

Supervised Learning Algorithm for Spiking Neurons Based on Nonlinear Inner Products of Spike Trains

Xiangwen Wang, Xianghong Lin^(✉), Jichang Zhao, and Huifang Ma

School of Computer Science and Engineering,
Northwest Normal University, Lanzhou 730070, China
linxh@nwnu.edu.cn

Abstract. Spiking neural networks are shown to be suitable tools for the processing of spatio-temporal information. However, due to their intricately discontinuous and implicit nonlinear mechanisms, the formulation of efficient supervised learning algorithms for spiking neural networks is difficult, which has become an important problem in the research area. This paper presents a new supervised, multi-spike learning algorithm for spiking neurons, which can implement the complex spatio-temporal pattern learning of spike trains. The proposed algorithm firstly defines nonlinear inner products operators to mathematically describe and manipulate spike trains, and then derive the learning rule from the common Widrow-Hoff rule with the nonlinear inner products of spike trains. The algorithm is successfully applied to learn sequences of spikes. The experimental results show that the proposed algorithm is effective for solving complex spatio-temporal pattern learning problems.

Keywords: Spiking neural networks · Supervised learning · Nonlinear inner products of spike trains · Widrow-Hoff rule

1 Introduction

Recent advances in neurosciences have revealed that neural information in the brain is encoded through precisely timed spike trains, not only through the neural firing rate [1]. Spiking Neural Networks (SNNs) are often referred to as the new generation of neural networks. They have more powerful computing capacity to simulate a variety of neuronal signals and approximate any continuous function, and have been shown to be suitable tools for the processing of spatio-temporal information [2]. Supervised learning in SNNs involves a mechanism of providing the desired outputs with the corresponding inputs. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are calculated to control the synaptic weight adjustment. This process occurs over and over until the synaptic weights converge to certain values. When the sample conditions are changed, synaptic weights can be modified to adapt to the new environment after supervised learning. The purpose of supervised learning with temporal encoding for spiking neurons is to make the neurons emit arbitrary spike trains in response to given synaptic inputs.

At present, researchers have conducted many studies on the supervised learning in SNNs [3, 4], and achieved some results, but many problems remain unsolved. Bohte et al. [5] first proposed a backpropagation training algorithm for feedforward SNNs, called SpikeProp, similar in concept to the BP algorithm developed for traditional Artificial Neural Networks (ANNs). The spike response model (SRM) [6] is used in this algorithm. To overcome the discontinuity of the internal state variable caused by spike firing, all neurons in the network can fire only one single spike. Shrestha et al. [7] analyzed the convergence characteristics of SpikeProp algorithm, and proposed an adaptive learning rate method, the experimental results show that the adaptive learning rate greatly improved the weight convergence and learning speed. Similarly, Ghosh-Dastidar and Adeli [8] put forward a BP learning algorithm named Multi-SpikeProp, with derivations of the learning rule based on the chain rule for a multi-spiking network model. Multi-SpikeProp was applied to the standard XOR problem and the Fisher Iris and EEG classification problems, and the experimental results show that the algorithm has higher classification accuracy than the SpikeProp algorithm. Recently, Xu et al. [9] have extended the Multi-SpikeProp algorithm to allow neurons to fire multiple spikes in all layers. That is, the algorithm can implement the complex spatio-temporal pattern learning of spike trains. The experimental results show that this algorithm has higher learning accuracy for a large number of output spikes. Considering the spike-timing-dependent plasticity (STDP) mechanism of spiking neurons, Ponulak et al. [10, 11] proposed the ReSuMe (Remote Supervised Method) algorithm, which adjusts the synaptic weights according to STDP and anti-STDP processes and is suitable for various types of spiking neuron models. However, the algorithm can only be applied to single layer networks or train readouts for reservoir networks. Sporea and Grüning [12] extended the ReSuMe algorithm to multilayer feedforward SNNs using backpropagation of the network error. The weights are updated according to STDP and anti-STDP rules, and the neurons in every layer can fire multiple spikes. Simulation experiments show that the algorithm can be successfully applied to various complex classification problems and permits precise spike train firing. Based on the ReSuMe algorithm with learning delay, Taherkhani et al. proposed DL-ReSuMe (delay learning remote supervised method) algorithm [13], and put forward a Multi-DL-ReSuMe algorithm [14] for multiple neurons. Mohammed et al. [15, 16] proposed a SPAN (Spike Pattern Association Neuron) algorithm based on a Hebbian interpretation of the Widrow-Hoff rule and kernel function convolution. Inspired by the SPAN algorithm, Yu et al. [17, 18] proposed a PSD (Precise-Spike-Driven) supervised learning rule that can be used to train neurons to associate an input spatio-temporal spike pattern with a desired spike train. Unlike the SPAN method that requires spike convolution on all the spike trains of the input, the desired output and the actual output, the PSD learning rule only convolves the input spike trains.

