BACK PROPAGATION IS NOT EFFICIENT

Jiří Šíma *
Department of Theoretical Informatics
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2
182 07 Prague 8
Czech Republic
e-mail: sima@uivt.cas.cz

Abstract

The back-propagation learning algorithm for multi-layered neural networks, which is often successfully used in practice, appears very time consuming even for small network architectures or training tasks. However, no results are yet known concerning the complexity of this algorithm. Blum and Rivest proved that training even a 3-node network is NP-complete for the case when a neuron computes the discrete linear threshold function. We generalize the technique from their NP-hardness proof for a continuous sigmoidal function used in back-propagation. We show that training a 3-node sigmoid network with an additional constraint on the output neuron function (e.g. zero threshold) is NP-hard. As a consequence of this, we find training sigmoid feedforward networks, with a single hidden layer and with zero threshold of output neuron, to be intractable. This implies that back-propagation is generally not an efficient algorithm, unless at least $P = NP$. We take advantage of these results by showing the NP-hardness of a special nonlinear programming problem.

1 Introduction

Recently, there has been growing interest in computer architectures, which are alternatives to the von Neumann model. The prominent position among them is occupied by neural network models that have been inspired by neurophysiological knowledge. The simulations of simplified mathematical models of neural networks exhibit surprisingly 'intelligent' behaviour, similar to human intelligence — e.g., the ability to learn and to generalize previous experience. However, it is difficult to formalize the advantages of the neural approach, because it is not clear what human intelligence is.

On the other hand, these models have been successfully applied to solving simple practical problems (up to hundreds of neurons) where the standard algorithmical approach failed (e.g., pattern recognition, signal processing, prediction, expert systems, etc.). Successful applications of neural networks have led to uncritical conviction that it would be possible to create a thinking robot in the future.

The most widely applied neural network model is a multi-layered neural network trained with the back-propagation learning algorithm (Rumelhart et al., 1986). Unfortunately, the drawback of this learning algorithm, likewise of many others,

*This research was partially supported by GA ČR Grant No. 201/95/0976.

is that it is very time consuming and learning bigger networks can be intractable (Tesauro, 1987; Tesauro and Janssens, 1988). The effort to speed up this algorithm has, so far, brought only limited results. The intrinsic causes of these difficulties are hidden in the complexity of the learning problem. However within the framework of computational learning theory no results are known about the performance of this algorithm (Maass, 1994).

The loading problem formulated by Judd (1990) seems to be a relevant model for supervised connectionist learning of the feedforward networks. It is known that the loading problem is generally NP-complete, and many strong restrictions on architectural design and/or training tasks do not help to avoid the intractability of the loading problem (Judd, 1990; Šíma, 1994). Wiklicky (1993) extended these results and proved that the loading problem for higher order networks is even undecidable.

Judd's proofs of NP-completeness do not depend on the type of node function (the function computed by one neuron), but are based on a type of architectures which has quite irregular connections. This makes the loading problem hard because the architecture is fixed. Baum (1991) objected to this approach, referring to the fact that, in practice, we typically have control over the net architecture we are loading. Thus, we can choose one that makes our problem easier. This consideration leads to constructive learning algorithms that seek in some sense minimal architecture for a given training task (Redding, 1993).

Another approach has been introduced by Blum and Rivest (1992), who have proved that training even a 3-node neural network is NP-complete. Considering such a simple regular architecture as is usually used in practice, their proof of NP-completeness is based on the type of the node function which is the linear threshold function. Lin and Vitter (1991) extended this result even for a 2-node cascade neural network. Neural network specialists have found this result to have a low significance, objecting to the neuron function used here being discrete, while in the back-propagation algorithm a continuous one is employed.

Much effort has been given to generalize this negative result for the standard neuron function used in back-propagation — a continuous sigmoid function applied to the weighted sum of inputs — but only with partial success. This is probably caused by the technical difficulties involved in handling the continuous sigmoid function. The authors of a recent paper (DasGupta et al., 1993) have found this issue as one of the main open problems in this area. They proved the NP-completeness of training a 3-node network having piecewise linear activation function that has appeared in the fundamental literature dealing with neural networks. Höffgen (1993) proved it for the sigmoid activation function but with restriction to binary weights. On the other hand, one positive result (Macintyre and Sontag, 1993) states that the loading problem for the sigmoid activation function is decidable (if widely believed Schanuel's conjecture in number theory is true).

We generalize Blum and Rivest's technique for the standard neuron function used in back-propagation. Thus, we show that training a 3-node sigmoid neural network is NP-hard. However, our result is valid with an additional constraint on the output neuron function. To satisfy the additional condition, for example, it is sufficient to consider the output neuron function with a zero threshold value. This is a quite usual presumption (Baum, 1990). As a consequence, we find training sigmoid feedforward networks with a single hidden layer and with zero threshold of output neuron to be intractable. Assuming $P \neq NP$, this result implies that the back-propagation learning algorithm is not efficient in principle.

Finally, we transform the training problem to a special nonlinear programming problem in real domain, namely, whether there exists a solution for some system of polynomial inequalities. Consequently we obtain a NP-hardness result for this general problem.

