Self-Organization of Spiking Neurons Using Action Potential Timing

Berthold Ruf, Michael Schmitt

Abstract

We propose a mechanism for unsupervised learning in networks of spiking neurons which is based on the timing of single firing events. Our results show that a topology preserving behaviour quite similar to that of Kohonen's self-organizing map can be achieved using temporal coding. In contrast to previous approaches, which use rate coding, the winner among competing neurons can be determined fast and locally. Our model is a further step towards a more realistic description of unsupervised learning in biological neural systems. Furthermore, it may provide a basis for fast implementations in pulsed VLSI.

Keywords

Self-organizing map, spiking neurons, temporal coding, unsupervised learning.

I. Introduction

In the area of modelling information processing in biological neural systems, there is an ongoing debate about which essentials have to be taken into account (see e.g. [1], [2], [3], [4]). Discrete models, such as threshold gates or McCulloch-Pitts neurons, are undoubtedly very simplistic descriptions of biological neurons. Models with real-valued output, such as the sigmoidal gate, where analogue values are interpreted as firing rates of biological neurons, are more suitable for the modelling of neural processes in terms of analogue computations. However, both types of models do not capture a phenomenon which is widely believed to be the basis of fast analogue computations in biological neural networks: the timing of single action potentials. Models of spiking neurons that use temporal coding are a first attempt to explore the computational significance of this phenomenon [5], [6], [3]. Furthermore, spiking neuron networks (SNNs), where the computations are based on this coding scheme, have recently been shown to be computationally more powerful than networks consisting of threshold or sigmoidal gates with respect to time and network complexity [4]. The issue of learning was raised in [7] where it was shown that supervised learning is possible in such SNNs.

In this paper we investigate unsupervised learning processes in SNNs. On the basis of a construction introduced in [8] we show how competitive learning can be performed by SNNs using temporal coding. We extend this idea to a learning mechanism that is closely related to one of the most successful paradigms of unsupervised learning: the self-organizing map (SOM) by Kohonen [9].

Topology preserving maps have been found in many regions of the brain, e.g. in the visual, auditory, or somatosensory cortex [5]. The SOM provides a possible explanation how such maps can develop. Previous versions of the SOM assume that the output of a neuron is characterized by its firing rate and not by the timing of single firing events. Using lateral connections the procedure for detecting the so-called winner neuron is usually implemented as a recurrent network. This has the consequence that the winner neuron is detected not before the network has settled down into an equilibrium state. However, since the computation relies on the convergence of the network, this approach disregards the benefits that temporal coding offers to fast information processing in SNNs.

In addition to these conventional implementations there has also been some research on biologically more realistic models of self-organizing map algorithms, e.g. by Kohonen [10], Sirosh and Miikkulainen [11], Choe and Miikkulainen [12]. Also in these approaches the output of a neuron is assumed to correspond to its firing rate, and learning takes place in terms of this rate after the network has reached a stable state of firing.

In the following we propose a mechanism for unsupervised learning in SNNs where computing and learning are based on the timing of single firing events. In contrast to the standard formulation of the SOM, our construction has the additional advantage that the winner among the competing neurons can be determined fast and locally by using lateral excitation and inhibition. These lateral connections also constitute the neighbourhood relationship among the neurons. We assume that initially neurons which are topologically close together have strong excitatory lateral connections whereas remote neurons have strong inhibitory connections. During the learning process the lateral weights are decreased, thus reducing the size of the neighbourhood.

In a series of computer simulations we have investigated the capability of the model to form topology preserving mappings. For the evaluation of these mappings, instead of relying on visual inspection, we used a measure for quantifying the neighbourhood preservation which was recently studied [13]. Our results show that the model exhibits the same characteristic behaviour as the SOM. The typical emergence of topology preserving behaviour could be observed for a wide range of parameters.

The authors are with the Institute for Theoretical Computer Science, Technische Universität Graz, Klosterwiesgasse 32/2, A-8010 Graz, Austria, e-mail: {bruf, mschmitt}@igi.tu-graz.ac.at.