For SNNs, input and output information is encoded through precisely timed spike trains, not only through the neural firing rate. In addition, the internal state variables of spiking neurons and error function do not satisfy the continuous differentiability. So, traditional learning algorithms of ANNs, especially the BP algorithm, cannot be used directly, and the formulation of efficient supervised learning algorithms for SNNs is a very challenging problem. In this paper, we present a new supervised learning algorithm for spiking neurons with the nonlinear inner products of spike trains. The rest of

this paper is organized as follows. In Sect. 2 we analyze and define the nonlinear inner products of spike trains. In Sect. 3 we derive the learning rule based on the nonlinear inner products of spike trains for spiking neurons. In Sect. 4 the flexibility and power of spiking neurons trained with our algorithm are showcased by a spike sequence learning problem. The conclusion is presented in Sect. 5.

2 Inner Products of Spike Trains

The spike train $s = \{t_i \in \Gamma : i = 1, \dots, N\}$ represents the ordered sequence of spike times fired by the spiking neuron in the interval $\Gamma = [0, T]$, and can be expressed formally as:

$$s(t) = \sum_{i=1}^N \delta(t - t_i) \quad (1)$$

where N is the number of spikes, and $\delta(\cdot)$ represents the Dirac delta function, $\delta(x) = 1$ if $x = 0$ and 0 otherwise.

In order to facilitate the analysis and calculation, we can choose a specific smoothing function h , using the convolution to convert the discrete spike train to a unique continuous function:

$$f_s(t) = s * h = \sum_{i=1}^N h(t - t_i) \quad (2)$$

Due to the limited time interval of the corresponding spike train and boundedness of the function $f_s(t)$, we can get:

$$\int_{\Gamma} f_s^2(t) dt < \infty \quad (3)$$

In other words, the function $f_s(t)$ is an element of $L_2(\Gamma)$ space.

For any two given spike trains $s_i, s_j \in s(\Gamma)$, we can define the inner products of the corresponding functions $f_{s_i}(t)$ and $f_{s_j}(t)$ on the $L_2(\Gamma)$ space as follows [19]:

$$F(s_i, s_j) = \langle f_{s_i}(t), f_{s_j}(t) \rangle_{L_2(\Gamma)} = \int_{\Gamma} f_{s_i}(t) f_{s_j}(t) dt \quad (4)$$

The effect of the spike train to the postsynaptic neuron expressed in Eq. 2 is linear. So, considering the biologically plausible nonlinear effect of spiking neuron, we can rewrite the continuous function corresponding to the spike train s as follow:

$$f_s^\dagger(t) = g \left(\sum_{i=1}^N h(t - t_i) \right) \quad (5)$$

where $g(\cdot)$ is the nonlinear function. So the inner products of the spike trains $s_i, s_j \in S(\Gamma)$ can be extended to the nonlinear form as follow [20]:

$$\begin{aligned} F^\dagger(s_i, s_j) &= \left\langle f_{s_i}^\dagger(t), f_{s_j}^\dagger(t) \right\rangle_{L_2(\Gamma)} \\ &= \int_{\Gamma} f_{s_i}^\dagger(t) f_{s_j}^\dagger(t) dt \\ &= \int_{\Gamma} g(f_{s_i}(t)) g(f_{s_j}(t)) dt \end{aligned} \quad (6)$$

