Int. J. Appl. Math. Comput. Sci., 2006, Vol. 16, No. 1, 101–113

COMPARISON OF SUPERVISED LEARNING METHODS FOR SPIKE TIME CODING IN SPIKING NEURAL NETWORKS

ANDRZEJ KASIŃSKI, FILIP PONULAK

Institute of Control and Information Engineering, Poznań University of Technology
ul. Piotrowo 3a, 60–965 Poznań, Poland
e-mail: {Andrzej.Kasinski,Filip.Ponulak}@put.poznan.pl

In this review we focus our attention on supervised learning methods for spike time coding in Spiking Neural Networks (SNNs). This study is motivated by recent experimental results regarding information coding in biological neural systems, which suggest that precise timing of individual spikes may be essential for efficient computation in the brain. We are concerned with the fundamental question: What paradigms of neural temporal coding can be implemented with the recent learning methods? In order to answer this question, we discuss various approaches to the learning task considered. We shortly describe the particular learning algorithms and report the results of experiments. Finally, we discuss the properties, assumptions and limitations of each method. We complete this review with a comprehensive list of pointers to the literature.

Keywords: supervised learning, spiking neural networks, time coding, temporal sequences of spikes

1. Introduction

For many years a common belief was that essential information in neurons is encoded in their firing rates. However, recent neurophysiological results suggest that efficient processing of information in neural systems can be founded also on precise timing of action potentials (spikes) (Bohte, 2004; VanRullen *et al.*, 2005; Thorpe *et al.*, 2001). In the barn owl auditory system, neurons detecting coincidence receive volleys of precisely timed spikes from both ears (Gabbiani and Midtgaard, 2001; Gerstner and Kistler, 2002a). Under the influence of a common oscillatory drive in the rat hippocampus, the strength of a constant stimulus is coded in relative timing of neuronal action potentials (Mehta *et al.*, 2002). In humans, precise timing of first spikes in tactile afferents encodes touch signals at finger tips (Thorpe *et al.*, 2001). Time codes have also been suggested for rapid visual processing (Thorpe *et al.*, 2001).

A precise temporal coding paradigm is required in some artificial control systems. Examples are neuroprosthetic systems which aim at producing functionally useful movements of paralysed limbs by exciting muscles or nerves with the sequences of short electrical impulses (Popović and Sinkjaer, 2000). Precise relative timing of impulses is critical for generating desired, smooth movement trajectories.

In addition to the above examples, it has been theoretically demonstrated that the temporal neural code is very efficient whenever fast information processing is required (Maass, 2002). All these arguments provide strong motivation for investigating computational properties of systems that compute with precisely timed spikes.

It is generally recognized that artificial Spiking Neural Networks (SNNs) (Gerstner and Kistler, 2002a; Maass, 1997; Maass and Bishop, 1999) are capable of exploiting time as a resource for coding and computation in a much more sophisticated manner than typical neural computational models (Maass, 1998; 2003). SNNs appear to be an interesting tool for investigating temporal neural coding and for exploiting its computational potential. Although significant progress has already been made to recognize information codes that can be beneficial for computation in SNNs (Gerstner and Kistler, 2002a; Maass, 1999; 2003; Maass and Bishop, 1999), it is still an open problem to determine efficient neural learning mechanisms that facilitate the implementation of these particular time coding schemes.

Unsupervised spike-based processes, such as Long Term Potentiation (LTP), Long Term Depression (LTD) and Spike-Timing Dependent Plasticity (STDP), have already been widely investigated and described in the literature (Bi, 2002; Bonhoeffer *et al.*, 1989; Gerstner and Kistler, 2002b; Gerstner *et al.*, 1996; Markram *et al.*, 1997; Kepecs *et al.*, 2002; Kistler, 2002). However, the unsupervised approach is not suitable for learning tasks that require an explicit goal definition.

In this article we focus on supervised learning methods for precise spike timing in SNNs. The goal of our

study is to determine what paradigms of neural information coding can be implemented with recent approaches.

First, we present supervised learning methods for spike timing, which are known from the literature. We classify these methods into more general groups representing particular learning approaches and shortly describe each of the learning algorithms. Finally, we summarize the main facts about the approaches and discuss their properties.

2. Review of Learning Methods

In this section we present some representative methods for supervised learning in SNNs. For all these methods the common goal of learning can be stated as follows: Given a sequence of the input spike trains $S^{\rm in}(t)$ and a sequence of the target output spikes $S^d(t)$, find a vector of the synaptic weights w, such that the outputs of the learning neurons $S^{\rm out}(t)$ are close to $S^d(t)$.

2.1. Methods Based on Gradient Evaluation

Learning in the traditional, artificial neural networks (ANNs) is usually performed by gradient ascent/descent techniques (Hertz *et al.*, 1991). However, explicit evaluation of the gradient in SNNs is infeasible due to the discontinuous-in-time nature of spiking neurons. Indirect approaches or special simplifications must be assumed to deal with this problem.

In (Bohte *et al.*, 2000; 2002), the authors presented one of such approaches. Their method, called *SpikeProp*, is analogous to the backpropagation algorithm (Rumelhart *et al.*, 1986) known from the traditional artificial neural networks.

The target of *SpikeProp* is to learn a set of the desired firing times, denoted by t_j^d, at the postsynaptic neurons $j \in J$ for a given set of the input patterns $S^{\rm in}(t)$. However, it is assumed that each neuron in a simulated network is allowed to fire only once during a single simulation cycle and the time course of the neuron's membrane potential after the firing is not considered. Thus the problem of the discontinuity of the membrane potential at the firing time is avoided here. On the other hand, this assumption limits the class of neural information coding schemes implementable in the *SpikeProp* method.

The *SpikeProp* algorithm has been derived for the neurons modeled by the Spike Response Model (SRM) (Gerstner and Kistler, 2002a). In this model, the membrane potential of the neuron j can be described by

$$V_j(t) = \sum_{i\in\Gamma_j}\sum_k w_{ij}^k\, \epsilon\left(t - t_i^{\rm out} - d_{ij}^k\right). \qquad (1)$$

Equation (1) holds for the region where membrane potential changes are mostly determined by postsynaptic potentials $\epsilon(t)$. The set Γ_j represents all pre-synaptic neurons of neuron j. The term w_{ij}^k is the weight of the synaptic terminal k of the connection between the neurons i and j. It is assumed that $\epsilon(t) = t/\tau \exp(1 - t/\tau)$, with some time constant τ. The parameter $t_i^{\rm out}$ is the firing time of the neuron i, and d_{ij}^k is the delay of the synaptic terminal.

