On Approximate Learning by Multi-layered Feedforward Circuits [*]

Bhaskar DasGupta [†‡] and Barbara Hammer [§]

July 17, 2002

Abstract

We deal with the problem of efficient learning of feedforward neural networks. First, we consider the objective to maximize the ratio of correctly classified points compared to the size of the training set. We show that it is NP-hard to approximate the ratio within some constant relative error if architectures with varying input dimension, one hidden layer, and two hidden neurons are considered where the activation function in the hidden layer is the sigmoid function, and the situation of epsilon-separation is assumed, or the activation function is the semilinear function. For single hidden layer threshold networks with varying input dimension and n hidden nodes, approximation within a relative error depending on n is NP-hard even if restricted to situations where the number of examples is limited with respect to n.

Afterwards, we consider the objective to minimize the failure ratio in the presence of misclassification errors. We show that it is NP-hard to approximate the failure ratio within any positive constant for a multilayered threshold network with varying input dimension and a fixed number of nodes in the hidden layer if the thresholds of the neurons in the first hidden layer are zero. Furthermore, even obtaining weak approximations is *almost* NP-hard in the same situation.

[*] An extended abstract of this paper appeared in 11th International Conference on Algorithmic Learning Theory, December 2000.

[†] Research supported by NSF grant CCR-9800086

[‡] Department of Computer Science, Rutgers University, Camden, NJ 08102, USA

[§] Department of Mathematics/Computer Science, University of Osnabrück, Germany

1 Introduction

Neural networks are a well established learning mechanism which offer a very simple method of learning an unknown hypothesis when some examples are given. 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. Commonly, they are trained very successfully with some modification of the backpropagation algorithm [35]. However, the inherent complexity of training neural network is till now an open problem for almost all practically relevant situations. In practice, a large variety of tricks and modifications of the representation of the data, the neural architecture, or the training algorithm is applied in order to obtain good results [28] - the methods are mostly based on heuristics. From a theoretical point of view, the following question is not yet answered satisfactorily: in which situations is training tractable or, conversely, does require a large amount of time? Until now it is only known that training a fixed network, as it appears in practice, is at least decidable assuming that the so called Schanuel conjecture holds [24]. In other words, till now it is only proved (up to some conjecture in pure mathematics) that training of standard neural networks can be performed on a computer in principle, but no bounds on the required amount of time have been derived in general. People have to rely on heuristics in order to design the training problems to ensure that training a neural network succeeds. In order to obtain theoretical guarantees and hints about which situations may cause troubles, researchers have turned to simpler situations in which theoretical results can be obtained. The main purpose of this paper is to consider situations which are closer to the training problems as they occur in practice.

In order to state the problems we are interested in, consider a standard *feedforward neural network*. Such a network consists of neurons connected in a directed acyclic graph. The overall behavior is determined by the *architecture* $\mathcal{A}$ and the network *parameters* $\boldsymbol{w}$. Given a *pattern set* P, i.e. a collection of points or *training examples*, and their labelings $(\boldsymbol{x}_i; y_i)$, we want to learn the regularity consistent with the mapping of the $\boldsymbol{x}_i$ points to the y_i points with such a network[1]. Frequently, this is performed by first choosing an architecture $\mathcal{A}$ which computes a function $\beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x})$ depending on $\boldsymbol{w}$. 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 training pattern $(\boldsymbol{x}_i; y_i)$. The *loading problem* is to find weights $\boldsymbol{w}$ for $\mathcal{A}$ such that these equalities hold for every pattern in P. The *decision version* of the loading problem is to decide (rather than to find the weights) whether such weights exist that load P onto $\mathcal{A}$. Obviously, finding optimal weights is at least as hard as the decision version of the loading problem.

Researchers have considered specific architectures for neural nets in which the so called activation function in the architecture is the threshold activation function, a particularly simple function. This function captures the asymptotic behavior of many common activation functions including the sigmoidal function, but it does not share other properties (such as differentiability). It has been shown that for every *fixed* threshold architecture training can be performed in polynomial time [13, 15, 25]. Starting with the work of Judd [19], researchers have considered situations where only architectural parameters are allowed to vary from one training instance to the next training instance in order to take into account that most existing training algorithms are uniform with respect to the architecture. That implies that common learning algorithms do not rely

[1] We will refer to both the point $\boldsymbol{x}_i$ and the point $\boldsymbol{x}_i$ together with its labeling $(\boldsymbol{x}_i; y_i)$, as a point or a training example. Note that an example $(\boldsymbol{x}_i; y_i)$ may occur more than once in P, i.e. the multiplicity of a point may be larger than 1.

on the number of neurons which are considered in the specific setting. Hence the complexity of the training problem should scale well with respect to the number of neurons. It turns out that the training problem is NP-hard in several situations, i.e., the respective problems are infeasible (under the standard complexity-theoretic assumption of P$\neq$NP [14]): Blum and Rivest [9] showed that a varying input dimension yields the NP-hardness of the training problem for architectures with only two hidden neurons using the threshold activation functions. The approaches in [15, 27] generalize this result to multilayered threshold networks. Investigation has been done to get around this boundary of NP-hardness of the training problem by using activation functions different from the threshold activation function. In fact, for some strange activation functions (which are not likely to be used in practice at all) or a setting where the number of examples and the number of hidden neurons coincide, loadability is trivial [31]. References [13, 16, 18, 30, 34] constitute approaches in order to generalize the NP-hardness result of Blum and Rivest to architectures with a continuous or the standard sigmoidal activation functions. Hence finding an optimum weight setting in a concrete learning task captured by the above settings may require a large amount of time.

However, most works in this area deal either with only very restricted architectures (e.g. only three nodes), an activation function not used in practice (e.g. the threshold function), or, generally, a training problem which is, in some sense, too strict compared to practical training situations. 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 loaded. Moreover, in the context of agnostic learning [33], a situation in which the neural architecture may not model the underlying regularity precisely, it may be possible that there are no choices for the weights that load a given set of examples. In structural complexity theory, it is common to investigate the possibility of proving NP-hardness of the decision versions of general optimization problems [14] and, moreover, the possibility of designing approximation algorithms for the original optimization problem together with guarantees on the quality of the approximate solutions returned by such an algorithm [12, 14]. A list of results concerning the complexity and approximability of various problems can be found, for example, in [3], where it can be seen that there are problems which can be approximated to within a high accuracy in polynomial time even though the problem itself is NP-complete. From these motivations, researchers have considered a modified version of the loading problem where the number of correctly classified points is to be maximized. References [1, 2, 17] consider the complexity of training single neurons with the threshold activation with some error ratio. The authors in [4] deal with depth 2 threshold networks. Formally, the maximization version of the loading problem (e.g., see [4]) L deals with a function m_L which is to be maximized: m_L computes the number of points in the training set P (counted with their multiplicities), such that $\beta_{\mathcal{A}}(\boldsymbol{w},\boldsymbol{x}) = y$, divided by the size of the training set P. That is, we would like to satisfy the largest possible fraction of the given collection of examples. We will consider this objective first and obtain NP-hardness results for approximately minimizing the relative error of m_L which deal with various more realistic activation functions and situations compared to [4]. In the second part of the paper, we consider another possible function for minimization which is used in [2] 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 (again counted with their multiplicities) produced by the learning algorithm to that of an optimal algorithm. We will obtain results of NP-hardness or almost NP-hardness, respectively, of approximating this failure ratio for multilayered threshold networks in the second 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 function m_L of a neural network within some error bound. For this purpose, following the approach in [4], 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 NP-hardness of approximation. Afterwards we apply this method to several situations as stated below. We show, as already shown in [4] except when $n_1 = 2$, that it is NP-hard to approximate m_L within a relative error smaller than $\varepsilon = 1/(68n_1 2^{n_1} + 136n_1^3 + 136n_1^2 + 170n_1)$ and multilayer threshold networks with varying input dimension and a *fixed* number n_1 of neurons in the first hidden layer for any $n_1 \geq 2$. In Section 3.2.1, we show that, for architectures with one hidden layer and two hidden neurons (the classical case considered by [9]), approximation of m_L with relative error smaller than $1/c$ 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 in the output 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 [6, 10, 13, 21]). As in [4] 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 smaller than c/k^3, for some positive constant c, is NP-hard even if restricted to situations where the number of examples is between $k^{3.5}$ and k^4. In the remaining part of the paper, we consider the objective to minimize the *failure ratio* m_f in the presence of misclassification errors (e.g., see [1, 2]) and show that it is NP-hard to approximate m_f within *any* constant $c > 1$ for a multilayered threshold network with varying input dimension and a fixed number of nodes in the first hidden layer if the thresholds of the nodes in the first hidden layer are zero. Assuming that NP $\not\subseteq$ DTIME$(n^{\text{poly}(\log n)})$ holds[2], a conjecture in structural complexity theory [3], we show that approximating m_f in the presence of errors for a multilayered threshold network with varying input dimension and a fixed number of nodes in the first hidden layer, in which the thresholds of the nodes in the first hidden layer are fixed to 0, within a factor of $2^{\log^{0.5-\epsilon} n}$, n denoting the varying number of input neurons in the respective instance, is not possible in polynomial time, where $\epsilon > 0$ is any fixed constant. Finally, we conclude in Section 5 with some open problems worth investigating further.

2 The Basic Model and Notations

The architecture of a feedforward net is described by a directed acyclic interconnection graph and the activation functions of the nodes. A node (processor or neuron) v of the network computes a function

$$\gamma\left(\sum_{i=1}^{k} w_i u_i + \theta\right)$$

[2] This assumption is referred to as being "almost NP-hardness" in the literature (e.g., see [2]).

of its inputs $u_1, \ldots, u_k$. The term $\sum_{i=1}^{k} w_i u_i + \theta$ is called the *activation* of the node v. The inputs u_i 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_i (resp. θ) are the *weights* (resp. *threshold*) of node v, and the function γ is the *activation function* of v. The output of a designated node provides the output of the network. An *architecture* specifies the interconnection graph and the γ's of each node, 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 in the acyclic interconnection graph. The depth of a node is the length of the longest path in the graph which ends in that node. 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. 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 at 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 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 $\mathbb{R}$. 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 $\beta_{\mathcal{A}} : \mathbb{R}^{r+n} \mapsto \mathbb{R}$. 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 $\text{sgd}(x) = 1/(1+\mathrm{e}^{-x})$.

In the learning context, the *loading problem* (e.g., see [13]) L is defined as follows: Given an architecture $\mathcal{A}$ and a collection P of training examples $(\boldsymbol{x}; y) \in \mathbb{R}^n \times \mathbb{R}$ (we allow multiplicities, i.e., an example may be contained more than once in a training set), find weights $\boldsymbol{w}$ so that for all pairs $(\boldsymbol{x}; y) \in P$:

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

Note that both, the architecture and the training set, are part of the input in general. In this paper we will deal with classification tasks where $y \in \{0, 1\}$ instead of $y \in \mathbb{R}$. Clearly, the NP-hardness results obtained with this restriction will be valid in the unrestricted case also. 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 (minimization)* problem C is characterized by a non-negative cost 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 (minimize) $m_C(x, y)$ for any particular x. Denote by $\text{opt}_C(x)$ (or simply by $\text{opt}(x)$ if the problem C is clear from the context) the maximum (minimum) value of $m_C(x, y)$. The two objectives that are of relevance to this paper are as follows (assume that x is the instance (architecture and training set) and y is a solution (values of weights) for x):

Success ratio function m_L:

$$m_L(x, y) = \frac{\text{number of examples } \boldsymbol{x}_i \text{ such that } \beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}_i) = y_i}{\text{size of } P}$$

(e.g., see [4]). Note that the examples are counted *with multiplicities* if they are contained in P more than once. In other words, m_L is the fraction of the correctly classified points compared to all points in a training set. Notice that $0 < \text{opt}_L(x) \leq 1$ holds for all instances x. The *relative error* of a solution y is the quantity $(\text{opt}_L(x) - m_L(x,y))/\text{opt}_L(x)$. Our interest in this paper lies in investigating the complexity of finding a solution such that the relative error is bounded from above by some constant.

Failure ratio function m_f: Define by $m_C(x,y)$ the number of examples $\boldsymbol{x}_i$ (counted with multiplicities) such that $\beta_{\mathcal{A}}(\boldsymbol{w}, \boldsymbol{x}_i) \neq y_i$. Then, provided $\text{opt}_C(x) > 0$,

$$m_f(x,y) = m_C(x,y)/\text{opt}_C(x)$$

(e.g., see [2]). 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 misclassification is unavoidable*. Our interest in this paper lies in investigating the complexity of finding a solution such that m_f is smaller than some value.

The NP-hardness of approximately optimizing these objctives will be the topic of this paper. For convenience, we repeat the formal definition of maximization/minimization as introduced above:

Definition 1 *A* maximization/minimization problem C *consists of a set of instances $\mathcal{I}$, a set of possible solutions $\mathcal{S}(x)$ for each $x \in \mathcal{I}$, and a cost function $m_C : (x,y) \in \mathcal{I} \times \mathcal{S}(x) \to \mathbb{R}^+$ ($\mathbb{R}^+$ denoting the positive real numbers) which computes the cost of a solution y for an instance x of the poblem. We assume that for each instance x a solution y with optimum, i.e. maximum/minimum value $m_C(x,y)$ exists. Denote by* $\text{opt}_C(x)$ *the respective optimum value, i.e.* $\text{opt}_C(x) = \max\{m_C(x,y) \mid y \in \mathcal{S}(x)\}$ *if C is a maximization problem and* $\text{opt}_C(x) = \min\{m_C(x,y) \mid y \in \mathcal{S}(x)\}$ *if C is a minimization problem, respectively.*

Assume $k \in]0,1[$ is some constant. Then approximating the relative error of the maximization problem C within the constant k is NP-hard *if every problem in NP can be reduced in polynomial time to the following problem: given an instance x of C, find a solution y such that the relative error can be limited by* $(\text{opt}_C(x) - m_C(x,y))/\text{opt}_C(x) < k$.

Assume $k > 1$ is a constant. Assume C is a minimization problem where by definition of m_C the fact $\text{opt}_C(x) > 0$ holds for all instances x. Then appoximation of the relative cost of the minimization problem within constant k is NP-hard *if every problem in NP can be reduced in polynomial time to the following problem: given an instance x of C, find a solution y such that the costs can be limited by $m_C(x,y)/\text{opt}_C(x) < k$.*

In our case instances are given by neural architectures and training sets, solutions are possible choices of the weights of the neural architecture. As already mentioned, we will deal with the following two objectives: minimizing the failure ratio function m_f and minimizing the relative error of the success ration function m_L, respectively. Note that both objectives, minimizing the relative error and minimizing the misclassification ratio, respectively, are defined via the cost function m_C or m_L, respectively, of the underlying maximatization or minimization problem, respectively. We will in the following refer to the above notation of NP-hardness as the NP-hardness of approximating C or the respective cost, respectively.

