# On a Learnability Question Associated to Neural Networks with Continuous Activations (Extended Abstract)[‡]

**Bhaskar DasGupta**[†]
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455-0159
dasgupta@cs.umn.edu

**Hava T. Siegelmann**[*]
Department of Computer Science
Bar-Ilan University
Ramat-Gan 52900, Israel
hava@bimacs.cs.biu.ac.il

**Eduardo Sontag**[*]
Department of Mathematics
Rutgers University
New Brunswick, NJ 08903
sontag@control.rutgers.edu

## Abstract

This paper deals with learnability of concept classes defined by neural networks, showing the hardness of PAC-learning (in the complexity, not merely information-theoretic sense) for networks with a particular class of activation. The obstruction lies not with the VC dimension, which is known to grow slowly; instead, the result follows the fact that the *loading* problem is NP-complete. (The complexity scales badly with input dimension; the loading problem is polynomial-time if the input dimension is constant.) Similar and well-known theorems had already been proved by Megiddo and by Blum and Rivest, for *binary-threshold* networks. It turns out the general problem for continuous sigmoidal-type functions, as used in practical applications involving steepest descent, is *not* NP-hard –there are "sigmoidals" for which the problem is in fact trivial– so it is an open question to determine what properties of the activation function cause difficulties. Ours is the first result on the hardness of loading networks which do not consist of binary neurons; we employ a piecewise-linear activation function that has been used in the neural network literature. Our theoretical results lend further justification to the use of incremental (architecture-changing) techniques for training networks.

[*]Research supported in part by US Air Force Grant AFOSR-91-0343

[†]Research supported in part by NSF Grant CCR-92-08913

[‡]A part of the results reported here will also appear in V. P. Roychowdhury, K. Y. Siu, and A. Orlitsky (eds.), *Theoretical Advances in Neural Computation and Learning*, Kluwer Academic Publishers, 1994

## 1 INTRODUCTION

We show (modulo RP≠NP) the hardness of PAC learning for concept classes defined by certain analog neural networks; this is a consequence of the fact that the loading problem for a typical such architecture (hypothesis class) is proved to be NP-complete. Our results are similar in spirit those due to Blum and Rivest, except that we focus on continuous as opposed to binary node functions. *Continuous* sigmoidal-type functions are used in all practical applications involving steepest descent, and for these, the loading problem is *not* in general NP-hard –there are "sigmoidals" for which the problem is in fact trivial– so it is an open question to determine what properties of the activation function cause difficulties. Ours is the first result on the hardness of loading networks which do not consist of binary neurons; we employ a piecewise-linear activation function that has been used in the neural network literature.

### 1.1 WHY NEURAL NETS?

Artificial Neural networks have become a very popular tool for machine learning, and many papers have been written dealing with their application to practical problems. In this role, a network is trained to recognize complex associations between inputs and outputs that were presented during a supervised training cycle. These associations are incorporated into the weights of the network, which encode a distributed representation of the information that was contained in the patterns. Once thus "trained," the network will compute an input/output mapping which, if the training data was representative enough, it is hoped will closely match the unknown rule which produced the original data. It is customary to find claims in the experimental literature to the extent that neural networks are particularly good for prediction. Our objective in this, as well as related work we have done, is to critically analyze such claims from a theoretical viewpoint. In particular, in the case of this paper, from the point of view of PAC learning. (Massive parallelism of computation, as well as noise and fault tolerance, are often also offered

as practical justifications for the use of neural nets as learning paradigms; we do not consider those aspects in this paper.)

By "neural network" we always mean, in this paper, feedforward ones of the type routinely employed in artificial neural nets applications. That is, a net consists of a number of processors ("nodes" or "neurons") each of which computes a function of the type

$$y = \sigma\left(\sum_{i=1}^{k} a_i u_i + b\right) \qquad (1)$$

of its inputs $u_1, \ldots, u_k$. These inputs are either external (input data is fed through them) or they represent the outputs $y$ of other nodes. No cycles are allowed in the connection graph and the output of one designated node is understood to provide the output value produced by the entire network for a given vector of input values. The possible coefficients $a_i$ and $b$ appearing in the different nodes are the *weights* of the network, and the functions $\sigma$ appearing in the various nodes are the *node* or *activation* functions. An *architecture* specifies the interconnection structure and the $\sigma$'s, but not the actual numerical values of the weights themselves.

In most practical work, $\sigma$ is the "standard sigmoid function" $1/(1+e^{-x})$, though other continuous activations (see below) are less frequently used as well. The "training" process is based on steepest descent minimization of a cost function, so as to fit weights to data samples; this is essentially the so-called "backpropagation" approach. (Sometimes more sophisticated techniques such as high-order (Newton), conjugate gradient, or sequential quadratic programming methods are used too.) It is well-known that these numerical techniques tend to run slowly, especially for high-dimensional data. Thus one may ask, as done by Judd and others (see for instance [14, 15, 5, 17, 28]) if there exists a *fundamental barrier* to training by general feedforward networks, a barrier that is insurmountable no matter which particular algorithm one uses. (Those techniques which *adapt* the architecture to the data, such as cascade correlation or incremental techniques, would not be subject to such a barrier.) Thus one must consider the tractability of the following decision problem ("loading"): *Given a network architecture (interconnection graph as well as choice of activation function) and a set of training examples, does there exist a set of weights so that the network produces the correct output for all examples?*

## 1.2 PAC LEARNING AND LOADING

The simplest neural network, i.e., the perceptron, consists of one threshold neuron only. It is easily verified that the computational time of the loading problem in this case is polynomial in the size of the training set irrespective of whether the input takes continuous or discrete values. This can be achieved via a linear programming technique. On the other hand, we show that, for networks employing a simple, saturated piecewise-linear activation function, and two hidden units only, the loading problem is NP-complete. This shows that, indeed, *any possible* neural net learning algorithm (for this activation function) based on fixed architectures faces severe computational barriers. Furthermore, our result implies non-learnability in the PAC sense under the complexity-theoretic assumption of $RP \neq NP$. We also generalize our result to another similar architecture.

The work most closely related to ours is that due to Blum and Rivest; see [5]. These authors showed a similar NP-completeness result for networks having the same architecture, but where the activation functions are all of a hard threshold type, that is, they provide a binary output $y$ equal to 1 if the sum in equation (1) is positive, and 0 otherwise. In their papers, Blum and Rivest explicitly posed as an open problem the question of establishing NP-completeness, for this architecture, when the activation function is "sigmoidal" and they conjecture that this is indeed the case. (For the architectures considered in Judd's work, in contrast, enough measurements of internal variables are provided that there is essentially no difference between results for varying activations, and the issue does not arise there. However, it is not clear what are the consequences for practical algorithms when the obstructions to learning are due to considering such architectures. In any case, we address here the open problem exactly as posed by Blum and Rivest.)

