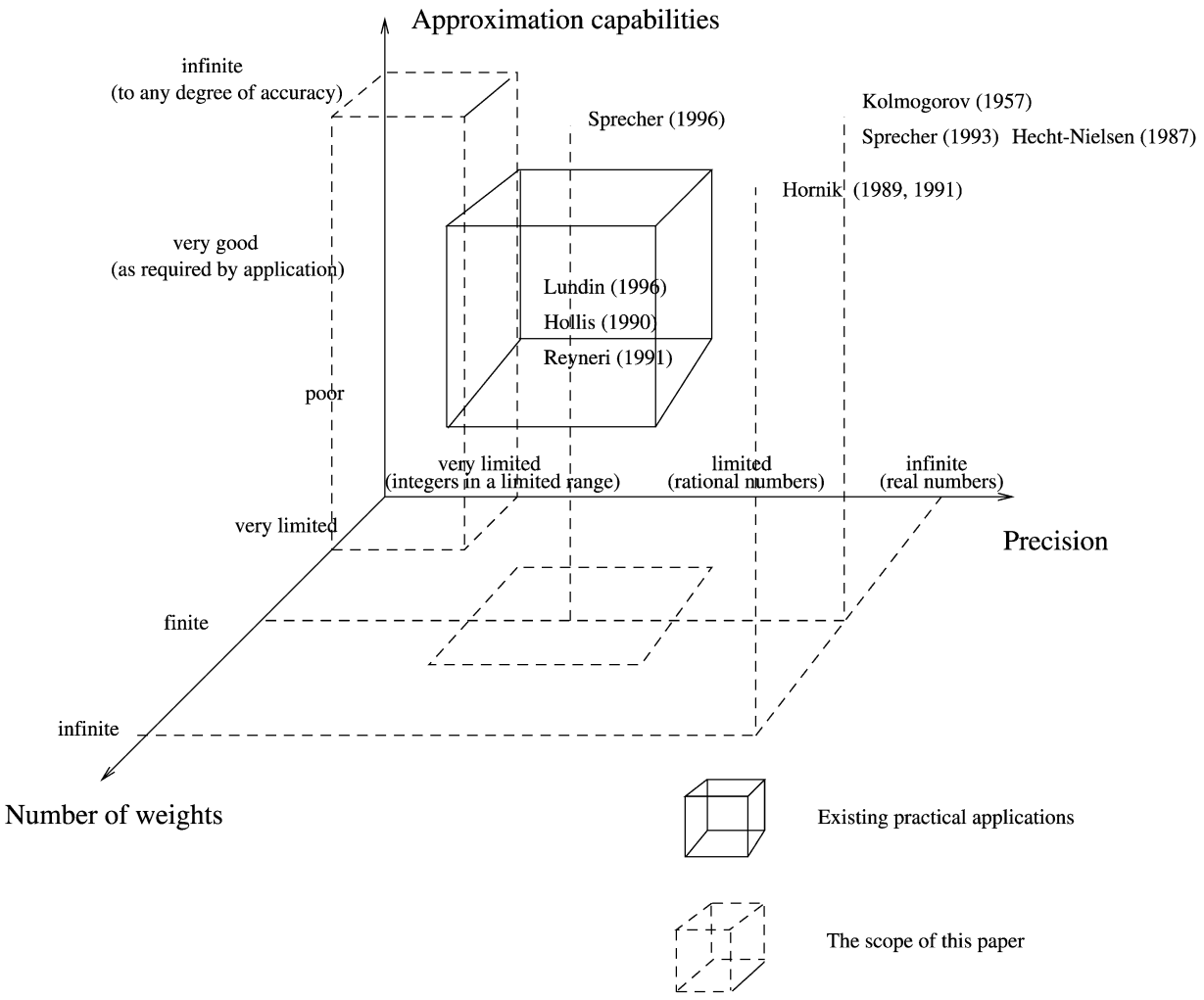

Fig. 1. The scope of this paper. The figure also includes a few landmark results on the capabilities of neural networks.

(Coggins & Jabri, 1994; Höhfeld & Fahlman, 1992; Tang & Kwan, 1993; Vincent & Myers, 1992; Xie & Jabri, 1991) or even powers of two weights (Dundar & Rose, 1995; Hollis & Paulos, 1994; Kwan & Tang, 1992, 1993; Marchesi, Orlandi, Piazza, Pollonara, & Uncini, 1990, Marchesi, Orlandi, Piazza, & Uncini, 1993; Simard & Graf, 1994; Tang & Kwan, 1993). This latter type of weights is important because it eliminates the need for multiplications in binary implementations. Another example of weight restrictions imposed by the type of implementation is the category of neural networks using discrete and positive weights (Fiesler et al., 1990; Moerland, Fiesler, & Saxena, 1998). Such networks are particularly suitable for optical implementations.

While the existing algorithms offer a way of training these weights in order to achieve a desired goal weight state, there are relatively few results that address the problem of whether a goal state exist for a given problem and a given architecture in the context of limited range integer weights.

The scope of this paper is the class of VLSI-friendly neural networks (i.e. networks using LPIW) and their capabilities in solving classification problems (see Fig. 1). This paper will show that: (i) if the range of the weights is not chosen appropriately, a network using hyperplanes and limited precision integer weights will not be able to find a solution regardless of the number of hyperplanes (units) used and (ii) one can calculate a weight range able to guarantee the existence of a solution as a function of the minimum distance between patterns of opposite classes.

The approach used here was inspired by Beiu (1996b). Other issues related to VLSI efficiency in the context of limited precision are discussed in Beiu, Draghici and Makaruk, (1997); Beiu, Draghici and Pauw, (1999). An algorithm based on this analysis is presented in Draghici, Beiu and Sethi, (1997).

3. Definitions and general considerations

A *network* is an acyclic graph which has several input nodes (also called *inputs*) and some (at least one) output

nodes (also called *outputs*). A *neural network* is a network in which each connection is associated with a *weight* (also called synaptic weight) and in which, each *node* (also called *neuron* or *unit*) calculates a function σ of the weighted sum of its m inputs as follows:

$$f(x) = \sigma\left(\sum_{i=0}^{m} w_i x_i + \theta\right)$$

The function σ (usually nonlinear) is called the *activation function* of the neuron, x_i are the input values (or the output values of the previous layer of neurons), w_i are the weights and the value θ is the *threshold*. If all neurons in the network are connected only to neurons on the previous layer (no short-cut connections), the network is said to be *strictly layered*. A neuron that uses a step activation function can be said to implement the hyperplane $\sum_{i=0}^{m} w_i x_i + \theta = 0$ in the m-dimensional space of its inputs because its output will be 1 for $\sum_{i=0}^{m} w_i x_i + \theta > 0$ and 0 for $\sum_{i=0}^{m} w_i x_i + \theta \leq 0$. Therefore, we will say that a two-class classification problem can be solved by a neural network with the weights $w_1, w_2, \ldots, w_k$ if there exists a set of hyperplanes h_j: $\sum_{i=0}^{m} w_{ij} x_i + \theta_j = 0$ with all w_{ij} in the set $\{w_1, w_2, \ldots, w_k\}$ such that the patterns from the two classes are separated by the hyperplanes h_j.

We will also use the following definitions.

Definition. For any two points $a, b \in \mathbb{R}^n$, we define the *closed segment of extremities a and b* (or simply the *segment* $[a, b]$) as the set:

$$[a, b] = \{z_\lambda = (1 - \lambda)\cdot a + \lambda \cdot b \mid \lambda \in [0, 1]\} \subseteq \mathbb{R}^n \quad (1)$$

For the dimensions $n = 1$, 2 and 3, this definition reduces to the usual definition of a segment.

Definition. For any connected set $V \subseteq \mathbb{R}^n$, we define the *simplex distance measure* or *sd-measure* of the set V as:

$$sd(V) = \sup\{\|a - b\| \, a, b \in \mathbb{R}^n, [a, b] \subset V\} \quad (2)$$

where $\|\cdot\|$ is the Euclidean norm.

In other words, the *sd*-measure associates to any connected set V, the largest distance between two points which are contained in the given set V such that the whole segment between the two points is also included in V. This distance can be used for connected sets.[1] However, if one uses only hyperplanes, any resulting volume will be convex (as a finite intersection of half-space which are convex sets). For convex sets, the *sd*-measure reduces to the maximum distance between any two points.

[1] A set S is connected if any two points can be connected with a polygonal line included in S.

Definition. Let V_1 and V_2 be two connected subsets of $\mathbb{R}^n$. We say that V_1 is *sd-larger* than V_2 if $sd(V_1) > sd(V_2)$. V_1 is *sd-smaller* than V_2 if V_2 is *sd-larger* than V_1.

In the following, we will use 'larger' and 'smaller' in the sense of *sd*-larger and *sd*-smaller when dealing with volumes in $\mathbb{R}^n$.

Definition. A *simplex* in n-dimensions is an n-polytope which is the convex hull of $n + 1$ affinely independent points.

The simplex is the simplest geometrical figure consisting (in n-dimensions) of $n + 1$ points (or vertices) and all their interconnecting line segments, polygonal faces, etc. In 2D, a simplex is a triangle. In 3D, a simplex is a tetrahedron (not necessarily the regular tetrahedron).

4. General considerations

We will study the classification capabilities of neural networks using integer weights in a given range. Without loss of generality, we will concentrate on a two-class classification problem. From our point of view the particular (strictly layered) architecture used as well as the training algorithm chosen are not relevant, as long as a solution exists. We would like to study the loss of capabilities brought by limiting the weights to integer values. If the weights are real numbers, the hyperplanes implemented with these weights can be placed in any positions by the training algorithm. In this situation, the only difficulty faced by the training is the convergence of the training algorithm. Eventually, if the training algorithm is appropriate, a solution weight state will be found so that all patterns from opposite classes are separated. If the precision of the weights is finite but sufficiently high (e.g. a software simulation), the positions available to the hyperplanes implemented by the network are sufficiently dense in the problem space and the availability of a hyperplane is not an issue. However, if the weight values are restricted to very few integer values, the positions available to the network for placing its hyperplanes during the training are very sparse in the problem space (see Fig. 2). If a given classification problem includes patterns from opposite classes that are situated in positions such that no hyperplane is available for their separation, the training will fail regardless of the training algorithm or the number of units implementing hyperplanes. The training algorithm is irrelevant because no matter what algorithm is used, the search will be performed in the finite and limited space of available hyperplanes. If no available hyperplane is able to separate the given patterns, the way the search is conducted becomes irrelevant. Also, the number of units is not important in this context because if the weight values are restricted, each individual neuron can only implement a particular hyperplane from the set of those available. Training with more units will mean

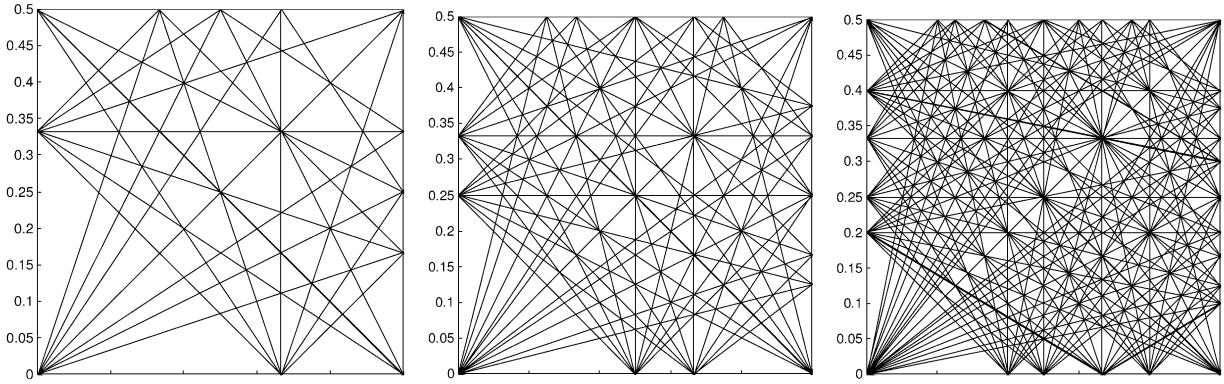

Fig. 2. The hyperplanes that can be implemented with integer weights in the ranges [−3,3], [−4,4] and [−5,5], respectively, drawn in the square [0,0.5].

more hyperplanes but, if none of these hyperplanes is able to separate the given patterns, the problem will not be solved.