For the nonlinear function $g(\cdot)$, it can be expressed as:

$$g(x) = \sigma \left[1 - \exp\left(-\frac{x}{2\sigma^2}\right) \right] \quad (7)$$

or

$$g(x) = \sigma \tanh\left(\frac{x}{\sigma}\right) \quad (8)$$

3 Learning Algorithm

The input and output signals of spike neurons are expressed in the form of spike trains; that is, the spike trains encode neural information or external stimuli. The computation performed by single spiking neurons can be defined as a mapping from the input spike trains to the appropriate output spike trains. For a given spiking neuron, we assume that the input spike trains are $s_i \in s(\Gamma)$, $i = 1, \dots, N$, and the output spike train is $s_o \in s(\Gamma)$. In order to analyze the relationship between the input and output spike trains, we use the linear Poisson neuron model [21]. This neuron model outputs a spike train, which is a realization of a Poisson process with the underlying intensity function estimation. The spiking activity of the postsynaptic neuron is defined by the estimated intensity functions of the presynaptic neurons. The contributions of all input spike trains are summed up linearly:

$$f_{s_o}(t) = \sum_{i=1}^N w_{oi} f_{s_i}(t) \quad (9)$$

where the weights w_{oi} represent the strength of the connection between the presynaptic neuron i and the postsynaptic neuron o .

We derive the proposed learning algorithm from the common Widrow-Hoff rule, also known as the Delta rule. For a synapse i , it is defined as:

$$\Delta w_i = \eta x_i (y_d - y_a) \quad (10)$$

where η is the learning rate, x_i is the input transferred through synapse i , and y_d and y_a refer to the desired and the actual neural output, respectively. By substituting x_i , y_d and

y_a with the continuous functions corresponding to the spike trains $f_{s_i}^\dagger(t)$, $f_{s_d}^\dagger(t)$ and $f_{s_o}^\dagger(t)$, a new learning rule for a spiking neuron is obtained:

$$\Delta w_i(t) = \eta f_{s_i}^\dagger(t) \left(f_{s_d}^\dagger(t) - f_{s_o}^\dagger(t) \right) \quad (11)$$

The equation formulates a real-time learning rule and so the synaptic weights change over time. By integrating Eq. 11, we derive the batch version of the learning rule which is under scrutiny in this paper:

$$\begin{aligned} \Delta w_i &= \eta \int_{\Gamma} f_{s_i}^\dagger(t) \left(f_{s_d}^\dagger(t) - f_{s_o}^\dagger(t) \right) dt \\ &= \eta \left(F^\dagger(s_i, s_d) - F^\dagger(s_i, s_o) \right) \end{aligned} \quad (12)$$

where $F^\dagger(\cdot, \cdot)$ is the nonlinear inner products of the spike trains expressed in Eq. 6. In order to facilitate the description, we called the algorithm Nonlinear1 when the nonlinear function $g(\cdot)$ is Eq. 7, and called the algorithm Nonlinear2 when the nonlinear function $g(\cdot)$ is Eq. 8.

4 Simulations

In this section, several experiments are presented to demonstrate the learning capabilities of our learning algorithm. At first, the algorithm is applied to the learning sequences of spikes, by demonstrating its ability to associate a spatio-temporal spike pattern with a desired spike train. Then, we analyze the factors that may influence the learning performance, such as the number of synaptic inputs and the firing rate of spike trains.

To quantitatively evaluate the learning performance, we use the correlation-based measure C [22] to express the distance between the desired and actual output spike trains. The metric is calculated after each learning epoch according to

$$C = \frac{s_d \cdot s_o}{|s_d| |s_o|} \quad (13)$$

where s_d and s_o are vectors representing convolution of the desired and actual output spike trains with a Gaussian low-pass filter, $s_d \cdot s_o$ is the inner product, and $|s_d|$, $|s_o|$ are the Euclidean norms of s_d and s_o , respectively. The measure $C = 1$ for the identical spike trains and decreases towards 0 for loosely correlated trains.