Depending on the minimum number of misclassifications that are unavoidable in a training scenario, the two objectives we are interested in can be related. Assume an input instance x of a training scenario and a solution y are given. Denote by P the size of the given training set, by opt the maximum number of points which can be classified correctly, and assume opt $< P$. Assume the number of points classified correctly by y is r. We would like to relate the two objectives we are interested in. Assume that the relative error of the success ratio function is smaller than some value $C \in]0,1[$. Hence $r >$ opt$(1-C)$. Hence the failure ratio can be limited by $(P-r)/(P-\text{opt}) < (P-\text{opt}(1-C))/(P-\text{opt}) = 1+\text{opt}/(P-\text{opt})\cdot C$. If a large number of errors is unavoidable, i.e. opt is much smaller than P, the term opt$/(P-\text{opt})$ is small. I.e. in this case bounds on the relative error of the success ratio can be transformed to small bounds on the failure ratio function. Conversely, bounds on the relative error of the success ratio lead to only very weak bounds on the failure ratio function if only a small number of points is necessarily misclassified and opt approaches P, hence the factor opt$/(P-\text{opt})$ is very large.

Assume conversely that the failure ratio is limited by $D > 1$. Hence $P-r < (P-\text{opt})D$. Then we can bound the relative error of the success ratio by $(\text{opt}-r)/\text{opt} < (P-\text{opt})(D-1)/\text{opt}$. The value $(P-\text{opt})/\text{opt}$ is small if opt is close to P and it is large if opt is much smaller than P. Hence we obtain small bounds on the relative error of the success ratio if the number of unavoidable misclassifications is small and only weak bounds if this number is large.

3 Hardness of Approximating the Success Ratio Function

We want to show that in several situations it is difficult to approximately minimize the relative error of m_L for a loading problem L. These results would extend and generalize the results of Bartlett and Ben-David [4] to more complex activation functions and several realistic situations.

3.1 A General Theorem

First, 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 2 *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_j, v_l) in the graph for which v_j and v_l are contained in the same cut are called monochromatic; all other edges are called bichromatic.*

Theorem 3 [20] *It is NP-hard to approximate the MAX-k-cut problem within relative error smaller than $1/(34(k-1))$ for any $k \geq 2$.*

The concept of an L-*reduction* was defined by Papadimitriou and Yannakakis [26]. The definition stated below is a slightly modified version of [26] (allowing an additional parameter a) that will be useful for our purposes.

Definition 4 *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, $\text{opt}(I_1)$ and $\text{opt}(I_2)$, respectively, satisfy $\text{opt}(I_2) \leq \alpha\ \text{opt}(I_1)$.*

(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 smaller than a, algorithm T_2 produces a solution I_1 of C_1 with cost c_1 satisfying $(\text{opt}(I_1) - c_1) \ \leq\ \beta\ (\text{opt}(I_2) - c_2)$.*

The following observation is easy.

Observation 5 *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 smaller than $a\alpha\beta$ is NP-hard, then approximation of C_2 with relative error smaller than a is also NP-hard.*

Since the reductions for various types of network architectures are very similar, we first state a general theorem following the approach in [4]. 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 $\boldsymbol{x}$ in the following discussions.

Consider an L-reduction from the MAX-k-cut problem to the loading problem L with success ratio function m_L satisfying 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 $\boldsymbol{x} \in \mathbb{R}^n$, $y \in \mathbb{R}$ 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 correctly 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 misclassified.

Theorem 6 *The reduction described above is an L-reduction with constants $\alpha = k/(k-1)$, $\beta = 2|P_0| + 3$, and parameter $a \leq (k-1)/(k^2\,(2|P_0|+3))$.*

Proof: Let opt(I_1) and opt(I_2) be the optimal values of the success ratio function of the instances I_1 and I_2. Remember that it is trivially true that[3] $1 \geq \text{opt}(I_1) \geq 1-(1/k)$. Hence, because of (i),

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

Hence, α can be chosen as $k/(k-1)$.

Next we show that β can be chosen as $2|P_0|+3$ provided $a \leq (k-1)/(k^2\,(2|P_0|+3))$. Assume that we are given an approximate solution of the instance I_2 of the loading problem with cost c_2.

- Assume that the relative error of the given solution is smaller than a. Then we can limit the cost of the solution by $c_2 \geq (1-a)\text{opt}(I_2) \geq (1-a)(2|P_0|+3-1/k)/(2|P_0|+3)$ due to the definition of the relative error.

- Hence 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,

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

  for $a < (2k-1)/(k(2|P_0|+3-1/k))$.

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

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

  c_1 denoting the cost of the solution of I_1. Hence, β can be chosen as $2|P_0|+3$.

□

Note that in particular every optimum solution must correctly classify P_0 if at least one solution with correct P_0 exists.

[3] Consider a very simple randomized algorithm in which a vertex is placed in any one of the k partitions with equal probability. Then, the expected value of the ratio $|\{(u,v) \in E \mid \psi(u) \neq \psi(v)\}| \ / \ |E|$ is $1-(1/k)$. Hence, $opt(I_1)$ is at least $1-(1/k)$.

Corollary 7 *Assume that a reduction as described above which fulfills properties (i) and (ii) can be found. Assume that approximation of the MAX-k-cut problem within relative error smaller than ϵ is NP-hard. Then approximation of the loading problem within relative error smaller than*

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

is NP-hard.

Proof: Obviously, since we can assume that $c_1 \geq 1-(1/k)$ and $\mathrm{opt}(I_1) \leq 1$, we can assume that $\epsilon \leq 1/k$. Hence the upper bound for a from Theorem 6 can be estimated via

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

Hence, using Theorem 6 and Observation 5, the result follows. □

3.2 Application to Neural Networks

This result can be applied to layered H-nets directly, $H(x)$ being the threshold activation function, as already shown in [4]. This type of architecture is common in theoretical study of neural nets as well as in applications. As defined in Section 1, the architecture is denoted by the tuple $(n, n_1, n_2, \ldots, n_h, 1)$. One can obtain the following hardness result, which can also be found in [4] (except for the case when $n_1 = 2$).

Theorem 8 *For any $h \geq 1$, constant $n_1 \geq 2$ and any $n_2, \ldots, n_h \in \mathbb{N}$, it is NP-hard to approximate the loading problem with instances (N, P), where N is the architecture of a layered $\{(n, n_1, \ldots, n_h, 1) \mid n \in \mathbb{N}\}$ H-net (n_1 is fixed, n_2,…,n_h may vary) and P is a set of examples from $\mathbb{Q}^n \times \{0,1\}$, with relative error smaller than*

$$\rho = \frac{1}{68n_1 2^{n_1} + 136n_1^3 + 136n_1^2 + 170n_1}\,.$$

Since the case of $n_1 = 2$ is not covered in [4] we provide a proof in the appendix.

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

The previous theorem deals with multilayer threshold networks which are common in theoretical studies. However, often a continuous and differentiable activation function, instead of the threshold activation function, is used in practical applications. One very common activation function is the sigmoidal activation $\mathrm{sgd}(x) = 1/(1+\mathrm{e}^{-x})$. Therefore it would be of interest to obtain a result for the sigmoidal activation function as well. In this section we deal with a feedforward architecture of the form $(n,2,1)$ where the input dimension n is allowed to vary from one instance to the next instance (this is the same architecture used in [9]). The activation function of the two hidden nodes is the sigmoidal activation function. Since the network is used for classification purposes, the output activation function is the following modification of the threshold activation function

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

This modification enforces 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 by an appropriate scaling of the output weights. Therefore we restrict to solutions with output weights bounded by some 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 of the outputs (for example, see [23]). Formally, the network computes the function

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

where $\boldsymbol{w} = (\alpha,\beta,\gamma,\boldsymbol{a},a_0,\boldsymbol{b},b_0)$ are the weights and thresholds, respectively, of the output node and the two hidden nodes 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 larger than ϵ for every pattern in the training set which is mapped correctly.

Theorem 9 *It is NP-hard to approximate the loading problem with relative error smaller than $1/2244$ for the architecture of a $\{(n,2,1) \mid n \in \mathbb{N}\}$-net with sigmoidal activation function for the two hidden nodes, activation function H_ϵ in the output node with $\epsilon < 0.5$ ($\epsilon \in \mathbb{Q}$), weight restriction $B \geq 2$ of the output weights ($B \in \mathbb{Q}$), and examples from $\mathbb{Q}^n \times \{0,1\}$.*

Proof: We use Theorem 6 and Corollary 7. Various steps in the proof are as follows:
(1) Definition of the training points:
T_1 maps an instance of the MAX-2-cut problem to the following loading problem. The input dimension n is given by $n = |V| + 5$. The points P_0 together with their labeling are

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

with $c > 1 + 8B/\epsilon \cdot \left(\operatorname{sgd}^{-1}(1-\epsilon/(2B)) - \operatorname{sgd}^{-1}(\epsilon/(2B))\right)$. Furthermore,

$\boldsymbol{e}_i$ $(i = 1,\ldots,|V|)$ is the i^{th} unit vector with labeling 0,
$\boldsymbol{e}_{ij}$ $((v_i,v_j) \in E)$ is the vector with 1 at positions i and j from left and 0 otherwise with labeling 1.

(2) Examination of the geometric form:
First we want to see how a classification looks like. The output neuron computes the activation

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

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. First we investigate the geometric form of the output of our network when the relative error is smaller than $1/2244$[4]. We will use properties of the geometric form

[4] Note that we will not use the geometric form for the construction of any algorithms but only for proofs.

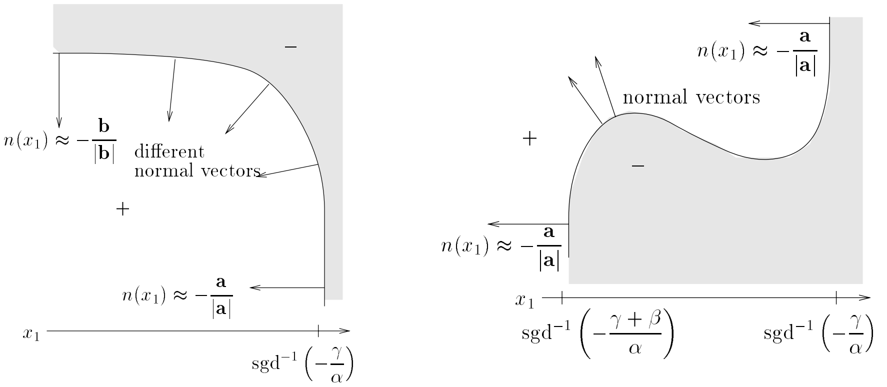
which are not affected by the concrete parameterization of the objects, in particular, we may scale, rotate, or translate the coordinates. For this purpose we examine the set of points M which form the classification boundary, i.e. the points x for which

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

holds. The set of points $\boldsymbol{x}$ for which the left-hand side of the equation $(*)$ above is positive (resp. negative) is called the positive (resp. negative) region. If α, β, $\boldsymbol{a}$, or $\boldsymbol{b}$ were 0, the output of the network would reduce to at most one hyperplane which separates the points. Obviously, P_0 would not be classified correctly in this case. Hence we will assume in the following investigation of the geometric form that these values do not vanish. Since we are *only interested in the geometric form*, we can substitute the current Euclidean coordinates by any coordinates which are obtained via a translation, rotation, or uniform scaling. Hence we can assume that $\boldsymbol{a}^t\boldsymbol{x} + a_0 = x_1$, where x_1 is the first component of $\boldsymbol{x}$.

Assume that $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent. Then the classification of an input $\boldsymbol{x}$ depends on the value of $\boldsymbol{a}^t\boldsymbol{x}$. Due to our parameterization, only the size of the first component of a point $\boldsymbol{x}$, the value x_1, determines whether $\boldsymbol{x}$ is contained in the positive region, the negative region, or the set M of points with output activation 0 of the network. Moreover, $\boldsymbol{b} = (b_1, 0, \ldots, 0)$. Depending on the weights in the network, the positive region is separated from negative region by up to three parallel hyperplanes of the form $(x, 0 \ldots, 0) + \boldsymbol{a}^\perp$ where x is a solution of the equality $\alpha \operatorname{sgd}(x) + \beta \operatorname{sgd}(b_1 x + b_0) + \gamma = 0$ and $\boldsymbol{a}^\perp$ denotes the vector space of vectors which are orthogonal to $\boldsymbol{a}$. In order to show this claim, we have to show that the above function $x \mapsto \alpha \operatorname{sgd}(x) + \beta \operatorname{sgd}(b_1 x + b_0) + \gamma$ yields zero for at most three points $x \in \mathbb{R}$. Remember that we assumed that α, β, and b_1 do not vanish. Assume for the sake of contradiction that the function equals 0 for four points x. These points fulfill the equality $\operatorname{sgd}(x) = (-\beta \operatorname{sgd}(b_1 x + b_0) - \gamma)/\alpha$. Since the mapping $x \mapsto \operatorname{sgd}(b_1 x + b_0)$ is monotonic, we can identify a connected open interval for x where $(-\beta \operatorname{sgd}(b_1 x + b_0) - \gamma)/\alpha$ is contained in $]0, 1[$ and hence the above equality can possibly be solved. Within this interval we can consider the function $\operatorname{sgd}^{-1}((-\beta \operatorname{sgd}(b_1 x + b_0) - \gamma)/\alpha)$ which then equals the identity x for four points. Hence the derivative of the function $\operatorname{sgd}^{-1}((-\beta \operatorname{sgd}(b_1 x + b_0) - \gamma)/\alpha)$ equals 1 for at least three different points within this interval. Using the equality $\operatorname{sgd}^{-1}(y) = -\ln(1/y - 1)$ for $y \in]0, 1[$ we can compute the derivative as $(\alpha\beta b_1 \operatorname{sgd}(b_1 x + b_0)(1 - \operatorname{sgd}(b_1 x + b_0)))/((\alpha + \gamma + \beta \operatorname{sgd}((b_1 x + b_0))(\beta \operatorname{sgd}(b_1 x + b_0)))$. If we set this term as 1, we obtain a quadratic equation for the term $\operatorname{sgd}(b_1 x + b_0)$ and hence at most two different values such that the derivative equals 1, hence at most 3 values x where the above function $\alpha \operatorname{sgd}(x) + \beta \operatorname{sgd}(b_1 x + b_0) + \gamma$ equals 0.