The learning method is based on explicit evaluation of the gradient of $E = 1/2\sum_j (t_j^{\rm out} - t_j^d)^2$ with respect to the weights of each synaptic input to j:

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial t_j}\left(t_j^{\rm out}\right)\frac{\partial t_j}{\partial w_{ij}^k}\left(t_j^{\rm out}\right) = \frac{\partial E}{\partial t_j}\left(t_j^{\rm out}\right)\frac{\partial t_j}{\partial V_j}\left(t_j^{\rm out}\right)\frac{\partial V_j}{\partial w_{ij}^k}\left(t_j^{\rm out}\right). \qquad (2)$$

In the factors on the right, t_j is expressed as a function of the membrane potential $V_j(t)$ around $t = t_j^{\rm out}$. In order to simplify the gradient evaluation, it is assumed that for a small region around $t = t_j^{\rm out}$ the function $V_j^m(t)$ is linearly approximated. Hence, the local derivative of t_j with respect to $V_j^m(t)$ is constant. Error-backpropagation equations derived for a fully connected feedforward network with hidden layers are as follows:

$$\frac{\partial E}{\partial w_{ij}^k} = y_{ij}^k(t)\ \delta_j, \qquad (3)$$

where δ_j for neurons in the output layer equals

$$\delta_j = \frac{-\left(t_j^{\rm out} - t_j^d\right)}{\sum\limits_{i\in\Gamma_j}\sum\limits_k w_{ij}^k \frac{\partial y_{ij}^k(t)}{\partial t}}. \qquad (4)$$

For hidden neurons, we have

$$\delta_j = \frac{\sum\limits_{i\in\Gamma^j}\delta_i \sum\limits_k w_{ij}^k \frac{\partial y_{ij}^k(t)}{\partial t}}{\sum\limits_{i\in\Gamma_j}\sum\limits_k w_{ij}^k \frac{\partial y_{ij}^k(t)}{\partial t}}. \qquad (5)$$

In (4) and (5), the set Γ_j represents again all the direct pre-synaptic neurons of the neuron j, while the set Γ^j represents all the direct successors of the neuron j.

Finally, the weights are modified according to

$$\Delta w_{ij}^k = -\eta \frac{\partial E}{\partial w_{ij}^k} = -\eta\ y_{ij}^k(t)\ \delta_j, \qquad (6)$$

with η defining the learning rate. The error is thus minimized by changing the weights according to the negative local gradient.

The presented *SpikeProp* algorithm was reinvestigated in (Moore, 2002; Schrauwen and Van Campenhout, 2004; Tiňo and Mills, 2005; Xin and Embrechts, 2001). It was found that weight initialization is a critical factor for good performance of the learning rule. In (Moore, 2002), the weights were initialized with values that led the network to the successful training in a similar number of iterations as in (Bohte *et al.*, 2000), but with large learning rates, although Bohte argued that the approximation of the threshold function implies that only small learning rates can be used (Bohte *et al.*, 2002). Other experiments (Moore, 2002) also provided evidence proving that negative weights could be allowed and still led to successful convergence, which was in contradiction to Bohte's conclusions. Xin and Embrechts (2001) proposed a modification of the learning algorithm by including the momentum term in the weight update equation. It was demonstrated that this modification significantly speeded up the convergence of $SpikeProp$. In (Schrauwen and Van Campenhout, 2004), the authors adapted the gradient descent method derived in $SpikeProp$ to adjust not only synaptic weights, but also synaptic delays, time constants and neurons' thresholds. This resulted in faster algorithm convergence and in smaller network topologies required for the given learning task. Finally, Tiňo and Mills (2005) extended *SpikeProp* to recurrent network topologies, to account for temporal dependencies in the input stream. Neither the original *SpikeProp* method nor any of the proposed modifications allow learning patterns composed of more than one spike per neuron.

The properties of the *SpikeProp* method were demonstrated in a set of classification experiments. These included the standard and interpolated XOR problems (Maass, 1999). The *SpikeProp* authors encoded the input and output values by time delays, associating the analog values with the corresponding "earlier" or "later" firing times. In the interpolated XOR experiment, the network could learn the presented input with an accuracy of the order of the algorithm integration time step.

The classification abilities of *SpikeProp* were also tested on a number of common benchmark datasets (the Iris dataset, the Wisconsin breast cancer dataset and the Statlog Landsat dataset). For these problems, the accuracy of SNNs trained with *SpikeProp* was comparable to that of a sigmoidal neural network. Moreover, in experiments on real-world datasets, the *SpikeProp* algorithm always converged, whereas the compared ANN algorithms, such as the Levenberg-Marquardt algorithm, occasionally failed.

The main drawback of the *SpikeProp* method is that there is no mechanism to "prop-up" synaptic weights once the postsynaptic neuron no longer fires for any input pattern. Moreover, in the *SpikeProp* approach only the first spike produced by a neuron is relevant and the rest of the time course of the neuron is ignored. Whenever a neuron fires a single spike, it is not allowed to fire again. For this reason the method is suitable to implement only the 'time-to-first-spike' coding scheme (Thorpe *et al.*, 2001).

2.2. Statistical Methods

In (Pfister *et al.*, 2003; 2005), the authors proposed to derive a supervised spike-based learning algorithm starting with statistical learning criteria. Their method is based on the approach proposed by Barber. However, in (Barber, 2003), the author considered supervised learning for neurons operating on a discrete time scale. Pfister *et al.* extended this study to the continuous case.

The fundamental hypothesis in (Pfister *et al.*, 2003; 2005) is to assume that the instantaneous firing rate of the postsynaptic neuron j is determined by a point process with the time-dependent stochastic intensity $\rho_j(t) = g(V_j(t))$ that depends nonlinearly upon the membrane potential $V_j(t)$. The firing rate $\rho_j(t)$ is known as the escape rate (Gerstner and Kistler, 2002a).

The goal of the learning rule considered is to optimise the weights w_j in order to maximise the likelihood of getting postsynaptic firing at the desired times, i.e. to obtain $S_j^{\text{out}}(t) = S_j^d(t)$, given the firing rate $\rho_j(t)$. The optimization is performed via the gradient ascent of the likelihood P_j of the postsynaptic firing for one or several desired firing times. The advantage of the discussed probabilistic approach is that it allows us to describe explicitly the likelihood $P_j\left(S_j^{\text{out}}(t)|S_j^{\text{in}}(t)\right)$ of emitting $S_j^{\text{out}}(t)$ for a given input $S_j^{\text{in}}(t)$:

$$P_j\left(S_j^{\text{out}}(t)|S_j^{\text{in}}(t)\right) = \exp\Big(\int_0^T \log\left(\rho_j\left(s|S_j^{\text{in}}(t), S_j^{\text{out}}(s)\right)\right)\hat{S}_j^{\text{out}}(s) - \rho_j\left(s|S_j^{\text{in}}(t), S_j^{\text{out}}(s)\right)\mathrm{d}s\Big), \quad (7)$$

where $S_j^{\text{out}}(t)$ denotes the set of postsynaptic spikes that occurred before t and $\hat{S}_j^{\text{out}}(t)$ is the entire postsynaptic spike train.

Since the likelihood P_j is a smooth function of its parameters, it is straightforward to differentiate it with respect to the synaptic efficacies w_j. On the basis of the evaluated derivative $\partial \log(P_j)/\partial w_j$ and the standard technique of gradient ascent, the authors derived a set of rules for the modification of synaptic weights. The particular rules corresponded to the different scenarios of neuron stimulation. The learning rules can be described by a two-phase learning window similar to that of Spike-Timing Dependent Plasticity (Kepecs *et al.*, 2002; Kistler, 2002). The authors demonstrated that the shape of the

learning window was strongly influenced by constraints imposed by the different scenarios of the optimization procedure.

In the approach considered, it is assumed that learning rules apply to all synaptic inputs of the learning neuron and the postsynaptic neuron j receives an additional 'teaching' input $I(t)$ that could either arise from a second group of neurons or from the intracellular current injection. The role of $I(t)$ is to increase the probability that the neuron fires at or close to the desired firing time t_j^d. In this context, the learning mechanism can also be viewed as a probabilistic version of the spike-based supervised Hebbian learning (described in Section 2.6).

