# On the Complexity of Training a Single Perceptron with Programmable Synaptic Delays

Jiří Šíma*

Department of Theoretical Computer Science,
Institute of Computer Science, Academy of Sciences of the Czech Republic,
P. O. Box 5, 182 07 Prague 8, Czech Republic, sima@cs.cas.cz

**Abstract.** We consider a single perceptron $\mathcal{N}$ with synaptic delays which generalizes a simplified model for a spiking neuron where not only the time that a pulse needs to travel through a synapse is taken into account but also the input firing rates may have more different levels. A synchronization technique is introduced so that the results concerning the learnability of spiking neurons with binary delays also apply to $\mathcal{N}$ with arbitrary delays. In particular, the consistency problem for $\mathcal{N}$ with programmable delays and its approximation version prove to be NP-hard. It follows that the perceptrons with programmable synaptic delays are not properly PAC-learnable and the spiking neurons with arbitrary delays do not allow robust learning unless $RP = NP$. In addition, we show that the representation problem for $\mathcal{N}$ which is an issue whether an $n$-variable Boolean function given in DNF (or as a disjunction of $O(n)$ threshold gates) can be computed by a spiking neuron is co-NP-hard.

## 1 Perceptrons with Synaptic Delays

Neural networks establish an important class of learning models that are widely applied in practical applications to solving artificial intelligence tasks [12]. We consider only a single (perceptron) neuron $\mathcal{N}$ having $n$ analog *inputs* that are encoded by *firing rates* $x_1, \ldots, x_n \in [-1, 1]$. Here the input values are normalized but any bounded domain $[-a, a]$ for a positive real $a \in \mathbb{R}^+$ can replace $[-1, 1]$ without loss of generality [21]. As usual each input $i$ ($1 \leq i \leq n$) is associated with a real synaptic *weight* $w_i \in \mathbb{R}$. In addition, $\mathcal{N}$ receives $i$'s analog input in the form of a unit-length rectangular *pulse (spike)* of height $|x_i|$ (for $x_i < 0$ upside down). This pulse travels through $i$'s synapse in continuous time producing a synaptic time *delay* $d_i \in \mathbb{R}_0^+$ which represents a nonnegative real parameter individual for each input $1 \leq i \leq n$. Taking these delays into account, current $i$'s input $x_i(t) \in [-1, 1]$ to $\mathcal{N}$ at continuous time $t \geq 0$ can be expressed as

$$x_i(t) = \begin{cases} x_i & \text{for } t \in D_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

* Research partially supported by project LN00A056 of The Ministry of Education of the Czech Republic.

where $D_i = [d_i, d_i + 1)$ is a time interval of the unit length during which $\mathcal{N}$ is influenced by the spike from input $i$. This determines the real *excitation*

$$\xi(t) = w_0 + \sum_{i=1}^{n} w_i x_i(t) \tag{2}$$

for $\mathcal{N}$ at time instant $t \geq 0$ as a weighted sum of current inputs including a real *bias* weight $w_0 \in \mathbb{R}$.

The real output $y(t) \in \mathbb{R}$ of $\mathcal{N}$ at continuous time $t \geq 0$ is computed by applying an *activation function* $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$ to the excitation, i.e.

$$y(t) = \sigma(\xi(t)) \,. \tag{3}$$

For *binary* outputs $y(t) \in \{0, 1\}$ the *Heaviside* activation function

$$\sigma(\xi) = \begin{cases} 1 & \text{for } \xi \geq 0 \\ 0 & \text{for } \xi < 0 \end{cases} \tag{4}$$

is usually employed. In this case, the output protocol can be defined so that $\mathcal{N}$ with weights $w_0, \ldots, w_n$ and delays $d_1, \ldots, d_n$ computes a *neuron function* $y_{\mathcal{N}} : [-1, 1]^n \longrightarrow \{0, 1\}$ defined for every input $x_1, \ldots, x_n \in [-1, 1]^n$ as $y_{\mathcal{N}}(x_1, \ldots, x_n) = 1$ iff there exists a time instant $t \geq 0$ such that $y(t) = 1$. Similarly, the *logistic sigmoid*

$$\sigma_L(\xi) = \frac{1}{1 + e^{-\xi}} \,, \tag{5}$$

which is well-known from the *back-propagation* learning [26] produces analog outputs $y(t) \in [0, 1]$ whereas the output protocol can specify a time instant $t_{out} \geq 0$ when the resulting output is read, that is $y_{\mathcal{N}}(x_1, \ldots, x_n) = y(t_{out})$. Unless otherwise stated we assume that neuron $\mathcal{N}$ employs the Heaviside activation function (4).