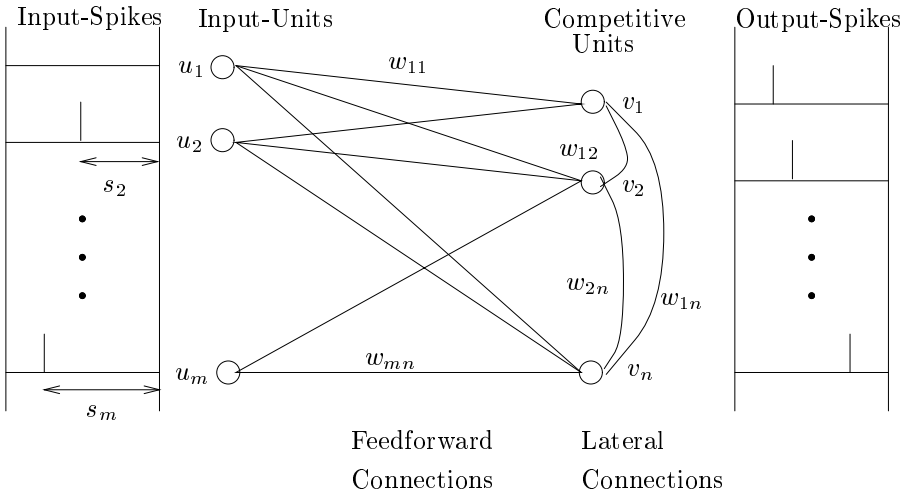

Fig. 1. The basic architecture.

We also studied the effect of weight normalization. It is used in several implementations of the SOM (see e.g. [12], [9], [11]) but its biological relevance is controversial. We performed simulations with and without weight normalization after each learning cycle. Comparing the results we discovered that after a certain number of learning cycles approximately the same degree of topology preservation could be achieved regardless whether the weights were normalized or not. The model of unsupervised learning that we propose in this paper is therefore a candidate for a more realistic description of fast analogue computation in biological neural systems. Moreover, it also provides a link to possible industrial applications via silicon implementations in pulse coded VLSI [14].

The paper is organized as follows: In Section II we review how computations with spiking neurons in temporal coding can be achieved. In Section III we propose the mechanism for unsupervised learning in networks of spiking neurons. The simulation results are summarized in Section IV.

An extended abstract of this article was presented in [15].

II. Computing with Spiking Neurons

Recently Maass has shown in [8] how leaky integrate-and-fire neurons can compute weighted sums in temporal coding, where the firing time of a neuron encodes a value in the sense that an early firing of the neuron represents a large value. More precisely, one considers a neuron v which receives excitatory input from m neurons $u_1, \ldots, u_m$, where the corresponding weights are denoted by $w_1, \ldots, w_m$. Each u_i fires exactly once within a sufficiently small time interval at a time t_i with $t_i = T_1 - s_i$, where s_i is the ith input to v and T_1 some constant. Usually T_1 is given by the time when a reference spike is fired by some additional input neuron to v. Under certain weak assumptions (basically the initial rising segments of the excitatory postsynaptic potentials have to be linear) one can guarantee that v fires at a time determined by $T_2 - \sum w_i s_i$ with T_2 being some constant.

This computation can also be performed on the basis of "competitive temporal coding", such that no explicit reference times T_1 and T_2 are necessary (see [8] for details).

III. Self-Organization

On the basis of this construction it is now possible to implement various types of unsupervised learning in the context of SNNs as follows: Given a set S of m-dimensional input vectors $\mathbf{s}^l = (s_1^l, \ldots, s_m^l)$ and an SNN with m input neurons and n competitive neurons, where each competitive neuron v_j receives synaptic "feedforward" input from each input neuron u_i with weight w_{ij} and "lateral" synaptic input from each competitive neuron $v_k, k \neq j$, with weight $\tilde{w}_{kj}$. At each cycle of the learning procedure one $\mathbf{s}^l \in S$ is randomly chosen and the input neurons are made fire such that they temporally encode $\mathbf{s}^l$. Each v_j then starts to compute $\sum_i w_{ij} s_i^l$ as described in Section II. If we assume that the input vector and the weight vector for each neuron are normalized, then this weighted sum represents the similarity between the two vectors with respect to the Euclidean distance. Hence the earlier v_j fires, the more similar is its weight vector to the input vector, i.e. the winner among the layer of competitive neurons fires first. Note that the firing time of the winner is not influenced by the firing of the other competitive neurons (see Figure 1).

