DIMACS Technical Report 99-60
November 1999

Hardness of Approximation of the Loading Problem for Multi-layered Feedforward Neural Nets

by

Bhaskar DasGupta[1,2]
Dept. of Computer Science
Rutgers University
Camden, New Jersey 08102, USA
Email: bhaskar@crab.rutgers.edu

Barbara Hammer
Department of Mathematics/Computer Science
University of Osnabrück
Germany
Email: hammer@informatik.uni-osnabrueck.de

[1]DIMACS Visitor
[2]Research supported by NSF grant CCR-9800086.

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, Telcordia Technologies (formerly Bellcore) and NEC Research Institute.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

In this paper we deal with the problem of efficient learning of feedforward neural networks. First, we consider the case when the objective is to maximize the ratio r of the correctly classified points compared to all points in a training set (e.g., see [2]). Using techniques of [2], we show that, for an arbitrary *multilayered threshold network* with varying input dimension and with a constant $n_1 \geq 2$ neurons in the first hidden layer, it is NP-hard to approximate r within a relative error of at most $\varepsilon = 1/(68n_1 2^{n_1} + 136n_1^3 + 136n_1^2 + 170n_1)$. If $n_1 \geq 3$, then the above result is true with $\varepsilon = c(n_1 - 1)/(5n_1^3 + 2n_1^3 2^{n_1} + 4n_1^5 + 4n_1^4)$ for some positive constant c, *even if* restricted to situations where a solution *without any* misclassifications exists. Restricted to architectures with *only one* hidden layer and *two hidden neurons* (as in, for example, [6]) approximation of r with relative error at most $1/c$ is NP-hard even if either (a) $c = 2244$, the threshold activation function in the hidden layer is substituted by the classical sigmoid function, and the situation of ϵ-separation of the output classification is assumed, or (b) $c = 2380$ and the threshold activation function is substituted by the semilinear activation function commonly used in the neural net literature (e.g., see [3, 7, 8, 16]). For a single hidden layer threshold network with varying input dimension and k hidden nodes, approximating r within a relative error of at most c/k^5, for some positive constant c, is NP-hard even if restricted to situations where the number of examples is at most k^4. Finally, we consider the case when the objective is to minimize the *failure ratio* f in the presence of missclassification errors (e.g., see [1]). We show that it is NP-hard to approximate f within *any* constant $c > 1$ for a multilayered threshold network if the input biases are *fixed to zero.*

1 Introduction

Neural networks are a well established learning mechanism which offer a very simple method of learning an unknown hypothesis given some examples. In addition to their success in various areas of application, the possibility of massive parallelism and their noise and fault tolerance are offered as a justification for their use. However, the inherent complexity of training neural network is till now an open problem for almost all practically relevant situations. Until now it is only known that training a fixed network as it appears in practice is at least decidable if the Schanuel conjecture in number theory holds [18]. Starting with the work of Judd [14] one has considered situations where architectural parameters are allowed to vary in order to take into account that most existing training algorithms are uniform with respect to the architecture. It turns out that training is NP-hard in several situations. However, most work in this area deals either with only very restricted architectures, an activation function not used in practice, or a training problem which is too strict compared to practical problems. In this paper we want to consider situations which are closer to the training problems as they occur in practice. Hence we consider the training of feedforward neural networks with various activation functions, various architectures, and a somewhat weaker condition than the original so-called loading problem [14].

Here we give an informal description of the problem. A *feedforward neural network* consists of many neurons which are connected in a directed acyclic graph. The overall behavior is determined by the interconnetion structure, i.e. the *architecture* $\mathcal{A}$, and the behavior of the single neurons, i.e. the network *parameters* $\boldsymbol{w}$. Given a *pattern set* P, i.e. a set of points $(\boldsymbol{x}_i; y_i)$, we want to learn the regularity consistent with the mapping of the $\boldsymbol{x}_i$ points to the y_i points with a neural network. Frequently, this is performed by first chosing an architecture $\mathcal{A}$ which computes a function $\beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x})$ which depends on the input $\boldsymbol{x}$ and the network parameters $\boldsymbol{w}$. In a second step, the parameters $\boldsymbol{w}$ are chosen such that $\beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}_i) = y_i$ holds for every pattern $(\boldsymbol{x}_i; y_i)$ in the training set. The *loading problem* is the problem to find weights $\boldsymbol{w}$ for $\mathcal{A}$ such that these equalities hold for every pattern in the set P. The *decision version* of the loading problem is to decide (rather than to find the weights) whether such weights exist that load M onto $\mathcal{A}$.

In order to estimate the inherent complexity of training with neural networks, the complexity of the loading problem is an interesting subject; obviously, the decision version of the loading problem is at least as hard as finding optimal weights.

There are a few results which show that this loading problem is NP-hard unless we consider only one fixed architecture or a specific situation. For every fixed architecture with the threshold activation function training is polynomial which is basically included in Megiddos algorithm for polynomial separability [8, 10, 19]. For some strange activation functions or a setting where the number of examples is appropriately restricted with respect to the number of hidden neurons loadability is trivial in the sence that every pattern set can be loaded [23]. However, Blum and Rivest [6] show that a varying input dimension yields the NP-hardness of training neural nets for the threshold activation function and an architecture with only two hidden neurons. Hammer [10] states a generalization of this result to arbitrary multilayered networks with the threshold activation function. References [8, 11, 13, 22, 25] constitute approaches in order to generalize this result to architectures with a continuous or the standard sigmoidal activation function. Hence finding an optimum weight setting in a concrete learning task may require a large amount of time.

Naturally, the constraint of the loading problem that all the examples must be satisfied is too strict. In a practical situation, one would be satisfied if a large fraction (but not necessarily all) of the examples can be satisfied. Moreover, in the context of agnostic learning, it may be possible that there are no choices for the weights and thresholds for a particular architecture that loads a given set of examples. From these motivations, researchers have considered a modified version of the loading problem where the number of correct classified points is to be maximized. References [1, 12] consider the complexity of training single neurons with the threshold activation with some error ratio. Bartlett and Ben-David [2] deals with a more complex network (depth 2) structure and the threshold activation function. Formally, the maximization version of the loading problem (e.g., see [2]) L deals with an objective function m_L where $m_L = \mid \{\boldsymbol{x} \mid \beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}) = y\} \mid / |P|$ which is to be maximized. That is, we need to satisfy the largest possible fraction of the given set of examples. We will consider this objective function first and obtain NP-hardness results for approximately minimizing the relative error of m_L which deal with realistic multilayered architectures (as opposed to just networks with one hidden layer in [2]) or various more realistic activation functions compared to [2]. In the second part of the paper, we consider another possible objective function for maximization which is used in [1] and which is more relevant in the context where a significant number of examples are not classified correctly. This is called the *failure ratio* of the network, i.e., the ratio of the number of misclassifications produced by the learning algorithm to that of an optimal algorithm. We will obtain a result of NP-hardness of approximating this failure ratio for multilayered networks with threshold activation function in this latter part of the paper.

The organization of the rest of the paper is as follows. First, in Section 2 we define the basic model and notations. Next, we consider the complexity of minimizing the relative error of the objective function m_L of a neural network within some bound. For this purpose, following the approach in [2], we show in Section 3.1 that a certain type of reduction from the MAX-k-cut problem to the loading problem constitutes an L-reduction, thereby preserving the hardness of approximation. Afterwards we apply this method to several situations in the remaining part of Section 3. In Section 3.2.1, we show that, for an arbitrary *multilayered threshold network* with varying input dimension and with a constant $n_1 \geq 2$ neurons in the first hidden layer, it is NP-hard to approximate m_L within a relative error of at most $\varepsilon = 1/(68n_1 2^{n_1} + 136n_1^3 + 136n_1^2 + 170n_1)$, and if $n_1 \geq 3$, then the above result is true with $\varepsilon = c(n_1 - 1)/(5n_1^3 + 2n_1^3 2^{n_1} + 4n_1^5 + 4n_1^4)$ for some positive constant c, *even if* restricted to situations where a solution *without any* misclassifications exists. In Section 3.2.2, we show that for architectures with *only one* hidden layer and *two hidden neurons* (the classical case considered by Blum and Rivest [6]) approximation of m_L with relative error at most $1/c$ is NP-hard even if either (a) $c = 2244$, the threshold activation function in the hidden layer is substituted by the classical sigmoid function, and the situation of ϵ-separation is considered, or (b)

$c = 2380$ and the threshold activation function is substituted by the semilinear activation function commonly used in the neural net literature (e.g., see [3, 7, 8, 16]). As in [2] the above reductions use example sets where some of the examples occur more than once. In Section 3.3, we discuss how these multiplicities can be avoided. In Section 3.4, we consider the situation where the number of examples is restricted with respect to the number of hidden neurons, and show that for a single hidden layer threshold network with varying input dimension and k hidden nodes, approximating m_L within a relative error of at most c/k^5, for some positive constant c, is NP-hard even if restricted to situations where the number of examples is at most k^4. Next, in Section 4 we consider the case when the objective is to minimize the *failure ratio* f in the presence of missclassification errors (e.g., see [1]) and show that it is NP-hard to approximate f within *any* constant $c > 1$ for a multilayered threshold network if the thresholds of the input neurons are zero. Finally, we conclude in Section 5 with some future open problems worth investigating.

2 The Basic Model and Notations

The architecture of a feedforward net $\mathcal{C}$ is described by a directed interconnection graph and the activation functions of $\mathcal{C}$. A node (processor or neuron) v of $\mathcal{C}$ computes a function

$$\gamma_v\left(\sum_{i=1}^{k} w_{v_i,v}u_{v_i} + b_v\right)$$

of its inputs $u_{v_1}, \ldots, u_{v_k}$. The term $\sum_{i=1}^{k} w_{v_i,v}u_{v_i} + b_v$ is called the *activation* of the neuron v. These inputs are either external (i.e., representing the input data) or internal (i.e., representing the outputs of the immediate predecessors of v). The coefficients $w_{v_i,v}$ (resp. b_v) are the *weights* (resp. *threshold*) of node v, and the function γ_v is the *activation function* of v. No cycles are allowed in the interconnection graph of $\mathcal{C}$ and the output of a designated node provides the output of the network. An *architecture* specifies the interconnection structure and the γ_v's, but not the actual numerical values of the weights or thresholds. The *depth* of a feedforward net is the length (number of nodes) of the longest path of the (acyclic) interconnection graph, and the *size* is the number of nodes. A *layered* feedforward neural net is one in which nodes at depth d are connected only to nodes at depth $d+1$, and all inputs are provided to nodes at depth 1 only. In other words, a layered feedforward neural net consists of a set of neurons which are arranged in layers in a directed acyclic graph such that each neuron of a layer is connected exactly to each neuron in the following layer and all inputs are provided to nodes in the first layer only. A layered $(n_0, n_1, \ldots, n_h)$ net is a layered net with n_i nodes at depth $i \geq 1$ where n_0 is the number of inputs, note that we assume $n_h = 1$ in the following. Nodes at depth j, for $1 \leq j < h$, are called *hidden nodes*, and all nodes as depth j, for a particular j with $1 \leq j < h$, constitute the j^{th} *hidden layer*. For simplicity, we will sometimes refer to the inputs as input nodes.

To emphasize the selection of activation functions we introduce the concept of Γ-nets for a class Γ of activation functions. A Γ-net $\mathcal{C}$ is a feedforward neural net in which only functions in Γ are assigned to nodes. We assume that each function in Γ is defined on some subset of $\mathbf{R}$. Hence each architecture $\mathcal{A}$ of a Γ-net 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 n inputs into an output value. We denote such a behavior as the function $\beta_{\mathcal{A}} \ : \ \mathbf{R}^{r+n} \mapsto \mathbf{R}$. Some popular choices of the activation functions are the threshold activation function $H(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$ and the standard sigmoid $\mathrm{sgd}(x) = 1/(1 + \mathrm{e}^{-x})$.

In the learning context, the *loading problem* (e.g., see [8]) L is defined as follows: Given an architecture $\mathcal{A}$ and a set of examples $P = \{(\boldsymbol{x}; y) \mid x \in \mathbf{R}^n, y \in \mathbf{R}\}$, find weights $\boldsymbol{w}$ so that for all pairs $(\boldsymbol{x}; y) \in M$:

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

In this paper we will deal with classification tasks where $y \in \{0, 1\}$ instead of the more general case $y \in \mathbf{R}$. Clearly, the hardness results obtained with this restriction will be valid in the unrestricted case also. In this restricted case an example $(\boldsymbol{x}; y)$ is called a *positive example* if $y = 1$, otherwise it is called a *negative example*. An example is *misclassified* by the network if $\beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}) \neq y$, otherwise it is *classified correctly*.

In general, a *maximization* problem C is characterized by a non-negative objective function $m_C(x, y)$, where x is an input instance of the problem, y is a solution for x, and $m_C(x, y)$ is the cost of the solution y; the goal of such a problem is to maximize $m_C(x, y)$ for any particular x. Denote by $\mathrm{opt}_C(x)$ (or shortly $\mathrm{opt}(x)$ if the problem C is clear from the context) the maximum value of $m_C(x, y)$. The *relative error* of a solution y is the quantity $(\mathrm{opt}_C(x) - m_C(x, y))/\mathrm{opt}_C(x)$. The two objective functions that are of relevance to this paper are as follows:

Success ratio function m_L: Here $m_L(x, y) = \mid \{\boldsymbol{x} \ \mid \ \beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}) = y\} \mid \ / \ |P|$ (e.g., see [2]). That is, m_L is the fraction of the correctly classified points compared to all points in a training set. Notice that $0 < \mathrm{opt}_L(x) \leq 1$ for all x.

Failure ratio function m_f: Define $m_C(x, y) = \mid \{\boldsymbol{x} \ \mid \ \beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}) \neq y\} \mid$. Then, $m_f(x, y) = m_C(x, y)/\mathrm{opt}_C(x)$ provided $\mathrm{opt}_C(x) > 0$ (e.g., see [1]). That is, m_f is the ratio of the number of misclassified points by the given network to the minimum possible number of misclassifications when at least one missclassification is unavoidable.

In the sequel, when we say that a neuron "computes an AND", we mean that it computes the function $\wedge : \{0, 1\}^d \rightarrow \{0, 1\}$, $\wedge(\boldsymbol{x}) = 1 \iff \boldsymbol{x} = (1^d)$ where d is the number of inputs to the neuron. For a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, x_i occupies position i of $\boldsymbol{x}$ and is referred to as the i^{th} component (or, component i) of $\boldsymbol{x}$.

3 Hardness of Approximating the Success Ratio Function m_L

Following the approach in [2], we first show in Section 3.1 that a certain type of reduction from the MAX-k-cut problem to the loading problem constitutes an L-reduction, thereby preserving the hardness of approximation. Afterwards we apply this method to several situations in the remaining part of this section and also discuss how multiplicities in pattern sets can be avoided.

3.1 A General Theorem

We want to show that in several situations it is difficult to approximate m_L for a loading problem L. These results would extend and generalize the results of Bartlett and Ben-David [2] for more complex networks and for more complex activation functions such that several realistic situations are covered. For this purpose, an L-reduction from the so-called MAX-k-cut problem to the loading problem is constructed. This reduction shows the NP-hardness of approximability of the latter problem since it is known that approximating the MAX-k-cut problem is NP-hard and an L-reduction preserves approximability.

Formally, the MAX-k-cut problem is defined as follows.