By restricting certain parameters in the preceding definition of $\mathcal{N}$ we obtain several computational units which are widely used in neurocomputing. For the classical *perceptrons* [25] all synaptic delays are zero, i.e. $d_i = 0$ for $i = 1, \ldots, n$, and also $t_{out} = 0$ when the logistic sigmoid (5) is employed [26]. Or assuming the spikes with a uniform firing rate, e.g. $x_i \in \{0, 1\}$ for $i = 1, \ldots, n$, neuron $\mathcal{N}$ coincides with a simplified model of a *spiking neuron* with binary coded inputs which was introduced and analyzed in [20]. Hence, the computational power of $\mathcal{N}$ computing the Boolean functions is the same as that of the spiking neuron with binary coded inputs [27] (cf. Section 4). In addition, the VC-dimension $\Theta(n \log n)$ of the spiking neuron still applies to $\mathcal{N}$ with $n$ analog inputs as can easily be verified by following the argument in [20]. From this point of view, $\mathcal{N}$ represents generalization of the spiking neuron in which the temporal delays are combined with the firing rates of perceptron units.

It follows that biological motivations for spiking neurons [10, 19] partially apply also to neuron $N$. For example, it is known that the synaptic delays are tuned in biological neural systems through a variety of mechanisms. On the other hand, the underlying computational model is still sufficiently simple providing easy silicon implementation in pulsed VLSI [19].

In this paper we deal with the computational complexity of training a single neuron $\mathcal{N}$ with programmable synaptic delays. The article is organized as follows. In Section 2, the so-called consistency problem proves to be NP-hard for $\mathcal{N}$, which implies that the perceptrons with delays are not properly PAC-learnable unless $RP = NP$. Furthermore, it is shown in Section 3 that even the approximate training can be hard for $\mathcal{N}$ with binary firing rates, which means that the spiking neurons with binary coded inputs do not allow robust learning if $RP \neq NP$. In addition, the representation problem for spiking neurons is proved to be co-NP-hard in Section 4.

## 2 A Single Perceptron with Delays Is Not Learnable

The computational complexity of training a neuron can be analyzed by using the *consistency (loading) problem* [17] which is the problem of finding the neuron parameters for a training task so that the neuron function is perfectly consistent with all training data. For example, an efficient algorithm for the consistency problem is required within the proper PAC learning framework [5] besides the polynomial VC-dimension that common neural network models usually possess [3, 24, 29]. Therefore, several learning heuristics have been proposed for networks of spiking neurons, e.g. *spike-propagation* [6]. On the other hand, NP-hardness of this problem implies that the neuron is not *properly PAC learnable* (i.e. for any training data that can be loaded into the neuron) under generally accepted complexity-theoretic assumption $RP \neq NP$ [22]. An almost exhaustive list of such NP-hardness results for feedforward perceptron networks was presented in [28].

We define a *training set*

$$T = \{(\boldsymbol{x}_k; b_k);\ \boldsymbol{x}_k = (x_{k1}, \ldots, x_{kn}) \in [-1, 1]^n,\ b_k \in \{0, 1\},\ k = 1, \ldots, m\} \quad (6)$$

containing $m$ *training examples*, each composed of $n$-dimensional input $\boldsymbol{x}_k$ from $[-1, 1]^n$ labeled with the desired scalar output value $b_k$ from $\{0, 1\}$ corresponding to *negative* and *positive examples*. The decision version for the consistency problem is formulated as follows:

**Consistency Problem for Neuron** $\mathcal{N}$ (CPN)
*Instance:* A training set $T$ for $\mathcal{N}$ having $n$ inputs.
*Question:* Are there weights $w_0, \ldots, w_n$ and delays $d_1, \ldots, d_n$ for $\mathcal{N}$ such that $y_{\mathcal{N}}(\boldsymbol{x}) = b$ for every training example $(\boldsymbol{x}; b) \in T$?

For ordinary perceptrons with *zero delays*, i.e. $d_i = 0$ for $i = 1, \ldots, n$, the consistency problem is solvable in polynomial time by linear programming although this problem restricted to binary weights is NP-complete [22]. However, already for *binary delays* $d_i \in \{0, 1\}$ the consistency problem becomes NP-complete, even for spiking neurons having binary firing rates $x_i \in \{0, 1\}$ and fixed weights [20]. This implies that neuron $\mathcal{N}$ with binary delays is not properly PAC learnable unless $RP = NP$. The result generalizes also to *bounded delay* values $d_i \in \{0, 1, \ldots, c\}$ for fixed $c \geq 2$. For the spiking neurons with unbounded delays, however, NP-hardness of the consistency problem was listed among open problems [20].

In this section we prove that the consistency problem is NP-hard for a single perceptron $\mathcal{N}$ with *arbitrary delays*, which partially answers the previous open question provided that several levels of firing rates are allowed. For this purpose a *synchronization technique* is introduced whose main idea can be described as follows. The consistency of negative example $(x_1, \ldots, x_n; 0)$ means that for every subset of inputs $I \subseteq \{1, \ldots, n\}$ whose spikes may simultaneously influence $\mathcal{N}$ (i.e. $\cap_{i \in I} D_i \neq \emptyset$) a corresponding excitation must satisfy $w_0 + \sum_{i \in I} w_i x_i < 0$. At the same time, by using the consistency of other (mostly positive) training examples we can enforce $w_0 + \sum_{i \in J} w_i x_i \geq 0$ for some $J \subseteq \{1, \ldots, n\}$. In this way we ensure that $\mathcal{N}$ is not simultaneously influenced by the spikes from inputs $J$, that is $\cap_{i \in J} D_i = \emptyset$ which is then exploited for the synchronization of the input spikes.

