Unsupervised Clustering with Spiking Neurons by Sparse Temporal Coding and Multi-Layer RBF Networks

Sander M. Bohte[1] Han A. La Poutré[1] Joost N. Kok[1,2]
S.M.Bohte@cwi.nl Han.La.Poutre@cwi.nl joost@liacs.nl
[1] *CWI*
P.O.Box 94079, 1090 GB Amsterdam, The Netherlands
[2] *LIACS, Leiden University*
P.O. Box 9512, 2300 RA Leiden, The Netherlands

ABSTRACT
We demonstrate that spiking neural networks encoding information in spike times are capable of computing and learning clusters from realistic data. We show how a spiking neural network based on spike-time coding and Hebbian learning can successfully perform unsupervised clustering on real-world data, and we demonstrate how temporal synchrony in a multi-layer network induces hierarchical clustering. We develop a temporal encoding of continuously valued data to obtain adjustable clustering capacity and precision with an efficient use of neurons: input variables are encoded in a population code by neurons with graded and overlapping sensitivity profiles. We also discuss methods for enhancing scale-sensitivity of the network and show how induced synchronization of neurons within early RBF layers allows for the subsequent detection of complex clusters.

2000 Mathematics Subject Classification: 82C32, 68T05, 68T10, 68T30, 92B20.
1998 ACM Computing Classification System: C.1.3, F.1.1, I.2.6, I.5.1.
Keywords and Phrases: Spiking neurons, unsupervised learning, high-dimensional clustering, complex clusters, Hebbian-learning, synchronous firing, sparse coding, temporal coding, coarse coding.
Note: Work carried out under theme SEN4 "Evolutionary Systems and Applied Algorithmics". This paper has been submitted for publication, a short version has been presented at the International Joint Conference on Neural Networks 2000 (IJCNN'2000) in Como, Italy.

1. Introduction

It is well known that cortical neurons produce all-or-none action potentials or spikes, but the significance of the timing of these pulses has only recently been recognized as a means of neuronal information coding. As the biological evidence has been mounting, e.g. [1], it has been shown theoretically that temporal coding with single spike times allows for powerful neuronal information processing [2]. Furthermore, it has been argued that coordinated spike-timing could be instrumental in solving dynamic combinatorial problems [3]. Since time-coding utilizes only a single spike to transfer information, as apposed to hundreds in firing-rate coding, the paradigm could also potentially be beneficial for efficient pulse-stream VLSI implementations.

These considerations have generated considerable interest in time-based artificial neural networks, e.g. [4; 5; 6]. In particular, Hopfield [7] presents a model of spiking neurons for discovering clusters in an input space akin to Radial Basis Functions. Extending on Hopfield's idea, Natschläger & Ruf [5] propose a learning algorithm that performs unsupervised clustering in spiking neural networks using spike-times as input. This model encodes the input patterns in the delays across its synapses and is shown to reliably find centers of high-dimensional clusters, but, as we argue in detail in section 2, is limited in both cluster capacity as well as precision.

We present methods to enhance the precision, capacity and clustering capability of a network of spiking neurons akin to [5] in a flexible and scalable manner, thus overcoming limitations associated

with the network architecture. Inspired by the local receptive fields of biological neurons, we encode continuous input variables by a population code obtained by neurons with graded and overlapping sensitivity profiles. In addition, each input dimension of a high dimensional dataset is encoded separately, avoiding an exponential increase in the number of input neurons with increasing dimensionality of the input data. With such encoding, we show that the spiking neural network is able to correctly cluster a number of datasets at low expense in terms of neurons while enhancing cluster capacity and precision. The proposed encoding allows for the reliable detection of clusters over a considerable and flexible range of spatial scales, a feature that is especially desirable for unsupervised classification tasks as scale-information is a-priori unknown.

By extending the network to multiple layers, we show how the temporal aspect of spiking neurons can be further exploited to enable the correct classification of non-globular or interlocking clusters. In a multi-layer RBF network, it is demonstrated that the neurons in the first layer center on components of extended clusters. When all neurons in the first RBF layer are allowed to fire, the (near) synchrony of neurons coding for nearby components of the same cluster is then distinguishable by a subsequent RBF layer, resulting in a form of hierarchical clustering with decreasing granularity. Building on this idea, we show how the addition of lateral excitatory connections with a SOM-like learning rule enables the network to correctly separate complex clusters by synchronizing the neurons coding for parts of the same cluster. Adding lateral connections thus maintains the low neuron count achieved by coarse coding, while increasing the complexity of classifiable clusters.

Summarizing, we show that temporal spike-time coding is a viable means for unsupervised neural computation in a network of spiking neurons, as the network is capable of clustering realistic and high-dimensional data. Adjustable precision and cluster capacity is achieved by employing a 1-dimensional array of graded overlapping receptive fields for the encoding of each input variable. By introducing a multi-layer extension of the architecture we also show that a spiking neural network can cluster complex, non-Gaussian clusters. Combined with our work on supervised learning in spiking neural networks ([8]), these results show that single spike-time coding is a viable means for neural information processing on real-world data.

The paper is organized as follows: we describe the spiking neural network and limitations in section 2. In section 3 we introduce a means of encoding input-data such to overcome these limitations, and clustering examples using this encoding are given in section 4. In section 5 we show how the architecture can be extended to a multi-layer RBF network capable of hierarchical clustering, and in section 6 we show how the addition of lateral connections enables the network to classify more complex clusters via synchronization of neurons within an RBF layer. The conclusions are given in section 7.