If the lateral connections are strongly inhibitory, such that the firing of the winner neuron, say v_k, inhibits all other neurons in the competitive layer from firing, one can implement in a straightforward way competitive learning: One simply has to apply the standard competitive learning rule (also known as instar learning rule) to the winner neuron

v_k. This rule is given for v_k by

$$\Delta w_{ik} = \eta(s_i^l - w_{ik}), \qquad i \in \{1, \ldots, m\}$$

where $\mathbf{s}^l$ is the current pattern and η the learning rate.

For realizing the SOM in our model, one has to find a way to implement a given neighbourhood matrix $(m_{kj})_{1 \le k,j \le n}$ on the basis of locally available information, where m_{kj} describes the distance between the kth and jth competitive neuron. We use a monotonously increasing function $m_{kj} \mapsto \tilde{w}_{kj}$ such that the lateral connections $\tilde{w}_{kj}$ among the competitive neurons reflect the structure of the neighbourhood: initially neurons which are topologically close together have strong excitatory lateral connections whereas remote neurons have strong inhibitory connections. This means that the firing of the winner neuron, say v_k, at time t_k drives the firing times of neurons in the neighbourhood of v_k towards t_k, thus increasing the values they encode. The firing of remote neurons is postponed by the lateral inhibition. In the example of Figure 1, the firing of the winner v_1 at t_1 shifts the firing time of a topologically close neuron v_2 towards t_1 and postpones the firing of a topologically remote neuron v_n. We suggest the following learning rule:

$$\Delta w_{ij} = \eta \frac{T_{out} - t_j}{T_{out}} (s_i^l - w_{ij}) \tag{1}$$

where t_j is the firing time of the jth competitive neuron. The learning rule applies only to neurons that have fired before a certain time T_{out} (which has to be chosen sufficiently large). The factor $(T_{out} - t_j)/T_{out}$ realizes the neighbourhood function, which is largest for the winner neuron and decreases for neurons which fire at later times.

During the learning process the lateral weights $\tilde{w}_{kj}$ are decreased, thus reducing the size of the neighbourhood.[1] As in the standard formulation of the SOM, η is slowly decreased during learning.

IV. Simulations

We tested our approach with one- and two-dimensional input patterns. In the one-dimensional case 10 input patterns, uniformly distributed over $[0, 1]$, were presented to a layer of 10 competitive units, which had initially random weights of values around the midpoint of the patterns. The lateral weights for the immediate neighbours were chosen slightly positive, for the second neighbours zero and for all other neurons negative. The goal was the formation of a linear map. Figure 2 shows that initially nearly all neurons react strongly, i.e. fire early on each input, whereas after learning only few neurons, being topologically close, react on a certain input pattern.