2 Training Problem for a 3-Node Network

We consider a *3-node network* with n binary inputs $x_1, \ldots, x_n \in \{0,1\}$. These inputs are all connected to two hidden nodes N_1 and N_2. The output of the hidden nodes are connected to output node N_3, which gives the output of the network (see figure 1). Each node of a 3-node network computes a continuous sigmoidal function on the weighted sum of inputs as follows:

$$\begin{aligned}
N_1 &: \quad y_1 = \sigma(W(\vec{x})),\ W(\vec{x}) = w_0 + \sum_{i=1}^{n} w_i x_i \\
N_2 &: \quad y_2 = \sigma(V(\vec{x})),\ V(\vec{x}) = v_0 + \sum_{i=1}^{n} v_i x_i \\
N_3 &: \quad y_3 = \sigma(u_0 + u_1 y_1 + u_2 y_2)
\end{aligned}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the standard sigmoid function, and $w_0, w_1, \ldots, w_n$, $v_0, v_1, \ldots, v_n$, u_0, u_1, u_2 are real weights. This determines the so-called *network function* $y_3 : \{0,1\}^n \longrightarrow (0,1)$.

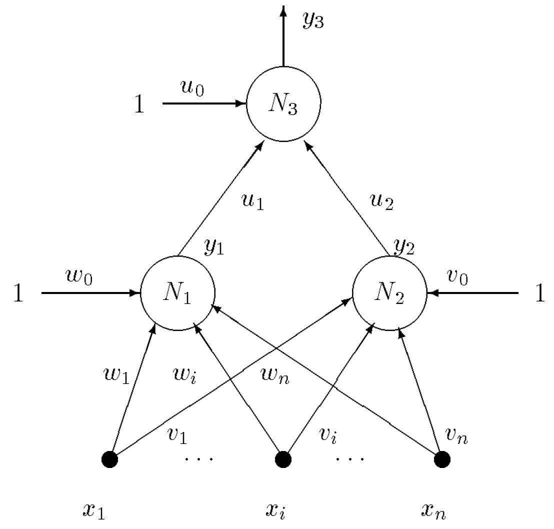

Figure 1: A 3-node network.

Our result concerning the non-efficiency of training this network holds for $u_0 = 0$. In fact, it is valid just for the following, slightly more general condition. Geometrically, it says that the line with the equation $u_0 + u_1 y_1 + u_2 y_2 = 0$ does not strictly separate any two points of $\{[0,0],[1,0],[0,1],[1,1]\}$ from the remaining two. This means that if u_0, u_0+u_1, u_0+u_2, and $u_0+u_1+u_2$, do not all have the same signs, then $u_0(u_0+u_1)(u_0+u_2)(u_0+u_1+u_2) \leq 0$. We call this constraint the *output separation condition* (OSC).

A set of training examples is given to a training algorithm. A training example is a pair composed of a n-dimensional binary input vector and a relevant desired binary output. So the training set has the form $T = \{(\vec{x}_k, d_k) \,/\, \vec{x}_k \in \{0,1\}^n, d_k \in \{0,1\}, k = 1, \ldots, t\}$. In our case, the *training problem* (TP) for a 3-node sigmoid network is given in the form of the decision question whether there exist real weights $w_0, \ldots, w_n, v_0, \ldots, v_n, u_0, u_1, u_2$ satisfying the OSC, so that this network computes a function consistent with the given instance of the training set T. Since the network function is continuous, the network output must be discretized in order to achieve the match in output values on instances from T. As it is usual in neural network applications, we define the training example $(\vec{x}, d) \in T$ to be *consistent* with the network function, if it holds that $d = 1 \leftrightarrow y_3(\vec{x}) \geq \frac{1}{2}$ (i.e. $u_0 + u_1 y_1 + u_2 y_2 \geq 0$). We can, without loss of consistency, multiply the last inequality by a sufficiently large positive number and obtain new values of u_0, u_1, u_2, so that y_3 is almost 1 (unless $u_0 + u_1 y_1 + u_2 y_2 = 0$). For $d = 0$, y_3 is almost 0.

3 Training a 3-Node Sigmoid Network Is Hard

In this section we will prove the main result implying the intractability of training a 3-node sigmoid network. For this purpose we will employ the technique of polynomial time reduction from the following *Set-Splitting problem* (SSP) that was proven to be NP-complete by Lovász (Garey and Johnson, 1979). The instance of the SSP is a pair (S, C) where S is a finite set and $C = \{c_j \,/\, c_j \subseteq S, j = 1, \ldots, r\}$ is a collection of subsets c_j of S. The question of this problem follows: Do there exist disjoint sets S_1, S_2 ($S_1 \cap S_2 = \emptyset$) such that $S = S_1 \cup S_2$ and $c_j \not\subseteq S_1$ and $c_j \not\subseteq S_2$, $j = 1, \ldots, r$? Without loss of generality assume $|c_j| \geq 2$, $j = 1, \ldots, r$.

