Learning Algorithm for Spiking Neural Networks

Hesham H. Amin and Robert H. Fujii

The University of Aizu, Aizu-Wakamatsu, Fukushima, Japan
{d8042201, fujii}@u-aizu.ac.jp

Abstract. Spiking Neural Networks (SNNs) use inter-spike time coding to process input data. In this paper, a new learning algorithm for SNNs that uses the inter-spike times within a spike train is introduced. The learning algorithm utilizes the spatio-temporal pattern produced by the spike train input mapping unit and adjusts synaptic weights during learning. The approach was applied to classification problems.

1 Introduction

Spiking Neural Networks (SNN) can be considered as the third generation of ANNs, after multi-layer perceptron neural networks and neurons which employ activation functions such as sigmoid functions [10]. The latter two types of neural networks use synchronized analog or digital amplitude values as inputs. SNNs do not require a synchronizing system clock (although they may use a local synchronizing signal) and utilize input inter-spike time data to process information. An SNN is composed of spiking neurons as processing units which are connected together with synapses. A spiking neuron receives spikes at its inputs and fires an output spike at a time dependent on the inter-spike times of the input spikes. Thus, SNNs use temporal information in coding and processing input data. Synaptic spike inputs with only one spike per each input synapse during a given time window are called spatio-temporal inputs. A synaptic input which consists of a sequence of spikes with various inter-spike intervals (ISIs) during a given time window is called a spike train. The ISI times within a spike train has a much larger encoding space than the rate code used in traditional neural networks [11]. Accordingly, the processing efficiency of SNNs can be higher than traditional rate code based ANNs for most applications.

Learning how to recognize the temporal information contained in spike trains is the main goal of this research. The literature is scant regarding this area of research. Some SNN learning models have been proposed in the past which make it possible to process spike trains in close to real-time [8], [9], [12], [13]. However, these models used recurrent networks and a large number of synapses which needed a relatively long time to map and process input spike trains. In this paper, a new learning algorithm for spiking neurons which use spike trains inputs is proposed. This learning algorithm utilizes the input spike mapping scheme, described in [1],[2], and input synapses with dynamically changeable weights.

2 Spiking Neural Network

The spiking neuron model employed in this paper is based on the Spike Response Model (SRM) [6] with some modifications. Input spikes come at times $\{t_1...t_n\}$ into the input synapse(s) of a neuron. The neuron outputs a spike when the internal neuron membrane potential $x_j(t)$ crosses the threshold potential ϑ from below at firing time $t_j = min\{t : x_j(t) \geq \vartheta\}$. The threshold potential ϑ is assumed to be constant for the neuron.

The relationship between input spike times and the internal potential of neuron j (or Post Synaptic Potential (PSP)) $x_j(t)$ can be described as follows:

$$x_j(t) = \sum_{i=1}^{n} W_i.\alpha(t - t_i), \quad \alpha(t) = \frac{t}{\tau} e^{1-\frac{t}{\tau}} \tag{1}$$

i represents the ith synapse, W_i is the ith synaptic weight variable which can change the amplitude of the neuron potential $x_j(t)$, t_i is the ith input spike arrival-time, $\alpha(t)$ is the spike response function, and τ represents the membrane potential decay time constant.

In this paper, the $\alpha(t)$ function is approximated as a linear function for $t << \tau$. It then follows that the internal neuron potential Equation 1, can be re-written as:

$$x_j(t) = \frac{t}{\tau_1} \sum_{i=1}^{n} W_i.u(t - t_i); \quad t \ll \tau_1 \tag{2}$$

$u(t)$ is the Heaviside function and $\tau_1 = \frac{e}{\tau}$.

3 Mapping-Learning Scheme for Spiking Neural Networks

A one-to-one correspondence between input spike trains and output spike firing times is necessary for the learning algorithm proposed in this paper. By selecting an appropriate set of synaptic weights for a neuron, a particular spike train or a set of spike trains which belong to the same class can be distinguished by the output firing time of the neuron because of the one-to-one correspondence between the input and output. The combined mapping-learning organization is shown in Figure 1.