It turns out that a definite answer to the question posed by Blum and Rivest is not possible. It is shown in [24] that for *certain* activation functions $\sigma$, the problem can be solved in constant time, independently of the input size, and hence the question is *not* NP-complete. In fact, there exist "sigmoidal" functions, innocent-looking qualitatively (bounded, infinite differentiable and even analytic, and so forth) for which any set of data can be loaded, and hence for which the loading problem is not in NP. The functions used in the construction in [24] are however extremely artificial and in no way likely to appear in practical implementations. Nonetheless, the mere *existence* of such examples means that the mathematical question is nontrivial.

The main open question, then, is to understand if "reasonable" activation functions lead to NP-completeness results similar to the ones in the work by Blum and Rivest or if they are closer to the other extreme, the purely mathematical construct in [24]. The most puzzling case is that of the standard sigmoid function, $1/(1+e^{-x})$. For that case we do not know the answer yet, but we conjecture that NP-completeness will indeed hold. (In this context we should mention the very recent and independent work by Höffgen in [12]; he proves the hardness of the interpolation problem by sigmoidal nets with two hidden units when the weights are –severely– restricted to take just *binary values*. The binary assumption simplifies matters considerably, and in any case however the interpolation as opposed to classification problem is completely different from the problem that we are considering). It is the purpose of this paper to show an NP-completeness result for the piecewise linear

or "saturating" activation function; this is an activation that has regularly appeared in the neural networks literature, especially in the context of hardware implementations –see e.g.[3, 6, 18, 28]– and which is much simpler to analyze than the standard sigmoid. We view our result as a first step in dealing with the general case of arbitrary piecewise linear functions, and as a further step towards elucidating the complexity of the problem in general.

### 1.3 OTHER QUESTIONS

For completeness, we mention other approaches to the critical analysis of the capabilities of neural networks for "learning" objectives. One line of work deals with *sample complexity* questions, that is, the quantification of the amount of information (number of samples) needed in order to characterize a given unknown mapping. Some recent references to such work, establishing sample complexity results, and hence "weak learnability" in the PAC model, for neural nets, are the papers [4, 20, 11, 19]; the first of these references deals with networks that employ hard threshold activations, the second and third cover continuous activation functions of a type (piecewise polynomial) close to those used in this paper, and the last one provides results for networks employing the standard sigmoid activation function. A different perspective to learnability questions takes a numerical analysis or approximation theoretic point of view. There one asks questions such as *how many* hidden units are necessary in order to approximate well, that is to say, with a small overall error, an unknown function. This type of research ignores the training question itself, asking instead what is the best one could do, in this sense of overall error, if the best possible network with a given architecture were to be eventually found. Some recent papers along these lines are [2, 13, 7], which deal with single hidden layer nets –the last reference comparing different error measures– and [8], which dealt with multiple hidden layers. Finally, one may ask about the difficulties of steepest descent "backprop" methods due to local minima in error surfaces; see [25] and references there. A related question deals with the multiplicity of possible networks that represent the same concept (and hence give rise to multiple weights achieving same cost in the "backprop" objective function); for that see [26, 1].

### 1.4 ORGANIZATION OF THE PAPER

The rest of the paper is organized as follows:

- In section 2 we introduce the model, distinguish the case of fixed versus varying input dimension (and analog versus binary inputs) and state our results precisely. We also observe that the loading problem is polynomial-time when the input dimension is fixed.
- In section 3 we prove the hardness of the loading problem for the 2 $\pi$-node architecture and use this result to show the impossibility of learnability for binary inputs under the assumption of $RP \neq NP$.
- In section 4 we prove the hardness of the loading problem for the "Restricted" $(2, r)$ $(\pi, \mathcal{H})$-node architecture with binary inputs

Due to space limitations, details of some proofs are omitted; they can be found in [9].

## 2 PRELIMINARIES AND STATEMENT OF RESULTS

Let $\Phi$ be a class of real-valued functions, where each function is defined on some subset of $\mathbb{R}$. A $\Phi$-net $C$ is an unbounded fan-in directed acyclic graph. To each vertex $v$, an activation function $\phi_v \in \Phi$ is assigned, and we assume that $C$ has a single sink $w$. The network $C$ computes a function $f_C : [0,1]^n \to \mathbb{R}$ as follows. The components of the input vector $x = (x_1, \ldots, x_n) \in [0,1]^n$ are assigned to the sources of $C$. Let $v_1, \ldots, v_k$ be the immediate predecessors of a vertex $v$. The input for $v$ is then $s_v(x) = \sum_{i=1}^{k} a_i y_i - b_v$, where $y_i$ is the value assigned to $v_i$ and $a$ and $b$ are the weights of $v$. We assign the value $\phi_v(s_v(x))$ to $v$. Then $f_C = \phi_z(s_z)$ is the function computed by $C$ where $z$ is the unique sink of $C$.

The architecture $\mathcal{A}$ of the $\Phi$-net $C$ is the structure of the underlying directed acyclic graph. Hence each architecture $\mathcal{A}$ defines a behavior function $\beta_{\mathcal{A}}$ that maps from the $r$ real weights (corresponding to all the weights and thresholds of the underlying directed acyclic graph) and the input string into a binary output. We denote such a behavior as the function $\beta_{\mathcal{A}}(\mathbb{R}^r, [0,1]^n) \mapsto \{0,1\}$ . The set of inputs which cause the output of the network to be 0 (resp. 1) are termed as the set of *negative* (resp. *positive*) examples. The *size* of the architecture $\mathcal{A}$ is the number of nodes and connections of $\mathcal{A}$ plus the maximum number of bits needed to represent any weight of $\mathcal{A}$.

The *loading problem* is defined as follows: Given an architecture $\mathcal{A}$ and a set of positive and negative examples $M = \{(x, y) \mid x \in [0,1]^n, y \in \{0,1\}\}$, so that $|M| = O(n)$; find weights $\vec{w}$ so that for all pairs $(x, y) \in M$:

$$\beta_{\mathcal{A}}(\vec{w}, x) = y \ .$$

The decision version of the loading problem is to decide (rather than to find the weights) whether such weights exist that load $M$ onto $\mathcal{A}$.