Theorem 1 *The training problem for a 3-node sigmoid network satisfying the OSC (especially when $u_0 = 0$) is NP-hard.*

Proof: Let $S = \{s_1, \ldots, s_p\}$ and $C = \{c_j \,/\, c_j \subseteq S, j = 1, \ldots, r\}$ be an instance of SSP. We create a relevant TP instance $T = T_0 \cup T_1 \cup T_0' \cup T_1'$ with $n = 2p + 1$ inputs such that the constructed instance has a solution iff the SSP instance has a solution:

$$\begin{aligned}
T_0 &= \{(0^{i-1}10^{n-i}, 0) \,/\, i = 1, \ldots, p\} \\
T_1 &= \{(0^n, 1)\} \cup \\
&\quad \cup\{(x_{j1} \ldots x_{jp} 0^{p+1}, 1) \,/\, x_{ji} = 1 \leftrightarrow s_i \in c_j, i = 1, \ldots, p; \;\; j = 1, \ldots, r\} \\
T_0' &= \{(0^{2p}1, 0)\} \cup \\
&\quad \cup\{(0^p x_{j1} \ldots x_{jp} 1, 0) \,/\, x_{ji} = 1 \leftrightarrow s_i \in c_j, i = 1, \ldots, p; \;\; j = 1, \ldots, r\} \\
T_1' &= \{(0^{p+i-1}10^{p-i}1, 1) \,/\, i = 1, \ldots, p\}.
\end{aligned}$$

Notice that the training examples of T_0 correspond to those from T_1' except the shift and the opposit output. There is a similar correspondence between T_1 and T_0'. Moreover, the examples of T_0, T_1 differs from the ones from T_0', T_1' in the last input bit. So $t = |T| = 2(p + r + 1)$ and this construction can be done in a polynomial time with respect to the size of the SSP instance.

In order to make this formal notation clearer we illustrate it by a small example of this construction. Let $S = \{s_1, s_2, s_3\}$, $C = \{c_1, c_2\}$, $c_1 = \{s_1, s_2\}$, $c_2 = \{s_2, s_3\}$. Then the corresponding instance of TP will take the following form:

$$\begin{aligned}
T_0 &= \{(100\,000\,0, 0), (010\,000\,0, 0), (001\,000\,0, 0)\} \\
T_1 &= \{(000\,000\,0, 1), (110\,000\,0, 1), (011\,000\,0, 1)\} \\
T_0' &= \{(000\,000\,1, 0), (000\,110\,1, 0), (000\,011\,1, 0)\} \\
T_1' &= \{(000\,100\,1, 1), (000\,010\,1, 1), (000\,001\,1, 1)\}.
\end{aligned}$$

First, we prove that if the SSP instance (S, C) had a solution then the relevant TP instance T has a solution. Assume that S_1, S_2 is the solution of the SSP instance

(S, C). We choose the weights for the 3-node network function satisfying the OSC ($u_0 = 0$), so that it is consistent with T:

$$
\begin{array}{llr}
w_0 = -1, & w_i = \left\{ \begin{array}{ll} 2 & s_i \in S_1 \\ -2p+2 & s_i \notin S_1 \end{array} \right. & i = 1, \ldots, p, \\[2ex]
w_{2p+1} = 1, & w_{p+i} = -1 & i = 1, \ldots, p, \\[2ex]
v_0 = 1, & v_i = \left\{ \begin{array}{ll} -2p & s_i \in S_2 \\ 2p^2 - 2p & s_i \notin S_2 \end{array} \right. & i = 1, \ldots, p, \\[2ex]
v_{2p+1} = 2p, & v_{p+i} = -2p & i = 1, \ldots, p, \\[2ex]
u_0 = 0, & u_1 = -2, \qquad u_2 = 1. &
\end{array}
\tag{1}
$$

Our example continues in figure 2 where the 3-node network weights corresponding to the SSP solution $S_1 = \{s_1, s_3\}$, $S_2 = \{s_2\}$ are depicted.

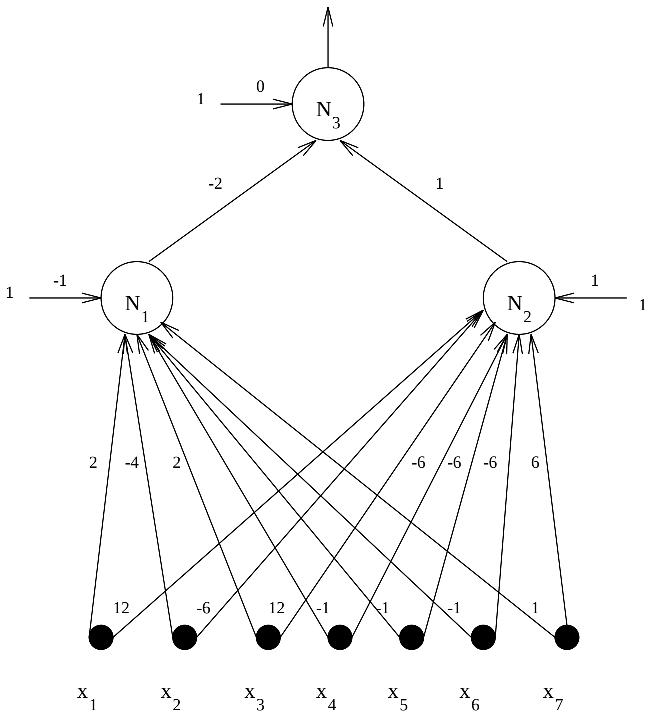

Figure 2: The TP solution for the SSP solution $S_1 = \{s_1, s_3\}$, $S_2 = \{s_2\}$.