**Theorem 1.** *CPN is NP-hard.*

*Proof.* In order to achieve the NP-hardness result, the following variant of the *set splitting* problem which is known to be NP-complete [9] will be reduced to CPN in polynomial time.

**3Set-Splitting Problem** (3SSP)
*Instance:* A finite set $S = \{s_1, \ldots, s_p\}$ and a collection $C = \{c_\ell \subseteq S\,; |c_\ell| = 3,\ \ell = 1, \ldots, r\}$ of three-element subsets $c_\ell$ of $S$.
*Question:* Is there a partition of $S$ into two disjoint subsets $S_1$ and $S_2$, i.e. $S = S_1 \cup S_2$ where $S_1 \cap S_2 = \emptyset$, such that $c_\ell \not\subseteq S_1$ and $c_\ell \not\subseteq S_2$ for every $\ell = 1, \ldots, r$?

The 3SSP problem was also used for proving the result restricted to binary delays [20]. The above-described synchronization technique generalizes the proof to arbitrary delays.

Given a 3SSP instance $S, C$, we construct a training set $T$ for neuron $\mathcal{N}$ with $n$ inputs where $n = 2p+2$. The input firing rates of training examples exploit only seven levels from $\left\{-1, -\frac{1}{4}, -\frac{1}{8}, 0, \frac{3}{8}, \frac{3}{4}, 1\right\} \subseteq [-1, 1]$. A list of training examples which are included in training set $T$ follows:

$$(0, \ldots, 0, \underset{\substack{\uparrow \\ 2i-1}}{\tfrac{3}{4}}, 0, \ldots, 0\,; 1) \qquad \text{for } i = 1, \ldots, p\,, \tag{7}$$

$$(0, \ldots, 0, \underset{\substack{\uparrow \\ 2i}}{-\tfrac{1}{4}}, 0, \ldots, 0\,; 1) \qquad \text{for } i = 1, \ldots, p\,, \tag{8}$$

$$(0,\ldots,0,\underset{\substack{\uparrow\\ 2i-1}}{\tfrac{3}{8}},\underset{\substack{\uparrow\\ 2i}}{-\tfrac{1}{8}},0,\ldots,0\,;\,0)\qquad \text{for } i=1,\ldots,p\,, \tag{9}$$

$$(0,\ldots,0,\underset{\substack{\uparrow\\ 2p+1}}{-\tfrac{1}{4}},0\,;\,1)\,, \tag{10}$$

$$(0,\ldots,0,\underset{\substack{\uparrow\\ 2p+2}}{-\tfrac{1}{4}}\,;\,1)\,, \tag{11}$$

$$(0,\ldots,0,\underset{\substack{\uparrow\\ 2p+1}}{-\tfrac{1}{8}},\underset{\substack{\uparrow\\ 2p+2}}{-\tfrac{1}{8}}\,;\,0)\,, \tag{12}$$

$$(0,\ldots,0,\underset{\substack{\uparrow\\ 2i-1}}{1},0,\ldots,0\,;\,1)\qquad \text{for } i=1,\ldots,p\,, \tag{13}$$

$$(0,\ldots,0,\underset{\substack{\uparrow\\ 2i-1}}{1},0,\ldots,0,\underset{\substack{\uparrow\\ 2p+1}}{1},\underset{\substack{\uparrow\\ 2p+2}}{1}\,;\,0)\qquad \text{for } i=1,\ldots,p\,, \tag{14}$$

$$(0,\ldots,0,\underset{\substack{\uparrow\\ 2i}}{-1},0,\ldots,0\,;\,1)\qquad \text{for } i=1,\ldots,p\,, \tag{15}$$

$$(0,\ldots,0,\underset{\substack{\uparrow\\ 2i}}{-1},0,\ldots,0,\underset{\substack{\uparrow\\ 2p+1}}{1},\underset{\substack{\uparrow\\ 2p+2}}{1}\,;\,0)\qquad \text{for } i=1,\ldots,p\,, \tag{16}$$

and

$$(0,\ldots,0,\underset{\substack{\uparrow\\ 2i-1}}{1},\underset{\substack{\uparrow\\ 2i}}{1},0,\ldots,0,\underset{\substack{\uparrow\\ 2j-1}}{1},\underset{\substack{\uparrow\\ 2j}}{1},0,\ldots,0,\underset{\substack{\uparrow\\ 2k-1}}{1},\underset{\substack{\uparrow\\ 2k}}{1},0,\ldots,0\,;\,0) \tag{17}$$

for each $c_\ell = \{s_i, s_j, s_k\} \in C$ $(1 \le \ell \le r)$.
The number of training examples is $|T| = 7p + r + 3$, and hence, the construction of $T$ can be done in polynomial time in terms of the size of $S, C$.