Definition 1 *Given an undirected graph $G = (V, E)$ and a positive integer $k \geq 2$, the MAX-k-cut problem is to find a function $\psi : V \mapsto \{1, 2, \ldots, k\}$, such that the ratio $|\{(u, v) \in E \mid \psi(u) \neq \psi(v)\}| \,/\, |E|$ is maximized. The set of nodes in V which are mapped to i in this setting is called the i^{th}* cut. *The edges (v_i, v_j) in the graph for which v_i and v_j are contained in the same cut are called monochromatic; all other edges are called bichromatic.*

Theorem 2 [15] *It is NP-hard to approximate the MAX-k-cut problem within relative error smaller than $1/(34(k-1))$ for $k \geq 2$. For $k \geq 3$ there exists a positive constant c such that it is NP-hard to approximate MAX-k-cut within relative error c/k^2 even if restricted to graphs which allow a solution without any monochromatic edges.*

The concept of an *L-reduction* was defined by Papadimitriou and Yannakakis [20]. The definition stated below is a slightly modified version of [20] (allowing an additional parameter a) that will be useful for this section, and since in this paper we are concerned with maximization problems, we state the definition for maximization problems only.

Definition 3 *An L-reduction from a maximization problem C_1 to a maximization problem C_2 consists of two polynomial time computable functions T_1 and T_2, and two constants $\alpha, \beta > 0$ and a parameter $0 \leq a \leq 1$ with the following properties:*

(a) *For each instance I_1 of C_1, algorithm T_1 produces an instance I_2 of C_2.*

(b) *The maxima of I_1 and I_2, $\mathrm{opt}(I_1)$ and $\mathrm{opt}(I_2)$, respectively, satisfy $\mathrm{opt}(I_2) \leq \alpha\, \mathrm{opt}(I_1)$.*

(c) *Given any solution of the instance I_2 of C_2 with cost c_2 such that the* relative error *of c_2 is at most a, algorithm T_2 produces a solution I_1 of C_1 with cost c_1 satisfying* $(\mathrm{opt}(I_1) - c_1) \leq \beta\,(\mathrm{opt}(I_2) - c_2)$.

The reduction is called maximality preserving if algorithm T_1 maps instances of C_1 with optimum value 1 *to instances of C_2 with optimum value* 1.

The following observation is easy.

Observation 4 *Assume that C_1 L-reduces to C_2 with constants α, β and parameter a. Then, if approximation of C_1 with relative error at most $a\alpha\beta$ is NP-hard, then approximation of C_2 with relative error smaller than a is also NP-hard.*

Furthermore, if approximation of C_1 is NP-hard with relative error at most $a\alpha\beta$ even if restricted to instances of C_1 with optimum value 1 *and, additionally, the algorithm T_1 is maximality preserving, then approximation of C_2 is hard with relative error at most a even if restricted to instances of C_2 with optimum value* 1.

Since the reductions for various types of network architectures are very similar, we first state a general theorem following the approach in [2].

Consider an L-reduction from the MAX-k-cut problem to the loading problem L (with objective function m_L) where the reductions performed by the algorithms T_1 and T_2 have the following additional properties. Given an instance $I_1 = (V, E)$ of the MAX-k-cut problem, assume that T_1 produces in polynomial time an instance I_2 (a specific architecture and a training set with examples $(\boldsymbol{x}; y)$ where $\boldsymbol{x} \in \mathbf{R}^n$, $y \in \mathbf{R}$ and n is polynomial in $|V| + |E|$) of the loading problem L where the points $\boldsymbol{x}$ are of the following form:

- $2|E|$ copies of each of some set of special points P_0 (e.g. the origin),
- for each node $v_i \in V$, d_i copies of one point $\boldsymbol{e}_i$, where d_i is the degree of v_i,
- for each edge $(v_i, v_j) \in E$, one point $\boldsymbol{e}_{ij}$.

Furthermore, assume that the reduction performed by T_1 and T_2 also satisfied the following properties:

(i) For an optimum solution for I_1 we can find using algorithm T_1 an optimum solution of the instance I_2 of the corresponding loading problem L in which all special points P_0 and all points $\boldsymbol{e}_i$ are correct classified and exactly those points $\boldsymbol{e}_{ij}$ are misclassified which correspond to a monochromatic edge (v_i, v_j) in an optimal solution of I_1.

(ii) For any approximate solution of the instance I_2 of the loading problem L which classifies all special points in the set P_0 correctly, we can use the algorithm T_2 to compute in polynomial time an approximate solution of the instance I_1 of the MAX-k-cut problem such that for every monochromatic edge (v_i, v_j) in this solution, either $\boldsymbol{e}_i$, $\boldsymbol{e}_j$, or $\boldsymbol{e}_{ij}$ is missclassified.

Theorem 5 *Under the assumptions stated above, the reduction described above is an L-reduction with constants* $\alpha = k/(k-1)$, $\beta = 2|P_0|+3$, *and parameter* $a = (k-1)/(k^2\,(2|P_0|+3))$. *Furthermore, if the entire training set in* I_2 *can be loaded correctly assuming that the MAX-k-cut problem instance* I_1 *has an optimum solution with value* 1 *then the reduction is also maximality preserving.*

Proof: The approach used here is very similar to Lemma 3 and Lemma 4 of [2]. Let $\mathrm{opt}(I_1)$ and $\mathrm{opt}(I_2)$ be the optimal values of the objective functions of the instances I_1 and I_2. Remember that it is trivially true that $1 \geq \mathrm{opt}(I_1) \geq 1-(1/k)$. Hence, because of (i),

$$\begin{aligned}
\mathrm{opt}(I_2) &= \frac{2|E||P_0| + 2|E| + |E|\mathrm{opt}(I_1)}{2|E||P_0| + 3|E|} \\
&\leq \frac{2|E||P_0| + 2|E| + |E|(1-1/k)}{(2|E||P_0| + 3|E|)(1-1/k)}\mathrm{opt}(I_1) \\
&= \left(\frac{k}{k-1} - \frac{|E|}{(k-1)(2|E||P_0| + 3|E|)}\right)\mathrm{opt}(I_1) \\
&\leq \frac{k}{k-1}\mathrm{opt}(I_1)
\end{aligned}$$

Hence, $\alpha = k/(k-1)$.

Next we show that $\beta = 2|P_0|+3$ provided $a = (k-1)/(k^2\,(2|P_0|+3))$. This is shown by the following steps. Assume that we are given an approximate solution of the instance I_2 of the loading problem with cost c_2.

- If the relative error of c_2 is at most a, then $c_2 \geq 1-a$.
- If $c_2 \geq 1-a$, then the solution must classify all special points from the set P_0 correctly $(*)$. Assume, for the sake of contradiction, that this is not true. Then,

$$\begin{aligned}
c_2 &\leq \frac{2|E|(|P_0|-1) + 3|E|}{2|E||P_0| + 3|E|} \\
&= \frac{2|P_0|+1}{2|P_0|+3} \\
&< 1-a
\end{aligned}$$

- If all the special points from the set P_0 are classified correctly, then, by (ii), we have

$$\begin{aligned}
&\mathrm{opt}(I_1) - c_1 \\
\leq\ & \frac{2|E||P_0| + 3|E|}{|E|}(\mathrm{opt}(I_2) - c_2) \\
=\ & (2|P_0|+3)(\mathrm{opt}(I_2) - c_2),
\end{aligned}$$

  c_1 denoting the cost of the solution of I_1. Hence, $\beta = 2|P_0|+3$.

The remaining "maximality preserving" part of the theorem is obvious from the definition of maximality preserving reductions. □

Note that because of (∗) every solution with relative error at most a, in particular every optimum solution itself, must correct classify P_0 if at least one solution with correct P_0 exists.

Corollary 6 *Assume that a reduction as decribed above which fulfills properties (i) and (ii) can be found. Assume that approximation of the k-cut problem within relative error ϵ is hard. Obviously, since we can assume that $c_1 \geq 1 - (1/k)$ and $\mathrm{opt}(I_1) \leq 1$, we can assume that $\epsilon \leq 1/k$. Then,*

$$a = \frac{(k-1)}{k^2\,(2|P_0|+3)} \geq \frac{(k-1)\epsilon}{k(2|P_0|+3)}$$

Hence, using the Theorem 5 and Observation 4, it follows that an approximation of the loading problem within relative error smaller than

$$\frac{(k-1)\epsilon}{k(2|P_0|+3)}$$

is also NP-hard.

3.2 Application to Neural Networks

In this section, we apply the results of Section 3.1 to various neural nets.

3.2.1 Layered H-nets

First we want to apply this result to layered H-nets, where $H(x)$ is the threshold activation function defined in Section 3.1. This type of neural architecture is common in theoretical study of neural nets as well as in applications. A layered feedforward neural network was defined in Section 3.1. The first layer contains the neurons 1, …, n (the inputs) and the last layer consists of only one neuron. Let $h+1$ be the depth of the H-net and let n_i be the number of nodes at depth i for $1 \leq i \leq h$. Hence, we are dealing with a $(n, n_1, n_2, \ldots, n_h, 1)$ H-net. An instance of the loading problem will be represented by such a tuple and by an example set with rational numbers (presented via their denominators and numerators in a standard way).

Theorem 7 *For any $h \geq 1$, constant $n_1 \geq 2$ and any $n_2, \ldots, n_h \in \mathbf{N}$, it is NP-hard to approximate the loading problem with instances (N, P), where N is the architecture of a layered $\{(n, n_1, \ldots, n_h, 1) \mid n \in \mathbf{N}\}$ H-net and P is a set of examples from $\mathbf{Q}^n \times \{0,1\}$, with relative error at most*

$$\rho_1 = \frac{1}{68n_1 2^{n_1} + 136n_1^3 + 136n_1^2 + 170n_1}$$

If $n_1 \geq 3$ it is NP-hard to approximate the loading problem with instances (N, P) as above with relative error at most

$$\rho_2 = \frac{c(n_1-1)}{5n_1^3 + 2n_1^3 2^{n_1} + 4n_1^5 + 4n_1^4}$$

for some positive constant c even if restricted to input sets which can be classified correctly.

Proof: The proof is via Corollary 6 by giving a reduction from the MAX-k-cut problem, with $k = n_1$ and $|P_0| = 2^{n_1} + 2n_1^2 + 2n_1 + 1$, that satisfies the properties **(i)** and **(ii)** required by Theorem 5 and is also maximality preserving. Notice that, by Theorem 2, we may assume that $\epsilon = 1/(34(k-1))$ for the proof of ρ_1 or $\epsilon = c/k^2$ for some constant c for the proof of ρ_2, and hence, by Corollary 6, it is NP-hard to approximate the loading problem with relative error at most ρ_1 or ρ_2 for the two cases.

The reduction is as follows. We use the same notations as used in Section 3.1 and in Theorem 5. An instance graph $I_1 = (V, E)$ of the MAX-n_1-cut problem is mapped to the following set I_2 of examples in $\mathbf{Q}^n \times \{0, 1\}$ with $n = |V| + n_1 + 1$:

- The set of special points P_0 (together with their labeling) consists of the following:
  - $(0^n; 1)$
  - $(0^{|V|}, \boldsymbol{p}_i, 1; 0)$ for all vectors $\boldsymbol{p}_i \in \{-1, 1\}^{n_1} \backslash (1, \ldots, 1)$
  - $(0^{|V|}, \boldsymbol{p}_0, 1; 1)$ where $\boldsymbol{p}_0 = \{1\}^{n_1}$
  - $(0^{|V|}, \tilde{\boldsymbol{z}}_i, 1; 1)$ for $i = 1, \ldots, n_1(n_1+1)$ and $(0^{|V|}, \bar{\boldsymbol{z}}_i, 1; 0)$ for $i = 1, \ldots, n_1(n_1+1)$, where the points $\tilde{\boldsymbol{z}}_i$ and $\bar{\boldsymbol{z}}_i$ are constructed as follows. Choose $n_1 + 1$ points in each set $H_i = \{\boldsymbol{x} = (x_1, x_2, \ldots, x_{n_1}) \in \mathbf{R}^{n_1} \mid x_i = 0, \forall j \neq i\; x_j > 0\}$, denote the points by $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, $\ldots$ and the entire set of points by Z. The points are chosen such that the following property holds: any given $n_1 + 1$ different points in Z lie on one hyperplane if and only if they are contained in one H_i. That means given $n_1 + 1$ different points $\boldsymbol{z}_{i_1}, \ldots, \boldsymbol{z}_{i_{n_1+1}}$ in Z then the equality $\det \begin{pmatrix} 1 & \cdots & 1 \\ \boldsymbol{z}_{i_1} & \cdots & \boldsymbol{z}_{i_{n_1+1}} \end{pmatrix} = 0$ holds if and only if there exists some i such that $\boldsymbol{z}_{i_j} \in H_i$ for all i_j. The above characterization tells us that this choice is possible. For $z_j \in H_i$ define $\tilde{\boldsymbol{z}}_j \in \mathbf{R}^{n_1}$ by

  $$\tilde{\boldsymbol{z}}_j = (z_{j1}, \ldots, z_{ji-1}, z_{ji} + \epsilon, z_{ji+1}, \ldots, z_{jn_1}),$$

  and $\bar{\boldsymbol{z}}_j \in \mathbf{R}^{n_1}$ by

  $$\bar{\boldsymbol{z}}_j = (z_{j1}, \ldots, z_{ji-1}, z_{ji} - \epsilon, z_{ji+1}, \ldots, z_{jn_1}),$$

  for some small value ϵ which is chosen such that the following property holds: if one hyperplane in $\mathbf{R}^{n_1}$ separates at least $n_1 + 1$ pairs $(\tilde{\boldsymbol{z}}_i, \bar{\boldsymbol{z}}_i)$, these pairs coincide with the $n_1 + 1$ pairs corresponding to the $n_1 + 1$ points in some H_i, and the separating hyperplane nearly coincides with the hyperplane through H_i. Formally, given $n_1 + 1$ pairs $(\tilde{\boldsymbol{z}}_{i_j}, \bar{\boldsymbol{z}}_{i_j})$ and for every pair one point $\boldsymbol{x}_{i_j}$ on the line from $\tilde{\boldsymbol{z}}_{i_j}$ to $\bar{\boldsymbol{z}}_{i_j}$ then $\det \begin{pmatrix} 1 & \cdots & 1 \\ \boldsymbol{x}_{i_1} & \cdots & \boldsymbol{x}_{i_{n_1+1}} \end{pmatrix} = 0$ can only hold if there exists some i such that for all pairs $(\tilde{\boldsymbol{z}}_{i_j}, \bar{\boldsymbol{z}}_{i_j})$ there exists a corresponding point $\boldsymbol{z}_{i_j}$ in H_i. Again this characterization tells us that this choice is possible. See Fig. 1 for an example if $n_1 = 2$.

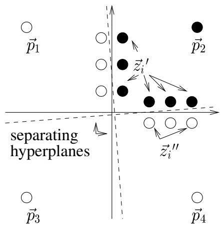


Figure 1: Example for the additional points if $n_1 = 2$, the black points are to be mapped to 1, the white points are to be mapped to 0.

> The role of the points $\tilde{\boldsymbol{z}}_i$ and $\bar{\boldsymbol{z}}_i$ is to enforce that in any network which classifies these points correctly the neurons in the first hidden layer nearly coincide with the hyperplanes through H_i. As a consequence, the points $\boldsymbol{p}_i$ are mapped to the entire set $\{0,1\}^{n_1}$ in the first hidden layer by such a network and therefore determine the function which the remainder of the network computes.