The PAC-learning problem for a $\Phi$-net $C$ is as follows. Assume that $C$ computes a function $f$ and let $n \in \mathcal{N}$. Let $f^{-1}(0) = \{x \mid x \in \{0,1\}^n, f(x) = 0\}$ (resp. $f^{-1}(1) = \{x \mid x \in \{0,1\}^n, f(x) = 1\}$) denote the set of negative (resp. positive) examples. Let $C_n$ be the set of Boolean functions on $n$ variables computable by a specific architecture $\mathcal{A}$. Then $C = \cup_{i=1}^{\infty} C_n$ is a class of representations achievable by the architecture $\mathcal{A}$ for all binary input strings. Given some function $f \in C$, $POS(f)$ (resp. $NEG(f)$) denotes the source

of positive (resp. negative) examples for $f$. Whenever $POS(f)$ (resp. $NEG(f)$) is called, a positive or $'+'$ (resp. negative or $'-'$) example is provided according to some arbitrary probability distribution $D^+$ (resp. $D^-$) satisfying the conditions $\sum_{x=f^{-1}(1)} D^+(x) = 1$ (resp. $\sum_{x=f^{-1}(0)} D^-(x) = 1$). A *learning algorithm* is an algorithm that may access $POS(f)$ and $NEG(f)$. Each access to $POS(f)$ or $NEG(f)$ is counted as one step. A class $C$ of representations of an architecture $\mathcal{A}$ is said to be *learnable*[16] iff, for any given constants $0 < \epsilon, \delta < 1$, there is a learning algorithm $L$ such that for all $n \in \mathcal{N}$, all functions $f \in C_n$, and all possible distributions $D^+$ and $D^-$,

**(a)** $L$ halts in number of steps polynomial in $n$, $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, and $||\mathcal{A}||$ (where $||\mathcal{A}||$ denotes the size of the architecture $\mathcal{A}$),

**(b)** $L$ outputs a hypothesis $g \in C_n$ such that with probability at least $1-\delta$ the following conditions are satisfied: $\sum_{x \in g^{-1}(0)} D^+(x) < \epsilon \quad \sum_{x \in g^{-1}(1)} D^-(x) < \epsilon$.

In this paper we focus on 1 hidden layer (1HL) architectures and we will be concerned with two very simple architectures as described below. The $k$ $\Phi$-node architecture is a 1HL architecture with $k$ hidden $\phi$-units (for some $\phi \in \Phi$), and an output node with the threshold activation $\mathcal{H}$. The 2 $\Phi$-node architecture consists of two hidden nodes $N_1$ and $N_2$ that compute:

$$N_1(\vec{a}, \vec{x}) = \phi(\sum_{i=1}^{n} a_i x_i) \quad N_2(\vec{b}, \vec{x}) = \phi(\sum_{i=1}^{n} b_i x_i),$$

The output node $N_3$ computes the threshold function of the inputs received from the two hidden nodes, namely a binary threshold function of the form

$$N_3(N_1, N_2, \alpha, \beta, \gamma) = \begin{cases} 1 & \alpha N_1(\vec{a}, \vec{x}) + \beta N_2(\vec{b}, \vec{x}) > \gamma \\ 0 & \alpha N_1(\vec{a}, \vec{x}) + \beta N_2(\vec{b}, \vec{x}) \leq \gamma \end{cases}$$

for some parameters $\alpha, \beta$, and $\gamma$. Figure 1 illustrates a 2 $\Phi$-node architecture.

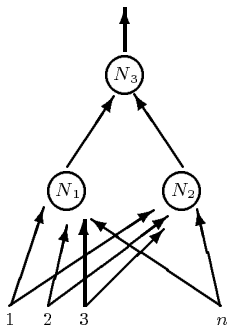


Figure 1: *A 2 $\Phi$-node architecture*

The two activations function classes $\Phi$ that we consider are the binary threshold functions $\mathcal{H}$ and the piecewise linear or "saturating" activation functions $\pi$ (which appears quite frequently in neural networks literature[3, 6, 18, 28]) defined as

$$\pi(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \leq x \leq 1 \\ 1 & \text{if } x > 1\,. \end{cases} \tag{2}$$

Inspired by Blum and Rivest[5] who considered loading a few variations of the $k$ $\mathcal{H}$-node network, in which all activations functions were discrete; the second architecture we consider is a 1HL-architecture in which the function of the output node is restricted. Consider a unit $G$ that computes $\mathcal{H}(\sum_{i=1}^{n} \alpha_i x_i - \eta)$, where $\alpha_i$'s are real constants and $x_1$ to $x_n$ are input variables which assume any real value in $[0, 1]$. Let $\alpha = \sum_{i=1}^{n} \alpha_i$.

We say that this unit $G$ computes a Boolean NAND (i.e., *negated* AND) function of its inputs provided its weights and threshold satisfy the following requirements:

$$\alpha_i < \eta < 0 \quad 1 \leq i \leq n \tag{3}$$

For justification, assume that the inputs to node $G$ are binary. Then, the output of $G$ is one iff *all* its inputs are zeroes.

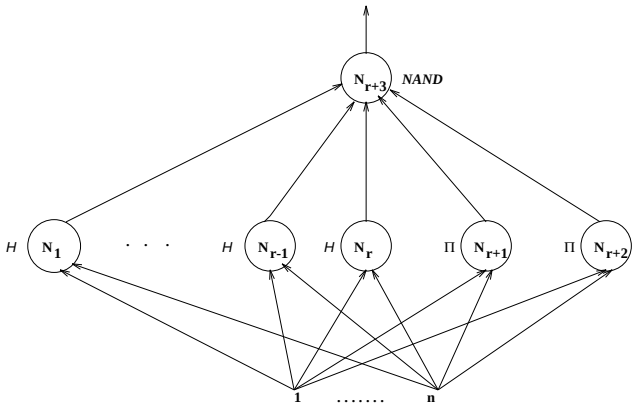


Figure 2: The "Restricted" $(2, r)$ ($\pi$, H)-node network.

Our second architecture consists of $r + 2$ hidden nodes $N_1, N_2, \ldots, N_{r+2}$ (where $r$ is a fixed polynomial in $n$, the number of inputs) and one output node. The nodes $N_1, N_2, \ldots, N_r$ in the hidden layer compute the binary threshold functions $\mathcal{H}$, and the two remaining hidden nodes $N_{r+1}$ and $N_{r+2}$ compute the "saturating activation" functions $\pi$ (equation 2). The output node $N_{r+3}$ computes a Boolean AND function. We term this as the *"Restricted" $(2, r)$ $(\pi, \mathcal{H})$-node* architecture.

We consider two kinds of inputs: *analog* and *binary*. An analog input is in $[0, 1]^d$, where $d$ is a fixed constant, also called the *input dimension*. In the binary case, the input is in $\{0, 1\}^n$ where $n$ is an input parameter.

### 2.1 FIXED INPUT DIMENSION

When the input is analog (and the dimension is hence constant), loading a $k$ $\pi$-node architecture or a $k$ $\mathcal{H}$-node architecture can be done in time polynomial on

the size of the training set. This result is very easy, and we include it here only for completeness. It is an immediate consequence of a result by Megiddo ([21]) (see also [20], Theorem 4.3). The above result for analog inputs can be generalized for the case of the "restricted" $(2, r)$ $(\pi, \mathcal{H})$-node architecture as well.