In (Pfister *et al.*, 2005), the authors present a set of experiments which differ in the stimulation mode and the specific tasks of the learning neuron. The learning algorithm is applied to the spike response model with escape noise as a generative model of the neuron (Gerstner and Kistler, 2002a). The authors consider different scenarios of the experiments:

- different sources of the 'teaching' signal (the signal is given by a supervisor as a train of spikes or as a strong current pulse of short duration);
- allowing (or not) other postsynaptic spikes to be generated spontaneously;
- implementing a temporal coding scheme where the postsynaptic neuron responds to one of the presynaptic spike patterns with a desired output spike train containing several spikes while staying inactive for other presynaptic spike patterns.

The experiments demonstrate the ability of the learning method to precisely set the time of single firings at the neuron output. However, since in all experiments a desired postsynaptic spike train consisted of at most 2 spikes, it is hard to estimate a potential, practical suitability of the proposed method to learn complex spike trains consisting of dozens of spikes.

2.3. Linear Algebra Methods

Carnell and Richardson proposed to apply linear algebra to the task of spike time learning (Carnell and Richardson, 2004). They begin with the formal introduction of the time series $S(t) = \sum_{i=1}^{N} s(t_i)$ and the weighted time series ${}^wS(t) = \sum_{i=1}^{N} w_i s(t_i)$, where $s(t_i)$ denotes a single spike at the time t_i and w_i is the weight corresponding to $s(t_i)$. The inner product of two weighted time series is defined as

$$\Big\langle \sum w_i s(t_i), \sum \omega_j s(\tau_j) \Big\rangle = \sum w_i \omega_j \exp\big(-\|t_i - \tau_j\|\big). \quad (8)$$

For the weighted time series ${}^wS(t)$, the metric $\mathrm{norm}({}^wS(t)) = \sqrt{\langle {}^wS(t), {}^wS(t)\rangle}$ is introduced. Here the $\mathrm{norm}\,({}^wS_1(t) - {}^wS_2(t))$ can be considered as a measure of the difference between ${}^wS_1(t)$ and ${}^wS_2(t)$. ${}^wS_1(t)$ is orthogonal to ${}^wS_2(t)$ if and only if $\langle {}^wS_1(t), {}^wS_2(t)\rangle = 0$. The operation of projecting ${}^wS_1(t)$ onto the direction of ${}^wS_2(t)$ is defined as $\mathrm{Proj}_{{}^wS_1(t)}({}^wS_2(t)) = (\langle {}^wS_1(t), {}^wS_2(t)\rangle \,/\, \langle {}^wS_1(t), {}^wS_1(t)\rangle)\,{}^wS_1(t)$. The projection can be understood as the best approximation to ${}^wS_2(t)$ expressed as a multiple of ${}^wS_1(t)$.

On the basis of these definitions, the authors formulate some algorithms for the approximation of the target pattern $S^d(t)$ given a set of the input patterns $S^{\mathrm{in}}(t)$ and a set of the adjustable synaptic weights w:

1. Gram-Schmidt solution: the Gram-Schmidt process (Cohen, 1993; Weisstein, 2006) is used to find an orthogonal basis for the subspace spanned by a set of the input time series $S^{\mathrm{in}}(t)$. Having the orthogonal basis, the best approximation in the subspace to any given element of $S^d(t)$ can be found.

2. Iterative solution: the projection of an error onto the direction of the times series S_i^{in} is determined, with i randomly chosen in each iteration. The error is defined as the difference between the target and the actual time series. The operation is evaluated until the norm of the error is sufficiently small. The algorithm is as follows:
   (a) define $E = S^d(t) - S^{\mathrm{out}}(t)$,
   (b) repeat the next steps until the $\mathrm{norm}(E)$ is small,
   (c) pick i at random,
   (d) define $\Delta w_i = \mathrm{Proj}_{\,S_i^{\mathrm{in}}(t)}\,(E)$,
   (e) let $w_i := w_i + \Delta w_i$,
   (f) proceed with network simulation, read the resulting $S^{\mathrm{out}}(t)$.

In a set of experiments the authors demonstrated that the iterative algorithm is able to approximate the target time series of spikes. The experiments were performed with the Liquid State Machine (LSM) network architecture (cf. Fig. 1(b)) (Maass *et al.*, 2002; Natschlaeger *et al.*, 2002) and the Leaky-Integrate-and-Fire (LIF) neuron models (Gerstner and Kistler, 2002a). Only a single neuron, considered as a network output, was subjected to learning. The approximated spike train consisted of 10 spikes (spanned within a 1 second interval). In the successful training case, an input vector $S^{\mathrm{in}}(t)$ was generated by 500 neurons. A good approximation of $S^d(t)$ was obtained after about 600 iterations. The presented results revealed that the ability of the method to produce the

desired target patterns is strongly influenced by the number and variability of spikes in $S^{\text{in}}(t)$. The quality of approximation is improved for the increased diversity of the spikes that populate the input to the learning neuron. This is a common conclusion for all LSM systems.

As a final remark, we state that the presented algorithm (Carnell and Richardson, 2004) is one out of only few algorithms that allow learning patterns consisting of multiple spikes. However, the algorithm updates weights in a batch mode and for this reason it is not suitable for online learning. In some applications this can be considered as a drawback.

2.4. Evolutionary Methods

In (Belatreche *et al.*, 2003), the authors investigate the viability of evolutionary strategies (ES) regarding supervised learning in spiking neural networks.

The use of the evolutionary strategy is motivated by the ability of ESs to work on real numbers without complex binary encoding schemes. ESs proved to be well suited for solving continuous optimization problems (Spears *et al.*, 1993). Unlike in genetic algorithms, the primary search operator in an ES is mutation. A number of different mutation operators have been proposed. The traditional mutation operator adds to the alleles of genes in the population some random value generated according to a Gaussian distribution. Other mutation operators include the use of the Cauchy distribution. The use of the Cauchy distribution allows exploring the search space by making large mutations and helping to prevent premature convergence. On the other hand, the use of the Gaussian mutation allows us to exploit the best solutions found in a local search. In this algorithm, not only synaptic strengths, but also synaptic delays are adjustable parameters. The spiking network is mapped to a vector of real values, which consists of the weights and delays of synapses. A set of such vectors (individuals) will form the population evolving according to the ES. The population is expected to converge to a globally optimal network, tuned to the particular input patterns.

The learning properties of the algorithm were tested on a set of classification tasks with XOR and the Iris benchmark dataset. SRM neuron models and feedforward fully connected spiking networks were used. Similarly to (Bohte *et al.*, 2000), continuous values were mapped here into firing delays. The authors reported results comparable to those obtained with the known classification algorithms (BP, LM, *SpikeProp*).

Some limitations of the algorithm arise due to the fact that each neuron is allowed to generate at most a single spike during the simulation time. Therefore, the method is not suitable for learning patterns consisting of multiple spikes. Another disadvantage, common to all evolutionary algorithms, is that computation with this approach is very time consuming.

