A New Supervised Learning Algorithm for Spiking Neurons

Yan Xu
xuyanhehai@163.com
Xiaoqin Zeng
xzeng@hhu.edu.cn
Institute of Intelligence Science and Technology, Hohai University, Nanjing, Jiangsu 211100, China

Shuiming Zhong
zhongyi@hhu.edu.cn
School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China

The purpose of supervised learning with temporal encoding for spiking neurons is to make the neurons emit a specific spike train encoded by the precise firing times of spikes. If only running time is considered, the supervised learning for a spiking neuron is equivalent to distinguishing the times of desired output spikes and the other time during the running process of the neuron through adjusting synaptic weights, which can be regarded as a classification problem. Based on this idea, this letter proposes a new supervised learning method for spiking neurons with temporal encoding; it first transforms the supervised learning into a classification problem and then solves the problem by using the perceptron learning rule. The experiment results show that the proposed method has higher learning accuracy and efficiency over the existing learning methods, so it is more powerful for solving complex and real-time problems.

1 Introduction

In biological nervous systems, a neuron transmits information to others via spike trains with specific frequency and amplitude (Bose & Liang, 1996; Gerstner & Kistler, 2002). The information is usually encoded in two ways: the frequency of spiking (rate encoding) (Adrian & Zotterman, 1926; Kandel, Schwartz, & Jessel, 1991) and the firing times of the spikes (pulse or temporal encoding) (Theunissen & Miller, 1995; Rullen & Thorpe, 2001; Rullen, Guyonneau, & Thorpe, 2005). The inputs and outputs of traditional artificial neuron models such as threshold perceptrons and sigmoidal neurons consider only rate encoding and will result in a loss of information expressed in the form of precise firing times of spikes. Spiking neurons (SNs) and spiking neural networks (SNNs), with inputs and outputs being firing times of spikes, were proposed (Maass, 1997b) to better simulate biological neurons. A spike train encoded by the firing times of spikes also contains the information of the spiking rate, so SNs are more powerful than sigmoidal neurons in terms of function. As is turned out, SNNs can simulate any sigmoidal neural network (Maass, 1997a). Furthermore, networks of noisy SNs with temporal encoding have more computational power than sigmoidal neural networks with the same number of neurons (Maass, 1997c).

The research on artificial neural networks—supervised learning methods for traditional neural networks, such as the perceptron learning rule (Minsky & Papert, 1969), the BP algorithm (Rumelhart, Hinton, & Williams, 1986), training methods for RBF (Broomhead & Lowe, 1988), and SNNs based on rate encoding (Kroese & van der Smagt, 1996; Rojas, 1996)—has established a solid foundation for the theory and application of neural networks, but their solutions cannot be directly applied to SNNs dealing with temporal encoding. The way SNs work is similar to the way that biological neurons do. However, the mechanism of supervised learning in the biological neurons remains unclear (Ponulak & Kasiński, 2010). In view of this, research on supervised learning for SNs and SNNs is still at an early stage, and many existing learning methods (Belatreche, Maguire, McGinnity, & Wu, 2003; Carnell & Richardson, 2005; Pfister, Toyoizumi, Barber, & Gerstner, 2006; Gütig & Sompolinsky, 2006; Schrauwen & Campenhout, 2006) have some weaknesses (Kasiński & Ponulak, 2006; Ponulak & Kasiński, 2010). This letter focuses on supervised learning based on temporal encoding; that is, during a certain running time, neurons can precisely emit spikes at appointed times through learning.

There are two types of supervised learning methods for SNNs based on temporal encoding, which are classified according to the number of spikes that need to be controlled precisely. The first type can control only the firing time of a single spike—single-spike learning. The most typical ones are SpikeProp (Bohte, Kok, & La Poutré, 2002) and its various improvements (Ghosh-Dastidar & Adeli, 2007; McKennoch, Liu, & Bushnell, 2006; Silva & Ruano, 2005; Xin & Embrechts, 2001; Schrauwen & Campenhout, 2004), which are based on gradient descent. These methods extend the traditional BP algorithm to SNNs and can be applied to solving classification problems. In addition to gradient descent–based methods, there are some other single-spike learning methods (Belatreche et al., 2003; Ruf & Schmitt, 1997). Although single-spike learning has good application capability, networks with only single-spike output will have limitations in the capacity and diversity of information that they transmit.

Relative to single-spike learning methods, multispike learning methods, which can control the firing times of multiple spikes, are more consistent with the running and information-transmitting mode of the biological neurons. Therefore, research on multispike learning methods is more meaningful. Because the complexity of the learning targets increases significantly, almost all the existing multispike learning methods focus on single SNs or single-layer SNNs. Carnell and Richardson (2005) put forward a spike sequence learning method using linear algebraic means. Spike-timing-dependent plasticity (STDP) (Markram, Lübke, Frotscher, & Sakmann, 1997; Bi & Poo, 1998) is a learning mode obtained from biological experiments. Although it can be run only under unsupervised mode, its theory and idea have appeared in many supervised learning methods (Legenstein, Naeger, & Maass, 2005; Pfister et al., 2006). Some learning methods (Florian, 2007; Legenstein, Pecevski, & Maass, 2008) combine reinforcement learning with STDP to realize supervised learning based on rate encoding or temporal encoding. Ponulak (2005) proposed a supervised learning algorithm for a spiking neuron (SN), which was referred to as the remote supervised method (ReSuMe) and gave detailed analyses of the algorithm afterward (Ponulak & Kasiński, 2010). In the ReSuMe learning process, the desired (supervisory) signal does not directly influence the membrane potential of the corresponding learning neuron. To be more specific, the basic idea of ReSuMe is from the Widrow-Hoff rule, according to which a practical learning algorithm composed of two weight update processes is derived. The two processes are the STDP process for strengthening synapses based on input spike trains and desired output spike train and the anti-STDP process (Kistler, 2002) for weakening synapses based on the input spike trains and actual output spike train. The results, which have already been presented, show that ReSuMe has good learning ability and wide application. Chronotron (Florian, 2012), another new supervised learning method, tries to minimize the distance defined by the Victor and Purpura (VP) metric (Victor & Purpura, 1996) between the actual and desired output spike trains to make neurons fire at the times of desired output spikes.

One common disadvantage of these multispike learning methods for SNs is that the learning efficiency is relatively low. If the desired output spike train is relatively long, the neuron needs to run a relatively long time in each learning epoch. The intricacy of the learning target determines that relatively more learning epochs are often needed during the learning process so that a better learning result can be obtained. For example, Ponulak and Kasiński (2010) needed about 450 learning epochs to achieve high learning accuracy when a neuron learned to emit a spike train of length 400 ms using ReSuMe. Thus, these methods need a lot of computation time, which lowers their learning efficiency and weakens their ability to solve real-time problems. In addition to learning efficiency, the learning accuracy of the existing learning methods will decrease considerably when the number of desired output spikes increases to a certain degree, and this limits their ability to solve complicated problems. The two disadvantages of the existing methods prompted us to search for a supervised learning method with higher learning efficiency and accuracy.

The running of a SN is a continuous process over a period of time. The running time can be divided into two categories: spike firing time and nonspike firing time. The supervised learning for a SN is equivalent to correctly distinguishing the times of desired output spikes and the other time during the running process of the neuron through adjusting synaptic weights. This is actually a classification problem; it uses the two kinds of times as two classes, and it can be solved by using existing classification methods. Traditional neural networks have good classification capability and a close relation with SNNs, so they can be given priority.

Among the traditional artificial neuron models, the perceptron (Minsky & Papert, 1969) is the simplest one; it uses a hard limit function as its activation function. The output of a perceptron is 1 when the weighted sum of its inputs is not less than 0 and is 0 when the weighted sum of its inputs is less than 0. Suppose a perceptron neuron has n inputs, which are denoted as $p = (p_1, p_2, \ldots, p_n)$; the corresponding n weights of the inputs are denoted as $W = (w_1, w_2, \ldots, w_n)$; the bias value is b, the neuron is expressed as

$$a = \text{hardlim}(Wp' + b) = \text{hardlim}(\alpha) \begin{cases} 1 & \alpha \geq 0, \\ 0 & \alpha < 0, \end{cases} \tag{1.1}$$

where hardlim is the hard limit function, and a is the output of the neuron.

The operation mechanism of a perceptron can be regarded as the simulation of an SN at a certain point in time. The output 1 of the perceptron corresponds to the spike firing, and the output 0 corresponds to the spike not firing. The critical point 0 of the hard limit function corresponds to the firing threshold.

The supervised learning method for a perceptron is the perceptron learning rule, with which a single perceptron neuron can solve linearly separable problems. If output 1 denotes the positive samples and output 0 denotes the negative samples and a training sample is denoted as $\{p,\ d\}$, where p is the inputs and d is the desired output of the sample, the perceptron learning rule can be expressed as

$$W^{\text{new}} = \begin{cases} W^{\text{old}} + p & \text{if } d = 1 \text{ and } a = 0, \\ W^{\text{old}} - p & \text{if } d = 0 \text{ and } a = 1, \\ W^{\text{old}} & \text{if } d = a. \end{cases} \tag{1.2}$$

Although the perceptron learning rule is simple, it is effective for solving linearly separable problems. Its validity and simplicity accord with the motivation of the research of a new supervised learning method for SNs in this letter. On the basis of the perceptron and the perceptron learning rule, this letter proposes a new supervised learning method for SNs, which we call perceptron-based spiking neuron learning rule(PBSNLR). The basic idea is to transform the supervised learning problem of SN into a classification problem of the perceptron model and then solve the classification problem using the perceptron learning rule to make the corresponding SN emit the desired output spike train precisely.

PBSNLR inherits the high-efficiency characteristic of the perceptron learning rule. Compared with existing learning methods, the training time that PBSNLR takes falls significantly. Its learning success rate and learning accuracy are significantly better than existing methods. Therefore, it has strong practicability.

The rest of this paper is organized as follows. In section 2 the SN model used in this letter is formally defined. The PBSNLR learning method is proposed in section 3. In section 4, some numerical experiments are given to investigate learning performance and some factors that influence PBSNLR. A classification problem of spike trains is solved by using PBSNLR in section 4. Discussion about PBSNLR and the conclusion are presented in section 5.

2 Spiking Neuron Model

Among all the existing SN models, the internal state of the spike response model (SRM) (Gerstner & Kistler, 2002) can be expressed intuitively. As a result, the SRM model is easy for depicting the learning method, and so we select it as the research object in this letter.

In the SRM model, the inputs of a neuron are spikes transmitted from presynaptic neurons to the neuron by numerous synapses. When each spike arrives, a postsynaptic potential (PSP) will be induced in the neuron if the neuron's internal state is under a threshold. The internal state of the neuron, the membrane potential, is defined as the sum of the PSPs induced by all the input spikes and affected by weights of the synapses that transmit these input spikes. Suppose a neuron has N input synapses; the ith synapse transmits G_i spikes whose times of arriving at the neuron are denoted as $\mathcal{G}_i = \{t_i^{(1)}, t_i^{(2)}, \ldots, t_i^{(G_i)}\}$; the time of the most recent output spike of the neuron prior to the current time $t(>0)$ is $t^{(f_r)}$; and the internal state of the neuron is expressed as

$$u(t) = \eta\big(t - t^{(f_r)}\big) + \sum_{i=1}^{N} w_i \sum_{g} \varepsilon_i\big(t - t^{(f_r)}, t - t_i^{(g)}\big) + \int_0^{\infty} \kappa\big(t - t^{(f_r)}, s\big) I^{\text{ext}}(t - s)\, ds, \tag{2.1}$$

where w_i is the weight of the ith synapse. The PSP induced by the spike $t_i^{(g)}$ is determined by the spike response function $\varepsilon_i(t - t^{(f_r)}, t - t_i^{(g)})$. The last integral term accounts for the effects of an external driving current I^{ext}.

When the internal state exceeds the neuron threshold ϑ from low to high, the neuron fires an output spike at exactly the same time as $t^{(f)}$ (here $t^{(f)}$

not only indicates the firing time of the spike but also represents the spike itself). At the same time, the internal state immediately starts dropping to 0 (an internal state equal to 0 is called the resting potential, or u_{rest}, of the neuron), and the neuron cannot fire regardless of the number or frequency of the input spikes. This phase is known as repolarization or absolute refractory period (Kandel et al., 1991). Subsequently, the internal state is kept at a value lower than u_{rest} by various biological processes. As a result, it becomes difficult for the neuron to reach the threshold and fire again for a period of time. This phase is known as hyperpolarization, or the relative refractory period.

The two phases affect the internal state of the neuron and the spike firing after $t^{(f)}$. In the expression of the neuron's internal state, the refractoriness function $\eta(t - t^{(f_r)})$ depicts the relative refractory period. This letter considers only the effect of refractoriness induced by the most recent output spike prior to time t and ignores the earlier spikes. If the neuron has not fired any spike, that is, for $t < t^{(1)}$, the refractoriness term is set to 0.