Denote $\tau_1 = \{(0^{i-1}10^{n-i}, 0) \;/\; s_i \in S_1\}$, $\tau_2 = \{(0^{i-1}10^{n-i}, 0) \;/\; s_i \in S_2\}$ which are two disjoint sets, such that $T_0 = \tau_1 \cup \tau_2$, due to the disjoint set-splitting of $S = S_1 \cup S_2$. Then $W(\vec{x}) = 1$ (as $w_i = 2$ and $w_0 = -1$), $V(\vec{x}) = 2p^2 - 2p + 1$ (as $v_i = 2p^2 - 2p$ and $v_0 = 1$) for $(\vec{x}, 0) \in \tau_1$ and $W(\vec{x}) = -2p + 1$ (as $w_i = -2p + 2$ and $w_0 = -1$), $V(\vec{x}) = -2p + 1$ (as $v_i = -2p$ and $v_0 = 1$) for $(\vec{x}, 0) \in \tau_2$ according to the weights defined (1). On the other hand it is easy to observe that $W(\vec{x}) \leq -1$, $V(\vec{x}) \geq 1$ for $(\vec{x}, 1) \in T_1$ since either $\vec{x} = 0^n$ or $\vec{x} = \vec{x}_j 0^{p+1}$ where $\vec{x}_j = x_{j1} \ldots x_{jp}$, $j = 1, \ldots, r$ have a 1 in two positions $1 \leq i_1, i_2 \leq p$ such that $s_{i_1} \notin S_1$, $s_{i_2} \notin S_2$ because $\vec{x}_j$ corresponds to $c_j \in C$ such that $c_j \not\subseteq S_1$, $c_j \not\subseteq S_2$

since S_1, S_2 is a solution of the SSP. After applying the sigmoid σ to the nodes N_1 and N_2 we obtain the input (y_1, y_2) for the node N_3 that computes $-2y_1 + y_2 = -2\sigma(W(\vec{x})) + \sigma(V(\vec{x}))$. Remember that σ is an increasing function within the range $(0,1)$ and that $\sigma(-x) = 1 - \sigma(x)$. Then $-2y_1 + y_2 = -2\sigma(1) + \sigma(2p^2 - 2p + 1) < 0$ for $(\vec{x}, 0) \in \tau_1$ and $-2y_1 + y_2 = -2\sigma(-2p+1) + \sigma(-2p+1) = -\sigma(-2p+1) < 0$ for $(\vec{x}, 0) \in \tau_2$. On the other hand $-2y_1 + y_2 \geq -2\sigma(-1) + \sigma(1) = 1 - 3\sigma(-1) = \frac{e-2}{e+1} \geq 0$ for $(\vec{x}, 1) \in T_1$.

Similarly it follows from (1) that $W(\vec{x}) = -1$, $V(\vec{x}) = 1$ for $(\vec{x}, 1) \in T'_1$ and $W(\vec{x}) = 0$, $V(\vec{x}) = 2p + 1$ for $\vec{x} = 0^{2p}1$. This implies that $-2y_1 + y_2 = -2\sigma(-1) + \sigma(1) = \frac{e-2}{e+1} \geq 0$ for $(\vec{x}, 1) \in T'_1$ and $-2y_1 + y_2 = -2\sigma(0) + \sigma(2p+1) = -1 + \sigma(2p+1) < 0$ for $(0^{2p}1, 0) \in T'_0$. Furthermore denote by $q_j = |c_j|$, $j = 1, \ldots, r$ the number of 1's in $\vec{x}_j = x_{j1} \ldots x_{jp}$ for $(0^p \vec{x}_j 1, 0) \in T'_0$ corresponding to $c_j \in C$. Then $W(\vec{x}) = -q_j$, $V(\vec{x}) = 1 + 2p - 2pq_j$ for $(\vec{x}, 0) \in T'_0$ where $\vec{x} = 0^p \vec{x}_j 1$, $j = 1, \ldots, r$. It follows from $2 \leq q_j \leq p$ that $-1 - 2p + 2pq_j - q_j > 0$ that implies $e^{q_j}\left(1 - 2e^{-1-2p+2pq_j - q_j}\right) < 0$. Further we obtain $-2 - 2e^{-1-2p+2pq_j} + 1 + e^{q_j} < 0$. Thus $-2\sigma(-q_j) + \sigma(1 + 2p - 2pq_j) < 0$ for $(0^p \vec{x}_j 1, 0) \in T'_0$, $j = 1, \ldots, r$ as follows from the explicit form of σ. So, we have checked the 3-node sigmoid network function with above-defined weights (1), to be consistent with T. This means that there exists a solution of the relevant TP.

Conversely, we prove that if the TP instance T has a solution then the SSP instance (S, C) had a solution. Let $w_0, \ldots, w_n, v_0, \ldots, v_n, u_0, u_1, u_2$ be real weights satisfying the OSC for the 3-node network function that is consistent with the TP instance $T = T_0 \cup T_1 \cup T'_0 \cup T'_1$. Since $T_0 \cup T'_0 \neq \emptyset$, $T_1 \cup T'_1 \neq \emptyset$ and $0 < y_1, y_2 < 1$, the real numbers $u_0, u_0 + u_1, u_0 + u_2, u_0 + u_1 + u_2$ corresponding to the limit points $[0,0], [1,0], [0,1], [1,1]$ do not all have the same sign. Then it holds that $u_0(u_0 + u_1)(u_0 + u_2)(u_0 + u_1 + u_2) \leq 0$ due to the OSC.