2.5. Learning in Synfire Chains

Human learning often involves relating two signals separated in time, or linking a signal, an action and a subsequent effect into a causal relationship. These events are often separated in time but, nonetheless, humans can link them, thereby allowing them to accurately predict the right moment for a particular action. Synfire chains (SFCs) are considered as a possible mechanism for representing such relations between delayed events. An SFC is a feedforward multi-layered architecture (a chain), in which spiking activity can propagate in a synchronous wave of neuronal firing (a pulse packet) from one layer of the chain to the successive ones (Bienenstock, 1995). Each step in the SFC requires a pool of neurons whose firings simultaneously raise the potential of the next pool of neurons to the firing level. In this mechanism each cell of the chain fires only once. In (Sougne, 2001), a specific neural architecture, called INFERNET, is introduced. The architecture is an instance of the SFC. Its structure is organized into clusters of nodes called the *subnets*. Each subnet is fully connected. Some subnet nodes have connections to external subnet nodes. The nodes are represented here by a simple model similar to SRM.

The learning task is to reproduce the temporal relation between two successive inputs (the first one presented to the first layer of SFC and the other one considered as the 'teaching' signal, given to the last layer). Thus the task is to find a link between the firing input nodes and the firing target nodes with a target time delay.

Two successive inputs can be separated by several tenths of a second, and a single connection cannot alone be responsible for such long delays. Therefore, a long chain of successive pools of node firings might be required.

In the reported approach, the author introduces a learning algorithm in which the particular synaptic connections are modified by a rule similar to STDP, with, however, an additional non-Hebbian term:

$$\Delta w_{ij} = W\left(t_j^f - t_i^f + d_{ij}\right) - \lambda \operatorname{sign}\left(W\left(t_j^f - t_i^f + d_{ij}\right)\right). \quad (9)$$

According to (9), the weight changes Δw_{ij} of the synaptic connection from the neuron i to j are determined by the learning window W defined over the time difference between the pre- and postsynaptic firings, t_i^f and t_j^f, respectively, and the synaptic delay d_{ij} (Sougne, 2001). The function $\operatorname{sign}(x) = -1$ for $x < 0$ and $\operatorname{sign}(x) = 1$ otherwise.

Consider an SFC with n layers and the desired average time delays between the firings of the consecutive layers. The learning algorithm is as follows:

For each input node firing t_i^f
  For each presynaptic node j
  Calculate Δw_{ij} and add it to w_{ij}
  Select the n best nodes J'
  For each node $j' \in J'$
    Set the level to 1
    For each node j'' presynaptic to node j'
    Calculate $\Delta w_{j'j''}$ and add it to $w_{j'j''}$
    Select the n best nodes J''
    For each node $j'' \in J''$
      Set the level to 2
      Etc. up to the layer n.

The learning algorithm implies that synaptic weights between the particular neurons must become strong enough to ensure that the wave of excitation reaches eventually the output subnet.

Sougne (2001) discussed experiments in which two inputs are presented: one (the probe) at the time 0 ms and one (the target) at some later time instant. The task for the network was to correctly reproduce the temporal association between these two inputs and therefore to build an SFC between them. While trained, the network was able to trigger this synfire chain whenever the first input was presented. In this task, the author reported some difficulties. The algorithm could correctly reinforce a connection that led to the probe node firing at the right time, but could not, in general, prevent the target nodes from firing earlier, if some other 'inter-nodes' fired several times before. Indeed, a careful analysis of learning equations confirms that there is no rule for avoiding spurious firing.

We conclude that the learning method under consideration represents an interesting approach to the spike time learning problem in SNNs. In this method, it is assumed that the time of postsynaptic neuron firing depends mostly on the signal propagation delay in presynatpic neurons. The 'time-weight' dependence is neglected. The author focuses on modifying the topology of the network, to obtain the desired delay between the signal fed to the network input and the signal generated at the network output.

However, with this approach, the objective function (the desired time delay) is not a continuous function of the parameters (synaptic weights) of the optimization algorithm. For this reason, the algorithm can be considered as a discrete optimization technique. This approach makes it possible to attain precision that takes values not from the continuous domain, but from a finite set of possible solutions (since the global delay is a combination of fixed component delays, constituting a finite set). An approximation quality depends, in general, on the number and diversity of connection delays. Another limitation of the method is, again, the fact that it can learn only single firing times and thus can be applied only to the 'time-to-first-spike' coding scheme.

The author claims that the method enables him to learn sequentially many synfire chains. This property would be very interesting in the context of real-life applications. Unfortunately, it is not described in the cited article how this multi-learning can be achieved.

2.6. Spike-Based Supervised Hebbian Learning

In this subsection we discuss methods that represent the so-called Supervised Hebbian Learning (SHL) approach. According to this approach, Hebbian processes (Hebb, 1949) are supervised by an additional 'teaching' signal that reinforces the postsynaptic neuron to fire at the target times. The 'teaching' signal can be transmitted to the neuron in the form of synaptic currents or as intracellularly injected currents.

Ruf and Schmitt (1997) proposed one of the first spike-based methods similar to the SHL approach. In this first attempt, they defined the learning rule for the monosynaptic excitation. The learning process was based on three spikes (two presynaptic ones and a postsynaptic one) generated during each learning cycle. The first presynaptic spike at the time t_1^{in} was considered as an input signal, whereas the second presynatpic spike at $t_2^{\text{in}} = t^d$ pointed to the target firing time for the postsynaptic neuron. The learning rule is

$$\Delta w = \eta(t^{\text{out}} - t^d), \tag{10}$$

where $\eta > 0$ is the learning rate and t^{out} is the actual time of the postsynaptic spike. This learning rule was applied after each learning cycle. It is easy to demonstrate, that under certain conditions, t^{out} converges to t^d.

With this method it was possible to train only a single synaptic input, whereas neurons usually receive their inputs from several presynaptic neurons. The corresponding synaptic weights could still be learned in the way described above if the weights were learned sequentially (a single synapse per learning cycle). This is, however, a very inefficient approach.

As a solution to this problem, the authors proposed a parallel algorithm. The learning rules for the parallel algorithm are

$$\begin{cases} \Delta w_i = \eta(t^d - t_i^{\text{in}}), \quad 1 \le i \le n, \\ \text{normalize the resulting weight vector } w, \\ \qquad\qquad \text{such that } \|w\| = 1. \end{cases} \tag{11}$$

Surprisingly, although the algorithm defined by (11) is considered as an extension to the monosynaptic rule, it does not aim at achieving the desired timing of the postsynaptic neuron. Instead, the goal is to modify synaptic weights to approach some target weight vector w^d given by the difference between pre- and postsynaptic firing times, i.e., $w_i^d = (t^d - t_i^{\text{in}})$ for any presynaptic neuron i. The authors claim that such an approach can be useful in temporal pattern analysis in SNNs, but no details are given to explain it.

A thorough analysis of the supervised Hebbian learning in the context of spiking neurons was performed by Legenstein *et al.* (2005). The learning method implements the STDP process with supervision realised by extra input currents injected to the learning neuron. These currents forced the learning neuron to fire at the target times and prevented it from firing at other times. The authors investigated the suitability of this approach to learn any given transformation of the input to output spiking sequences.

In STDP, it is commonly assumed that the weight changes Δw of the particular synaptic connection are proportional to

$$\begin{cases} +A_+ \exp\left(-s/\tau_+\right) & \text{if} \quad s > 0, \\ -A_- \exp\left(s/\tau_-\right) & \text{if} \quad s \leq 0, \end{cases} \tag{12}$$

with the constants $A_+, A_-, \tau_+, \tau_- > 0$ and s being the delay between the pre- and postsynaptic firings.