The kernel function $\kappa(t - t^{(f_r)}, s)$ is the linear response of the membrane potential to the input current I^{ext}. It describes the time course of a deviation of the membrane potential from its resting value that is caused by a short current pulse (impulse response). The input current term has diversity in the expression of SN's internal state. For example, it can be omitted in simplified models; some complex models can have various forms of external currents; and in some cases, the neurons will run in the environment with the background noise of external currents.

The three terms on the right-hand side of equation 2.1 all contain $t - t^{(f_r)}$. This indicates that they are all related to $t^{(f_r)}$, the time of the most recent output spike prior to the current time t. In addition, in the standard SRM model, the firing threshold of a neuron is a variable that is also related to $t^{(f_r)}$. However, Gerstner and Kistler (2002) point out that the time-related variable threshold can be easily transformed into a fixed threshold. So for simplification, the firing threshold in this letter is set to a fixed value.

3 Learning Method: PBSNLR

The difference between perceptrons and SNs is obvious and includes the running mechanism and the input and output forms. A perceptron does not involve the concept of time. The inputs of a perceptron are real numbers, of which the weighted sum is calculated in the perceptron. The output is also a real number. The running of an SN is based on a continuous period of time, and the inputs of the SN are the times of discrete spikes. These input spikes induce PSPs in the neuron, and the weighted sum of the PSPs determines whether a spike fires at the current moment. The outputs of the SN are the times of fired spikes.

To use the perceptron learning rule to solve the supervised learning problems of SNs, we first need to transform the problems into perceptron

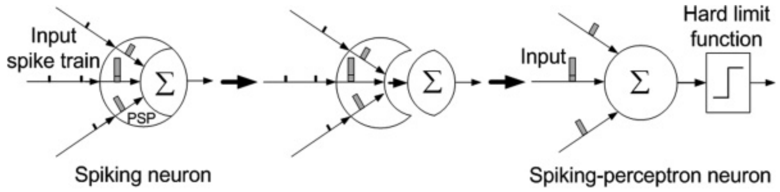


Figure 1: Transforming SN into SPN. The calculation of PSPs is moved to the outside of the SN, and the PSPs are considered as the inputs of the SPN.

form. The firing of a spike is determined by two factors: the neuron's internal state and the firing threshold. For threshold ϑ, it corresponds to the critical point 0 of the hard limit function and is easy to transform. For example, as long as the internal state is transformed into $u - \vartheta$, the threshold will be changed to 0.

If we consider only the sum term of PSPs and ignore the other two terms on the right-hand side of equation 2.1, we can find a common feature between perceptrons and SNs: the calculation of a weighted sum is needed in both kinds of neuron. The difference is that a perceptron needs to calculate the weighted sum only once, while SNs need to continuously calculate the weighted sum according to running time. If a fixed point in the running time of a SN is considered, the neuron needs to calculate the weighted sum one time at the point, which is more similar to a perceptron. But in this case, the SN and perceptron are not exactly the same, because the PSPs need to be calculated from the input spikes in the SN.

As shown in Figure 1, if the calculation of PSPs is regarded as an external operation of an SN or this part of operation is moved to the outside of the SN and only the calculation of weighted sum is retained in the inside of the SN, the internal operations of the two kinds of neurons can be regarded as the same. The PSPs can be regarded as the inputs of the SN (i.e., the inputs of the SN are changed from the input spikes to the PSPs induced by them). Because the PSPs are all real numbers, the inputs and the internal operation of the SN are both transformed into the forms of a perceptron at the considered time point.

Finally, there are only two kinds of output states of an SN at a point in time: firing a spike or not. The state of firing spike can correspond to the output 1, and the opposite case can correspond to the output 0 of a perceptron. Thus, the SN is transformed into perceptron form at a point in its running time. We call the neuron obtained after transformation the spiking perceptron neuron (SPN).

SPN does not change the core running mechanism of a SN. The purpose of introducing this concept is to use the supervised learning method for perceptrons—the perceptron learning rule—to solve the supervised learning problems of SNs. For this purpose, the input spike trains and the desired output spike trains of SNs need to be transformed into the inputs and desired outputs of the training samples of SPNs.

Suppose an SN has N input synapses; the ith synapse transmits G_i spikes whose times of arriving at the neuron are denoted as $\mathcal{G}_i = \{t_i^{(1)}, t_i^{(2)}, \ldots, t_i^{(G_i)}\}$; the desired output spike train is composed of F spikes, which are denoted as $\hat{T} = \{\hat{t}^{(1)}, \hat{t}^{(2)}, \ldots, \hat{t}^{(F)}\}$; and the length of the desired spike train is T. The input spike trains and the desired output spike train of the SN are transformed into the inputs and the desired output of a training sample of an SPN at a time point t_d ($t_d \in T$).

The PSPs induced by the input spikes in the SN at t_d are the inputs of a training sample of the SPN. To be specific, it can be obtained using equation 2.1; the sum of PSPs induced by all the spikes that have arrived through the ith synapse at t_d is computed as

$$P_i^{t_d} = \sum_g \varepsilon_i\left(t_d - \hat{t}^{(f_r)}, t_d - t_i^{(g)}\right). \tag{3.1}$$

What needs special attention is that the $\hat{t}^{(f_r)}$ in equation 3.1 is not the most recent actual but the most recent desired output spike of the neuron prior to t_d. The change of firing time of an actual output spike will lead to a change of subsequent PSP values, which are the inputs of the SPN. The perceptron learning rule requires that the inputs of training samples remain unchanged in different learning epochs of the learning process. Therefore, the PSPs must be calculated according to the desired output spikes when the input spike trains of the SN are transformed into the inputs of the SPN. In other words, the training samples of the SPN are obtained based on the state that the SN has been able to emit the desired output spike train. Then these samples are trained by using the perceptron learning rule. If the learning succeeds, the inputs of the SPN can correctly correspond to the actual running process of the original SN, and the weights of the SPN are those where the SN can emit the desired output spike train correctly.

The refractoriness function and the external input current terms on the right-hand side of equation 2.1 need considering in the inputs of the SPN. At t_d, the value of refractoriness function is computed as

$$R^{t_d} = \eta\left(t_d - \hat{t}^{(f_r)}\right). \tag{3.2}$$

The external input current term is computed as

$$I^{t_d} = \int_0^\infty \kappa\left(t_d - \hat{t}^{(f_r)}, s\right) I^{\text{ext}}(t_d - s)\, ds. \tag{3.3}$$

Similarly, the $\hat{t}^{(f_r)}$ in the two equations refers to the most recent desired output spike prior to t_d.

According to equation 2.1, the weighted sum in the SPN at t_d is computed as

$$\Phi^{t_d} = \sum_{i=1}^{N} w_i \cdot P_i^{t_d} + R^{t_d} + I^{t_d} = W\boldsymbol{P}^{t_d\prime} + R^{t_d} + I^{t_d}, \tag{3.4}$$

where $W = (w_1, w_2, \ldots, w_N)$ denotes the weight vector and $\boldsymbol{P}^{t_d} = (P_1^{t_d}, P_2^{t_d}, \ldots, P_N^{t_d})$ denotes the vector composed of the PSPs. Because R^{t_d} and I^{t_d} do not involve weight adjustment, only $\boldsymbol{P}^{t_d}$ is regarded as the input vector and $R^{t_d} + I^{t_d}$ can be regarded as the bias value of the SPN.

Finally, we use $\Phi^{t_d} - \vartheta$ to adjust the firing threshold ϑ. Thus, at t_d, the SPN is expressed as

$$a^{t_d} = \text{hardlim}(\Phi^{t_d} - \vartheta) = \begin{cases} 1 & \Phi^{t_d} - \vartheta \geq 0, \\ 0 & \Phi^{t_d} - \vartheta < 0. \end{cases} \tag{3.5}$$

In the perceptron learning rule, there is a desired output corresponding to every input. For PBSNLR, the desired output is also required, corresponding to the input that is obtained at t_d. If t_d is exactly the firing time of a desired output spike, the desired output d^{t_d} of the SPN at t_d is 1. If t_d is not the firing time of a desired output spike, the desired output d^{t_d} is 0. Obviously the times of all the desired output spikes of the SN should be selected to constitute the positive training sample set of the SPN, which can be expressed as

$$\mathbb{P}_{\text{pos}} = \{\boldsymbol{P}^{t_d}; d^{t_d} = 1 \mid t_d \in \hat{T} = \{\hat{t}^{(1)}, \hat{t}^{(2)}, \ldots, \hat{t}^{(F)}\}\}. \tag{3.6}$$

The negative training sample set of the SPN can be arbitrarily selected at any time but the times of desired output spikes. A simple conclusion is that the larger the number of negative samples is, the closer the spike firing that directly corresponds to the sample outputs of the SPN and the actual running result of the original SN is, and therefore, the higher the learning accuracy is. But more negative samples means more computation and storage space, so the selection should be made according to the actual situation. The negative training sample set of the SPN can be expressed as

$$\mathbb{P}_{\text{neg}} = \{\boldsymbol{P}^{t_d}; d^{t_d} = 0 \mid t_d \in T \setminus \hat{T}\}. \tag{3.7}$$

Figure 2 gives a simple example to illustrate how the inputs and desired outputs of an SPN are obtained from the input and desired output spike trains of an SN. According to the values in the figure, the inputs and desired outputs of the SPN and the weighted sums in the SPN at the four time points

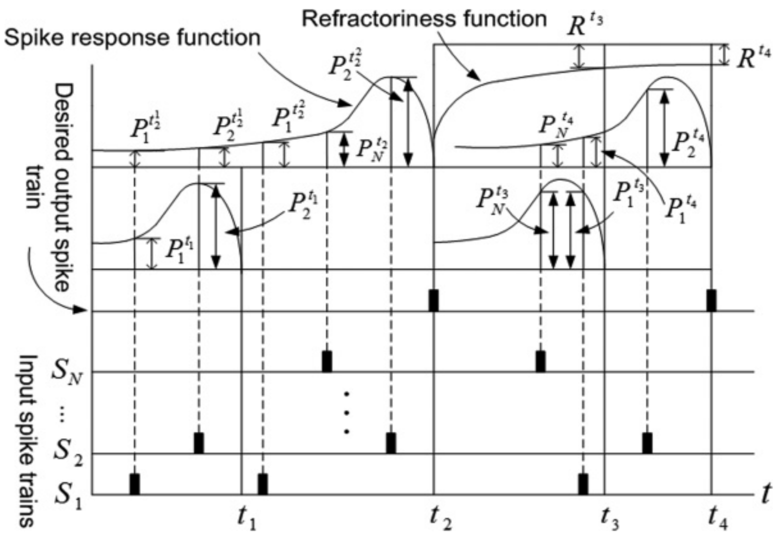


Figure 2: A simple example used to illustrate how the inputs and desired outputs of an SPN are obtained from the input and desired output spike trains of an SN. A single SN is needed to learn to emit two desired output spikes over a period of time. Four time points t_1, t_2, t_3, t_4 are selected in this period; t_2 and t_4 are the firing times of two desired output spikes. The calculated value of every PSP of every input synapse at every time point and the values of the relative refractory period after the time of the first desired output spike are listed.

are computed as

$$t_1\!: \ \boldsymbol{P}^{t_1} = \left(P_1^{t_1}, P_2^{t_1}, \ldots, 0\right), \ \ d^{t_1} = 0, \ \ \Phi^{t_1} = \sum_{i=1}^{N} w_i \cdot P_i^{t_1},$$

$$t_2\!: \ \boldsymbol{P}^{t_2} = \left(P_1^{t_2^1} + P_1^{t_2^2}, P_2^{t_2^1} + P_2^{t_2^2}, \ldots, P_N^{t_2}\right), \ \ d^{t_2} = 1, \ \ \Phi^{t_2} = \sum_{i=1}^{N} w_i \cdot P_i^{t_2},$$

$$t_3\!: \ \boldsymbol{P}^{t_3} = \left(P_1^{t_3}, 0, \ldots, P_N^{t_3}\right), \ \ d^{t_3} = 0, \ \ \Phi^{t_3} = \sum_{i=1}^{N} w_i \cdot P_i^{t_3} + R^{t_3},$$

$$t_4\!: \ \boldsymbol{P}^{t_4} = \left(P_1^{t_4}, P_2^{t_4}, \ldots, P_N^{t_4}\right), \ \ d^{t_4} = 1, \ \ \Phi^{t_4} = \sum_{i=1}^{N} w_i \cdot P_i^{t_4} + R^{t_4}.$$

The next step is to take advantage of the perceptron learning rule and the training samples obtained to train the SPN. The expression of the weight update is

$$W^{\text{new}} = \begin{cases} W^{\text{old}} + \beta \boldsymbol{P}^{t_d} & \text{if } d^{t_d} = 1 \text{ and } a^{t_d} = 0, \\ W^{\text{old}} - \beta \boldsymbol{P}^{t_d} & \text{if } d^{t_d} = 0 \text{ and } a^{t_d} = 1, \\ W^{\text{old}} & \text{if } d^{t_d} = a^{t_d}, \end{cases} \tag{3.8}$$

where β is the learning rate.

