Spatial and Temporal Pattern Analysis via Spiking Neurons

Thomas Natschläger, Berthold Ruf
Institute for Theoretical Computer Science
Technische Universität Graz
Klosterwiesgasse 32/2
A-8010 Graz, Austria
e-mail: {tnatschl, bruf}@igi.tu-graz.ac.at

Abstract. Spiking neurons, receiving temporally encoded inputs, can compute radial basis functions (RBFs) by storing the relevant information in their delays. In this paper we show how these delays can be learned using exclusively locally available information (basically the time difference between the pre- and postsynaptic spike). Our approach gives rise to a biologically plausible algorithm for finding clusters in a high dimensional input space with networks of spiking neurons, even if the environment is changing dynamically. Furthermore, we show that our learning mechanism makes it possible that such RBF neurons can perform some kind of feature extraction where they recognize that only certain input coordinates carry relevant information. Finally we demonstrate that this model allows the recognition of temporal sequences even if they are distorted in various ways.

1. Introduction

Radial basis functions (RBFs) have turned out to be among the most powerful artificial neural network types, e.g. in the area of function approximation, pattern classification and data clustering (see [Haykin, 1994] for an overview). There is also growing interest in the relevance of this approach to biological neural networks. The question if biological neurons can realize one of the main advantages of RBF neurons, namely their ability to discover clusters in the input space is not yet resolved. We present a learning algorithm for spiking neurons realizing RBFs which is not based on rate coding but on the timing of single spikes and which is able to find centers of clusters in an unsupervised fashion. There exists substantial evidence that this type of temporal coding is important for biological systems, especially for fast neural information processing [Thorpe and Imbert, 1996]. In temporal coding the relevant information may be represented by the firing times of neurons relative to the stimulus onset [Gawne et al., 1996] or relative to the firing times of other neurons [O'Keefe and Reece, 1993, Thorpe and Imbert, 1996].

In this context, Hopfield presented a model for computing RBFs with spiking neurons [Hopfield, 1995]. The basic idea is that an "RBF neuron" encodes a particular

input spike pattern in the delays available across its synapses. If an input pattern is close to the encoded spike pattern of an RBF neuron (called center of the RBF neuron in the following), the delays even out the differences of the firing times of the input neurons such that the RBF neuron fires. This approach is motivated by many neurobiological findings, e.g. in [O'Keefe and Reece, 1993] evidence has been given that the relative timing of spikes in the rat hippocampus might carry information about the relative location in an environment which is common to the rat. In [Haberly, 1985] it was shown that in the rat piriform cortex a wide range of time delays (up to 20 ms) can occur, which would allow a large interval for the input firing times.

In [Gerstner et al., 1996] a learning mechanism was introduced for a *single* RBF neuron. The proper delays, which encode the center of an RBF neuron, are chosen by using a neuronal learning rule which increases the efficacy of synapses with corresponding proper delays and decreases the efficacy of other synapses. A basic assumption underlying this approach is that there are several paths with different delays between each input and each RBF neuron.

Here we extend this approach by considering not only the firing/non-firing of a neuron but also its firing *time*. We show how on the basis of these ideas networks of such RBF neurons can be constructed and trained to divide the input space into several clusters, where the selection of the proper delays uses exclusively local information (essentially the difference between pre- and postsynaptic firing time). We performed computer simulations with neurons which were basically of the leaky integrate-and-fire type and achieved extremely promising results:

- The RBF neurons converged very reliably to the centers of the clusters even in the presence of noise.
- The RBF neurons were able to reconfigure themselves dynamically, if during the learning process clusters were added or removed. This adaptation to a changing environment seems to be of particular importance for biological systems.
- In the case where each cluster is formed in a certain subspace of the input space we show that with our learning mechanism RBF neurons can find the proper delays for the coordinates of the subspace and "deactivate" the remaining inputs. This can be considered as some kind of feature extraction, where the coordinates of the subspace carry the relevant information.
- Especially in the context of biological neural networks temporal sequences and their analysis seem to be of great importance. It turns out that by employing postsynaptic potentials of variable width such RBF neurons can detect temporal sequences even if those sequences are distorted in various ways.

We would like to emphasize that this paper is not about biology but about possibilities of computing with spiking neurons which are inspired by biology. Spiking neuron networks have turned out to be very powerful (see e.g.[Maass, 1997b]), but there is still not much known about possible learning and higher computational mechanisms

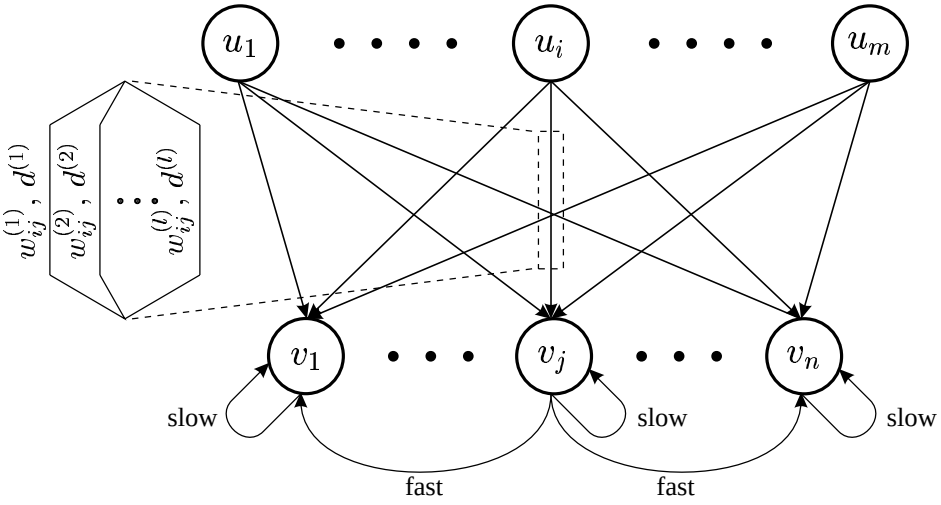

Figure 1. Our basic architecture for an RBF network of spiking neurons. Each connection from an input neuron u_i to an RBF neuron v_j consists actually of a set of l excitatory synapses with different weights $w_{ij}^{(k)}$ and corresponding delays $d^{(k)}$, as indicated by the magnification on the left. Each RBF neuron forms a slow inhibitory connection to itself and fast inhibitory connections to all other RBF neurons.

for such networks. A thorough understanding of such networks, which are rather simplified in comparison to real biological networks, is necessary for constructing corresponding circuits e.g. in pulse-coded VLSI but also for understanding possible mechanisms in biological systems.