Now, the correctness of the reduction will be verified, i.e. it will be shown that the 3SSP instance has a solution iff the corresponding CPN instance is solvable. So first assume that there exists a solution $S_1, S_2$ of the 3SSP instance. Define the weights and delays for $\mathcal{N}$ as follows:

$$w_0 = -1\,, \qquad w_{2p+1} = w_{2p+2} = -4\,, \tag{18}$$

$$w_{2i-1} = 2\,, \qquad w_{2i} = -4 \qquad \text{for } i = 1,\ldots,p\,, \tag{19}$$

$$d_{2i-1} = \begin{cases} 0 & \text{for } s_i \in S_1 \\ 1 & \text{for } s_i \in S_2 \end{cases}, \qquad d_{2i} = 1 - d_{2i-1} \qquad \text{for } i = 1,\ldots,p\,, \tag{20}$$

$$d_{2p+1} = 0 \qquad d_{2p+2} = 1\,. \tag{21}$$

Clearly,

$$D_{2i-1} \cap D_{2i} = \emptyset \tag{22}$$

for $i = 1, \ldots, p+1$ according to (20) and (21). It can easily be checked that $\mathcal{N}$ with parameters (18)–(21) is consistent with training examples (7)–(16). For instance, for any positive training example (7), excitation $\xi(t) = -1 + 2 \cdot \frac{3}{4} \geq 0$ when $t \in D_{2i-1}$, which is sufficient for $\mathcal{N}$ to output 1. Or for any negative training example (9), excitation $\xi(t) = -1 + 2 \cdot \frac{3}{8} < 0$ for all $t \in D_{2i-1}$ and $\xi(t) = -1 - 4 \cdot (-\frac{1}{8}) < 0$ for all $t \in D_{2i}$, whereas $\xi(t) = -1 < 0$ for $t \geq 2$, which implies that $\mathcal{N}$ outputs desired 0. The verification for the remaining training examples (7)–(16) is similar. Furthermore, $D_{2i-1} \cap D_{2j-1} \cap D_{2k-1} = \emptyset$ holds for any $c_\ell = \{s_i, s_j, s_k\} \in C$ according to (20) since $c_\ell \not\subseteq S_1$. Hence, for a negative training example (17) corresponding to $c_\ell$ excitation $\xi(t) \leq -1+2\cdot 1+2\cdot 1-4\cdot 1 < 0$ for every $t \geq 0$ due to (22) which produces zero output. This completes the argument for the CPN instance to be solvable.

On the other hand, assume that there exist weights $w_0, \ldots, w_n$ and delays $d_1, \ldots, d_n$ for $\mathcal{N}$ such that $\mathcal{N}$ is consistent with training examples (7)–(17). Any consistent negative example ensures

$$w_0 < 0 \tag{23}$$

since the excitation must satisfy $\xi(t) < 0$ also for $t \notin \cup_{i=1}^{n} D_i$. Hence, it follows from (7) and (8) that $w_0 + \frac{3}{4} w_{2i-1} \geq 0$ and $w_0 - \frac{1}{4} w_{2i} \geq 0$, respectively, which sums up to

$$w_0 + \frac{3}{8} w_{2i-1} - \frac{1}{8} w_{2i} \geq 0 \qquad \text{for } i = 1, \ldots, p\,. \tag{24}$$

On the other hand, by comparing inequality (24) with the consistency of negative examples (9) we conclude that

$$D_{2i-1} \cap D_{2i} = \emptyset \qquad \text{for } i = 1, \ldots, p\,. \tag{25}$$

Similarly, positive training examples (10) and (11) compel inequality

$$w_0 - \frac{1}{8} w_{2p+1} - \frac{1}{8} w_{2p+2} \geq 0 \tag{26}$$

which implies

$$D_{2p+1} \cap D_{2p+2} = \emptyset \tag{27}$$

when the consistency of negative example (12) is required.

Furthermore, positive training examples (13) ensure

$$w_0 + w_{2i-1} \geq 0 \qquad \text{for } i = 1, \ldots, p \tag{28}$$

which, confronted with the consistency of negative examples (14), implies

$$D_{2i-1} \subseteq D_{2p+1} \cup D_{2p+2} \qquad \text{for } i = 1, \ldots, p\,. \tag{29}$$

Similarly, the simultaneous consistency of positive examples (15) and negative examples (16) gives

$$D_{2i} \subseteq D_{2p+1} \cup D_{2p+2} \qquad \text{for } i = 1, \ldots, p\,. \tag{30}$$

It follows from (25), (27), (29), and (30) that for each $1 \leq i \leq p$

$$\begin{array}{ll} \text{either} & (D_{2i-1} = D_{2p+1} \text{ and } D_{2i} = D_{2p+2}) \\ \text{or} & (D_{2i-1} = D_{2p+2} \text{ and } D_{2i} = D_{2p+1})\ , \end{array} \tag{31}$$

which represents the synchronization of the input spikes according to (27).