### 2.2 VARYING INPUT DIMENSION (AND BINARY INPUTS)

Blum and Rivest[5] showed when the inputs are binary and the training set is sparse the loading problem is NP-Complete for the 2 $\mathcal{H}$-node architecture. In another related paper, Lin and Vitter[17] proved a slightly stronger result by showing that the loading problem of 2-cascade threshold net with binary input is NP-complete.

In contrast to the case of fixed input dimension, we show in this section that the loading problem for the 2 $\pi$-node architecture is NP-complete when (binary) inputs of arbitrary dimension are considered. The main theorem of this section is as follows.

**Theorem 2.1** *The loading problem for the 2 $\pi$-node architecture (L$\pi$AP) with binary inputs is NP-complete.*

In fact, allowing more hidden units may still make the loading problem hard, if a suitable restriction on the function computed by the output unit is assumed, as illustrated by the following theorem.

**Theorem 2.2** *The loading problem for the "Restricted" $(2, r)$ $(\pi, \mathcal{H})$-node architecture with binary inputs is NP-complete.*

A corollary of the above theorems is as follows.

**Corollary 2.1** *The class of Boolean functions computable by the 2 $\pi$-node architecture or by the "Restricted" $(2, r)$ $(\pi, \mathcal{H})$-node architecture with binary inputs is not learnable, unless $RP = NP$.*

## 3 PROOF FOR HARDNESS OF LOADING FOR 2 $\pi$-NODE ARCHITECTURE WITH VARYING INPUT DIMENSION

To prove theorem 2.1 we reduce a restricted version of the set splitting problem, which is known to be NP-complete[10], to this problem in polynomial time. However, due to the continuity of this activation function, many technical difficulties arise. The proof is organized as follows:

1. Providing a geometric view of the problem [section 3.1].
2. Introducing the $(k, l)$-set splitting problem and the symmetric 2-SAT problem [section 3.2].
3. Proving the existence of a polynomial algorithm that transforms a solution of the (3,3)-set splitting problem into a solution of its associated (2,3)-set splitting problem (using the symmetric 2-SAT problem) [section 3.3].
4. Defining the 3-hyperplane problem and proving it is NP-complete by reducing from the (2,3)-set splitting problem [section 3.4].
5. Proving that the L$\pi$AP is NP-complete. This is done using all the above items[section 3.5].

### 3.1 A GEOMETRIC VIEW OF THE LOADING PROBLEM

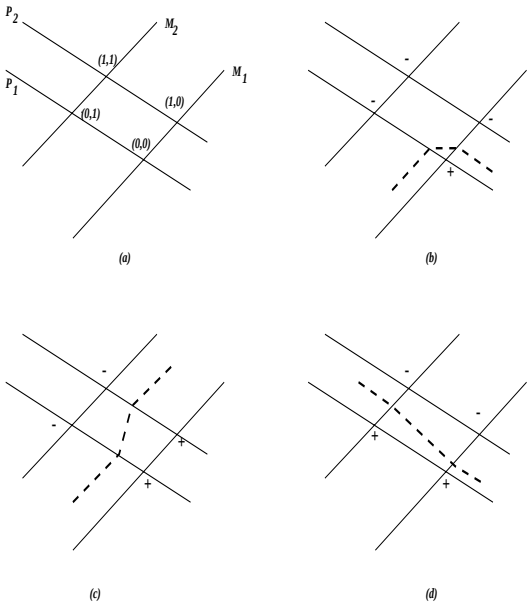


Figure 3: Different classifications produced by the 3-node network.

We start by categorizing the different types of classifications produced by the 2 $\pi$-node architecture. Without loss of generality we assume $\alpha, \beta \neq 0$. Consider the 4 hyperplanes $M_1 : \sum_{i=1}^{n} a_i x_i = 0$, $M_2 : \sum_{i=1}^{n} a_i x_i = 1$, $P_1 : \sum_{i=1}^{n} b_i x_i = 0$, and $P_2 : \sum_{i=1}^{n} b_i x_i = 1$ (refer to fig. 3). Let $(c_1, c_2)$ denote the set of points in the $(n-1)$-dimensional facet corresponding to $\sum_{i=1}^{n} a_i x_i = c_1$ and $\sum_{i=1}^{n} b_i x_i = c_2$. As all points belonging to one facet are labeled equally, we consider "labeling the facets" rather than the single points.

**Type 1.** All facets are labeled either $'+'$ or $'-'$. In that case, all the examples are labeled $'+'$ or $'-'$, respectively.

**Type 2.** Exactly one facet is labeled $'+'$. Assume that this facet is (0,0). Then, two different types of separations exist:

**(a)** There exist two halfspaces $H_1$ and $H_2$ such that all the $'+'$ points belong to $H_1 \wedge H_2$ and all the $'-'$ points belong to $\overline{H_1} \vee \overline{H_2}$ ($H_1$ and $H_2$ may be identical).

**(b)** There exist three hyperplanes of the following form (fig. 3(b)):

$$H_1 : \alpha(\sum_{i=1}^{n} a_i x_i) > \gamma \quad H_2 : \beta(\sum_{i=1}^{n} b_i x_i) > \gamma$$

$$H_3 : \sum_{i=1}^{n} (\alpha a_i + \beta b_i) x_i > \gamma$$

where $0 > \gamma$, $\alpha, \beta \leq \gamma < 0$ (hence $\gamma > 2\gamma$), and all the $'+'$ and $'-'$ points belong to $H_1 \wedge H_2 \wedge H_3$ and $\overline{H_1} \vee \overline{H_2} \vee \overline{H_3}$, respectively.

If any other facet is marked $'+'$, a similar separation is produced.

**Type 3.** Two facets are marked $'+'$ and the remaining two are labeled $'-'$. Because the labeling must be linearly separable, only the following types of classifications are possible:

**(a)** $(0,1)$ and $(0,0)$ are $'+'$ (fig. 3(d)). Then, the input space is partitioned via the three halfspaces:

$$H_1 : \alpha(\sum_{i=1}^{n} a_i x_i) > \gamma - \beta \quad H_2 : \alpha(\sum_{i=1}^{n} a_i x_i) > \gamma$$

$$H_3 : \sum_{i=1}^{n} (\alpha a_i + \beta b_i) x_i > \gamma$$

where $\beta > \gamma, \alpha \leq \gamma < 0$ and $\alpha + \beta \leq \gamma$. If $\beta < 0$ (resp. $\beta > 0$) then all the $'+'$ and $'-'$ points lie in $H_1 \vee (H_2 \wedge H_3)$ (resp. $H_2 \vee (H_1 \wedge H_3)$) and $\overline{H_2} \vee (\overline{H_1} \wedge \overline{H_3})$ (resp. $\overline{H_1} \vee (\overline{H_2} \wedge \overline{H_3})$), respectively.