Learning is performed in two stages: (1) The *mapping stage* is composed of neural mapping units (MUs) as shown in Figure 1. This stage was described in [1],[2] and it is used for mapping the input spike train(s) into unique spatio-temporal output patterns. The one-to-one relationship between the inputs and outputs of the mapping stage was proved in Appendix A of [2]. (2) The *learning stage* consists of several learning units (LUs) as shown in Figure 1. The learning stage receives the spatio-temporal output pattern produced by the mapping stage. Each learning unit is composed of sub-learning units as shown in Figure 2(A). Each sub-learning unit (e.g LUA1) takes inputs from one mapping unit (MU) as shown in Figure 2(B). As shown in Figure 2(B), the outputs t_1 and t_2 from the mapping unit are input into the sub-learning unit ISI blocks. The ISI block performs the same function as the ISI block used in the mapping units used in [1],[2]; the learning unit ISI block input synaptic weights are assigned using $W_i = \beta.t_i$ and $W_i = \frac{\beta}{t_i}$ for the ISI1 and ISI2 blocks respectively. It should be noted that in a

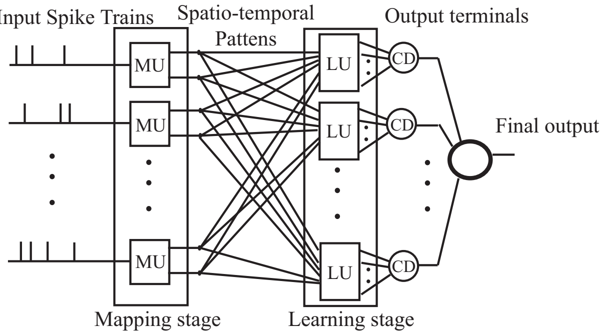

Fig. 1. Combined Mapping-Learning Organization

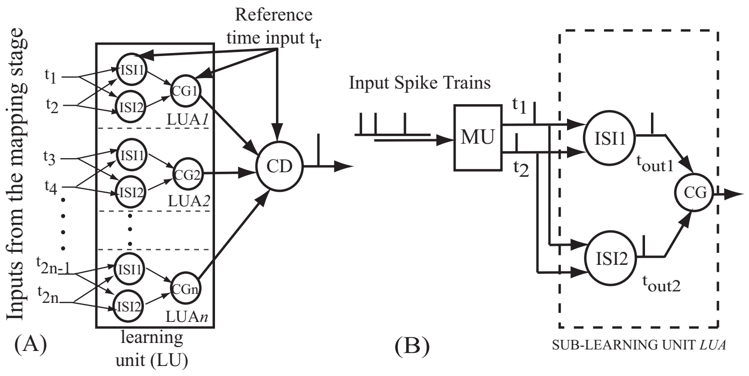

Fig. 2. (A) Details of Learning Unit (LU). (B) MU and sub-learning unit

learning unit there are $2n$ ISI blocks where n is the number of input spike trains. A one-to-one mapping between inputs and outputs is also necessary in the learning unit. The t_r reference time input shown in Figure 2(A) is used as a local reference signal for the combined mapping-learning organization shown in Figure 1. The coincidence generation (CG) neurons in a sub-learning unit perform the function of aligning their output spike times. When all CG neurons in an LU fire simultaneously, the coincidence detection (CD) neuron fires.

Past learning algorithms for spiking neural networks such as back-propagation (SpikeProp) [4], self-organizing map (SOM) [15], and radial basis function (RBF) [14] used synaptic weights and delays as well as multiple sub-synapses as the learning parameters. The learning algorithm proposed in this paper can perform learning in one step and utilizes only synaptic weights for learning. Hence, the proposed algorithm is simpler than past approaches and more practical to implement in hardware.

3.1 The Learning Algorithm

The spatio-temporal patterns generated by the ISI1 and ISI2 blocks in the mapping stage, described in [1],[2], are used as inputs for the learning stage where a supervised learning method is used to classify input patterns. Clustering of input patterns which belong to the same class is achieved by setting the synaptic weights for a learning unit (LU) so that its output fires at approximately the same time for as many input spike trains as possible that belong to the same class.