In the first experiment, we demonstrate the learning ability of the proposed algorithm to reproduce the desired spatio-temporal spike pattern. Unless stated otherwise, the basic parameter settings are: the number of input neurons is 500 and one output neuron. Initially, the synaptic weights are generated as the uniform distribution in the interval $(0, 0.2)$. Every input spike train and desired output spike train are generated randomly by a homogeneous Poisson process with rate $r = 20$ Hz and $r = 50$ Hz respectively within the time interval of $[0, T]$, and we set $T = 100$ ms here. The results

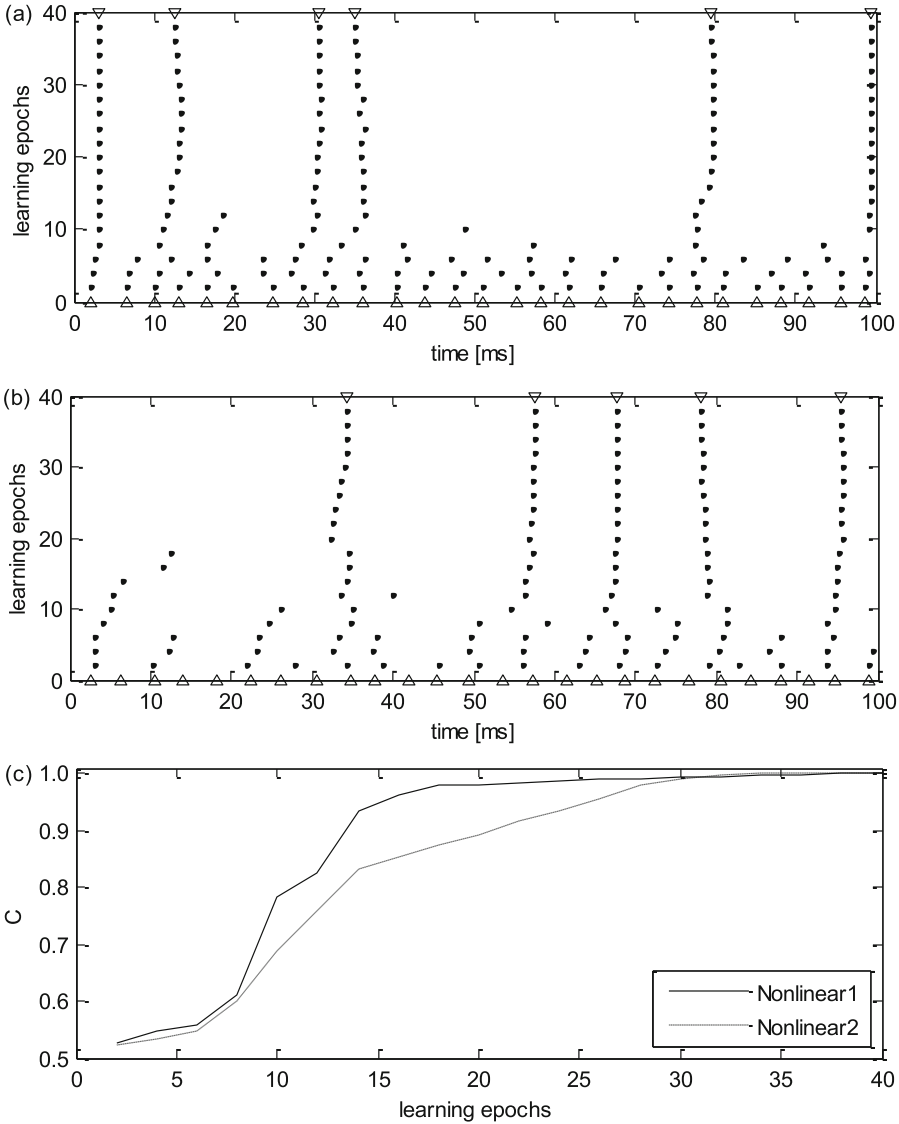


Fig. 1. The learning process of our algorithm. (a) The complete learning process of Nonlinear1. Δ , initial actual output spike train; ∇ , desired output spike train; \bullet , actual output spike trains at some learning epochs. (b) The complete learning process of Nonlinear2. (c) The evolution of the learning accuracy of Nonlinear1 and Nonlinear2 with measure C .

are averaged over 50 trials, and on each testing trial the learning algorithm is applied for a maximum of 500 learning epochs or until the measure $C = 1$. The neurons are described by the short term memory SRM [9] so that only the last firing spike contributes to the refractoriness. The parameter values of the SRM neurons used in the experiments are: the time constant of postsynaptic potential $\tau = 2$ ms, the time constant

of refractory period $\tau_R = 50$ ms, the neuron threshold $\theta = 1$, and the length of the absolute refractory period $t_R = 1$ ms. The learning rate η of our algorithm is 0.0005.