- $\boldsymbol{e}_i \in \{0,1\}^n$ is the i^{th} unit vector (for $1 \le i \le |V|$). The corresponding d_i examples for each $\boldsymbol{e}_i$ are $(\boldsymbol{e}_i; 0)$.
- $\boldsymbol{e}_{ij} \in \{0,1\}^n$ is a vector with 1 at positions i and j from left and 0 otherwise. The corresponding example is $(\boldsymbol{e}_{ij}; 1)$.

We first establish property **(i)** as required by Theorem 5. Given an optimum solution of the n_1-cut problem, we choose the threshold of the i^{th} neuron in the first hidden layer as 0.5 and the weights as $(a_1, \ldots, a_{|V|}, \mathbf{e}_i, -0.5)$ where

$$a_j = \begin{cases} -1 & \text{the node } v_j \text{ is contained in the } i^{\text{th}} \text{ cut} \\ 1 & \text{otherwise} \end{cases}$$

and $\mathbf{e}_i$ is the i^{th} unit vector. All other neurons compute an AND. This maps all points correctly, i.e. for all points $(\boldsymbol{x}; y)$ and these weights $\boldsymbol{w}$ the equality $\beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}) = y$ holds, except for the points $\boldsymbol{x} = \boldsymbol{e}_{ij}$ for which the edge (v_i, v_j) is monochromatic.

Assume conversely that an optimum solution of the loading problem is given. Since we have constructed a solution without errors on P_0, every optimum solution or solution with relative error at most ρ_1 classifies P_0 correctly. Because of counting arguments it classifies P_0 correctly. After changing the thresholds if necessary one can assume that no point in the training set leads to an activation of 0 for one of the neurons. Consider the n_1 hyperplanes defined by $\{\boldsymbol{x} \in \mathbf{R}^{n_1} \,|\, f_i(0^{|V|}, \boldsymbol{x}, 1) = 0\}$ where f_i is the output of the i^{th} hidden neuron in the first hidden layer. Because the points $(0^{|V|}, \tilde{\boldsymbol{z}}_i, 1)$ and $(0^{|V|}, \bar{\boldsymbol{z}}_i, 1)$ are classified differently for each i, the points $\tilde{\boldsymbol{z}}_i$ and $\bar{\boldsymbol{z}}_i$ lie on different sides of at least one of these hyperplanes. Hence the hyperplanes nearly coincide with the hyperplanes through H_i used in the construction. The points $(0^{|V|}, \boldsymbol{p}_i, 1)$ are mapped to the entire set $\{0,1\}^{n_1}$ by $(f_1, \ldots, f_{n_1})$. After a weight change if necessary we can assume that the point $(0^{|V|}, \boldsymbol{p}_0, 1)$ is mapped to 1 by all neurons in the first hidden layer. This uses the fact that for every point $x \neq 0$ the equality $H(x) = 1 - H(-x)$ holds. Because of the classification of the points $(0^{|V|}, \boldsymbol{p}_i, 1)$ the remaining part of the network necessarily computes an AND, that is $\beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}) = f_1(\boldsymbol{x}) \wedge \ldots \wedge f_{n_1}(\boldsymbol{x})$ holds for every $\boldsymbol{x}$ and the network function $\beta_{\mathcal{A}}(\boldsymbol{w}, \_)$. Enlarging the respective weight in the first hidden layer if necessary, one can easily construct a solution of at least the same quality where all points $\boldsymbol{e}_i$ are classified correctly.

Now we define a solution of the MAX-k-cut problem by setting the i^{th} cut $\{v_j \in V \mid f_i(\boldsymbol{e}_j) = 0, \forall k < i \; f_k(\boldsymbol{e}_j) \neq 0\}$ where f_i denotes the function computed by the i^{th} neuron in the first hidden layer. Note that for every $\boldsymbol{e}_j$ at least one neuron with activation 0 exists. Assume an edge (v_i, v_j) is monochromatic, i.e. both nodes are contained in the same cut l. Consequently, the activation of neuron l is zero for both $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$. Because the activation is 1 for the origin the threshold of this neuron is positive, i.e. $\boldsymbol{e}_{ij}$ has the activation 0, too, and is classified wrong. As a consequence property **(i)** is established.

In order to establish property **(ii)** of Theorem 5 define T_2 as follows. Given a solution of the loading problem which classifies P_0 correctly one can compute in polynomial time by changing the signs of some weights if necessary an equivalent solution such that the origin is mapped to 1 by all neurons in the first hidden layer. The node v_j is contained in the i th cut if $\boldsymbol{e}_j$ is mapped to 0 by the i^{th} neuron in the first hidden layer, but to 1 by neurons 1, $\ldots$, $i-1$. All nodes v_i where $\boldsymbol{e}_i$ is mapped to 1 by all neurons in the first hidden layer are contained in the first cut. Since P_0 is correct classified the network function is of the form $\beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}) = f_1(\boldsymbol{x}) \wedge \ldots \wedge f_{n_1}(\boldsymbol{x})$, f_i being the output of the i^{th} hidden neuron in the first hidden layer. Hence every monochromatic edge for which $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are correct classified and hence some f_{j_i} or f_{j_j}, respectively, yield 0, leads to a wrong classified $\boldsymbol{e}_{ij}$ because of the classification of the origin.

Obviously, every MAX-k-cut problem which has a solution without monochromatic edges leads to a loading problem which can be solved without misclassified points. Hence the above reduction is maximality preserving.

Finally, notice that the algorithms T_1 and T_2 in the reduction are polynomial time since n_1 is a constant. $\square$

3.2.2 The $(n, 2, 1)$-$\{\text{sgd}, H_\epsilon\}$-net

The results in the last section deal with realistic multilayer network structures. However, usually a continuous and differentiable activation function instead of the threshold activation function is used in practical applications. The most common activation function is the sigmoidal activation $\text{sgd}(x) = 1/(1 + \mathrm{e}^{-x})$. Therefore it would be interesting to obtain a result for the sigmoidal activation function as well. We consider the following loading problem. We deal with a feedforward architecture of the form $(n, 2, 1)$ where the input dimension n is allowed to vary (this is, for example, the same architecture used in [6]). The activation function of the two hidden nodes is the sigmoidal activation function. Since the network is used for classification purpose, the output activation function should be a step function. Here we consider the following modification of the threshold activation function

$$H_\epsilon(x) = \begin{cases} 0 & \text{if } x < -\epsilon\,, \\ \text{undefined} & \text{if } -\epsilon \le x \le \epsilon\,, \\ 1 & \text{otherwise}\,. \end{cases}$$

The purpose of this definition is to enforce that any classification is performed with a minimum separation accuracy ϵ. It is necessary to restrict the output weights, too, since otherwise any separation accuracy could be obtained simply by an appropriate scaling of the output values. Therefore we demand that any solution has output weights whose absolute values are bounded by some positive constant B (we term this as the weight restriction of the output weights). This setting is captured by the notion of so-called ϵ-separation (for example, see [17]). Formally, the network computes the function

$$\beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}) = H_\epsilon(\alpha\, \text{sgd}(\boldsymbol{a}^t\boldsymbol{x} + a_0) + \beta\, \text{sgd}(\boldsymbol{b}^t\boldsymbol{x} + b_0) + \gamma)$$

where $\boldsymbol{w} = (\alpha, \beta, \gamma, \boldsymbol{a}, a_0, \boldsymbol{b}, b_0)$ are the weights and thresholds, respectively, of the output neuron and the two hidden neurons and $|\alpha|, |\beta| < B$ for some positive constant B. Since we deal with only those examples $(\boldsymbol{x}; y)$ where $y \in \{0, 1\}$, the absolute value of the activation of the output node has to be at least ϵ for every pattern in the training set which is mapped correctly.

Theorem 8 *It is NP-hard to approximate the loading problem with relative error smaller than* $1/2244$ *for the architecture of a* $\{(n, 2, 1) | n \in \mathbf{N}\}$*-net with sigmoidal activation function for the two hidden nodes, activation function* H_ϵ *in the output node with* $\epsilon < 0.5$ *(*$\epsilon \in \mathbf{Q}$*), weight restriction* $B \ge 2$ *of the output weights (*$B \in \mathbf{Q}$*), and examples from* $\mathbf{Q}^n \times \{0, 1\}$*.*

Proof: Again, we use Theorem 5. The reduction of algorithm T_1 of the MAX-2-cut problem to the loading problem is as follows. The input dimension equals $n = |V| + 5$ and the points P_0 together with their labelling are

$$\begin{array}{l}
(0^n; 1) \\
(0^{|V|}, 1, 1, 0, 0, 0; 1) \\
(0^{|V|}, 0, 1, 1, 0, 0; 1) \\
(0^{|V|}, 0, 0, 0, -0.5, 0.5; 1) \\
(0^{|V|}, 0, 0, 0, 0.5, 0.5; 1) \\
(0^{|V|}, 0, 0, 0, c, c; 1) \\
(0^{|V|}, 0, 0, 0, -c, c; 1) \\
(0^{|V|}, 1, 0, 0, 0, 0; 0), \\
(0^{|V|}, 0, 1, 0, 0, 0; 0), \\
(0^{|V|}, 0, 0, 1, 0, 0; 0), \\
(0^{|V|}, 1, 1, 1, 0, 0; 0), \\
(0^{|V|}, 0, 0, 0, -1.5, 0.5; 0), \\
(0^{|V|}, 0, 0, 0, 1.5, 0.5; 0), \\
(0^{|V|}, 0, 0, 0, 1 + c, c; 0), \\
(0^{|V|}, 0, 0, 0, -1 - c, c; 0)
\end{array}$$

with $c > 1 + 2B/\epsilon \cdot \left(\mathrm{sgd}^{-1}(1-\epsilon/(2B)) - \mathrm{sgd}^{-1}(\epsilon/(2B))\right)$. Furthermore,

$\boldsymbol{e}_i$ is the i^{th} unit vector, its labelling is 0,
$\boldsymbol{e}_{ij}$ is the vector with 1 at positions i and j from left and 0 otherwise, its labelling is 1.

First we want to see how a classification looks like. The output neuron computes the activation

$$\alpha\,\mathrm{sgd}(\boldsymbol{a}^t\boldsymbol{x}+a_0)+\beta\,\mathrm{sgd}(\boldsymbol{b}^t\boldsymbol{x}+b_0)+\gamma$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively, are the weight vectors of the two hidden neurons, a_0, b_0 are their respective thresholds, and α, β, and γ are the weights and threshold of the output neuron. We can assume α, $\beta \neq 0$ if P_0 is loaded correctly because only one hyperplane cannot classify P_0 correctly. For the same reason we can assume that $\boldsymbol{a}$ is not the zero vector. If we are interested in the geometrical form of the set of points M which form the classification boundary, i.e. the points x for which

$$(*)\quad \alpha\,\mathrm{sgd}(\boldsymbol{a}^t\boldsymbol{x}+a_0)+\beta\,\mathrm{sgd}(\boldsymbol{b}^t\boldsymbol{x}+b_0)+\gamma=0$$

holds, we can assume that $\boldsymbol{a}^t\boldsymbol{x}+a_0=x_1$, where x_1 is the first component of $\boldsymbol{x}$ after a rotation, translation, and scaling, if necessary, which does not affect the in-principle form. M can be described by the curve which is obtained intersecting the set with a plane spanned by $\boldsymbol{a}$ and $\boldsymbol{b}$. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent then M forms an $n-1$ dimensional manifold which is connected because it is given by the coordinates $(x_1, \mathrm{sgd}^{-1}((-\gamma-\alpha\,\mathrm{sgd}(x_1))/\beta)-b_0)$ (where this term is defined) with respect to the coordinates $\boldsymbol{a}$ and $\boldsymbol{b}$. Hence the curve which describes M can be parameterized by x_1. A normal vector can be parameterized by $n(x_1)=\alpha\,\mathrm{sgd}'(x_1)\cdot\boldsymbol{a}+\beta\,\mathrm{sgd}'(\boldsymbol{b}^t\boldsymbol{x}+b_0)\cdot\boldsymbol{b}$, where the term $\mathrm{sgd}'(\boldsymbol{b}^t\boldsymbol{x}+b_0)$ can be substituted using $(*)$ which means

$$n(x_1)=\alpha\,\mathrm{sgd}'(x_1)\cdot\boldsymbol{a}+(-\gamma-\alpha\,\mathrm{sgd}(x_1))\left(1-\frac{-\gamma-\alpha\,\mathrm{sgd}(x_1)}{\beta}\right)\cdot\boldsymbol{b}\,.$$

Define $\tilde{n}(x_1)=n(x_1)/|n(x_1)|$. Considering the four values γ, $\gamma+\alpha$, $\gamma+\beta$, and $\gamma+\alpha+\beta$ several cases result:

Case 1: All values are ≥ 0 or all values are ≤ 0. Then there exists at least one point in P_0 which is not classified correctly.

Case 2: One value is > 0, the other three are ≤ 0.
We can assume that $\gamma > 0$ since $\mathrm{sgd}(-x)=1-\mathrm{sgd}(x)$. We may have to change the sign of the weights and the thresholds beforehand: if $\alpha+\gamma$ is positive, we substitute α by $-\alpha$, γ by $\alpha+\gamma$, $\boldsymbol{a}$ by $-\boldsymbol{a}$, and a_0 by $-a_0$ without changing the entire mapping; if $\beta+\gamma$ is positive, we substitute β by $-\beta$, γ by $\beta+\gamma$, $\boldsymbol{b}$ by $-\boldsymbol{b}$, and b_0 by $-b_0$; if $\alpha+\beta+\gamma$ is positive we compose the above two changes.

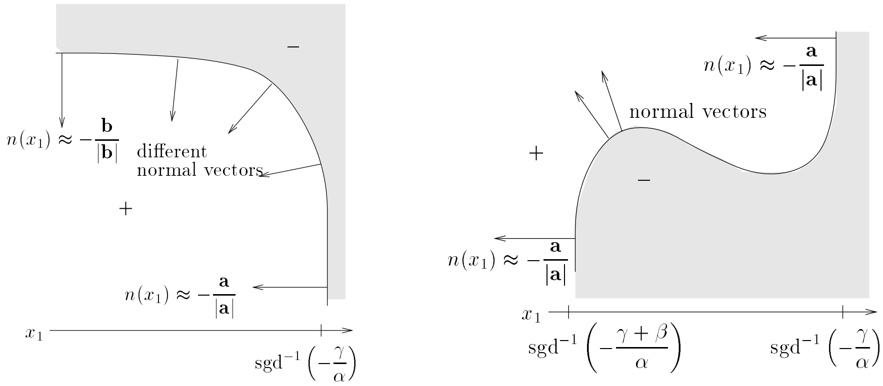


Figure 2: Left: classification in the second case; right: classification in the third case.