The supervised learning algorithm works as follows:

1. Choose an input pattern vector (say P_A) at random from the set of $P_l = (P_A, P_B, ...)$ pattern vectors to be used for the learning phase. Each pattern P_l consists of the spatio-temporal outputs generated by the mapping stage. The randomly chosen pattern P_A is used to assign weights to all the ISI blocks in a learning unit. This learning unit will represent the class to which pattern P_A belongs. Once the weights have been assigned, they are temporarily fixed. The weights selected for the initial input pattern works as a center vector which can later be modified slightly to accommodate more than one input pattern; in this manner, similar input patterns can then be clustered together and fewer learning units will be needed.
2. Another input pattern (say P_B) belonging to the same class as pattern P_A chosen in step 1 above is selected. This new pattern is applied to the learning unit for P_A and the output of the ISI blocks times for P_B $\{t_{out1}, t_{out2}, ..., t_{out2n}\}$ are compared against the output times for $P_A\{t^*_{out1}, t^*_{out2}, ...t^*_{out2n}\}$. This new pattern ($P_B$) is assigned to the learning unit (e.g. learning unit for P_A) with which each of the output times differ by less than ϵ.

$$|t^*_{out1} - t_{out1}| \leq \epsilon \quad , |t^*_{out2} - t_{out2}| \leq \epsilon \quad , ... \quad \text{and} \quad |t^*_{out2n} - t_{out2n}| \leq \epsilon \tag{3}$$

   ϵ is a small error value determined empirically. If the error is larger than ϵ for any one of the error conditions in Equation 3 , a new learning unit is added as is done in incremental learning.
3. Steps 1 and 2 are repeated for all the remaining input patterns in the learning set P_l.

In this learning scheme, all input spike train samples used for learning must be known a priori. However, the total number of learning units (clusters) which will be needed for classification with clustering cannot be known a priori. It may be possible to cluster m input patterns belonging to one class into a single learning unit (cluster) or as many as m learning units may be needed.

This learning scheme is similar to the algorithm proposed in [14] but without the need for synapse delays. This could help to make the model more practical for an IC circuit design implementation. Furthermore, each synapse in the model is not composed of multiple sub-synapses as proposed in [3], [14] and this leads to a reduction in complexity.

The proposed learning algorithm produces locally optimal input clustering because the input patterns for a given class are sequentially chosen at random; the consequence of this is that a larger neural network than necessary may result.

3.2 Learning Unit Output Time Uniqueness

A one-to-one relationship between inputs and outputs for each of the learning units must be achieved in order to guarantee that each learning unit outputs a spike at a time

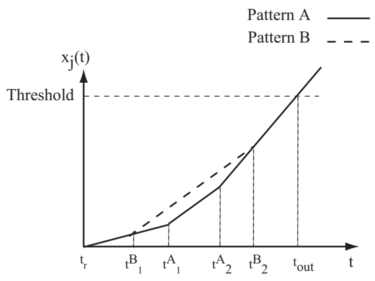

Fig. 3. Two Different Input Patterns Producing an Identical Output Time

which is different from the output times corresponding to other inputs. This one-to-one relationship will be shown using one MU and one sub-learning unit. When a new pattern (e.g. pattern P_B with MU output times t_1^B and t_2^B) is input into a sub-learning unit within an LU which had its synaptic weights fixed during the learning of pattern P_A,the following will result: $(\{t_{out1}^A, t_{out2}^A\} \neq \{t_{out1}^B, t_{out2}^B\}$, where t_{out} is the output firing time of an ISI block. This can be proved by contradiction:

Assume that P_B produces the same t_{out1} or t_{out2} as P_A. For the moment, t_{out1} and t_{out2} will not be distinguished and they will simply be referred to as t_{out}.

Then the internal neuron potentials $x_j(t)$ (Equation 2) for P_A and P_B at time t_{out} can be written as follows:

$$\sum_{i=1}^{2} W_i^A.u(t - t_i^A) = \sum_{i=1}^{2} W_i^A.u(t - t_i^B) \tag{4}$$

W_i^A's are the synaptic weights which have been fixed for the learning unit P_A. Two different input patterns P_A and P_B producing an identical output at time t_{out} can occur only if the neuron internal potential $x_j(t)$ for one of the input patterns becomes equal to the other input pattern's neuron internal potential (at t_2^B) and then both increase at identical rates until crossing the threshold potential ϑ at time t_{out} as shown in Figure 3.

For the ISI1 block, $W_i^A = \beta.t_i^A$; Equation 4 can be rewritten as follows:

$$\frac{t_1^A}{t_2^A} = \frac{u(t - t_2^B) - u(t - t_2^A)}{u(t - t_1^A) - u(t - t_1^B)} \tag{5}$$

For the ISI2 block, $W_i^A = \frac{\beta}{t_i^A}$; thus Equation 4 can be rewritten as follows:

$$\frac{t_1^A}{t_2^A} = \frac{u(t - t_1^A) - u(t - t_1^B)}{u(t - t_2^B) - u(t - t_2^A)} \tag{6}$$

Equations 5 and 6 can have a solution only if $t_1^A = t_2^A$ which cannot happen because 2 distinct spikes output times from the mapping unit are assumed[1]. In other words, if the ISI1 block outputs a spike at the same t_{out} time for both P_A and P_B, the ISI2 output times will be not equal and vice versa.