The common version of STDP always produces a bimodal distribution of weights, where each weight assumes either its minimal or its maximal possible value. Therefore, in that article the authors considered mostly the target transformations that could be implemented with that bimodal distribution of weights. However, the learning algorithm was also tested with a multiplicative variation of STDP (Guetig *et al.*, 2003) which takes the form

$$\Delta w = \begin{cases} +A_+(1-w)^\mu \exp\left(-s/\tau_+\right) & \text{if} \quad s > 0, \\ -A_- w^\mu \exp\left(s/\tau_-\right) & \text{if} \quad s \leq 0, \end{cases} \tag{13}$$

where μ is a non-negative exponent. In contrast to the standard STDP, this modified rule allowed producing intermediate stable weight values. However, the authors reported that learning with this modified version of STDP was highly sensitive to input signal distributions.

In (Legenstein *et al.*, 2005), the authors demonstrate a set of experiments in which they consider different options of uncorrelated and correlated inputs with a pure and noisy teacher signal. In all experiments, LIF neuron models and dynamic synapses models were used (Maass and Zador, 1999; Markram *et al.*, 1998). However, synaptic plasticity was considered only for excitatory connections.

The results reported in (Legenstein *et al.*, 2005) demonstrated that the learning algorithm was able to approximate the given target transformations quite well. These good results were achieved not only for the case where synaptic weights were adjustable parameters, but also for a more realistic interpretation suggested by experimental results, where STDP modulated the initial release probability of dynamic synapses (Maass and Zador, 1999).

Legenstein *et al.* proved that the method has the convergence property in the average case for arbitrary uncorrelated Poisson input spike trains. On the other hand, the authors demonstrated that convergence cannot be guaranteed in a general case.

The authors reported the following drawback of the algorithm considered: Since teacher currents suppress all undesired firings during the training, the only correlations of pre- and postsynaptic activities occur around the target firing times. At other times, there is no correlation and thus no mechanism to weaken these synaptic weights that led the neuron to fire at undesired times during the testing phase.

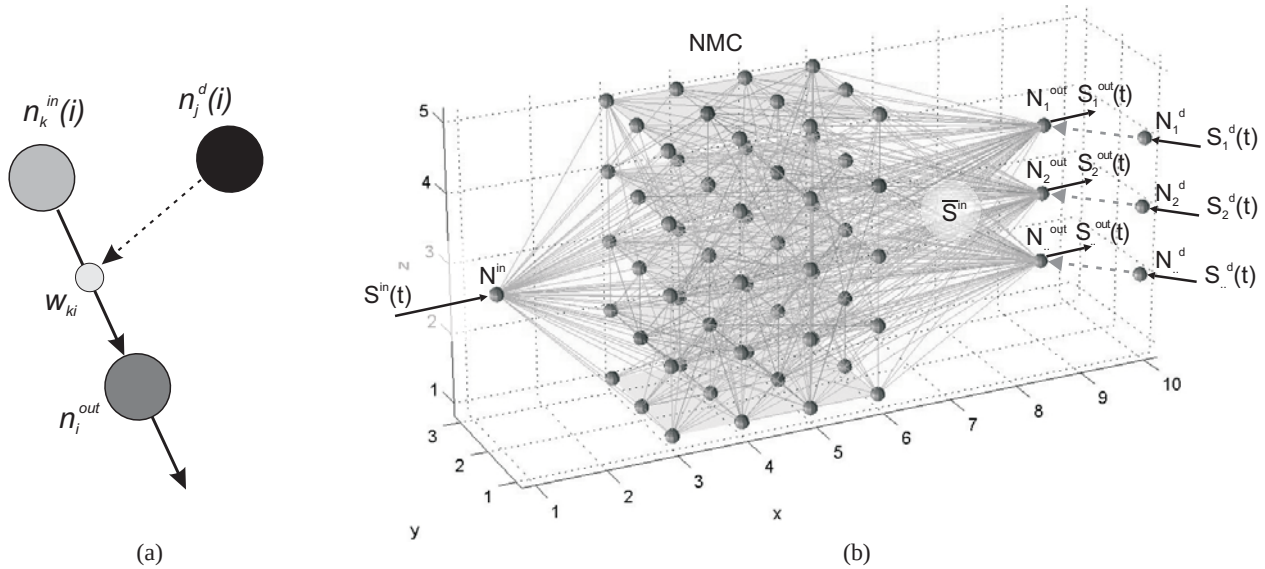
Another reported problem is common to all supervised Hebbian approaches: Synapses continue to change their parameters even if the neuron fires already exactly at the desired times. Thus, stable solutions can be achieved only by applying some additional constraints or extra learning rules to the original SHL.

Despite these problems, the presented approach proves a great ability to implement the precise spike timing coding scheme. Moreover, this is the first method out of so far presented in this article that allows learning the target transformations from the input to output spike trains.

2.7. ReSuMe – Remote Supervision

In Section 2.6 we have seen that the supervised Hebbian approach demonstrated interesting learning properties. With this approach it was feasible not only to learn the desired sequences of spikes, but also to reconstruct the target input-output transformations. Moreover, this approach inherited interesting properties of the traditional Hebbian paradigm: it is local in time and space, simple and thus suitable for online processing. On the other hand, it was demonstrated that SHL displays several serious disadvantages that may yield problems when more complex learning tasks are considered.

Here we discuss *ReSuMe*, the Remote Supervised Method proposed in (Ponulak, 2005). It is argued that the method possesses interesting properties of the supervised Hebbian approach while avoiding its drawbacks.



(a) (b)

Fig. 1. (a) Concept of *remote supervision*: The target spike train, transmitted via the neuron $n_j^d(i)$, is not directly delivered to the learning neuron n_i^{out}. However, it determines (illustrated by the dotted line) the changes of the synaptic efficacy w_{ki}, between a presynaptic neuron $n_k^{\text{in}}(i)$ and n_i^{out}; (b) *Network architecture*: ReSuMe implemented in the modified Liquid State Machine.

The goal of *ReSuMe* learning is to impose on a neural network the desired input-output properties, i.e., to produce the desired spike trains in response to the given input sequences.

ReSuMe takes advantage of Hebbian (correlation) processes and integrates them with a novel concept of remote supervision. The name *remote supervision* comes from the fact that the target signals are not directly fed to learning neurons (as is the case in SHL). However, they still co-determine the changes in synaptic efficacies in the connections terminating at the learning neurons. This is schematically illustrated in Fig. 1(a).

In *ReSuMe*, synaptic weights are modified according to the following equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}w(t) = \left[S^d(t) - S^l(t)\right]\left[a + \int_0^\infty W(s)\, S^{\text{in}}(t-s)\, \mathrm{d}s\right], \quad (14)$$

where $S^d(t)$, $S^{\text{in}}(t)$ and $S^l(t)$ are the target, pre- and postsynaptic spike trains, respectively. The parameter a expresses the amplitude of non-correlation contribution to the total weight change while the convolution function represents Hebbian-like modifications of w. The integral kernel $W(s)$ is defined over a time delay s between spikes occurring at synaptic sites (Ponulak, 2005). In the case of excitatory synapses, the term a is positive and the learning window $W(s)$ has a shape similar as in STDP, cf. (12). In the case of inhibitory synapses, a is negative and $W(s)$ is defined similarly as for anti-STDP rules.

The *ReSuMe* method is biologically plausible since it is based on Hebbian-like processes. Also the remote supervision concept, applied to *ReSuMe*, can be biologically justified on the basis of heterosynaptic plasticity – a phenomenon recently observed in neurophysiological experiments (Bi, 2002; Bonhoeffer *et al.*, 1989; Tao *et al.*, 2000).