PBSNLR can be summarized by the following steps:

1. Transform the supervised learning problem of a SN into a classification problem. This step includes selecting some points in the running time of the SN to constitute training samples and transforming the internal state, the input spike trains, and the desired output spike train of the SN into corresponding elements of an SPN.
2. Train the SPN by using the training samples and the perceptron learning rule.
3. Run the original SN with the learned weights after training.

It should be noted that PBSNLR does not work in an online manner because all of the PSPs, induced by every synapse at the times of all the training samples, need to be calculated and stored before training. The learning process is related to the SPNs only, and the original SNs are not run in every learning epoch, which is contrary to some existing online supervised learning methods, such as ReSuMe.

4 Results of Experiments

4.1 Learning Sequences of Spikes. To quantitatively evaluate the learning performance, as Ponulak and Kasiński (2010) did, we use the correlation-based measure C (Schreiber, Fellous, Whitmer, Tiesinga, & Sejnowski, 2003) to express the similarity between the desired and actual output spike trains. The desired output spike trains are required to be learnable in the experiments because the minimum time interval of two adjacent spikes emitted by the neurons is 3 ms according to the length of the absolute refractory period and the characteristic of the spike response function. Therefore, the minimum time interval of any two adjacent spikes in the desired output spike trains is set to 3 ms in the experiments. (See the appendix for details on the experimental strategies and parameter settings.)

4.1.1 Learning Results. In the first example, a neuron with 200 synaptic inputs is trained to emit a desired spike train with the length 500 ms. Every input spike train and the desired output spike train are generated randomly according to the homogeneous Poisson process with rate $r = 10$ Hz and 60 Hz, respectively (hereafter, we refer to the resulting spike trains as rHz Poisson spike trains).

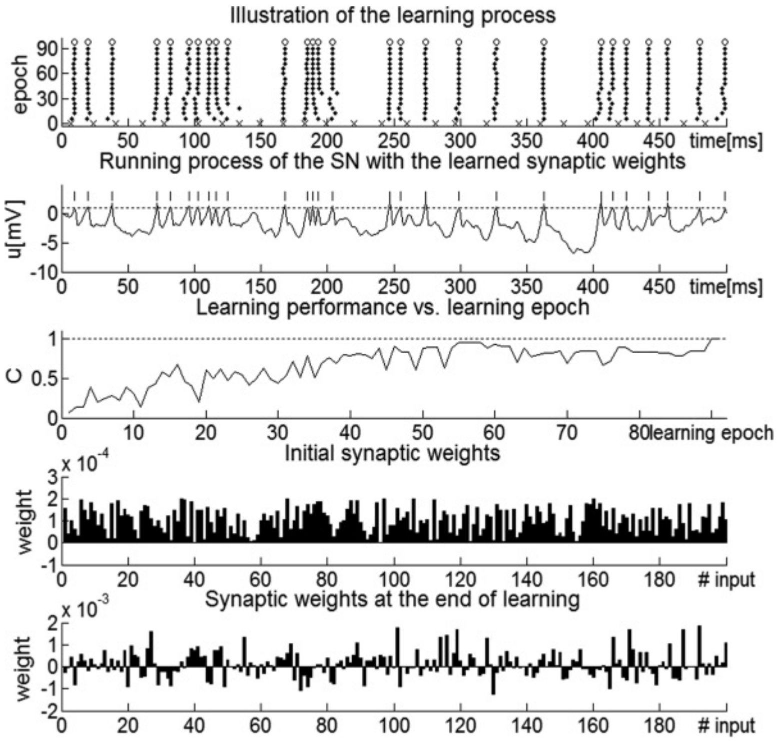


Figure 3: The temporal sequence learning with PBSNLR. A single SN with 200 synaptic inputs is trained on a 60 Hz Poisson target spike train of length 500 ms. The desired output spikes are shown by ○. × denotes the initial output spikes of the SN. • denotes the actual output spike trains at some learning epochs. The SN successfully emits the desired spike train after about 90 learning epochs of PBSNLR.

In this example, the original SN is run by using the learned synaptic weights of the SPN after every learning epoch of PBSNLR; therefore, the measure C between the actual and desired output spike trains can be investigated after every learning epoch. The initial synaptic weights are selected as the uniform distribution in the interval $(0, 0.2 \cdot 10^{-3})$. When the learning rate β is selected as 0.05, the SN can successfully emit the desired spike train after 92 learning epochs. The top row of Figure 3 shows the complete learning process, which includes the desired output spike train, the initial output spike train, and the actual output spike trains at some learning epochs during the learning process. The measure C plotted as a function of the learning epoch is shown in Figure 3, which indicates that learning accuracy is high within about 60 epochs, and about 30 more epochs lead to learning success.

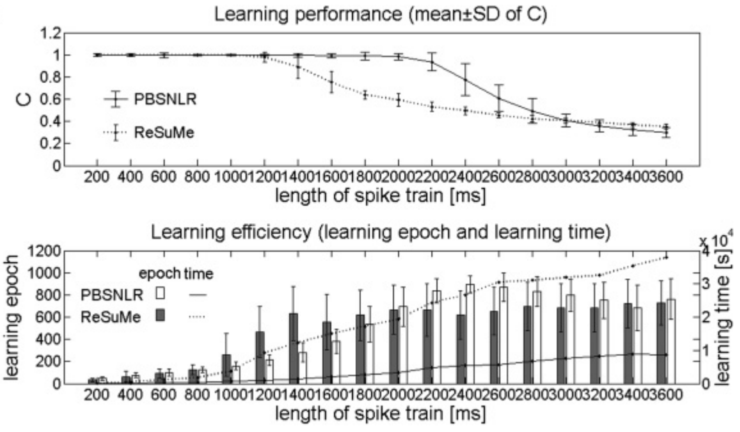
We next look at some groups of experiments used to measure the overall learning performance of PBSNLR. In the experiments, we investigate the effect of different factors on learning performance. The factors include the length of spike trains, the firing rate of spike trains, and the number of synaptic inputs. We will compare PBSNLR with the existing supervised learning method, ReSuMe.

In the first and second groups of experiments, the length of desired output spike trains varies from short to long, while the other settings remain the same. In the first group, the number of the synaptic inputs is 400; every input spike train and the desired output spike train are Poisson spike trains with rates 10 Hz and 100 Hz, respectively. The length of the desired output spike trains is increased from 200 ms to 3600 ms with an interval of 200 ms. In the second group, the number of the synaptic inputs is 400; every input spike train and the desired output spike train are Poisson spike trains with the same rate, 200 Hz. The length of the desired output spike trains is increased from 200 ms to 2800 ms with an interval of 200 ms. The experimental results are shown in Figure 4.

First, learning accuracy is investigated. Figure 4 shows that the learning accuracies of the two methods are very high (measure C is close to 1) when the length of spike trains is not so long; accuracy decreases when the train length increases gradually in the two groups of experiments. The accuracy curve of ReSuMe declines relatively early (from 1200 ms in Figure 4A and 800 ms in Figure 4B) and slowly, while the accuracy curve of PBSNLR declines relatively late (from 2200 ms in Figure 4A and 1400 ms in Figure 4B) and steeply. Finally, the two curves intersect. The learning accuracy of PBSNLR is significantly higher than the learning accuracy of ReSuMe within a range of train length (1400–2800 ms in Figure 4A and 1000–1600 ms in Figure 4B). When the length of spike trains exceeds a certain value (3000 ms in Figure 4A and 1800 ms in Figure 4B), the learning accuracies of the two methods reduce to a very low level, with the accuracy of ReSuMe slightly higher than PBSNLR. The error bars on the accuracy curves in Figure 4 show that the standard deviation (SD) of mean accuracy of PBSNLR has a larger increase with an increase in train length. So the stability of PBSNLAR’s learning accuracy is weaker than that of ReSuMe with its decreased learning accuracy.

We next investigate learning efficiency. When the length of spike trains is relatively short (i.e., when the learning accuracy is very high), the PBSNLR learning epochs are smaller than the learning epochs of ReSuMe. When learning accuracy begins to fall, PBSNLR has more learning epochs than ReSuMe. On the whole, the difference between the learning epochs of the two methods is not obvious. However, in the 50 experiments, PBSNLR’s computing time was significantly less than ReSuMe’s, and the advantage becomes more and more significant as train length increases. For example, in Figure 4B, ReSuMe took 20 times longer to complete 50 experiments than did PBSNLR when the length of spike trains was 2800 ms.

A



B

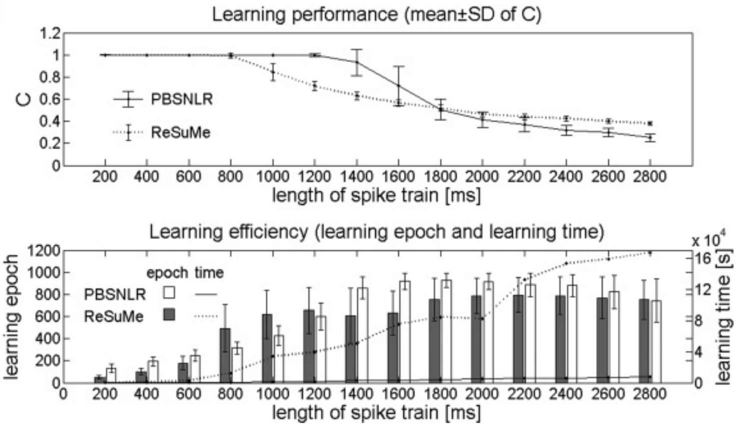


Figure 4: Learning results and the comparison of learning performance between PBSNLR and ReSuMe when the length of desired spike trains increases gradually. The number of the synaptic inputs is 400. (A) Every input spike train and the desired output spike train are Poisson spike trains with rates 10 Hz and 100 Hz, respectively. (B) Every input spike train and the desired output spike train are Poisson spike trains with the same rate, 200 Hz.

By comparing Figures 4A and 4B, we can observe the effect that the increase in firing rate has on the learning results. The maximum length of spike trains for PBSNLR and ReSuMe to keep learning accuracy very high is 2000 ms and 1200 ms, respectively, in Figure 4A and 1200 ms and 800 ms, respectively, in Figure 4B. Thus, the increased firing rate leads to decreased learning performance. As to learning efficiency, the increase in firing rate significantly increases the ReSuMe learning time but almost does not for PBSNLR.

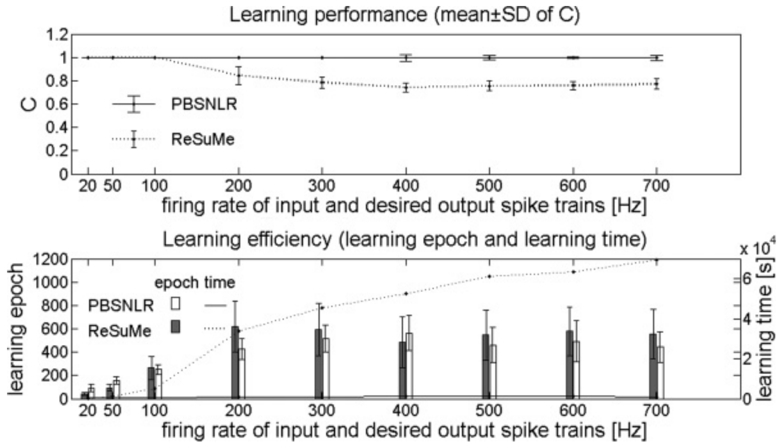


Figure 5: The learning results and the comparison of learning performance between PBSNLR and ReSuMe when the firing rates of the input and desired spike trains increase gradually. The number of the synaptic inputs is 400; the length of the input spike trains and the desired output spike train is 1000 ms; the firing rates of the input spike trains are the same as the firing rates of the desired output spike trains.