**(b)** $(0,0)$ and $(1,0)$ are $'+'$ (fig. 3(c)). Then, the input space is partitioned via the three halfspaces:

$$H_1 : \beta(\sum_{i=1}^{n} b_i x_i) > \gamma - \alpha \quad H_2 : \beta(\sum_{i=1}^{n} b_i x_i) > \gamma$$

$$H_3 : \sum_{i=1}^{n} (\alpha a_i + \beta b_i) x_i > \gamma$$

where $\alpha > \gamma, \beta \leq \gamma < 0$ and $\alpha + \beta \leq \gamma$.
If $\alpha < 0$ (resp. $\alpha > 0$) then all the $'+'$ and $'-'$ points lie in $H_1 \vee (H_2 \wedge H_3)$ (resp. $H_2 \vee (H_1 \wedge H_3)$) and $\overline{H_2} \vee (\overline{H_1} \wedge \overline{H_3})$ (resp. $\overline{H_1} \vee (\overline{H_2} \wedge \overline{H_3})$), respectively.

**(c)** $(1,0)$ and $(1,1)$ are $'+'$ (similar to fig. 3(d) with the labeling of $'+'$ and $'-'$ points interchanged). This is the symmetrically opposite case of type 3(a).

**(d)** $(0,1)$ and $(1,1)$ are $'+'$ (similar to fig. 3(c) with the labeling of $'+'$ and $'-'$ points interchanged). This is the symmetrically opposite case of type 3(b).

**Type 4.** Three facets are labeled $'+'$. This case is symmetrically opposite to type 2, and thus details are precluded.

### 3.2 THE SET SPLITTING AND SYMMETRIC 2-SAT PROBLEMS

The following problem is referred to as the $(k,l)$-set splitting problem (SSP) for $k \geq 2$.

**INSTANCE:** A set $S = \{s_i \mid 1 \leq i \leq n\}$, and a collection $C = \{c_j \mid 1 \leq j \leq m\}$ of subsets of $S$, all of exactly size $l$.

**QUESTION:** Are there $k$ sets $S_1, \ldots, S_k$, such that $S_i \cap S_j = \phi$ for $i \neq j$, $\cup_{i=1}^{k} S_i = S$, and $c_j \not\subseteq S_i$ for $1 \leq i \leq k$ and $1 \leq j \leq m$?

Note that the $(k,l)$-SSP is solvable in polynomial time if both $k \leq 2$ and $l \leq 2$, but remains NP-complete if $k \geq 2$ and $l = 3$ (see [10]).

For later purposes we consider the *symmetric* 2-SAT problem:

**INSTANCE:** Variables $v_1, v_2, \cdots, v_n$ and a collection $D$ of one or two literal disjunctive clauses satisfying the condition: $\forall i, j \quad [(v_i \vee (\neg v_j)) \notin D]$ & $[((\neg v_i) \vee v_j) \notin D]$.

**QUESTION:** Decide whether there exists a satisfying assignment, and find one if exists.

In a manner similar to [23], we create a directed graph $G = (V, E)$, where where $V = \{d_i, \overline{d_i} \mid v_i$ is a variable$\}$, and $E = \{(l_i, l_j) \mid (i, j \in \{1, \ldots, n\})$, $(l_i \in \{d_i, \neg d_i\})$, $(l_j \in \{d_j, \neg d_j\})$, $(l_i \rightarrow l_j) \in D\}$. Note that an edge $(x, y)$ in $E$ is directed from $x$ to $y$. In the symmetric 2-SAT problem, the graph $G$ has the following crucial property:

(♣) Complemented and uncomplemented vertices alternate in any path. This is because the edges in $G$ are only of the form $(d_i, \overline{d_j})$ or $(\overline{d_i}, d_j)$ for some two indices $i$ and $j$ ($i = j$ is possible).

It is easy to design a polynomial-time algorithm that produces a satisfying assignment provided the following condition holds (see, for example, [23, pp. 377-378]):

> *The instance of the 2-SAT problem has a solution if and only if there is no directed cycle in $G$ which contains both the vertices $d_i$ and $\overline{d_i}$ for some $i$.*

It is easy to check the above condition in $O(\mid V \mid) = O(n)$ time by finding the strongly connected components of $G$. Hence, computing a satisfying assignment (or, reporting that no such assignment exists) can be done in time polynomial in the input size.

### 3.3 THE $(k,l)$-REDUCTION PROBLEM

We prove that under certain conditions, a solution of the $(k,l)$-set splitting instance $(S, C)$ can be transformed into a solution of the associated $(k-1, l)$-set splitting problem. More formally, we define the $(k,l)$-reduction problem ($(k,l)$-RP) as follows:

**INSTANCE:** An instance $(S, C)$ of the $(k,l)$-SSP, and a solution $(S_1, S_2, \ldots, S_k)$.

**QUESTION:** Decide whether there exists a solution $(S'_1, S'_2, \ldots, S'_{k-1})$ to the associated $(k-1, l)$-SSP and construct one if exists, where, for all $i, j \in \{1, 2, \ldots, k-1\}$ $i \neq j$:

$$S'_i = S_i \cup T_i \quad T_i \subseteq S_k \quad (T_i \cap T_j) = \phi \quad \cup_{p=1}^{k-1} T_p = S_k$$

We next state the existence of a polynomial algorithm for the $(3,3)$-reduction problem. Since we are interested in placing elements of $S_3$ in $S_1$ or $S_2$, we focus on sets having at least one element of $S_3$. Since $(S_1, S_2, S_3)$ is a solution of the $(3,3)$-SSP, no set contains 3 elements of $S_3$. Let $C' = \{c_j \mid 1 \leq i \leq m\} \subseteq C$ be the collection of sets which contain at least one element of $S_3$. Obviously, $\forall j (c_j \not\subseteq S_1) \wedge (c_j \not\subseteq S_2) \wedge (c_j \not\subseteq S_3)$.

Let $A = \{a_i \mid 1 \leq i \leq |S|\}$ and $B = \{b_i \mid 1 \leq i \leq |S|\}$ be two disjoint sets. Each element of $A \cup B$ is to be colored *'red'* or *'blue'* so that the overall coloring satisfies the *valid coloring conditions*:

**(a)** For each set $\{x_i, x_j, x_p\} \in C'$, where $x_i, x_j \in S_3$, at least one of $a_i$ or $a_j$ should be colored red if $x_p \in S_1$ and at least one of $b_i$ or $b_j$ has to be colored red if $x_p \in S_2$.

**(b)** For each $i$, $1 \leq i \leq |S|$, at least one of $a_i$ or $b_i$ has to be colored blue.

**(c)** For each set $\{x_i, x_j, x_p\}$ such that $x_p \in S_3$ and $x_i, x_j \in S_1$ (resp. $x_i, x_j \in S_2$), $a_p$ (resp. $b_p$) must be colored red.