If we restrict ourselves to 2D, the set of available hyperplanes can be imagined as a mesh in the problem space (see Fig. 2). If the weights are continuous or can be stored with a reasonable precision, this mesh is very dense, like a fabric. If the weights are restricted to integer weights in a very restricted range, the mesh becomes very rare with large interstices in-between hyperplanes. If, for a given classification problem and weight range there are two patterns from different classes which happen to be in such an interstice, the training will fail regardless on how the hyperplanes are chosen (the training algorithm) or how many such hyperplanes are used. Fortunately, one can establish a simple relationship between the difficulty of the classification problem as described by the minimum distance between patterns of opposite classes and the precision necessary for the weights. We will show that for any two patterns of opposite classes contained in a reference hypercube and situated at a distance of at least $\sqrt{n}/2p$ of each other, there exists a hyperplane that separates the given patterns and that can be implemented with integer weights in the range $[-p, p]$. In order to do this, we will consider all hyperplanes that can be implemented with weights in the given range. Some of these hyperplanes will cut the hypercube considered into simplexes. We will show that the longest distance between two points internal to such simplexes is smaller than or equal to $\sqrt{n}/2p$. From this, it will follow that any two patterns situated at a distance larger than $\sqrt{n}/2p$ will be situated in different regions. In other words, there will be at least one hyperplane separating them.

5. Main results

5.1. Theoretical limits

We shall start by re-stating our goal: we want to be able to relate the difficulty of a classification problem as described by some measure to the weight range necessary to guarantee the existence of a solution for a given classification problem. In other words, given a classification problem, we want to calculate a weight range that guarantees the existence of a solution for an appropriately chosen architecture.

Let us consider the intersection of the hyperplanes of the form

$$\sum_{i=0}^{n} w_i x_i = 0 \tag{3}$$

where $x_0 = 1$ and $w_i \in \mathbb{Z} \cap [-p, p]$ with the hypercube $[-0.5, 0.5]^n \subseteq \mathbb{R}^n$.

Before we can present the main results, we need to prove two lemmas. The following result holds true and justifies the name chosen for the measure we are using.

Lemma 1. *The sd-measure of a simplex equals the length of the longest edge.*

Proof. A simplex is convex and closed. Let l be the *sd*-measure of the simplex S. From the definition of the *sd*-measure, it follows that $\exists x_i, x_j \in S$ such that $[x_i, x_j] \subseteq S$, $\|x_i - x_j\| = l$ and $\forall a, b \in S$ such that $[a, b] \subseteq S$, $\|x_i - x_j\| \geq \|a - b\|$. In other words, there exist two points x_i and x_j belonging to S such that the segment $[x_i, x_j]$ is the longest segment included in S and $\|x_i - x_j\| = l$.

We will assume that x_i and x_j are internal points of S, i.e. they do not belong to any facet. We will show that this assumption leads to a contradiction.

Let d be the straight line determined by x_i and x_j which are internal to S. Let x_i' and x_j' be the intersections of d with the facets of the simplex. Since x_i' and x_j' belong to S and S is convex (by definition), the whole segment $[x_i', x_j']$ will be included in S (see Fig. 3 for examples in 2D and 3D). By construction x_i and x_j are internal points to the segment $[x_i', x_j']$. Therefore, $\|x_i' - x_j'\| > \|x_i - x_j\|$ which contradicts our assumption that the segment $[x_i, x_j]$ was the longest segment in S. Therefore, at least one of x_i and x_j must be on at least one facet.

Without loss of generality, we assume x_i is on at least one facet. We repeat the previous argument extending the segment $[x_i, x_j]$ along d in the direction of x_j until we inter-

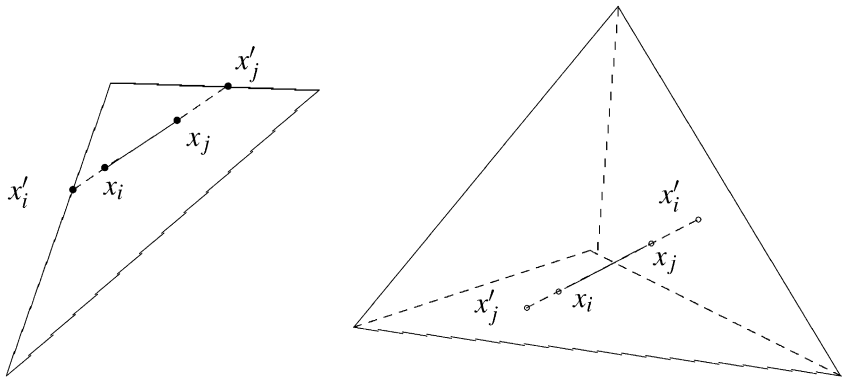

Fig. 3. If we assume that x_i and x_j are internal points (not on any facet), we can construct the segment $[x'_i, x'_j]$ which will be included in the given simplex and longer than $[x_i, x_j]$ which contradicts the hypothesis.

sect a facet. Let x'_j be the intersection point. Again, the segment $[x_i, x'_j]$ is included in S and is longer than $[x_i, x_j]$ which contradicts the hypothesis. Therefore, both x_i and x_j must be on at least one facet. We will show that, in fact, they are vertices.

Let F_j be the facet containing x_j and $x_1, x_2, \ldots, x_n$ the vertices of F_j. Let us assume that x_j is not a vertex of F_j.

We construct the hyperplane H that passes through x_j and is perpendicular on $\overrightarrow{x_i x_j}$ (see Fig. 4). This hyperplane divides $\mathbb{R}^n$ in two half-spaces. Let H_{x_i} be the half-space that contains x_i and $\overline{H_{x_i}}$ the half-space that does not contain x_i.

We assumed x_j was not a vertex. There are two other possibilities: either x_j is on a one-dimensional (1D) edge or x_j is an interior point of F_j. If x_j is an interior point of F_j, it follows that $\overline{H_{x_i}} \cap F_j \neq \Phi$, i.e. there are points belonging to F_j in the opposite half-space of H with respect with x_i. Then $\overline{H_{x_i}}$ will contain at least a vertex x_k of F_j. Since $\overrightarrow{x_i x_j} \perp H$ and x_k is in $\overline{H_{x_i}}$, i.e. on the other side of H with respect to x_i, the segment $[x_i x_k]$ will intersect the hyperplane H. Let p be that intersection point (see Fig. 4). The points x_i, x_j and p form a triangle in the 2D plane that they define. In that triangle, the $\vec{j}_x i_x$ is perpendicular on $\vec{p} j_x$ and therefore $\|x_i x_j\|$ is smaller than $\|x_i p\|$ which contradicts our hypothesis that $\|x_i x_j\|$ is the largest internal distance in S.

If x_j is on a 1D edge, we can distinguish another two cases: this 1D edge is completely included in the hyperplane H or it just intersects it in one point. In the latter case, it follows that there are points on this edge belonging to $\overline{H_{x_i}}$ (the opposite half-space of H with respect to x_i). Following the reasoning in the case above, we arrive at the same contradiction.

Finally, if the 1D edge that includes x_j is included in the

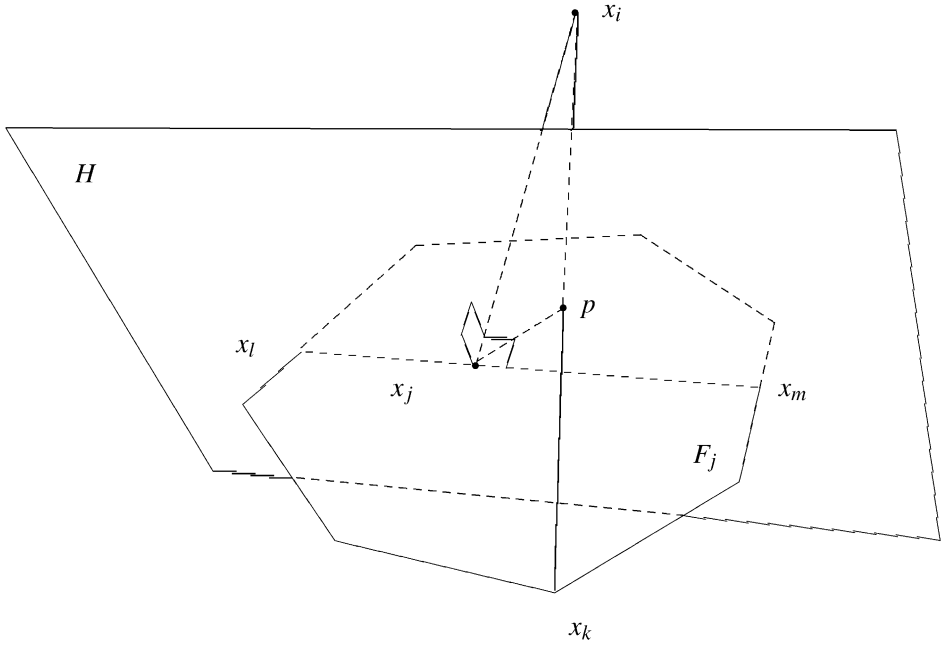

Fig. 4. For any facet F_j and exterior point x_i we construct the hyperplane that passes through x_j and is perpendicular on $x_i x_j$. For any vertex x_k situated either on H or in the opposite half space of H with respect to x_i, the distance $\|x_i x_k\|$ will be larger than $\|x_i x_j\|$.

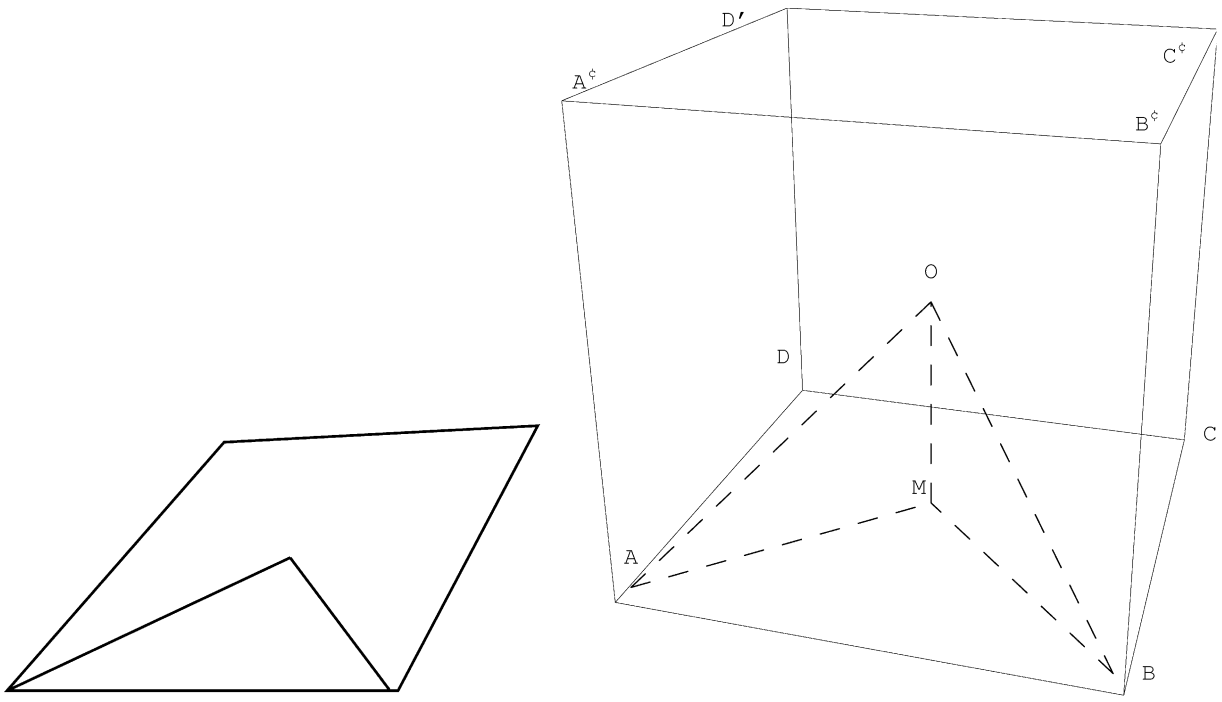

Fig. 5. Two simplexes that can be rotated and translated to induce a shattering of the 2D square and 3D cube, respectively.

hyperplane H, then H will also include the vertices situated at the extremities of this edge. Let x_l and x_m be these vertices. Since x_l and x_m belong to H and $\overrightarrow{x_i x_j} \perp H$, both $\|x_i x_l\|$ and $\|x_i x_m\|$ will be larger than $\|x_i x_j\|$ which again contradicts our hypothesis.