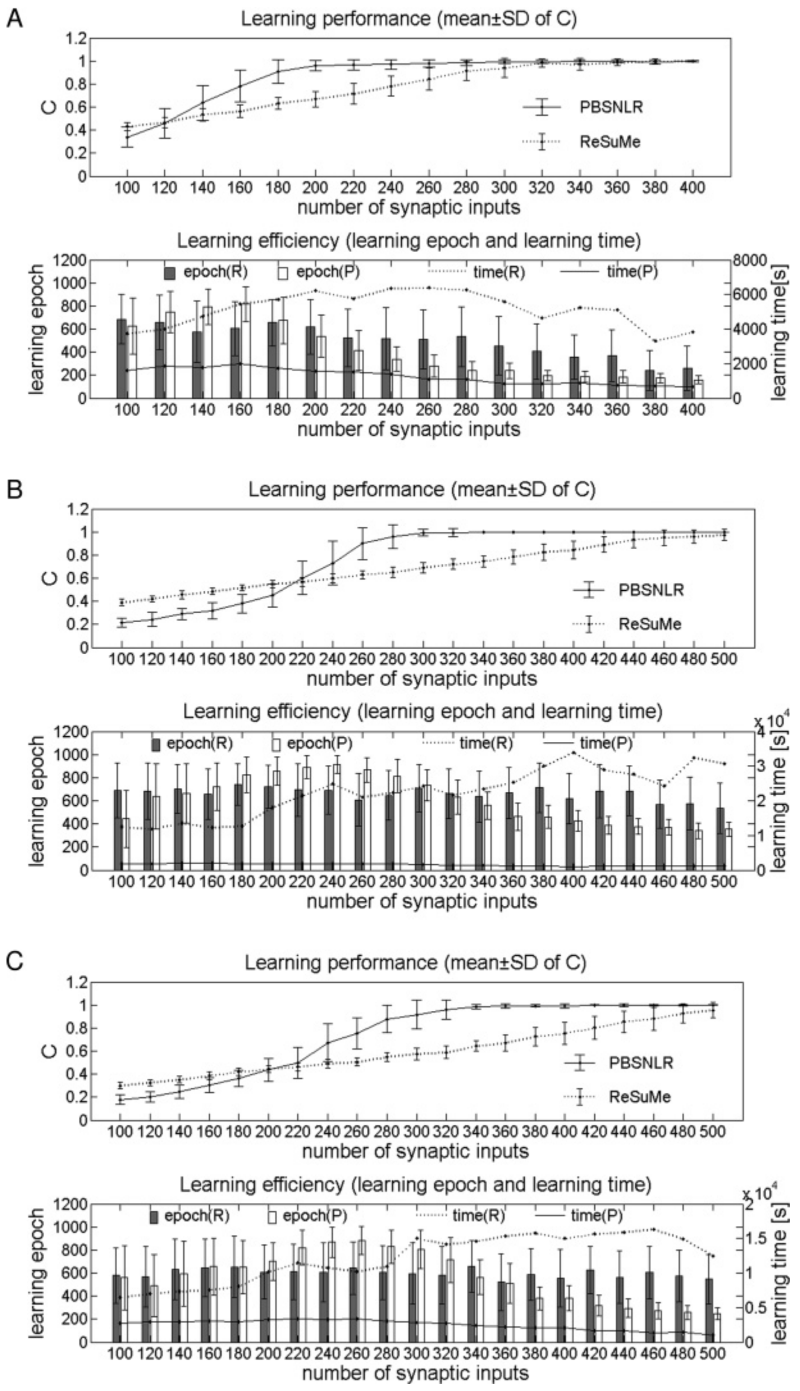
In the third group of experiments, the firing rates of input and desired output spike trains vary from low to high while the other settings remain the same. To be specific, the number of the synaptic inputs is 400; the length of the input and desired output spike trains is 1000 ms; every input spike train and the desired output spike train is a Poisson spike train with the same rate of 20, 50, 100, 200, 300, 400, 500, 600, and 700 Hz. The experimental results are shown in Figure 5.

Figure 5 shows that the learning accuracy of PBSBLR almost does not decline and always maintains very high values with the increase in spike train firing rate. Although the learning accuracy of ReSuMe declines a little the downward trend appears only within the firing rate of 100 Hz to 300 Hz, after which learning accuracy remains stable. The reason this happens is that although the increase in firing rate will increase the number of desired output spikes, it leads to a more uniform distribution of the time interval between two adjacent desired spikes. Therefore, the difficulty of learning is reduced, and both the mean values and the standard deviations (SDs) of the two methods' learning accuracies are stable. The PBSNLR learning accuracy is higher than the learning accuracy of ReSuMe when the firing rate is high.

In the experiments, the difference between the two methods' learning epochs that are needed to reach the highest accuracies is small. PBSNLR needs slightly fewer epochs than ReSuMe. Figure 5 shows that the computing time PBSNLR took to complete 50 experiments barely increases with an increased firing rate of desired output spike trains, while ReSuMe's computing time increases substantially. At 700 Hz,the computing time of ReSuMe is 60 times as much as that of PBSNLR. The reason for this is the difference between the operation mechanisms of the two learning methods. ReSuMe is an online learning algorithm; as long as an actual output is emitted or a time point of desired output spike is encountered during the running process, the synaptic weights will be updated. Therefore, the increased firing rate of desired output spike trains will lead to an increase of desired output spikes and increased computing time of a weight update. PBSNLR is an offline learning algorithm. If the length of desired output spike trains is fixed, the total number of SPNs' positive and negative samples often remains the same regardless of the increase of desired output spikes. The computing time of the perceptron learning rule is determined by the number of training samples, so the increase in firing rate will not increase the computing time of PBSNLR.

In the fourth, fifth, and sixth groups of experiments, the number of synaptic inputs varies from fewer to more while the other settings remain the same. In the fourth group, every input spike train and the desired output spike train are Poisson spike trains with the same length of 1000 ms and different rates of 10 Hz and 100 Hz, respectively; the number of the synaptic inputs is increased from 100 to 400 with an interval of 20. In the fifth group, every input spike train and the desired output spike train are Poisson spike trains with the same length of 1000 ms and the same rate of 200 Hz; the number of the synaptic inputs is increased from 100 to 500 with an interval of 20. In the sixth group, every input spike train and the desired output spike train are Poisson spike trains with the same length of 1600 ms and different rates of 10 Hz and 100 Hz, respectively; the number of the synaptic inputs is increased from 100 to 500 with an interval of 20. The experimental results are shown in Figure 6.

In Figure 6, the number of synaptic inputs gradually increases from left to right, and the learning accuracies of the two methods also increase from left to right in the three groups of experiments. PBSNLR can quickly reach a very high learning accuracy (measure C is close to 1) within a relatively short interval for the number of synaptic inputs. Although the learning accuracy of ReSuMe increases slowly, it also can make the measure C close to 1. The learning accuracy of PBSNLR is higher than the learning accuracy of ReSuMe in most cases. When there are fewer synaptic inputs (fewer than 120 in Figure 6A, 220 in Figure 6B, and 200 in Figure 6C), the learning accuracies of the two methods are both very low and the ReSuMe's accuracy is slightly higher than that of the PBSNLR. The relatively large SDs during the rising phase of learning accuracy show that the stability of PBSNLR's learning performance is relatively low when the mean accuracy is not very high. However, the accuracy deviation of PBSNLR is very small when the mean accuracy is very high.


A
Learning performance (mean±SD of C)
C
number of synaptic inputs
PBSNLR
ReSuMe
Learning efficiency (learning epoch and learning time)
learning epoch
learning time[s]
epoch(R)
epoch(P)
time(R)
time(P)
B
Learning performance (mean±SD of C)
C
number of synaptic inputs
PBSNLR
ReSuMe
Learning efficiency (learning epoch and learning time)
learning epoch
learning time [s]
x 10^4
epoch(R)
epoch(P)
time(R)
time(P)
C
Learning performance (mean±SD of C)
C
number of synaptic inputs
PBSNLR
ReSuMe
Learning efficiency (learning epoch and learning time)
learning epoch
learning time [s]
x 10^4
epoch(R)
epoch(P)
time(R)
time(P)

The learning efficiency presents a situation similar to the previous experiments. PBSNLR's epochs are more than the ReSuMe's when the learning accuracy is not very high and less than the ReSuMe's when learning accuracy is very high. The PBSNLR's advantage on learning epoch with high learning accuracy is more obvious than that of ReSuMe with low learning accuracy. For computing time, the PBSNLR's is still significantly less than the ReSuMe's.

Comparing the three subfigures in Figure 6, we see that the increase of train length and the firing rate add to the difficulty of learning. More synaptic inputs are needed to achieve high learning accuracy. At the same time, the learning epochs that are needed to reach the highest accuracies and the computing time are also increased. But the increase of the two factors does not change the difference between the two methods.

4.1.2 Learning in the Presence of Noise. Some kinds of noise can significantly disturb the timing accuracy and reliability of the neural responses (Rieke, Warland, de Ruyter van Steveninck, & Bialek, 1997; Schneidman, 2001; van Rossum, O'Brien, & Smith, 2003). But some research results indicate that the nervous system can employ some anti-interference strategies to deal with the noise and produce accurate and reliable responses (Hertz, Krogh, & Palmer, 1991; Mainen & Sejnowski, 1995; Shmiel et al., 2005; Montemurro et al., 2007; Tiesinga, Fellous, & Sejnowski, 2008). Ponulak and Kasiński's (2010) experimental results show that ReSuMe can make SNs reproduce target sequences of spikes precisely and reliably even under highly noisy conditions by using noisy training.

Here we employ an experimental strategy similar to that of Ponulak and Kasiński (2010) to investigate the antinoise capability of PBSNLR. In the experiments, a neuron with 400 synaptic inputs is trained to emit a desired output spike train with a length of 1,000 ms. Every input spike train and the desired output spike train are Poisson spike trains with rates 10 Hz and 100 Hz, respectively. After training, the neurons are subjected to background noise simulated by injecting gaussian white-noise current to the neurons. The mean value of the current is assumed to be zero, and the variance of the current is assumed to be from 2 nA to 60 nA with an interval of 2 nA. The

Figure 6: Learning results and a comparison of learning performance between PBSNLR and ReSuMe when the number of synaptic inputs increases gradually. (A) Every input spike train and the desired output spike train are Poisson spike trains with the same length of 1000 ms and different rates of 10 Hz and 100 Hz, respectively. (B) Every input spike train and the desired output spike train are Poisson spike trains with the same length of 1000 ms and the same rate, 200 Hz. (C) Every input spike train and the desired output spike train are Poisson spike trains with the same length of 1600 ms and different rates of 10 Hz and 100 Hz, respectively.

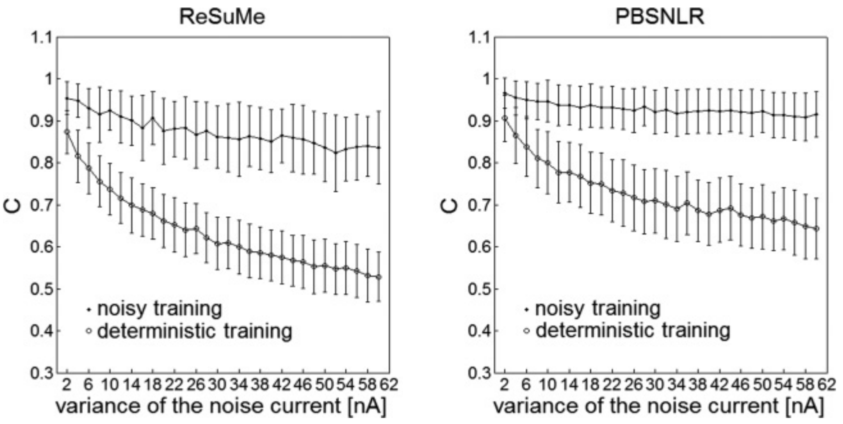


Figure 7: Average values and SDs of the measure *C* of the similarity between the desired output spike trains and the observed output spike trains of the test running after deterministic training and noisy training. The background white noise can affect the neural responses that have been learned via PBSNLR and ReSuMe (deterministic training). However, the neurons are robust to the given noise type if the training was performed in the presence of the same noise source (noisy training). Whether in deterministic training or in noisy training, PBSNLR has a stronger antinoise capability than ReSuMe.

training is performed (1) under noiseless conditions (deterministic training), which investigates the effect of the noise on the learning results, and (2) in the presence of the same noise source (noisy training), which investigates the anti-interference capability against background noise of PBSNLR. The experimental results are shown in Figure 7.

From Figure 4A, we know that the training accuracies of the two methods are both very high and almost the same in the first scenario. When the neurons are run with background noise after training, the noise has a great effect on the learning results of the two methods and greatly reduces the degree of correlation between the desired and the actual output spike trains (see Figure 7). The measure *C* of ReSuMe drops more obviously than that of PBSNLR. That is, the synaptic weights obtained by the deterministic training of PBSNLR are more robust than those of ReSuMe against the disruption of noise.

In the second scenario, the neurons are trained in the presence of noise. When the neurons are run with background noise after training, the measure *C* in the test running shows that the effect of the noise on the learning results of the two methods is significantly less than the effect in the first scenario. This indicates that PBSNLR can obtain anti-interference capability against background noise through noisy training, which is similar to the antinoise capability of ReSuMe shown in Ponulak and Kasiński

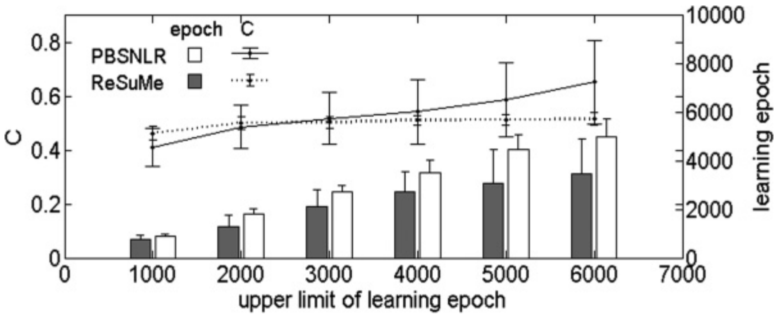


Figure 8: Learning results (learning accuracy and epochs) and a comparison of learning performance between PBSNLR and ReSuMe when the upper limit for the number of learning epochs increases gradually. A single neuron with 400 synaptic inputs is trained to emit a desired spike train with a length of 2000 ms and a firing rate of 200 Hz, the same as the firing rate of input spike trains.