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent, then it follows that the region which has a positive activation in the output node is convex and divided from the negative region by at most two parallel lines with normal vector parallel to $\boldsymbol{a}$. Assume that $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent. We find $\alpha \leq -\gamma$, and $\beta \leq -\gamma$. Dividing $(*)$ by γ we obtain $\gamma = 1$, $\alpha \leq -1$, and $\beta \leq -1$. The curve describing M looks like that depicted in Fig. 2, in particular, the region classified positively is convex, as can be seen as follows: For $\mathrm{sgd}(x_1) \approx -1/\alpha$ the normal vector is $\tilde{n}(x_1) \approx -\boldsymbol{a}/|\boldsymbol{a}|$. For $\mathrm{sgd}(x_1) \approx 0$ it is $\tilde{n}(x_1) \approx -\boldsymbol{b}/|\boldsymbol{b}|$. In general, $\tilde{n}(x_1) = \lambda_1(x_1)\,\boldsymbol{a} + \lambda_2(x_1)\,\boldsymbol{b}$ for appropriate functions λ_1 and λ_2. Assume that the curve is not convex. Then there would exist at least two points on the curve with identical $\tilde{n}$, identical λ_1/λ_2 and, consequently, at least one point x_1 with $(\lambda_1/\lambda_2)'(x_1) = 0$. But one can compute $(\lambda_1/\lambda_2)'(x_1) = C(x_1)\cdot(-\beta-1+\mathrm{sgd}(x_1)(2\beta+2)+\mathrm{sgd}^2(x_1)(2\alpha+\alpha^2+\alpha\beta))$ with some factor $C(x_1) = \alpha\beta\,\mathrm{sgd}'(x_1)/((-1-\alpha\,\mathrm{sgd}(x_1))^2(1+\beta+\alpha\,\mathrm{sgd}(x_1))^2) \neq 0$. If $(\lambda_1/\lambda_2)'(x_1)$ was 0, $\alpha = \beta = -1$ or

$$(**) \qquad \mathrm{sgd}(x_1) = \frac{-\beta-1}{\alpha(\alpha+\beta+2)} \pm \sqrt{\frac{(1+\beta)((\alpha+1)^2+\beta(\alpha+1))}{\alpha^2(\alpha+\beta+2)^2}}\,,$$

where the term the square root is taken from is negative except for $\alpha = -1$ or $\beta = -1$ because $(1+\alpha)$, $(1+\beta)$, and $(1+\alpha+\beta)$ are negative. If $\alpha = \beta = -1$ the normal vector is constant and the separating curve coincides with a line, hence it is convex. If only one of the values α or β is -1 the equality $(**)$ reduces to $\mathrm{sgd}(x_1) = 1$ or $\mathrm{sgd}(x_1) = 0$ which both cannot be solved. Note that for α or β equal to -1 the normal vector does not necessarily become almost parallel to $\boldsymbol{a}$ or $\boldsymbol{b}$, respectively, but may stay in a proper subset of the sector spanned by $\boldsymbol{a}$ and $\boldsymbol{b}$. This is the case if both α and β are -1, hence the normal vector is constant. However, in all cases the positive region is convex and the normal vector can be found in the sector spanned by $\boldsymbol{a}$ and $\boldsymbol{b}$.

Case 3: Two values are > 0, two values are < 0.
We can assume that the positive values are γ and $\beta+\gamma$. Maybe we have to change the role of the two hidden nodes or the signs of the weights and the thresholds beforehand.

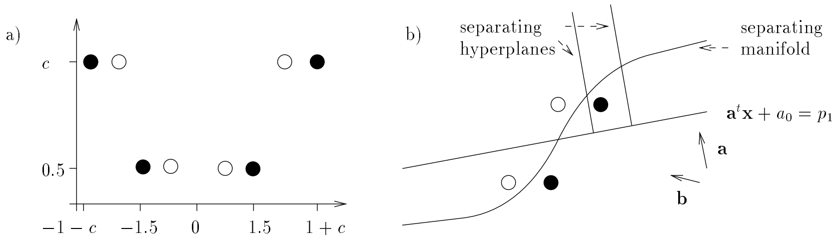


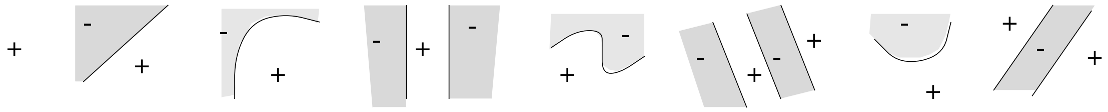
Figure 3: a) Classification problem in the last two dimensions; b) outside the relevant region.

If $\alpha+\gamma$ and $\beta+\gamma$ are positive then we substitute α by $-\alpha$, γ by $\alpha+\gamma$, $\boldsymbol{a}$ by $-\boldsymbol{a}$, and a_0 by $-a_0$; if $\alpha+\gamma$ and $\alpha+\beta+\gamma$ are positive then we substitute α and β by $-\alpha$ and $-\beta$, respectively, γ by $\alpha+\beta+\gamma$, $\boldsymbol{a}$ by $-\boldsymbol{a}$, $\boldsymbol{b}$ by $-\boldsymbol{b}$, a_0 by $-a_0$ and b_0 by $-b_0$; if γ and $\alpha+\gamma$, or $\beta+\gamma$ and $\alpha+\beta+\gamma$ are positive we change the role of the two hidden neurons and end up with one of the first two situations. Note that γ and $\alpha+\beta+\gamma$ cannot both be positive in this situation.

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent we obtain a positive region which is separated from the negative region by at most three parallel lines with normal vector parallel to $\boldsymbol{a}$. We consider the situation that $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent. If $\mathrm{sgd}(x_1) \approx -\gamma/\alpha$ or $\mathrm{sgd}(x_1) \approx (-\gamma-\beta)/\alpha$ then it holds that $\tilde{n}(x_1) \approx -\boldsymbol{a}/|\boldsymbol{a}|$. Dividing $(*)$ by γ, we can assume $\gamma = 1$, $\alpha < -1$, $\beta > -1$, and $\alpha+\beta < -1$. The curve describing M has an S-shaped form (see Fig. 2) because there exists at most one point on the curve where $(\lambda_1/\lambda_2)'(x_1)$ vanishes: this point is $\mathrm{sgd}(x_1) = 0.5$ if $\alpha+\beta+2 = 0$, the point is the solution $(**)$ with the positive sign if $\alpha+\beta+2 < 0$, and the solution $(**)$ with the negative sign if $\alpha+\beta+2 > 0$.

Case 4: 3 values are ≥ 0, one value is < 0. This case is dual to Case 2. We obtain the possible forms by changing the output sign, i.e. the positive and negative region.

To summarize, the classification can have one of the following forms:



Next we want to show that for any classification only in Case 2 all the special points P_0 can be classified correctly. Obviously, Case 1 is not possible.

In order to exclude Case 3 consider the last two dimensions of special points we have constructed. The following classification scenario occurs: (we drop the first $|V|+3$ coefficients which are 0.) the points $(-0.5, 0.5)$, $(0.5, 0.5)$, (c, c), $(-c, c)$ are mapped to 1 and the points $(-1.5, 0.5)$, $(1.5, 0.5)$, $(1+c, c)$, $(-1-c, c)$ are mapped to 0 (see Fig. 3a). They cannot be separated by at most three parallel lines, as a consequence, $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent which means that the separating manifold has an S-shaped form. Define $p_0 := \mathrm{sgd}^{-1}(\epsilon/(2B))$

and $p_1 := \mathrm{sgd}^{-1}(1-\epsilon/(2B))$. $\{\boldsymbol{x} \mid p_0 \leq \boldsymbol{a}^t\boldsymbol{x} + a_0 \leq p_1\}$ and $\{\boldsymbol{x} \mid p_0 \leq \boldsymbol{b}^t\boldsymbol{x} + b_0 \leq p_1\}$ are called the $\boldsymbol{a}$- or $\boldsymbol{b}$-relevant region, respectively. Outside, $\mathrm{sgd}(\boldsymbol{a}^t\boldsymbol{x} + a_0)$ or $\mathrm{sgd}(\boldsymbol{b}^t\boldsymbol{x} + b_0)$, respectively, can be substituted by a constant, the difference of the output being at most $\epsilon/2$.

Now the argument proceeds as follows. First it is shown that three points forming a triangle with sides of length at least $c-1$ are contained in the $\boldsymbol{a}$-relevant region. This leads to a bound for the absolute value of $\boldsymbol{a}$. Second, if three points forming an analogous triangle are contained in the $\boldsymbol{b}$-relevant region the same argument leads to a bound for the absolute value of $\boldsymbol{b}$. Using these bounds it can be seen that neighboring points cannot be classified differently. Third, if no such triangle is contained in the $\boldsymbol{b}$-relevant region, the part $\boldsymbol{b}^t\boldsymbol{x} + b_0$ does not contribute to the classification of neighboring points outside the $\boldsymbol{b}$-relevant region and the two points cannot be classified differently.

First step: Since the points with second component 0.5 cannot be separated by one hyperplane, one point $(x, 0.5)$ with $x \in [-1.5, 1.5]$ exists inside the $\boldsymbol{a}$- and $\boldsymbol{b}$-relevant region, respectively. If the points (c, c) and $(1+c, c)$ were both outside the $\boldsymbol{a}$-relevant region then they would be separated by any hyperplane with normal vector $\boldsymbol{b}$ which intersects the separating manifold outside the $\boldsymbol{a}$-relevant region because the part given by $\boldsymbol{a}$ does not contribute to the classification (see Fig. 3b). The normal vector of the manifold is approximately $-\boldsymbol{a}/|\boldsymbol{a}|$ for large and small $\boldsymbol{b}^t\boldsymbol{x} + b_0$, respectively. Therefore we can find a hyperplane where both points are located on the same side, a contradiction. The same argument holds for $(-c, c)$ and $(-1-c, c)$. Therefore the diameter of the $\boldsymbol{a}$-relevant region restricted to the last two dimensions is at least $c-1$. Consequently, $a \leq (p_1 - p_0)/(c-1) < \epsilon/(2B)$, where $a = |(a_{n+4}, a_{n+5})|$.

Second step: If one of the points (c, c) and $(1+c, c)$ and one of the points $(-c, c)$ and $(-1-c, c)$ is contained in the $\boldsymbol{b}$-relevant region, it follows that $b \leq \epsilon/(2B)$ for $b = |(b_{n+4}, b_{n+5})|$. This leads to the following contradiction: for the points $\boldsymbol{x}_1 = (0, \ldots, 0, c, c)$ and $\boldsymbol{x}_2 = (0, \ldots, 0, 1+c, c)$ we have

$$\begin{aligned}&|\alpha\,\mathrm{sgd}(\boldsymbol{a}^t\boldsymbol{x}_1 + a_0) + \beta\,\mathrm{sgd}(\boldsymbol{b}^t\boldsymbol{x}_1 + b_0) + \gamma - \alpha\,\mathrm{sgd}(\boldsymbol{a}^t\boldsymbol{x}_2 + a_0) - \beta\,\mathrm{sgd}(\boldsymbol{b}^t\boldsymbol{x}_1 + b_0) - \gamma| \\ &\leq |\alpha|\,|\boldsymbol{a}^t\boldsymbol{x}_1 - \boldsymbol{a}^t\boldsymbol{x}_2| + |\beta|\,|\boldsymbol{b}^t\boldsymbol{x}_1 - \boldsymbol{b}^t\boldsymbol{x}_2| \leq \epsilon\,.\end{aligned}$$

because $|\alpha|$, $|\beta| \leq B$ and $|\mathrm{sgd}(x) - \mathrm{sgd}(x+\delta)| \leq \delta$ for every $\delta > 0$.

Third step: If both points (c, c) and $(1+c, c)$ or both points $(-c, c)$ and $(-1-c, c)$ are outside the $\boldsymbol{b}$-relevant region, the difference of the values $\mathrm{sgd}(\boldsymbol{b}^t\boldsymbol{x} - b_0)$ with corresponding $\boldsymbol{x}$ is at most $\epsilon/(2B)$. The same contradiction results.

This excludes Case 3. Next we exclude Case 4: The classification includes in the dimensions $|V|+1$ to $|V|+3$ the situation depicted in Fig. 4. Obviously at most two parallel planes cannot separate a convex negative region. Consequently, $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent. The negative points are contained in a convex region, each positive point is separated by at least one tangential hyperplane of the separating manifold M from all negative points. Consider the projection to a plane parallel to $\boldsymbol{a}$ and $\boldsymbol{b}$. Following the convex curve which describes M the signs of the coefficients of a normal vector can change at most once. But a normal vector oriented in such a way that it points to the positive points which separates

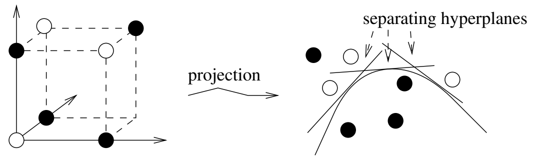


Figure 4: Classification problem; projection of the classification to the plane which is described by the weight vectors of the two hidden neurons, at least one negative point is not classified correctly

a positive point necessarily has the signs $(+,+,-)$ for $(1,1,0)$, $(-,+,+)$ for $(0,1,1)$, and $(-,-,-)$ for $(0,0,0)$ in the dimensions $|V|+1$ to $|V|+3$, a contradiction. Hence we have shown that any classification which maps P_0 correctly is of Case 2.

Next we show that the correspondence demanded in property **(i)** of Theorem 5 holds: an optimum solution of the MAX-2-cut problem leads to the following approximate solution of the loading problem with weights $\alpha = \beta = -1$, $\gamma = 0.5$, $\boldsymbol{a} = K \cdot (a_1, \ldots, a_n, 1, -1, 1, 1, -1)$, $\boldsymbol{b} = K \cdot (b_1, \ldots, b_n, -1, 1, -1, -1, -1)$, $a_0 = -0.5 \cdot K$, $b_0 = -0.5 \cdot K$, where K is a positive constant and

$$a_i = \begin{cases} 1 & \text{if } v_i \text{ is in the first cut} \\ -2 & \text{otherwise} \end{cases} \quad \text{and} \quad b_i = \begin{cases} 1 & \text{if } v_i \text{ is in the second cut} \\ -2 & \text{otherwise} \end{cases} .$$

For appropriate K, this misclassifies at most the points $\boldsymbol{e}_{ij}$ for which the edge (v_i, v_j) is monochromatic. Note that we can obtain every accuracy $\epsilon < 0.5$ with appropriate K.
Conversely, assume that an optimum solution of the loading problem is given. Then we can assume that all special points and all points corresponding to a node v_i are classified correctly because a solution classifying P_0 correctly exists. Consequently, we are in Case 2. We can easily compute an equivalent solution with $\gamma > 0$ performing the weight changes as described in the single cases. Define a solution of the MAX-2-cut problem such that exactly those nodes v_i with $a_i > 0$ are in the first cut. Any edge (v_i, v_j) such that $\boldsymbol{e}_{ij}$ is classified correctly is bichromatic because in dimension i and j the following classification can be found: $(0,0)$ and $(1,1)$ are mapped to 1, $(0,1)$ and $(1,0)$ are mapped to 0 (see Fig. 5). If $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent a normal vector of the two lines which bounds the positive region has obviously two different signs. Consequently, a_i and a_j have different signs. In the linearly dependent case, i.e. when the positive region is separated by a convex curve with normal vector contained in the sector spanned by $\boldsymbol{a}$ and $\boldsymbol{b}$, we can find two tangential lines of the curve such that these lines each separate one of the negative points. Necessarily, the i^{th} or j^{th} component of a normal vector (oriented towards the negative region) is positive in order to separate $\boldsymbol{e}_i$ or $\boldsymbol{e}_j$ from 0, respectively, but the signs cannot be equal because of the classification of $\boldsymbol{e}_{ij}$. Furthermore, each sign of a component of a normal vector can change at most once if we follow the curve which separates the two regions because the positive region is convex. Consequently, the signs of a_i and a_j are to be different.