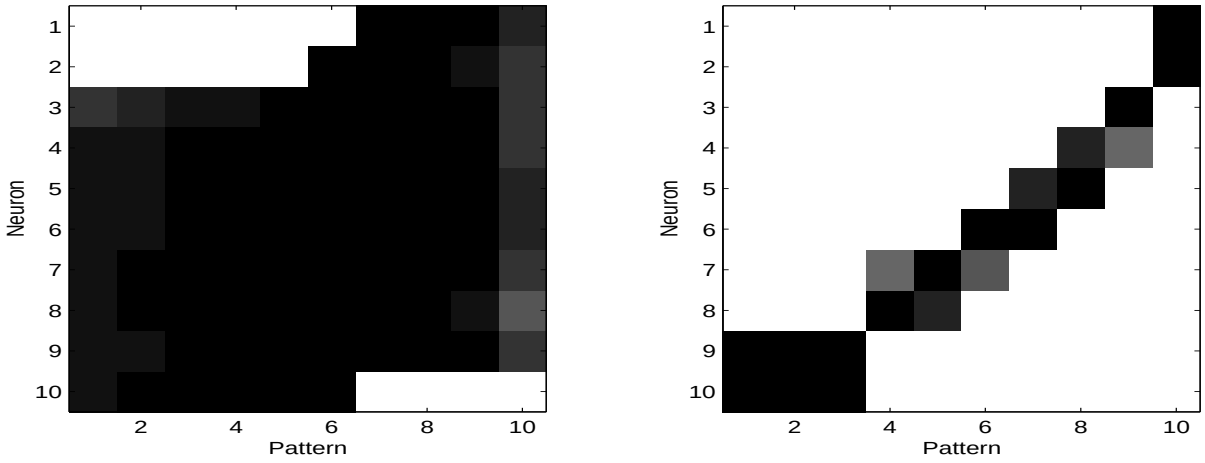

Fig. 2. Learning process for one-dimensional input patterns after 10 iterations (left) and 4000 iterations (right). Each column represents the reaction of the network to a particular input pattern. The colors of the squares indicate the firing times: the darker a square, the earlier the corresponding firing time. The non-firing of a neuron is represented by a white square.

Furthermore, we investigated the behaviour of our construction on one of the standard examples for the SOM, where two-dimensional input patterns are chosen randomly from a square and the competitive units are expected to organize themselves in a topology preserving grid. For the tests we used an array of 5×5 competitive units. We normalized the inputs by adding a third input component, which was chosen such that all input vectors have the same norm. The weights were initialized in the same fashion as in the first experiment. In order to examine the effect of normalization

[1] This requires that the lateral weights change their sign during learning, which is not very realistic in the context of biological networks. However, one can consider instead two connections, one excitatory, which decreases and one inhibitory, which increases during learning.

we performed two series of experiments. In the first one the feedforward weights were normalized after each application of (1), in the second they remained unnormalized.

The topology preserving quality of self-organizing maps is usually assessed by visual inspection. This may be appropriate for a single mapping but it is hardly possible to compare the degree of topology preservation of two different mappings. In our second experiment we therefore used a measure for quantifying the neighbourhood preservation known as "metric multidimensional scaling" (see e.g. [13]). It is based on the objective function

$$E_{\mathrm{MDS}} = \sum_{i=1}^{N} \sum_{j<i} (F(i,j) - G(M(i), M(j)))^2. \tag{2}$$

where N is the number of input patterns and M denotes the mapping of the network, i.e. $M(i)$ is the index of the winner neuron in the competitive layer for input $\mathbf{s}^i$. The matrix F represents the dissimilarity of a pair of input patterns $\mathbf{s}^i, \mathbf{s}^j$ expressed here by the Euclidean distance. The neighbourhood matrix which defines the distances between pairs of competitive units is denoted by G. We used the familiar rectangular grid in two dimension with the Manhattan metric for the neighbourhood relationships in this example. Obviously, an optimal topology preserving mapping minimizes the value of E_{MDS}.

In order to scale the values of E_{MDS} into the interval $[0,1]$ and to make the results for different initializations comparable, we defined the *relative neighbourhood distortion* which is the actual value of E_{MDS} divided by the maximum initial value of E_{MDS}. The results for two typical simulations are shown in Figure 3. Regardless whether the weights are normalized or not – the relative E_{MDS} stabilizes indeed at approximately the same small value.

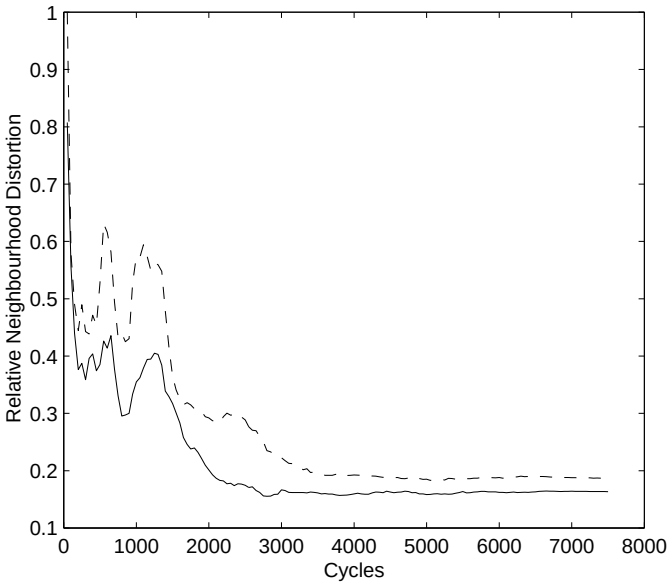

Fig. 3. Development of the relative neighbourhood distortion for the second example described in Section IV, where the solid (dashed) line shows the case with (without) weight normalization after each cycle.