We will restrict ourself to the case when there is just one positive number among $u_0, u_0 + u_1, u_0 + u_2, u_0 + u_1 + u_2$ and the other three numbers are non-positive. In this case we will use only the training examples from $T_0 \cup T_1$ and the corresponding weights $w_0, \ldots, w_p, v_0, \ldots, v_p, u_0, u_1, u_2$ throughout the proof. The complementary case when one number is negative and the three others are non-negative can be transformed to the previous case by using the complemantary training examples from $T'_0 \cup T'_1$. For this purpose the new weights $w'_0, \ldots, w'_p, v'_0, \ldots, v'_p, u'_0, u'_1, u'_2$ are defined:

$$w'_0 = w_0 + w_{2p+1}, \quad w'_i = w_{p+i} \quad i = 1, \ldots, p,$$

$$v'_0 = v_0 + v_{2p+1}, \quad v'_i = v_{p+i} \quad i = 1, \ldots, p,$$

$$u'_0 = -u_0, \quad u'_1 = -u_1, \quad u'_2 = -u_2.$$

Thus, there is just one positive number among $u'_0, u'_0 + u'_1, u'_0 + u'_2, u'_0 + u'_1 + u'_2$ and the other three numbers are non-positive as desired. Moreover the 3-node network function with these new weights is consistent with the training examples from $T_0 \cup T_1$ because the network function with the original weights was consistent with the corresponding $T'_0 \cup T'_1$. So the transformation is successfully accomplished.

Furthermore, without loss of generality, we assume that:

$$u_0 < 0, \ u_0 + u_1 \leq 0, \ u_0 + u_2 \leq 0, \ u_0 + u_1 + u_2 > 0. \tag{2}$$

The first inequality in (2) is strict due to the others. The other three combinations can be transformed to (2) without changing the network function, using operations preserving the function equivalence (Kůrková and Kainen, 1994). We outline such a transformation just for one case, the other ones being similar. For example let us consider that $u_0 \leq 0$, $u_0 + u_1 < 0$, $u_0 + u_2 > 0$, $u_0 + u_1 + u_2 \leq 0$. We define the new weights $w'_i = -w_i$, $v'_i = v_i$, $i = 0, \ldots, n$; $u'_0 = u_0 + u_1$, $u'_1 = -u_1$, $u'_2 = u_2$. Since

$\sigma(-x) = 1 - \sigma(x)$, the new values of the hidden neuron outputs are $y_1' = 1 - y_1$, $y_2' = y_2$. We check the network with the new weights to compute the same function:

$$u_0' + u_1'y_1' + u_2'y_2' = u_0 + u_1 - u_1(1 - y_1) + u_2y_2 = u_0 + u_1y_1 + u_2y_2.$$

The new weights also satisfy the required conditions (2):

$$\begin{aligned} u_0' &= u_0 + u_1 < 0, \\ u_0' + u_1' &= u_0 + u_1 - u_1 = u_0 \leq 0, \\ u_0' + u_2' &= u_0 + u_1 + u_2 \leq 0, \\ u_0' + u_1' + u_2' &= u_0 + u_1 - u_1 + u_2 = u_0 + u_2 > 0. \end{aligned}$$

Now we can define $S_1 = \{s_i \in S / w_i \geq 0\}$, $S_2 = S - S_1$. It is clear that $S = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$. We prove that S_1, S_2 is the solution of the SSP instance (S, C). On the contrary suppose that there is $c_j \in C$ such that either $c_j \subseteq S_1$ or $c_j \subseteq S_2$. Without loss of generality denote the elements of $c_j = \{s_1, \ldots, s_m\} \subseteq S$ where $m = q_j \leq p$. Since $w_0, \ldots, w_n, v_0, \ldots, v_n, u_0, u_1, u_2$ is the solution of the TP instance T, the training examples from $T_0 \cup T_1$ are consistent with the network function, especially $(0^{i-1}10^{n-i}, 0) \in T_0$, $i = 1, \ldots, m$; $(0^n, 1) \in T_1$, and $(1^m0^{n-m}, 1) \in T_1$ corresponding to c_j, as follows:

$$\begin{array}{llll} (i) & u_0 + u_1\sigma(w_0 + w_i) + u_2\sigma(v_0 + v_i) & < \; 0 & i = 1, \ldots, m \\ (ii) & u_0 + u_1\sigma(w_0) + u_2\sigma(v_0) & \geq \; 0 & \\ (iii) & u_0 + u_1\sigma(w_0 + \sum_{i=1}^{m} w_i) + u_2\sigma(v_0 + \sum_{i=1}^{m} v_i) & \geq \; 0. & \end{array}$$

Using the explicit form of the sigmoidal function σ and after some calculations we obtain

$$\begin{array}{llll} (i) & e^{v_0+v_i}\left(U_2 + U_3e^{w_i}\right) & < \; U_0 + U_1e^{w_i} & i = 1, \ldots, m \\ (ii) & e^{v_0}\left(U_2 + U_3\right) & \geq \; U_0 + U_1 & \\ (iii) & e^{v_0+V_m}\left(U_2 + U_3e^{W_m}\right) & \geq \; U_0 + U_1e^{W_m} & \end{array}$$

where

$$\begin{aligned} U_0 &= -u_0 > 0 \\ U_1 &= -(u_0 + u_1)e^{w_0} \geq 0 \\ U_2 &= u_0 + u_2 \leq 0 \\ U_3 &= (u_0 + u_1 + u_2)e^{w_0} > 0, \end{aligned} \tag{3}$$

and $W_k = \sum_{i=1}^{k} w_i$, $V_k = \sum_{i=1}^{k} v_i$ for $k = 1, \ldots, m$.