(2010). Through observing the difference between the testing accuracy of noisy training in Figure 7, the antinoise capability through noisy training of PBSNLR is also stronger than that of ReSuMe. Even if the variance of noise is as much as 60 nA, PBSNLR can still get very high testing accuracy (measure *C* is more than 0.9), which indicates that the antinoise capability of PBSNLR through noisy training is very strong.

4.1.3 Learning with More Learning Epochs. In previous experiments, the upper limit for the number of learning epochs has been to 1000. However, more learning epochs will certainly affect the learning results, which will be investigated in an experiment in which a single neuron with 400 synaptic inputs is trained to emit a desired spike train with length 2000 ms. Every input spike train and the desired output spike train are Poisson spike trains with the same rate, 200 Hz. The upper limit for the number of learning epochs is increased from 1000 to 6000 with an interval of 1000. The experimental results are shown in Figure 8.

From Figure 8, we see that the mean value of PBSNLR's learning accuracy gradually increases with the increase of the upper limit of learning epochs, while the mean accuracy of ReSuMe almost does not change. This indicates that ReSuMe's learning accuracy can reach its upper limit with relatively fewer learning epochs and the number of epochs has a significant effect on PBSNLR's learning accuracy. In Figure 8, the SDs of PBSNLR's mean accuracies are much larger than those of ReSuMe. However, this result is in line with the learning result at 2000 ms in Figure 4B. The learning epochs of the two methods in Figure 8 still keep the comparison: PBSNLR has more learning epochs than ReSuMe. Therefore, the comparison results between PBSNLR and ReSuMe do not change obviously with more learning epochs and the choice of the epochs' upper limit in the previous experiments is reasonable.

In the previous experiments, we investigated the effect of some factors on the learning performance of PBSNLR. According to Figure 5, increasing the firing rates of only desired output spike trains will barely affect PBSNLR's learning accuracy and efficiency. Figures 4 and 6 show that the increase in the length of spike trains and the decrease in the number of synaptic inputs will affect the learning performance substantially. The two factors are associated: more synaptic inputs can maintain high learning accuracy on longer spike trains. As long as the length of spike trains is decreased to a certain degree or the number of synaptic inputs is increased to a certain degree, the learning accuracies of both PBSNLR and ReSuMe can increase to a very high value. PBSNLR is better able to make the learning accuracy increase, and it can obtain a high learning accuracy with longer spike trains or fewer synaptic inputs.

When the synaptic inputs are very few or the spike trains are very long, the learning accuracies of the two methods are not high; the ReSuMe's accuracy is slightly higher than that of PBSNLR, for two reasons. The first is that a single perceptron can solve only linearly separable problems. The decrease in the number of synaptic inputs decreases the input dimension of the training samples, and the increase in the length of spike trains increases the number of training samples. Both results will decrease the linear separability of the training samples and affect PBSNLR's learning accuracy. The second reason is that the upper limit for the number of learning epochs limits the further increase of the learning accuracy of PBSNLR, as Figure 8 shows. However, the application values of the learning methods are relatively low when learning accuracy is low, and they are better reflected in the ability to achieve high learning accuracy.

As to learning efficiency, we can see from the figures that the learning epochs that PBSNLR needed are slightly more than ReSuMe needed when the learning accuracies are rising or falling. When the learning accuracies reach very high values, the PBSNLR has fewer epochs than ReSuMe, with a relatively large amplitude. The PBSNLR's advantage of learning epochs is not very obvious. The obvious advantage of PBSNLR is reflected in learning time. PBSNLR spends significantly less learning time than ReSuMe except in very few situations when the spike trains are very short or the firing rates are very low. And the longer the spike trains are or the higher the firing rates are, the more obvious the advantage is.

Therefore, the main advantages of PBSNLR are higher learning accuracy, absolute advantage in learning time, and stronger antinoise capability.

4.2 Effect of the Number of Negative Samples. In the previous examples, every time point after the discretization of the neurons' runtime is selected to constitute the training samples of the SPNs. The advantage of this choice is that very high learning accuracy can be obtained; the disadvantage is that the number of samples is large and will increase the storage burden. Here we investigate the effect of reducing the number of training samples. Because the number of positive samples is fixed, only the number of negative samples can be changed. Selection of negative samples is done in a few steps. First, all of the one discrete times before the times of the desired output spikes are selected as the points of negative samples. For example, if the times of desired output spikes are 3 ms, 10 ms, and 15 ms, the one discrete times before them are, respectively, 2 ms, 9 ms, and 14 ms, which will be the discrete samples. If the set of the negative training sample is composed of only these points, the number of negative samples is the least and the same as the number of the positive samples. For the rest of the discrete time points, we investigate the points between every two adjacent desired output spikes in which some points are selected as the points of negative samples according to certain intervals. For example, if there are 10 time points between two adjacent desired output spikes and the interval is selected as 3, the third, sixth, and the ninth in the 10 points are selected as the points of negative samples. Here a simple selection method is given. The key to the selection is that the time points of negative samples should be uniformly distributed in the time without desired output spikes.

Three groups of experiments are carried out with 400 synaptic inputs. In the first group, every input spike train and the desired output spike train are Poisson spike trains with the same length, 1000 ms; and different rates of 10 Hz and 60 Hz, respectively. In the second group, every input spike train and the desired output spike train are Poisson spike trains with the same length, 2000 ms, and different rates of 10 Hz and 100 Hz, respectively. In the third group, every input spike train and the desired output spike train are Poisson spike trains with the same length, 1000 ms, and the same rate, 200 Hz. The number of negative samples begins from the number of all the discrete times except the times of desired output spikes. With the gradual increase of the selection interval of the negative samples between every adjacent desired output spikes, in turn selected as 2, 4, 6, . . . , 50 in the first group of experiments, 2, 4, 6, . . . , 40 in the second, and 2, 4, 6, . . . , 30 in the third, the number of the negative samples decreases gradually. Finally, the number of the negative samples is the same as the number of the positive samples. The experimental results are shown in Figure 9.

Learning accuracy decreases with the decrease in the number of negative samples, but the decrease is not significant (see Figure 9). The measure C can still lie over 0.7 in Figure 9A and over 0.8 in Figure 9C when the number of negative samples is the smallest, that is, only 5.5% in Figure 9A and 15.4% in Figure 9C of the maximum number. When the selection interval of the negative samples is 10 ms, the mean value of measure C is 0.635 in Figure 9B and 0.846 in Figure 9C, better than the learning results of ReSuMe in the previous experiments. At this time, the number of negative

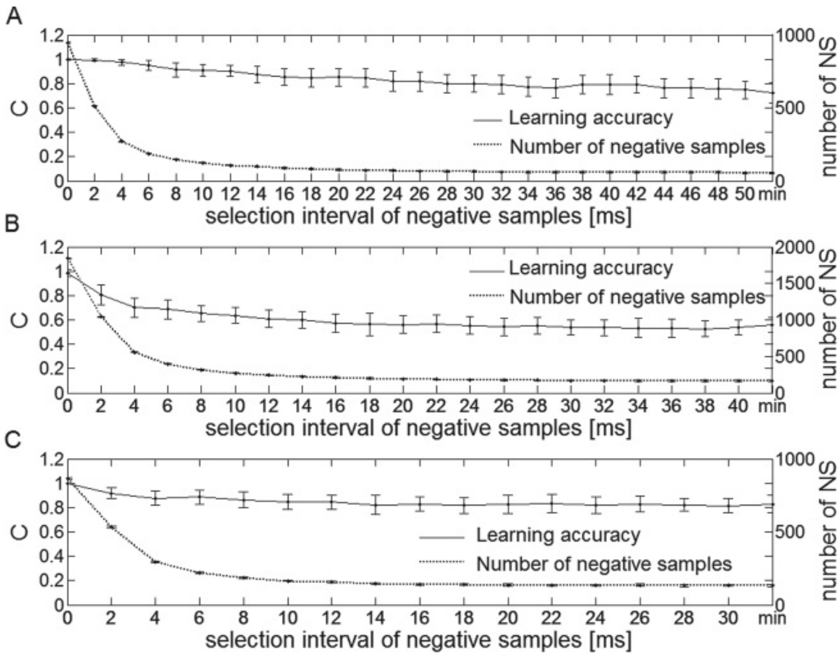


Figure 9: Learning accuracy and number of negative training samples (NS) when the number of negative training samples is gradually reduced, which is represented as the gradual increase of the selection interval of the negative samples. The "min" on the abscissa means the minimum number of negative samples, which are all of the one discrete times before the times of the desired output spikes. The selection interval "0" means the maximum number of negative samples, which are the discrete times expect the times of desired output spikes. The number of the synaptic inputs is 400. (A) Every input spike train and the desired output spike train are Poisson spike trains with the same length, 1000 ms, and different rates, 10 Hz and 60 Hz, respectively. (B) Every input spike train and the desired output spike train are Poisson spike trains with the same length, 2000 ms, and different rates, 10 Hz and 100 Hz, respectively. (C) Every input spike train and the desired output spike train are Poisson spike trains with the same length, 1000 ms, and the same rate, 200 Hz.

samples is only 15.8% in Figure 9B and 19.1% in Figure 9C of the maximum number. When the number of negative samples is about 19.8% in Figure 9A and 21.4% in Figure 9C of the maximum number, PBSNLR can make the measure C reach about 0.95 and 0.85, respectively, which indicates that the learning accuracies are high. Therefore, as long as the requirement of learning accuracy is slightly reduced, the requirement of storage space for PBSNLR can be greatly reduced. In addition, the decrease in the number of negative samples does not increase the SDs of the mean accuracies, so it does not affect the stability of PBSNLR's learning performance. Figure 9B shows that the decrease in the number of negative samples has a relatively great effect on the PBSNLR's learning performance when the length of the spike trains is relatively long.

4.3 Learning with LIF Neuron Model. This letter takes advantage of the intuitive SRM neuron model to introduce PBSNLR. If PBSNLR is limited only in the SRM model, its practicability will be largely restricted. According to the properties of PBSNLR, we can summarize what kinds of neuron models are suitable for it. Before training, the SNs are transformed into SPNs based on invariable states. Except the synaptic weights, the internal elements of the neurons will not be changed during the learning process. Therefore, if PBSNLR is to be applied to an SN model, all the information that the neuron can correctly emit—the desired output spike trains except the synaptic weights of the state—needs to be determined before training, which includes the PSPs and the values of the refractoriness function. It is difficult for the neuron models, which use differential equations as their internal state's expressions, such as LIF and HH models (Gerstner & Kistler, 2002), to achieve it.

Of course, it is a simple solution if other SN models can be transformed into an SRM model. Through solving differential equations, the internal states of the LIF neurons can be expressed intuitively, which can be regarded as the SRM form (Gerstner & Kistler, 2002; Shen et al., 2008). A simple experiment of the LIF neuron model with PBSNLR is carried out in which a single LIF neuron with 200 synaptic inputs is trained to emit a desired spike train with length 300 ms. Every input spike train and the desired output spike train are Poisson spike trains with different rates, 20 Hz and 80 Hz, respectively. The initial synaptic weights are selected as the uniform distribution in the interval $(0, 1.6 \cdot 10^{-3})$. In order to simulate the differential equation more precisely, the discrete precision is set to 0.01 ms. The LIF neuron is transformed into SRM form, and then the SRM neuron is trained by using PBSNLR. After training, PBSNLR can make the LIF neuron emit the desired output spikes within 200 learning epochs when the learning rate is selected as 0.5. The experimental result is shown in Figure 10.

In this experiment, a 0.01 ms time step is selected. The advantage of this choice is that the simulation precision of the neuron's internal state can be improved. The disadvantage is that the number of SPN's training samples is increased greatly, which will increase the memory consumption of the training samples and the training time. However, the lower time step does not affect the convergence of PBSNLR.