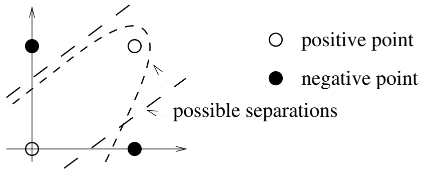


Figure 5: Classification corresponding to a bichromatic edge

Finally we establish property **(ii)** of Theorem 5: given a weight setting for the loading problem corresponding to a graph it is possible to compute a cut as demanded in property **(ii)**. Because P_0 is classified correctly we can assume that the classification is of Case 2 and w.l.o.g. $\gamma > 0$. As before, one can define a solution of the cut problem via the sign of a_i, i.e., the nodes v_i with positive a_i are in the first cut. If (v_i, v_j) is monochromatic and $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are correct then $\boldsymbol{e}_{ij}$ is not classified correctly which is shown using the same argument as before. $\square$

Unfortunately, the above situation restricts to the case of ϵ-separation which seems realistic for concrete applications but is nontheless a modification of the problem. However, this restriction offers the possibility of transferring the result to activation functions which are *similar* to the standard sigmoid.

Corollary 9 *It is NP-hard to approximate the loading problem with relative error smaller than* $1/2244$ *for the architecture of a* $\{(n, 2, 1) \mid n \in \mathbf{N}\}$*-net with activation function* σ *in the hidden layer, activation function* H_ϵ *in the output with* $\epsilon < 1/3$ *(*$\epsilon \in \mathbf{Q}$*), weight restriction* $B \geq 2$ *of the output weights (*$B \in \mathbf{Q}$*), and examples from* $\mathbf{Q}^n \times \{0, 1\}$ *if* $|\sigma(x) - \mathrm{sgd}(x)| \leq \epsilon/(4B)$ *holds for all* $x \in \mathbf{R}$*.*

Proof: The proof goes via L-reduction from the MAX-2-cut problem and Theorem 5. In fact, the proof is identical to the proof of Theorem 8 except that sigmoidal networks are substituted by σ-networks which is possible because sgd and σ are almost identical. In the following proof, we only scetch the parts which are identical to the proof of Theorem 8 and carry out the modifications due to the activation function σ.

Assume that we are given an instance of the MAX-2-cut problem. One can reduce this cut problem to a loading problem for the sigmoidal network with weight restriction B and minimum accuracy $\epsilon/2$. This training set can be loaded with a network with activation function sgd and every accuracy < 0.5 such that only the points corresponding to monochromatic edges are classified wrong. Now we substitute the function sgd by σ in this network. Since the weights of the output neurons are at most B, the outputs are changed by at most $\epsilon/2$. Hence the σ-network classifies all points but points corresponding to monochromatic edges correctly with accuracy $\epsilon < 1/3 \leq 0.5 - \epsilon/2$. Conversely, any solution of this loading problem with a network with activation function σ, accuracy ϵ, and weight restriction B leads to a solution for a network with activation function sgd with accuracy $\epsilon/2$ of the same quality.

This is due to the fact that σ and sgd differ by at most $\epsilon/(4B)$ and hence the outputs are changed by at most $\epsilon/2$. Considering the signs of the weights in this sigmoidal network, we can construct a cut in the same way as in the proof of Theorem 8. At most the edges (v_i, v_j) corresponding to misclassified points $\boldsymbol{e}_{ij}$ are monochromatic and the same holds for the σ-network, too. Hence property **(i)** of Theorem 5 holds.

Conversely, given a solution of the loading problem for the activation function σ with accuracy ϵ one can first substitute the activation σ by sgd obtaining a solution for sgd of the same quality with accuracy $\epsilon/2$. A computation of a cut in the same way as in the sigmoidal case leads to a cut where every misclassified point for sgd comes from either a misclassification of $\boldsymbol{e}_i$, $\boldsymbol{e}_j$, or $\boldsymbol{e}_{ij}$. Hence this point was misclassified by the network with activation σ as well. Hence property **(ii)** of Theorem 5 follows. □

3.2.3 The $(n, 2, 1)$-$\{\text{lin}, H\}$-net

In this section, we prove the NP-hardness of the approximability of the loading problem with the semilinear activation function commonly used in the neural net literature [3, 7, 8, 16]. The semilinear activation function is defined as:

$$\text{lin}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } 0 < x \leq 1 \\ 1 & \text{otherwise} \end{cases} .$$

This function is continuous and captures the linearity of the sigmoidal activation at the origin as well as the asymptotic behaviour of the sigmoid for large values. The following result is of interest since it is not necessary to restrict to the situation of ϵ-separation now.

Theorem 10 *It is NP-hard to approximate the loading problem with relative error smaller than* $1/2380$ *for the architecture of* $\{(n, 2, 1) \mid n \in \mathbf{N}\}$*-nets with the semilinear activation function in the hidden layer, the threshold activation function in the output, and examples from* $\mathbf{Q}^n \times \{0, 1\}$.

Proof: Again, the proof is via L-reduction from the MAX-2-cut problem using Theorem 5. An instance graph $I_1 = (V, E)$ of the MAX-2-cut problem is mapped to the following set of examples in $\mathbf{Q}^n \times \{0, 1\}$ with $n = |V| + 6$:

- The set of special points P_0 together with their labeling consists of the following:

$$
\begin{array}{l}
(0^n;1)\\
(0^{|V|},1,1,0,0,0,0;1)\\
(0^{|V|},0,1,1,0,0,0;1)\\
(0^{|V|},1,0,0,0,0,0;0)\\
(0^{|V|},0,1,0,0,0,0;0)\\
(0^{|V|},0,0,1,0,0,0;0)\\
(0^{|V|},1,1,1,0,0,0;0)\\
(0^{|V|},0,0,0,1,0.5,1;1)\\
(0^{|V|},0,0,0,1.5,0.5,1;0)\\
(0^{|V|},0,0,0,-1,0.5,1;1)\\
(0^{|V|},0,0,0,-1.5,0.5,1;0)\\
(0^{|V|},0,0,0,6,5.5,1;1)\\
(0^{|V|},0,0,0,6.5,5.5,1;0)\\
(0^{|V|},0,0,0,-6,5.5,1;1)\\
(0^{|V|},0,0,0,-6.5,5.5,1;0)\\
(0^{|V|},0,0,0,0,-0.4,1;0)
\end{array}
$$

- $\boldsymbol{e}_i$ and $\boldsymbol{e}_{ij}$ are the same vectors with the same labelling as in the sigmoidal case (see proof of Theorem 8).

Again we want to see how a classification looks like, that is, we consider the points $\boldsymbol{x}$ for which

$$\alpha \operatorname{lin}(\boldsymbol{a}^t\boldsymbol{x} + a_0) + \beta \operatorname{lin}(\boldsymbol{b}^t\boldsymbol{x} + b_0) + \gamma = 0$$

holds. The weights are denoted as in the proof of Theorem 8. We can assume $\alpha \neq 0$, $\beta \neq 0$, $\boldsymbol{a} \neq 0$ if P_0 is loaded correctly; otherwise the points would be separated by at most one hyperplane which is impossible. Furthermore, we assume $\boldsymbol{a}^t\boldsymbol{x} + a_0 = x_1$. Considering the four values γ, $\gamma+\alpha$, $\gamma+\beta$, and $\gamma+\alpha+\beta$ the following cases can be found for the curve in the plane spanned by $\boldsymbol{a}$ and $\boldsymbol{b}$. Note that because of the equality $\operatorname{lin}(x) = 1 - \operatorname{lin}(1-x)$ we can perform similar weight changes as in the sigmoidal case without changing the mapping. To be more precise, one can first substitute the activation function lin by the activation function $\operatorname{lin}_{0.5}$ with $\operatorname{lin}_{0.5}(x) = \operatorname{lin}(x - 0.5)$, perform exactly the same weight changes as in the sigmoidal case since $\operatorname{lin}_{0.5}$ possesses the same symmetry as sgd, and substitute $\operatorname{lin}_{0.5}$ by lin, afterwards.

Case 1: All values are ≥ 0 or all values are ≤ 0. Then there would exist misclassified points in P_0.

Case 2: Exactly one value is positive, w.l.o.g. $\gamma > 0$.

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are parallel then the positive region is convex and separated from the negative region by at most two lines with normal vector parallel to $\boldsymbol{a}$.

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent then the positive region is separated from the negative region by the lines defined by $\{\boldsymbol{x} \mid x_1 \leq 0, \boldsymbol{b}^t\boldsymbol{x} + b_0 = -\gamma/\beta\}$, $\{\boldsymbol{x} \mid x_1 = -\gamma/\alpha, \boldsymbol{b}^t\boldsymbol{x} + b_0 \leq 0\}$, and $\{\boldsymbol{x} \mid (\alpha\boldsymbol{a} + \beta\boldsymbol{b})^t\boldsymbol{x} + \beta b_0 + \gamma = 0\}$. The positive region

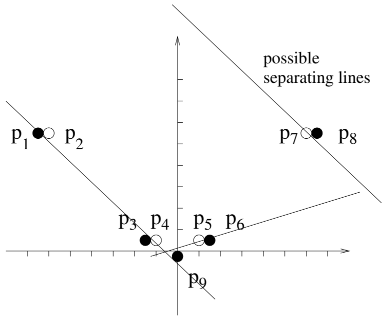


Figure 6: Points which exclude Case 3, the black points are to be mapped to 1, the white points are to be mapped to 0.

consists of the intersection of the three halfspaces defined by these lines. Hence the positive region is convex and separated from the negative region by a C^0 curve with three linear pieces with normal vectors parallel to $\boldsymbol{a}$, $\boldsymbol{b}$, or a convex combination of $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively.

Case 3: Two values, w.l.o.g. γ and $\beta+\gamma$ are positive, all other values are negative.

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent then the positive region is separated from the negative region by up to three parallel lines.

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent then one obtains the separating lines $\{\boldsymbol{x} \mid x_1 = -\gamma/\alpha, \boldsymbol{b}^t\boldsymbol{x}+b_0 \leq 0\}$, $\{\boldsymbol{x}|x_1 = (-\beta-\gamma)/\alpha, \boldsymbol{b}^t\boldsymbol{x}+b_0 \geq 1\}$, and $\{\boldsymbol{x}|(\alpha\boldsymbol{a}+\beta\boldsymbol{b})^t\boldsymbol{x}+\beta b_0+\gamma = 0\}$. That means, the positive region is separated from the negative region by a C^0 curve consisting of three linear pieces. Two of them have a normal vector parallel to $\boldsymbol{a}$. Denoting by H_1, H_2, and H_3, respectively, the halfspaces defined by the points with the weakened conditions $x_1 \leq -\gamma/\alpha$, $x_1 \leq (-\gamma-\beta)/\alpha$, or $(\alpha\boldsymbol{a}+\beta\boldsymbol{b})^t\boldsymbol{x}+\beta b_0+\gamma \geq 0$, respectively, the positive region lies in the set $H_1 \cup (H_2 \cap H_3)$ or $(H_1 \cap H_2) \cup H_3$ depending on the sign of β.

Case 4: three values are ≥ 0 and one value is < 0. Dual to Case 2.

Next we want to show that only Case 2 can take place if all the special points P_0 are classified correctly.

Obviously, Case 1 cannot take place. Case 4 is excluded by the points which are nonvanishing in dimension $|V|+1$ to $|V|+3$ in the same way as in Theorem 8: More precisely, one finds three separating planes which must have normal vectors with signs $(+,+,-)$, $(-,+,+)$, or $(-,-,+)$, respectively, in order to separate the positive points. But this cannot be the case if one of the vectors is a convex combination of the other two vectors.

Case 3 can be excluded by the points with nonvanishing coefficients in the last but one and last but two components (see Fig.6). The points are separated by a C^0 curve consisting of 3 linear pieces two of which are parallel. One line necessarily separates two of the pairs (p_1, p_2), (p_3, p_4), (p_5, p_6), and (p_7, p_8), w.l.o.g. the first two pairs. This line cannot separate another pair or the point p_9 because the line through p_1 and p_4 intersects the axes at $(0, -0.4\overline{09})$. Since a parallel line cannot separate both other pairs we find that the two other pieces separate p_9 and (p_5, p_6), or (p_7, p_8), respectively, the latter being parallel to the first line. But since the pair (p_7, p_8) lies on the same side of the second line as (p_1, p_2), we find that the positive part would be contained in the intersection of the respective halfspaces.

Hence every classification which maps P_0 correctly is of Case 2.

Now we show that the correspondence demanded in property **(i)** of Theorem 5 holds. An optimum solution of the MAX-2-cut problem leads to a solution with weights $\alpha = \beta = -1$, $\gamma = 0.5$, $\boldsymbol{a} = (a_1, \ldots, a_n, 1, -1, 1, 0.5, -0.5, 0.625)$, $a_0 = -0.5$, $\boldsymbol{b} = (b_1, \ldots, b_n, -1, 1, -1, -0.5, -0.5, 0.625)$, $b_0 = -0.5$, where

$$a_i = \begin{cases} 1 & \text{if } v_i \text{ is in the first cut} \\ -2 & \text{otherwise} \end{cases} \quad \text{and} \quad b_i = \begin{cases} 1 & \text{if } v_i \text{ is in the second cut} \\ -2 & \text{otherwise} \end{cases} .$$

With these weights at most the points $\boldsymbol{e}_{ij}$ corresponding to monochromatic edges are misclassified.

Assume conversely that we are given an optimum solution of the loading problem. Since a solution classifying P_0 correctly exists, this optimum solution classifies all points P_0 correctly. Hence the classification is of Case 2. W.l.o.g. $\gamma > 0$ and all points $\boldsymbol{e}_i$ are correct. Define a solution of the MAX-2-cut problem such that exactly those nodes v_i are in the first cut where $a_i > 0$. Now it needs to be shown that any edge (v_i, v_j) such that $\boldsymbol{e}_{ij}$ is classified correctly is bichromatic. Assume for the sake of contradiction that this is not the case. Then we find in the dimensions i and j the situation depicted in Fig. 7 where the positive region is convex and either separated from the negative region by two parallel lines with normal vector $\boldsymbol{a}$ or by three lines with normal vector $\boldsymbol{a}$, $\boldsymbol{b}$, and a convex combination of $\boldsymbol{a}$ and $\boldsymbol{b}$. A line separating $\boldsymbol{e}_i$ or $\boldsymbol{e}_j$, respectively, has necessarily a normal vector with two different signs and a negative i^{th} or j^{th} component, respectively. If $\boldsymbol{a}$ had two equal signs the separating lines could not contain two lines with these properties.

Finally, property **(ii)** of Theorem 5 can be shown: given a weight setting such that P_0 is classified correctly one can assume that Case 2 takes place and hence $\gamma > 0$. Define a solution of the MAX-2-cut problem where v_i is in the first cut if and only if $a_i > 0$. If (v_i, v_j) is monochromatic $\boldsymbol{e}_i$, $\boldsymbol{e}_j$, or $\boldsymbol{e}_{ij}$ is misclassified which can be shown using the same argument as before. $\square$

This generalizes the approximation results to more realistic activation functions.

3.3 Avoiding Multiplicities

In the reductions of previous sections examples with multiplicities were contained in the training sets in order to enforce that they are classified correctly. One may ask the question if this is avoidable since the training set for an artificial learning algorithm is unlikely to contain exactly the same pattern many times.