On the other hand, if $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent then M forms an $n - 1$ dimensional manifold because it is defined as zero set of a function $\alpha \operatorname{sgd}(\boldsymbol{a}^t\boldsymbol{x} + a_0) + \beta \operatorname{sgd}(\boldsymbol{b}^t\boldsymbol{x} + b_0) + \gamma$ where the Jacobian has full rank. For a definition of manifolds and related terms such as the tangential bundle, curves, and convexity see e.g. [11, 32]. The manifold M has a very specific form: The tangential space for a point $\boldsymbol{x} \in M$ consists of the directions which are orthogonal to the Jacobian at point $\boldsymbol{x}$ of the above function. Hence the vector space $\boldsymbol{a}^\perp \cap \boldsymbol{b}^\perp$ is always included in the tangential space where $\boldsymbol{a}^\perp$ as above denotes the vectors which are perpendicular to $\boldsymbol{a}$ and $\boldsymbol{b}^\perp$ denotes the vectors perpendicular to $\boldsymbol{b}$. If a point $\boldsymbol{x}$ is contained in M then every point $\boldsymbol{x} + \boldsymbol{v}$ for $\boldsymbol{v} \in \boldsymbol{a}^\perp \cap \boldsymbol{b}^\perp$ is contained in M, too. Note that every vector $\boldsymbol{x} \in \mathbb{R}^n$ can be uniquely decomposed into $\boldsymbol{x}_{ab} + \boldsymbol{x}_{a^\perp b^\perp}$ where $\boldsymbol{x}_{ab}$ denotes a vector in the plane spanned by $\boldsymbol{a}$ and $\boldsymbol{b}$, and $\boldsymbol{x}_{a^\perp b^\perp}$ denotes a vector in the orthogonal space



$-$
$n(x_1) \approx -\frac{\mathbf{b}}{|\mathbf{b}|}$
different
normal vectors
$+$
$n(x_1) \approx -\frac{\mathbf{a}}{|\mathbf{a}|}$
x_1
$\mathrm{sgd}^{-1}\left(-\frac{\gamma}{\alpha}\right)$
$n(x_1) \approx -\frac{\mathbf{a}}{|\mathbf{a}|}$
normal vectors
$+$
$-$
$n(x_1) \approx -\frac{\mathbf{a}}{|\mathbf{a}|}$
x_1
$\mathrm{sgd}^{-1}\left(-\frac{\gamma+\beta}{\alpha}\right)$
$\mathrm{sgd}^{-1}\left(-\frac{\gamma}{\alpha}\right)$

Figure 1: Left: classification in the second case; right: classification in the third case.

$\boldsymbol{a}^\perp \cap \boldsymbol{b}^\perp$. Note that only the first part, $\boldsymbol{x}_{ab}$ determines whether $\boldsymbol{x}$ is contained in M, the positive region, or the negative region. Hence we can entirely describe the classification given by the network if we only consider the projection of the points to the plane spanned by $\boldsymbol{a}$ and $\boldsymbol{b}$. Hence we can in the following restrict our investigation to the one-dimensional manifold which is obtained if we project M onto the plane spanned by $\boldsymbol{a}$ and $\boldsymbol{b}$. This one-dimensional manifold entirely determines the classification of points by the network. Next we show that this projection is a simple curve and we derive an explicit parameterization for the curve and a normal vector field to the curve. Assume $\boldsymbol{x}$ is an element of the one-dimensional manifold. As above we assume a parameterization of the manifold such that $x_1 = \boldsymbol{a}^t\boldsymbol{x} + a_0$. If x_1 is chosen then we can uniquely determine $\boldsymbol{b}^t\boldsymbol{x} = \mathrm{sgd}^{-1}((-\alpha\,\mathrm{sgd}(x_1) - \gamma)/\beta)$ because of $(*)$ where this value is defined, i.e. for $\mathrm{sgd}(x_1) \in](-\gamma-\beta)/\alpha, -\gamma/\alpha[$ if $(-\gamma-\beta)/\alpha < -\gamma/\alpha$, or $\mathrm{sgd}(x_1) \in]-\gamma/\alpha, (-\gamma-\beta)/\alpha[$ if $-\gamma/\alpha < (-\gamma-\beta)/\alpha$, respectively. Hence we find the parameterization $(x_1, \mathrm{sgd}^{-1}((-\alpha\,\mathrm{sgd}(x_1) - \gamma)/\beta)$ of $(\boldsymbol{a}^t\boldsymbol{x}, \boldsymbol{b}^t\boldsymbol{x})$ for $x_1 \in]l,h[$ where $l = \min\{\mathrm{sgd}^{-1}((-\gamma-\beta)/\alpha), \mathrm{sgd}^{-1}(-\gamma/\alpha)\}$ and $r = \max\{\mathrm{sgd}^{-1}((-\gamma-\beta)/\alpha), \mathrm{sgd}^{-1}(-\gamma/\alpha)\}$ where we set $\mathrm{sgd}^{-1}(t) = -\infty$ if $t \leq 0$ and $\mathrm{sgd}^{-1}(t) = \infty$ if $t \geq 1$. Note that the components of this mapping are both monotonic functions. From $(\boldsymbol{a}^t\boldsymbol{x}, \boldsymbol{b}^t\boldsymbol{x})$ we obtain the parameterization $x_1 \mapsto (\boldsymbol{x}^t\boldsymbol{a}|\boldsymbol{b}|^2 - \boldsymbol{x}^t\boldsymbol{b}\boldsymbol{a}^t\boldsymbol{b})/(|\boldsymbol{a}|^2|\boldsymbol{b}|^2 - (\boldsymbol{a}^t\boldsymbol{b})^2) \cdot \boldsymbol{a} + (\boldsymbol{x}^t\boldsymbol{b}|\boldsymbol{a}|^2 - \boldsymbol{x}^t\boldsymbol{a}\boldsymbol{a}^t\boldsymbol{b})/(|\boldsymbol{a}|^2|\boldsymbol{b}|^2 - (\boldsymbol{a}^t\boldsymbol{b})^2) \cdot \boldsymbol{b}$ of the curve. We refer to this curve as the curve which describes M. In particular, this constitutes a simple connected curve parameterized by x_1 because the mapping is continuous and the function $x_1 \mapsto \boldsymbol{a}^t\boldsymbol{x} = x_1$ is obviously injective. A normal vector along the curve can be parameterized by $n(x_1) = \alpha\,\mathrm{sgd}'(x_1)\cdot\boldsymbol{a} + \beta\,\mathrm{sgd}'(\boldsymbol{b}^t\boldsymbol{x} + b_0)\cdot\boldsymbol{b}$, the Jacobian. The term $\mathrm{sgd}'(\boldsymbol{b}^t\boldsymbol{x} + b_0)$ can again be substituted using the equality $(*)$. We obtain

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

Define by $\tilde{n}(x_1) = n(x_1)/|n(x_1)|$ the normalization of n.

Now considering in more detail the four values γ, $\gamma+\alpha$, $\gamma+\beta$, and $\gamma+\alpha+\beta$ several cases can be distinguished for the curve which describes M if $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent, at most three parallel hyperplanes separate both regions.

Case 1: All values are ≥ 0 or all values are ≤ 0. Then the activation of the network is positive for every input or negative for every input. In particular, there exists at least one point in P_0 which is not classified correctly.

Case 2: One value is > 0, the others are < 0.

We can assume that $\gamma > 0$ since $\operatorname{sgd}(-x) = 1 - \operatorname{sgd}(x)$. We may have to change the sign of the weights and the thresholds beforehand without affecting the activation due to the symmetries of the sigmoidal activation function: if $\alpha+\gamma$ is positive, we substitute α by $-\alpha$, γ by $\alpha+\gamma$, $\boldsymbol{a}$ by $-\boldsymbol{a}$, and a_0 by $-a_0$; 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.

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent, at most three parallel hyperplanes with normal vector $\boldsymbol{a}$ separate the positive and negative region. The number of hyperplanes is determined by the number of points for which $\alpha \operatorname{sgd}(x) + \beta \operatorname{sgd}(b_1 x + b_0) + \gamma$ yields 0. If $b_1 > 0$ then the above function is strictly monotonically decreasing with $x_1 \to \infty$, hence at most one point 0 can be observed and the positive region is separated from the negative region by one hyperplane. If $b_1 < 0$, we find $\lim_{x\to\infty} \alpha \operatorname{sgd}(x) + \beta \operatorname{sgd}(b_1 x + b_0) + \gamma = \alpha + \gamma < 0$ and $\lim_{x\to-\infty} \alpha \operatorname{sgd}(x) + \beta \operatorname{sgd}(b_1 x + b_0) + \gamma = \beta + \gamma < 0$ hence the function can have at most two points with value 0. The positive region is empty or convex and separated from the negative region by two parallel hyperplanes with normal vector $\boldsymbol{a}$.

Assume that $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent. We find $\alpha < -\gamma$, and $\beta < -\gamma$. Dividing $(*)$ by γ we obtain $\gamma = 1$, $\alpha < -1$, and $\beta < -1$. The curve describing M looks like depicted in Fig. 1, in particular, the positive region is convex, as can be shown as follows: The normal vector $\tilde{n}$ decomposes into a combination $\lambda_1(x_1) \cdot \boldsymbol{a} + \lambda_2(x_1) \cdot \boldsymbol{b}$ where $\lambda_1(x_1) = \alpha \operatorname{sgd}'(x_1)/|\tilde{n}(x_1)|$ and $\lambda_2(x_1) = \beta \operatorname{sgd}'(\boldsymbol{b}^t\boldsymbol{x}+b_0)/|\tilde{n}(x_1)| = (-1-\alpha \operatorname{sgd}(x_1))(1-(-1-\alpha \operatorname{sgd}(x_1))/\beta)/|\tilde{n}(x_1)|$ and $\alpha \in]l, h[$ as above. λ_1 and λ_2 are in Case 2 both negative because α and β are both negative. We now show the convexity. First we show that the curve describing M is convex. Assume for contradiction that it was not convex. Then there would exist at least two points on the curve with identical normal vector $\tilde{n}$, identical coefficients λ_1 and λ_2 because $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent, i.e. identical λ_1/λ_2. Consequently, there would exist at least one point x_1 with $(\lambda_1/\lambda_2)'(x_1) = 0$. Note that λ_1/λ_2 is obviously differentiable, though $\tilde{n}$ is not, since the ratio $|n(x_1)|/|n(x_1)|$ is a constant 1. One can compute $(\lambda_1/\lambda_2)'(x_1) = C(x_1) \cdot (-\beta - 1 + \operatorname{sgd}(x_1)(2\beta+2) + \operatorname{sgd}^2(x_1)(2\alpha + \alpha^2 + \alpha\beta))$ where $C(x_1) = \alpha\beta \operatorname{sgd}'(x_1)/((-1-\alpha \operatorname{sgd}(x_1))^2(1+\beta+\alpha \operatorname{sgd}(x_1))^2) \neq 0$. If $(\lambda_1/\lambda_2)'(x_1)$ was 0, $\alpha = \beta = -1$ or

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

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. Hence the curve is convex unless α or β equals -1 which cannot take place by assumption. Hence the positive region or the negative region is convex. We show that the negative region is not convex, hence the positive region is necessarily convex: since $\boldsymbol{b}$ and $\boldsymbol{a}$ are linearly independent, we can assume w.l.o.g. that the second component b_2 of $\boldsymbol{b}$ does not vanish. (For other nonvanishing components the argumentation is analogous.) It holds $\alpha < -1$, $\beta < -1$, and $\gamma = 1$. Choose $L > 0$ such that $\alpha \operatorname{sgd}(L) + \beta \operatorname{sgd}(L) + 1 > 0$. Consider the line $\{(L, (L(1-b_1) - b_0)/b_2, 0, \ldots, 0) + T \cdot (1, (-1-b_1)/b_2, 0, \ldots, 0) \,|\, T \in \mathbb{R}\}$. Thereby only the first two coefficients are nonvanishing. We find for the point where $T = 0$ a positive activation of the network by assumption on L, whereas $T \to \infty$ yields the activation of the network $\alpha + 1 < 0$ and $T \to -\infty$ yields $\beta + 1 < 0$.

We finally compute the limits of $\tilde{n}$ if x_1 approaches the borders of the interval $]l, h[$ as defined above. Because $-1/\alpha > 0$, $(-1 - \beta)/\alpha > 1$ we find $]l, h[=]\mathrm{sgd}^{-1}(-1/\alpha), \infty[$. We can compute using the above parameterization of $\tilde{n}$: $\lim_{x_1 \to \mathrm{sgd}^{-1}(-1/\alpha)} \tilde{n}(x_1) = -\boldsymbol{a}/|\boldsymbol{a}|$ and $\lim_{x_1 \to \infty} \tilde{n}(x_1) = -\boldsymbol{b}/|\boldsymbol{b}|$. Hence in this case the positive region is convex with normal vector approaching a parallel vector to $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively. For every point on the curve the normal vector is parallel to a convex combination of $-\boldsymbol{a}$ and $-\boldsymbol{b}$. Moreover, the coefficients λ_1 and λ_2 defined in $\tilde{n}$ are strictly monotonic functions. Note that a convex manifold coincides with the intersection of its tangential hyperplanes.

Case 3: Two values are ≥ 0, two values are ≤ 0.

We have already seen that if $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent, at most three parallel hyperplanes separate the positive and negative regions.

Assume $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent. We can assume that the nonnegative values are γ and $\beta + \gamma$. We may have to change the role of the two hidden nodes or the signs of the weights and the thresholds beforehand: if $\alpha + \gamma$ and $\beta + \gamma$ are nonnegative 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 nonnegative 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 nonnegative 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 nonnegative in this situation unless all four values equal 0 which is Case 1. Moreover, we can assume that at least one value is nonzero, otherwise Case 1 would take place.