Our paper is organized as follows: Section 2 describes the model for computing radial basis functions with spiking neurons. Section 3 outlines an appropriate learning algorithm and presents the results of computer simulations for online clustering with this approach. Section 4 gives an extension for finding clusters in subspaces, Section 5 describes how temporal sequences can be discovered by such RBF neurons. In Section 6 we address the question of stability during learning and in Section 7 we draw some conclusions from our work.

2. A computational model for radial basis functions using spiking neurons

The model for a spiking neuron which we use in the following is basically of the leaky integrate-and-fire type, where the incoming postsynaptic potentials are simply added up. The neuron fires, if its excitatory input is strong enough to push the potential above its threshold. Formal definitions of this model have been presented in [Maass, 1996] and in [Gerstner et al., 1993]. Here we consider a network of such spiking neurons with input neurons $u_1, \ldots, u_m$ and output neurons $v_1, \ldots v_n$, the latter ones will be denoted in the following as RBF neurons (see Figure 1). In the simplest case each input neuron u_i forms one synaptic connection to each RBF neuron v_j with weight w_{ji} and delay d_{ji}, where the delay is given by the difference between the presynaptic firing time and

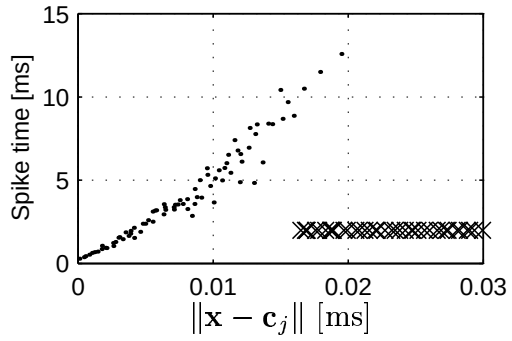

Figure 2. Dependence of the firing time of an RBF neuron v_j on the distance of the input vector $\mathbf{x}$ to the center $\mathbf{c}_j$. For this simulation 120 randomly chosen inputs $\mathbf{x} \in [0, 20\,\text{ms}]^{20}$ are presented to v_j, with equal weights and randomly chosen delays from $[0, 20\,\text{ms}]$. Crosses indicate the case that the RBF neuron has not fired.

the time the postsynaptic potential starts rising. The center of an RBF neuron v_j is given by the vector $\mathbf{c}_j = \langle c_{j1}, \ldots, c_{jm} \rangle$ with $c_{ji} = d_{ji} - \min\{d_{ji} | 1 \leq i \leq m\}$. For our basic construction we consider the simple coding scheme, where each input neuron u_i fires exactly once at time t_i within a certain time interval $[0, T]$, to which we will refer as the *coding interval* in the following. The firing times represent the input vector $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$ with $x_i = \max\{t_i | 1 \leq i \leq m\} - t_i$. No reference spike is needed for this type of competitive coding. The input $\mathbf{x}$ is close to the center $\mathbf{c}_j$ of an RBF neuron v_j if the spikes of the input neurons reach the soma of v_j due to the corresponding delays at similar times, i.e. if $\|\mathbf{x} - \mathbf{c}_j\|$ is small. This is basically the approach presented in [Hopfield, 1995]. He considers the case, where the input vector is close enough to the center of an RBF neuron to make v_j fire. However, in addition the firing time of v_j indicates, how close an input vector $\mathbf{x}$ is to $\mathbf{c}_j$ (see Figure 2). If the distance between $\mathbf{x}$ and v_j is too large, v_j does not fire at all. Let us consider now a set of RBF neurons $\{v_1, \ldots, v_n\}$. If for some input vector $\mathbf{x}$ the difference $\|\mathbf{x} - \mathbf{c}_j\|$ is small enough for various j to make v_j fire then the RBF neuron whose center is closest to $\mathbf{x}$ fires first. Hence a set of such RBF neurons can be used to separate inputs into various clusters.

3. Learning RBFs and Clustering with spiking neurons

If one uses RBF neurons for clustering as described above, then the centers of the clusters can be found by the RBF neurons in an unsupervised way: we assume similar as in [Gerstner et al., 1996] that each u_i forms to each v_j instead of one synapse a set of synapses with varying delays (see Figure 1). Throughout this paper we use the set $D = \{d^{(1)}, \ldots, d^{(l)}\}$ of delays which is the same for every input with $d^{(k)} - d^{(k-1)} > 0$ for $2 \leq k \leq l$. The length T of the coding interval where the input neurons are allowed to fire is assumed to be smaller than $d^{(l)} - d^{(1)}$. We say the delay of length $d^{(k)}$ for the connection from u_i to v_j is *active* if its corresponding weight $w_{ij}^{(k)}$ is significantly greater than zero.

The goal of our learning algorithm can be formulated as follows: given a cluster C of input vectors $\mathbf{x}$ in the input space, an RBF neuron v_j should activate for each input coordinate one delay, such that the resulting center $\mathbf{c}_j$ minimizes $\sum_{\mathbf{x} \in C} \|\mathbf{x} - \mathbf{c}_j\|$. In addition we also want to allow that several neighboring delays become activated

because the center of a cluster may not be representable due to a finite number of available delays. As it will turn out (see Section 3.1), the activation of a larger number of neighboring delays may also reflect a larger variance of the input cluster.