4.4 Classification. An important application area for neural networks is to solve classification problems. In the past decade, spiking neural networks have been successfully applied to various classification tasks. One kind of

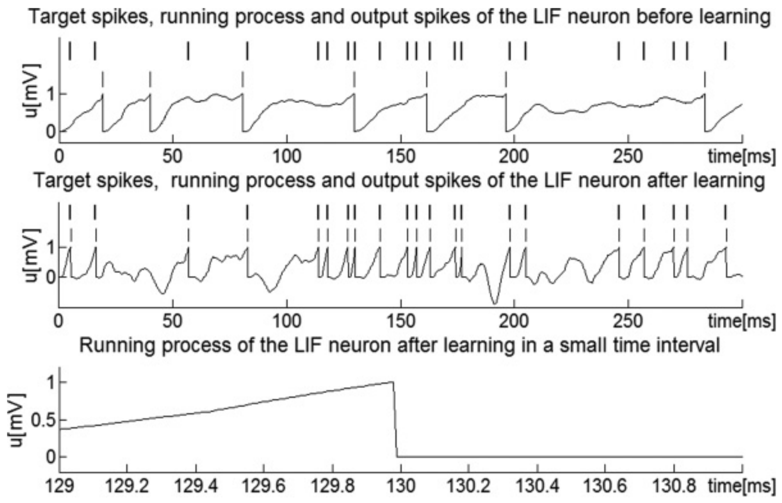


Figure 10: A single LIF neuron with 200 synaptic inputs is trained on an 80 Hz Poisson target spike train of length 300 ms. The discrete precision of running is set to 0.01 ms. Through transforming the LIF neuron to SRM form, PBSNLR can cause the neuron to emit the desired output spikes within 200 learning epochs. The lower discrete precision of time will increase the memory consumption of the training samples, but it does not affect the convergence of PBSNLR. When a lower time step is used the internal state of the neuron can be simulated more precisely.

the classification task is a practical problem, such as the XOR and Iris problems, which are solved by using multilayer spiking neural networks with very simple output (only one spike) (Bohte et al., 2002; Ghosh-Dastidar & Adeli, 2007, 2009). The other kind is the classification of spike trains emitted by a single neuron without practical application background (Gütig & Sompolinsky, 2006; Maass, Natschlager, & Markram, 2002; Ponulak & Kasiński, 2010). In this letter we use PBSNLR to solve a classification problem of spike trains, which can contribute to solving XOR problems. First, as a classification problem of spike trains, every input spike train is generated by choosing for each segment one of two spike templates T_1, T_2. The templates are divided into two equal segments S_1, S_2, and the two segments are exactly the same if they are regarded as two independent spike trains. The two templates with two segments can be combined into four complete input spike trains (see Figure 11A). As Figure 11A shows, if 0 is encoded by the segment S_1 of T_1 and 1 is encoded by the segment S_1 of T_2, the obtained four-input spike trains can be regarded as the spike coding of {0, 0}, {0, 1}, {1, 0}, and {1, 1}, the four inputs of the XOR problem. The classification objects are the input spike trains, among which the input spike trains

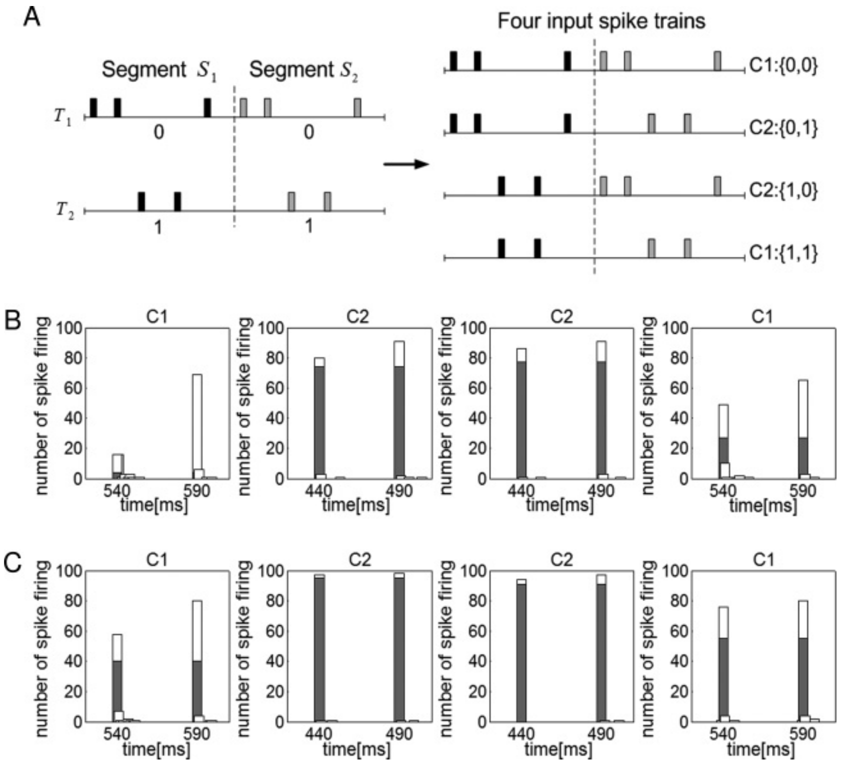


Figure 11: A classification task of spike trains that is similar to the XOR problem. (A) Generation of the input spike trains for the classification task. The four input trains consist of two segments (S_1, S_2), and each segment is drawn from one of two templates (T_1, T_2). The two segments of one template are completely overlapping, so the input trains can be regarded as the spike coding of the four inputs of the XOR problem. (B, C) The output results of the output neuron after training in the 100 experiments. The open histograms at the times of spikes show the distribution of the actual output spikes of the single output neuron (the number of the spikes belongs to the final outputs of the neuron in the 100 experiments) corresponding to the four input spike trains. The filled histograms in the open histograms at the times of the desired output spikes show the precise firing number of the two desired output spikes. (B) 200 neurons in the reservoir. (C) 800 neurons in the reservoir.

corresponding to {1, 1} and {0, 0} are one class (C1) and the input spike trains corresponding to {1, 0} and {0, 1} are the other class (C2). Thus the classification task of the spike trains is similar to the XOR problem. In other words, we try to solve the XOR problem by using the classification of spike trains.

Because every input of the classification problem is a single spike train, we employ the concept of reservoir computing with a reservoir network (Maass et al., 2002; Jaeger, Maass, & Principe, 2007; Ponulak & Kasiński, 2010) to perform the experiments. The recurrent network acts as a filter projecting the inputs into a higher-dimensional vector of spike trains. The structure of the reservoir network in this letter is a single input, a single output, and a noise-free reservoir consisting of a large number of SRM neurons (see the appendix for details on the network parameters). Only the synaptic weights between the reservoir and the output neuron are updated through learning.

The two spike templates are generated randomly. The segments S_1 of the two templates are selected as Poisson spike trains of 300 ms along with rate 20 Hz. The segments S_2 copy S_1, that is, all the spike times of S_1 plus 300 ms. The desired outputs of the output neuron corresponding to C1 and C2 are set to the spike firing times [540, 590] ms and [440, 490] ms, respectively.

Two reservoirs are used in this letter; the number of neurons in each is 200 and 800, respectively. One hundred classification experiments are carried out with each reservoir. In each experiment, the positive and negative samples transformed from the four input spike trains and their desired output spike trains are combined into a positive sample set and a negative sample set of the SPN that is transformed from the output neuron. The results of the experiments are shown in Figures 11B and 11C.

Figures 11B and 11C show the distribution of the actual output spikes of the single-output neuron corresponding to the four input spike trains (the open histograms at the firing times of spikes). The filled histograms in the open histograms at the times of the desired output spikes show the precise firing number of the two desired output spikes. For example, as the third panel from the left in Figure 11C shows, all of the actual output spikes of the output neuron after learning in the 100 experiments are 440 ms (94 times), 490 ms (97 times), 492 ms (1 time), and 501 ms (1 time), among which the completely correct firing of the two desired output spikes is 91 times. If the supervised learning is considered only, PBSNLR can obtain good learning results. Most of the desired output spikes are emitted about 80 times in 100 experiments. The convergence of the neuron's output is also good. There are fewer undesired output spikes, and all of them are around the desired output spikes. It can be found from the desired output spikes that the learning results in Figure 11C are better than the learning results in Figure 11B. Among all the experiments with a 200 neuron reservoir, only 2% training can make the output neuron precisely emit the desired output spikes of the four inputs. That is, not much of the learning is completely successful, whereas when the number of neurons in the reservoir is increased to 800, 31% of the training is completely successful. The reason is that the XOR problem is not a linearly separable problem, and only the synaptic weights between the reservoir and the output neuron are adjusted by PBSNLR; that is, the learning involves only a single output neuron. Even under multispike conditions, the classification learning of the linearly inseparable problems does not easily achieve perfect results by using a single SN. However, the reservoir increases the input dimension of the neuron, which makes it possible to transform the XOR problem into a linearly separable problem. More neurons in the reservoir lead to a higher probability of this transformation.

Although the learning success rate is not high, the classification results can still be good. If there are small errors between the actual output and the desired output spikes of the output neuron after learning, the purpose of classification can still be achieved. For example, if the actual firing times of output spikes are, for C1 [540 ms], [540, 590] ms; C2: [440, 490, 540] ms, [438, 490] ms, the two classes can still be correctly classified. Therefore, if the allowed error of the spike number is 1 (one more or less) and the allowed error of the spike time is 5 ms, the success rates of classification for 200 and 800 reservoir neurons are 90% and 98%, respectively, and the number of the learning epochs that the neuron needs to obtain correct classification is about 130 in the two cases. This illustrates that although the training of a single SN with multispike output has relative difficulty obtaining perfect output results for the linear inseparable problems, the flexibility of the multiple spikes' number and time provides the possibility of successful classification.

5 Discussion and Conclusion

In the experiments, the PBSNLR learning time is much shorter than that of ReSuMe. The difference in the learning time between PBSNLR and ReSuMe is due to the difference in the operation of the two methods—offline versus online. PBSNLR is offline, and ReSuMe is online. The reason that the learning time is so much shorter for PBSNLR than ReSuMe is based on what follows. For PBSNLR, the original SN need not be run in every learning epoch; the training samples of the SPN will be trained according to the perceptron learning rule, which includes the weighted sum of inputs and the weight update (weights plus or minus the input of sample) when misclassification occurs. For ReSuMe, the SN needs to be run in every learning epoch, and the weights will be updated as soon as a spike is emitted or a desired output spike is encountered. In the process of running an SN, the PSPs need to be first calculated according to the input spikes and the spike response function; then the weighted sum of the PSPs is calculated. The weight update is calculated according to the related spikes and the learning window function. All of these computations are much more complex than the computation of the perceptron learning rule.

Compared to ReSuMe, another advantage of PBSNLR is that the adjusting parameter of PBSNLR except the parameters of the neurons themselves is the learning rate only, while the learning windows of ReSuMe have multiple adjusting parameters that will affect its learning performance significantly. These parameters must be set carefully in the process of learning, which will increase the application difficulty and complexity of ReSuMe. Only one adjusting parameter makes the application of PBSNLR much simpler.

In section 4, we compared the learning performance of PBSNLR and ReSuMe; here we discuss some similarities and differences among PBSNLR, ReSuMe, and other supervised learning methods. Through comparing equation 3.8 and the weight update rule of ReSuMe in the appendix, if d^{t_d}, a^{t_d}, and $\boldsymbol{P}^{t_d}$ respectively correspond to $S^d(t)$, $S^o(t)$ and the second term in the bracket, the weight update rules of the two methods are very similar in form. This similarity is also reflected in the comparison between PBSNLR and the method in Pfister et al. (2006) or its generalization in Brea, Senn, and Pfister (2011). In fact, the weight update rules of some supervised learning methods for spiking neural networks have a certain similarity. The reason for the similarity is that the weight update of one synapse is usually based on information on the synapse. The available information includes the PSPs induced by the spikes, which are input along the synapse (PBSNLR, SpikeProp, Pfister and Brea's methods, Tempotron (Gütig & Sompolinsky, 2006)) and the weight update rule of STDP on the synapse (ReSuMe). Moreover, the two kinds of information are also somewhat similar.

Although the weight update rules of the methods are similar, their derived methods and operation mechanisms are very different. PBSNLR is derived from the classification of the neuron's running time and the perceptron learning rule. It is an offline algorithm. ReSuMe is derived from the Widrow-Hoff rule. Pfister's and Brea's methods are based on statistical methods, and their learning rules are derived by optimizing the likelihood of postsynaptic firing at one or several desired firing times. The three methods are performed online. These differences lead to the differences in the learning performance of PBSNLR and other methods.