The high learning ability of *ReSuMe* has been confirmed through extensive simulation experiments (Kasiński and Ponulak, 2005; Ponulak, 2005; Ponulak and Kasiński, 2005). Here we present the results of an experiment discussed in (Ponulak, 2005), where *ReSuMe* was used to train a network to produce the desired sequence of the spikes $S^d(t)$ in response to a given, specified input spike train $S^{\text{in}}(t)$ (Fig. 2). *ReSuMe* was applied to the LSM network consisting of 800 LIF neurons (Fig. 1(b)). Both $S^{\text{in}}(t)$ and $S^d(t)$ signals were generated randomly over a time interval of 400 ms (Figs. 2(a) and (c)). A single learning neuron was trained over 100 learning sessions. An output signal $S^{\text{out}}(t)$ at the initial state of the learning and the trained output $\hat{S}^{\text{out}}(t)$ are depicted in Figs. 2(b) and (d), respectively. It is evident that after the training, the resulting output sequence $\hat{S}^{\text{out}}(t)$ is almost identical with the target $S^d(t)$.

Other experiments presented in (Kasiński and Ponulak, 2005) demonstrated that *ReSuMe* is able not only to learn temporal sequences of spikes with high accuracy, but

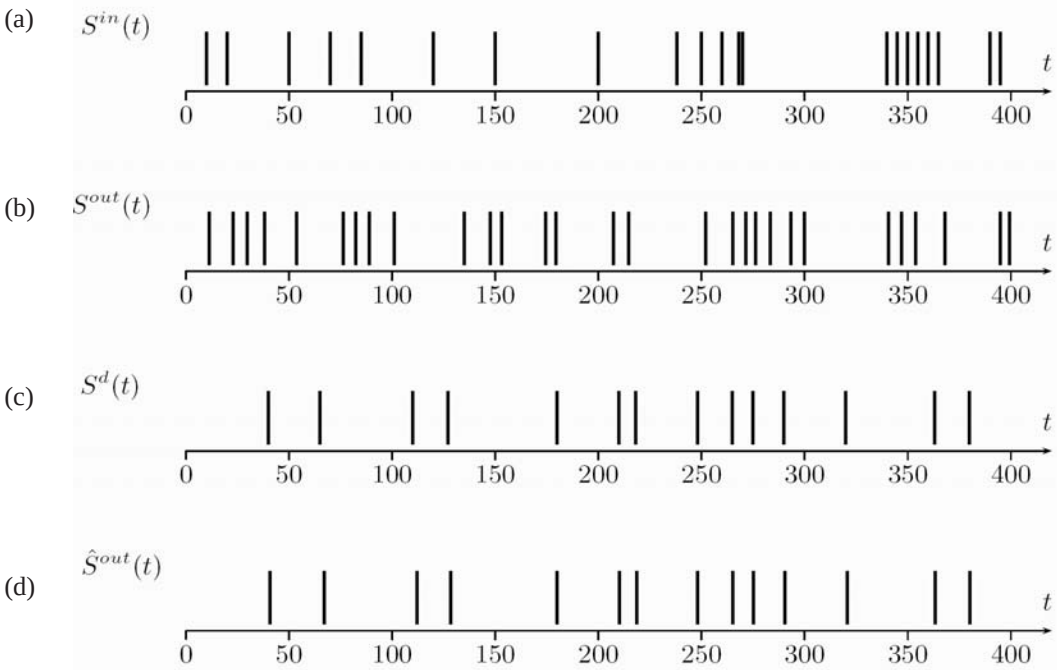


Fig. 2. Results of an experiment on spike sequence learning with ReSuMe: (a) Input spike train $S^{\text{in}}(t)$, (b) output $S^{\text{out}}(t)$ before training, (c) target spike train $S^{d}(t)$ desired at the output, (d) output $\hat{S}^{\text{out}}(t)$ after 100 learning sessions.

also to model the input/output properties of static objects, by learning multiple pairs of input-output patterns. Moreover, with *ReSuMe* implemented in the LSM architecture it was possible to successfully assign different tasks to the individual network outputs.

Another advantage of *ReSuMe* is that the method is independent of the neuron models used. For this reason, it can be implemented in networks with combined different neuron models and potentially also in networks of biological neurons. This was confirmed in the experiments where *ReSuMe* was successfully tested with LIF and Hodgkin-Huxley neuron models (Kasiński and Ponulak, 2005).

In (Ponulak and Kasiński, 2005), it was demonstrated that *ReSuMe* could efficiently generate the desired trajectories of movement for biologically realistic dynamic models of limbs. The network was trained to reconstruct spatio-temporal patterns of spikes corresponding to the patterns of activity in the pools of motoneurons. Each pool, consisting of 40 neurons, activated the particular muscle model. The simplified limb model, driven by the SNN, was able to follow the desired trajectory of movement with a satisfactory precision.

The experiments discussed here confirmed that *ReSuMe* can efficiently learn the desired temporal and spatio-temporal patterns of spikes and that the learning process converges quickly. On the basis of these results it can be concluded that *ReSuMe* is an efficient learning method, potentially suitable for real-life and real-time applications.

3. Discussion

In this review we discussed a number of learning methods suitable for the spike time coding scheme. These methods use diverse approaches to perform the learning task. Bohte *et al.* (2000) proposed to estimate the gradient of an error function explicitly. This, however, requires a special assumption on the linearity of the threshold function. Pfister *et al.* (2003) performed learning by gradient ascent on the probabilistic basis. A strictly mathematical approach was introduced in (Carnell and Richardson, 2004), where the authors defined special algebraic operations on the time series and applied these operations to the iterative algorithm for spiking patterns learning. Another interesting solution was presented in (Belatreche *et al.*, 2003). The authors introduced there an evolutionary algorithm that modifies synaptic weights and delays in order to obtain precise timing of a postsynaptic spike. Learning in the Synfire Chain network is discussed in (Sougne, 2001). The author presents there a chaining rule which allows finding links between two neuron layers firing in the desired temporal contiguity. The methods developed in (Ruf and Schmitt, 1997) and (Legenstein *et al.*, 2005) represent the supervised Hebbian approach. This approach possesses several interesting properties, such as locality, scalability and the ability of on-line processing. Legenstein *et al.* demonstrated that their method enables efficient learning of temporal patterns of spikes (Legenstein *et al.*, 2005). Moreover, it also allows imposing on a

Table 1. Supervised learning methods for spike time coding in SNN

No	Author/ *Method*	Approach	Presented Tasks	Coding Scheme	References
1	S. Bohte/ *SpikeProp*	Gradient estimation	Classification	Time-to-first-spike	(Bohte, 2003; Bohte *et al.*, 2000; 2002; Moore, 2002; Schrauwen and Van Campenhout, 2004; Xin and Embrechts, 2001)
2	J. Pfister, D. Barber, W. Gerstner	Statistical approach	Spike sequence learning, Classification	Precise spike timing	(Barber, 2003; Pfister *et al.*, 2003; 2005)
3	A. Carnell, D. Richardson	Linear algebra formalisms	Spike sequence learning	Precise spike timing	(Carnell and Richardson, 2004)
4	A. Belatreche, L. Maguire, T. McGinnity, Q. Wu	Evolutionary strategy	Classification	Time-to-first-spike	(Belatreche *et al.*, 2003)
5	J. Sougne	Learning in synfire chains	Single temporal interval learning	Relative spike time	(Sougne, 2001)
6	B. Ruf, M. Schmitt	Supervised Hebbian learning	Classification	Time-to-first-spike	(Ruf, 1998; Ruf and Schmitt, 1997)
7	R. Legenstein, Ch. Naeger, W. Maass	Supervised Hebbian learning	Spike sequence learning, Input-output mapping	Precise spike timing	(Legenstein *et al.*, 2005)
8	F. Ponulak/ *ReSuMe*	Remote supervision	Spike sequence learning, Input-output mapping, Neuron model independence, Real-life applications	Precise spike timing	(Kasiński and Ponulak, 2005; Ponulak, 2005; Ponulak and Kasiński, 2005)

neural network the desired input-output transformations through learning. On the other hand, this approach suffers from some drawbacks that make the learning task more difficult (cf. Section 2.6).