Therefore, x_j must be a vertex. Swapping the indexes i and j and considering the same argument, x_i must be a vertex. Since both x_i and x_j are vertices and a simplex has a (1D) edge between any two vertices, the segment $[x_i, x_j]$ is an edge. Since we know already that $[x_i, x_j]$ is the longest segment in the whole of S, our result is proved.

Lemma 2. *Consider an n-dimensional hypercube H of side length l. There exists an m and a set of simplexes* $S_1, S_2, \ldots, S_m$ *with the properties that*

$$S_1 \cup S_2 \cup \cdots \cup S_m = H \tag{4}$$

such that:

$$sd(S_i) \leq \max\left\{l, \frac{\sqrt{n}}{2} l\right\}, \qquad 1 \leq i \leq m \tag{5}$$

Proof. We will prove the results by constructing a set of simplexes that satisfy the properties above. Without loss of generality we will consider the hypercube H of side l with a vertex in the origin and all other vertices in the positive half-axes of each dimension. Fig. 5 presents two simplexes that can be rotated and translated to induce a shattering of the 2D square and 3D cube, respectively. For the n-dimensional case, we can construct a simplex using two vertices of the hypercube and the $n-1$ centers of $2, 3, \ldots, n$ dimensional hypercubes containing the chosen vertices. Let us choose for instance the vertices

$$\begin{aligned} &1 : (l, 0, \ldots, 0) \\ &2 : (0, 0, \ldots, 0) \end{aligned} \tag{6}$$

The centers of the lower dimensional hypercubes containing these points are:

$$\begin{aligned} &3 : (l/2, l/2, 0, \ldots, 0) \\ &\vdots \\ &n+1 : (l/2, l/2, l/2, \ldots, l/2) \end{aligned} \tag{7}$$

Point 3 in Eq. (7) is the center of one of the 2D squares containing points 1 and 2. Point $n+1$ in Eq. (7) is the center of the whole nD hypercube. According to Lemma 1, the *sd*-distance of a simplex is the length of the longest edge. The distances between the vertices of this simplex are as follows:

$$\rho(1,2) = l$$

$$\rho(1,i) = \frac{l}{2}\sqrt{i-1} \leq \frac{l}{2}\sqrt{n}, \quad i = \overline{3, n+1},$$

$$\rho(2,i) = \frac{l}{2}\sqrt{i-1} \leq \frac{l}{2}\sqrt{n}, \quad i = \overline{3, n+1},$$

$$\rho(i,j) = \frac{l}{2}\sqrt{|i-j|} \leq \frac{l}{2}\sqrt{n-2}, \quad i,j = \overline{3, n+1}$$

Similar simplexes can be obtained in the same way by choosing different initial vertices of the hypercube or by

applying rotations and translations (which are isometries) to the one obtained above.

In order to show the second part of the lemma, we shall consider an arbitrary point A belonging to the hypercube H and we shall construct the simplex that contains the point A.

We note that the affine spaces of the facets of the simplexes considered above are hyperplanes of the form $x_i - x_j = 0$, $x_i + x_j - l = 0$ or affine spaces of facets of the hypercube. Let S_H be the set of all such hyperplanes. All these hyperplanes either include the 1D edges of the simplexes above or do not intersect them at all. Therefore, the intersection of the half-spaces determined by such hyperplanes is either a general (infinite) half-space intersection or a simplex as above. Each such hyperplane divides the space into two half-spaces. In order to find the simplex that contains the point A, we simple select for each such hyperplane in S_H, the half-space that contains the point A. The intersection of all half-spaces including A will be the desired simplex.

Proposition 1. *A strictly layered neural network using integer weights in the range* $[-p, p]$ *can classify correctly any set of patterns included in a hypercube of side length* $2l$, $l < 0.5$ *centered around the origin of* $\mathbb{R}^n$, $n \geq 2$, *for which the minimum Euclidean distance between two patterns of opposite classes is* $d_{\min} \geq l_0 = \max\{l, \sqrt{n}/2p\}$.

Proof. We will point out the fact that the restriction to a cube of side length equal to one does not affect the generality of the result. If the patterns are spread outside the unit hypercube, the problem can be scaled down with a proper modification of $d_{\min}$. Fig. 2 presents the set of hyperplanes (in 2D) which can be implemented with weights in the set $\{-3,-2,-1,0,1,2,3\}$ $(p=3)$, $\{-4,-3,\ldots,3,4\}$ $(p=4)$ and $\{-5,-4,\ldots,4,5\}$ $(p=5)$, respectively. The analysis can be restricted to the quadrant $[0, 0.5]^n$ without loss of generality because of symmetry.

The proof will be conducted along the following lines. We shall consider the cube of side $l = 1/p$ with a vertex in the origin and located in the positive half of each axis. We shall consider a subset of n-dimensional simplexes that can be constructed by cutting this hypercube with hyperplanes with integer coefficients not less than p. At this point, we shall show that the maximum internal distance in all such simplexes is less or equal to $d_{\min}$.

Finally, we shall show that for any two points situated at a distance of at least $d_{\min} = l_0$ from each other, one can find a hyperplane that separates them and that this hyperplane can be implemented with integer weights in the given range.

In order to help the reader follow the argument presented, we will use the example of the 3D case presented in Fig. 5. However, the arguments presented will remain general and all computations will be performed in n-dimensions.

We shall consider the hypercube H of side $l = 1/p$ with a vertex in the origin and situated in the positive half-space of each axis. From Lemma 2 we know that the hypercube H can be shattered in simplexes with the sd-measure l_0. As shown above, the longest edge (1D subspace) of such simplexes has the length l_0. Such edges connect vertices of the hypercube with the center of the hypercube.

Now, we shall show that all hyperplanes involved need only integer coefficients in the given range. Let us consider again the vertices of the simplex defined in Lemma 2:

$$\begin{aligned} &1 : (l, 0, 0, \ldots, 0) \\ &2 : (0, 0, 0, \ldots, 0) \\ &3 : (l/2, l/2, 0, \ldots, 0) \\ &\quad\vdots \\ &n+1 : (l/2, l/2, l/2, \ldots, l/2) \end{aligned} \tag{8}$$

The affine spaces of the facets of this simplex are the hyperplanes determined by all combinations of n points from the $n + 1$ points above, i.e. eliminating one point at a time for each hyperplane. The first hyperplane will be determined by the points remaining after the elimination of the first point, etc. The equation of the first hyperplane can be written as:

$$\begin{vmatrix} x_1 & x_2 & x_3 & \cdots & x_n & 1 \\ 0 & 0 & 0 & & 0 & 1 \\ l/2 & l/2 & 0 & & 0 & 1 \\ \vdots & & & & & \\ l/2 & l/2 & l/2 & & l/2 & 1 \end{vmatrix} = 0 \tag{9}$$

which can be re-written as:

$$x_1 - x_2 = 0 \tag{10}$$

The equation of the second hyperplane can be obtained by considering all but the second point in Eq. (8), as follows:

$$\begin{vmatrix} x_1 & x_2 & x_3 & \cdots & x_n & 1 \\ l & 0 & 0 & & 0 & 1 \\ l/2 & l/2 & 0 & & 0 & 1 \\ \vdots & & & & & \\ l/2 & l/2 & l/2 & & l/2 & 1 \end{vmatrix} = 0 \tag{11}$$

which can be re-written as: $x_1 + x_2 - l = 0$. By substituting $l = 1/p$, this equation becomes:

$$px_1 + px_2 - 1 = 0 \tag{12}$$

The equations of all remaining hyperplanes can be obtained by considering all but the i-th point in Eq. (8) which leads to

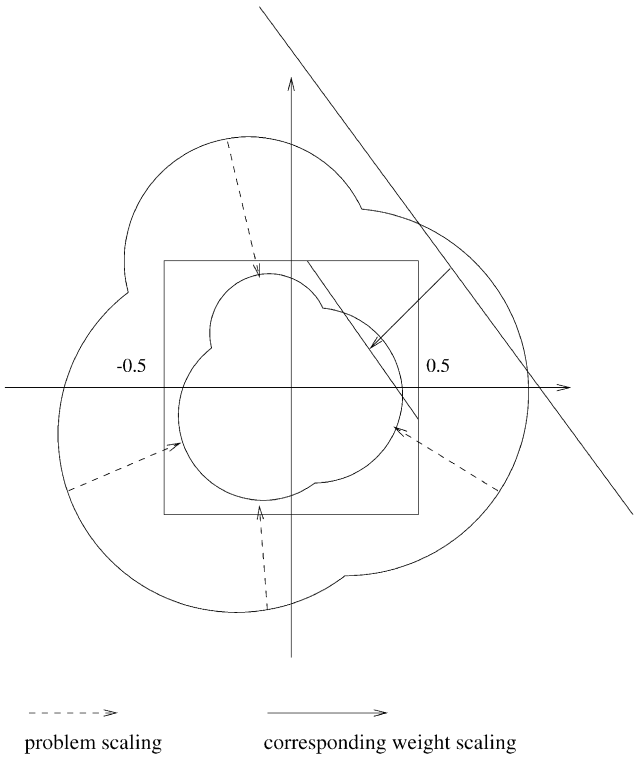

Fig. 6. The necessary weight range for an arbitrary problem can be calculated by scaling the problem to the $[-1,1]^n$ hypercube, calculating the necessary range and then restoring the problem to its initial domain and adjusting the weight range accordingly.

the generic equation:

$$\begin{vmatrix} x_1 & x_2 & x_3 & \cdots & x_{i-1} & x_i & x_{i+1} & \cdots & x_n & 1 \\ l & 0 & 0 & & & & & & 0 & 1 \\ 0 & 0 & 0 & & & & & & 0 & 1 \\ l/2 & l/2 & 0 & & & & & & 0 & 1 \\ \vdots & & & & & & & & & \\ l/2 & l/2 & l/2 & \cdots & l/2 & 0 & 0 & \cdots & 0 & 1 \\ l/2 & l/2 & l/2 & & l/2 & l/2 & l/2 & \cdots & 0 & 1 \\ \vdots & & & & & & & & & \\ l/2 & l/2 & l/2 & & l/2 & l/2 & l/2 & \cdots & l/2 & 1 \end{vmatrix} = 0 \tag{13}$$

that can be re-written as:

$$x_{i+1} - x_i = 0 \tag{14}$$

Since Eqs. (10), (12) and (14) can be written using only coefficients in the given range, we can conclude that a neural network can implement any and all such hyperplanes.

Now, let us consider two random points a and b in the hypercube $[0, 1/p]^n$. Considering the shattering defined by the simplexes above, there are two possibilities: either a and b fall in the same simplex or they fall in different simplexes. If they fall in the same simplex, the length of the segment $[a, b]$ is less than l_0 because the *sd*-measure of these simplexes is l_0. If they fall in different simplexes, the segment $[a, b]$ will intersect at least one simplex facet. The affine space of that facet can be chosen as the hyperplane to separate the patterns.