We know that $U_0 + U_1$, $U_0 + U_1e^{W_m}$ are both positive due to (3). Thus, it follows from (ii), (iii) that $U_2 + U_3 > 0$ and $U_2 + U_3e^{W_m} > 0$. In addition we prove $U_2 + U_3e^{w_i} > 0$, and $U_2 + U_3e^{W_i} > 0$, for $i = 1, \ldots, m$. In case that $c_j \subseteq S_1$, then $w_i \geq 0$ for all $i = 1, \ldots, m$, and we have $U_2 + U_3e^{W_i} \geq U_2 + U_3e^{w_i} \geq U_2 + U_3 > 0$, for all $i = 1, \ldots, m$. In case that $c_j \subseteq S_2$, then $w_i < 0$ for all $i = 1, \ldots, m$, and we have $U_2 + U_3e^{w_i} \geq U_2 + U_3e^{W_i} \geq U_2 + U_3e^{W_m} > 0$, for all $i = 1, \ldots, m$. This consideration enables us to express (i), (ii) in the following form:

$$e^{v_0+v_i} < \frac{U_0 + U_1e^{w_i}}{U_2 + U_3e^{w_i}} \qquad i = 1, \ldots, m \tag{4}$$

$$e^{-v_0} \leq \frac{U_2 + U_3}{U_0 + U_1}. \tag{5}$$

To derive a desired contradiction we prove that (4), (5) imply that

$$e^{v_0+V_k} < \frac{U_0 + U_1 e^{W_k}}{U_2 + U_3 e^{W_k}} \tag{6}$$

holds for all $k = 1, \ldots, m$. This is done by induction for k. For $k = 1$, (6) is exactly the first inequality of (4). Suppose that (6) holds for some $k \in \{1, \ldots, m-1\}$. Then using (4), (5) and the fact that all terms are positive we obtain

$$e^{v_0+V_{k+1}} = e^{v_0+V_k} e^{v_0+v_{k+1}} e^{-v_0} < \frac{U_0 + U_1 e^{W_k}}{U_2 + U_3 e^{W_k}} \cdot \frac{U_0 + U_1 e^{w_{k+1}}}{U_2 + U_3 e^{w_{k+1}}} \cdot \frac{U_2 + U_3}{U_0 + U_1}. \tag{7}$$

We want to prove (6) with k replaced by $k+1$. From (7) it suffices to show that

$$\frac{U_0 + U_1 e^{W_k}}{U_2 + U_3 e^{W_k}} \cdot \frac{U_0 + U_1 e^{w_{k+1}}}{U_2 + U_3 e^{w_{k+1}}} \cdot \frac{U_2 + U_3}{U_0 + U_1} \leq \frac{U_0 + U_1 e^{W_{k+1}}}{U_2 + U_3 e^{W_{k+1}}} \tag{8}$$

which can be rewritten as

$$(U_1 U_2 - U_0 U_3)\left(e^{W_k} - 1\right)\left(e^{w_{k+1}} - 1\right)\left(U_1 U_3 e^{W_{k+1}} - U_0 U_2\right) \leq 0. \tag{9}$$

The middle two terms, $e^{W_k} - 1$ and $e^{w_{k+1}} - 1$, are either both non-negative when $c_j \subseteq S_1$ or both negative when $c_j \subseteq S_2$. Since by (3) the first term is negative and the last term is non-negative, this completes the induction step. We have proven (6), especially for $k = m$, i.e. the negation of (iii), which is a contradiction and so $c_j \not\subseteq S_1$, $c_j \not\subseteq S_2$ and S_1, S_2 is the solution of the SSP instance (S, C). This completes the proof of the whole theorem. □

It is an open question whether the TP in theorem 1 remains NP-hard when we do not assume the OSC. The 'gadget' technique introduced by Blum and Rivest (1992) cannot be straightforwardly used here because it is not clear which functions, exactly, can be realized by the sigmoid network.

Corollary 1 *The training problem for sigmoid feedforward networks with one layer of a given number of hidden nodes that are connected to one output node with zero threshold, is NP-hard.*

Proof: To prove this training problem for sigmoid feedforward networks with a given number of nodes in a single hidden layer to be NP-hard, use the identical reduction from the TP for a 3-node network with zero threshold of output neuron that is NP-hard according to the previous theorem, because the latter architecture with two hidden nodes is a special case of the former one. □

4 Solving Inequalities Instead of Training

In this section the correspondence between neural network training and nonlinear programming is used to apply the negative result from section 3 for more general problems. This correspondence can be also useful for seeking more sophisticated learning heuristics that solve some special cases of related training problems where the corresponding nonlinear programming problem is effectively solvable (e.g. geometric programming, Fletcher, 1981).