A novel concept of remote supervision (*ReSuMe*) was introduced and discussed in (Ponulak, 2005). It was shown that this method inherits the advantages of the supervised Hebbian learning, while avoiding its drawbacks. It was demonstrated that *ReSuMe* can efficiently learn multiple temporal and spatio-temporal patterns of spikes with a high precision. Moreover, *ReSuMe* was successfully applied to the real-life task of movement generation (Ponulak and Kasiński, 2005).

Finally, we observe that only four of the discussed methods demonstrate the ability to learn temporal sequences consisting of multiple spikes. The remaining algorithms address rather single spike coding schemes.

The main facts about all reviewed methods are summarized in Table 1.

4. Conclusions

In this review we considered supervised methods for spike time learning in neural networks. Our study was motivated by the recent experimental results on information coding in biological neural networks which suggested that precise timing of individual spikes may be fundamental for efficient computation in the brain.

We presented various approaches to the task considered and discussed the properties of the particular approaches. The review revealed that the methods presented in (Carnell and Richardson, 2004; Legenstein *et al.*, 2005; Pfister *et al.*, 2005; Ponulak, 2005) are capable of

learning complex temporal and spatio-temporal spike sequences. This capability allows us to create a universal tool to implement any of the temporal coding schemes proposed in the literature (Bohte, 2004; Gerstner and Kistler, 2002a; Maass and Bishop, 1999; Thorpe *et al.*, 2001).

It is worth mentioning that there is a number of other supervised methods for learning in SNNs. They were proposed and investigated, e.g., in (Koerding and Koenig, 2000; 2001; Maass *et al.*, 2002; Natschlaeger *et al.*, 2002; Pavlidis *et al.*, 2005; Seung, 2003; Xie and Seung, 2004). However, these methods are beyond the scope of this review, since they address other information coding paradigms than spike-time coding.

Finally, we remark that further investigations in the area considered are important in order to fully understand the mechanisms of information processing in biological neural systems and to use their computational power in ANN/SNN.

Acknowledgments

The work was partially supported by the Polish Ministry of Education and Science, project no. 1445/T11/2004/27.

References

Barber D. (2003): *Learning in spiking neural assemblies*, In: Advances in Neural Information Processing Systems 15, (S.T.S. Becker and K. Obermayer, Eds.). — MIT Press, Cambridge, MA, pp. 149–156.

Belatreche A., Maguire L.P., McGinnity M. and Wu Q.X. (2003): *A Method for Supervised Training of Spiking Neural Networks*. — Proc. IEEE Conf. *Cybernetics Intelligence – Challenges and Advances, CICA'2003*, Reading, UK, pp. 39–44.

Bi G.Q. (2002): *Spatiotemporal specificity of synaptic plasticity: Cellular rules and mechanisms*. — Biol. Cybern., Vol. 87, No. 5–6, pp. 319–332.

Bienenstock E. (1995): *A Model of Neocortex*. — Network: Computation in Neural Systems, Vol. 6, No. 2, pp. 179–224.

Bohte S. (2003): *Spiking Neural Networks*. — Ph.D. thesis, University of Amsterdam, Faculty of Mathematics and Natural Sciences, available at: `http://homepages.cwi.nl/~bohte`.

Bohte S., Kok J. and La Poutré H. (2000): *Spike-prop: Error-backprogation in multi-layer networks of spiking neurons*. — Proc. Euro. Symp. *Artificial Neural Networks ESANN'2000*, Bruges, Belgium, pp. 419–425.

Bohte S., Kok J. and Poutr'e H.L. (2002): *Error-backpropagation in temporally encoded networks of spiking neurons*. — Neurocomp., Vol. 48, No. 1–4, pp. 17–37.

Bohte S.M. (2004): *The Evidence for Neural Information Processing with Precise Spike-times: A Survey*. — Natural Comput., Vol. 3, No. 4, pp. 195–206.

Bonhoeffer T., Staiger V. and Aertsen A. (1989): *Synaptic plasticity in rat hippocampal slice cultures: Local Hebbian conjunction of pre- and postsynaptic stimulation leads to distributed synaptic enhancement*. — Proc. Nat. Acad. Sci. USA, Vol. 86, No. 20, pp. 8113–8117.

Carnell A. and Richardson D. (2004): *Linear algebra for time series of spikes*. — Available at: `http://www.bath.ac.uk/~Emasdr/spike.ps`

Cohen H. (1993): *A Course in Computational Algebraic Number Theory*. — New York: Springer.

Gabbiani F. and Midtgaard J. (2001): *Neural information processing*, In: Encyclopedia of Life Sciences, Nature Publishing Group, `www.els.net`, Vol. 0, pp. 1–12.

Gerstner W. and Kistler W. (2002a): *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. — Cambridge: Cambridge University Press.

Gerstner W. and Kistler W. (2002b): *Mathematical formulations of Hebbian learning*. — Biol. Cybern., Vol. 87, No. 5–6, pp. 404–415.

Gerstner W., Kempter R., van Hemmen J. and Wagner H. (1996): *A neuronal learning rule for sub-millisecond temporal coding*. — Nature, Vol. 383, No. 6595, pp. 76–78.

Guetig R., Aharonov R., Rotter S. and Sompolinsky H. (2003): *Learning input correlations through non-linear temporally asymmetric Hebbian plasticity*. — J. Neurosci., Vol. 23, No. 9, pp. 3697–3714.

Hebb D. (1949): *The Organization of Behavior*. — Cambridge: Wiley.

Hertz J., Krogh A. and Palmer R. (1991): *Introduction to the Theory of Neural Networks*. — Redwood-City, CA: Addison-Wesley.

Kasiński A. and Ponulak F. (2005): *Experimental demonstration of learning properties of a new supervised learning method for the spiking neural networks*, In: Proc. 15-th Int. Conf. *Artificial Neural Networks: Biological Inspirations*, Vol. 3696, Lecture Notes in Computer Science. — Berlin: Springer, pp. 145–153.

Kepecs A., Van Rossum M., Song S. and Tegner J. (2002): *Spike-timing-dependent plasticity: common themes and divergent vistas*. — Biol. Cybern., Vol. 87, No. 5–6, pp. 446–458.

Kistler W. (2002): *Spike-timing dependent synaptic plasticity: A phenomenological framework*. — Biol. Cybern., Vol. 87, No. 5–6, pp. 416–427.

Koerding K. and Koenig P. (2000): *Learning with two sites of synaptic integration*. — Network: Comput. Neural Syst., Vol. 11, pp. 1–15.