For such a constellation of several active neighboring delays we define the center $\mathbf{c}_j$ of an RBF neuron v_j as follows: we compute for every input coordinate i the mean delay $\tilde{d}_{ji} = \sum_{k=1}^{l} w_{ij}^{(k)} \cdot d^{(k)} / \sum_{k=1}^{l} w_{ij}^{(k)}$. The center of v_j is then the vector $\mathbf{c}_j = \langle c_{j1}, \ldots, c_{jm} \rangle$ with $c_{ji} = \tilde{d}_{ji} - \min\{\tilde{d}_{ji} | 1 \leq i \leq m\}$.

Each RBF neuron should converge to the center of some cluster, which can be achieved in the following way: initially all weights are set to random values such that no RBF neuron can fire until it has received at least one spike from every $u_i, 1 \leq i \leq m$. As soon as an RBF neuron fires, the spike is propagated backwards to its synapses, where a weight is changed according to some "learning-function" $L(\Delta t)$, with $\Delta t = t_{pre} - t_{post}$ denoting the difference between the arrival times t_{pre} and t_{post} of the pre- and postsynaptic spike at the synapse (see Figure 3a). This is basically the learning mechanism suggested in [Markram and Tsodyks, 1996]. In the following, we will use the learning function as shown in Figure 3b. $L(\Delta t)$ is chosen here such that the weights of those synapses, which received shortly before the postsynaptic firing a presynaptic spike, are increased (i.e. the peak of L is shifted by 2 ms from 0), whereas synapses, which received a presynaptic spike much earlier or later are decreased.

More precisely, the weight $w_{ij}^{(k)}$ of the synapse between input neuron u_i and RBF neuron v_j with delay $d^{(k)}$ is updated during the whole learning process due to the following learning rule:

$$\Delta w_{ij}^{(k)} = \eta L\left((t_i + d^{(k)}) - (t_j + d^{back})\right) \tag{1}$$

for every pair of input firing time t_i and firing time t_j of the RBF neuron (see Figure 3b for an example). $\eta > 0$ denotes the learning rate. The delay d^{back} describes the time the postsynaptic spike needs to propagate backwards to the synapses. For all these synapses d^{back} has the same value. Furthermore, we assume that the weights saturate at 0 respectively at a certain $w_{max} > 0$ (see Section 6 for details).

By adding fast lateral inhibition among the RBF neurons, one can implement a winner-take-all mechanism where only the RBF neuron fires which is closest to the current input vector. Hence, only the weights of the winning RBF neuron are modified, such that its center moves towards the current input vector. This is a similar approach as in competitive learning and it enables RBF neurons to learn clusters in the input space.

Since we are interested in the firing *time* of an RBF neuron, we have to make sure that the sum of the weights of all incoming synapses remains approximately constant in order to guarantee that the firing time does not shift due to a simple weight scaling but only because of peaks in the weight distribution. Such peaks can be seen in Figure 4, where there are only a few neighboring delays with large corresponding weights after learning. To achieve such a constant sum, one would usually have to require some non-local interaction. However, if one chooses a proper learning-function (where

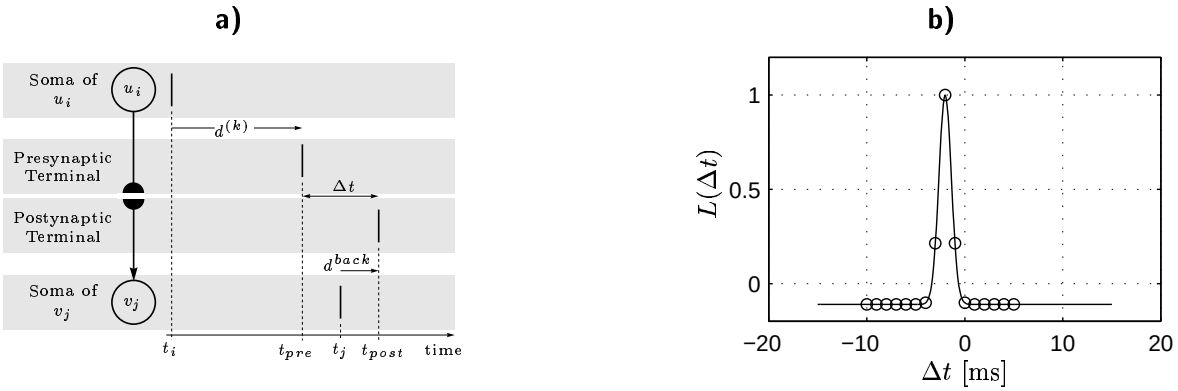

Figure 3. a) A spike emitted at u_i (v_j) at time t_i (t_j) reaches the synapse between u_i and v_j at time $t_{pre} = t_i + d^{(k)}$ ($t_{post} = t_j + d^{back}$). b) Learning function $L(\Delta t)$, describing how the weight change is influenced by the difference $\Delta t = t_{pre} - t_{post}$ between pre- and postsynaptic firing times at the synapse. The circles denote the weight changes for various delays occurring when u_i fires 10 ms before v_j. The leftmost (rightmost) circle corresponds to a delay of 1 ms (16 ms). We chose here $d^{back} = 1$ ms and $L(\Delta t) = (1 - b) \cdot \exp\left(-(\Delta t - c)^2/\beta^2\right) + b$, with $b = -0.11$, $\beta = 1.11$ ms and $c = -2$ ms.

$\int_{-\infty}^{\infty} L(t)dt \approx 0$), this effect can be also achieved using only locally available information. Figure 5b shows that the peaks of the membrane potential caused by one input vector indeed stabilize during the learning process at a certain level.

A general problem with clustering, which also occurs here, is that it is very likely that some RBF neurons specialize on sets of clusters (thus computing their mean) and other RBF neurons are not used at all. We solved this problem by adding slow self-inhibition to every RBF neuron, decreasing the probability that an RBF neuron fires twice within a few iterations of the learning rule. A similar "conscience" mechanism is often used in competitive learning [Grossberg, 1976]. However, with a surplus of RBF neurons and a proper choice for the learning function and the initialization of the weights, such that each RBF neuron needs at least one spike from every input neuron, it is also possible to achieve good results without self-inhibition.