Thus define the splitting of $S = S_1 \cup S_2$ as

$$S_1 = \{s_i \in S\,;\, D_{2i-1} = D_{2p+1}\}\ , \qquad S_2 = S \setminus S_1\,. \tag{32}$$

It will be proved that $S_1, S_2$ is a solution of the 3SSP. On the contrary, assume that there is $c_\ell = \{s_i, s_j, s_k\} \in C$ such that $c_\ell \subseteq S_1$ or $c_\ell \subseteq S_2$. According to definition (32), $D_{2i-1} = D_{2j-1} = D_{2k-1} = D_{2p+1}$ holds for $c_\ell \subseteq S_1$. Hence, the consistency of a corresponding negative example (17) would require

$$w_0 + w_{2i-1} + w_{2j-1} + w_{2k-1} < 0 \tag{33}$$

due to (25), but inequalities (23) and (28) imply the opposite. Similarly, $D_{2i-1} = D_{2j-1} = D_{2k-1} = D_{2p+2}$ for $c_\ell \subseteq S_2$ because of (32) and (31) providing contradiction (33). This completes the proof that the 3SSP instance is solvable. □

**Corollary 2.** *If $RP \neq NP$, then a single perceptron $\mathcal{N}$ with programmable synaptic delays is not properly PAC-learnable.*

## 3 A Spiking Neuron Does Not Allow Robust Learning

A single perceptron $\mathcal{N}$ with delays can compute only very simple neuron functions. Therefore the consistency problem introduced in Section 2 does not frequently have a solution: there are no weight and delay parameters such that the neuron function is consistent with all training data. In this case, one would be satisfied with a good approximation in practice, that is with the neuron parameters yielding a small training error. For example, in the incremental learning algorithms (e.g. [8]) that adapt single neurons before these are wired to a neural network, an efficient procedure for minimizing the training error is crucial to keep the network size small for successful generalization. Thus the decision version for the approximation problem is formulated as follows:

**Approximation Problem for Neuron** $\mathcal{N}$ (APN)
*Instance:* A training set $T$ for $\mathcal{N}$ and a positive integer $k$.
*Question:* Are there weights $w_0, \ldots, w_n$ and delays $d_1, \ldots, d_n$ for $\mathcal{N}$ such that $y_{\mathcal{N}}(\boldsymbol{x}) \neq b$ for at most $k$ training examples $(\boldsymbol{x}; b) \in T$?

Within the PAC framework, the NP-hardness of this problem implies that the neuron does not allow *robust learning* (i.e. probably approximately optimal learning for any training task) unless $RP = NP$ [14].

For the perceptrons with *zero delays*, the complexity of the approximation problem has been resolved. Several authors proved that the approximation problem is NP-complete in this case [14, 23] even if the bias is assumed to be zero [2, 16]. This means that the perceptrons with zero delays do not allow robust learning unless $RP = NP$. In addition, it is NP-hard to achieve a fixed error that is a constant multiple of the optimum [4]. These results were also generalized to analog outputs, e.g. for the logistic sigmoid (5) it is NP-hard to minimize the training error under the $L_1$ [15] or $L_2$ [28] norm within a given absolute bound or within 1 of its infimum.

In this section the approximation problem is proved to be NP-hard for perceptron $\mathcal{N}$ with *arbitrary delays*. The proof exploits only binary firing rates, which means the result is also valid for spiking neurons with binary coded inputs.

**Theorem 3.** *APN for $\mathcal{N}$ with binary firing rates is NP-hard.*

*Proof.* The following *vertex cover* problem that is known to be NP-complete [18] will be reduced to APN in polynomial time:

**Vertex Cover Problem** (VCP)
*Instance:* A graph $G = (V, E)$ and a positive integer $k \leq |V|$.
*Question:* Is there a *vertex cover* $U \subseteq V$ of size at most $k \geq |U|$ vertices, that is for each edge $\{u, v\} \in E$ at least one of $u$ and $v$ belongs to $U$?

A similar reduction was originally used for the NP-hardness result concerning the approximate training an ordinary perceptron with zero synaptic delays [14]. The technique generalizes for arbitrary delays.

Thus given a VCP instance $G = (V, E), k$ with $n = |V|$ vertices $V = \{v_1, \ldots, v_n\}$ and $r = |E|$ edges we construct a training set $T$ for neuron $\mathcal{N}$ with $n$ inputs. Training set $T$ contains the following $m = n + r$ examples:

$$(0,\ldots,0,\underset{\underset{i}{\uparrow}}{1},0,\ldots,0\,;\,1) \qquad \text{for } i = 1, \ldots, n\,, \tag{34}$$

$$(0,\ldots,0,\underset{\underset{i}{\uparrow}}{1},0,\ldots,0,\underset{\underset{j}{\uparrow}}{1},0,\ldots,0\,;\,0) \qquad \text{for each } \{v_i, v_j\} \in E\,, \tag{35}$$

which can be constructed in polynomial time in terms of the size of the VCP instance. Moreover, in this APN instance at most $k$ inconsistent training examples are allowed.