This concludes the proof for the hypercubes of side $l = 1/p$ with a vertex in the origin. Now we need to show that the same result holds for the larger hypercube (of side length 0.5). In order to do this, we shall consider the affine transformations of the form:

$$x_i' = x_i + \frac{k}{p} \qquad \text{where } k \in \mathbb{Z} \cap \left[-\left\lceil \frac{p}{2} \right\rceil, \left\lceil \frac{p}{2} \right\rceil \right] \tag{15}$$

Such translations applied to the hyperplanes used in Eqs. (10), (12) and (14) effectively move the hypercube of side $l = 1/p$ with steps of length l until they cover the whole hypercube of side 0.5. The interval for the value k was calculated such that $k/p \geq 0.5$ for the largest positive value of k. Note that all hyperplanes translated in this way will still have all coefficients integers in the given interval.

The proof is completed by observing that if the first layer of the network implements hyperplanes such that the classes are separated, it is always possible for subsequent layers to combine the outputs of these neurons in order to construct the final solution (e.g. by implementing a layer of AND units and a layer of OR units).

It can be noted that the result is general since any set of patterns can be scaled to fit in the given domain. Apparently, this would lead to the conclusion that any network can learn any problem within a given weight range. However, this is not the case because when the input patterns are scaled, the weights also have to be scaled and this maintains the validity of the result across different scales. The necessary weight range for an arbitrary problem can be calculated by scaling the problem to the $[-1,1]^n$ hypercube, calculating the necessary range and then restoring the problem to its initial domain and adjusting the weight range accordingly (see Fig. 6).

Networks with arbitrary topology are at least as powerful as strictly layered networks since strictly layered nets can be obtained from an arbitrarily connected network trivially, by setting the inter-layer weights to zero. Therefore, the results will still hold for arbitrarily connected networks. This means that at the very least, such networks are able to implement solutions similar to those used in the proof of Lemma 2. However, it may be the case that such networks can find more efficient solutions.

Proposition 2. *Let us consider a set of m patterns (in general positions) from two classes in the hypercube of side 1 centered in the origin of* $\mathbb{R}^n$. *Let d and D be the minimum and maximum distance between patterns of opposite classes, respectively. The number of information entropy bits necessary in the worst-case for the correct classification of the patterns using weights in the set* $\{-p, -p+1, \ldots, 0, \ldots, p\}$

where $p = \lceil \sqrt{n}/2d \rceil$ *is lower bounded by*:

$$N_H = \left\lceil m \cdot n \cdot \log \frac{4pD}{\sqrt{n}} \right\rceil \tag{16}$$

Proof. From Proposition 1, it follows that a solution can be constructed if the minimum distance between patterns of opposite classes is $d > \sqrt{n}/(2p)$.

The hyperplanes will slice the space in a number of volumes V_i. One can calculate the number of bits necessary for the representation of one example p_i as:

$$\text{nbits}_{p_i} = \left\lceil \log_2 \frac{V_{\text{total}}}{V_i} \right\rceil \tag{17}$$

As shown in Proposition 1, the largest *sd*-measure of these volumes V_i will be d. The volume of all such individual volumes will be less than the volume of a sphere of diameter d. Therefore, we can write:

$$\text{nbits}_{p_i} = \left\lceil \log_2 \frac{V_{\text{total}}}{V_i} \right\rceil > \left\lceil \log_2 \frac{V_{\text{total}}}{V_d} \right\rceil \tag{18}$$

where V_d is the volume of the sphere of diameter d:

$$V_d = \frac{\pi^{n/2}}{\Gamma(n/2+1)} \cdot \left(\frac{d}{2}\right)^n = \alpha(n) \cdot \left(\frac{d}{2}\right)^n \tag{19}$$

It can be shown that, if D is the largest distance between patterns of opposite classes, then there exists a sphere of radius D that will include all patterns (Beiu & Draghici, 1997). With this, we can calculate a worst-case lower bound on the number of bits necessary to separate the patterns:

$$\text{nbits}_{p_i} > \left\lceil \log_2 \frac{V_D}{V_d} \right\rceil = \left\lceil \log_2 \frac{\alpha(n) D^n}{\alpha(n)(d/2)^n} \right\rceil = \left\lceil n \cdot \log_2 \frac{2D}{d} \right\rceil \tag{20}$$

$$\text{nbits}_{p_i} > \left\lceil n \cdot \log_2 \frac{2D}{\sqrt{n}/2p} \right\rceil = \left\lceil n \cdot \log_2 \frac{4pD}{\sqrt{n}} \right\rceil \tag{21}$$

In the worst-case scenario, each pattern will be classified in its own volume and there is no better representation for the solution than simply putting together the networks carving up the individual volumes as above. In this case, the number of bits for the whole problem can be obtained by multiplying the expression above corresponding to an individual pattern by the number of pattern m to obtain the desired bound and complete the proof:

$$N_H = \left\lceil m \cdot n \cdot \log \frac{4pD}{\sqrt{n}} \right\rceil \tag{22}$$

A similar result can be derived using the unit hypercube as the whole classification space and uniform hypercubes of side $1/2p$ as classification cells.

In this expression and throughout the rest of the paper, all logarithms are in base 2. Furthermore, we will drop the ceiling function keeping in mind that whenever discrete quantities are involved, the values should be rounded up to the nearest integer.

It is interesting to see how these results fit with other known results. For instance, let us consider the set of patterns formed by the vertices of the hypercube $[-1/2, 1/2]^n$. If we shift this hypercube by adding 1/2, we obtain the unit cube whose vertices are the various possible combinations of the n variables. Thus, an arbitrary labelling of the patterns will correspond to a boolean function of n variables. In this case, the minimum distance between patterns will be $d = 1$, the maximum distance between any two patterns will be $D = \sqrt{n}$ and the number of patterns will be 2^n. Proposition 2 gives in this scenario (see Eq. (20)):

$$N_H = \lceil 2^n \cdot n \cdot \log 2\sqrt{n} \rceil = \text{O}(2^n \cdot n \cdot \log n) \tag{23}$$

The DNF or CNF of a boolean function will have in general 2^n terms, each with n literals. It is known that DNFs can be implemented by neural networks using constant weights and therefore, the complexity of such an implementation will be $\text{O}(2^n n)$. This shows that such an optimal DNF implementation is more efficient than the implementation considered by Proposition 2 by a factor of $\log n$.

5.2. Further considerations

Propositions 1 and 2 offer the theoretical base for designing a network in accordance with the necessities of a given problem and using integer weights in a limited range. The weight range can be calculated easily after finding the minimum distance between patterns of opposite classes d. Subsequently, one can calculate lower bounds for the number of bits that the network will need in order to solve the problem. The maximum distance between patterns of opposite classes can be used as the radius D of the smallest sphere containing all patterns.

Using the lower bound on the number of bits calculated as above and the number of bits used to store one weight (which is $\lceil \log(2p+1) \rceil$ for integer weights in the range $[-p, p]$), one can calculate a lower bound on the necessary number of weights:

$$w \geq \frac{N_H}{\log(2p+1)} = \frac{N_H}{\log(2\sqrt{n}/2d+1)} = \frac{N_H}{log(\sqrt{n}/d+1)} \tag{24}$$

A network such designed will offer the guaranties that: (i) there exists a weight state with weights in the given range that solves the problem and (ii) the architecture is not undersized for the given problem.

Two important observation are needed here. Firstly, the number of bits which is calculated by the formula given in Proposition 2 refers to bits in the informational entropy sense and these bits do not correspond directly to the bits used to store the weights because the neural codification is

less than optimal. One should consider an implementation factor k, which would characterize the efficiency of the implementation. In other words, k would be the average number of storage bits corresponding to one bit in the informational entropy sense. In principle, this implementation efficiency would be the number of storage bits needed to implement a network able to make a binary decision in the context of the given problem. Thus:

$$k = \frac{N_S}{N_H} \tag{25}$$

where N_S is the number of storage bits and N_H is the number of bits in the information entropy sense. The number of weights of the network could then be chosen according to:

$$w \geq \left\lceil \frac{k \cdot N_H}{\log(2p+1)} \right\rceil \tag{26}$$

$$w \geq \left\lceil \frac{k \cdot N_H}{\log(\sqrt{n}/d+1)} \right\rceil \tag{27}$$

For input patterns in arbitrary positions, a neural network cannot implement a binary decision using less than one hyperplane. A neuron implementing a hyperplane in an n-dimensional space needs $n+1$ weights. If the weights are integers in the given range, a weight will need $\log(2p+1)$ storage bits. Therefore, the minimum number of storage bits necessary to implement a binary decision in the general case will be $k = (n+1)\log(2p+1)$. This value can be interpreted as the worst-case implementation efficiency. As the network scales up, this interpretation may become less accurate because most networks will need a second and perhaps a third layer in order to combine the outputs of the units in the first layer. These units will tend to decrease the overall efficiency. However, as the problem scales up, it is very probable that several patterns will be classified together by the same hyperplane or group of hyperplanes and this will tend to increase the overall efficiency.

Taking into consideration the worst-case implementation efficiency, the lower bound on the number of weights becomes:

$$w \geq \frac{(n+1)\cdot\log(2p+1)\cdot N_H}{\log(2p+1)} = (n+1)\cdot N_H \tag{28}$$

Recently, it has been shown that using neurons with small, constant fan-in[2] is VLSI optimal in the AT^2 sense (see Beiu et al. (1997) and references therein). If we now consider this particular case in which all neurons have the same fan-in, we can extract the number of neurons as:

$$\text{neurons} \geq \frac{w}{\text{fan-in}} \tag{29}$$

The second observation is that the above lower bound was calculated in the worst-case scenario which assumes that the network will need to isolate each pattern in their own classification region. In practice, this happens only very seldom. In most cases, a classification region will contain several patterns belonging to the same class. In consequence, the number of bits in the information entropy sense, must be corrected by a factor nl which will take into consideration the average number of patterns of the same class grouped together in the same classification region:

$$N_{Hnl} = \frac{N_H}{nl} \tag{30}$$

This factor is highly dependent on the problem and, intuitively, it is inversely proportional with the degree of nonlinearity of the problem. At one extreme, there is the class of linearly separable problems. If the weights are real valued, i.e. the dividing hyperplanes can take any positions, a linearly separable problem can be solved by grouping all patterns of the same class in just one classification region and the nonlinearity factor nl will be maximum for the given problem and equal to the number of patterns in the least numerous class (since in a two-class problem, we only need to separate one class from the other). At the other extreme, one can imagine an infinite n-dimensional hypergrid in which adjacent patterns always belong to opposite classes. Such problem is representative for the worst-case situation considered in Proposition 2 in which each pattern will be classified in its own classification region. In this worst-case scenario, nl will be at its minimum value of 1. In general, one could consider an average value of nl which will represent the average number of patterns classified in the same region. One can imagine various ways of extracting approximate values of nl from the pattern set but this computation is outside the scope of the present paper.

A last observation is due on the number of patterns m considered in calculating the number of entropy bits in Proposition 2. In theory, one should consider the minimum number of patterns in the two classes. This is because in a two class problem as considered here, it is sufficient to separate just the class with fewer patterns. However, since one cannot know in advance which class is going to be used by the training algorithm (or because the training algorithm uses both), it is more appropriate to take the number of patterns equal to the average of the number of patterns in the two classes $m/2$.

5.3. A statistical view

One can trade-in the absolute guarantee regarding the existence of the solution in exchange for a better approximation for practical use. In other words, an alternative bound can be calculated that will only work on average (as opposed to always) but can be calculated more easily from the given patterns. The worst-case considered calculated the necessary weight range based on the minimum distance between patterns of opposite classes. The approximation proposed here is based on the assumption that the

[2] The fan-in of a neuron is the number of its inputs or incoming connections. For a fully connected, strictly layered architecture, this number will be equal to the number of neurons on the previous layer.

data set is such that the distance between patterns of opposite classes follows a Gaussian distribution. If this assumption holds, most pairs of patterns will be separated by a distance situated in an interval centered in the average distance and with a width equal to approximately 3σ where σ is the standard deviation. Therefore, most patterns from opposite classes will be separated by at least $\bar{d} = \bar{d} - 1.5\sigma$ where $\bar{d}$ is the mean distance between patterns of opposite classes.