3.1. Simulation Results

We performed computer simulations in order to investigate how these RBF neurons behave on a high-dimensional input space. In the following we use RBF neurons with 40 inputs. The length of the coding-interval was 10 ms, the delay-interval 15 ms such that the available delays were 1 ms, 2 ms, ..., 16 ms. Every 40 ms a new learning cycle was started by presenting a new input vector $\mathbf{x}$ to the network. The input vectors were randomly chosen from an input space which has p clusters C_μ for $1 \leq \mu \leq p$. For each component of an input vector $\mathbf{x} \in C_\mu$ the standard deviation was chosen between 2 ms and 4 ms. There was no overlap between the clusters. We set $L(\Delta t) = 0$ for $|\Delta t| > 15$ ms to make sure that weights are changed only due to the current input vector.

First we considered the case $n = p$, i.e. there were as many RBF neurons as clusters.

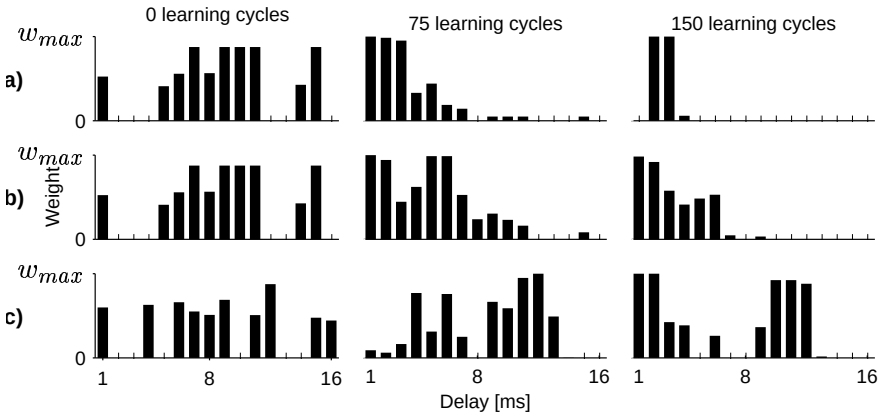

Figure 4. Development of the weights for a set of delays $D = \{1\,\text{ms}, \ldots, 16\,\text{ms}\}$ for one input neuron and one RBF neuron. a) Input patterns chosen from a cluster C_μ with small standard deviation. b) Same as (a), but with large standard deviation. c) Input patterns chosen from two different clusters.

Figure 5. a) Membrane potential of an RBF neuron for one input vector before (dashed line) and after (solid line) learning, causing a shift of the firing time from 19 ms to 14 ms, denoted by the horizontal bar at the height of the threshold. b) Development of the peaks of the membrane potential for an RBF neuron during the learning process. The jitter results from the varying input vectors chosen from one cluster. For the threshold we chose for both figures $\Theta = 1$.

After about 50 examples from each cluster each RBF neuron had converged to one cluster. As Figure 4 indicates, the variance of the clusters is reflected in the distribution of the weights of the delays. A small standard deviation of a cluster results in a sharp peak, whereas larger values cause several similar delays to have strong weights nearby the saturation value w_{max}, such that the RBF neuron will respond stronger, i.e. fires earlier, for a wider range of input vectors.

Furthermore, we could observe that an RBF neuron fires considerably earlier after learning a cluster, when a pattern from that cluster is presented. This results from the fact that the membrane potential increases more steeply after learning (see Figure 5a).

If one considers the definition of a center $\mathbf{c}_j$ given in section 2, there can be several values of $\tilde{d}_{ji}$ which yield the same center $\mathbf{c}_j$. Observe that our model implies

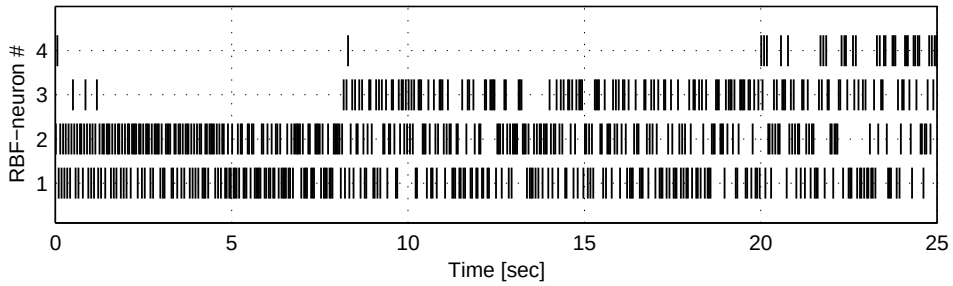

Figure 6. Spike trains for four RBF neurons. At the beginning, two clusters were presented, after 8 s a third and after 20 s a fourth cluster was added.

that $\min\{\tilde{d}_{ji}|1 \leq i \leq m\}$ almost always is nearly equal to the smallest available delay $d^{(1)} = 1\,\text{ms}$. This is not obvious since we use no reference spike to define our input interval. It has the advantage that the firing times of several RBF neurons are comparable on an absolute time scale as suggested in [Thorpe and Imbert, 1996].

In the case of more clusters than RBF neurons, some RBF neurons will converge to the mean of a subset of the clusters. This may result in one wide peak or several smaller peaks in the distribution of the strengths of the delays, which is the best one can expect in this case (cf. Figure 4c). The latter case occurs if an RBF neuron has specialized on several clusters, i.e. it has more than one center.

If there are initially more RBF neurons than clusters, it is obvious that several RBF neurons can specialize on the same cluster or that, depending on the initial conditions, some RBF neurons do not learn anything. Furthermore, we observed that a dynamic reconfiguration of the RBF neurons in a changing environment is possible. We assume that some of the RBF neurons have already learned to represent some clusters. If one adds during the learning process another cluster, one of the "free" RBF neurons will quickly specialize on this cluster (see Figure 6). If one continues adding clusters, such that the number of clusters exceeds the number of RBF neurons, a dynamic reconfiguration occurs in the sense, that an RBF neuron switches from an old to a new cluster or learns the mean of a set of clusters including new ones.