[1] $t_1^A = t_2^A$ can happen only if an input spike train consists of only two spikes at times 0 and 1 when the input spike train time window size is assumed to be equal to 1.

3.3 Firing of Only One Learning Unit

Assume that patten P_A was learned by the learning unit A(LUA) and that patten P_B was learned by the learning unit B (LUB). Assume that the sub-learning units LUA1 and LUB1 get inputs from the same mapping unit (MU). If pattern P_A is input into both LUA1 and LUB1, the neuron internal potentials for LUA's ISI1 or ISI2 and LUB's ISI1 or ISI2 will increase according to equation 2. If $t^A_{out1} = t^B_{out1} = t_{out1}$ and $t^A_{out2} = t^B_{out2} = t_{out2}$ are assumed, the following relationship will be established:

$$\sum_{i=1}^{2} W_i^A.u(t - t_i^A) = \sum_{i=1}^{2} W_i^B.u(t - t_i^A) \tag{7}$$

The only way for LUA1 and LUB1 to produce an output spike at the same $t_{out1}(t_{out2})$ time is to have the following condition satisfied:

$$\sum_{i=1}^{2} W_i^A = \sum_{i=1}^{2} W_i^B \tag{8}$$

Thus, if the condition specified by Equation 8 is not satisfied by any one of the sub-learning units, only one of the learning units will respond to an input pattern. The learning algorithm has to include a checking phase to guarantee that the condition specified by Equation 8 is not satisfied.

3.4 Coincidence Detection Neuron

In order to have only one learning unit fire for a given input pattern, output times of the CG neurons in the sub-learning units (Figure 2(A)) have to be made coincident by changing the input synaptic weight values of the coincidence generation (CG) neurons. The coincidence detection neuron (CD), shown in Figure 2(A), uses the exponential response function (Equation 1) of a spiking neuron.

The outputs of the ISI1 and ISI2 blocks of each sub-learning unit (Figure 2(A)) fire at certain times according to the assigned synaptic weight centers. The other patterns which have been joined to the same learning unit cause the outputs to fire at times which are close to the ones corresponding to the center pattern. The coincidence detection neuron threshold value ϑ is adjusted so as to allow some fuzziness in the input spike times.

3.5 Local Reference Time

In section 3.2 it was proved that the output combination $\{t_{out1}, t_{out2}\}$ for the ISI1 and ISI2 blocks will be unique for each sub-learning unit; however, the relative time $|t_{out1} - t_{out2}|$ should also be considered for all the sub-learning units of different learning units (LUs). In other words, two different sub-learning units in two different learning units can fire at different output times, t_{out1} and t_{out2}, but the relative time $|t_{out1} - t_{out2}|$ may be the same; this would lead to two (or more) learning units firing outputs for the same input pattern. Thus, a reference time (bias) t_r input is necessary to differentiate these outputs as shown in Figure 2(A). This reference time t_r is the time when the first input spike arrives at one of the mapping stage inputs.

4 Simulations

4.1 Realization of the XOR Function

Due to its nonlinearly separable input characteristics, a two-input exclusive OR (XOR) function has often been used to test the function approximation or classification capability of a neural network [7]. The XOR problem has non-linearly separable classes. One of these classes is represented by x_1x_2 inputs 00 and 11. The other class is represented by x_1x_2 inputs 01 and 10. The logical inputs ”0” and ”1” are represented by spikes at times 0 and 0.1 respectively in Table 1. The spike time can be defined with any appropriate unit of time (e.g. ms, ns).

For a spiking neural network, the inputs $x_1x_2 = 00$ and $x_1x_2 = 11$ are not distinguishable in the time domain because the inputs are not referenced to a clock. Thus, in order to distinguish the $x_1x_2 = 00$ and $x_1x_2 = 11$ cases, a third reference (bias) input $x_0 = 0$ is used as shown in Figure 4. Thus, the logical input $x1x2 = 00$ and $x1x2 = 11$ can for example be distinguished in the time domain as ”$0sec, 0sec, 0sec$” and ”$0sec, 0.1sec, 0.1sec$” respectively.