We can as before consider a normal vector of the curve describing M, $\tilde{n}$. The limits $]l, h[$ of the parameter x_1 yield for $\beta > 0$ the value $l = \mathrm{sgd}^{-1}(-\gamma/\alpha)$ (or $-\infty$ if $\gamma = 0$), and $h = \mathrm{sgd}^{-1}((-\gamma - \beta)/\alpha)$ (or ∞ if $\alpha + \beta + \gamma = 0$). For $\beta < 0$ the limits are $l = \mathrm{sgd}^{-1}((-\gamma - \beta)/\alpha)$ (or $-\infty$ if $\beta + \gamma = 0$), and $h = \mathrm{sgd}^{-1}(-\gamma/\alpha)$ (or ∞ if $\alpha + \gamma = 0$). One can compute for all limits with finite bounds l and h the normal vector $\lim_{x_1 \to l} \tilde{n}(x_1) = \lim_{x_1 \to h} \tilde{n}(x_1) = -\boldsymbol{a}/|\boldsymbol{a}|$. For the limits where l or h equals $\pm\infty$ we can consider the fraction $\lambda_1(x_1)/\lambda_2(x_1)$ of the coefficients of the linear combination of $\tilde{n}$ and obtain $\lim_{x_1 \to -\infty} \tilde{n}(x_1) = (-\boldsymbol{a} + \boldsymbol{b})/|-\boldsymbol{a} + \boldsymbol{b}|$ for $\gamma = 0$, the same vector is obtained for $\alpha + \beta + \gamma = 0$ and $\lim_{x_1 \to \infty}$. We obtain $\lim_{x_1 \to -\infty} \tilde{n}(x_1) = (-\boldsymbol{a} - \boldsymbol{b})/|\boldsymbol{a} + \boldsymbol{b}|$ if $\beta + \gamma = 0$ and the same vector is obtained for $\lim_{x_1 \to \infty}$ and $\alpha + \gamma = 0$. For values in between we can as before consider the fraction λ_1/λ_2 of the parameterization of a normal vector $\tilde{n}$ and its derivative to estimate the overall geometric form. We obtain a formula analogous to $(**)$ for possible points zero for the derivative of this function: $-\gamma\beta - \gamma^2 + \mathrm{sgd}(x_1)(2\gamma\beta + 2\gamma^2) + \mathrm{sgd}^2(x_1)(\alpha\beta + 2\alpha\gamma + \alpha^2) = 0$. This is constant 0 if $\alpha + \beta + 2\gamma = 0$ and $\gamma = 0$ or $\beta + \gamma = 0$ in which cases the normal vector equals $(-\boldsymbol{a} - \boldsymbol{b})/|\boldsymbol{a} + \boldsymbol{b}|$ or $(-\boldsymbol{a} + \boldsymbol{b})/|-\boldsymbol{a} + \boldsymbol{b}|$, respectively. Otherwise, we obtain at most one possible solution if $\alpha + \beta + 2\gamma = 0$. If $\alpha + \beta + 2\gamma \neq 0$ we obtain

$$\mathrm{sgd}(x) = \frac{-\gamma(\beta + \gamma)}{\alpha(\alpha + \beta + \gamma)} \pm \sqrt{\frac{\gamma(\beta + \gamma)(\alpha + \gamma)(\alpha + \beta + \gamma)}{\alpha^2(\alpha + \beta + \gamma)^2}}$$

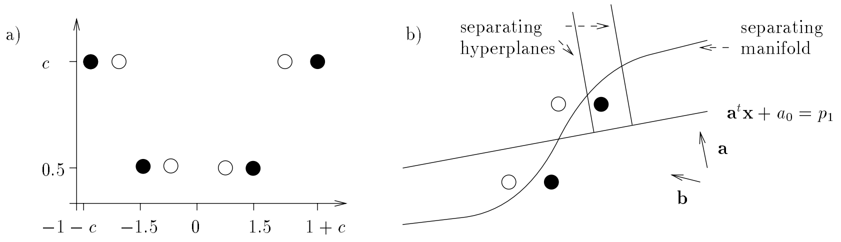
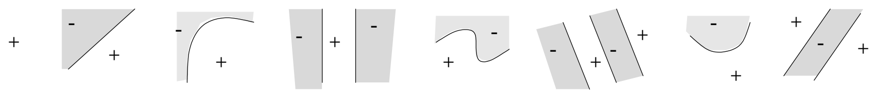


Figure 2: a) Classification problem in the last two dimensions; b) outside the relevant region.

which has at most one solution because $\gamma(\beta+\gamma)$ is positive by assumption hence only the solution with '$+$' might yield positive values. Hence the separating curve equals either a simple line, or it has an S-shaped form with limiting normal vectors $-\boldsymbol{a}/|\boldsymbol{a}|$ or, in the above special cases, $(-\boldsymbol{a}-\boldsymbol{b})/|\boldsymbol{a}+\boldsymbol{b}|$ or $(-\boldsymbol{a}+\boldsymbol{b})/|-\boldsymbol{a}+\boldsymbol{b}|$, respectively (see Fig. 1). We will use the following property in the latter case: the term $\boldsymbol{b}^t\boldsymbol{x} + b_0$ cannot be limited from above or below, respectively, for points $\boldsymbol{x}$ in M if x_1 approaches the borders l or h, respectively. I.e. for $x_1 \to l$, the term $\boldsymbol{b}^t\boldsymbol{x} + b_0$ becomes arbitrary large or arbitrary small, respectively, and an analogous result holds for $x_1 \to h$. Thereby, this fact follows from the above limits because $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent. Hence a unit tangential vector of the curve which is perpendicular to the normal vector can be decomposed into $\mu_1\boldsymbol{b} + \mu_2\boldsymbol{b}^\perp$ where $\boldsymbol{b}^\perp$ is some vector perpendicular to $\boldsymbol{b}$, μ_1 and $\mu_2 \in \mathbb{R}$, and the coefficient μ_1 approaches a fixed and nonvanishing limit if x_1 approaches l or h, respectively, due to the limits of $\tilde{n}$.

Case 4: 3 **values are** > 0**, one value is** < 0**.** This case is dual to Case 2. We obtain the possible geometric forms by changing the output sign, i.e. the positive and negative region. Hence the negative region is convex and separated from the positive region by up to two lines or a convex curve. The normal vector of the curve can be written as a linear combination of $\boldsymbol{a}$ and $\boldsymbol{b}$: $\tilde{n} = \lambda_1(x_1)\boldsymbol{a} + \lambda_2(x_1)\boldsymbol{b}$ where λ_1 and λ_2 are strictly monotonic. If x_1 approaches l or h, respectively, $\tilde{n}$ becomes parallel to $\boldsymbol{a}$ or $\boldsymbol{b}$, respectively.

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



Trivial cases Case 2 Case 3 Case 4

(3) Only Case 2 **solves** P_0**:**

Next we 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 scenario occurs (we drop the first $|V| + 3$ coefficients which are 0 for clarity): 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. 2a). They cannot be separated by at most three parallel lines. Assume for contradiction that at most three parallel lines separated the points. Then one line had to separate at least two pairs of points $(-0.5, 0.5)$, $(-1.5, 0.5)$ or $(0.5, 0.5)$, $(1.5, 0.5)$

or (c, c), $(1 + c, c)$ or $(-c, c)$, $(-1 - c, c)$. Since the points with second component 0.5 are contained in a single line, we can assume w.l.o.g. that the line separates the second and third pair, the argumentation for the other situations is equivalent. Hence we can limit the tangent vector of the line to be contained in the sector $(c - 1.5, c - 0.5)$ and $(c + 0.5, c - 0.5)$. Hence each of the remaining at most two lines which are parallel can only separate one of the pairs $(-0.5, 0.5)$, $(-1.5, 0.5)$ or $(-c, c)$, $(-1 - c, c)$ or $(-1.5, 0.5)$, $(-c, c)$, contradiction.

Hence, $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent which means that the separating manifold has an S-shaped form. Note that we will only use properties on the size of $\boldsymbol{b}^t\boldsymbol{x} + b_0$ for points $\boldsymbol{x}$ on the curve where $x_1 \to l$ or $x_1 \to h$, respectively, in order to derive a contradiction. Define $p_0 :=$ sgd$^{-1}(\epsilon/(2B))$ and $p_1 :=$ sgd$^{-1}(1 - \epsilon/(2B))$. The set of points $\{\boldsymbol{x} \mid p_0 \leq \boldsymbol{a}^t\boldsymbol{x} + a_0 \leq p_1\}$ 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, sgd$(\boldsymbol{a}^t\boldsymbol{x} + a_0)$ or sgd$(\boldsymbol{b}^t\boldsymbol{x} + b_0)$, respectively, can be substituted by a constant, the difference of the output activation being at most $\epsilon/2$. More precisely, every constant sgd$(\boldsymbol{a}^t\boldsymbol{x} + a_0)$ or sgd$(\boldsymbol{b}^t\boldsymbol{x} + b_0)$, respectively, for a fixed $\boldsymbol{x} \in M$ with $\boldsymbol{a}^t\boldsymbol{x} + a_0 < p_0$ or $> p_1$, respectively, or $\boldsymbol{b}^t\boldsymbol{x} + b_0 < p_0$ or $> p_1$, respectively, will do. Note that due to the monotonicity of both components of the map $x_1 \mapsto (\boldsymbol{a}^t\boldsymbol{x} + a_0, \boldsymbol{b}^t\boldsymbol{x} + b_0)$ for $x_1 \in]l, h[$ the curve will stay outside the $\boldsymbol{a}$- respectively $\boldsymbol{b}$-relevant region if the curve has crossed the boundary towards the respective region.

Now it is first shown that three points forming an isosceles triangle with height at least $c - 1$ and base length at least 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 (c, c) and $(1 + c, c)$ or the two points $(-c, c)$ and $(-1 - c, c)$, respectively, 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. Assume the points (c, c) and $(1 + c, c)$ were both outside the $\boldsymbol{a}$-relevant region. Then they were contained either on different sides i.e. $p_0 > \boldsymbol{a}^t\boldsymbol{x} + a_0$ for one of the points and $\boldsymbol{a}^t\boldsymbol{x} + a_0 > p_1$ for the other point or they were both contained at the same side, e.g. $p_0 > \boldsymbol{a}^t\boldsymbol{x} + a_0$ for both points. (The case where $\boldsymbol{a}^t\boldsymbol{x} + a_0 > p_1$ holds for both points is analogous.) We first consider the latter case (see Fig. 2b): as already mentioned, we could then substitute the respective parts sgd$(\boldsymbol{a}^t\boldsymbol{x} + a_0)$ by the value obtained for any further point $\boldsymbol{x}$ of the curve with $p_0 > \boldsymbol{a}^t\boldsymbol{x} + a_0$, the difference being at most $\epsilon/2$. The term $\boldsymbol{b}^t\boldsymbol{x} + b_0$ is unlimited either from above or from below if x_1 approaches the respective limit of the parameterization l or h, as shown beforehand. Hence we can find a point $\boldsymbol{x}_p$ of the curve with $p_0 > \boldsymbol{a}^t\boldsymbol{x}_p + a_0$ such that the corresponding value $\boldsymbol{b}^t\boldsymbol{x}_p + b_0$ is larger than $\boldsymbol{b}^t\boldsymbol{x} + b_0$ for both, $\boldsymbol{x} = (c, c)$ and $\boldsymbol{x} = (1 + c, c)$, or it is smaller than the value $\boldsymbol{b}^t\boldsymbol{x} + b_0$ for both, $\boldsymbol{x} = (c, c)$ and $\boldsymbol{x} = (1 + c, c)$. Because $\boldsymbol{x}_p$ yields the activation 0 and the first part α sgd$(\boldsymbol{a}^t\boldsymbol{x} + a_0)$ differs at most $\epsilon/2$ for $\boldsymbol{x}_p$, (c, c), and $(1 + c, c)$, the points (c, c) and $(1 + c, c)$ cannot be classified differently with an activation of absolute value larger than ϵ. Contradiction. If the two points (c, c) and $(1 + c, c)$ were contained in different sides outside the $\boldsymbol{a}$-relevant region then the points $(-c, c)$ and $(-1 - c, c)$ were both not contained in the $\boldsymbol{a}$-relevant region and they were both contained in the same side. Hence we would obtain an analogous contradiction for these latter two points.

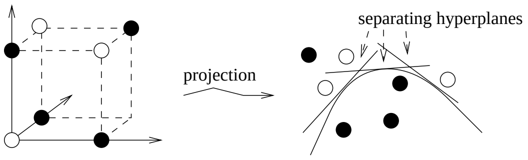


Figure 3: Classification problem; projection of the classification to the plane which is described by the weight vectors of the two hidden neurons, at least one negative point is not classified correctly

The same argument shows that one of $(-c, c)$ and $(-1-c, c)$ is contained inside the $\boldsymbol{a}$-relevant region. Therefore the $\boldsymbol{a}$-relevant region contains an isosceles triangle with heights at least $c-1$ and hence a circle with diameter at least $(c-1)/4$. Consequently, $a \leq 4(p_1 - p_0)/(c-1) < \epsilon/(2B)$, 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, we could conclude in the same way $b \leq \epsilon/(2B)$ for $b = |(b_{n+4}, b_{n+5})|$. This leads to the following contradiction: we find for the points $\boldsymbol{x}_1 = (0, \dots, 0, c, c)$ and $\boldsymbol{x}_2 = (0, \dots, 0, 1+c, c)$

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

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 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. 3. At most two parallel planes cannot separate a convex negative region. To show this claim, assume for contradiction that at most two parallel planes classified the points correctly such that the negative region is convex. The points are obviously not linearly separable. Hence one plane has to separate the point $(0,0,0)$ (we drop all but dimensions $|V|+1$ to $|V|+3$ for clarity) and all points $(1,0,0)$, $(0,1,0)$, $(0,0,1)$, and $(1,1,1)$, one plane separates $(1,1,0)$ from the four negative points, one plane separates $(0,1,1)$. Assume a normal vector of the respective plane is given by the coefficients (n_1, n_2, n_3). For the plane separating $(0,0,0)$ we find $n_1 < 0$ (because of $(1,0,0)$), $n_2 < 0$ (because of $(0,1,0)$), $n_3 < 0$ (because of $(0,0,1)$). For the plane separating $(1,1,0)$ we find $n_3 < 0$, $n_1 > 0$, $n_2 > 0$; for the plane separating $(0,1,1)$ we find $n_1 < 0, n_2 > 0, n_3 > 0$. Hence we would need three different and not parallel planes. Contradiction.

Consequently, $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent. The negative points are contained in a convex region. Because the negative region can then be obtained as the intersection of all tangential hyperplanes of the separating manifold, each positive point is separated by at least one tangential hyperplane of the separating manifold M from all negative points. Consider the projection to the plane spanned by $\boldsymbol{a}$ and $\boldsymbol{b}$ which determines the classification of the points. Following the convex curve which describes M, i.e. for increasing parameter x_1 of the parameterization, the signs of the coefficients of a normal vector can change at most once because $\tilde{n}(x_1) = \lambda_1(x_1)\boldsymbol{a} + \lambda_2(x_1)\boldsymbol{b}$ with strictly monotonic λ_1 and λ_2, λ_1 is negative and increasing, λ_2 is negative and decreasing. The limits of $\tilde{n}$ for $x_1 \to l$ or $x_1 \to h$, respectively, are $-\boldsymbol{a}/|\boldsymbol{a}|$ and $-\boldsymbol{b}/|\boldsymbol{b}|$, respectively. In particular, each component of $\tilde{n}$ can change its sign at most once. But a normal vector of a hyperplane which separates one of the positive points necessarily has the signs $(+,+,-)$ for $(1,1,0)$, $(-,+,+)$ for $(0,1,1)$, and $(-,-,-)$ for $(0,0,0)$ in the dimensions $|V|+1$ to $|V|+3$; these signs can be computed as the necessary signs for a normal vector of a plane which separates the respective positive point from all negative points as in the linearly dependent case. Independent of the order in which the points are visited if x_1 increases, at least one component of $\tilde{n}$ changes the sign twice, a contradiction. Hence any classification which maps P_0 correctly is of the type as in Case 2.