We also investigated the influence of noise by making each input neuron fire in addition at random time points. This means that each input neuron generates a Poisson spike train (of low frequency) and fires in addition corresponding to the sequence of input patterns. Figure 7 shows that despite of these underlying Poisson spike trains the RBF neurons are still able to detect the patterns in the noisy environment and assign them to the proper cluster. We were able to show that such noise may be even present during the learning phase without deteriorating the learning quality and speed considerably.

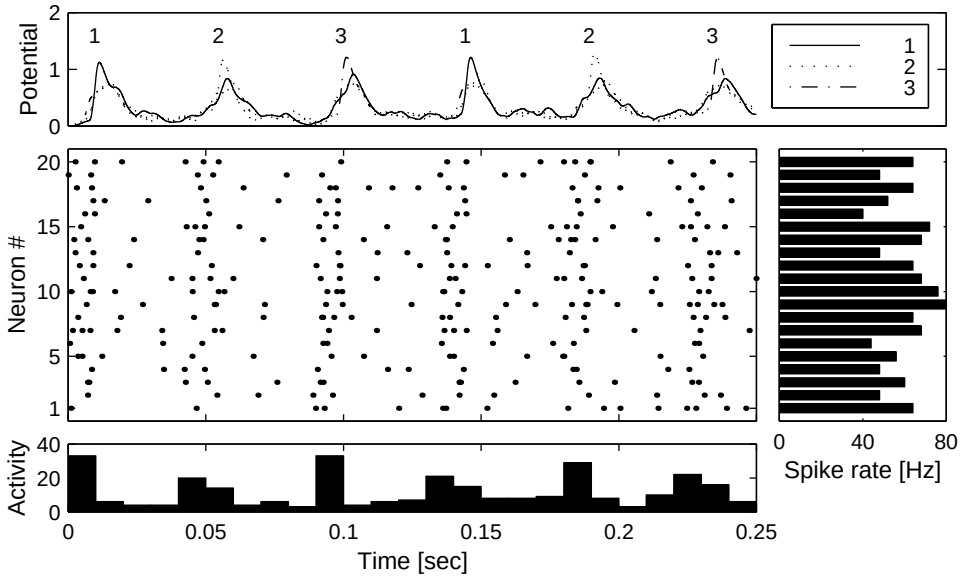

Figure 7. Behavior of 3 RBF neurons on spike trains including Poisson distributed spikes of 25 spikes/sec after 270 learning cycles. The vectors presented to the network (every 50 ms) are chosen cyclically from 3 clusters. The lower box shows the number of spikes within time bins of size 10 ms. Although the input neurons fire all approximately with the same frequency (right box) the RBF neurons can distinguish the pattern in the noise (upper box). The threshold was set to 1.0.

4. Finding Clusters in Subspaces

If one interprets the input coordinates as a representation of certain features, then the clusters represent typical constellations of frequently occurring combinations of input values. A cluster may then be formed out of some features, whereas other features are irrelevant. In our context, this means that only certain input neurons describe a cluster whereas the remaining input neurons simply produce noise. The task of the RBF neuron is to extract the "relevant" inputs for every cluster.

More precisely, we consider an input space of dimension n with basis $E = \{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$, where $\mathbf{e}_i$ is the ith unit vector. Let us assume that for an input $\mathbf{x} = \sum_{i=1}^{n} \lambda_i \mathbf{e}_i$ the input neuron u_i receives the value $\lambda_i \in [0, T]$, with $[0, T]$ being the coding interval. We consider only subspaces where the basis E_{sub} is a subset of E. Samples from a cluster C in such a subspace are presented to the network as

$$\mathbf{x} = \sum_{\mathbf{e}_i \in E_{sub}} \lambda_i \mathbf{e}_i + \sum_{\mathbf{e}_i \in E \setminus E_{sub}} n_i \mathbf{e}_i$$

where the n_i are random variables, uniformly distributed over $[0, T]$ (see Figure 8 for an example). Assume that RBF neuron v_j is to converge to the center $\mathbf{c}$ of C. The goal is that besides finding the proper delays for the input neurons receiving the inputs for E_{sub}, the remaining "noisy" coordinates should have no impact on v_j, i.e. the corresponding weights should become zero.

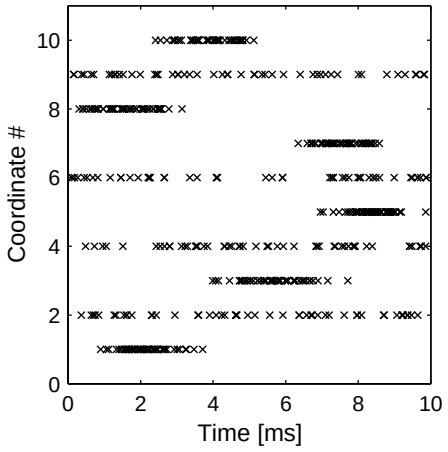

Figure 8. Distribution of 100 input vectors for a cluster of dimension 5 in the subspace $E_{sub} = \{\mathbf{e}_1, \mathbf{e}_3, \mathbf{e}_5, \mathbf{e}_7, \mathbf{e}_8, \mathbf{e}_{10}\}$. The coding interval is of length 10 ms.

Our construction described in the previous section can handle this situation if one slightly changes the learning function, such that larger values of $|\Delta t|$ cause a larger weight decay. This results on the long run in a weight decay for all delays of these noisy coordinates and in sharp peaks of the weight distribution for the other inputs u_i with $\mathbf{e}_i \in E_{sub}$. Figure 9 shows an example of a delay distribution after a successful learning process. It also indicates that RBF neurons which have finally not specialized on some cluster may have quite similar weight distributions as "successful" RBF neurons. This results from a rather long period of competition.