For a supervised learning problem of an SN, whether the desired output spike train is learnable often cannot be determined. PBSNLR is based on the perceptron learning rule, with which a single perceptron can solve linearly separable problems. In theory, if the positive samples and the negative samples of the SPN derived from the input spike trains and the desired output spike train of the SN are not linearly separable, the desired output spike train cannot be learned successfully. Two simplified supervised learning problems of an SN are given in Figure 12 to illustrate this conclusion. In Figure 12A, the specific spike response function of the neuron shows that the value of PSP is 1 when the interval between the current time and the time of an input spike is 0 ms and the value of PSP is 0.5 when the interval is 1 ms. We ignore the time length but reserve the effect of the absolute refractory period in the example. As Figure 12B shows, the classification problem in the right panel is not linearly separable (The open and filled circles are the PSPs corresponding to the two input synapses at the times of the undesired and the desired output spikes, respectively), so the desired output spike train of the original sequence learning problem in the left panel cannot be

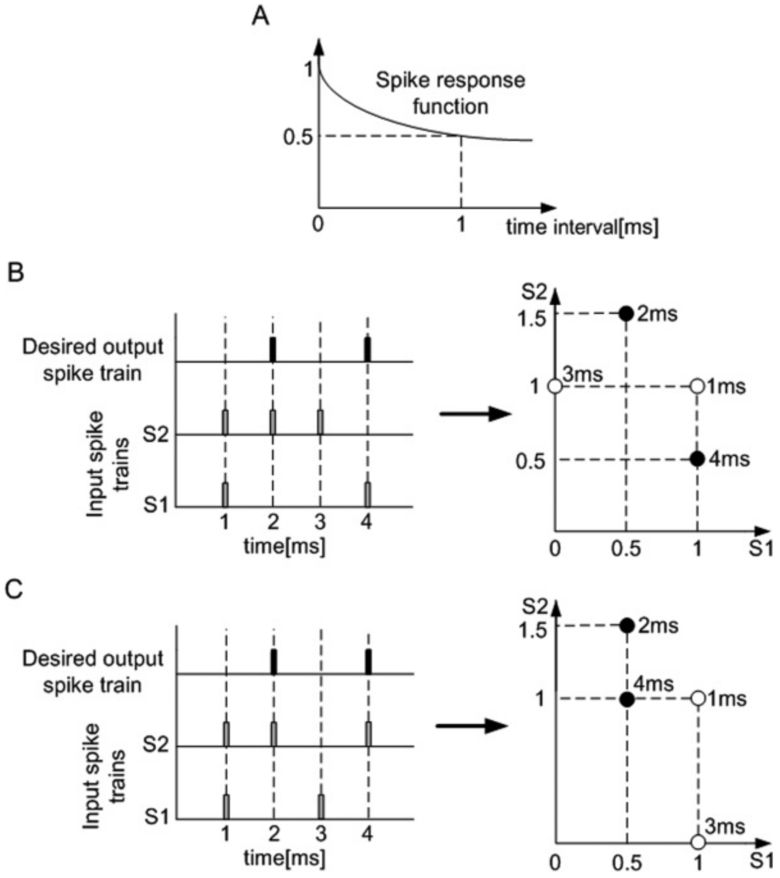


Figure 12: The relation between the linear separability of the SPN's classification problem and the learnability of the supervised learning problem of an SN. (A) The spike response function of the neuron. (B) A simple supervised learning problem whose desired output spike train is not learnable. The positive samples at spike times of 2 ms and 4 ms (filled circles) and the negative samples at the spike times 1 ms and 3 ms (open circles) are not linearly separable, so the desired output spike train cannot be learned successfully. (C) A simple supervised learning problem whose desired output spike train is learnable. The positive samples at spike times 2 ms and 4 ms (filled circles) and the negative samples at the spike times 1 ms and 3 ms (open circles) are linearly separable, so the desired output spike train can be learned successfully.

emitted correctly by the neuron through learning. Figure 12C shows the other supervised learning problem, in which the desired output spike train is learnable.

However, it is not easy to directly determine whether the classification problem of the SPN derived from a supervised learning problem of an SN is linearly separable when the number of inputs synapsed is large. It is an application limitation of perceptrons that a single perceptron with the perceptron learning rule can solve only linearly separable problems, which will also have an effect on PBSNLR. For perceptrons, the linear separability of classification problems will increase as the input dimensions increase, so the learning performance of PBSNLR will increase when the number of synaptic inputs of neurons is increased. From Figure 6, a very large number of synaptic inputs is not necessary for relatively high learning accuracy. Therefore, although the distribution of the classification samples derived from the input and desired output spike trains of an SN is very complex, the linear separability of the samples is not very low when the input dimension is relatively large. According to the research findings of biological neurons, we know that a neuron in the brain exchanges information with other neurons via large numbers (thousands or more) of synapses. So there are usually relatively many synaptic inputs in the actual supervised learning problems of SNs. Therefore, it does not significantly limit PBSNLR to solving problems that single perceptrons can solve only if they are linearly separable, and PBSNLR still has a high practicality.

In our experiments, the selection of positive and negative training samples is based on the discretization of the neurons' running time. Although this strategy is not biologically plausible, it is a universal simulation method of SNs at this time, and the discrete running process can be infinitely close to the continuous running process of neurons by adjusting the discrete precision. In addition, the discretization strategy is not necessary. The time points of training samples of an SPN can be selected from continuous time, among which the selection method of positive samples is fixed and the selection method of negative samples is flexible. The time points of negative samples can be randomly selected from the continuous time according to certain rules. The more the negative samples are selected, the higher the learning accuracy is, but the longer training time and more storage space are needed. On the contrary, the less the negative samples are selected, the shorter the training time and less storage space are needed, but the lower the learning accuracy is. It can be known from the continuity of the neuron's internal state and the discontinuity of the spike times that the training samples of the SPN can perfectly match an input and output process of the original SN when the intervals between the time points of the negative samples are small enough.

Compared with the existing learning methods, a disadvantage of PBSNLR is that it requires relatively large storage space because all the PSPs induced by every synapse at the times of all the training samples need to be calculated and stored before training. The size of the storage space is determined by the length of the desired spike train, the number of input synapses, and training samples. Other learning methods calculate the PSPs in real time with the running of neurons in every learning epoch, so they do not need to store these quantities. For this disadvantage, we offer three considerations. First, with the rapid development of computer technology, large capacity storage has been very cheap. Therefore, the storage cost is not very high when PBSNLR is used to solve practical problems. Second, compared to the increase of storage space, increased learning efficiency is more significant. In other words, our method uses the storage space in exchange for substantial savings in calculation time. Third, the requirement of storage space can be reduced through some technical means. For instance, the number of negative samples can be decreased on the basis of meeting the requirement of learning accuracy, or the sparse matrix can be used to store the input vectors of SPNs so as to save storage space when the firing rates of the input and output spike trains of SNs are relatively low, because there will be a lot of 0 elements in the vectors.

The main purpose of this letter is to better solve the problem of spike sequence learning by putting forward the PBSNLR algorithm and comprehensively measuring its learning performance, which is the basis of further research on this area. A two-class classification problem of spike trains is considered in the application section. However, the firing times of multispikes can indicate multiclasses, so it is conceptually possible for spike sequence learning to solve multiclass classification problems. The ability of multiclass classification by PBSNLR is well worth exploring in our future work.

Finally, we discuss the biological plausibility of PBSNLR, which can be embodied by two aspects. The first is the perceptron learning rule, the basis of PBSNLR. As the first learning method for artificial neural networks, the perceptron learning rule is based on the Hebb rule (Hebb, 1949) and has a reasonable biological explanation. The second is the offline operation manner. PBSNLR is not an online algorithm. However, online is not the only way of operating in the nervous system. Research has shown that sleep plays an important role in learning and the memory's reprocessing in the brain. In sleep, these activities of the nervous system work in an offline manner (Stickgold, Hobson, Fosse, & Fosse, 2001), which can be regarded as the biological basis of the offline way that PBSNLR works. Because of this PBSNLR does not need to run spiking neurons in every learning epoch, which leads to very high learning efficiency. In addition, PBSNLR can obtain higher learning accuracy than the existing learning methods in most cases. Learning accuracy and efficiency are two extremely important performance indexes of the supervised learning methods for SNs and play a key role in practical applications. Therefore, PBSNLR makes up for the main disadvantages of the existing supervised learning methods for SNs to a large degree and is significant to the practical application and theoretical research of SNs.

Appendix: Details of Models and Simulations

A.1 SRM Model. In this letter, the concrete SRM neurons used in the experiments are relatively simple. The simplification is mainly reflected in the effect that the most recent output spike $t^{(f_r)}$ has on the internal state of the neurons. As mentioned in section 2, a neuron will successively enter absolute and relative refractory periods after firing. The absolute refractory period is often reflected through the change of firing threshold. The threshold will suddenly become very large to prevent the neuron from firing again. Because the firing threshold in this letter is set to a fixed value, the length of the absolute refractory period is artificially set to R_a. It is assumed that the neuron is insensitive in the absolute refractory period; that is, it does not respond to any input. Therefore, after the most recent output spike $t^{(f_r)}$, the internal state of the neuron must be calculated by using the input spikes whose arrival times are after $t^{(f_r)} + R_a$ and $t^{(f_r)} + R_a$ is set to 0 when no spike has been fired. In addition, the effect that $t^{(f_r)}$ has on the spike response function ε and the kernel function κ is reflected in the refractoriness function η; thus, the expression of the two functions can be simplified.

In this letter, the spike response function is expressed as

$$\varepsilon(t - t_i^{(g)}) = \begin{cases} \dfrac{t - t_i^{(g)}}{\tau} \exp\left(1 - \dfrac{t - t_i^{(g)}}{\tau}\right) & t - t_i^{(g)} > 0, \\ 0 & t - t_i^{(g)} \leq 0, \end{cases} \tag{A.1}$$

where τ is the time decay constant that determines the spread shape of the function. The kernel function of the external input current is expressed as

$$\kappa(s) = \begin{cases} \dfrac{R}{\tau_m} \exp\left(1 - \dfrac{s}{\tau_m}\right) & s > 0, \\ 0 & s \leq 0, \end{cases} \tag{A.2}$$

where τ_m is the time constant that determines the spread shape of the function and R is the input resistance. The refractoriness function is expressed as

$$\eta(t - t^{(f_r)}) = \begin{cases} -\eta_0 \exp\left(\dfrac{t - t^{(f_r)}}{\tau_R}\right) & t - t^{(f_r)} > 0, \\ 0 & t - t^{(f_r)} \leq 0, \end{cases} \tag{A.3}$$

where η_0 is a scale factor for the refractory function and τ_R is the time decay constant that determines the spread shape of the refractoriness function.

Table 1: Parameters of the SRM Neuron.

Parameter	τ	η_0	τ_R	τ_m	R	ϑ
Value	7 ms	$2 \cdot 10^{-3}$	80 ms	4 ms	0.2 MΩ	1 mV

In the experiments, the parameter values of the neurons are listed in Table 1. The length of the absolute refractory period R_a is selected as 1 ms.

In addition, because the computer program is used to simulate the SNs, the neurons need to be run in discrete mode (i.e., the running time of the neurons need to be discretized). The discrete precision of SRM neurons is set to 1 ms except the experiment in Figure 10.

There are two kinds of synapses of biological neurons: excitatory and inhibitory. They can be distinguished by the sign of weights in the expression of the internal state of a SN: a positive sign represents the excitatory synapse, and a negative sign represents the inhibitory one. Like Gütig and Sompolinsky (2006) and Ponulak and Kasiński (2010), we stipulate that the two kinds of synapses can transform into each other in the experiments, that is, the sign of weights can be changed during the learning process.

A.2 LIF Model. The LIF neuron model used in this letter is:

$$\tau_m \frac{du(t)}{dt} = -u(t) + R_m \sum_j \frac{w_j}{\tau_s} \sum_f H\left(t - t_j^f\right) e^{-\frac{t - t_j^f}{\tau_s}},$$

where $H(t)$ is the Heaviside step function. From Gerstner and Kistler (2002) or Shen, Lin, and De Wilde (2008), the corresponding SRM model of the LIF model can be expressed as

$$\begin{aligned} u(t) &= u_1(t) + u_2(t), \\ u_1(t) &= C \cdot \frac{R_m}{\tau_s - \tau_m} \sum_j w_j \sum_f e^{\frac{t_j^f}{\tau_s} - \frac{t}{\tau_m}}, \\ u_2(t) &= \frac{R_m}{\tau_s - \tau_m} \sum_j w_j \sum_f \left[H\left(t - t_j^f\right) e^{-\frac{t - t_j^f}{\tau_s}} - H\left(t - t_j^f\right) e^{-\frac{t - t_j^f}{\tau_m}} \right], \end{aligned}$$

where C is the integral constant. The parameter values used in our simulation are: $\tau_s = 5$ ms, $\tau_m = 4$ ms, $R_m = 0.4\ M\Omega$.

Table 2: Learning Rates of the Experiments.

	ReSuMe	PBSNLR
Figure 4A	0.3(200 ms)–0.02(3600 ms)	0.5(200 ms)–0.02(3600 ms)
Figure 4B	0.02(200 ms)–0.0008(2800 ms)	0.04(200 ms)–0.0008(2800 ms)
Figure 5	0.2(20 Hz)–0.004(700 Hz)	0.4(20 Hz)–0.006(700 Hz)
Figure 6A	0.04(100)–0.3(400)	0.04(100)–0.5(400)
Figure 6B	0.004(100)–0.008(500)	0.006(100)–0.01(500)
Figure 6C	0.04(100)–0.1(500)	0.04(100)–0.3(500)
Figure 7	0.3	0.5
Figure 8	0.001	0.001
Figure 9		0.5(A), 0.08(B), 0.008(C)
Figure 11		0.01

A.3 Learning Rules and Parameters of ReSuMe Algorithm. The following form of ReSuMe algorithm is used in this letter:

$$\frac{d}{dt}w(t) = [S^d(t) - S^o(t)]\left[a + \int_0^\infty W(s)S^{in}(t-s)ds\right],$$

with the learning window $W(s) = +A_+ \cdot \exp\left(-s/\tau_+\right)$ for $s > 0$, $W(s) = 0$ elsewhere. The parameter values used in our simulations are $a = 0.001$, $+A_+ = 0.5$, $\tau_+ = 5$ ms.