Figure 1 shows the learning process of our algorithm. Figure 1(a) and (b) show the complete learning process of Nonlinear1 and Nonlinear2 respectively, which include 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 evolution of the learning accuracy of Nonlinear1 and Nonlinear2 with measure C in the time interval T is represented in Fig. 1(c). For both Nonlinear1 and Nonlinear2, the measure C increases rapidly at the beginning of the learning, and reached to 1 after 38 and 36 learning epochs, respectively.

Figure 2 shows the learning results with different number of synaptic inputs. The number of synaptic inputs increases from 100 to 1000 with an interval of 100, while the

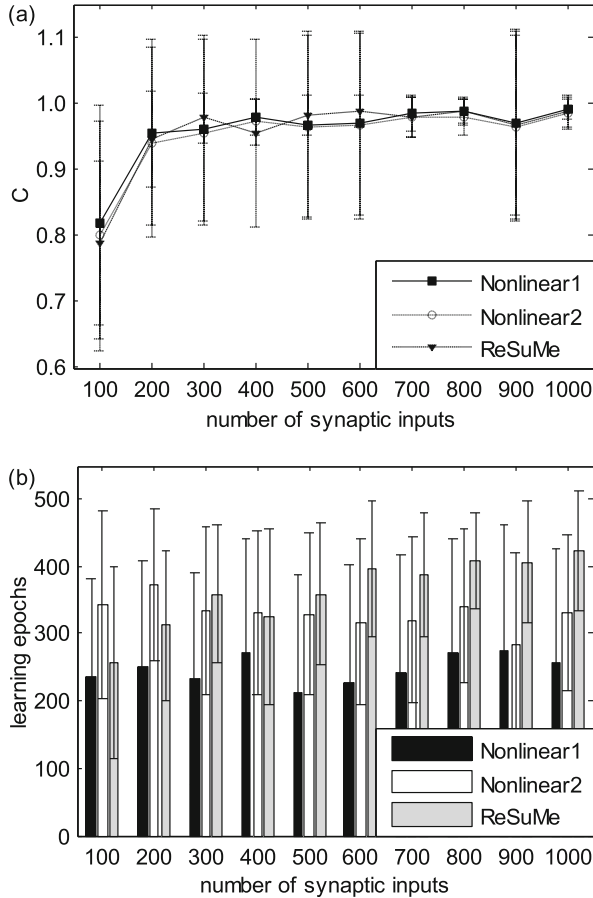


Fig. 2. The learning results with different number of synaptic inputs for our algorithm and ReSuMe algorithm after 500 learning epochs. (a) The learning accuracy with measure C . (b) The learning epochs when the measure C reaches the maximum value.

other settings remain the same. In this experiment, we compare our method with the ReSuMe algorithm [10]. The parameters in the ReSuMe algorithm are the same as those described by Ponulak et al. [10]. The learning rate η of ReSuMe algorithm is 0.005. Figure 2(a) shows the learning accuracy after 500 learning epochs and Fig. 2(b) shows the learning epochs when the measure C reaches the maximum value. Through Fig. 2(a) we can see that both Nonlinear1, Nonlinear2 and ReSuMe can learn with high accuracy. The learning accuracy of our algorithm and ReSuMe increase when the number of synaptic inputs increases gradually. Our algorithm has higher learning accuracy than ReSuMe except the numbers of synaptic inputs are 300, 500 and 600. For example, the measure $C = 0.9792$ for Nonlinear1, $C = 0.9719$ for Nonlinear2 and $C = 0.9552$ for ReSuMe when the number of synaptic inputs is 400. When the number of synaptic inputs is 1000, the measure $C = 0.9916$ for Nonlinear1, $C = 0.9854$ for