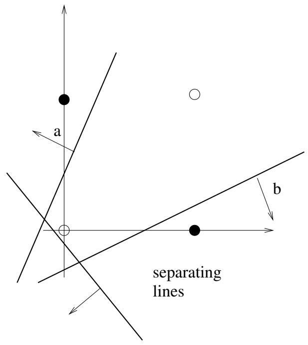


Figure 7: Classification in case 2. The black points are to be mapped to 0, the white points are to be mapped to 1.

Looking at the proofs it turns out that this is possible. Consider the following modification of the reduction of the MAX-k-cut problem to a loading problem: T_1 yields the following points:

- $3|E|$ copies of each point in some set P_0,
- for each node v_i, $2d_i$ copies of some point $\boldsymbol{e}_i$ where d_i is the degree of v_i,
- for each edge (v_i, v_j), points $\boldsymbol{e}_{ij}$ and $\boldsymbol{o}_{ij}$.

and, assume that the algorithms T_1 and T_2 satisfied the following properties:

(iii) For an optimum solution of the MAX-k-cut problem one can find an optimum solution of the instance of the corresponding loading problem L in which the special points P_0 and all $\boldsymbol{e}_i$ points are correct classified and exactly the monochromatic edges (v_i, v_j) lead to misclassified points $\boldsymbol{e}_{ij}$ or $\boldsymbol{o}_{ij}$.

(iv) For any approximate solution of the instance of the loading problem which classifies the special points in the set P_0 correctly, one can use the algorithm T_2 to compute in polynomial time an approximate solution of the instance of the MAX-k-cut problem such that for any monochromatic edge (v_i, v_j) in this solution, either $\boldsymbol{e}_i$, $\boldsymbol{e}_j$, $\boldsymbol{e}_{ij}$, or $\boldsymbol{o}_{ij}$ is missclassified.

Theorem 11 *Under the assumptions stated above the reduction is a maximality preserving L-reduction with constants $\alpha = k/(k-1)$, $\beta = 3|P_0| + 6$, and $a = (k-1)/(k^2(3|P_0| + 6))$.*

The proof is analogous to Theorem 5; in the following, we will show a more general result which allows us to avoid multiplicities as well. Hence a reduction from the MAX-k-cut problem as above would show that approximate loading is hard to approximate within relative error $1/(34k(3|P_0|+6))$ for $k \geq 2$, and for $k \geq 3$ within relative error $(k-1)/(ck^3(3|P_0|+6))$

for some constant c even if restricted to situations where a solution without misclassifications exists.

The arguments in the theorems in previous sections can be transferred to this new settings immediately as follows. Note that the origin is used in Theorems 7, 8, and 10 for two different purposes: regarded as a point in P_0 it is used to exclude certain geometrical situations such as Case 3 in the proof of Theorem 8, and, additionally, it is used for every edge (v_i, v_j) in the graph to guarantee the correspondence of bichromatic edges (v_i, v_j) and correctly classified points $\boldsymbol{e}_{ij}$. Making this latter role explicit we set $\boldsymbol{o}_{ij} = 0$. Note that even if $\boldsymbol{o}_{ij}$ was not guaranteed to be correct because of the multiplicity of the origin it would hold that in an optimum solution w.l.o.g. $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are correct and one of $\boldsymbol{e}_{ij}$ and $\boldsymbol{o}_{ij}$ is wrong if and only if (v_i, v_j) is monochromatic. Furthermore, the reduction T_2 would lead to a monochromatic edge if and only if at least one of $\boldsymbol{e}_i$, $\boldsymbol{e}_j$, $\boldsymbol{e}_{ij}$, or $\boldsymbol{o}_{ij}$ is classified wrong. Hence the reductions in Theorems 7, 8, 10 give rise to L-reductions as described in Theorem 11 as well.

Although the bounds are somewhat weaker, this offers the possibility of substituting every point $\boldsymbol{x}$ with multiplicity d in this reduction by d points which nearly coincide with $\boldsymbol{x}$. Looking at the proofs it will turn out that we can modify T_1 and T_2 in the following way: all examples (in particular $\boldsymbol{o}_{ij}$) are mutually different, $\boldsymbol{e}_i$ is substituted by $2d_i$ different points $\boldsymbol{e}_i^k$, each point p in P_0 is substituted by $3|E|$ different points p^k, and the following properties hold:

(v) the same as (iii),

(vi) if for each $p \in P_0$ at least one p^k is correct then T_2 computes in polynomial time an approximate solution where for each monochromatic edge (v_i, v_j) one of the points $\boldsymbol{e}_i$, $\boldsymbol{e}_j$, $\boldsymbol{e}_{ij}$, or $\boldsymbol{o}_{ij}$ is misclassified.

Following the proof of Theorem 5 it can be seen that this constitutes an L-reduction:

We obtain for an optimum solution with value $\mathrm{opt}(I_1)$ of an instance I_1 of the k-cut problem and an optimum solution with value $\mathrm{opt}(I_2)$ of an instance I_2 of the corresponding loading problem because of **(v)**:

$$\begin{aligned} \mathrm{opt}(I_1) &\leq \frac{3|E||P_0| + 5|E| + |E|\mathrm{opt}(I_1)}{3|E||P_0| + 6|E|} \\ &\leq \frac{k}{k-1}\mathrm{opt}(I_1)\,. \end{aligned}$$

Any approximate solution of I_2 with cost $c_2 \leq a = (k-1)/(k^2(3|P_0|+6))$ classifies at least one point of each set $\{p^i \mid i\}$ of the special points P_0 correct because otherwise

$$\begin{aligned} c_2 &\leq \frac{3|E|(|P_0|-1) + 6|E|}{3|E||P_0| + 6|E|} \\ &= \frac{|P_0|+1}{|P_0|+2} < 1 - a\,. \end{aligned}$$

If at least one p^i of each set in P_0 is correct we obtain because of **(vi)**, c_1 and c_2 denoting the cost of a solution of I_1 or I_2, respectively:

$$\mathrm{opt}(I_1) - c_1 \;\leq\; \frac{3|E||P_0| + 6|E|}{|E|}(\mathrm{opt}(I_2) - c_2)\,.$$

Hence hardness of approximate loading within the bounds as described in Theorem 11 follows.

Corollary 12 *The reductions in Theorems 7, 8, and 10 can be modified such that (v) and (vi) hold.*

Proof: Note that we constructed explicit weights which solved the respective pattern sets such that only examples corresponding to monochromatic edges (v_i, v_j) were misclassified. The output activation allows a slight change in every case because it was not exactly 0 in the threshold or semilinear case and of absolute value larger than ϵ in the sigmoidal case. Furthermore, no activation was exactly 0 in the threshold case even for the other hidden neurons. Hence the continuity of the network function allows us to find some δ such that the classification of the respective points does not change if they are substituted by any point contained in the open ball of radius δ whose center is the respective point. Hence for every modiciation of the training set such that the points p^k lie in a small neighborhood of p, $\boldsymbol{e}_i^k$ lie in a small neighborhood of $\boldsymbol{e}_i$, and $\boldsymbol{o}_{ij}$ lie in a small neighborhood of the origin one can find a solution of the same quality as above.

Next we show that every point $p \in P_0$ in the respective solutions can be substituted by any point in an appropriate set such that the same geometrical situations are excluded:

- In Theorem 7 one can substitute the points $\tilde{\boldsymbol{z}}_j = (z_{j1}, \ldots, z_{ji-1}, z_{ji}+\epsilon, z_{ji+1}, \ldots, z_{jn_1})$ by $(z_{j1}, \ldots, z_{ji-1}, z_{ji}+\epsilon', z_{ji+1}, \ldots, z_{jn_1})$ and $\bar{\boldsymbol{z}}_j \in \mathbf{R}^{n_1}$ by $(z_{j1}, \ldots, z_{ji-1}, z_{ji}-\epsilon, z_{ji+1}, \ldots, z_{jn_1})$ for some $0 < \epsilon' < \epsilon$ which can be chosen independently for each pattern; we can substitute $(0^{|V|}, \boldsymbol{p}_i, 1; y)$ by $(0^{|V|}, \boldsymbol{p}_i', 1; y)$ where $\boldsymbol{p}_i'$ is any point in a neighborhood of $\boldsymbol{p}_i$ depending on ϵ. Still the points obtained from $\tilde{\boldsymbol{z}}_j$ and $\bar{\boldsymbol{z}}_j$ guarantee that the hidden neurons in the first hidden layer nearly coincide with the hyperplanes H_i used in the construction, the points $\boldsymbol{p}_j$ determine the remainder of the network to compute an AND.

- In Theorem 8 we can substitute the points:
  $(0^{|V|}, 0, 0, 0, -0.5, 0.5; 1)$ by $(0^{|V|}, 0, 0, 0, -0.5-\epsilon, 0.5; 1)$,
  $(0^{|V|}, 0, 0, 0, 0.5, 0.5; 1)$ by $(0^{|V|}, 0, 0, 0, 0.5+\epsilon, 0.5; 1)$,
  $(0^{|V|}, 0, 0, 0, c, c; 1)$ by $(0^{|V|}, 0, 0, 0, c+\epsilon, c; 1)$,
  $(0^{|V|}, 0, 0, 0, -c, c; 1)$ by $(0^{|V|}, 0, 0, 0, -c-\epsilon, c; 1)$,
  $(0^{|V|}, 0, 0, 0, -1.5, 0.5; 0)$ by $(0^{|V|}, 0, 0, 0, -1.5+\epsilon, 0.5; 0)$,
  $(0^{|V|}, 0, 0, 0, 1.5, 0.5; 0)$ by $(0^{|V|}, 0, 0, 0, 1.5-\epsilon, 0.5; 0)$,
  $(0^{|V|}, 0, 0, 0, 1+c, c; 0)$ by $(0^{|V|}, 0, 0, 0, 1+c-\epsilon, c; 0)$,
  $(0^{|V|}, 0, 0, 0, -1-c, c; 0)$ by $(0^{|V|}, 0, 0, 0, -1-c+\epsilon, c; 0)$

where $0 < \epsilon < 0.5$ can be chosen independently for each vector. Case 3 is still excluded with the same argument.

We substitute
$(0^{|V|}, 0, 0, 0, 0, 0; 1)$ by $(0^{|V|}, \epsilon, \epsilon, \epsilon, 0, 0; 1)$,
$(0^{|V|}, 1, 1, 0, 0, 0; 1)$ by $(0^{|V|}, 1-\epsilon, 1-\epsilon, \epsilon, 0, 0; 1)$, $\quad (**)$
$(0^{|V|}, 0, 1, 1, 0, 0; 1)$ by $(0^{|V|}, \epsilon, 1-\epsilon, 1-\epsilon, 0, 0; 1)$,
$(0^{|V|}, 1, 0, 0, 0, 0; 0)$ by
$(0^{|V|}, 1+0.5\epsilon_1, -\epsilon_1, -0.5\epsilon_1, 0, 0; 0)$ $(*)$ and $(0^{|V|}, 1+\epsilon_2, -0.5\epsilon_2, -0.5\epsilon_2, 0, 0; 0)$,
$(0^{|V|}, 0, 1, 0, 0, 0; 0)$ by $(0^{|V|}, -\epsilon_1, 1+0.5\epsilon_1, -0.5\epsilon_1, 0, 0; 0)$ $(*)$,
$(0^{|V|}, -0.5\epsilon_2, 1+\epsilon_2, -0.5\epsilon_2, 0, 0; 0)$, and $(0^{|V|}, -0.5\epsilon_3, 1+0.5\epsilon_3, -\epsilon_3, 0, 0; 0)$,
$(0^{|V|}, 0, 0, 1, 0, 0; 0)$ by
$(0^{|V|}, -0.5\epsilon_1, -0.5\epsilon_1, 1+\epsilon_1, 0, 0; 0)$ and $(0^{|V|}, -0.5\epsilon_2, -\epsilon_2, 1+0.5\epsilon_2, 0, 0; 0)$,
$(0^{|V|}, 1, 1, 1, 0, 0; 0)$ by
$(0^{|V|}, 1+0.5\epsilon_1, 1+0.5\epsilon_1, 1+\epsilon_1, 0, 0; 0)$ $(*)$ and $(0^{|V|}, 1+\epsilon_2, 1+0.5\epsilon_2, 1+0.5\epsilon_2, 0, 0; 0)$,
where $0 < \epsilon, \epsilon_1, \epsilon_2, \epsilon_3 < 0.5$ can be chosen independently for each point. Note that some points are substituted by two or three sets, respectively, corresponding to their different role to exclude Case 4. E.g. the points $(*)$ form a triangle enforcing the following: If the point $(**)$ is to be separated by any hyperplane from this triangle then the normal vector of the hyperplane has necessarily the signs $(+,+,+)$. Analogous triangles can be found for the other two positive points and hence Case 4 is excluded.

- In Theorem 10 the same substitutions can be performed for the points to exclude Case 4. The points to exclude Case 3 become
  $(0^{|V|}, 0, 0, 0, 1+\epsilon, 0.5, 1; 1)$
  $(0^{|V|}, 0, 0, 0, 1.5-\epsilon, 0.5, 1; 0)$
  $(0^{|V|}, 0, 0, 0, -1-\epsilon, 0.5, 1; 1)$
  $(0^{|V|}, 0, 0, 0, -1.5+\epsilon, 0.5, 1; 0)$
  $(0^{|V|}, 0, 0, 0, 6+\epsilon, 5.5, 1; 1)$
  $(0^{|V|}, 0, 0, 0, 6.5-\epsilon, 5.5, 1; 0)$
  $(0^{|V|}, 0, 0, 0, -6-\epsilon, 5.5, 1; 1)$
  $(0^{|V|}, 0, 0, 0, -6.5+\epsilon, 5.5, 1; 0)$
  $(0^{|V|}, 0, 0, 0, 0, -0.4+\epsilon, 1; 0)$

  for some $0 < \epsilon < 0.2$ which can be chosen independently for each point.

Hence the points can be substituted by points in appropriate line segments which are to be intersected with the regions in which the classification of the optimum solutions constructed above does not change. Since the multiplicities of the points are polynomial, appropriate coefficients of the substituting points can be computed in polynomial time.

Define the points $\boldsymbol{o}_{ij}$ together with their labeling by

$$(0, \ldots, 0, -\epsilon, 0, \ldots, 0, -\epsilon, 0, \ldots; 1)$$

with nonzero coefficients at positions i and j, and substitute $\boldsymbol{e}_i$ by points $\boldsymbol{e}_i^k$ ($k \in \{1, \ldots, d_i\}$) which together with their labeling are

$$(0, \ldots, 0, 1-\epsilon, 0, \ldots, 0, \ldots; 0)$$

where $0 < \epsilon < 0.5$ can be chosen independently for each point. The points have the following property:

> ($*$) The normal vector of any line which separates $\boldsymbol{e}_i^k$ or $\boldsymbol{e}_j^k$ from $\boldsymbol{o}_{ij}$ and $\boldsymbol{e}_{ij}$ has the signs $(+,-)$ or $(-,+)$, respectively, in dimensions i and j.