Such an approximation is made even more plausible by the observation that a pair of very close patterns of opposite classes will cause a training failure only if they are situated in particular positions which cannot be separated by the hyperplanes available. Two patterns can be even infinitely close to one another if they are positioned on different sides of at least one available hyperplane. It is reasonable to believe that there are relatively few such particularly unfavorable positions. Therefore, the new limit can be taken as:

$$w \geq \left\lceil (n+1)\cdot m\cdot n\cdot \log \frac{2D}{d} \right\rceil$$

$$\approx \lceil (n+1)\cdot m\cdot n[1 + \log D - \log(\bar{d} - 1.5\sigma)] \rceil \qquad (31)$$

Keeping these observations in mind, let us see how such results could be used. If one could indeed customize the size of the network to the given problem, one could apply the training algorithm of one's choice being confident that: (i) the given network has the possibility of solving the problem (i.e. there exists a solution weight state for the given configuration) and (ii) the chosen configuration is not wasteful and can be translated into a VLSI chip with a cost close to the minimum possible (for the given problem).

6. Experiments

This section of the paper is aimed at investigating how well the results given by these theoretical instruments correspond to the practical reality of training neural networks. This can be done by comparing the theoretical bounds obtained using Eq. (31) with the number of weights effectively used by trained networks in a few indicative experiments.

6.1. Experimental setup

Experiments were run using both classical benchmark problems (the exclusive-or XOR and the 2-spirals problem) and a real world problem. The XOR was chosen because is the simplest nonlinear problem one can imagine. Such a problem allows us to test the validity of our approach at the lower end of the complexity scale when parameters like the nonlinearity factor and the number of units on the second and third hidden layer are kept to a minimum. The 2-spiral problem can be seen as 'pathologically difficult' (Baffes & Zelle, 1992) and allows us to test the validity of the approach as the difficulty of the problem increases. Since numerous experiments have been performed on these problems in the past, the results presented here can be easily put into perspective. One real world problem has been chosen to investigate the validity of the approach in those cases in which there is no a priori information available regarding the degree of nonlinearity of the problem. The USER data set represents two classes of computer usage at the University of London Computer Centre (Hand, 1981).

Although the theoretical method presented and the bounds calculated using it do not depend on any particular training algorithm, all experimental results will unavoidably depend on such algorithm. Different algorithms might yield solutions using different number of neurons and weights but this corresponds to the practical reality of training neural networks.

The particular algorithm used here was the VLSI-friendly version of the Constraint Based Decomposition (CBD) presented in Draghici (2000b) and Draghici and Sethi (1997). This algorithm is a constructive algorithm which builds the network starting from scratch and adds neurons as they are needed. Although the algorithm is able to produce networks with a limited, user-chosen fan-in, this feature was disabled for these experiments because the mechanism which ensures the limited fan-in tends to increase the number of weights used.

6.2. Results

Sample solutions for the XOR, USER and 2-spiral problems are given in Fig. 8, Fig. 9 and Fig. 10, respectively. In the particular case of the XOR problem, the patterns have been shifted with -0.5 along both axes to avoid having some patterns on the very boundary of the region in which the results where demonstrated. Both the 2-spiral and the USER patterns have been scaled to fit in the $[-1, 1]$ square. All I/O images are drawn in the same square.

Since for the chosen algorithm the shape of the solution can depend on the order of the patterns in the training set, a set of 30 trials were performed for each problem. The patterns were randomly shuffled before each trial. The number of hyperplanes and weights used in each trial are presented in Fig. 11, Fig. 12 and Fig. 13, respectively for the XOR, USER and 2-spiral problems.

For each problem, the minimum distance between two patterns of opposite classes (d) and the maximum distance between any two patterns (D) were calculated.

The minimum distance between patterns of opposite classes was used to calculate the minimum theoretical weight range p. The values obtained were 1, 37.20, and 6.96 for the XOR, USER and 2-spirals, respectively (see row 5 in Fig. 7). Then, experiments were performed trying to obtain a solution using a range p as low as possible. The lowest values obtained were 1, 5 and 5 for the XOR, USER and 2-spirals, respectively (row 7 in Fig. 7).

	XOR	User	2 spirals
Min. distance between patterns of opposite classes (d)	1	0.026877	0.143527
Max. distance between any two patterns (D)	1.414213	1.659804	2
Average distance between patterns	1.138071	0.627921	0.764195
Standard deviation	0.203945	0.345186	0.38158
Weight range (theoretical)	1	37.20585	6.967348
Weight range (statistical)	1.2017	0.07920	5.2131
Weight range (experimental)	1	5	5
Lower bound for entropy bits (theoretical)	6	294	970
Lower bound for entropy bits (statistical)	8	245	873
Min.# of weights necessary (with theoretical p)	18	1021.4	2793.9
Min.# of weights necessary (with statistical p)	21.1	722.2	2550.4
Number of patterns classified in the same region (experimental average)	1.333	4.067	2.419
Min. # of weights necessary (with theory p)	13.5	251.1	1154.9
Min. # of weights necessary (with statistical p)	15.8	177.5	1054.2
Number of weights (experimental average)	12.4	73.2	551.4

Fig. 7. Summary of experimental results.

One can note that in practice, for two out of the three problems tested, it was possible to use weight ranges much below the theoretical values. This is because the theoretical value was derived in a worst-case situation and offers the guarantee that the solution exists for any pattern set respecting the minimum distance condition. The discrepancy between the theoretical value and the practical range used can be also seen as a measure of the difficulty of the problem. Note that for the 2-spiral problem the theoretical value is very close to the value actually needed. This might be taken as a confirmation that the 2-spiral problem is indeed 'pathologically difficult' as characterized in Baffes and Zelle (1992).

However, the USER data set shows that the theoretical bounds can be quite far from the values that can be used experimentally. Using the statistical assumptions detailed above, we have used the average distance between patterns and the standard deviation to calculate a statistical estimation for the necessary weight range p. These values are presented in row 6 in Fig. 7. A comparison between these values and the values actually needed (row 7 in Fig. 7) shows that the estimated yielded by the statistical approach was much more accurate than the worst-case estimate given by the theoretical approach.

The theoretical and statistical estimates of p were used to calculate a theoretical and statistical estimate for the lower bounds on the number of entropy bits (calculated according to Eq. (16)). These values are given in rows 8 and 9. Eq. (28) was then used to calculate theoretical and statistical estimates for the number of weights necessary. These values are presented in rows 10 and 11. As explained in Section 6.1, these values were calculated in the assumption that each pattern will be classified in its own region. In order to be able to assess the results, we have calculated the average number of patterns classified in the same region for each of the problems analyzed. These values correspond to the nonlinearity factor nl discussed in Section 6.1 and were used to correct the values calculated. Rows 13 and 14 present the theoretical and statistical estimated which take

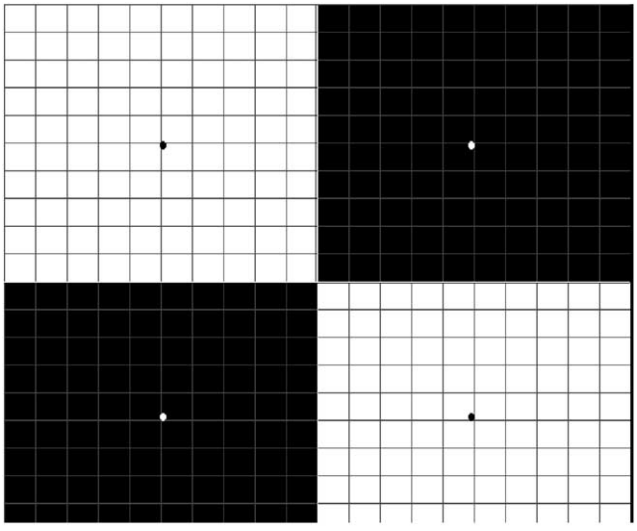

Fig. 8. The XOR problem solved with integer weights in the range [−1,1].

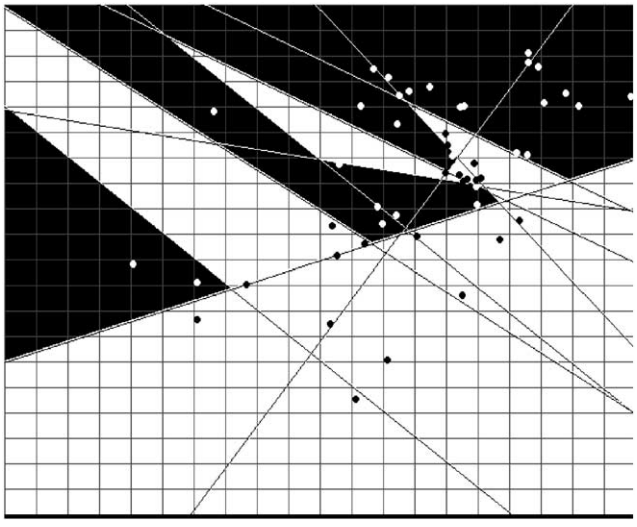

Fig. 9. The USER problem solved with 11 hyperplanes with 78 integer weights in the range [−5,5].

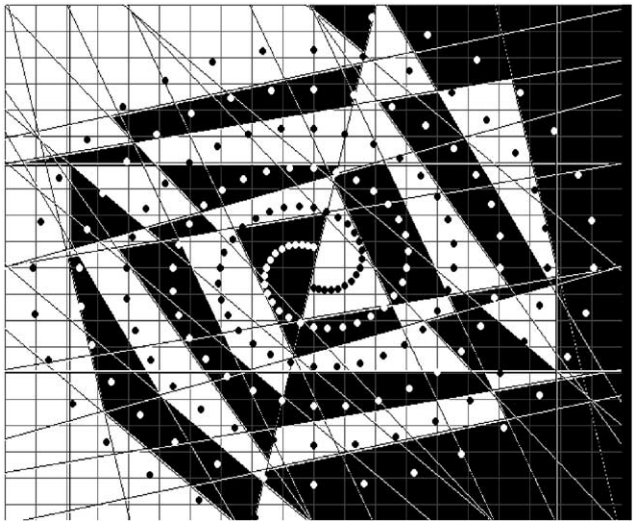

Fig. 10. The 2-spiral problem solved with 272 integer weights in the range [−5,5].

into consideration the average number of patterns per region. These values are to be compared with the average number of weights used for each problem.

The experiments presented have shown that the experimental values obtained for the number of weights are consistently smaller than the values calculated in the worst-case scenario considered by the approach presented. In practical terms this means that if a network is designed according to the method given, the size of the network will be sufficient even for the worst-case scenario but rather wasteful in most practical cases. It is believed that this is mainly due to the approximation introduced by approximating the radius D of the smallest hypersphere containing all patterns with the largest distance between any two patterns.

In practice, the number of weights calculated as above (with D) can be used as an upper limit for the network design. The network will never need more weights than the number of weights resulted from the given method. If a more precise estimation is needed, one can calculate the

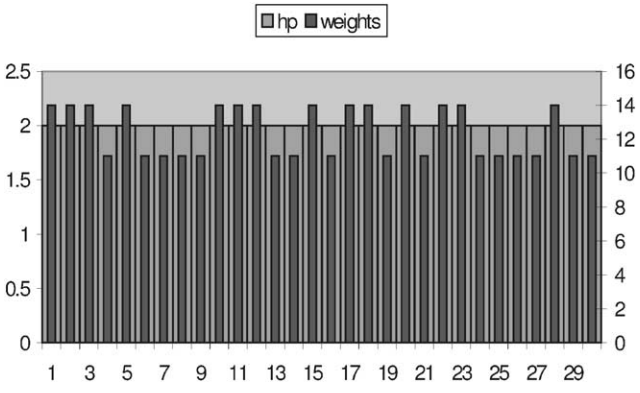

Fig. 11. The number of hyperplanes and weights for the XOR problem (30 trials).