(4) Property (i):

Next we show that the correspondence demanded in property **(i)** of Theorem 6 holds. An optimum solution of the MAX-2-cut problem leads to the following approximate solution of the loading problem with weights $\alpha = \beta = -1$, $\gamma = 0.5$, $\boldsymbol{a} = K \cdot (a_1, \dots, a_n, 1, -1, 1, 1, -1)$, $\boldsymbol{b} = K \cdot (b_1, \dots, b_n, -1, 1, -1, -1, -1)$, $a_0 = -0.5 \cdot K$, $b_0 = -0.5 \cdot K$, where K is a positive constant and

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

and

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

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 which is independent of the particular instance.

Conversely, assume that an optimum solution of the loading problem is given. This classifies P_0 correctly. Consequently, we are in Case 2. We can easily compute an equivalent solution with $\gamma > 0$ performing the weight changes as described in Case 2. 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 dimensions 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. 4). (The above are dimensions i and j, the other are 0 and hence dropped for clarity.) If $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent then the positive region is separated from the negative region by at most two parallel lines (in dimensions i and j) with normal vector orthogonal to $\boldsymbol{a}$, the positive region is convex. A normal vector of the two lines which bounds the positive region in dimension i and j , respectively, has obviously two different signs: $(-,+)$ for $(1,0)$ and $(+,-)$ for $(0,1)$. Consequently, a_i and a_j have different signs. In the linearly independent 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 hyperplanes of the separating manifold such that these hyperplanes each separate one of the negative points. In the dimensions i and j, which are the only relevant dimensions for this case because all other dimensions equal 0 for the considered points, we then find tangential lines which separate the respective negative point from $(0,0)$ and $(1,1)$, respectively. The i^{th} or j^{th} component, respectively, of a normal vector of such a separating line is necessarily 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

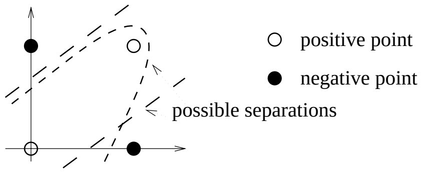


Figure 4: Classification corresponding to a bichromatic edge

once if we follow the convex curve for increasing x_1. The limit for $x_1 \to l$ equals $-\boldsymbol{a}/|\boldsymbol{a}|$. Consequently, the signs of a_i and a_j have to be different.

(5) Property (ii):

Finally we establish property **(ii)** of Theorem 6: given a set of weights forming a solution for the loading problem corresponding to an architecture 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 without loss of generality, we can assume again that $\gamma > 0$ (note that the weight changes in Case 2 can be computed in polynomial time). As before, we can define a solution of the instance of the MAX-2-cut problem via the sign of a_i, i.e., the nodes v_i with positive a_i are in the first cut (again, note, that this can be computed in polynomial time). If (v_i, v_j) is monochromatic and $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are correctly classified then $\boldsymbol{e}_{ij}$ is not classified correctly which is shown using the same argument as before. □

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

Definition 10 *Two functions $f, g : \mathbb{R} \to \mathbb{R}$ are ϵ-approximates of each other if $|f(x) - g(x)| \leq \epsilon$ for all $\boldsymbol{x} \in \mathbb{R}$.*

Corollary 11 *It is NP-hard to approximate the loading problem with relative error smaller than $1/2244$ for the architecture of a $\{(n, 2, 1) \mid n \in \mathbb{N}\}$-net with activation function σ in the hidden layer, activation function H_ϵ in the output with $\epsilon < 1/3$ ($\epsilon \in \mathbb{Q}$), weight restriction $B \geq 2$ of the output weights ($B \in \mathbb{Q}$), and examples from $\mathbb{Q}^n \times \{0,1\}$,* provided that *$\sigma$ is $\epsilon/(4B)$-approximate of* sgd*.*

Proof: The proof goes via L-reduction from the MAX-2-cut problem and Theorem 6. Since it is almost identical to the proof of Theorem 9, except that sigmoidal networks are substituted by σ-networks which is possible because σ is $\epsilon/(4B)$-approximate of sgd, we only sketch the identical parts and show the modifications due to the new activation function σ.

Assume that we are given an instance of the MAX-2-cut problem. One can reduce this instance of the MAX-2-cut problem to an instance of the 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 of value less than 0.5 such that only the points corresponding to monochromatic edges are misclassified. We substitute the function sgd by σ in this network. Since the weights of the output neuron are at most B, the output activation is 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 output activation is 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 9. 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 6 holds.

Conversely, given a solution of the loading problem for the activation function σ with accuracy ϵ we 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 6 follows. Since the factors concerning the L-reduction are the same as in Theorem 9, we obtain the same approximation bound. □

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

In this section, we consider the approximability of the loading problem with the semilinear activation function commonly used in the neural net literature [6, 10, 13, 21]. This activation function is defined as:

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

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

Theorem 12 *It is NP-hard to approximate the loading problem with relative error smaller than* $1/2380$ *for* $\{(n, 2, 1) \mid n \in \mathbb{N}\}$*-architectures with the semilinear activation function in the hidden layer, the threshold activation function in the output, and examples from* $\mathbb{Q}^n \times \{0, 1\}$.

Hence we have generalized the approximation results to more realistic activation functions. The proof, which is similar to Theorem 9, can be found in the appendix.

3.3 Avoiding Multiplicities

In the reductions of previous sections examples with multiplicities were contained in the training sets. One may ask the question as to whether this is avoidable since the training set in some learning situations may not contain the same pattern many times.

Consider the following modification of the reduction of the MAX-k-cut problem to a loading problem: T_1 yields the following *mutually different* points:

- points p_i^l, $l = 1, \ldots, 3|E|$ forming the set P_0,
- for each node v_i, points $\boldsymbol{e}_i^l$, $l = 1, \ldots, 2d_i$, 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 satisfy the following properties:

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

(ii') For any approximate solution of the instance of the loading problem which classifies, for each i, at least one point p_i^l in P_0 correctly, we can use the algorithm T_2 to compute, in polynomial time, an approximate solution of the instance of the MAX-k-cut problem with the following property: for any monochromatic edge (v_i, v_j) in this solution, either $\boldsymbol{e}_{ij}$ or $\boldsymbol{o}_{ij}$ or $\boldsymbol{e}_i^l$ for all l or $\boldsymbol{e}_j^l$ for all l are misclassified.

Theorem 13 *Under the assumptions stated above the reduction is an L-reduction with constants $\alpha = k/(k-1)$, $\beta = 3p_0 + 6$, and $a = (k-1)/(k^2(3p_0+6))$, where $p_0 = |P_0|/(3|E|)$,*

Corollary 14 *The reductions in Theorems 8, 9, and 12 can be modified such that both* **(i')** *and* **(ii')** *hold. Hence minimizing the relative error within some (smaller compared to those in Theorems 8, 9, and 12) constant is NP-hard even for training sets where no example is repeated more than once.*

The proofs of Theorem 13 and Corollary 14 can be found in the appendix.

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 strategy would be to choose the architecture such that valid generalization can be expected using the well known bounds in the agnostic or PAC setting [33].

Naturally the question arises about what happens to the complexity of training in these situations. 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 [31]:

Observation 15 *If the number of neurons in the first hidden layer is at least equal to the number of training examples and the activation function is the threshold function, the standard sigmoidal function, or the semilinear function (or any activation function σ such that the class of σ-networks possesses the universal approximation capability as defined in [31]) 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 even if the number of examples and hidden nodes are correlated.

Theorem 16 *Approximation of the success ratio function m_L with relative error smaller than c/k^3 (c is a constant, k is the number of hidden nodes) is NP-hard for the loading problem with instances $(\mathcal{A}, P)$, where $\mathcal{A}$ is a $(n, k, 1)$-H-architecture (n and k may vary) and $P \subset \mathbb{Q}^n \times \{0, 1\}$ is an example set with $k^{3.5} \leq |P| \leq k^4$.*

Proof: The proof is via a modification of an L-reduction from the MAX-3-cut problem. Assume that, in Definition 4 of an L-reduction, the algorithm T_1, given an instance I_1, produces in polynomial time an instance I_2 of C_2 and a parameter $\beta(I_1) > 0$, which *may depend on the instance I_1*, such that the maxima opt(I_1) and opt(I_2), respectively, satisfy opt$(I_2) \leq \alpha$ opt(I_1), and the algorithm T_2 maps in polynomial time a solution of the instance I_2 of cost c_2 with relative error at most $\epsilon/(\alpha\beta(I_1))$ to a solution of the instance I_1 of cost c_1 such that the costs c_1 and c_2 satisfy $(\text{opt}(I_1) - c_1) \leq \beta(I_1)(\text{opt}(I_2) - c_2)$. Notice that β need not be a constant. Then, assuming that the problem C_1 is NP-hard to approximate within relative error ϵ, we can conclude immediately that it is NP-hard to find an approximate solution of instances $T_1(I_1)$ of problem C_2 with relative error smaller than $\epsilon/(\alpha\beta(I_1))$. We term this modified reduction a *generalized L-reduction*.

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 an appropriate k and then to an instance of the loading problem as in Theorem 6, 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. assume that $k \geq 2$) and (V', E') with $V' = V \cup \{v_{|V|+1}, \ldots, v_{|V|+k-3}\}$, $E' = E \cup \{(v_i, v_j) \mid i \in \{|V|+1, \ldots, |V|+k-3\}$ for $j \in \{1, \ldots, |V|+k-3\} \backslash \{i\}\}$ where the new edges in E' have the multiplicity $2|E|$ (i.e., $2|E|$ copies of each new edge are contained in E'). Reduce (V', E') to an instance of the loading problem for the $(n, k, 1)$-H-architecture with $n = |V'| + 3$, $k = |V| \cdot |E|$ and the following examples:

1. $2|E'|$ copies of the origin $(0^n; 1)$,

2. d_i copies of the point $\boldsymbol{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0; 0)$ (where the 1 is at the i^{th} position from left) for each $v_i \in V'$, d_i being the degree of v_i,

3. a vector $\boldsymbol{e}_{ij} = (0, \ldots, 0, 1, 0 \ldots, 0, 1, 0, \ldots, 0; 1)$ for each edge (v_i, v_j) in E' (where the two ones 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 $i \in \{1, \ldots, k\}$ and $j \in \{1, 2, 3\}$. 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. 5). More precisely , $\boldsymbol{p}^{ij} = \boldsymbol{x}^{ij} + \epsilon\boldsymbol{N}_i$ and $\boldsymbol{n}^{ij} = \boldsymbol{x}^{ij} - \epsilon\boldsymbol{N}_i$, where $\boldsymbol{N}_i$ is the normalized normal vector of the line $[\boldsymbol{x}^{i1}, \boldsymbol{x}^{i3}]$ and ϵ is a small value which can be chosen in such a way that the following holds:

   > 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 $i_1 = i_2 = i_3$.

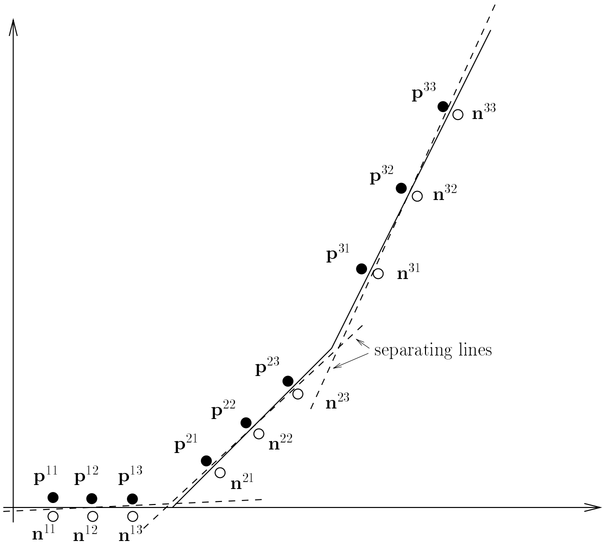


Figure 5: Construction of $\boldsymbol{p}^{ij}$ and $\boldsymbol{n}^{ij}$: The points result by dividing each point x^{ij} on the lines into a pair.

> Using Proposition 6 of [25] it is sufficient to choose $\epsilon \leq 1/(24k(k-1)+6)$. Hence the representation of $\boldsymbol{n}^{ij}$ and $\boldsymbol{p}^{ij}$ is polynomial in n and k.

Note that the number of points equals $5|E'|+12k|E'|$ which is at most $(|V||E|)^4 = k^4$ and at least $(|V||E|)^{3.5} = k^{3.5}$ for large enough $|V|$. 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:

- for node i in the hidden layer:
  - the j^{th} weight, for $1 \leq j \leq |V|$, is chosen as
  $$\begin{cases} -1 & \text{if } v_j \text{ is in the } i^{\text{th}} \text{ cut} \\ 2 & \text{otherwise} \end{cases}$$
  - the threshold is chosen as 0.5,
  - the $(|V'|+1)^{\text{th}}$, $(|V'|+2)^{\text{th}}$ and $(|V'|+3)^{\text{th}}$ weights are chosen as $(-i+1, 1, -0.5+2i(i-1))$ which corresponds to the line through the points $\boldsymbol{x}^{i1}$, $\boldsymbol{x}^{i2}$, and $\boldsymbol{x}^{i3}$.
- the output unit has the threshold $-k+0.5$ and all weights are 1, i.e. it computes the function AND: $(x_1,\ldots,x_k) \mapsto x_1 \wedge \ldots \wedge x_k$ of its inputs x_i.

With these choices of weights it is easily seen that all examples except the points $\boldsymbol{e}_{ij}$ corresponding to monochromatic edges are classified 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 $E'\backslash E$ correctly because of the multiplicities of the respective points. As in the proof of Theorem 8 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

$$\{(0,\ldots,0,x_{n+1},x_{n+2},1) \mid x_{n+1},x_{n+2} \in \mathbb{R}\}.$$

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 nodes. Hence the lines nearly coincide with the line through $[\boldsymbol{x}^{i1},\boldsymbol{x}^{i3}]$, $i=1,\ldots,k$. Denote the weights and the threshold of the output node of the network by $w_1,\ldots,w_k$ and θ, respectively. We can assume that the i^{th} node maps $\boldsymbol{p}^{ij}$ to 0 and $\boldsymbol{n}^{ij}$ to 1 for all j, since otherwise we change all signs of the weights and the threshold in neuron i, we change the sign of the weight w_i, and increase θ by w_i to satisfy this condition. Hence $\theta > 0$, $\theta + w_i < 0$ 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 $\text{NAND} : (x_1,\ldots,x_n) \mapsto \neg x_1 \wedge \ldots \wedge \neg x_n$ on binary values of $x_1,x_2,\ldots,x_n$.