As describes in section 3.4, each learning unit in conjunction with a coincident detection neuron generates a spike when the appropriate spatio-temporal pattern is input.

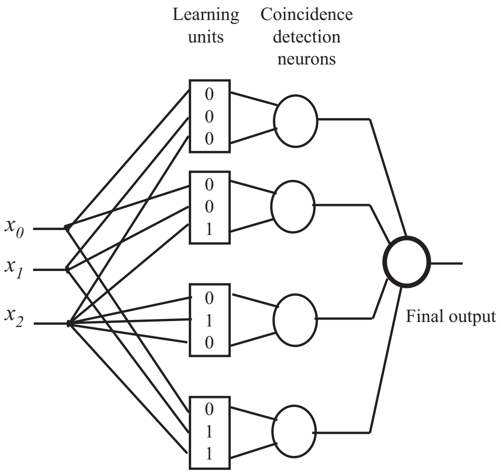

Fig. 4. Spiking neural network for XOR function. Details of learning unit also shown.

Table 1. XOR Input spike times (including the bias) and output times

Input Patterns	Coincident firing time	Final output time
0 0 0	1.464	4
0 0 0.1	1.910	2
0 0.1 0	1.910	2
0 0.1 0.1	3.013	4

The XOR neural network organization is shown in Figure 4. The final output neuron, shown in Figure 4, is used to represent the XOR output value in the time domain (e.g. output time $= 2$ corresponds to the logical output ”1”).

4.2 Classification of Spike Trains

The robustness of the learning algorithm was tested using a set of randomly generated spike trains as inputs. These spike trains were generated by adding noise to the original spike trains. Noise consisted of input spike shifts in time or addition/deletion of spikes within a spike train. These types of noise are realistic since a correct spike sequence can be altered by short-lived interferences. Spike time skews were produced by adding Gaussian white noise (GWN) to the spike train, or by time shifting one or two spikes in a spike train randomly. The deletion/addition of spikes was also done randomly.

The spike trains used in the simulations were generated using Poisson distributed inter-spike intervals [5] at a low frequency. By injecting various amounts of GWN into a spike train, noisy time shifted versions of the original spike trains could be generated as shown in Figure 5, where spike train number 1 is the original spike train for each class.

Each of the generated spike trains shown in Figure 5 was used as an input to the mapping stage (shown in Figure 1). The spatio-temporal pattern output from the mapping stage was then used as an input to the learning stage. The mapping stage used multiple mapping units with different β values in the range of $[0.25, 1.0]$ in order to increase the input dimension of the learning stage.

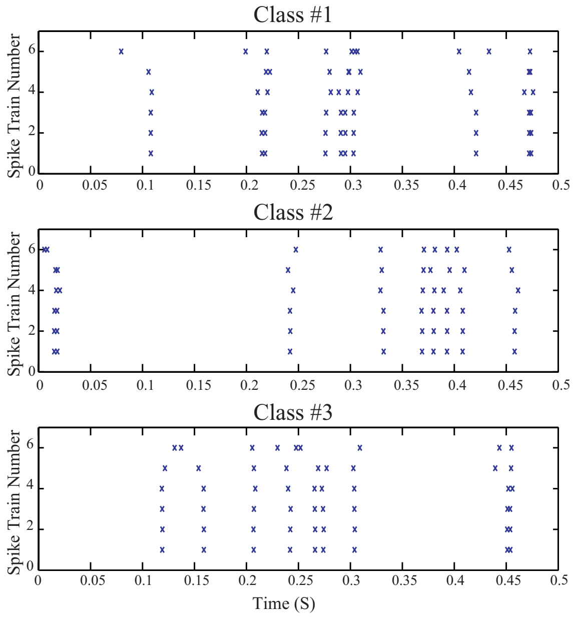

Fig. 5. Three classes of input spike trains. The original spike train for each class is spike train number 1 and the other five trains are noisy versions of it.

Table 2. Input spike train classification, clustering, and final output times

Class No.	Learning unit #	# Learning patterns	# Test patterns	Final output time
1	1	5	5	4.0
	2	1	-	
2	3	4	3	5.0
	4	2	2	
3	5	4	3	6.0
	6	2	2	

After generating the noisy versions of each of the original spike trains, all the patterns including the noisy patterns were used as a learning set. The closer the noisy versions were to the original spike train, the likelihood of being able to use an already assigned learning unit increased.