Hence we can establish properties **(v)** and **(vi)** as follows: An optimum solution of an instance of the MAX-k-cut problem gives rise to a solution of the corresponding instance of the loading problem where at most the points $\boldsymbol{e}_{ij}$ corresponding to monochromatic edges are misclassifed if the parameters ϵ are small enough. Conversely, any optimum solution of the loading problem classifies for each $p \in P_0$ at least one p_i^k correct and hence the same geometrical situations as before take place. Furthermore, we can assume that for each $\boldsymbol{e}_i$ at least one $\boldsymbol{e}_i^k$ is correct, otherwise a weight change would lead to a solution of at least the same quality and all $\boldsymbol{e}_i^k$ correct. W.l.o.g. $\gamma > 0$ if one adapts the proof of Theorems 8 and 10; or w.l.o.g. the second part of the network computes an AND when adapting the proof of Theorem 7. One can define the j^{th} cut via $\{v_i \mid$ the j^{th} hidden neuron in the first hidden layer maps $\boldsymbol{e}_i^k$ to 0 and v_i is not in the $1^{\text{st}}, \ldots, (j-1)^{\text{st}}$ cut $\}$, putting all remaining nodes in the first cut if adapting Theorem 7. One can define the first cut via $\{v_i \mid a_i > 0\}$ if adapting Theorems 8 and 10. At most those edges (v_i, v_j) are monochromatic where either $\boldsymbol{e}_{ij}$ or $\boldsymbol{o}_{ij}$ is misclassified because only one hyperplane cannot separate both $\boldsymbol{e}_{ij}$ and $\boldsymbol{o}_{ij}$ from $\boldsymbol{e}_i^k$ and $\boldsymbol{e}_j^k$ in Theorem 7 or because of ($*$) in Theorems 8 and 10. Hence property **(v)** holds. In the same way, property **(vi)** can be established. □

3.4 Correlated Architecture and Training Set Size

The reductions in the previous sections deal with situations where the number of examples is much larger than the number of hidden neurons. This may be unrealistic in some practical applications where one would allow larger architectures if a large amount of data is to be trained. One reasonable strategie would be to choose the size of the archtecture such that valid generalization can be expected using the well known bounds in the agnostic or PAC setting [24].

Naturally the question arises about what happens to the complexity of training if one is restricted to situations where the number of examples is limited with respect to the number of hidden neurons. One extreme position would be to allow the number of training examples to be at most equal to the number of hidden neurons. Although this may not yield valid generalization, the decision version of the loading problem becomes trivial because of [23]:

Observation 13 *If the number of hidden neurons in the first hidden layer is at least equal to the number of training examples and the activation function is the threshold activation* *function, the standard sigmoidal function, or the semilinear activation function (or any activation function σ such that the class of σ-networks possesses the universal approximation capability as defined in [23]) then the error of an optimum solution of the loading problem is determined by the number of contradictory points in the training set (i.e. points $(x; y_1)$ and $(x; y_2)$ with $y_1 \neq y_2$.)*

The following theorem yields a NP-hardness result if we restrict to situations where the number of examples and hidden neurons are correlated.

Theorem 14 *Approximate loading with relative error smaller than c/k^5 (c is a constant) is NP-hard for the loading problem with instances $(\mathcal{A}, P)$ where $\mathcal{A}$ is a layered $(n, k, 1)$-H-architecture, n and k are allowed to vary, $P \subset \mathbf{Q}^n \times \{0, 1\}$ is an example set with $|P| \leq k^4$, and all examples in P can be loaded correctly with some network of the architecture $\mathcal{A}$.*

Proof: The proof is via L-reduction from the MAX-3-cut problem. The algorithms T_1 and T_2, respectively, will be defined in two steps: mapping an instance of the MAX-3-cut problem to an instance of the MAX-k-cut problem with appropriate k and then to an instance of the loading problem as in Theorem 5, afterwards, or mapping a solution for the loading problem to a solution of the MAX-k-cut problem and then to a solution of the MAX-3-cut problem afterwards, respectively.

We first define T_1: given a graph (V, E) define $k = |V| \cdot |E|$ (w.l.o.g. $k \geq 3$) and (V', E') with $V' = V \cup \{v_{|V|+1}, \ldots, v_{|V|+k-3}\}$, $E' = E \cup \{(v_i, v_j) \mid i \in \{|V|+1, \ldots, |V|+k-3\}, j \in \{1, \ldots, |V|+k-3\} \setminus \{i\}\}$ where the new edges in E' have the multiplicity $2|E|$, meaning that $2|E|$ copies of each new edge are contained in E'. Reduce (V', E') to a loading problem for the architecture with $n = |V'| + 3$, k as above, and examples

1. $2|E'|$ copies of the origin $(0^n; 1)$,
2. d_i copies of the point $\boldsymbol{e}_i$, i.e. $(0, \ldots, 0, 1, 0, \ldots, 0; 0)$ (the 1 is at the i^{th} position from left) for each node $v_i \in V'$ where d_i is the degree of v_i,
3. a vector $\boldsymbol{e}_{ij}$ for each edge $(v_i, v_j) \in E'$: $(0, \ldots, 0, 1, 0 \ldots, 0, 1, 0, \ldots, 0; 1)$ (the numbers 1 are at the i^{th} and j^{th} positions from left),
4. $2|E'|$ copies of each of the points $(0^{|V'|}, \boldsymbol{p}^{ij}, 1; 1)$, $(0^{|V'|}, \boldsymbol{n}^{ij}, 1; 0)$, where $\boldsymbol{p}^{ij}$ and $\boldsymbol{n}^{ij}$ are constructed as follows: define the points $\boldsymbol{x}^{ij} = (4(i-1)+j, j(i-1)+4((i-2)+\ldots+1))$ for $i \in \{1, \ldots, k\}$, $j \in \{1, 2, 3\}$. These $3k$ points have the property that if three of them lie on one line then we can find an i such that the three points coincide with $\boldsymbol{x}^{i1}$, $\boldsymbol{x}^{i2}$, and $\boldsymbol{x}^{i3}$. Now we divide each point into a pair $\boldsymbol{p}^{ij}$ and $\boldsymbol{n}^{ij}$ of points which are obtained by a slight shift of $\boldsymbol{x}^{ij}$ in a direction that is orthogonal to the line $[\boldsymbol{x}^{i1}, \boldsymbol{x}^{i3}]$ (see Fig. 8). Formally, $\boldsymbol{p}^{ij} = \boldsymbol{x}^{ij} + \epsilon \boldsymbol{N}_i$ and $\boldsymbol{n}^{ij} = \boldsymbol{x}^{ij} - \epsilon \boldsymbol{N}_i$, where $\boldsymbol{N}_i$ is a normal vector of the line $[\boldsymbol{x}^{i1}, \boldsymbol{x}^{i3}]$ with a positive second coefficient and ϵ is a small positive value. ϵ can be chosen in such a way that the following holds:

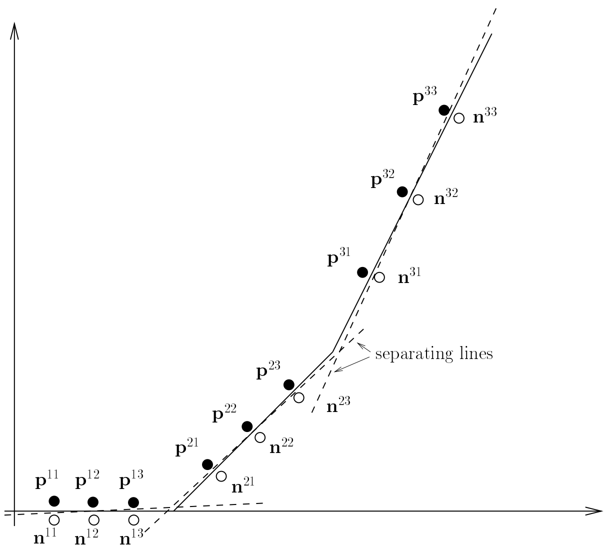


Figure 8: Construction of $\boldsymbol{p}^{ij}$ and $\boldsymbol{n}^{ij}$: The points result by dividing each point x^{ij} on the lines into a pair.

> Assume one line separates three pairs $(\boldsymbol{n}^{i_1j_1}, \boldsymbol{p}^{i_1j_1})$, $(\boldsymbol{n}^{i_2j_2}, \boldsymbol{p}^{i_2j_2})$, and $(\boldsymbol{n}^{i_3j_3}, \boldsymbol{p}^{i_3j_3})$, then the three pairs necessarily correspond to the three points on one line, which means $i_1 = i_2 = i_3$.

This property is fulfilled if the determinant of all triples of points $(1, y_1^{ij}, y_2^{ij})$, where (y_1^{ij}, y_2^{ij}) is some point in $[\boldsymbol{x}^{ij} + \epsilon N, \boldsymbol{x}^{ij} - \epsilon N]$ and the indices i are different in such a triple, does not vanish. Using Proposition 6 of [19] it is sufficient for this purpose to choose $\epsilon \leq 1/(24 \cdot k(k-1) + 6)$ if N is a vector of length 1. Consequently, the representation of the points $\boldsymbol{n}^{ij}$ and $\boldsymbol{p}^{ij}$ is polynomial in n and k.

Note that the number of points equals $5|E'| + 12k|E'| \leq (|V||E|)^4 = k^4$ for large $|V|$. An optimum solution of the instance of the MAX-3-cut problem gives rise to a solution of the instance of the MAX-k-cut problem with the same number of monochromatic edges via mapping the nodes in $V \cap V'$ to the same three cuts as before and defining the i^{th} cut by $\{v_{|V|+i}\}$ for $i \in \{1, \ldots, k-3\}$. This solution can be used to define a solution of the instance of the loading problem as follows: The j^{th} weight of neuron i in the hidden layer is chosen as $\begin{cases} -1 & \text{if } v_j \text{ is in the } i^{\text{th}} \text{ cut} \\ 2 & \text{otherwise,} \end{cases}$ and the bias is chosen as 0.5. The weights $(|V'|+1, |V'|+2, |V'|+3)$ of the i^{th} neuron are chosen as $(-i+1, 1, -0.5 + 2 \cdot i(i-1))$ which corresponds to the line through the points $\boldsymbol{x}^{i1}$, $\boldsymbol{x}^{i2}$, and $\boldsymbol{x}^{i3}$. The output unit has the bias $-k + 0.5$ and weights 1, i.e. it computes an AND. With this choice of weights one can compute that all examples except the points $\boldsymbol{e}_{ij}$ corresponding to monochromatic edges are mapped correctly.

Conversely, an optimum solution of the loading problem classifies all points in 1, 2, and 4 and all points $\boldsymbol{e}_{ij}$ corresponding to edges in $E'\backslash E$ correct because of the multiplicities of the respective points. We can assume that the activations of the neurons do not exactly coincide with 0 when the outputs on P are computed. Consider the mapping which is defined by the network on the plane $\{(0,\ldots,0,x_{n+1},x_{n+2},1)\,|\,x_{n+1},x_{n+2}\in\mathbf{R}\}$. The points $\boldsymbol{p}^{ij}$ and $\boldsymbol{n}^{ij}$ are contained in this plane. Because of the different outputs each pair $(\boldsymbol{p}^{ij},\boldsymbol{n}^{ij})$ is to be separated by at least one line defined by the hidden neurons. A number $3k$ of such pairs exists. Therefore, each of the lines defined by the hidden neurons necessarily separates three pairs $(\boldsymbol{p}^{ij},\boldsymbol{n}^{ij})$ with $j\in\{1,2,3\}$ and nearly coincides with the line defined by $[\boldsymbol{x}^{i1},\boldsymbol{x}^{i3}]$. Denote the output weights of the network by $w_1,\ldots,w_k$ and the output bias by θ. We can assume that the i^{th} neuron nearly coincides with the ith line and that the points $\boldsymbol{p}^{ij}$ are mapped by the neuron to the value 0. Otherwise we change all signs of the weights and the bias in neuron i, we change the sign of the weight w_i, and increase θ by w_i. But then the points $\boldsymbol{p}^{i2}$ are mapped to 0 by all hidden neurons, the points $\boldsymbol{n}^{i2}$ are mapped to 0 by all but one hidden neuron. This means that $\theta>0$, $\theta+w_i<0$ for all i and therefore $\theta+w_{i_1}+\ldots+w_{i_l}<0$ for all $i_1,\ldots,i_l\in\{1,\ldots,k\}$ with $l\geq 1$. This means that the output unit computes the function NAND : $(x_1,\ldots,x_n)\mapsto\neg x_1\wedge\ldots\wedge\neg x_n$ on binary values.

Define a solution of the instance of the MAX-k-cut problem by setting the i^{th} cut c_i as $\{v_j\,|\,\text{the } i\text{ th hidden neuron maps } \boldsymbol{e}_j \text{ to } 1\}\backslash(c_1\cup\ldots\cup c_{i-1})$. Because of the classification of the points $\boldsymbol{e}_i$ all nodes are contained in some cut. Assume some edge (v_i,v_j) is monochromatic. Then $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are mapped to 1 by the same hidden neuron, hence the vector $\boldsymbol{e}_{ij}$ is mapped to 1 as well because of the classification of the origin. Therefore $\boldsymbol{e}_{ij}$ is classified wrong. Note that all $\boldsymbol{e}_{ij}$ corresponding to edges in $E\backslash E'$ are correct, hence the nodes $v_{|V|+1},\ldots,v_{|V|+k-3}$ each form one cut and the remaining nodes are contained in the remaining three cuts. Hence these three cuts define a solution of the instance of the MAX-3-cut problem such that almost edges corresponding to misclassified $\boldsymbol{e}_{ij}$ are monochromatic.

Denote by opt_1 the value of an optimum solution of the MAX-3-cut problem and by opt_2 the optimum value of the loading problem. We have shown that

$$\text{opt}_2=\frac{|E|\text{opt}_1+(|E'|-|E|)+4|E'|+12|E'|k}{5|E'|+12|E'|k}\leq\frac{3}{2}\,\text{opt}_1\,.$$

Next we construct T_2. Assume that a solution of the loading problem with relative error smaller than c/k^5 is given. Then the points 1 and 4 are correct due to their multiplicities. Otherwise the cost of the problem would be at most

$$\frac{12|E'|k+4|E'|}{5|E'|+12|E'|k}\leq 1-\frac{c}{k^5}$$

for $c\geq 1$, $k\geq 2$. As before we can assume that the output neuron computes the function $\boldsymbol{x}\mapsto\neg x_1\wedge\ldots\wedge\neg x_k$. Define opt_2 to be the value of an optimum solution of the loading problem and I_2 the value of the given solution. Assume some point $\boldsymbol{e}_{ij}$ corresponding to an edge in $E'\backslash E$ is misclassified. Then T_2 yields an arbitrary solution of the MAX-3-cut problem. For the quality I_1 of this solution compared to an optimum opt_1 we can compute

$$\text{opt}_1 - I_1 \leq 1 \leq \frac{5|E'| + 12|E'|k}{|E|}(\text{opt}_2 - I_2)\,.$$

This holds because an optimum solution of the loading problem classifies at least a number of $|E|$ points more correct than in the solution considered here.

If all $\boldsymbol{e}_{ij}$ corresponding to edges in $E' \backslash E$ are correct then we define a solution of the MAX-3-cut problem via the activation of the hidden neurons as above. Remaining nodes become members of the first cut. An argument as above shows that each monochromatic edge comes from a misclassification of either $\boldsymbol{e}_i$, $\boldsymbol{e}_j$, or $\boldsymbol{e}_{ij}$. Hence

$$\text{opt}_1 - I_1 \leq \frac{5|E'| + 12|E'|k}{|E|}(\text{opt}_2 - I_2)\,.$$

Setting $\alpha = 3/2$, $\beta = \tilde{c} \cdot k^3 \geq (5|E'| + 12|E'|k)/|E|$ for some constant $\tilde{c}$ and using Theorem 2 yields the result as stated above. □

4 Hardness of Approximating the Failure Ratio Function m_f

In this section we consider another objective function which is more important if a large amount of training data is misclassified. Given a network architecture and a training set we are interested in finding weights such that the number of misclassified points it bounded from above by the minimum achievable number of errors times a constant. Formally, given an instance x of the loading problem, denote by $m_C(x, y)$ the number of points in the training set which are not classified correctly by the network represented by y. Now given a constant c, we want to find weights such that $\text{opt}_C(x) \leq m_C(x, y) \leq c \cdot \text{opt}_C(x)$ where y denotes the resulting network. Notice that in the case when $\text{opt}_C(x) > 0$, this is equivalent to investigating if the failure ratio m_f can be bounded above by a constant and hence this problem is referred to as the problem of *approximating the minimum failure ratio* within a constant c while learning in the presence of errors [1]. If restricted to situations where a solution without errors exists this only yields the original loading problem since no errors are allowed in the approximation either. Hence we restrict to situations in the following where no solution without misclassified points exists.