**Theorem 3.1** *The following two statements are true:*

**(a)** *The $(3,3)$-reduction problem is polynomially solvable.*

**(b)** *If the $(3,3)$-RP has no solution, no valid coloring of $A \bigcup B$ exists.*

**Proof Idea.** Part (a) is proved by showing how to reduce the $(3,3)$-reduction problem in polynomial time to the symmetric 2-SAT and noting that the later problem is polynomially solvable. To prove part (b) construct the graph $G$ from the collection of clauses $D$ as described in section 3.2. If no satisfying assignment exists, the graph $G$ has a directed cycle containing both $d_i$ and $\overline{d_i}$ for some $i$. The proof can be completed by showing that in that case no valid coloring of all the elements of $A \cup B$ is possible. □

### 3.4 THE 3-HYPERPLANE PROBLEM

We prove the following problem, which we term as the 3-hyperplane problem (3HP), to be NP-complete.

**INSTANCE:** A set of points in an $n$-dimensional hypercube labeled $'+'$ and $'-'$.

**QUESTION:** Does there exist a separation of one or more of the following forms:

**(a)** A set of two halfspaces $\vec{a}\vec{x} > a_0$ and $H_2 : \vec{b}\vec{x} > b_0$ such that all the $'+'$ points are in $H_1 \wedge H_2$, and all the $'-'$ points belong to $\overline{H_1} \vee \overline{H_2}$?

**(b)** A set of 3 halfspaces $H_1 : \vec{a}\vec{x} > a_0$, $H_2 : \vec{b}\vec{x} > b_0$ and $H_3 : (\vec{a + b})\vec{x} > c_0$ such that all the $'+'$ points belong to $H_1 \wedge H_2 \wedge H_3$ and all the $'-'$ points belong to $\overline{H_1} \vee \overline{H_2} \vee \overline{H_3}$?

**Theorem 3.2** *The 3-hyperplane problem is NP-complete.*

**Proof.** We first notice that this problem is in NP as an affirmative solution can be verified in polynomial time. To prove NP-completeness of the 3HL, we reduce the (2,3)-set splitting problem to it. Given an instance $I$ of the (2,3)-SSP:

$$I: \quad S = \{s_i\}, C = \{c_j\}, c_j \subseteq S, \mid S \mid = n, \mid c_j \mid = 3 \text{ for all } j$$

we create the instance $I'$ of the 3-hyperplane problem (like in [5]):

> ⋆ The origin $(0^n)$ is labeled $'+'$; for each element $s_j$, the point $p_j$ having 1 in the $j^{th}$ coordinate only is labeled $'-'$; and for each clause $c_l = \{s_i, s_j, s_k\}$, we label with $'+'$ the point $p_{ijk}$ which has 1 in its $i^{th}$, $j^{th}$, and $k^{th}$ coordinates.

The theorem is proved by showing that an instance $I'$ of the 3-hyperplane problem has a solution if and only if instance $I$ of the (2,3)-SSP has a solution using Theorem 3.1 and some additional arguments. □

### 3.5 LOADING THE 2 $\pi$-NODE ARCHITECTURE IS NP-COMPLETE

Next, we prove that loading the 2 $\pi$-node architecture is NP-complete. We do so by comparing it to the 3-hyperplane problem. To this end, we construct a gadget that will allow the architecture to produce only separations of type 2 (section 3.1), which are similar to those of the 3HP.

We construct such a gadget with two steps: first, in Lemma 3.1, we exclude separation of type 3, and then we exclude in separations of type 4 in Lemma 3.2. Their proofs are omitted in this abstract.

**Lemma 3.1** *Consider the 2-dimensional hypercube in which (0,0), (1,1) are labeled $'+'$, and (1,0), (0,1) are labeled $'-'$. Then the following statements are true:*

**(a)** *There do not exist three halfspaces $H_1$, $H_2$, $H_3$ as described in type 3(a)-(d) in section 3.1 which correctly classify this set of points.*

**(b)** *There exist two halfspaces of the form $H_1 : \vec{a}\vec{x} > a_0$ and $H_2 : \vec{b}\vec{x} > b_0$, where $a_0, b_0 < 0$, such that all the $'+'$ and $'-'$ points belong to $H_1 \wedge H_2$ and $\overline{H_1} \vee \overline{H_2}$, respectively.*

**Lemma 3.2** *Consider the labeled set A: (0,0,0), (1,0,1), (0,1,1) are labeled '+', and (0,0,1), (0,1,0), (1,0,0), (1,1,1) are labeled '−'. Then, there does not exist a separation of these points by type 4 halfspaces as described in section 3.1.*

**Proof of theorem 2.1.** First we observe that the problem is in NP as follows. The classifications of the labeled points produced by the 2 $\pi$-node architecture (as discussed in section 3.1) are 3-polyhedrally separable. Hence, from the result of [22] one can restrict all the weights to have at most $O(n \log n)$ bits. Thus, a "guessed" solution can be verified in polynomial time.

Next, we show that the problem is NP-complete. Consider an instance $I = (S, C)$ of the (2,3)-SSP. We transform it into an instance $I'$ of the problem of loading the 2 $\pi$-node architecture as follows: we label points on the $(|S| + 5)$ hypercube similar to as is $\star$ (section 3.4).

> The origin $(0^{|S|+5})$ is labeled '+'; for each element $s_j$, the point $p_j$ having 1 in the $j^{th}$ coordinate only is labeled '−'; and for each clause $c_l = \{s_i, s_j, s_k\}$, we label with '+' the point $p_{ijk}$ which has 1 in its $i^{th}$, $j^{th}$, and $k^{th}$ coordinates. The points $(0^n, 0, 0, 0, 0, 0)$, $(0^n, 0, 0, 0, 1, 1)$, $(0^n, 1, 0, 1, 0, 0)$ and $(0^n, 0, 1, 1, 0, 0)$ are marked '+', and the points $(0^n, 0, 0, 0, 1, 0)$, $(0^n, 0, 0, 0, 0, 1)$, $(0^n, 0, 0, 1, 0, 0)$, $(0^n, 0, 1, 0, 0, 0)$, $(0^n, 1, 0, 0, 0, 0)$ and $(0^n, 1, 1, 1, 0, 0)$ are labeled '−'.