Thus, theorem 1 has a more general interpretation. To decide whether there exist weights of the sigmoid 3-node network that are consistent with the given training set is in fact equivalent to decide whether there exists a solution of a special system for polynomial inequalities. Namely, the TP instance $T = \{(\vec{x}_k, d_k) \,/\, \vec{x}_k \in \{0,1\}^n, d_k \in \{0,1\}, k = 1, \ldots, t\}$ has a solution iff there exist real weights $w_0, \ldots, w_n$, $v_0, \ldots, v_n$, u_0, u_1, u_2 such that:

$$\begin{aligned}
u_0 + u_1\sigma(w_0 + \sum_{i=1}^{n} w_i x_i) + u_2\sigma(v_0 + \sum_{i=1}^{n} v_i x_i) &< 0 \quad (x_1 \ldots x_n, 0) \in T \\
u_0 + u_1\sigma(w_0 + \sum_{i=1}^{n} w_i x_i) + u_2\sigma(v_0 + \sum_{i=1}^{n} v_i x_i) &\geq 0 \quad (x_1 \ldots x_n, 1) \in T.
\end{aligned} \tag{10}$$

Using the explicit form of the sigmoidal funcion σ and after some calculations we can transform (10) to the following system of polynomial inequalities:

$$\begin{aligned}
Z_1 + Z_2 - Z_3 + Z_1 \prod_{i \in I} \xi_i + Z_2 \prod_{i \in I} \eta_i + Z_3 \prod_{i \in I} \xi_i \eta_i &< 0 \quad I \in \mathcal{J}_0 \\
Z_1 + Z_2 - Z_3 + Z_1 \prod_{i \in I} \xi_i + Z_2 \prod_{i \in I} \eta_i + Z_3 \prod_{i \in I} \xi_i \eta_i &\geq 0 \quad I \in \mathcal{J}_1 \\
\xi_0, \xi_1, \ldots, \xi_n, \eta_0, \eta_1, \ldots, \eta_n &> 0
\end{aligned} \tag{11}$$

where $\xi_i = e^{w_i}$, $\eta_i = e^{v_i}$, $i = 0, \ldots, n$; $Z_1 = u_0 + u_1$, $Z_2 = u_0 + u_2$, $Z_3 = u_0 + u_1 + u_2$ are real unknown variables and

$$\begin{aligned}
\mathcal{J}_0 &= \{0 \in I \subseteq \{0, \ldots, n\} \,/\, \text{for } (x_1 \ldots x_n, 0) \in T \ i \in I \leftrightarrow x_i = 1\} \\
\mathcal{J}_1 &= \{0 \in I \subseteq \{0, \ldots, n\} \,/\, \text{for } (x_1 \ldots x_n, 1) \in T \ i \in I \leftrightarrow x_i = 1\}
\end{aligned}$$

are collections of index sets. This consideration enables us to formulate the following theorem.

Theorem 2 *Given two disjoint non-empty sets $\emptyset \neq \mathcal{J}_0, \mathcal{J}_1 \subseteq \{0 \in I \subseteq \{0, \ldots, n\}\}$ $(\mathcal{J}_0 \cap \mathcal{J}_1 = \emptyset)$, it is NP-hard to decide whether there exist real numbers $\xi_0, \ldots, \xi_n$, $\eta_0, \ldots, \eta_n$, Z_1, Z_2, Z_3 satisfying the following system of polynomial inequalities:*

$$\begin{aligned}
Z_1 + Z_2 - Z_3 + Z_1 \prod_{i \in I} \xi_i + Z_2 \prod_{i \in I} \eta_i + Z_3 \prod_{i \in I} \xi_i \eta_i &< 0 \quad I \in \mathcal{J}_0 \\
Z_1 + Z_2 - Z_3 + Z_1 \prod_{i \in I} \xi_i + Z_2 \prod_{i \in I} \eta_i + Z_3 \prod_{i \in I} \xi_i \eta_i &\geq 0 \quad I \in \mathcal{J}_1 \\
Z_1^2 Z_2 Z_3 + Z_1 Z_2^2 Z_3 - Z_1 Z_2 Z_3^2 &\leq 0 \\
\xi_0, \ldots, \xi_n, \eta_0, \ldots, \eta_n &> 0.
\end{aligned}$$

Proof: Notice $Z_1^2 Z_2 Z_3 + Z_1 Z_2^2 Z_3 - Z_1 Z_2 Z_3^2 = (Z_1 + Z_2 - Z_3) Z_1 Z_2 Z_3 \leq 0$ is in fact the OSC because $Z_1 + Z_2 - Z_3 = u_0$, $Z_1 = u_0 + u_1$, $Z_2 = u_0 + u_2$, $Z_3 = u_0 + u_1 + u_2$ do not all have the same sign due to $\mathcal{J}_0, \mathcal{J}_1 \neq \emptyset$. Then, the proposition follows from the theorem 1 and the equivalence of (10) and (11). □

There is still an open question whether the problem from theorem 2 is NP-hard when we remove the inequality $Z_1^2 Z_2 Z_3 + Z_1 Z_2^2 Z_3 - Z_1 Z_2 Z_3^2 \leq 0$. This issue is important for the neural network training interpretation because this determines whether the training problem remains NP-hard with no additional constraint (like the OSC). On the other hand we obtain a more regular system of inequalities in the case when $Z_1 + Z_2 - Z_3 = u_0 = 0$:

Theorem 3 *Given two disjoint non-empty sets* $\emptyset \neq \mathcal{J}_0, \mathcal{J}_1 \subseteq \{0 \in I \subseteq \{0,\ldots,n\}\}$ *(*$\mathcal{J}_0 \cap \mathcal{J}_1 = \emptyset$*), it is NP-hard to decide whether there exist real numbers* $\xi_0,\ldots,\xi_n$*,* $\eta_0,\ldots,\eta_n$*,* u_1, u_2 *satisfying the following system of polynomial inequalities:*

$$\begin{aligned} u_1 \prod_{i\in I} \xi_i + u_2 \prod_{i\in I} \eta_i + (u_1+u_2) \prod_{i\in I} \xi_i\eta_i &< 0 \quad I \in \mathcal{J}_0 \\ u_1 \prod_{i\in I} \xi_i + u_2 \prod_{i\in I} \eta_i + (u_1+u_2) \prod_{i\in I} \xi_i\eta_i &\geq 0 \quad I \in \mathcal{J}_1 \\ \xi_0,\ldots,\xi_n,\eta_0,\ldots,\eta_n &> 0. \end{aligned}$$

Proof: It follows from theorem 1 for $u_0 = 0$ and from the equivalence of (10) and (11). □

The theorems 2, and 3 establish new NP-hardness results concerning special nonlinear programming problems defined on the real domains. Consequently, any task that involves these special problems is intractable.

5 Conclusions

In the present paper we have proved that training even the 3-node sigmoid network is NP-hard which was reported by several researchers to be an open problem. For this purpose the technique by Blum and Rivest has been generalized for the standard sigmoid activation function. This is a relevant result for neural networks learning because most of the practical applications employ the neural learning heuristics back-propagation which uses this activation function. It also confirms the practical experience that shows this algorithm to be not efficient for larger tasks. The result holds only for fixed architectures. Thus the possible way out of this situation could be a constructive learning that allows a learning algorithm to choose or to change the topology of the network. We also believe that the initial setting of weights in the neural network plays an important role in the neural learning complexity. Further, the training problem has been interpreted as a special nonlinear programming problem that helps to understand the learning issue better. Consequently, the intractability results have been obtained for this nonlinear programming task.

Acknowledgement

I am grateful to Jiří Wiedermann for many valuable comments that helped me to complete this paper. Further I thank Michael Schmitt and the anonymous referee who found a mistake in the preliminary version of the paper.

References

Baum, E.B. 1990. A Polynomial Time Algorithm That Learns Two Hidden Unit Nets. Neural Computation, 2 (4), 510-522.

Baum, E.B. 1991. Book Reviews. IEEE Transactions on Neural Networks, 2 (1), 181-182.

Blum, A.L., Rivest, R.L. 1992. Training a 3-Node Neural Network Is NP-Complete. Neural Networks, 5 (1), 117-127.

DasGupta, B., Siegelmann, H.T., Sontag, E.D. 1993. On the Complexity of Training Neural Networks with Continuous Activation Functions. Technical Report # 93-61, Department of Computer Science, University of Minesota.

Fletcher, R. 1981. Practical Methods of Optimization. Vol. 2: Constrained optimization. A Wiley-Interscience Publication.

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

Höffgen, K.-U. 1993. Computational Limitations on Training Sigmoid Neural Networks. Information Processing Letters (Elsevier), 46, 269-274.

Judd, J.S. 1990. Neural Network Design and the Complexity of Learning. The MIT Press, Cambridge.

Kůrková, V., Kainen, P.C. 1994. Functionally Equivalent Feedforward Neural Networks. Neural Computation, 6 (3), 543-558.

Lin J.-H., Vitter J.S. 1991. Complexity Results on Learning by Neural Nets. Machine Learning, 6, 211-230.

Maass, W. 1994. Perspectives of Current Research about the Complexity of Learning on Neural Nets. In: Theoretical Advances in Neural Computation and Learning, Kluwer Academic Publishers, 295-336.

Macintyre, A., Sontag, E.D. 1993. Finiteness Results for Sigmoidal "Neural" Networks. In Proc. 25th ACM Symp. Theory of Computing, San Diego, 325-334.

Redding, N.J., Kowalczyk, A., Downs T. 1993. Constructive Higher-Order Network Algorithm That Is Polynomial Time. Neural Networks, 6 (7), 997-1010.

Rumelhart, D.E., Hinton, G.E., Williams, R.J. 1986. Learning Representations by Back-Propagating Errors. Nature, 323, 533-536.

Šíma, J. 1994. Loading Deep Networks is Hard. Neural Computation, 6 (5), 842-850.

Tesauro, G. 1987. Scaling Relationships in Back-Propagation Learning: Dependence on Training Set Size. Complex Systems, 1, 367-372.

Tesauro, G., Janssens, B. 1988. Scaling Relationships in Back-Propagation Learning. Complex Systems, 2, 39-44.

Wiklicky, H. 1994. The Neural Network Loading Problem is Undecidable. In Proc. Euro-Colt'93 (J. Shawe-Taylor and M. Anthony, eds.), Clarendon Press, Oxford, 183-192.