We want to show NP-hardness of approximation of m_f. It turns out that the approximation bound for hardness is *independent* of the number of neurons of the network architecture. For our purpose we use a reduction from the set-covering problem.

Definition 15 (Set Covering Problem [9]) *Given a set of points $S = \{s_1, \ldots, s_p\}$ and a set of subsets $C = \{C_1, \ldots, C_m\}$, find indices $I \subset \{1, \ldots, m\}$ such that $\bigcup_{i \in I} C_i = S$. In this case the sets $C_i, i \in I$, are called a cover of S. A cover is called exact if the sets in a cover are mutually disjoint.*

For the set-covering problem the following holds, showing that it is hard to approximate within every factor $c > 1$:

Theorem 16 *[4] For every $c > 1$ there is a polynomial time reduction that, given an instance φ of SAT, produces an instance of the set-covering problem and a number $K \in \mathbf{N}$ with the properties: if φ is satisfiable then there exists an exact cover of size K, if φ is not satisfiable then every cover has size at least $c \cdot K$.*

Using Theorem 16 Arora et.al. [1] show that approximating the minumum failure ratio is NP-hard for the simple perceptron model for every constant $c > 1$ if the threshold is dropped. Following their argument one can obtain a similar result for arbitrary layered H-nets where the input thresholds are fixed to 0.

Theorem 17 *Assume that we are given a layered H-net where the thresholds of the neurons in the first hidden layer are fixed to 0 and the input dimension varies. Let $c > 1$ be any given constant. Then the problem of approximating minimum failure ratio while learning in the presence of errors within a factor of c is NP-hard.*

Proof: Because of [1] we can assume that the network contains at least one hidden layer. Assume that we are given a formula φ. Transform this formula (with the given constant c) to an instance $(S = \{s_1, \ldots, s_p\}, C = \{C_1, \ldots, C_m\})$ of the set-covering problem and a constant K such that the properties in Theorem 16 hold. Transform this instance of the set-covering problem to an instance of the loading problem for the given architecture with input dimension $n = |C| + 2 + n_1 + 1$ (where n_1 denotes the number of hidden neurons in the first hidden layer) and the following examples from $\mathbf{Q}^n \times \{0, 1\}$:

(I) $(\mathbf{e}_i, 0, 1, 0^{n_1+1}; 1)$, $(-\mathbf{e}_i, 0, 1, 0^{n_1+1}; 1)$, where $\mathbf{e}_i$ is the i^{th} unit vector in $\mathbf{R}^{|C|}$,

(II) $c \cdot K$ copies of each of the points $(\mathbf{e}_{s_i}, -1, 1, 0^{n_1+1}; 1)$, $(-\mathbf{e}_{s_i}, 1, 1, 0^{n_1+1}; 1)$, where $\mathbf{e}_{s_i} \in \{0, 1\}^{|C|}$ is the vector with j^{th} component as 1 if and only if $s_i \in C_j$, $i \in \{1, \ldots, p\}$,

(III) $c \cdot K$ copies of $(0^{|C|}, 1, 0, 0^{n_1+1}; 1)$, $(0^{|C|}, 1/(2m), 1, 0^{n_1+1}; 1)$, $(0^{|C|}, -1/(2m), 1, 0^{n_1+1}; 0)$, where the component $|C| + 1$ is nonzero in all three points and the component $|C| + 2$ is nonzero in the latter two points, $m = |C|$,

(IV) $c \cdot K$ copies of each of $(0^{|C|+2}, \boldsymbol{p}_i, 1; 0)$, $(0^{|C|+2}, \boldsymbol{p}_0, 1; 1)$, $(0^{|C|+2}, \tilde{\boldsymbol{z}}_i, 1; 1)$, $(0^{|C|+2}, \bar{\boldsymbol{z}}_i, 1; 0)$, where the points $\boldsymbol{p}_i$, $\tilde{\boldsymbol{z}}_i$, $\bar{\boldsymbol{z}}_i$ are the same points as in the proof of Theorem 7.

For an exact cover of size K, let the corresponding set of indices be $I = \{i_1, \ldots, i_K\}$. Define the weights of a threshold network such that the i^{th} neuron in the first hidden layer has the weights $(\mathbf{e}_I, 1, 1/(4m), \boldsymbol{e}_i, 0)$ where the j^{th} component of $\mathbf{e}_I \in \{0, 1\}^{|S|}$ is 1 if and only if $j \in I$ and $\mathbf{e}_i$ is the i^{th} unit vector in $\mathbf{R}^{n_1}$. The remaining neurons in the other layers compute an AND. Since the cover is exact, this maps all examples correctly except K examples in **(I)** which correspond to sets in the cover.

Conversely, assume that every cover has size at least $c \cdot K$. Assume some weight setting misclassifies less than $c \cdot K$ examples. We can assume that the activation of every neuron is different from 0 on the training set: for the examples in **(IV)** the weight w_n serves as a threshold, for the points in **(I)**, **(II)**, and **(III)** except for $(0^{|C|}, 1, 0^{n_1+2}; 1)$ the weight $w_{|C|+2}$ serves as a threshold, hence one can slightly change the respective weight which serves as a threshold without changing the classification of these examples such that the activation becomes nonzero. Assuming that the activation of $(0^{|C|}, 1, 0^{n_1+2}; 1)$ is zero we can slightly increase the weight $w_{|C|+1}$ such that the sign of the activation of all other points which are affected does not change. Because of the multiplicity of the examples we can assume that the examples in **(II)-(IV)** are correctly classified. We can assume that the network function has the form $\beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}) = f_1(\boldsymbol{x}) \wedge \ldots \wedge f_{n_1}(\boldsymbol{x})$ where f_i is the function computed by the i^{th} hidden neuron in the first hidden layer because of the points in **(IV)**. This is due to the fact that the points $\tilde{\boldsymbol{z}}_i$ and $\bar{\boldsymbol{z}}_i$ enforce the respective weights of the hidden neurons in the first hidden layer to nearly conincide with weights describing the hyperplane with i^{th} coefficient zero. Hence the points $\boldsymbol{p}_i$ are mapped to the entire set $\{0, 1\}^{n_1}$ by the hidden neurons in the first hidden layer and determine the remainder of the network function. Hence all neurons in the first hidden layer classify all positive examples except less than $c \cdot K$ points of **(I)** correctly and there exists one neuron in the first hidden layer which classifies the negative example in **(III)** correctly as well. Consider this last neuron. Denote by $\boldsymbol{w}$ the weights of this neuron. Because of **(III)**, $w_{|C|+1} > 0$. Define $I = \{i \in \{1, \ldots, |C|\} \,|\, |w_i| \geq w_{|C|+1}/(2m)\}$.

Assume that $\{C_i \,|\, i \in I\}$ forms a cover. Because of **(III)** we find $w_{|C|+1}/(2m) + w_{|C|+2} > 0$ and $-w_{|C|+1}/(2m) + w_{|C|+2} < 0$. Therefore one of the examples in **(I)** is classified wrong for every $i \in I$. This leads to at least $c \cdot K$ misclassified examples because every cover is of size at least $c \cdot K$.

Assume that $\{C_i \mid i \in I\}$ does not form a cover. Then one can find for some $i \leq |S|$ and the point $(\mathbf{e}_{s_i}, -1, 1, 0^{n_1+1})$ in **(II)** an activation $< m \cdot w_{|C|+1}/(2m) - w_{|C|+1} + w_{|C|+2} = w_{|C|+2} - w_{|C|+1}/2$ which is < 0 because $-w_{|C|+1}/(2m) + w_{|C|+2} < 0$, $w_{|C|+1} > 0$ **(III)**. This yields a misclassified example with multiplicity $c \cdot K$. □

Since an arbitrary constant $c > 1$, which is in particular independent of the architecture, may be used in the above theorem, this theorem suggests that in the presence of errors training may be extremely difficult.

5 Conclusion and open questions

We have shown the NP-hardness of finding approximate solutions for the loading problem in several different situations. They can be seen as generalizations of the classical result of [6] to more realistic situations. We have considered the question as to whether approximating relative error within a constant factor is NP-complete. Compared to [2] we considered arbitrary multilayer threshold networks, and the $(n, 2, 1)$-network with the sigmoidal (with ϵ-separation) or the semilinear activation function. Furthermore, we discussed how to avoid training using multiple copies of the example in the hardness results. We also considered the case where the number of examples is correlated to the number of hidden neurons. Investigating the problem of minimizing the failure ratio in the presence of errors yields NP-hardness within every constant factor $c > 1$ for multilayer threshold networks (with input biases fixed to 0).

Several problems still remain open in this context, some of which are unsolved even if we ask the existence of an exact solution instead of an approximate solution:

1. What is the complexity of training multilayer threshold networks if restricted to binary examples? In [6], Blum and Rivest show the NP-completeness for the $(n, 2, 1)$ architecture and binary examples. For more hidden neurons this is solved neither in the exact nor in the approximate setting.

2. What is the complexity of training $(n, n_1, 1)$ networks with the sigmoidal activation function in the hidden nodes? For the exact setting, [13, 25] show some situations to be NP-hard; however, they consider networks which are used for interpolation instead of classification, i.e., the quadratic error is to be minimized. The classification problem is a more easy task and hence NP-hardness seems more difficult to prove.

3. For which classes of activation functions can the result for the sigmoidal case still hold? Actually, we only use some properties of the sigmoid, such as the fact, that it is continuously differentiable, monotonous, symmetric, bounded and, which is the most important point, that in case 2 in the proof the set of points classified positive is convex.

4. What is the complexity of finding an approximate solution if the number of examples is restricted with respect to the number of hidden neurons? We obtained one result in this context, but only with error bounds which depend on the number of hidden neurons and hence the number of examples. Furthermore, although the decision version of the loading problem becomes trivial if the number of examples is at most equal to the number of hidden neurons in the first layer, a polynomial algorithm in order to find the actual weights is not obtained immediately from [23].

5. Can a general argument be found which will show the validity of the hardness results for examples without multiplicities? We used a step by step analysis. An argument showing that every pattern set with multiple points (maybe with additional conditions like general position) can be substituted by an equivalent pattern set without multiplicities would avoid that.

6. What are the characteristics of a set of examples for which loading is hard? Can, for example, an example set with limited correlation be loaded efficiently? Some investigation concerning this topic can be found in [21]. [5] show that the situation of [6] changes if the input examples come from a specific (realistic) input distribution. Then learning is possible in polynomial time.

References

[1] S. Arora, L. Babai, J. Stern and Z. Sweedyk, The hardness of approximate optima in lattices, codes and systems of linear equations, Journal of Computer and System Sciences, 54, pp. 317-331, 1997.

[2] P. Bartlett and S. Ben-David, Hardness results for neural network approximation problems, in Fischer P. and Simon H. U. (eds.), Computational Learning Theory, Lecture Notes in Artificial Intelligence 1572, Springer, pp. 639-644, 1999.

[3] R. Batruni, A multilayer neural network with piecewise-linear structure and backpropagation learning, IEEE Transactions on Neural Networks, 2, pp. 395-403, 1991.

[4] M. Bellare, S. Goldwasser, C. Lund, and A. Russell, Efficient multi-prover interactive proofs with applications to approximation problems, in Proceedings of the 25th ACM Symposium on the Theory of Computing, pp. 113-131, 1993.

[5] A.L. Blum and R. Kannan, Learning an intersection of K halfspaces over the uniform distribution, in Roychowdhury V. P., Siu K. Y., and Orlitsky A. (eds.), Theoretical Advances in Neural Computation and Learning, Kluwer Academic Publishers, pp. 337-356, 1994.

[6] A. Blum and R. L. Rivest, Training a 3-node neural network is NP-complete, Neural Networks 5, pp. 117-127, 1992.

[7] J. Brown, 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, pp. 43-47, 1988.

[8] B. DasGupta, H. T. Siegelmann, and E. D. Sontag, On the Intractability of Loading Neural Networks, in Roychowdhury V. P., Siu K. Y., and Orlitsky A. (eds.), Theoretical Advances in Neural Computation and Learning, Kluwer Academic Publishers, pp. 357-389, 1994.

[9] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Franscisco, 1979.

[10] B. Hammer, Some complexity results for perceptron networks, in Niklasson L., Bodén M., and Ziemke, T. (eds.), ICANN'98, Springer, pp. 639-644, 1998.

[11] B. Hammer, Training a sigmoidal network is difficult, in Verleysen M. (ed.), European Symposium on Artificial Neural Networks, D-Facto publications, pp. 255-260, 1998.

[12] K.-U. Höffgen, H.-U. Simon, and K. S. Van Horn, Robust trainability of single neurons, Journal of Computer and System Sciences 50(1), pp.114-125, 1995.

[13] L. K. Jones, The computational intractability of training sigmoidal neural networks, IEEE Transactions on Information Theory 43(1), pp. 167-713, 1997.

[14] J. S. Judd, Neural network design and the complexity of learning, MIT Press, Cambridge, MA, 1990.

[15] V. Kann, S. Khanna, J. Lagergren, and A. Panconesi, On the hardness of approximating max-k-cut and its dual, Technical Report CJTCS-1997-2, Chicago Journal of Theoretical Computer Science, 1997.

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

[17] W. Maass, G. Schnittger, and E. D. Sontag, A comparison of the computational power of sigmoid versus boolean threshold circuits, in Roychowdhury V. P., Siu K. Y., and Orlitsky A. (eds.), Theoretical Advances in Neural Computation and Learning, Kluwer Academic Publishers, pp. 127-151, 1994.

[18] A. Macintyre and E. D. Sontag, Finiteness results for sigmoidal 'neural' networks, in Proceedings of the 25th ACM Symposium on the Theory of Computing, San Diego, pp. 325-334, 1993.

[19] M. Megiddo, On the complexity of holyhedral separability, Discrete Computational Geometry 3, pp. 325-337, 1988.

[20] C. H. Papadimtriou and M. Yannakakis. Optimization, Approximation and Complexity Classes, Journal of Computer & System Sciences 43, pp. 425-440, 1991.

[21] M. Schmitt, Komplexität neuronaler Lernprobleme, Peter Lang, 1996.

[22] J. Šimà, Back-propagation is not efficient, Neural Networks 9(6), pp. 1017-1023, 1996.

[23] E. D. Sontag, Feedforward nets for interpolation and classification, Journal of Computer and System Sciences 45, pp.20-48, 1992.

[24] M. Vidyasagar, A theory of learning and generalization, Springer, 1997.

[25] V. H. Vu, On the infeasibility of training with small squared errors, in Jordan M. I., Kearns M. J., and Solla S. A. (eds.), Advances in Neural Information Processing Systems 10, MIT Press, pp. 371-377, 1998.