To complete the proof we need to show that a solution for $I$ exists iff the given architecture correctly classifies all the '+' and '−' points of the instance $I'$. This can be done using Lemma 3.1, Lemma 3.2, the results in [5] and the different types of classifications produced by this architecture as described in section 3.1. □

**Remark 3.1** *From the above proof of theorem 2.1 it is clear that the NP-completeness result holds even if all the weights are constrained to lie in the set $\{-2, -1, 1\}$. Thus the hardness of the loading problem holds even if all the weights are "small" constants.*

# 4 PROOF FOR THE HARDNESS OF LOADING FOR THE "RESTRICTED" $(2, r)$ ($\pi$, H)-NODE ARCHITECTURE WITH VARYING INPUT DIMENSION

In this section we discuss the proof of Theorem 2.2. Before proving Theorem 2.2 we show, given an instance $I$ of the $(2, 3)$-SSP, how to construct an instance $I'$ of the $(r + 2, 3)$-SSP such that $I$ has a solution iff $I'$ has one.

Let $I = (S, C)$ be a given instance of the $(2, 3)$-SSP. We construct $I'$ by adding $2r + 2$ new elements $Y = \{y_i \mid 1 \leq i \leq 2r + 2\}$ and creating the following new sets:

- Create the sets $\{s_i, y_j, y_k\}$ for all $1 \leq i \leq n$, $1 \leq j, k \leq 2r + 2$, $j \neq k$. This ensures that if a set in a solution of the set-splitting problem contains an element of $S$, it may contain at most one more element of $Y$.
- Create the sets $\{y_i, y_j, y_k\}$ for all $1 \leq i, j, k \leq 2r+2$, $i \neq j \neq k$. This ensures that no set in a solution of the set-splitting problem may contain more than two elements of $Y$.

Let $I' = (S'.C')$ be the new instance of the $(r + 2, 3)$-SSP, where $S' = S \cup Y$, and $C'$ contains all the sets of $C$ and the additional sets as described above.

**Lemma 4.1** *The instance $I'$ of the $(r + 2, 3)$-SSP has a solution if and only if the instance $I$ of the $(2, 3)$-SSP has a solution.*

**Proof.**

$\Leftarrow$

Let $(S_1, S_2)$ be a solution of $I$. Then, a solution $(T_1, T_2, \cdots, T_{r+2})$ of the instance $I'$ is as follows:

$$T_i = \{y_{2i-1}, y_{2i}\} \quad \text{for} \quad 1 \leq i \leq r$$

$$T_{r+1} = S_1 \cup \{y_{2r+1}\}$$

$$T_{r+2} = S_2 \cup \{y_{2r+2}\}$$

$\Rightarrow$

Let $(T_1, T_2, \cdots, T_{r+2})$ be a solution of $I'$.

**Case 1.** There are at most two sets of $T_1, T_2, \cdots, T_{r+2}$ which contain all the elements of $S$. Then these two sets constitute a solution of $I$.

**Case 2.** Otherwise, there are $m$ ($m \geq 3$) sets, $T_1, \ldots, T_m$, each containing a distinct element of $S$. At most one element of $Y$ occurs in each $T_i$ (since two elements of $Y$ cannot be in the same set with an element of $S$ without violating the set-splitting constraint), hence $m < r + 2$. So, there are $r + 2 - m$ remaining sets in the solution of the instance $I'$ and at least $2r + 2 - m$ elements of $Y$ to be placed in those sets. By the pigeonhole principle, one of these remaining $r + 2 - m$ sets must contain at least three elements of $Y$ (since $m \geq 3$), thus violating the set-splitting constraint. So, case 2 is not possible. □

**Proof of Theorem 2.2.** The '+' and '−' points are $(r+3)$-polyhedrally separated by the output of the network in which the Boolean formula for the polyhedral separation is the formula for the NAND function. Hence,

from the result of [22] we can restrict all the weights to have at most $p(n+r)$ number of bits (where $p(x)$ is some polynomial in $x$). Since $r$ is a polynomial in $n$, any "guessed" solution may be verified in polynomial time. So, the problem is in NP.

We next show that the problem is NP-complete. Given an instance $I$ of the $(2,3)$-SSP, we construct an instance $I'$ of the "Restricted" $(2,r)$ $(\pi,\mathcal{H})$-node architecture as follows. We create first an instance $I''$ of the $(r+2,3)$-SSP (see Lemma 4.1). We then add the following labeled points, thus constructing the associated instance $I'$.

> The instance $I'$ is the architecture along with the following set of points: The origin $(0^{|S'|})$ is labeled $'+'$; for each element $s_j \in S'$, the point $p_j$ having 1 in the $j^{th}$ coordinate only is labeled $'-'$; and for each clause $c_l = \{s_i, s_j, s_k\} \in C'$, we label with $'+'$ the point $p_{ijk}$ which has 1 in its $i^{th}$, $j^{th}$, and $k^{th}$ coordinates.

$\Leftarrow$
Given a solution $(S_1, S_2)$ of $I$, we construct a solution $(T_1, T_2, \cdots, T_{r+2})$ of $I''$ as described in the proof of Lemma 4.1. Consider the following $r+2$ halfspaces:

$$H_i : \sum_{j=1}^{n} \delta_{i,j} x_j > -\frac{1}{2} \quad (1 \le i \le r+2)$$

where,

$$\delta_{i,j} = \begin{cases} -1 & \text{if } s_j \in T_i \\ 2 & \text{otherwise} \end{cases}$$

All labeled points of $I'$ are separated by these halfspaces: the $'+'$ points lie in $\wedge_{i=1}^{r+2} H_i$ and the $'-'$ points lie in $\vee_{i=1}^{r+2} \overline{H_i}$.

We map the $r+2$ halfspaces to the "Restricted" $(2,r)$ $(\pi,\mathcal{H})$-node architecture as follows. The hidden nodes compute:

$$\begin{aligned} N_i &= \mathcal{H}[-(\sum_{j=1}^{n} \delta_{i,j} x_j)] \qquad i = 1 \ldots r \\ N_{r+1} &= \pi[-(\sum_{j=1}^{n} \delta_{r+1,j} x_j)] \\ N_{r+2} &= \pi[-(\sum_{j=1}^{n} \delta_{r+2,j} x_j)] \ , \end{aligned}$$

and the output node $N_{r+3}$ computes:

$$N_{r+3} = \begin{cases} 1 & \text{if } -(\sum_{i=1}^{r+2} N_i) > -\frac{1}{2} \\ 0 & \text{if } -(\sum_{i=1}^{r+2} N_i) \le -\frac{1}{2} \end{cases}$$

$\Rightarrow$
Conversely, given a solution to the instance $I'$, we construct a solution to $I$. The classification produced by the "Restricted" $(2,r)$ $(\pi,\mathcal{H})$-node architecture is as follows. Each hidden threshold node $N_i$ $(1 \le i \le r)$ defines a halfspace $H_i$:

$$H_i : \sum_{j=1}^{n} \delta_{i,j} x_j > \eta_i$$

for some real numbers $\delta_{i,1}, \cdots, \delta_{i,n}$ and $\eta_i$. From the classifications produced by the 2 $\pi$-node architecture as described in section 3.1 and since the output node $N_{r+3}$ computes a Boolean NAND function, there are at most $j$ halfspaces (for $2 \le j \le 3$) corresponding to the nodes $N_{r+1}$ and $N_{r+2}$:

$$H_{r+1} : \sum_{i=1}^{n} a_i x_i > a_0$$

$$H_{r+2} : \sum_{i=1}^{n} b_i x_i > b_0$$

$$H_t : \sum_{i=1}^{n} (a_i + b_i) x_i > c_0 \quad (r+3 \le t \le r+j)$$

All the $'+'$ points lie in $\wedge_{i=1}^{r+j} H_i$, and all the $'-'$ points lie in $\vee_{i=1}^{r+j} \overline{H_i}$.