Define a solution of the instance of the MAX-k-cut problem by setting the i^{th} cut c_i as $\{v_j \mid \text{the } i^{\text{th}} \text{ hidden node maps } \boldsymbol{e}_j \text{ to } 1\}\backslash(c_1 \cup \ldots \cup c_{i-1})$. 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 node, hence the vector $\boldsymbol{e}_{ij}$ is mapped to 1 also because of the classification of the origin. Hence $\boldsymbol{e}_{ij}$ is classified incorrectly. All $\boldsymbol{e}_{ij}$ points corresponding to edges in $E\backslash E'$ are classified correctly, hence each of the nodes $v_{|V|+1},\ldots,v_{|V|+k-3}$ form one cut and the remaining nodes are contained in the remaining three cuts. These three cuts define a solution of the instance of MAX-3-cut such that all edges corresponding to misclassified $\boldsymbol{e}_{ij}$'s 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

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

The quantity β which is computed by the algorithm T_1 equals $\tilde{c}k^3$, $\tilde{c}$ being a fixed positive constant which can be chosen appropriately such that $\tilde{c}\cdot k^3 \geq (5|E'| + 12|E'|k)/|E|$.

Next we construct T_2. Assume that a solution of the loading problem with relative error smaller than c/k^3 is given. Then the points 1 and 4 are correct due to their multiplicities. Otherwise the relative error of the problem would be at least $|E'|/(5|E'| + 12|E'|k) \geq c/k^3$ for appropriately small c and large k. As before we can assume that the output node computes the function NAND: $\boldsymbol{x} \mapsto \neg x_1 \wedge \ldots \wedge \neg x_k$. Define opt_2 to be the value of an optimum solution of the loading problem and c_2 the value of the given solution. Assume that 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 c_1 of this solution compared to an optimum opt_1 we have

$$\text{opt}_1 - c_1 \leq 1 \leq \frac{5|E'| + 12|E'|k}{|E|}(\text{opt}_2 - c_2)$$

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

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

$$\text{opt}_1 - c_1 \leq \frac{5|E'| + 12|E'|k}{|E|}(\text{opt}_2 - c_2)$$

With $\alpha = 3/2, \beta = \tilde{c} \cdot k^3 \geq (5|E'| + 12|E'|k)/|E|$ for appropriate constant $\tilde{c}$, and using Theorem 3, our result follows. □

In the above theorem, the number of points is upper bounded by a term involving the number of hidden nodes. Since the approximation factor depends on the number of hidden nodes, we added the lower bound $k^{3.5}$ which excludes situations where every point is to be classified correctly due to the bound of the approximation ratio and the size of the training set.

4 Hardness of Approximating the Failure Ratio Function

In the remaining part of this paper we consider another objective function, the objective of minimizing the failure ratio. We use the notations introduced in Section 2. Given an instance x of the loading problem, denote by $m_C(x, y)$ the number of points in the training set (counted with multiplicities) misclassified by a network y. 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 network with our weights. Notice that if $\text{opt}_C(x) > 0$, this is equivalent to investigating if the failure ratio m_f can be bounded above by a constant. 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 [2]. 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 where no solution without misclassified points exists.

4.1 Approximation within constant factors

We want to show NP-hardness of approximation of m_f within some bound by a layered H-net. It turns out that the bound on approximation of m_f for which we can prove NP-hardness is a constant *independent* of the number of nodes of the network architecture. For our purpose we use a reduction from the set-covering problem.

Definition 17 (Set Covering Problem [14])
Given a set of points $S = \{s_1, \ldots, s_p\}$ and a set of subsets $C = \{C_1, \ldots, C_m\}$ of S, find indices $I \subset \{1, \ldots, m\}$ such that $\bigcup_{i \in I} C_i = S$. If such a set of indices exists, then the sets $C_i, i \in I$, are called a cover of S (or, said to cover S). A cover is called exact if the sets in the cover are mutually disjoint. The goal in the optimization version of the set covering problem is to find a set of indices I for a cover with $|I|$ being the minimum possible.

Definition 18 (Satisfiability Problem [14])
Given a Boolean formula φ, in conjunctive normal form, over a set of variables U, find a truth assignment which satisfies the formula (i.e. makes the value of the formula true).

Both the satisfiability problem (or SAT problem for short) and the set-covering problem is known to be NP-hard [14]. For the set-covering problem the following result also holds, showing that it is NP-hard to approximate this problem within every constant factor $c > 1$.

Theorem 19 *[7] 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 $K \in \mathbb{N}$ with the properties: if φ is satisfiable then there exists an exact cover of size K, but if φ is not satisfiable then every cover has size at least $c \cdot K$.*

Using Theorem 19, Arora et.al. [2] show that approximating the minimum failure ratio is NP-hard for the simple perceptron model, i.e. $\{(n, 1) \mid n \in \mathbb{N}\}$ nets with threshold activation function, for every constant $c > 1$ if the threshold of the output node is zero. We can obtain a similar result for arbitrary layered H-nets where the thresholds of the neurons in the first hidden layer are fixed to 0.

Theorem 20 *Assume that we are given a layered H-net where the thresholds of the neurons in the first hidden layer are fixed to 0, the number of nodes in the first hidden layer is fixed, and the input dimension n varies. Let $c > 1$ be any given constant. Then the problem of approximating the minimum failure ratio for such an architecture while learning in the presence of errors within a factor $< c$ is NP-hard.*

Proof: The case without any nodes in the hidden layers is already proved in [2], hence we assume that at least one hidden layer is present. Assume that we are given a formula φ of SAT. First, we transform this formula in polynomial time (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 19 hold. Next, we transform this instance of the set-covering problem to an instance of the loading problem for the given architecture with input dimension $n = m + 2 + n_1 + 1$ (where n_1 denotes the number of nodes in the first hidden layer) and the following examples from $\mathbb{Q}^n \times \{0, 1\}$:

(I) $(\mathbf{e}_i, 0, 1, 0^{n_1+1}; 1)$, $(-\mathbf{e}_i, 0, 1, 0^{n_1+1}; 1)$, $\mathbf{e}_i \in \mathbb{R}^m$ being the i^{th} unit vector (for $1 \leq i \leq m$),

(II) $c \cdot K$ copies 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 $\mathbf{e}_{s_i} \in \{0, 1\}^m$ 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 the points $(0^m, 1, 0, 0^{n_1+1}; 1)$, $(0^m, 1/(2m), 1, 0^{n_1+1}; 1)$, and $(0^m, -1/(2m), 1, 0^{n_1+1}; 0)$, where the component $m + 1$ is nonzero in all three points and the component $m + 2$ is nonzero in the latter two points,

(IV) $c \cdot K$ copies of $(0^{m+2}, \boldsymbol{p}_i, 1; 0)$, $(0^{m+2}, \boldsymbol{p}_0, 1; 1)$, $(0^{m+2}, \tilde{\boldsymbol{z}}_i, 1; 1)$, $(0^{m+2}, \bar{\boldsymbol{z}}_i, 1; 0)$, for all vectors $\boldsymbol{p}_i \in \{-1, 1\}^{n_1} \backslash (1, \ldots, 1)$, $\boldsymbol{p}_0 = \{1\}^{n_1}$, where the points $\tilde{\boldsymbol{z}}_i$ and $\bar{\boldsymbol{z}}_i$ (for $i = 1, \ldots, n_1(n_1 + 1)$) are constructed as follows:

> select $n_1 + 1$ points in each set $H_i = \{\boldsymbol{x} = (x_1, x_2, \ldots, x_{n_1}) \in \mathbb{R}^{n_1} \mid x_i = 0, x_j > 0 \; \forall j \neq i\}$ (denote the points by $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, $\ldots$) such that any $n_1 + 1$ of these points lie on one hyperplane if and only if they are contained in one H_i. Such points exist as shown in [25]. For $z_j \in H_i$ define $\tilde{\boldsymbol{z}}_j \in \mathbb{R}^{n_1}$ by $\tilde{\boldsymbol{z}}_j = (z_{j1}, \ldots, z_{ji-1}, \epsilon, z_{ji+1}, \ldots, z_{jn_1})$, and $\bar{\boldsymbol{z}}_j \in \mathbb{R}^{n_1}$ by $\bar{\boldsymbol{z}}_j = (z_{j1}, \ldots, z_{ji-1}, -\epsilon, z_{ji+1}, \ldots, z_{jn_1})$, for some small value ϵ which is chosen such that the following property holds: if one hyperplane in $\mathbb{R}^{n_1}$

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. It is shown in [25] that such points exist and an appropriate ϵ can be computed depending on the points $\boldsymbol{z}_i$.

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} node in the first hidden layer has the weights $(\mathbf{e}_I, 1, 1/(4m), \mathbf{e}_i, 0)$ 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 $\mathbb{R}^{n_1}$. Each of the remaining nodes in the other layers computes the function $\boldsymbol{x} \mapsto x_1 \wedge \ldots \wedge x_i \wedge \ldots$ of their inputs x_i. Since the cover is exact, this maps all examples correctly except K examples in **(I)** corresponding to sets in the cover.

Conversely, assume that every cover has size at least $c \cdot K$. Assume, for the sake of contradiction, that there is some weight setting that misclassifies less than $c \cdot K$ examples. We can assume that the activation of every node is different from 0 on the set of examples: for the examples in **(IV)** the weight w_n serves as a threshold, for the points in **(I)**, **(II)**, and **(III)** except for $(0^m, 1, 0^{n_1+2}; 1)$ the weight w_{m+2} serves as a threshold, hence one can change the respective weight which serves as a threshold without changing the classification of these examples such that the activation becomes nonzero via enlarging the respective weight by $|d|/4$, d being the maximum negative activation of the neuron. Assuming that the activation of $(0^m, 1, 0^{n_1+2}; 1)$ is zero we can increase the weight w_{m+1} such that the sign of the activation of all other points which are affected does not change. The precise value can be computed in polynomial time depending on the other weights and activations of the points. 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} node 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 nodes in the first hidden layer to nearly coincide 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 nodes 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 node in the first hidden layer which classifies the negative example in **(III)** correctly as well. Consider this last node. Denote by $\boldsymbol{w}$ the (vector of) weights of this node. Because of the examples in **(III)**, $w_{m+1} > 0$. Define $I = \{i \in \{1, \ldots, m\} \mid |w_i| \geq w_{m+1}/(2m)\}$.

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

Otherwise, 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 $< m \cdot w_{m+1}/(2m) - w_{m+1} + w_{m+2} = w_{m+2} - w_{m+1}/2$ which is < 0 because $-w_{m+1}/(2m) + w_{m+2} < 0$, $w_{m+1} > 0$ **(III)**. This yields a misclassified example with multiplicity $c \cdot K$. This is again a contradiction.

We can now complete the proof of the theorem very easily. Assume, for the sake of contradiction, that we can approximate the minimum failure ratio of the loading problem within a factor of c. Then we could transform, in polynomial time, a given instance φ of the SAT problem to an instance of the set cover problem and then to an instance of the the loading problem as described above with the following property:

- if φ is satisfiable, then the loading problem has a solution with K misclassifications,
- if φ is not satisfiable, then every solution of the loading problem has at least cK misclassifications.

Since, we can approximate the minimum failure ratio within a factor $< c$ in polynomial time, we can decide, given an approximate solution of the loading problem, whether the loading problem has a solution with K misclassifications, or whether every solution of the loading problem has at least cK misclassifications. This means that we would then know if φ is satisfiable or not, thereby solving the SAT problem, an NP-hard problem, in polynomial time. □

Since an arbitrary constant $c > 1$, which is 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.

4.2 Approximation within large factors

Assuming that NP$\not\subseteq$DTIME($n^{\text{poly}(\log n)}$) one can show that even obtaining weak approximations, i.e. approximations within some large factor which depends on the input dimension, is not possible[5]. For this purpose a reduction from the so-called label cover problem is used.

Definition 21 (Label Cover) *Given a bipartite graph $G = (V, W, E)$ with $E \subset V \times W$, labels in sets B and D, and a set $\Pi \subseteq E \times B \times D$, a* labeling *of G consists of functions $P : V \to 2^B$ and $Q : W \to 2^D$ which assign labels to the nodes in the graph. The* cost *of this labeling is $\sum_{v \in V} |P(v)|$. An edge $e = (v, w)$ is* covered *if both $P(v)$ and $Q(w)$ are not empty and for all $d \in Q(w)$ there exists some $b \in P(v)$ with $(e, b, d) \in \Pi$. A* total cover *of G is a labeling such that each edge is covered. The goal for the optimization version of the label cover problem is to find a* total *cover with* minimum *cost.*

For the label cover problem the following result holds, showing that it is almost NP-hard to obtain weak approximations.

Theorem 22 [2, 22] *For every fixed $\epsilon > 0$ there exists a quasi-polynomial time reduction from the SAT problem to the label cover problem which maps an instance φ of size n to an instance (G, Π) of size*[6] *$N \leq 2^{\text{poly}(\log n)}$ with the following properties:*

- *If φ is satisfiable then (G, Π) has a total cover with cost $|V|$.*
- *If φ is not satisfiable then every total cover has a cost of at least $2^{\log^{0.5-\epsilon} N}|V|$.*
- *Furthermore, in both cases (G, Π) satisfies the property that, for each edge $e = (v, w)$ and $b \in B$, at most one $d \in D$ exists with $(e, b, d) \in \Pi$.*

[5] A quasi-polynomial time algorithm is an algorithm that runs in $O(n^{\text{poly}(\log n)})$ time, where n is the size of the input and poly$(\log n)$ is a fixed polynomial in $\log n$. DTIME($n^{\text{poly}(\log n)}$) refers to the class of problems that can be solved by a deterministic Turing machine in quasi-polynomial time. More information about these and related topics is available in any standard textbook in structural complexity theory, such as [5, 14].

[6] We omit any precise definition of the size of an instance φ of the SAT problem and the size of an instance (G, Π) of the label cover problem, since those will not be necessary.

Using this Theorem and ideas of Arora et.al. [2] we can prove the following theorem.