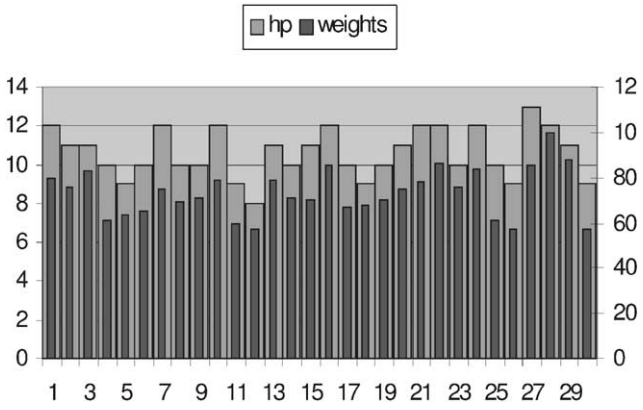

Fig. 12. The number of hyperplanes and weights for the USER data set (30 trials).

centroid $C(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$ of the patterns and calculate the radius as $\max_{x \in C_1 \cup C_2} \|x - C\|$.[3]

In all problems considered here, this radius can be calculated easily and would lead to much more accurate bounds for the number of weights. However, in the general case, the largest distance is easier to calculate than the radius of the hypersphere. For this reason, we have chosen to present the results obtained with the largest distance even if they underestimate the accuracy of the bounds presented.

6.3. Discussion

The results presented above are important because they allow the user of a limited precision integer weights neural network (e.g. the designer of a neural network VLSI chip) to choose the range of the weights in such a way that a solution is guaranteed to exist for the given category of problems.

One important problem that this paper does *not* address is how the solution weight state is found, i.e. how the neural network is trained. Our results simply guarantee that for a given set of classification problems there exists a solution with weights within a certain range which can be calculated from the problem itself. Therefore, this paper does not present an alternative to any of the reduced precision training algorithms mentioned above. Instead, the results presented here are to be used in conjunction with such techniques, either to initialize them with the desired weight range or to monitor their performance with respect to the precision reduction process.

Another issue is the choice of the minimum distance between classes as a measure for the difficulty of the classification problem. Clearly, this measure is not able to capture all aspects that determine the difficulty of such a problem. For instance, a linearly separable problem with a very small minimum distance between patterns of opposite classes can

[3] D is the maximum distance between any two patterns. From the definition of the centroid, it follows that the distance from the centroid to any pattern cannot be larger than the maximum distance between patterns

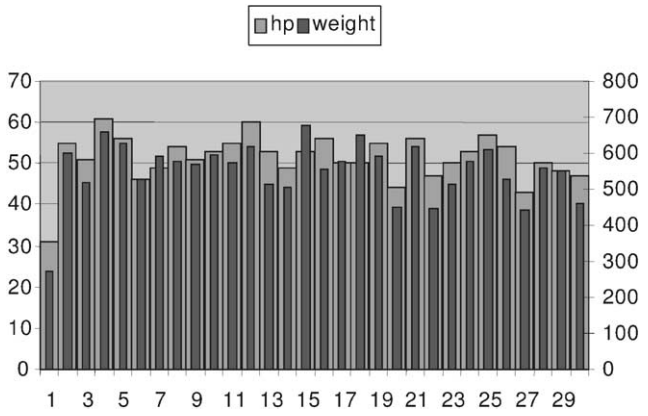

Fig. 13. The number of hyperplanes and weights for the 2-spiral problem (30 trials).

be solved with a single hyperplane and a single layer whereas a nonlinearly separable problem with a large minimum distance between classes will need several hyperplanes and at least two layers. Since it is generally accepted that a linearly inseparable problem is 'more difficult' than a linearly separable one, it appears that our measure is not very useful since in the case above it will suggest exactly the contrary. The apparent contradiction is caused by the fact that one tends to ignore the precision issue. Thus, in the example above we tend to take for granted the availability of that unique hyperplane that solves our linearly separable problem. In fact, if the weight values are integers in a very small interval, that hyperplane might not be available and no algorithm will be able to find a solution weight state. This argument reveals the fact that the two aspects, linear separability and weight range (which can be related to precision) are orthogonal. In other words, both need to be taken into consideration when assessing the abilities of a network to solve a given problem. The linear separability property of the problem is closely related to the *architecture* of the network whereas the minimum distance between patterns of different classes is related to the *weight range/precision*. Using an appropriate weight range only guarantees the existence of the solution if the architecture is appropriate. In the same way, an appropriate architecture guarantees the existence of the solution only if the weight precision is sufficient.

Another interesting aspect is how the approach presented above relates to other approaches trying to characterize the capabilities of a network. Probably the most interesting such approach is the one using the Vapnik–Chervonenkis (VC) dimension (Vapnik & Chervonenkis, 1971). Important results in this direction have been obtained by Baum and Haussler (1989), Haussler, Kearns, and Schapire (1994), Koiran and Sontag (1996), Vapnik (1992, 1998), etc. There are several important observations that need to be made here. First of all, the VC-complexity approach relies on choosing a network architecture and then estimating what that architecture can do. Indeed, the VC-dimension is defined as the maximum number of patterns that can be learned by a given architecture for all possible labelings. Our approach is different in principle inasmuch it characterizes the problem as related to the weight range while completely disregarding the architecture by assuming an appropriate architecture can be found. Again, the two approaches are complementary: the VC-complexity can be used to design an appropriate architecture while our approach can be used to calculate a sufficient weight range.

The connection between the VC-dimension and our approach is beautifully made by a series of recent papers by Suyari and Matsuba (1999a,b). In these papers, the authors calculate the storage capacity of a neural network using both the VC-dimension approach and the minimum distance between patterns. The validity of the minimum distance between patterns as a means to characterize the problem is confirmed by the good agreement between the capacity calculated using the classical replica method (Gutfreund & Stein, 1990; Mertens & Engel, 1997) and the capacity calculated using the minimum distance between patterns.[4]

Finally, one can suggest an approach in which all weights have limited precision but the set of nodes is divided into nodes carrying high-order information (coarse resolution) and nodes carrying low-order information (fine resolution). The question is whether such nodes can be combined to provide better capabilities for the network. We will point out that such a division of labor may in principle be implemented. However, this would require knowing when to stop positioning the coarse resolution hyperplanes and move to the fine resolution level. At this time, there are no known approaches able to address this problem. Furthermore, such decision has to be taken for each individual hyperplane and the ensemble of decisions will depend on the problem. At this time, there seems to be no simple solution for achieving this in a systematic way. This approach will be at the center of future research.

A different way of circumventing the limitations of limited precision is to use several layers of nonlinear neurons. By using a sigmoid or such, the clear-cut border between regions given by a sharp nonlinearity becomes a ridge whose width depends on the slope of the sigmoid. By combining several layers of such neurons, the ridges given by individual neurons can combine to create a complicated landscape which can, in principle, overcome the limitations of the reduced weight resolution. Two observations can be made here. Firstly, the results of this paper give only a worst-case bound. There will be cases in which subtle interactions between nonlinearities will provide solutions better than the worst-case solution. Secondly, such complicated interactions may be very difficult to find in the search associated to the training process. If only certain very specific

[4] Note that since the VC-approach considers all possible ways of labeling the patterns, our distance d between the closest patterns of opposite classes simply becomes the closest distance between any patterns.

combinations of parameters give the desired effect, it is likely that such solutions will correspond to points of the error surface that are surrounded by plateaus or local minima.

VLSI manufacturers tends to prefer architectures that: (i) use consistent elements fabricated with uniform technology and have the same precision over the whole chip; (ii) offer some guarantees for the worst case even if they do not find the best possible solutions in particular cases and (iii) use the same training algorithm for all units and all data points. The results given here can be useful in this context even if better solutions can be obtained for specific problems using more sophisticated algorithms.

7. Conclusions

This paper is centered around the neural networks capabilities related to the range of values available for the weights. It is suggested that the capabilities of feed-forward neural networks become more limited as restrictions are imposed on their weights. At one extreme, one has neural networks using real weights. Such neural networks have been shown to be very powerful from an approximation/classification point of view. At the other extreme, one can place neural networks using weights taking only integer values in a limited interval. The motivation of our interest in such networks is the cost of the implementation. In general, the higher the precision necessary in a neural network implementation, the higher the cost of the implementation. If the capabilities of networks using exclusively integer values in a limited range can be related to a measure of the difficulty of the problems they need to solve, such networks can open the road for very cost efficient implementations and new circuit designs.

This paper presents some results regarding the capabilities of such networks in classification problems. These results relate the difficulty of a classification problem (as characterized by the minimum distance between patterns of opposite classes) to the weight range in a worst-case situation. Thus, for any given classification problem, these results allow the computation of a weight range that guarantees the existence of a solution. This allows the designer of a neural network circuit to tailor the weight range to the difficulty of the problem by choosing the minimum range for which a solution is still guaranteed to exist.

The approach presented here does not deal with the training, i.e. the problem of how to find a set of weights for a given problem. In other words, our analysis is not a substitute for any of the quantization methods or training algorithms able to work with limited precision. Instead, the results presented here are to be used in conjunction with such algorithms to offer some degree of control as to how far such precision reduction procedures should be asked to go.

The paper also discusses the relationship between the weight range and the architecture of the network. It is suggested that the weight range and the architecture are two independent issues. The architecture of the network is related to certain properties of the classification problems such as linear separability whereas the precision/range of the weights is related to other properties such as the minimum distance between patterns. Both types of issues need to be taken into consideration when assessing the solvability of a particular problem by a given network. A similar relationship exists between the approach used in this paper to assess the capabilities of a neural network and its capability as described by the VC capacity. It is being suggested that the capability of solving a given classification problem and the optimal storage capacity calculated using the VC approach are two independent issues. The VC-dimension and the optimal storage capacity are closely related to architectural issues whereas the classification abilities investigated in this paper are closely related to weight precision issues. Thus, the approach presented here does not aim to be an alternative approach to the VC-dimension analysis but rather a complementary point of view able to introduce in the equation the precision and the range of the weights which are currently missing from the VC approach. The paper also presented a method for calculating a worst-case estimate for the size of a VLSI-friendly network for a given classification problem in the context of using limited precision integer weights. Several experiments involving two benchmarks and a real-world problem have been presented. The experiments showed that the theoretical results based on a worst-case scenario yielded unnecessary large lower bounds for the necessary weight range. Modifications based on statistical considerations have been suggested and tested. The experiments performed showed that these modifications yield good approximations for the weight range. However, the estimations obtained for the number of necessary weights are still rather pessimistic and may lead to an oversized network.

The worst-case values calculated are valuable as an upper limit for the number of weights to be used. Thus, the approach guarantees that a network having the appropriate number of weights will be able to solve any classification problem with integer weights in the given range.

References

Akers, L. A., Walker, M. R., Ferry, D. K., & Grondin, R. O. (1988). Limited interconnectivity in synthetic neural systems. In R. Eckmiller & C. van der Malsburg, *Neural computers* (pp. 407–416). Berlin: Springer.

Albrecht, A. (1992). *On bounded-depth fhreshold circuits for pattern functions* (GOSLER Report No. 15/92). Technische Hochschule Leipzig, FB Mathematik, and Informatik.

Albrecht, A. (1995). On lower bounds for the depth of thresh-old circuits with weights from (−1,0, + 1). *Lecture Notes in Computer Science*, *961*, 404–416.

Alon, N., & Bruck, J. (1994). Explicit constructions of depth-2 majority circuits for comparison and addition. *SIAM Journal on Discrete Mathematics*, *7* (1), 1–8.