It will be shown that the VCP instance has a solution iff the corresponding APN instance is solvable. So first assume that there exists a vertex cover $U \subseteq V$

of size at most $k \geq |U|$ vertices. Define the weights and delays for $\mathcal{N}$ as follows:

$$w_0 = -1\,, \tag{36}$$

$$w_i = \begin{cases} -1 & \text{if } v_i \in U \\ 1 & \text{if } v_i \notin U \end{cases} \qquad \text{for } i = 1, \ldots, n\,, \tag{37}$$

$$d_i = 0 \qquad \text{for } i = 1, \ldots, n\,. \tag{38}$$

Obviously, negative examples (35) corresponding to edges $\{v_i, v_j\} \in E$ produce excitations either $\xi(t) = -3$ when both endpoints in $U$ or $\xi(t) = -1$ when only one endpoint in $U$, for $t \in [0, 1)$, while $\xi(t) = w_0 = -1$ for $t \geq 1$, which means $\mathcal{N}$ outputs desired 0. Furthermore, the positive examples (34) that correspond to vertices $v_i \notin U$ give excitations $\xi(t) = 0$ for $t \in [0, 1)$ and hence $\mathcal{N}$ classifies them correctly. On the other hand, $\mathcal{N}$ is not consistent with the positive examples (34) corresponding to vertices $v_i \in U$ since $\xi(t) = -2$ for $t \in [0, 1)$ and $\xi(t) = -1$ for $t \geq 1$. Nevertheless, the size of vertex cover $U$ is at most $k$, which also upper-bounds the number of inconsistent training examples. This completes the argument for the APN instance to be solvable.

On the other hand, assume that there exist weights $w_0, \ldots, w_n$ and delays $d_1, \ldots, d_n$ making $\mathcal{N}$ consistent with all but at most $k$ training examples (34)–(35). Define $U \subseteq V$ so that $U$ contains vertex $v_i$ for each *inconsistent* positive example (34) corresponding to $v_i$. In addition, $U$ includes just one of $v_i$ and $v_j$ (chosen arbitrarily) for each *inconsistent* negative example (35) corresponding to edge $\{v_i, v_j\}$. Clearly, $|U| \leq k$ since there are at most $k$ inconsistent training examples. It will be proved that $U$ is a vertex cover for $G$. On the contrary, assume that there is an edge $\{v_i, v_j\} \in E$ such that $v_i, v_j \notin U$. It follows from the definition of $U$ that $\mathcal{N}$ is consistent with the negative example (35) corresponding to edge $\{v_i, v_j\}$, which implies

$$\xi(t) = w_0 < 0 \qquad \text{for } t \notin D_i \cup D_j\,, \tag{39}$$

and it is consistent with the positive examples (34) corresponding to vertices $v_i, v_j$, which ensures

$$\xi(t) = w_0 + w_i \geq 0 \qquad \text{for } t \in D_i \tag{40}$$

$$\xi(t) = w_0 + w_j \geq 0 \qquad \text{for } t \in D_j \tag{41}$$

because of (39). By summing inequalities (39)–(41), we obtain

$$w_0 + w_i + w_j > 0\,. \tag{42}$$

On the other hand, by comparing inequalities (40) and (41) with the consistency of the negative example (35) corresponding to edge $\{v_i, v_j\}$ we conclude that $D_i = D_j$ (synchronization technique) and hence

$$\xi(t) = w_0 + w_i + w_j < 0 \qquad \text{for } t \in D_i = D_j\,, \tag{43}$$

which contradicts inequality (42). This completes the proof that $U$ is a solution of VCP. □

**Corollary 4.** *If $RP \neq NP$, then a single spiking neuron with binary coded inputs and arbitrary delays does not allow robust learning.*

## 4 The Representation Problem for Spiking Neurons

In this section we deal with the *representation (membership) problem* for the spiking neurons with binary coded inputs:

**Representation Problem for Spiking Neuron** $\mathcal{N}$ (RPN)
*Instance:* A Boolean function $f$ in DNF (disjunctive normal form).
*Question:* Is $f$ computable by a single spiking neuron $\mathcal{N}$, i.e. are there weights $w_0, \ldots, w_n$ and delays $d_1, \ldots, d_n$ for $\mathcal{N}$ such that $y_{\mathcal{N}}(\boldsymbol{x}) = f(\boldsymbol{x})$ for every $\boldsymbol{x} \in \{0,1\}^n$?

The representation problem for perceptrons with zero delays, known as the *linear separability* problem, was proved to be co-NP-complete [13]. We generalize the co-NP-hardness result for spiking neurons with arbitrary delays. On the other hand, the RPN is clearly in $\Sigma_2^p$ whereas its hardness for $\Sigma_2^p$ (or for NP) which would imply [1] that the spiking neurons with arbitrary delays are not learnable with membership and equivalence queries (unless $NP = co-NP$) remains an open problem.