5. Temporal Sequence Recognition

Since the input patterns are temporally encoded in our approach, it is quite natural to use it for the recognition of regularities in temporal sequences of input stimuli. This is of particular importance in biological systems (e.g. in audition and vision) but also for related tasks in hardware implementations. We will show that it is possible to recognize sequences which are distorted in time and form.

Let us consider a spatio-temporal input firing pattern of the form $P = \langle (i_1, t_1), \ldots, (i_N, t_N) \rangle$ with $t_1 \leq t_2 \leq \ldots \leq t_N$, where (i_j, t_j) indicates that at time t_j input neuron u_{i_j} generated a spike.‡ In contrast to our previous assumption an input neuron may fire here several times, i.e. we allow $i_j = i_k$ for $j \neq k$. The detection of such patterns corresponds to the clustering problem described above, except that we allow here that an input neuron may fire more than once within one sequence. In section 3.1 we generated the clusters by adding Gaussian noise to the centers of the clusters. Since we are now dealing with temporal sequences, we consider two types of noise, namely distortion of the sequences in form and in time: the form of a sequence may be distorted if an input spike at a certain time is produced by an erroneous input. A time warp can occur by stretching (squeezing) the sequence, i.e. the firing

‡ We assume that P describes a valid spike train, where a neuron does not fire within its absolute refractory period.

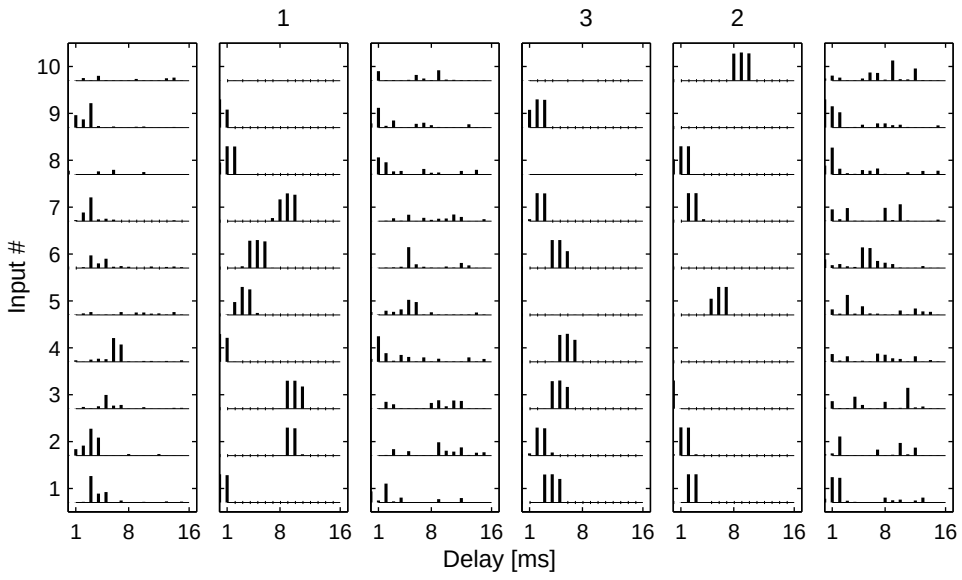

Figure 9. Weight distribution for a set of delays $D = \{1\,ms, \ldots, 16\,ms\}$ after a successful learning process of 660 learning cycles with three clusters. Each column represents one RBF neuron with 40 inputs, of which 10 are shown. A number above a column indicates that the corresponding RBF neuron has specialized on the cluster with that number. Cluster 1 had 5, cluster 2 had 10 and cluster 3 had 15 noisy coordinates. RBF neuron 1 has a similar weigh distribution as RBF neuron 4, since it has competed for a rather long period with neuron 4 for cluster 3. The parameters for $L(\Delta t)$ (see Figure 3b) were chosen as $c = -1.5\,\text{ms}$, $\beta = 1.67\,\text{ms}$ and $b = -0.2$.

times are scaled such that the firings take place in a larger (smaller) time interval $K \cdot (t_N - t_1)$ for $K > 1$ ($K < 1$). Furthermore, it may be possible that "wrong" spikes appear in the sequence postponing the subsequent spikes of the sequence, i.e. a spike at time t of neuron u_{i_k} with $t_j \leq t \leq t_{j+1}$ would change P to the pattern $\tilde{P} = \langle (i_1, t_1), \ldots, (i_j, t_j), (i_k, t), (i_{j+1}, (t - t_j) + t_{j+1}), \ldots, (i_N, (t - t_j) + t_N) \rangle$. This type of noise is of particular importance for temporal sequences (e.g. sound waves like speech), if the "correct" sequence is interrupted by some noise and then continued.

An RBF neuron v, which has been tuned for such a P, can still fire for an input pattern $\tilde{P}$, if the EPSPs caused by the input spikes before t have a longer impact on the potential of v. This can be achieved, if the EPSPs of these earlier spikes last longer (see Figure 10), such that a shift of the later EPSPs of the amount $t - t_j$, caused by the input spikes after t, still suffices to make v fire. Generally we assume that the earlier an input spike occurs the wider is the resulting EPSP.

This approach is similar to an idea presented in [Tank and Hopfield, 1987], where it is shown how an artificial neural network can recognize such sequences using certain "delay filters". We were able to realize this concentration in time very naturally by spiking neurons employing EPSPs of variable width which roughly correspond to the delay filters.

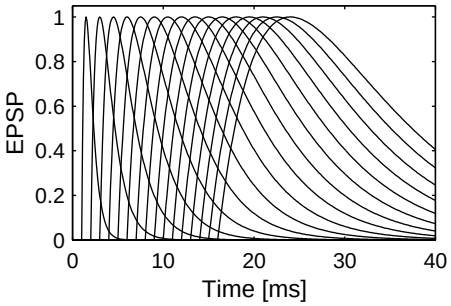

Figure 10. EPSPs used for our simulations described in Section 5. The EPSPs shown here were α-functions, given by $\varepsilon(t) = t/t_p \exp(1 - t/t_p)$ with $t_p \in \{0.5\,ms, 1.0\,ms, \ldots, 8\,ms\}$. The delays (i.e. the time until the onset of the EPSP) were chosen $2 \cdot t_p$ such that a wider EPSP corresponds to a larger delay.