Koerding K. and Koenig P. (2001): *Supervised and unsupervise learning with two sites of synaptic integration*. — J. Comp. Neurosci., Vol. 11, No. 3, pp. 207–215.

Legenstein R., Naeger C. and Maass W. (2005): *What can a neuron learn with spike-timing-dependent plasticity?* — (submitted).

Maass W. (1997): *Networks of spiking neurons: The third generation of neural network models*. — Neural Netw., Vol. 10, No. 9, pp. 1659–1671.

Maass W. (1998): *On the role of time and space in neural computation*, In: Proc. Federated Conf. *CLS'98* and *MFCS'98*, Mathematical Foundations of Computer Science 1998, Vol. 1450, Lecture Notes in Computer Science. — Berlin: Springer, pp. 72–83.

Maass W. (1999): *Paradigms for computing with spiking neurons*, In: Models of Neural Networks, (L. van Hemmen, Ed.). — Berlin: Springer.

Maass W. (2002): *Paradigms for computing with spiking neurons*, In: Models of Neural Networks. Early Vision and Attention, (J.L. van Hemmen, J.D. Cowan and E. Domany, Eds.). — New York: Springer, pp. 373–402.

Maass W. (2003): *Computation with spiking neurons*, In: The Handbook of Brain Theory and Neural Networks, 2nd Ed., (M. Arbib, Ed.). — MIT Press, Cambridge, pp. 1080–1083.

Maass W. and Bishop C. (Eds.) (1999): *Pulsed Neural Networks*. — Cambridge: MIT Press.

Maass W. and Zador A. (1999): *Computing and learning with dynamic synapses*. — NeuroCOLT2 Technical Report Series NC2-TR-1999-041.

Maass W., Natschlaeger T. and Markram H. (2002): *Real-time computing without stable states: A new framework for neural computation based on perturbations*. — Neural Comput., Vol. 14, No. 11, pp. 2531–2560.

Markram H., Wang Y. and Tsodyks M. (1998): *Differential signaling via the same axon of neocortical pyramidal neurons*. — Proc. Nat. Acad. Sci., Vol. 95, No. 9, pp. 5323–5328.

Markram H., Luebke J. and Frotscher B.S.M. (1997): *Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs*. — Science, Vol. 275, No. 5297, pp. 213–215.

Mehta M.R., Lee A.K. and Wilson M.A. (2002): *Role of experience of oscillations in transforming a rate code into a temporal code*. — Nature, Vol. 417, No. 6891, pp. 741–746.

Moore S.C. (2002): *Back-Propagation in Spiking Neural Networks*. — M.Sc. thesis, University of Bath, available at: `http://www.simonchristianmoore.co.uk`.

Natschlaeger T., Maass W. and Markram H. (2002): *The "liquid computer", a novel strategy for real-time computing on time series*. — Special issue on Foundations of Information Processing of TELEMATIK, Vol. 8, No. 1, pp. 32–36.

Pavlidis N.G., Tasoulis D.K., Plagianakos V.P., Nikiforidis G. and Vrahatis M.N. (2005): *Spiking neural network training using evolutionary algorithms*. — Proc. Int. Joint Conf. *Neural Networks, IJCNN'05*, Montreal, Canada.

Pfister J.P., Barber D. and Gerstner W. (2003): *Optimal Hebbian Learning: A Probabilistic Point of View*, In: ICANN/ICONIP 2003, Vol. 2714, Lecture Notes in Computer Science. — Berlin: Springer, pp. 92–98.

Pfister J.-P., Toyoizumi T., Barber D. and Gerstner W. (2005): *Optimal spike-timing dependent plasticity for precise action potential firing*. — (submitted), available at: `http://diwww.epfl.ch/~jpfister/papers/Pfister_05a.pdf`.

Ponulak F. (2005): *ReSuMe—new supervised learning method for spiking neural networks*. — Tech. Rep., Institute of Control and Information Engineering, Poznan University of Technology, available at: `http://d1.cie.put.poznan.pl/~fp/`.

Ponulak F. and Kasiński A. (2005): *A novel approach towards movement control with spiking neural networks*. — Proc. 3-rd Int. Symp. *Adaptive Motion in Animals and Machines*, Ilmenau, (Abstract).

Popović D. and Sinkjaer T. (2000): *Control of Movement for the Physically Disabled*. — London: Springer.

Ruf B. (1998): *Computing and Learning with Spiking Neurons – Theory and Simulations*. — Ph.D. thesis, Institute for Theoretical Computer Science, Technische Universitaet Graz, Austria.

Ruf B. and Schmitt M. (1997): *Learning temporally encoded patterns in networks of spiking neurons*. — Neural Proces. Lett., Vol. 5, No. 1, pp. 9–18.

Rumelhart D., Hinton G. and Williams R. (1986): *Learning representations by back-propagating errors*. — Nature, Vol. 323, pp. 533–536.

Schrauwen B. and Van Campenhout J. (2004): *Improving SpikeProp: Enhancements to an Error-Backpropagation Rule for Spiking Neural Networks*. — Proc. 15-th *ProRISC Workshop*, Veldhoven, the Netherlands.

Seung S. (2003): *Learning in spiking neural networks by reinforcement of stochastic synaptic transmission*. — Neuron, Vol. 40, No. 6, pp. 1063–1073.

Sougne J.P. (2001): *A learning algorithm for synfire chains*, In: Connectionist Models of Learning, Development and Evolution, (R.M. French and J.P. Sougne, Eds.). — London: Springer, pp. 23–32.

Spears W.M., Jong K.A.D., Baeck T., Fogel D.B. and de Garis H. (1993): *An overview of evolutionary computation*. — Proc. Europ. Conf. *Machine Learning*, Vienna, Austria, Vol. 667, pp. 442–459.

Tao H.-Z.W., Zhang L.I., Bi G.-Q. and Poo M.-M. (2000): *Selective presynaptic propagation of long-term potentiation in defined neural networks*. — J. Neurosci., Vol. 20, No. 9, pp. 3233–3243.

Thorpe S. J., Delorme A. and VanRullen R. (2001): *Spike-based strategies for rapid processing*. — Neural Netw., Vol. 14, No. 6–7, pp. 715–726.

Tiňo P. and Mills A.J. (2005): *Learning beyond finite memory in recurrent networks of spiking neurons*, In: Advances in Natural Computation – ICNC 2005, (L. Wang, K. Chen and Y. Ong, Eds.), Lecture Notes in Computer Science. — Berlin: Springer, pp. 666–675.

VanRullen R., Guyonneau R. and Thorpe S.J. (2005): *Spike times make sense*. — TRENDS in Neurosci., Vol. 28, No. 1, pp. 1–4.

Weisstein E.W. (2006): *Gram-Schmidt Orthonormalization, from MathWorld–A Wolfram Web Resource*. — Available at: `http://mathworld.wolfram.com/Gram-SchmidtOrthonormalization.html`.

Xie X. and Seung S. (2004): *Learning in neural networks by reinforcement of irregular spiking*. — Phys. Rev., Vol. 69, No. 4, pp. 1–10.

Xin J. and Embrechts M.J. (2001): *Supervised Learning with Spiking Neuron Networks*. — Proc. IEEE Int. Joint Conf. *Neural Networks, IJCNN'01*, Washington D.C., pp. 1772–1777.

Received: 17 November 2005
Revised: 2 March 2006