Alspector, J., Gupta, B., & Allen, R. B. (1989). *Performance of a stochastic learning microchip*, *Advances in neural information processing systems* (Vol. 1, pp. 748–760). San Mateo, CA: Morgan Kaufman.

Baffes, P. T., & Zelle, J. M. (1992). Growing layers of perceptrons: introducing the extentron algorithm. *International Joint Conference on Neural Networks* (Vol. 2, pp. 392–397). Baltimore.

Baker, T., & Hammerstrom, D. (1988). *Modifications to artificial neural network models for digital hardware implementations* (Technical Report No. CS/E 88-035). Department of Computer Science and Engineering, Oregon Graduate Center.

Baum, E. B., & Haussler, D. (1989). What size net gives good generalization? *Neural Computation*, *1*, 151–160.

Beigel, R., & Tarui, J. (1991). On ACC. In IEEE, *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science* (pp. 783–792). San Juan, Porto Rico: IEEE Computer Society Press.

Beiu, V. (1994). *Neural networks using threshold gates: A complexity analysis of their area and time efficient VLSI implementation*. PhD Thesis, Katholieke Universiteit Leuven, Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium.

Beiu, V. (1996a). Digital integrated circuits implementations. In E. Fiesler & R. Beale, *Handbook of neural computation*Oxford University Press/ Institute of Physics Publication.

Beiu, V. (1996b). Entropy bounds for classification algorithms. *Neural Network World*, *6* (4), 497–505.

Beiu, V. (1997). Constant fan-in digital neural networks are VLSI-optimal. In S. Ellacott, J. Mason & I. Anderson, *Mathematics of neural networks: Models, algorithms and applications* (pp. 89–94). Boston: Kluwer Academic Publishers.

Beiu, V. (1998). On the circuit and VLSI complexity of threshhold gate comparison. *Neurocomputing*, *19* (1–3), 77–98.

Beiu, V. (2000). High-speed noise robust threshold gates. *Proceedings of the International Semiconductor Conference CAS'2000* (pp. 79–82). IEEE CS Press.

Beiu, V. (2000). Ultra-fast noise immune CMOS threshold gates. *Proceedings of the Midwest Symposium on Circuits and Systems MWSCAS'2OOO* (pp. 1310–1313). IEEE Press.

Beiu, V. (2001). *Adder and multiplier circuits employing logic gates having discrete, weighted inputs and methods of performing combinatorial operations therewith*. US Patent US6205458. (http://www.delphion.com/details?pn = US06205458).

Beiu, V. (2001). *Logic gate having reduced power dissipation and method of operation thereof*. US Patent US6259275. (http://www.delphion.com/details?pn = US06259275).

Beiu, V. (2001). On higher order noise immune perceptrons. *Proceedings of the International Joint Conference on Neural Networks IJCNN'2001* (pp. 246–251). IEEE Press.

Beiu, V. (2001). On VLSI-optimal neural computations. *Proceedings of the 13th International Conference on Control Systems and Computer Science* (pp. 158–169). Politechnica University Press.

Beiu, V., & Draghici, S. (1997). Limited weights neural networks: Very tight entropy based bounds. In D. W. Pearson, *Proc. of the Second International ICSC Symposium on Soft Computing SOCO'97* (pp. 111–118). Nimes, France: Academic Press ISBN 3-906454-06-l.

Beiu, V., & Makaruk, H. (1998). Deeper sparser nets can be optimal. *Neural Processing Letters*, *8* (13), 201–210.

Beiu, V., & Taylor, J. G. (1996). On the circuit complexity of sigmoid feedforward neural networks. *Neural Networks*, *9* (7), 1155–1171.

Beiu, V., Peperstraete, J. A., & Lauwereins, R. (1992). Simpler neural networks by fan-in reduction. *Proceedings of the IJCNN'92 (Beijing, China)* (pp. 204–209). IEEE Press/PHEI Press.

Beiu, V., Peperstraete, J., Vandewalle, J., & Lauwereins, R. (1994). On the circuit complexity of feedforward neural networks. In M. Marinaro & P. G. Morasso, *Artificial Neural Networks 4-Proceedings of ICANN'94, Sorrento, Italy* (pp. 521–524). Berlin: Springer.

Beiu, V., Draghici, S., & Makaruk, H. E., (1997). On limited fan-in optimal neural networks. *Proceedings of the Brazilian Symposium on Neural Networks IV SBRN* (pp. 19–30).

Beiu, V., Draghici, S., & Pauw, T. D. (1999). A constructive approach to calculating lower bounds. *Neural Processing Letters*, *9* (1), 1–12.

Bruck, J. (1990). Harmonic analysis of polynomial threshold functions. *SIAM Journal on Discrete Mathematics*, *3* (2), 168–177.

Bruck, J., & Goodman, J. W. (1990). On the power of neural networks for solving hard problems. *Journal of Complexity*, *6* (2), 129–135.

Bruck, J., & Smolensky, R. (1992). Polynomial threshold functions. ACO functions, and spectral norms. *SIAM Journal on Computing*, *21* (1), 33–42.

Cauwenberghs, G. (1993). A fast stochastic error-descent algorithm for supervised learning and optimisation. In S. J. Hanson, J. D. Cowan & C. L. Giles, *Advances in neural information processing systems* (Vol. 5, pp. 244–251). Los Altos: Morgan (Kaufmann).

Cauwenberghs, G. (1998). Neuromorphic learning VLSI systems: A survey. In T. S. Lande, *Neuromorphic systems engineering* (pp. 381–408). Dordrecht: Kluwer Academic Publishers.

Chandra, A. K., Stockmeyer, L., & Vishkin, U. (1984). Constant depth reducibility. *SIAM Journal on Computing*, *13* (2), 423–439.

Coggins, R., & Jabri, M. (1994). *Wattle: A trainable gain analogue VLSI neural network*, *Advances in neural information processing systems NIPS'93* (Vol. 6, pp. 874–881). Los Altos: Morgan Kaufmann.

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

Dertouzos, M. L. (1965). *Threshold logic: A synthesis approach*, Cambridge, MA: MIT Press.

Draghici, S. (2000a). Neural networks in analog hardware-design and implementation issues. *International Journal of Neural Systems*, *10* (1), 1–25.

Draghici, S. (2000b). The constraint based decomposition (CBD). *Neural Networks*, *14* (4–5), 527–550.

Draghici, S., & Sethi, I. K. (1997). Adapting theoretical constructive algorithms to hardware implementations for classification problems. *Proceedings of the International Conference on Engineering Applications of Neural Networks EANN'97* (pp. 303–308). Systeemitekniikan Seurary.

Draghici, S., Beiu, V., & Sethi, I. (1997). A VLSI optimal constructive algorithm for classification problems. In *Smart engineering systems: Neural networks, fuzzy logic, data mining and evolutionary programming* (pp. 141–151). St Louis, MO: ASME Press .

Dundar, G., & Rose, K. (1995). The effect of quantization on multilayer neural networks. *IEEE Transactions on Neural Networks*, *6*, 1446–1451.

Duong, T. A., & Daud, T. (1999). Cascade error projection: A learning algorithm for hardware implementation. In J Mira & J. V. S'anchez-Andre'es, *Foundations and tools for neural modeling, lecture notes in computer science* (pp. 450–457). Berlin: Springer.

Duong, T. A., Eberhardt, S. P., Daud, T., & Thakoor, A. (1996). *Learning in neural networks: VLSI implementation strategies*, New York: McGraw-Hill.

Eberhardt, S., Duong, T., & Thakoor, A. (1989). Design of parallel hardware neural network systems from custom analog VLSI building block chips. *Proceedings of the International Joint Conference on Neural Networks* (Vol. 2, pp. 183–190).

Fiesler, E., & Beale, R. (1996). *Handbook of neural computation*. Philadelphia, PA: Oxford University Press/Institute of Physics Publication.

Fiesler, E., Choudry, A., & Caulfield, H. J. (1990). A weight discretization paradigm for optical neural networks. *Proceedings of the International Congress on Optical Science and Engineering ICOSE'90 SPIE* (Vol. 1281, pp. 164–l73).Bellingham, WA: SPIE.

Fischler, M. A. (1962). *Investigations concerning the theory and synthesis of linearly separable switching functions*. PhD thesis, Stanford University. EE Department.

Flower, B., & Jabri, M. (1993). *Summed weight neuron perturbation: An O(n) improvement over weight perturbation*, *Advances in Neural Information Processing Systems* (Vol. 5, pp. 212–219). Los Altos: Morgan Kaufmann.

Glessner, M., & Pöchmüller, W. (1994). *An overview of neural networks in VLSI*, London: Chapman & Hall.

Goldmann, M., Håstad, J., & Razborov, A. A. (1992). Majority gates vs. general weighted threshold gates. *Computational Complexity*, *2* (4), 277–300.

Goldmann, M., & Karpinski, M. (1993). Simulating threshold circuits by majority circuits. *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing* (pp. 551–560). San Diego, California.

Graf, H. P., & Henderson, D. (1990). A reconfigurable CMOS neural network. *Technical Digest of IEEE International Solid-State Circuits Conference* (pp. 144–145).

Graf, H. P., Jackel, L. D., Howard, R. E., & Straughn, B. (1986). VLSI implementation of a neural network memory with several hundreds of neurons. In J. S. Denker, *Neural Networks for Computing, AIP Conference Proceedings* (Vol. 151, pp. 182–187). Snowbird, UT.

Graf, H. P., Jackel, L. D., & Hubbard, W. E. (1988). VLSI implementation of a neural network model. *Computer*, 41–49.

Gutfreund, H., & Stein, Y. (1990). Capacity of neural networks with discrete synaptic couplings. *Journal of Physics A*, *23* (12), 2613–2630.

Hajnal, A., Maass, W., Pudlàk, P., Szegedy, M., & Turàn, G. (1993). Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, *46* (2), 129–154.

Hammerstrom, D. (1988). The connectivity analysis of simple associations: How many connections do you need? In D. Z. Anderson, *Neural Information Processing Systems, Proceedings of NIPS'87* (pp. 338–347). New York: American Institute of Physics.

Hand, D. J. (1981). *Discrimination and classification*, New York: Wiley.

Håstad, J. (1986). Almost optimal lower bounds for small depth circuits. *Proceedings of the 18th Annual ACM Symposium on Theory of Computing* (pp. 6–20). Berkeley, California.

Håstad, J., & Goldmann, M. (1991). On the power of small-depth threshold circuits. *Computational Complexity*, *1* (2), 113–129.

Håstad, J., Wegener, I., Wurm, N., & Yi, S. -Z. (1994). Optimal depth, very small size circuits for symmetric functions in AC^0. *Information and Computation*, *108* (2), 200–211.

Haussler, D., Kearns, M., & Schapire, R. (1994). Bounds on the sample complexity of bayesian learning using information theory and the VC dimension. *Machine Learning*, *14*, 83–113.

Hecht-Nielsen, R. (1987). Kolmogorov's mapping neural network existence theorem. *Proceedings of IEEE International Conference on Neural Networks* (pp. 11–13). IEEE Press.

Hirai, Y. (1993). Hardware implementation of neural networks in Japan. *Neurocomputing*, *5* (3), 3–16.

Höhfeld, M., & Fahlman, S. E. (1992). Learning with limited numerical precision using the Cascade-Correlation algorithm. *IEEE Transactions on Neural Networks*, *3* (4), 602–611.

Hofmeister, T., Hohberg, W., & Köhling, S. (1991). Some notes on threshold circuits, and multiplication in depth 4. In L. Budach, *Proceedings of Fundamentals of Computation Theory (FCT '91)* (Vol. 529, pp. 230–239). Berlin: Springer.

Holler, M. (1991). *VLSI implementations of learning and memory systems*, *Advances in Neural Information Processing Systems*, (Vol. 3, pp. 993–1000). San Mateo, CA: Morgan Kaufmann.