Theorem 23 *Assume that we are given a layered $(n, n_1, n_2, \ldots, n_h)$ H-net (where n_1 is fixed and n is the varying input dimension) where the thresholds of all the nodes in the first hidden layer are fixed to 0 and let $\epsilon > 0$ be any given constant. If the problem of approximating minimum failure ratio m_f while learning in the presence of errors for this architecture within a factor $< 2^{\log^{0.5-\epsilon} n}$ can be solved in polynomial time, then NP $\subset$ DTIME($n^{\mathit{poly}(\log n)}$).*

Proof: We can assume that $h > 0$ since otherwise the result is already proven in [2]. Assume that we are given a formula φ of the SAT problem of size α. We transform φ with the given constant ϵ to an instance (G, Π) of the label cover problem of size N with the properties as described in Theorem 22 for this ϵ.

First, following the same approach as in [2], we delete all $(e = (v, w), b, d)$ in Π such that for some edge e' incident to v no d' exists with $(e', b, d') \in \Pi$. The remaining labels are called *valid* labels. The cost of a total cover still remains $|V|$ if φ is satisfiable (since the label cover in such a case uses only 1 label for a vertex). Otherwise, if φ is not satisfiable, then this deletion can only increase the cost of a total cover. After these deletions, by Theorem 22, for each $e \in E$ and $b \in B$ there exists a unique $d = d(e, b) \in D$ such that $(e, b, d) \in \Pi$. We can assume that a total cover exists, since this can be easily checked in polynomial time.

Now transform this instance to an instance of the loading problem. The input dimension is $n = x + 2 + n_1 + 1$ where $x = |V||B| + |W||D|$, $E \subset V \times W$ are the edges, and B and D are the labels. By the results in [2] (see Theorem 7, Lemma 9 and Theorem 13 of [2]) $N \geq |E|(1 + |D|) + |V||B| \geq x + 3$ (with $|E| \geq 3$) Hence $n \leq n_1 + N$. Let $m = \max\{|B|, |D|\}$ and $K = |B||E|$ for notational simplicity. The following examples from $\mathbb{Q}^n \times \{0, 1\}$ are constructed: (the first x components are successively identified with the tuples in $V \times B$ and $W \times D$ and denoted via the corresponding indices.)

(I) K copies of the points $(0^{x+2}, \boldsymbol{p}_i, 1; 0)$ ($i \geq 1$), $(0^{x+2}, \boldsymbol{p}_0, 1; 1)$, $(0^{x+2}, \tilde{\boldsymbol{z}}_i, 1; 1)$, $(0^{x+2}, \bar{\boldsymbol{z}}_i, 1; 0)$, where the points $\boldsymbol{p}_i$, $\tilde{\boldsymbol{z}}_i$, $\bar{\boldsymbol{z}}_i$ are the same points as in the proof of Theorem 20.

(II) K copies of $(0^{|x|}, 1, 0, 0^{n_1+1}; 1)$,

(III) K copies of $(0^{|x|}, 1/(16m^2), 1, 0^{n_1+1}; 1)$ and $(0^{|x|}, -1/(16m^2), 1, 0^{n_1+1}; 0)$,

(IV) K copies of each of $(\mathbf{e}_v, -1, 1, 0^{n_1+1}; 1)$, $(\mathbf{e}_w, -1, 1, 0^{n_1+1}; 1)$, where $\mathbf{e}_v$ is 1 precisely at those places (v, b) such that b is a valid label for v and 0 otherwise, and $\mathbf{e}_w$ is 1 precisely at the places (w, d) such that $d \in D$ ($v \in V$, $w \in W$).

(V) K copies of each of $(-\mathbf{e}_{v \to w, d}, 1, 1, 0^{n_1+1}; 1)$, where $-\mathbf{e}_{v \to w, d}$ is -1 precisely at those places (v, b) such that b is a valid label for v and d is not assigned to $(v \to w, b)$ and at the place (w, d) and 0 otherwise ($v \to w \in E$).

(VI) $(-\mathbf{e}_{v,b}, 0, 1, 0^{n_1+1}; 1)$, where $-\mathbf{e}_{v,b}$ is -1 precisely at those places (v, b) such that b is a valid label for v.

We now prove the following two claims:
(a) If a total cover with cost $|V|$ exists, then the number of misclassified points in an optimum solution of the loading problem is at most $|V|$.

(b) The number of misclassified points in any optimum solution of the loading problem is at least C, the minimum possible cost of a total cover.

Assuming both (a) and (b) are true, we can complete the proof of our theorem easily as follows. Given an instance φ of the SAT problem, we can transform this instance in quasi-polynomial time to an instance of the label cover problem for the given ϵ and then to an instance of the loading problem as shown above such that the following holds:

- if φ is satisfiable, then the loading problem has a solution with $|V|$ misclassifications,
- if φ is not satisfiable, then every solution of the loading problem misclassifies at least $2^{\log^{0.5-\epsilon} N}|V| \geq 2^{\log^{0.5-\epsilon}(n-n_1)}|V|$ points.

Assume, for the sake of contradiction, that we can approximate the minimum failure ratio of the loading problem within a factor smaller than $2^{\log^{0.5-\epsilon}(n-n_1)}|V|$. Then, we can decide in quasi-polynomial time, given an approximate solution of the loading problem, whether the loading problem has a solution with $|V|$ misclassification, or whether every solution of the loading problem has at least $2^{\log^{0.5-\epsilon}(n-n_1)}|V|$ misclassifications. This means that we would then know if φ is satisfiable or not, thereby solving the SAT problem, an NP-hard problem, in quasi-polynomial time. Since this holds for every $\epsilon > 0$ and $2^{\log^{0.5-2\epsilon} n} \leq 2^{\log^{0.5-\epsilon}(n-n_1)}$ for large n, the result as stated in the theorem follows.

The remainder of this proof is devoted to proving the claims (a) and (b) above. First, we prove claim (a). Assume that a label cover with costs $|V|$ exists. Define the weights for the neurons in the first computation layer by $w_{(v,b)} = 1 \iff b$ is assigned to v, $w_{(w,d)} = 1 \iff d$ is assigned to w, $w_{x+1} = 1$, $w_{x+2} = 1/(32m^2)$. The remaining coefficients of the i^{th} node in the first hidden layer are defined by: $w_{x+2+i} = 1$, the remaining coefficients are 0. The nodes in other layers compute the function $\boldsymbol{x} \mapsto x_1 \wedge \ldots \wedge x_i \wedge \ldots$ of their inputs x_i. This maps all points but at most $|V|$ points in **(VI)** to correct outputs. Note that the points in **(V)** are correct since each v is assigned precisely one b. This concludes the proof of (a).

Now we prove (b). Assume that a solution of the loading problem is given. We show that it has a number of misclassified points which is at least the cost C of an optimum total cover. Assume for the sake of contradiction that less than C points are classified incorrectly. Since a cover has a cost at most K we can assume that all points with multiplicities are classified correctly. Because of the same reasoning as in Theorem 20 we can assume that the activation of every node is different from 0 on the training set. Additionally, we can assume that the output of the circuit 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} node in the first hidden layer, because of the points in **(I)**. Hence all nodes in the first hidden layer classify all positive examples except less than C points of **(V)** correctly and there exists one node in the first hidden layer which classifies the negative example in **(III)** correctly as well.

Denote by $\boldsymbol{w}$ the weights of this node. Because of **(II)**, $w_{|x|+1} > 0$. Label the node v with those valid labels b such that the inequality $w_{(v,b)} > w_{x+1}/(4m^2)$ holds. Label the node w with those labels d such that $w_{(w,d)} > w_{x+1}/(2m)$.

If this labeling forms a total cover, then we find for all b assigned to v in **(VI)** an activation smaller than $-w_{x+1}/(4m^2)+w_{x+2}$. Due to **(III)**, $w_{x+2} < 1/(16m^2)\cdot w_{x+1}$, hence the activation is smaller than 0 and leads to a number of misclassified points which is at least C.

Assume otherwise that this labeling does not form a total cover. Then some v or w is not labeled, or for some label d for w and edge $v \to w$ no b is assigned to v with $(v \to w, b, d) \in \Pi$. Due to **(IV)** we find $\sum_{b \text{ valid for } v} w_{(v,b)} - w_{x+1} + w_{x+2} > 0$, hence together with **(III)** $\sum_{b \text{ valid for } v} w_{(v,b)} > w_{x+1} - w_{x+1}/(16m^2)$, hence at least one $w_{(v,b)}$ is of size at least $w_{x+1}/(2m)$. In the same way we find $\sum_d w_{(w,d)} - w_{x+1} + w_{x+2} > 0$, hence at least one $w_{(w,d)}$ is of size at least $w_{x+1}/(2m)$. Consequently, each node is assigned some label. Assume that the node w is assigned some d such that the edge $v \to w$ is not covered. Hence $w_{(w,d)} > w_{x+1}/(2m)$. Due to the points in **(V)** we find that the inequality $-\sum_{b \text{ valid for } v,\, d(v \to w,\, b) \neq d} w_{(v,b)} - w_{(w,d)} + w_{x+1} + w_{x+2} > 0$ holds and due to **(IV)** we find $\sum_{b \text{ valid for } v} w_{(v,b)} - w_{x+1} + w_{x+2} > 0$, hence we can conclude: $\sum_{b \text{ valid for } v,\, d(v \to w,\, b) = d} w_{(v,b)} > w_{x+1} - w_{x+2} - \sum_{b \text{ valid for } v,\, d(v \to w,\, b) \neq d} w_{(v,b)} > w_{x+1} - w_{x+2} + w_{(w,d)} - w_{x+1} - w_{x+2} = w_{(w,d)} - 2w_{x+2} > w_{x+1}(1/(2m) - 1/(8m^2)) > w_{x+1}/(4m)$. Hence at least one weight corresponding to a label which can be used to cover this edge is of size at least $w_{x+1}/(4m^2)$. This concludes the proof of (b). □

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 [9] to more realistic situations. We have considered the question as to whether approximating relative error within a constant factor is NP-complete. Compared to [4] we considered 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 NP-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 a fixed number of nodes in the first hidden layer and all thresholds in the first hidden layer fixed to 0). Assuming stronger conjectures in complexity theory, we established that even weak approximations cannot be obtained in the same situation.

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 [9], the NP-completeness for the $(n, 2, 1)$ architecture with binary examples is shown. For more hidden nodes this is unsolved if only one output node is present. Some work for multilayered architectures can be found in [27].

2. What is the complexity of training $(n, n_1, 1)$ networks with the sigmoidal activation function in the hidden nodes? [18, 34] 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. Since classification is an easier task 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 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 nodes in the hidden layers? We obtained one result in this context, but only with error bounds which depend on the number of hidden neurons.

5. Can a general argument be found which will show the validity of the NP-hardness results for examples without multiplicities? We used a step by step analysis.

6. What are the characteristics of a set of examples for which loading is NP-hard? It is well known that pairwise orthogonal training examples can be classified correctly even without hidden neurons. Can, for example, an example set with limited correlation of the points, i.e. bounded values $|\boldsymbol{x}_i^t \boldsymbol{x}_j|/(|\boldsymbol{x}_i||\boldsymbol{x}_j|) \leq C$ for all pattern $\boldsymbol{x}_i \neq \boldsymbol{x}_j$ and some constant C, be loaded efficiently? Some investigation concerning this topic can be found in [29]. The authors in [8] show that the situation of [9] changes if the input examples come from a specific (realistic) input distribution; then training is possible in polynomial time.

References

[1] E. Amaldi and V. Kann, The complexity and approximability of finding maximum feasible subsystems of linear relations, Theoretical Computer Science 147 (1-2), pp. 181-210, 1995.

[2] 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.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, Complexity and Approximation, Springer, 1999.

[4] P. Bartlett and S. Ben-David, Hardness results for neural network approximation problems, to appear in Theoretical Computer Science (conference version in Fischer P. and Simon H. U. (eds.), Computational Learning Theory, Lecture Notes in Artificial Intelligence 1572, Springer, pp. 50-62, 1999).

[5] J. L. Balcázar, J. Díaz and J. Gabarro, Structural Complexity I, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1988.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] M. P. do Carmo, Differential Geometry of Curves and Surfaces, Prentice Hall, 1976.

[12] T. H. Cormen, C. E. Leiserson and R. L. Rivest, Introduction to Algorithms, The MIT Press, 1990.

[13] 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.

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

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

[16] 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.

[17] 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.

[18] L. K. Jones, The computational intractability of training sigmoidal neural networks, IEEE Transactions on Information Theory 43(1), pp. 167-173, 1997.

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

[20] 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.

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

[22] C. Lund and M. Yannakakis, On the hardness of approximate minimization problems, Journal of the ACM, 41(5), pp. 960-981, 1994.

[23] W. Maass, G. Schnitger, 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.

[24] 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.

[25] N. Megiddo, On the complexity of polyhedral separability, Discrete Computational Geometry 3, pp. 325-337, 1988.

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

[27] C. C. Pinter, Complexity of network training for classes of neural networks, Jantke K. P., Shinohara T., and Zeugmann T.(eds.), Algorithmic Learning Theory, ALT 95, Springer, pp. 215-227, 1995.

[28] R. D. Reed and R. J. Marks, Neural smithing, MIT Press, 1999.

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

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

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

[32] M. Spivak, A comprehensive introduction to differential geometry, vol.1-5, Boston, Mass.: Publish or Perish, 1970-1975.

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

[34] 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.

[35] P. Werbos, The roots of backpropagation, Wiley, 1994.

APPENDIX

Proof of Theorem 8: The proof is via Corollary 7 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)**. By Theorem 3 we may assume that $\epsilon = 1/(34(k-1))$ for the proof of ρ.

The reduction is as follows. 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 $\mathbb{Q}^n \times \{0,1\}$ with $n = |V| + n_1 + 1$:

- The set of special points P_0 (together with their labeling) are the points:
  - $(0^n; 1)$
  - $(0^{|V|}, \boldsymbol{p}_i, 1; 0)$ 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 $H_i = \{\boldsymbol{x} = (x_1, x_2, \ldots, x_{n_1}) \in \mathbb{R}^{n_1} \mid x_i = 0, \forall j \neq i \; x_j > 0\}$, denote the points by $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, … 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. Such points exist as shown in [25]. For $\boldsymbol{z}_j \in H_i$ define $\tilde{\boldsymbol{z}}_j \in \mathbb{R}^{n_1}$ by $\tilde{\boldsymbol{z}}_j = (z_{j1}, \ldots, z_{ji-1}, \epsilon, z_{ji+1}, \ldots, z_{jn_1})$, and $\bar{\boldsymbol{z}}_j \in \mathbb{R}^{n_1}$ by $\bar{\boldsymbol{z}}_j = (z_{j1}, \ldots, z_{ji-1}, -\epsilon, z_{ji+1}, \ldots, z_{jn_1})$, for some small value ϵ which is chosen such that the following property holds: if

one hyperplane in $\mathbb{R}^{n_1}$ 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. It is shown in [25] that an appropriate ϵ can be computed depending on the points $\boldsymbol{z}_i$.