The learning and input pattern clustering simulation results are shown in Table 2. For example, for the three classes a total of six clusters were needed. For input class 1, learning unit 1 was used for clustering five input patterns and learning unit 2 was used for clustering one input pattern. Similar clusterings were possible for classes 2 and 3 as shown in Table 2.

After the learning phase was completed, additional noisy spike trains for each of the three classes were used to test the neural network. These additional noisy spike trains are called test patterns in Table 2. The testing phase spike trains were generated with the same range of noise used during the learning phase. For example, for input class 3, three input patterns were recognized by learning unit 5 and two input patterns were recognized by learning unit 6. Similar test patterns recognition were possible for classes 1 and 2 as shown in Table 2.

A final output neuron (refer to Figure 1) is used to represent the final output time value for each of the three classes as shown in Table 2.

5 Conclusions

Spiking neural networks can be used to process time domain analog real world signals once these signals have been converted into spike trains. A new learning algorithm for spiking neural networks was proposed. After learning, the resulting spiking neural network could classify input spike trains. Simulations have shown that incremental learning for classification learning of input spike trains with noise could be achieved by either adding learning units or clustering. The learning algorithm is relatively simple when compared with other neural networks learning algorithms such as back-propagation.

References

1. Hesham H. Amin and Robert H. Fujii: Input Arrival-Time-Dependent Decoding Scheme for a Spiking Neural Network. Proceeding of The 12th European Symosium of Artificial Neural Networks (ESANN 2004). (2004) 355–360.

2. Hesham H. Amin and Robert H. Fujii: Spike Train Decoding Scheme for a Spiking Neural Network. Proceedings of the 2004 International Joint Conference on Neural Networks. IEEE. (2004) 477–482.
3. S.M. Bohte, H. La Poutré and J.N. Kok: Unsupervised classification in a Network of spiking neurons. IEEE Transactions on Neural Networks. **13, 2** (2002) 426–435.
4. S.M. Bohte, J.N. Kok and H. La Poutré: Spike-prop: error-backprogation in multi-layer networks of spiking neurons. Proceedings of the European Symposium on Artificial Neural Networks ESANN'2000. (2000) 419–425.
5. Peter Dayan, and L. F. Abbott: Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems. MIT Press (2001).
6. W. Gerstner and W. Kistler: Spiking Neuron Models. Single Neurons, Populations, Plasticity. Cambridge University Press (2002).
7. Simon Haykin: Neural Networks, A Comprehensive Foundation. Prentice Hall International Inc. (1999).
8. J. J. Hopfield and C. D. Brody: What is a Moment? Cortical Sensory Integration Over a Brief Interval. Proc. Natl. Acad. Sci. **97(25)** (2000) 13919–13924.
9. J. J. Hopfield and C. D. Brody: What is a Moment? Transient Synchrony as a Collective Mechanism for Spatiotemporal Integration. Proc. Natl. Acad. Sci. **98(3)** (2001) 1282–1287.
10. W. Maass, Networks of spiking neurons: the third generation of neural network models. Neural Networks. **10**. (1997) 1659–1671
11. W. Maass and C. Bishop, editors: Pulsed Neural Networks. MIT press. Cambridge (1999).
12. W. Maass, T. Natschläger, and H. Markram: A Model for Real-Time Computation in Generic Neural Microcircuits, in *Proc. of NIPS 2002*, Advances in Neural Information Processing Systems, volume 15, ed. S. Becker, S. Thrun, and K. Obermayer. MIT Press (2003) 229–236.
13. W. Maass, T. Natschläger, and H. Markram: Computational Models for Generic Cortical Microcircuits, in *Computational Neuroscience: A Comprehensive Approach*, chapter 18, ed. J. Feng. CRC-Press (2003).
14. Berthold Ruf: Computing and Learning with Spiking Neurons - Theory and Simulations, Chapter (8), *Doctoral Thesis*, Technische Universitaet Graz, Austria (1997).
    Available: ftp://ftp.eccc.uni-trier.de/pub/eccc/theses/ruf.ps.gz
15. B. Ruf and M. Schmitt: Self-organization of spiking neurons using action potential timing. IEEE-Trans. Neural Networks. **9(3)** (1998) 575–578