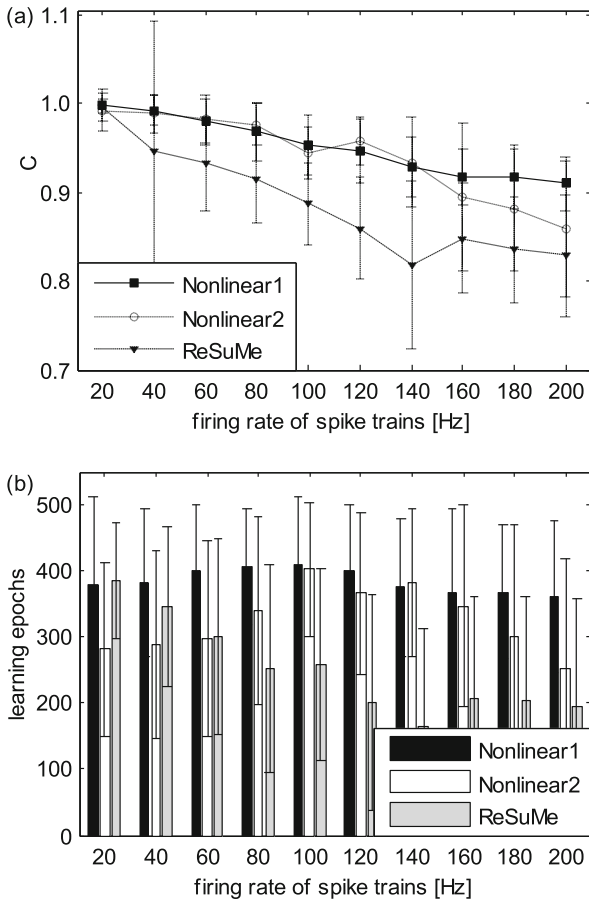


Fig. 3. The learning results with different firing rate of spike trains for our algorithm and ReSuMe algorithm after 500 learning epochs. (a) The learning accuracy with measure C . (b) The learning epochs when the measure C reaches the maximum value.

Nonlinear2 and $C = 0.9871$ for ReSuMe. From Fig. 2(b) we can see that both Nonlinear1 and Nonlinear2 have less learning epochs than ReSuMe. For example, the learning epochs of Nonlinear1 and Nonlinear2 are 213.66 and 328.98 respectively, which are less than 357.70 of ReSuMe when the number of synaptic inputs is 500. When the number of synaptic inputs is 1000, the learning epochs of Nonlinear1 and Nonlinear2 are 256.72 and 331.18 respectively, which are less than 422.02 of ReSuMe.

Figure 3 shows the learning results with different firing rate of spike trains. The firing rate of input and desired output spike trains increases from 20 Hz to 200 Hz with an interval of 20 Hz and the firing rate of input spike trains equals to that of desired output spike trains, while the other settings remain the same. Figure 3(a) shows the learning accuracy after 500 learning epochs and Fig. 3(b) shows the learning epochs when the measure C reaches the maximum value. Through Fig. 3(a) we can see that our algorithm can learn with higher accuracy than ReSuMe algorithm. The learning accuracy of our algorithm and ReSuMe algorithm decrease when the firing rate of input and desired output spike trains increases gradually. For example, the measure $C = 0.9530$ for Nonlinear1, $C = 0.9440$ for Nonlinear2 and $C = 0.8870$ for ReSuMe when the firing rate of input and desired output spike trains is 100 Hz. When the firing rate of input and desired output spike trains is 200 Hz, the measure $C = 0.9093$ for Nonlinear1, $C = 0.8581$ for Nonlinear2 and $C = 0.8289$ for ReSuMe.

5 Conclusions

In this paper, we introduced a new supervised learning algorithm based on nonlinear inner products of spike trains for spiking neurons. Our learning algorithm uses the nonlinear inner products of spike trains to deduce the learning rule. The inner products of spike trains have been used on similarity measures theory for spike trains, but less used for supervised learning algorithms of SNNs at present. The synaptic weight modification rules only depend on the input, actual output and desired output spike trains and do not depend on the specific dynamic of the neuron model. The algorithm is tested on learning sequences of spikes and compared with ReSuMe algorithm. The experimental results indicate that our method is an effective supervised multi-spike learning algorithm for spiking neurons. It can learn spike trains with high accuracy.