Moreover, it was shown [20] that the class of $n$-variable Boolean functions computable by spiking neurons is *strictly* contained in the class DLLT that consists of functions representable as disjunctions of $O(n)$ *Boolean linear threshold functions* over $n$ variables (from the class LT containing functions computable by threshold gates) where the smallest number of threshold gates is called the *threshold number* [11]. For example, class DLLT corresponds to two-layer networks with linear number of hidden perceptrons (with zero delays) and one output OR gate. It was shown [27] that the threshold number of spiking neurons with $n$ inputs is at most $n-1$ and can be lower-bounded by $\lfloor n/2 \rfloor$. On the other hand, there exists a Boolean function with threshold number 2 that cannot be computed by a single spiking neuron [27]. We prove that a modified version of RPN, denoted as DLLT-RPN, whose instances are Boolean functions $f$ from DLLT (instead of DNF) is also co-NP-hard. This means that it is hard to decide whether a given $n$-variable Boolean function expressed as a disjunction of $O(n)$ threshold gates can be computed by a single spiking neuron.

**Theorem 5.** *RPN and DLLT-RPN are co-NP-hard and belong to* $\Sigma_2^p$.

*Proof.* The *tautology* problem that is known to be co-NP-complete [7] will be reduced to RPN in polynomial time in a similar way as it was done for the linear separability problem [13]:

**Tautology Problem** (TP)
*Instance:* A Boolean function $g$ in DNF.
*Question:* Is $g$ a tautology, i.e. $g(\boldsymbol{x}) = 1$ for every $\boldsymbol{x} \in \{0,1\}^n$?

For the DLLT-RPN, a modified version of TP, denoted as DLLT-TP, whose instances are Boolean functions $g$ from DLLT will be exploited. For proving that the DLLT-TP remains co-NP-complete, any TP instance $\vee_{j=1}^m C_j$ with $m$ monomials (conjunctions of literals over $n$ variables) can be equivalently rewritten in DNF $\vee_{j=1}^m ((C_j \wedge x'_j) \vee (C_j \wedge \bar{x}'_j))$ where $x'_1, \ldots, x'_m$ are $m$ new variables. Clearly, in the new DNF formula the number of monomials is linear in terms of the number of variables. Moreover, any monomial can obviously be computed by a single threshold gate.

Thus given a TP (DLLT-TP) instance $g$ over $n$ variables $x_1, \ldots, x_n$, we construct a corresponding RPN (DLLT-RPN) instance $f$ over $n+2$ variables $x_1, \ldots, x_n, y_1, y_2$ in polynomial time as follows:

$$f(x_1, \ldots, x_n, y_1, y_2) = (g(x_1, \ldots, x_n) \wedge y_1) \vee (y_1 \wedge \bar{y}_2) \vee (\bar{y}_1 \wedge y_2) \,. \tag{44}$$

For TP instance $g$, function $f$ is actually in DNF as required for the RPN. For DLLT-TP instance $g = \vee_{j=1}^{m} g_j$ with $g_j$ from LT, formula (44) contains terms $g_j \wedge y_1$ that are equivalent with $\overline{\bar{g}_j \vee \bar{y}_1}$ which belong to LT since class LT is closed under negation [21] and summand $W(1 - y_1)$ with a sufficiently large weight $W$ can be added to the weighted sum for $\bar{g}_j$ to evaluate $\bar{g}_j \vee \bar{y}_1$. This implies that $f$ is from DLLT representing a DLLT-RPN instance.

It will be shown that the TP (DLLT-TP) instance has a solution iff the corresponding RPN (DLLT-RPN) instance is solvable. So first assume that $g$ is a tautology. Hence $f$ given by (44) can be equivalently rewritten as $y_1 \vee y_2$ which is trivially computable by a spiking neuron. On the other hand, assume that there exists $\boldsymbol{a} \in \{0,1\}^n$ such that $g(\boldsymbol{a}) = 0$. In this case, $f(\boldsymbol{a}, y_1, y_2)$ reduces to $XOR(y_1, y_2)$ which cannot be implemented by a single spiking neuron [20].

For proving that $RPN \in \Sigma_2^p$ (similarly for DLLT-RPN) consider an alternating algorithm for the RPN that, given $f$ in DNF, guesses polynomial-size representations [20] of weights and delays for spiking neuron $\mathcal{N}$ first in its existential state, and then verifies $y_{\mathcal{N}}(\boldsymbol{x}) = f(\boldsymbol{x})$ for every $\boldsymbol{x} \in \{0,1\}^n$ ($y_{\mathcal{N}}(\boldsymbol{x})$ can be computed in polynomial time since there are only linear number of time intervals to check) in its universal state. □

## 5 Conclusion

The computational complexity of training a single perceptron with programmable synaptic delays which is a model that covers certain aspects of spiking neurons (with binary coded inputs) has been analyzed. We have developed a synchronization technique that generalizes the known non-learnability results for arbitrary synaptic delays. In particular, we have proved that the perceptrons with delays are not properly PAC-learnable and the spiking neurons do not allow robust learning unless $RP = NP$. This represents a step towards solving an open issue concerning the PAC-learnability of spiking neurons with arbitrary delays. In addition, we have shown that it is co-NP-hard to decide whether a disjunction of $O(n)$ threshold gates, which is known to implement any spiking neuron, can reversely be computed by a single spiking neuron. An open issue remains for further research whether the spiking neurons are learnable with membership and equivalence queries.