Holler, M., Tam, S., Castro, H., & Benson, R. (1989). An electrically trainable artificial neural network (ETANN) with 10240 floating gate synapses. *Proceedings of IEEE/INNS International Joint Conference on Neural Networks* (Vol. 2, pp. 191–196).

Hollis, P., & Paulos, J. J. (1990). Artificial neural networks using MOS analog multiplier. *IEEE Journal of Solid State Circuits*, *25*, 849–855.

Hollis, P., & Paulos, J. (1994). A neural network learning algorithm tailored for VLSI implementation. *IEEE Transactions on Neural Networks*, *5* (5), 784–791.

Hollis, P., Harper, J., & Paulos, J. (1990). The effects of precision constraints in a backpropagation learning network. *Neural Computation*, *2* (3), 363–373.

Hong, J. (1987). *On connectionist models* (Technical Report No. 87-012). Department of Computer Science, University of Chicago.

Horne, B., & Hush, D. (1994). On the node complexity of neural networks. *Neural Networks*, *7* (9), 1413–1426.

Hornik, K. (1991). Approximation capabilities of multiplayer feed forward networks. *Neural Networks*, *4* (2), 251–257.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayered feed forward networks are universal approximators. *Neural Networks*, *2* (5), 359–366.

Hornik, K., Stinchcombe, M., & White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feed forward networks. *Neural Networks*, *3* (5), 551–560.

Hu, S. (1965). *Threshold logic*, Berkeley: University of California Press.

Impagliazzo, R., Paturi, R., & Saks, M. (1993). Size-depth tradeoffs for threshold circuits. *Proceedings of the 25th ACM Symposium on Theory of Computing* (pp. 541–550).

Jabri, M. A., & Flower, B. (1992). Weight perturbation: An optimal architecture and learning technique for analog VLSI feed forward and recurrent multilayer networks. *IEEE Transactions on Neural Networks*, *3* (1), 154–157.

Jabri, M. A., Coggins, R. J., & Flower, B. (1996). *Adaptive analog VLSI neural systems*, London: Chapman & Hall.

Koiran, P., & Sontag, E. D. (1996). *Neural networks with quadratic VC dimension*, *Advances in Neural Information Processing (NIPS) 8* (pp. 197–203). Cambridge, MA: MIT Press.

Kolmogorov, A. N. (1957). On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition. *Doklady Akademii Nauk*, *114*, 953–956.

Konig, A. (1995). Survey and current status of neural network hardware. In F. Fogelman-Soulie & P. Gallinari, *Proceedings of the International Conference on Artificial Neural Networks* (pp. 391–410).

Kwan, H., & Tang, C. Z. (1992). Designing multilayer feedforward neural networks using simplified activation functions and one-power-of-two weights. *Electronics Letters*, *28* (25), 2343–2344.

Kwan, H., & Tang, C. Z. (1993). Multiplierless multiplayer feedforward neural networks design suitable for continuous input–output mapping. *Electronics Letters*, *29* (14), 1259–1260.

Lande, T. S. (1997). Special issue on neuromorphic engineering. *Int. J. Analog Int. Circ. Signal Proc*.

Lande, T. S. (1998). *Neuromorphic systems engineering*. Dordrecht: Kluwer Academic.

Lundin, T., Fiesler, E., & Moerland, P. (1996). Connectionist quantization functions. *SIPAR Workshop '96 Proceedings* (pp. 33–36).

Marchesi, M., Orlandi, G., Piazza, F., Pollonara, L., & Uncini, A. (1990). Multilayer perceptrons with discrete weights. *Proceedings of International Joint Conference on Neural Networks* (Vol. 2, pp. 623–630).

Marchesi, M., Orlandi, G., Piazza, F., & Uncini, A. (1993). Fast neural networks without multipliers. *IEEE Transactions on Neural Networks*, *4* (1), 53–62.

Mayoraz, E. (1991). On the power of networks of majority functions. In A. Prieto, *Proceedings of Artificial Neural Networks (IWANN '91)* (pp. 78–85). Vol. 540. Berlin: Springer.

Mead, C. A. (1989). *Analog VLSI and neural systems*, Reading, MA: Addison-Wesley.

Meir, R., & Fontanari, J. F. (1993). On the precision constraints of threshold elements. *Network*, *4*, 381–391.

Mertens, S., & Engel, A. (1997). Vapnik–Chervonenkis dimension of neural networks with binary weights. *Physics Reviews*, *E55*, 4478–4488.

Minnick, R. C. (1961). Linear input logic. *IRE Transactions on Electronic Computers*, *EC-10*, 6–16.

Moerland, P., & Fiesler, E. (1997). *Neural network adaptations to hardware implementations*, Bristol, UK/New York, USA: Oxford University Press/Institute of Physics.

Moerland, P., Fiesler, E., & Saxena, I. (1998). Discrete all positive multilayer perceptrons for optical implementation. *Optical Engineering*, *37* (4), 1305–1315.

Morgan, N. (1990). *Artificial neural networks: Electronic implementations*. Los Alamos, CA: IEEE Computer Society Press.

Muroga, S. (1962). Generation of self-dual threshold functions and lower bound of the number of threshold functions and a maximum weight. *Switching Circuit Theory and Logical Design*, *S135*, 170–184.

Muroga, S. (1965). Lower bounds on the number of threshold functions and a maximum weight. *IEEE Transactions on Electronic Computers*, *EC-14* (2), 136–148.

Muroga, S. (1971). *Threshold logic and its applications*, New York: Wiley-Interscience.

Nakayama, K., Inomata, S., & Takeuchi, Y. (1990). A digital multilayer neural network with limited binary expressions. *Proceedings of International Joint Conference on Neural Networks IJCNN'90* (Vol. 2, pp. 587–592).

Parberry, I. (1994). *Circuit complexity and neural networks*, Cambridge, MA: MIT Press.

Parberry, I., & Schnitger, G. (1988). Parallel computation with threshold functions. *Journal of Computer and System Sciences*, *36* (3), 278–302.

Raghavan, P. (1988). Learning in threshold networks. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 19–27). MIT: ACM Press.

Razborov, A. A. (1992). On small depth threshold circuits (invited). In O. Nurmi & E. Ukkonen, *Proceedings of Algorithm Theory (SWAT '92)* (Vol. 621, pp. 42–52). Berlin: Springer.

Redkin, N. P. (1970). Synthesis of threshold circuits for certain classes of boolean functions. *Kibernetika*, *5*, 6–9.

Reyneri, L. M., & Filippi, E. (1991). An analysis on the performance of silicon implementations of backpropagation algorithms for artificial neural networks. *IEEE Computers*, *40* (12), 1380–1389.

Sánchez-Sinencio, E. & Lau, C. (1992). *Artificial neural networks: Electronic implementations*. Los Alamos, CA: IEEE Computer Society Press.

Sánchez-Sinencio, E., & Newcomb, R. (1992). Special issue on neural network hardware. *IEEE Transactions on Neural Networks*, *3* (3).

Sheng, C. L. (1969). *Threshold logic*, New York: Academic Press.

Siegelmann, H. T. (1995). Computation beyond the Turing limit. *Science*, *238* (28), 632–637.

Simard, P. Y., & Graf, H. P. (1994). Backpropagation without multiplications. In J. D. Cowan, G. Tesauro & J. Alspector, *Advances in Neural Information Processing Systems* (pp. 232–239). San Mateo, CA: Morgan Kaufmann.

Siu, K. -Y. (1995). *Discrete neural computation*, Englewood Cliffs: Prentice Hall.

Siu, K. -Y., & Bruck, J. (1990). Neural computation of arithmetic functions. *Proceedings of the IEEE*, *78* (10), 1669–1675.

Siu, K. -Y., & Bruck, J. (1991). On the power of threshold circuits with small weights. *SIAM Journal on Discrete Mathematics*, *4* (3), 423–435.

Siu, K. -Y., & Roychowdhury, V. (1993). Optimal depth neural networks for multiplication and related problems. In S. J. Hanson, J. D. Cowan & C. L. Giles, *Advances in Neural Information Processing Systems* (Vol. 5, pp. 59–64). San Mateo, CA: Morgan Kaufmann.

Siu, K. -Y., & Roychowdhury, V. P. (1994). On optimal depth threshold circuits for multiplication and related problems. *SIAM Journal on Discrete Mathematics*, *7* (2), 284–292.

Siu, K.-Y., Roychowdhury, V., & Kailath, T. (1990). *Computing with almost optimal size threshold circuits* (Technical Report). Information Systems Laboratory, Stanford University.

Siu, K. -Y., Roychowdhury, V. P., & Kailath, T. (1991). Depth-size tradeoffs for neural computation. *IEEE Transactions on Computers—Special Issue on Artificial Neural Networks*, *40* (12), 1402–1412.

Siu, K. -Y., Roychowdhury, V., & Kailath, T. (1993). Computing with almost optimal size neural networks. In S. J. Hanson, J. D. Cowan & C. L. Giles, *Advances in neural information processing systems* (Vol. 5, pp. 19–26). San Mateo, CA: Morgan Kaufmann.

Sontag, E. D. (1990). *On the recognition capabilities of feed-forward nets* (Technical Report No. NJ 08903). New Brunswick: Rutgers University.

Sprecher, D. A. (1965). On the structure of continuous functions of several variables. *Transactions of the American Mathematical Society*, *115*, 340–355.

Sprecher, D. A. (1993). A universal mapping for Kolmogorov's superposition theorem. *Neural Networks*, *6* (8), 1089–1094.

Sprecher, D. A. (1996). A numerical implementation of Kolmogorov's superpositions. *Neural Networks*, *9* (5), 765–772.

Sprecher, D. A. (1997). A numerical implementation of Kolmogorov's superpositions II. *Neural Networks*, *10* (3), 447–457.

Stinchcombe, M., & White, H. (1989). *Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions*, San Diego: SOS Printing.

Suyari, H., & Matsuba, I. (1999a). Information theoretical approach to the storage capacity of neural networks with binary weights. *Physics Review E*, *60* (4), 4576–4579.

Suyari, H., & Matsuba, I. (1999). New information theoretical approach to the storage capacity of neural networks with binary weights. *Proceedings of the ICANN'99* (pp. 43l–436).

Takahashi, H., Tomita, E., Kawabata, T., & Kyuma, K. (1991). A quantized backpropagation learning rule and its applications to optical neural networks. *Optical Computing and Processing*, *1* (2), 175–182.

Tang, C., & Kwan, H. (1993). Multilayer feedforward neural networks with single power-of-two weights. *IEEE Transactions on Signal Processing*, *41* (8), 2724–2727.

Vapnik, V. N. (1992). *Principles of risk minimization for learning theory*, *Advances in Neural Information Processing NIPS 4* (pp. 831–838). San Mateo, CA: Morgan Kaufmann.

Vapnik, V. N. (1998). *Statistical learning theory*, New York: Wiley.

Vapnik, V. N., & Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theoretical Probability and its Applications*, *16*, 264–280.

Vincent, J. M., & Myers, D. J. (1992). Weight dithering and wordlength selection for digital backpropagation networks. *BT Technology Journal*, *10* (3), 1180–1190.

Wegener, I. (1993). Optimal lower bounds on the depth of polynomial-size threshold circuits for some arithmetic functions. *Information Processing Letters*, *46* (2), 85–87.

Widrow, B., & Lehr, M. (1990). 30 years of adaptive neural networks: perceptron, madaline and backpropagation. *Proceedings of the IEEE* (Vol. 78, pp. 1415–1442). IEEE.

Winder, R. O., (1962). *Threshold logic*. Unpublished Doctoral Dissertation, Mathematical Department, Princeton University.

Winder, R. O. (1963). Bounds on threshold gate realizability. *IRE Transactions on Electronic Computers*, *EC-12*, 561–564.

Xie, Y., Jabri, M. A. (1991). *Training algorithms for Limited precision feedforward neural networks* (Technical Report No. SEDAL TR 1991-8-3). School of EE, University of Sydney, Australia.