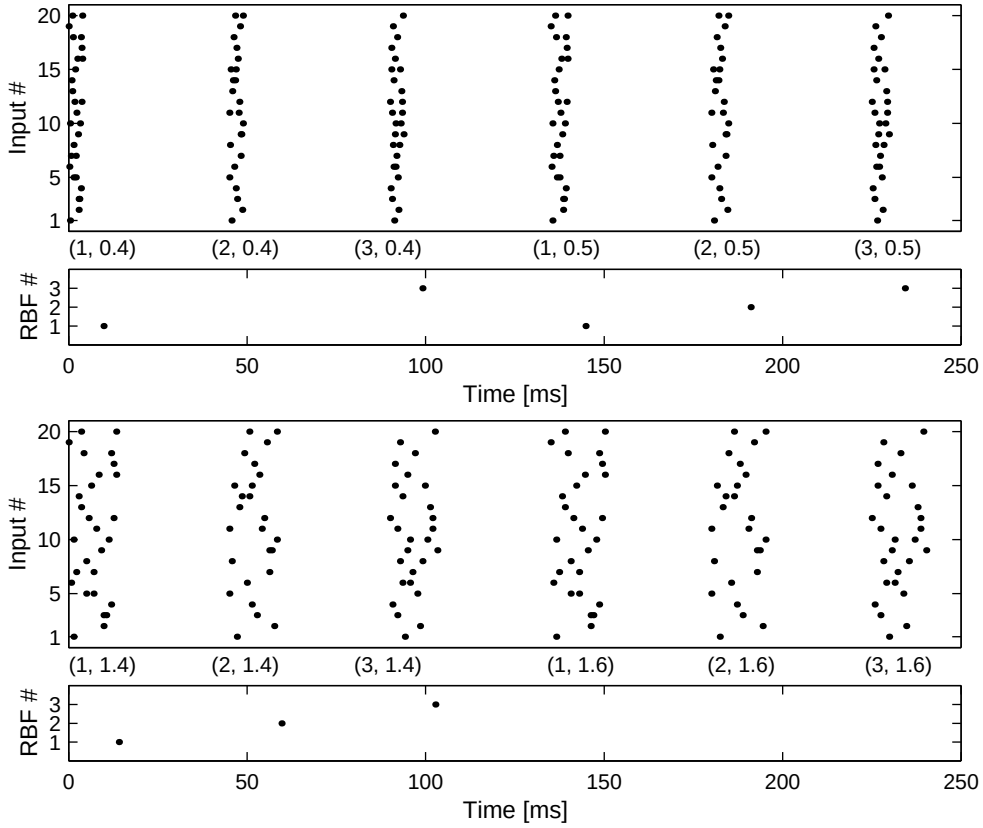

Figure 11. Input- and output spike trains of an RBF network with 20 inputs and 3 RBF neurons. The network was trained on three clusters. After 270 learning cycles RBF neuron v_j had converged to the center of cluster j. A tuple (j, K) indicates that an input vector from the jth cluster with a stretch/squeeze factor of K was presented.

Our construction is able to deal with all these above-mentioned types of noise. Figure 11 shows as an example that a stretching or squeezing of an input sequence up to a factor of $K = 1.4$ respectively $K = 0.5$ still allows a proper detection of each sequence. If a sequence is too strongly squeezed or stretched, the RBF unit, which was tuned to the corresponding sequence, does not fire at all, i.e. the network does either respond correctly or does not respond at all.

6. Stability

It is not obvious that the RBF neurons stabilize at a certain delay constellation during a long learning process. Indeed, it could be the case that the center of an RBF neuron becomes smaller during the learning process such that for certain inputs only the smallest available delays are active (i.e. have weights greater zero) whereas for other inputs the active delays vanish (see Figure 12). This undesirable behavior occurs if some weights become too large during the learning process. Large weights make the neuron fire earlier, such that the learning rule (1) increases the weights corresponding to shorter delays. Our learning rule tends to increase weights until they have reached the saturation value w_{max} (i.e. the largest possible weight). Hence this shift of delays can be avoided if w_{max} is not too large. The opposite effect, where larger delays are activated, may occur if w_{max} is too small. Thus the choice of w_{max} is crucial for the stability of the learning process.§

To overcome this problem one can use a continuous saturation function $S(w)$ such that a weight which is close to saturation is less strongly modified than a weight being more in the middle of $[0, w_{max}]$ (see Figure 12c for an example). Hence, such a saturation function supports the stabilization of possible peaks in the weight distribution. One may use the following modification of the learning rule (1):

$$w_{ji}^{(k),new} = w_{ji}^{(k),old} + \eta \cdot S(w_{ji}^{(k),old}) \cdot L((t_i + d^{(k)}) - (t_j + d^{back})) \qquad (2)$$

If the weights are outside of the interval $[0, w_{max}]$ after applying this rule, an additional weight clipping is made, such that a weight can never exceed its boundaries. In order to allow dynamic reconfigurations (like adding/removing clusters during the learning process), one has to assume that $S(0) > 0$ and $S(w_{max}) > 0$.

A reasonable choice of w_{max} is important especially in the context of finding clusters in subspaces (Section 4): One basic assumption in our construction is that one spike of each input neuron within the coding interval suffices to make an RBF neuron fire. If there are some noisy coordinates, the corresponding input neurons have no impact on the RBF neuron after a successful learning process. Hence the weights for the remaining input neurons must be large enough to make the RBF neuron fire. The straightforward solution to this problem, namely to have various w_{max} for different RBF neurons depending on the number of noisy coordinates is not very desirable in the framework of unsupervised learning since one must put some knowledge about the training set into the parameters of the network. In our simulations we were able to show that it suffices to use a saturation function as shown in Figure 12c and one *global* value for w_{max}, which is large enough that an RBF neuron with many noisy coordinates can still fire but also can stabilize at a certain delay constellation.