Acknowledgement. This work is supported by the National Natural Science Foundation of China (Nos. 61165002 and 61363058), the Natural Science Foundation of Gansu Province of China (No. 1506RJZA127), and Scientific Research Project of Universities of Gansu Province (No. 2015A-013).

References

1. Quiroga, R.Q., Panzeri, S.: Principles of Neural Coding. CRC Press, Boca Raton (2013)
2. Ghosh-Dastidar, S., Adeli, H.: Spiking neural networks. *Int. J. Neural Syst.* **19**(4), 295–308 (2009)

3. Kasiński, A., Ponulak, F.: Comparison of supervised learning methods for spike time coding in spiking neural networks. *Int. J. Appl. Math. Comput. Sci.* **16**(1), 101–113 (2006)
4. Lin, X., Wang, X., Zhang, N., et al.: Supervised learning algorithms for spiking neural networks: a review. *Acta Electronica Sinica* **43**(3), 577–586 (2015)
5. Bohte, S.M., Kok, J.N., La Poutré, J.A.: Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* **48**(1), 17–37 (2002)
6. Gerstner, W., Kistler, W.M.: *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, Cambridge (2002)
7. Shrestha, S.B., Song, Q.: Adaptive learning rate of SpikeProp based on weight convergence analysis. *Neural Networks* **63**, 185–198 (2015)
8. Ghosh-Dastidar, S., Adeli, H.: A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks* **22**(10), 1419–1431 (2009)
9. Xu, Y., Zeng, X., Han, L., et al.: A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. *Neural Networks* **43**, 99–113 (2013)
10. Ponulak, F., Kasinski, A.: Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. *Neural Comput.* **22**(2), 467–510 (2010)
11. Ponulak, F.: Analysis of the ReSuMe learning process for spiking neural networks. *Int. J. Appl. Math. Comput. Sci.* **18**(2), 117–127 (2008)
12. Sporea, I., Grüning, A.: Supervised learning in multilayer spiking neural networks. *Neural Comput.* **25**(2), 473–509 (2013)
13. Taherkhani, A., Belatreche, A., Li, Y., et al.: DL-ReSuMe: a delay learning-based remote supervised method for spiking neurons. *IEEE Trans. Neural Netw. Learn. Syst.* **26**(12), 3137–3149 (2015)
14. Taherkhani, A., Belatreche, A., Li, Y., et al.: Multi-DL-ReSuMe: multiple neurons delay learning remote supervised method. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE (2015)
15. Mohemmed, A., Schliebs, S., Matsuda, S., et al.: SPAN: spike pattern association neuron for learning spatio-temporal spike patterns. *Int. J. Neural Syst.* **22**(4), 786–803 (2012)
16. Mohemmed, A., Schliebs, S., Matsuda, S., et al.: Training spiking neural networks to associate spatio-temporal input–output spike patterns. *Neurocomputing* **107**, 3–10 (2013)
17. Yu, Q., Tang, H., Tan, K.C., et al.: Precise-spike-driven synaptic plasticity: learning hetero-association of spatiotemporal spike patterns. *PLoS One* **8**(11), e78318 (2013)
18. Yu, Q., Tang, H., Tan, K.C., et al.: A brain-inspired spiking neural network model with temporal encoding and learning. *Neurocomputing* **138**, 3–13 (2014)
19. Park, I.M., Seth, S., Rao, M., et al.: Strictly positive-definite spike train kernels for point-process divergences. *Neural Comput.* **24**(8), 2223–2250 (2012)
20. Paiva, A.R.C., Park, I., Principe, J.C.: Inner products for representation and learning in the spike train domain. In: *Statistical Signal Processing for Neuroscience and Neurotechnology*, vol. 8 (2010)
21. Gütig, R., Aharonov, R., Rotter, S., et al.: Learning input correlations through nonlinear temporally asymmetric Hebbian plasticity. *J. Neurosci.* **23**(9), 3697–3714 (2003)
22. Schreiber, S., Fellous, J.M., Whitmer, D., et al.: A new correlation-based measure of spike timing reliability. *Neurocomputing* **52**, 925–931 (2003)