V. Conclusions

To the best of our knowledge, this is the first implementation of Kohonen's learning algorithm for SNNs using temporal coding. This work is a further step towards showing that biological neurons can indeed achieve a topology preserving behaviour using a similar learning procedure like the one suggested by Kohonen. Our approach also may give rise to fast hardware implementations of self-organizing networks in pulse coded VLSI.

As mentioned above we have assumed as in [8] that the neurons are of the leaky integrate-and-fire type, where the initial segment of the postsynaptic potentials rises linearly. Recent simulations [16] have shown that even when using a more detailed neural model which includes nonlinear effects and more realistic shapes (e.g. α-functions) for the postsynaptic potentials, spiking neurons can still compute weighted sums in the way described in Section II. This indicates that the abovementioned simplifying assumptions can be dropped.

References

[1] W. Gerstner and L. van Hemmen, "How to describe neuronal activity: spikes, rates or assemblies?," in *Advances in Neural Information Processing Systems*. 1994, vol. 6, Morgan Kaufmann.

[2] T. Sejnowski, "Time for a new neural code?," *Nature*, vol. 376, pp. 21 – 22, 1995.

[3] F. Rieke, D. Warland, W. Bialek, and R. de Ruyter van Steveninck, *SPIKES: Exploring the Neural Code*, MIT-Press, Cambridge, 1996.

[4] W. Maass, "Networks of spiking neurons: The third generation of neural network modells," *To appear in: Neural Networks*, 1997.

[5] M. A. Arbib, Ed., *The Handbook of Brain Theory and Neural Networks*, MIT Press, Cambridge, 1995.

[6] W. Maass, "Lower bounds for the computational power of networks of spiking neurons," *Neural Computation*, vol. 8, pp. 1–40, 1996.

[7] B. Ruf and M. Schmitt, "Learning temporally encoded patterns in networks of spiking neurons," *Neural Processing Letters*, vol. 5, no. 1, pp. 9–18, 1997.

[8] W. Maass, "Fast sigmoidal networks via spiking neurons," *Neural Computation*, vol. 9, pp. 279–304, 1997.

[9] T. Kohonen, *Self-Organizing Maps*, Springer, Berlin, 1995.

[10] T. Kohonen, "Physiological interpretation of the self-organizing map algorithm," *Neural Networks*, vol. 6, pp. 895 – 905, 1993.

[11] J. Sirosh and R. Miikkulainen, "Topographic receptive fields and patterned lateral interaction in a self-organizing model of the primary visual cortex.," *Neural Computation*, vol. 9, pp. 577 – 594, 1997.

[12] Y. Choe and R. Miikkulainen, "Self-organization and segmentation with laterally connected spiking neurons," Tech. Rep. AI TR 96-251, Department of Computer Science, University of Texas at Austin, 1996.

[13] G. J. Goodhill and T. J. Sejnowski, "A unifying objective function for topographic mappings," *Neural Computation*, vol. 9, pp. 1291 – 1303, 1997.

[14] A. Murray and L. Tarassenko, *Analogue Neural VLSI: A Pulse Stream Approach*, Chapman & Hall, 1994.

[15] B. Ruf and M. Schmitt, "Unsupervised learning in networks of spiking neurons using temporal coding," in *Proc. of the 7th International Conference on Artificial Neural Networks*, 1997, pp. 361 – 366.

[16] B. Ruf, "Computing functions with spiking neurons using temporal coding," in *Biological and Artificial Computation: From Neuroscience to Technology. Proc. of the International Work-Conference on Artificial and Natural Neural Networks*, J. Mira, R. Moreno-Díaz, and J. Cabestany, Eds. 1997, vol. 1240 of *Lecture Notes on Computer Science*, pp. 265 – 272, Springer, Berlin.