§ In our simulations in Section 3.1 we used $w_{max} = \Theta/(m \cdot h)$ where m is the number of inputs, Θ the threshold and h is the length of the interval $[\Delta t_1, \Delta t_2]$ for which the learning function $L(\Delta t)$ is greater zero.

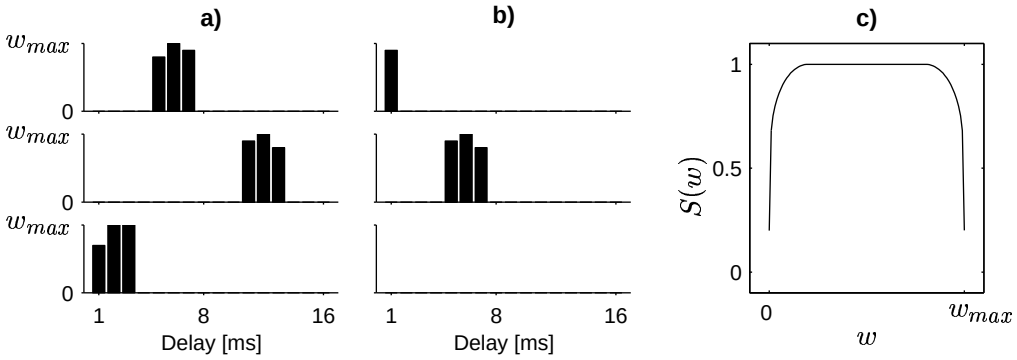

Figure 12. a) and b) Shift of the weight distribution for a set of delays $D = \{1\,\text{ms}, \ldots, 16\,\text{ms}\}$. The maximum weight w_{max} was chosen too large, such that an initial learning success after 120 learning cycles (a) shifted towards smaller delays, where for certain inputs even no delay had a weight greater zero (b). c) One possible choice for a saturation function.

7. Conclusions

We have presented a method for clustering temporally encoded input patterns with networks of spiking neurons. It has turned out that this method is very noise-robust and can be extended in various ways, such as for feature extraction and recognizing temporal sequences.

In biological neural systems there exist at least two different types of inhibition: fast GABA_A and slow GABA_B mediated inhibitory synapses (see [Segev, 1995] for an overview of time constants). Our model combines the two in a new way: we used fast lateral inhibition to implement the winner-take-all mechanism and slow self-inhibition to allow a competition among all RBF neurons, such that no RBF neuron dominates the competition over a longer period of time.

In RBF layers of artificial neurons one frequently considers an additional linear gate behind the RBF layer, which can be used for classification and function approximation tasks. If the inputs are temporally encoded as described in section 2, spiking neurons can compute such linear gates in a very simple and straightforward way as shown in [Maass, 1997a].This idea can be integrated into our construction as follows: if one omits the fast lateral inhibition among the RBF neurons, the firing time of each RBF neuron relative to the others expresses its similarity to the input vector. A spiking neuron receiving input from all these RBF neurons can then compute the linear gate.

Finally we want to mention that our approach also seems to be of interest in the context of pulse-stream VLSI. For VLSI implementations of spiking neurons the interspike interval can be in the microsecond range. Our learning rule is restricted to local information; if it is applied after each pulse, very fast and parallel learning is possible.

Acknowledgments

We would like to thank Oswin Aichholzer and Peter Auer for helpful discussions. Peter Auer also pointed us to the problem of finding clusters in subspaces. T.N. acknowledges support by the "Fonds zur Förderung der wissenschaftlichen Forschung (FWF), Austrian Science Fund", project number P12153.

References

[Gawne et al., 1996] Gawne, T. J., T.Kjaer, and Richmond, B. (1996). Latency: Another potential code for feature binding in striate cortex. *Journal of Neuroscience*, 76(2):1356 – 1360.

[Gerstner et al., 1996] Gerstner, W., Kempter, R., van Hemmen, L., and Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383:76–78.

[Gerstner et al., 1993] Gerstner, W., Ritz, R., and van Hemmen, J. L. (1993). Why spikes? Hebbian learning and retrieval of time-resolved excitation patterns. *Biological Cybernetics*, 69:503–515.

[Grossberg, 1976] Grossberg, S. (1976). Adaptive pattern classification and universal recording: Ii. feedback, expectation, olfaction, illusions. *Biological Cybernetics*, 23:187–202.

[Haberly, 1985] Haberly, L. (1985). Neuronal circuitry in olfactory cortex: anatomy and functional implications. *Chemical Senses*, 10(2):219–238.

[Haykin, 1994] Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing, New York.

[Hopfield, 1995] Hopfield, J. J. (1995). Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376:33–36.

[Maass, 1996] Maass, W. (1996). Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8:1–40.

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

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

[Markram and Tsodyks, 1996] Markram, H. and Tsodyks, M. (1996). Redistribution of synaptic efficacy between neocortical pyramidal neurons. *Nature*, 382:807–810.

[O'Keefe and Reece, 1993] O'Keefe, J. and Reece, M. L. (1993). Phase relationship between hippocampal place units and the EEG theta rhythm. *Hippocampus*, 3(3):3317–30.

[Segev, 1995] Segev, I. (1995). Dendritic processing. In Arbib, M. A., editor, *Handbook of Brain Theory and Neural Networks*, pages 282–289. MIT Press, Cambridge.

[Tank and Hopfield, 1987] Tank, D. W. and Hopfield, J. J. (1987). Neural computation by concentrating information in time. *Proc. Natl. Acad. Sci. USA*, 84:1896–1900.

[Thorpe and Imbert, 1996] Thorpe, S. and Imbert, M. (1996). Rapid visual processing using spike synchrony. In *Advances in neural processing systems*, volume 9, pages 901–907. MIT Press, Cambridge.