The role of the points $\tilde{\boldsymbol{z}}_i$ and $\bar{\boldsymbol{z}}_i$ is to enforce that in any network classifying 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 $1 \leq i \leq |V|$). 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 6. Given an optimum solution of the MAX-n_1-cut problem, we choose the threshold of the i^{th} node 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 nodes compute the function $\boldsymbol{x} \mapsto x_1 \wedge \ldots \wedge x_i \wedge \ldots$ of their inputs. This maps all points correctly 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 smaller than ρ classifies P_0 correctly. After changing the thresholds if necessary, we can assume that no point in the training set leads to an activation of exactly 0 for one of the nodes. It is sufficient to increase each threshold by $|d|/4$, d being the maximum negative activation of the (finite set of) different inputs to this neuron.

Consider the n_1 hyperplanes defined by $\{\boldsymbol{x} \in \mathbb{R}^{n_1} \mid f_i(0^{|V|}, \boldsymbol{x}, 1) = 0\}$ where f_i is the output of the i^{th} node 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 $(f_1, \ldots, f_{n_1})$. Since $H(x) = 1 - H(-x)$ for $x \neq 0$ we can assume, after a standard weight change if necessary, that the point $(0^{|V|}, \boldsymbol{p}_0, 1)$ is mapped to 1 by all nodes in the first hidden layer. Because of the classification of the points $(0^{|V|}, \boldsymbol{p}_i, 1)$ the remaining part of the network necessarily computes the logical function 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 i^{th} weight in any node in the first hidden layer if necessary, such that the sum of this weight and the threshold is at least 0, we can obtain a solution of at least the same quality where the point $\boldsymbol{e}_i$ is classified correctly for any i.

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} node in the first hidden layer as before. Note that for every $\boldsymbol{e}_j$ at least one node with activation 0 exists. Assume an edge (v_j, v_l) is monochromatic. Consequently, the output of at least one node, say hidden node h, is zero for both $\boldsymbol{e}_j$ and $\boldsymbol{e}_l$. Because of the classification of the origin the threshold of this node is positive, i.e. $\boldsymbol{e}_{jl}$ has the output 0 for node h, too, and is classified incorrectly. Hence, property **(i)** is established.

In order to establish property **(ii)** of Theorem 6 define T_2 as follows. Given a solution of the loading problem which classifies P_0 correctly we have already seen how to 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 nodes 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, …, $i-1$. 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 correctly 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 correctly classified and hence some f_{j_i} or f_{j_j}, respectively, yield 0, leads to a incorrectly classified $\boldsymbol{e}_{ij}$ because of the classification of the origin.

The algorithms T_1 and T_2 are polynomial time since n_1 is a constant. □

Proof of Theorem 12: The proof is via Theorem 6. An instance graph $I_1 = (V, E)$ of the MAX-2-cut problem is mapped to the following set of examples in $\mathbb{Q}^n \times \{0,1\}$ with $n = |V| + 6$:

- The set of special points P_0 together with their labeling is as follows:

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

- $\boldsymbol{e}_i$ and $\boldsymbol{e}_{ij}$ are the same vectors with the same labeling as in the sigmoidal case (see proof of Theorem 9).

Again we want to see how a classification looks like. 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 9. We can assume $\alpha \neq 0$, $\beta \neq 0$, $\boldsymbol{a} \neq 0$ if P_0 is loaded correctly. If we are only interested in the geometric form of the output of the network, we can assume, by an argument similar to what was presented in Step **(2)** of Theorem 9, that $\boldsymbol{a}^t \boldsymbol{x} + a_0 = x_1$. 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

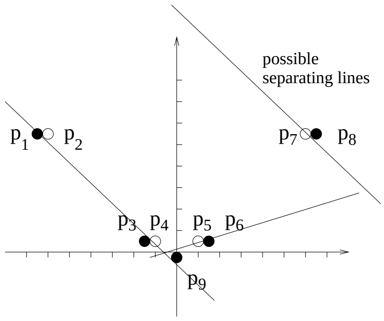


Figure 6: Points which exclude Case 3, the black points are to be mapped to 1, the white points are to be mapped to 0.

weight changes as in the sigmoidal case since $\text{lin}_{0.5}$ possesses the same symmetry as sgd, and substitute $\text{lin}_{0.5}$ 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. We can assume that $\gamma > 0$ by an argument similar to Case **(2)** in Step **(2)** of Theorem 9. 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 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 continuous curve with at most 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: Exactly two values are positive. We can assume that γ and $\beta+\gamma$ are positive and all other values are negative by an argument similar to Case **(3)** in Step **(2)** of Theorem 9.

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 $\{\boldsymbol{x} \mid x_1 = -\gamma/\alpha, \boldsymbol{b}^t\boldsymbol{x} + b_0 \leq 0\}$, $\{\boldsymbol{x} \mid x_1 = (-\beta - \gamma)/\alpha, \boldsymbol{b}^t\boldsymbol{x} + b_0 \geq 1\}$, and $\{\boldsymbol{x} \mid (\alpha\boldsymbol{a} + \beta\boldsymbol{b})^t\boldsymbol{x} + \beta b_0 + \gamma = 0\}$. That means that 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. This case is dual to Case 2. We obtain the possible forms by changing the output sign, i.e. the positive and negative region.

Next we show that only Case 2 can take place if P_0 is classified correctly.

Obviously, Case 1 cannot take place. Case 4 is excluded by the points which are nonvanishing in dimension $|V|+1$ to $|V|+3$: we need 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 must separate two of the pairs (p_1,p_2), (p_3,p_4), (p_5,p_6), and (p_7,p_8), without loss of generality, say 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 6 holds. An optimum solution of the MAX-2-cut problem leads to a solution with weights $\alpha=\beta=-1$, $\gamma=0.5$, $a_0=-0.5$, $b_0=-0.5$,

$$\begin{aligned}\boldsymbol{a}&=(a_1,\dots,a_n,1,-1,1,0.5,-0.5,0.625),\\ \boldsymbol{b}&=(b_1,\dots,b_n,-1,1,-1,-0.5,-0.5,0.625),\end{aligned}$$

where

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

and

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

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. This solution classifies P_0 correctly. Hence the classification is of Case 2. Without loss of generality, assume that $\gamma>0$ (computing the appropriate weight changes in polynomial time, if necessary) and all points $\boldsymbol{e}_i$ are correctly classified (decrease the i^{th} weight in every node, if necessary, such that the sum of the weight and the respective threshold is <0). 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) for which $\boldsymbol{e}_{ij}$ 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 6 can be shown: given a weight setting such that P_0 is classified correctly we 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

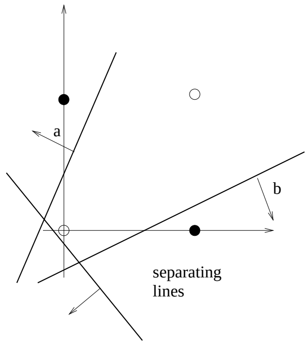


Figure 7: Classification in case 2. The black points are to be mapped to 0, the white points are to be mapped to 1.

$a_i > 0$. 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. □

Proof of Theorem 13:
The proof is analogous to Theorem 6. We obtain the following inequality, for an optimum solution with value opt(I_1) of an instance I_1 of the MAX-k-cut problem and an optimum solution with value opt(I_2) of an instance I_2 of the corresponding loading problem, because of **(i')**:

$$\begin{aligned}\text{opt}(I_2) &\leq \frac{3|E|p_0 + 5|E| + |E|\text{opt}(I_1)}{3|E|p_0 + 6|E|} \\ &\leq \frac{k}{k-1}\text{opt}(I_1)\,.\end{aligned}$$

Hence α can be chosen as $k/(k-1)$. Any approximate solution of I_2 with relative error of c_2 smaller than $a = (k-1)/(k^2(3p_0+6))$ classifies at least one point of each set $\{p^i \mid i\}$ of the special points P_0 correctly because otherwise

$$\begin{aligned}c_2 &\leq \frac{3|E|(p_0-1) + 6|E|}{3|E|p_0 + 6|E|} \\ &= \frac{p_0+1}{p_0+2} < (1-a)\text{opt}(I_2)\end{aligned}$$

for $a < ((3-2/k)p_0 + 6 - 4/k)/((p_0+2)(3p_0+6-2/k))$ because opt$(I_2) \geq (3|E|p_0 + 4|E| + 2|E|(1-1/k))/(3|E|p_0+6|E|)$. The above inequality cannot hold due to the definition of the relative error. If at least one p^i of each set in P_0 is correct we obtain because of **(ii')**, c_1 and c_2 denoting the cost of a solution of I_1 or I_2, respectively:

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

Hence the NP-hardness of approximate loading follows. □

Proof of Corollary 14:
A reduction from the MAX-k-cut problem as stated in Theorem 13 would show that approximate loading is NP-hard within relative error $1/(34k(3|P_0|+6))$ for $k \geq 2$.

The arguments in the theorems in previous sections can be transferred to this new settings. Note that the origin is used in Theorems 8, 9, and 12 for two different purposes: regarded as a point in P_0 it is used to exclude certain geometrical situations 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 introduced $\boldsymbol{o}_{ij}$.

Note that we constructed explicit weights 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 nodes in the hidden layer. Hence the continuity of the network function allows us to find some δ such that the classifications of the respective points do not change if they are substituted by any point contained in the open ball of radius δ whose center is the respective point. Note that δ does not depend on the specific instance in all cases.

Hence for every modification of the training set such that the points p_i^k lie in a small neighborhood of p_i, $\boldsymbol{e}_i^k$ lie in a small neighborhood of $\boldsymbol{e}_i$, and $\boldsymbol{o}_{ij}$ lie in a small neighborhood of the origin we can find a solution of the same quality as before.

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 8 the point $\tilde{\boldsymbol{z}}_j = (z_{j1}, \ldots, z_{ji-1}, \epsilon, z_{ji+1}, \ldots, z_{jn_1})$ can be substituted by $(z_{j1}, \ldots, z_{ji-1}, \epsilon', z_{ji+1}, \ldots, z_{jn_1})$ and the point $\bar{\boldsymbol{z}}_j \in \mathbb{R}^{n_1}$ can be substituted by $(z_{j1}, \ldots, z_{ji-1}, -\epsilon, z_{ji+1}, \ldots, z_{jn_1})$ 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 nodes 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 the logical function AND.

- We can substitute the points in Theorem 9 as follows:

  - $(0^{|V|}, 0, 0, 0, -0.5, 0.5; 1)$ 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 $(0^{|V|}, 0, 0, 0, -1 - c + \epsilon, c; 0)$.

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

  We substitute

  1. $(0^{|V|}, 0, 0, 0, 0, 0; 1)$
  2. $(0^{|V|}, 1, 1, 0, 0, 0; 1)$

3. $(0^{|V|}, 0, 1, 1, 0, 0; 1)$
4. $(0^{|V|}, 1, 0, 0, 0, 0; 0)$
5. $(0^{|V|}, 0, 1, 0, 0, 0; 0)$
6. $(0^{|V|}, 0, 0, 1, 0, 0; 0)$
7. $(0^{|V|}, 1, 1, 1, 0, 0; 0)$

by the points

1. $(0^{|V|}, \epsilon, \epsilon, \epsilon, 0, 0; 1)$,
2. $(0^{|V|}, 1-\epsilon, 1-\epsilon, \epsilon, 0, 0; 1)$, $\quad (**)$
3. $(0^{|V|}, \epsilon, 1-\epsilon, 1-\epsilon, 0, 0; 1)$,
4. $(0^{|V|}, 1+0.5\epsilon_1, -\epsilon_1, -0.5\epsilon_1, 0, 0; 0)$ $\qquad (*)$
   and $(0^{|V|}, 1+\epsilon_2, -0.5\epsilon_2, -0.5\epsilon_2, 0, 0; 0)$,
5. $(0^{|V|}, -\epsilon_1, 1+0.5\epsilon_1, -0.5\epsilon_1, 0, 0; 0)$, $\qquad (*)$
   $(0^{|V|}, -0.5\epsilon_2, 1+\epsilon_2, -0.5\epsilon_2, 0, 0; 0)$,
   and $(0^{|V|}, -0.5\epsilon_3, 1+0.5\epsilon_3, -\epsilon_3, 0, 0; 0)$,
6. $(0^{|V|}, -0.5\epsilon_1, -0.5\epsilon_1, 1+\epsilon_1, 0, 0; 0)$
   and $(0^{|V|}, -0.5\epsilon_2, -\epsilon_2, 1+0.5\epsilon_2, 0, 0; 0)$,
7. $(0^{|V|}, 1+0.5\epsilon_1, 1+0.5\epsilon_1, 1+\epsilon_1, 0, 0; 0)$ $\qquad (*)$
   and $(0^{|V|}, 1+\epsilon_2, 1+0.5\epsilon_2, 1+0.5\epsilon_2, 0, 0; 0)$,

where $0 < \epsilon, \epsilon_1, \epsilon_2, \epsilon_3 < 0.5$ are 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. For example, 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 12 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 $0 < \epsilon < 0.2$ 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 number of points and their multiplicities 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 separating $\boldsymbol{e}_i^k$ or $\boldsymbol{e}_j^k$ from $\boldsymbol{o}_{ij}$ and $\boldsymbol{e}_{ij}$ has signs $(+,-)$ or $(-,+)$, respectively, in dimensions i and j.

Hence we can establish properties **(i')** and **(ii')** 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 misclassified for small parameters ϵ. Conversely, any optimum solution of the loading problem classifies for each i at least one p_i^k correct and hence the same geometrical situations as before take place. Furthermore, we can assume that for each i at least one $\boldsymbol{e}_i^k$ is correct, otherwise a weight change (which is computable in polynomial time) would lead to a solution of at least the same quality and all $\boldsymbol{e}_i^k$ correct. Without loss of generality, $\gamma > 0$ if we adapt the proof of Theorems 9 and 12; or the second part of the network computes the logical function $(x_1,\ldots,x_{n_1}) \to x_1 \wedge \ldots \wedge x_{n_1}$ of its inputs x_i if we adapt the proof of Theorem 8. We can define the j^{th} cut via $\{v_i \mid$ the j^{th} node 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 we adapt the proof of Theorem 8. We can define the first cut via $\{v_i \mid a_i > 0\}$ if we adapt the proofs in Theorems 9 and 12. 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 8 or because of $(*)$ in Theorems 9 and 12. Hence Property **(i')** holds. Property **(ii')** follows in the same way. □