Let $T_i$ be the $'-'$ points separated from the origin by the halfspace $H_i$ of the output of the network (for $1 \le i \le r+j$, $2 \le j \le 3$). No $T_i$ contains three elements of the same set of the instance $I'$, otherwise the set itself will be in $T_i$ as well, contradicting its positive labeling. Consider the sets $T_i$ as the solution of the instance $I''$. By Theorem 3.2 the sets $T_{r+1}$ to $T_{r+j}$ can be combined to 2 sets, say $T'_{r+1}$ and $T'_{r+2}$, without violating the set-splitting constraints. Hence, we have $r+2$ sets, $T_1, T_2, \cdots, T_r, T'_{r+1}, T'_{r+2}$, as the $r+2$ solution sets for the instance $I''$ which satisfy the set-splitting constraint. Hence, by Lemma 4.1 we can construct the two solution sets $(S_1, S_2)$ of the instance $I$. □

**Remark 4.1** *The proof of Corollary 2.1 follows by using the Theorems 2.1 and 2.2 and a technique used in the proof of Theorem 9 in [16].*

## References


[1] Albertini, F., E.D. Sontag, and V. Maillot, "Uniqueness of weights for neural networks," in *Artificial Neural Networks for Speech and Vision* (R. Mammone, ed.), Chapman and Hall, London, 1993, pp. 115-125.

[2] Barron, A.R., "Approximation and estimation bounds for artificial neural networks", *Proc. 4th Annual Workshop on Computational Learning Theory*, Morgan Kaufmann, 1991, pp. 243-249. See also "Universal Approximation Bounds for Superpositions of a Sigmoidal Function," *IEEE Trans. Info. Theory* **39**(1993): 930-945.

[3] Batruni, R., "A multilayer neural network with piecewise-linear structure and back-propagation learning," *IEEE Transactions on Neural Networks* **2**(1991): 395-403.

[4] Baum, E.B., and Haussler, D., "What size net gives valid generalization?," *Neural Computation,* **1**(1989): 151-160

[5] Blum, A., and Rivest, R. L., "Training a 3-Node Neural Network is NP-Complete," *Neural Networks,* **5**(1992): 117-127.

[6] Brown, J., M. Garber, and S. Vanable, "Artificial neural network on a SIMD architecture," in *Proc. 2nd Symposium on the Frontier of Massively Parallel Computation*, Fairfax, VA, 1988, pp. 43-47.

[7] Darken, C., Donahue, M., Gurvits, L., and Sontag, E., "Rate of approximation results motivated by robust neural network learning," *Proc. 6th ACM Workshop on Computational Learning Theory*, Santa Cruz, July 1993, pp. 303-309.

[8] DasGupta, B., and Schnitger, G., "The power of approximating: a comparison of activation functions," in *Advances in Neural Information Processing Systems 5* (Giles, C.L., Hanson, S.J., and Cowan, J.D., eds), Morgan Kaufmann, San Mateo, CA, 1993, pp. 615-622.

[9] DasGupta, B., Siegelmann, H. T., and Sontag, E., "On the Complexity of Training Neural Networks with Continuous Activation Functions", Tech Report # 93-61, Department of Computer Science, University of Minnesota, September, 1993.

[10] Garey, M. R., and Johnson, D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H.Freeman and Company, San Francisco, 1979.

[11] Goldberg, P., and Jerrum, M., "Bounding the Vapnik-Chervonenkis dimension of concept classes parametrized by real numbers," *Proc. 6th ACM Workshop on Computational Learning Theory*, Santa Cruz, July 1993, pp. 361-369.

[12] Höffgen K-U., "Computational limitations on training sigmoidal neural networks," *Information Processing Letters*, 46(1993), pp. 269-274.

[13] Jones, K.L., "A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training," *Annals of Statistics*, **20**(1992): 608-613.

[14] Judd, J.S., "On the complexity of learning shallow neural networks," *J. of Complexity,* **4**(1988): 177-192.

[15] Judd, J.S., *Neural Network Design and the Complexity of Learning*, MIT Press, Cambridge, MA, 1990.

[16] Kearns, M., Li, M., Pitt, L., and Valiant, L., "On the learnability of Boolean formulae," *Proc. of the 19th ACM Symp. Theory of Computing*, 1987, pp. 285-295.

[17] Lin, J-H., and Vitter, J. S., "Complexity results on learning by neural networks," *Machine Learning,* **6**(1991): 211-230.

[18] Lippmann, R., "An introduction to computing with neural nets," *IEEE Acoustics, Speech, and Signal Processing Magazine*, 1987, pp. 4-22.

[19] Macintyre, A., and Sontag, E. D., "Finiteness results for sigmoidal 'neural' networks," *Proc. 25th Annual Symp. Theory Computing*, San Diego, May 1993, pp. 325-334.

[20] Maass, W., "Bounds for the computational power and learning complexity of analog neural nets," *Proc. of the 25th ACM Symp. Theory of Computing*, May 1993, pp. 335-344 .

[21] Megiddo, M., "On the complexity of polyhedral separability," *Discrete Computational Geometry,* **3**(1988): 325-337.

[22] Muroga, S., *Threshold Logic and its Applications,* John Wiley & Sons Inc., 1971.

[23] Papadimitriou, C.H., and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, 1982.

[24] Sontag, E.D., "Feedforward nets for interpolation and classification," *J. Comp. Syst. Sci.,* **45**(1992): 20-48.

[25] Sontag, E.D. and H.J. Sussmann, "Backpropagation can give rise to spurious local minima even for networks without hidden layers," *Complex Systems* **3**(1989): 91-106.

[26] Sussmann, H.J., "Uniqueness of the weights for minimal feedforward nets with a given input-output map," *Neural Networks* **5**(1992): 589-593.

[27] Zhang, B., L. Zhang, and H. Zhang, "A quantitative analysis of the behavior of the PLN network," *Neural Networks* **5**(1992): 639-661.

[28] Zhang, X-D., "Complexity of neural network learning in the real number model," preprint, Comp. Sci. Dept., U. Mass., 1992.