## References


1. Aizenstein, H., Hegedüs, T., Hellerstein, L., Pitt, L.: Complexity Theoretic Hardness Results for Query Learning. Computational Complexity **7** (1) (1998) 19–53
2. Amaldi, E.: On the complexity of training perceptrons. In: Kohonen, T., Mäkisara, K., Simula, O., Kangas, J. (eds.): Proceedings of the ICANN'91 First International Conference on Artificial Neural Networks. Elsevier Science Publisher, North-Holland, Amsterdam (1991) 55–60
3. Anthony, M., Bartlett, P.L.: Neural Network Learning: Theoretical Foundations. Cambridge University Press, Cambridge, UK (1999)
4. Arora, S., Babai, L., Stern, J., Sweedyk, Z.: The hardness of approximate optima in lattices, codes, and systems of linear equations. Journal of Computer and System Sciences **54** (2) (1997) 317–331
5. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Learnability and the Vapnik-Chervonenkis dimension. Journal of the ACM **36** (4) (1989) 929–965
6. Bohte, M., Kok, J.N., La Poutré, H.: Spike-prop: error-backpropagation in multi-layer networks of spiking neurons. In: Proceedings of the ESANN'2000 European Symposium on Artificial Neural Networks. D-Facto Publications, Brussels (2000) 419–425
7. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the STOC'71 Third Annual ACM Symposium on Theory of Computing. ACM Press, New York (1971) 151–158
8. Fahlman, S.E., Lebiere, C.: The cascade-correlation learning architecture. In: Touretzky, D.S. (ed.): Advances in Neural Information Processing Systems (NIPS'89), Vol. 2. Morgan Kaufmann, San Mateo (1990) 524–532
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, San Francisco (1979)
10. Gerstner, W., Kistler, W.M.: Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press, Cambridge, UK (2002)
11. Hammer, P.L., Ibaraki, T., Peled, U.N.: Threshold numbers and threshold completions. In: Hansen, P. (ed.): Studies on Graphs and Discrete Programming, Annals of Discrete Mathematics **11**, Mathematics Studies, Vol. 59. North-Holland, Amsterdam (1981) 125–145
12. Haykin, S.: Neural Networks: A Comprehensive Foundation. 2nd edn. Prentice-Hall, Upper Saddle River, NJ (1999)
13. Hegedüs, T., Megiddo, N.: On the geometric separability of Boolean functions. Discrete Applied Mathematics **66** (3) (1996) 205–218
14. Höffgen, K.-U., Simon, H.-U., Van Horn, K.S.: Robust trainability of single neurons. Journal of Computer and System Sciences **50** (1) (1995) 114–125
15. Hush, D.R.: Training a sigmoidal node is hard. Neural Computation **11** (5) (1999) 1249–1260
16. Johnson, D.S., Preparata, F.P.: The densest hemisphere problem. Theoretical Computer Science **6** (1) (1978) 93–107
17. Judd, J.S.: Neural Network Design and the Complexity of Learning. The MIT Press, Cambridge, MA (1990)
18. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.): Complexity of Computer Computations. Plenum Press, New York (1972) 85–103
19. Maass, W., Bishop, C.M. (eds.): Pulsed Neural Networks. The MIT Press, Cambridge, MA (1999)

20. Maass, W., Schmitt, M.: On the complexity of learning for spiking neurons with temporal coding. Information and Computation **153** (1) (1999) 26–46
21. Parberry, I.: Circuit Complexity and Neural Networks. The MIT Press, Cambridge, MA (1994)
22. Pitt, L., Valiant, L.G.: Computational limitations on learning from examples. Journal of the ACM **35** (4) (1988) 965–984
23. Roychowdhury, V.P., Siu, K.-Y., Kailath, T.: Classification of linearly non-separable patterns by linear threshold elements. IEEE Transactions on Neural Networks **6** (2) (1995) 318–331
24. Roychowdhury, V.P., Siu, K.-Y., Orlitsky, A. (eds.): Theoretical Advances in Neural Computation and Learning. Kluwer Academic Publishers, Boston (1994)
25. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review **65** (6) (1958) 386–408
26. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323** (1986) 533–536
27. Schmitt, M.: On computing Boolean functions by a spiking neuron. Annals of Mathematics and Artificial Intelligence **24** (1-4) (1998) 181–191
28. Šíma, J.: Training a single sigmoidal neuron is hard. Neural Computation **14** (11) (2002) 2709–2728
29. Vidyasagar, M.: A Theory of Learning and Generalization. Springer-Verlag, London (1997)