A.4 Experimental Strategy Details. Because this letter stipulates that the SNs do not respond to any input and do not fire spikes in the absolute refractory period, the discrete times that belong to the absolute refractory period can be ignored when the SNs are transformed into SPNs. We stipulate that the input vector and the desired output of a training sample of the SPNs are 0 vector and 0, respectively, when the time point falls into the absolute refractory period.

In Figures 4 to 9, the initial synaptic weights are selected as the uniform distribution in the interval $(0, 0.2 \cdot 10^{-3})$. The learning rates of the two methods in the experiments are listed in Table 2.

In Figures 4, 5, and 6, 50 experiments are carried out with two learning methods under each experimental case. The average values of the maximum that the measure C can reach through training and the average values of epochs that are needed to reach the maximum of C are calculated. The SDs of the two mean values are also calculated. In addition, the computing times that the two methods take to complete 50 experiments are recorded to compare.

In Figure 7, 50 experiments are carried out with two learning methods under each variance of the current noise. After training, 20 groups of noise (different noise in noisy training) are injected to the neurons, and the average values and the SDs of the measure *C*s of the test running are calculated.

In Figure 8, 50 experiments are carried out with two learning methods under each upper limit for the number of learning epochs. The average values and the SDs of the learning accuracies and the learning epochs that are needed to reach the maximum of measure *C*s in each case are calculated.

In Figure 9, 50 experiments are carried out under each selection interval of negative samples. The average values and the SDs of the learning accuracies and the number of the negative samples in each case are calculated.

The term for external input currents in expression of the neuron's internal state is omitted in Figures 3 to 6, 8, and 9. In Figures 3 to 8 and 10, every discrete time point after the discretization of the running time is selected to constitute the training samples of the SPNs.

In Figures 4 to 9, various experiments are carried out with various input and output patterns. In the experiments, the learning performance of PBSNLR and ReSuMe needs to be compared, so the learning accuracy during the learning process is recorded and the highest accuracy is selected for a comparison. Unless the learning can make the neurons precisely emit the desired output spikes, the experiments will stop at the upper limit for the number of learning epochs. The upper limit for the number of learning epochs is set to 1000 in Figures 4 to 7 and 9.

For ReSuMe, the SNs are run in every learning epoch, so the measure *C* can be easily recorded and investigated during the learning process. Because the original SNs need not be run after every learning epoch of PBSNLR, the measure *C* cannot be directly investigated during the learning process. Therefore, except Figure 3, we record and investigate the number of training samples that are misclassified by the SPNs during the learning process. The synaptic weights that can obtain the minimum number of misclassifications are selected to run the original SNs, and the measure *C*s are calculated as the learning performance plotted in the figures.

A.5 Details to the Reservoir Network. In the classification, we used reservoir networks with the parameters listed below.

The reservoirs were $4 \times 5 \times 10$ neurons and $8 \times 5 \times 20$ neurons. Parameters of connections from inputs to the reservoir are as follows: the probability of connections from the input neuron to any reservoir neuron $p = 0.5$, all the connections are excitatory, and the strength of synaptic connections selected as the uniform distribution is in the interval $(0, 1.2 \cdot 10^{-3})$.

Eighty percent of the connections between the neurons in the reservoir are excitatory, and the absolute values of the synaptic weights are selected as the uniform distribution in the interval $(0.23 \cdot 10^{-3}, 0.28 \cdot 10^{-3})$ (200 neuron reservoir) and $(0.21 \cdot 10^{-3}, 0.26 \cdot 10^{-3})$ (800 neuron reservoir). The probability of a synaptic connection from neuron a to neuron b and from b to a is defined as $C_{scale} \cdot c \cdot \exp(-D(a, b)^2/\lambda)$, where C_{scale}, c, and λ are positive constants and $D(a, b)$ is the Euclidean distance between neurons a and b.

Here $C_{scale} = 10$, $\lambda = 2.5$, and depending on whether neurons a and b are excitatory (E) or inhibitory(I), the value of c is 0.3 (EE), 0.2 (EI), 0.4 (IE), or 0.1 (II).

All reservoir neurons are connected to the output neuron, and only the synaptic weights between the reservoir and the output neuron are adjusted using PBSNLR. The initial synaptic weights are selected as the uniform distribution in the interval $(0, 0.2 \cdot 10^{-3})$.

The synaptic delays are uniformly selected from the set {1, 2, 3, 4, 5, 6} ms.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under grant 60971088.

References

Adrian, E. D., & Zotterman, Y. (1926). The impulses produced by sensory nerve endings: Part II: The response of a single end organ. *Journal of Physiology*, *61*, 151–171.

Belatreche, A., Maguire, L. P., McGinnity, M., & Wu, Q. X. (2003). A method for supervised training of spiking neural networks. In *Proceedings of IEEE Cybernetics Intelligence—Challenges and Advances (CICA) 2003* (pp. 39–44). Piscataway, NJ: IEEE.

Bi, G. Q., & Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, *18*, 10464–10472.

Bohte, S. M., Kok, J. N., & La Poutré, J. A. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*, 17–37.

Bose, N. K., & Liang, P. (1996). *Neural network fundamentals with graphs, algorithms and applications*. New York: McGraw-Hill.

Brea, J., Senn, W., & Pfister, J. P. (2011). Sequence learning with hidden units in spiking neural networks. In J. Shawe-Taylor, R. S. Zemal, P. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 24* (pp. 1422–1430). Red Hook, NY: Curran.

Broomhead, D. S., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, *2*, 321–355.

Carnell, A., & Richardson, D. (2005). Linear algebra for time series of spikes. In *Proceedings of European Symposium on Artificial Neural Networks, ESANN' 2005* (pp. 363–368). Brussels: d-side.

Florian, R. V. (2007). Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, *19*, 1468–1502.

Florian, R. V. (2012). The chronotron: A neuron that learns to fire temporally precise spike patterns. *PLoS ONE*, *7*, e40233. doi:10.1371/journal.pone.0040233.

Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge: Cambridge University Press.

Ghosh-Dastidar, S., & Adeli, H. (2007). Improved spiking neural networks for EEG classification and epilepsy and seizure detection. *Integrated Computer-Aided Engineering*, *14*, 187–212.

Ghosh-Dastidar, S., & Adeli, H. (2009). A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks*, *22*, 1419–1431.

Gütig, R., & Sompolinsky, H. (2006). The tempotron: A neuron that learns spike timing-based decisions. *Nature Neuroscience*, *9*, 420–428.

Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley.

Hertz, J., Krogh, A., & Palmer, R. (1991). *Introduction to the theory of neural networks*. Reading, MA: Addison-Wesley.

Jaeger, H., Maass, W., & Principe, J. (2007). Special issue on echo state networks and liquid state machines: Editorial. *Neural Networks*, *20*, 287–432.

Kandel, E., Schwartz, J., & Jessel, T. (1991). *Principles of neural science*. New York: Elsevier.

Kasiński, A., & Ponulak, F. (2006). Comparison of supervised learning methods for spike time coding in spiking neural networks. *International Journal of Applied Mathematics and Computer Science*, *16*, 101–113.

Kistler, W. (2002). Spike-timing dependent synaptic plasticity: A phenomenological framework. *Biological Cybernetics*, *87*, 416–427.

Kroese, B., & van der Smagt, P. (1996). *An introduction to neural networks*. (8th ed.). Amsterdam: University of Amsterdam.

Legenstein, R., Naeger, C., & Maass, W. (2005). What can a neuron learn with spike-timing-dependent plasticity? *Neural Computation*, *17*, 2337–2382.

Legenstein, R., Pecevski, D., & Maass, W. (2008). A learning theory for reward-modulated spike timing-dependent plasticity with application to biofeedback. *PLoS Computational Biology*, *4*, 1–27.

Maass, W. (1997a). Fast sigmoidal networks via spiking neurons. *Neural Computation*, *9*, 279–304.

Maass, W. (1997b). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, *10*, 1659–1671.

Maass, W. (1997c). Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. In M. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems, 9* (pp. 211–217). Cambridge, MA: MIT Press.

Maass, W., Natschlager, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*, 2531–2560.

Mainen, Z. F., & Sejnowski, T. J. (1995). Reliability of spike timing in neocortical neurons. *Science*, *268*, 1503–1506.

Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, *275*, 213–215.

McKennoch, S., Liu, D., & Bushnell, L. G. (2006). Fast modifications of the spikeprop algorithm. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 3970–3977). Piscataway, NJ: IEEE.

Minsky, M., & Papert, S. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

Montemurro, M. A., Panzeri, S., Maravall, M., Alenda, A., Bale, M. R., Brambilla, M., et al. (2007). Role of precise spike timing in coding of dynamic vibrissa stimuli in somatosensory thalamus. *Journal of Neurophysiology*, *98*, 1871–1882.

Pfister, J.-P., Toyoizumi, T., Barber, D., & Gerstner, W. (2006). Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Computation*, *18*, 1318–1348.

Ponulak, F. (2005). *ReSuMe-new supervised learning method for spiking neural networks* (Tech. Rep.). Poznan: Institute of Control and Information Engineering, Poznan University of Technology.

Ponulak, F., & Kasiński, A. (2010). Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification and spike-shifting. *Neural Computation*, *22*, 467–510.

Rieke, F., Warland, D., de Ruyter van Steveninck, R., & Bialek, W. (1997). *Spikes: Exploring the neural code*. Cambridge, MA: MIT Press.

Rojas, R. (1996). *Neural networks: A systematic introduction*. Berlin: Springer-Verlag.

Ruf, B., & Schmitt, M. (1997). Learning temporally encoded patterns in networks of spiking neurons. *Neural Processing Letters*, *5*, 9–18.

Rullen, R. V., Guyonneau, R., & Thorpe, S. J. (2005). Spike times make sense. *Trends in Neurosciences*, *28*, 1–4.

Rullen, R. V., & Thorpe, S. J. (2001). Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *neural computation*, *13*, 1255–1283.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Schneidman, E. (2001). *Noise and information in neural codes*. (doctoral dissertation, Hebrew University, Jerusalem, Israel).

Schrauwen, B., & Campenhout, J. V. (2004). Extending spikeprop. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 471–476). Piscataway, NJ: IEEE.

Schrauwen, B., & Campenhout, J. V. (2006). Backpropagation for population-temporal coded spiking neural networks. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 1797–1804). Piscataway, NJ: IEEE.

Schreiber, S., Fellous, J. M., Whitmer, D., Tiesinga, P., & Sejnowski, T. J. (2003). A new correlation-based measure of spike timing reliability. *Neurocomputing*, *52–54*, 925–931.

Shen, X., Lin, X., & De Wilde, P. (2008). Oscillations and spiking pairs: Behavior of a neuronal model with STDP learning. *Neural Computation*, *20*, 2037–2069.

Shmiel, T., Drori, R., Shmiel, O., Ben-Shaul, Y., Nadasdy, Z., Shemesh, M., et al. (2005). Neurons of the cerebral cortex exhibit precise interspike timing in correspondence to behavior. *Proceedings of the National Academy of Sciences of USA*, *102*, 18655–18657.

Silva, S. M., & Ruano, A. E. (2005). Application of Levenberg-Marquardt method to the training of spiking neural networks. In *Proceedings of the International Conference on Neural Networks and Brain* (Vol. 3, pp. 1354–1358). Piscataway, NJ: IEEE.

Stickgold, R., Hobson, J. A., Fosse, R., & Fosse, M. (2001). Sleep, learning, and dreams: Off-line memory reprocessing. *Science*, *294*, 1052–1057.

Theunissen, F., & Miller, J. P. (1995). Temporal encoding in nervous systems: A rigorous definition. *Journal of Computational Neuroscience*, *2*, 149–162.

Tiesinga, P., Fellous, J.-M., & Sejnowski, T. J. (2008). Regulation of spike timing in visual cortical circuits. *Nature Reviews Neuroscience*, *9*, 97–107.

van Rossum, M. C., O'Brien, B. J., & Smith, R. G. (2003). Effects of noise on the spike timing precision of retinal ganglion cells. *Journal of Neurophysiology*, *89*, 2406–2419.

Victor, J. D., & Purpura, K. P. (1996). Nature and precision of temporal coding in the visual cortex: A metric-space analysis. *Journal of Neurophysiology*, *76*, 1310–1326.

Xin, J., & Embrechts, M. J. (2001). Supervised learning with spiking neural networks. In *Proceedings of the International Joint Conference on Neural Networks* (Vol. *3*, pp. 1772–1777).

Received July 24, 2012; accepted January 3, 2013.