2. Networks of delayed spiking neurons

In this section, we describe the spiking neural network as introduced in [5] and the results and open questions associated with this type of network.

The network architecture consists of a fully connected feedforward network of spiking neurons with connections implemented as multiple delayed synaptic terminals (figure 1A). A neuron j in the network generates a spike when the internal neuron state variable x_j, identified with "membrane potential", crosses a threshold ϑ. This neuron j, connected to a set of immediate predecessors ("pre-synaptic neurons") Γ_j, receives a set of spikes with firing times t_i, $i \in \Gamma_j$. The internal state variable $x_j(t)$ is determined by the time-dynamics of the impact of impinging spikes on neuron j. As a practical model, we use the Spike Response Model (SRM) introduced by Gerstner [9], where the time-varying impact of a spike is described by a spike-response function. Depending on the choice of suitable spike-response functions one can adapt the SRM to reflect the dynamics of a large variety of different spiking neurons. In the SRM description, the internal state variable $x_j(t)$ is simply the sum of spike-response functions

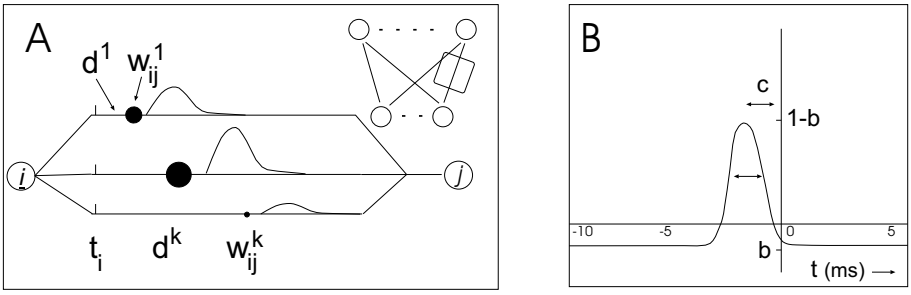

Figure 1: (a) Network connectivity and a single connection composed of multiple delayed synapses. (b) Graph of the learning function $L(\Delta t)$. Δt denotes the time-difference between the onset of a PSP at a synapse and the time of the spike generated in the target neuron.

$\varepsilon(t, t_i)$ weighted by the synaptic efficacy w_{ij}:

$$x_j(t) = \sum_{i \in \Gamma_j} w_{ij} \varepsilon(t - t_i). \tag{2.1}$$

In the network as introduced in [5], an individual connection consists of a fixed number of m synaptic terminals, where each terminal serves as a sub-connection that is associated with a different delay and weight (figure 1a). The delay d^k of a synaptic terminal k is defined by the difference between the firing time of the pre-synaptic neuron, and the time the post-synaptic potential starts rising. Effectively, the input to a neuron j then becomes:

$$x_j(t) = \sum_{i \in \Gamma_j} \sum_{k=1}^{m} w_{ij}^k \varepsilon(t - t_i - d^k). \tag{2.2}$$

Input patterns can be encoded in the synaptic weights by local Hebbian delay-learning where, after learning, the firing time of an output neuron reflects the distance of the evaluated pattern to its learned input pattern thus realizing a kind of RBF neurons [5]. For unsupervised learning, the Winner-Take-All learning rule modifies the weights between the source neurons and the neuron first to fire in the target layer using a time-variant of Hebbian learning: If the start of the PSP at a synapse slightly precedes a spike in the target neuron, the weight of this synapse is increased, as it exerted significant influence on the spike-time via a relatively large contribution to the membrane potential. Earlier and later synapses are decreased in weight, reflecting their lesser impact on the target neuron's spike time. For a weight with delay d^k from neuron i to neuron j we use

$$\Delta w_{ij}^k = \eta \left[L(\Delta t) = \eta (1 - b) e^{-\frac{(\Delta t - c)^2}{\beta^2}} + b \right], \tag{2.3}$$

after [5] (depicted in figure 1b), where the parameters b determines the effective integral over the entire learning window, β sets the width of the positive learning window and c determines the position of this peak. The value of Δt denotes the time difference between the onset of a PSP at a synapse and the time of the spike generated in the target neuron. The weight of a single terminal is limited by a minimum and maximum value, respectively 0 and w_{max}.

An input (data-point) to the network is coded by a pattern of firing times within a coding interval ΔT and each input neuron is required to fire at most once during this coding interval. In our experiments, we set ΔT to [0 – 9] ms and delays d^k to 1 – 15 ms in 1 ms intervals ($m = 16$). For the reported simulations, the parameter values for the learning function $L(\Delta t)$ are set to: $b = -0.2$,

$c = -2.85$, $\beta = 1.67$, $\eta = 0.0025$ and $w_{max} = 2.75$. To model the (strictly excitatory) post-synaptic potentials, we used an α-function:

$$\varepsilon(t) = \frac{t}{\tau} e^{(1-\frac{t}{\tau})}, \tag{2.4}$$

with $\tau = 3.0$ ms, effectively implementing leaky-integrate-and-fire spiking neurons.

Results obtained for a formal model of spike-based Hebbian learning by Kempter *et al.* [10] show that the class of learning rules as in (2.3) can pick up temporal correlations on the time scale of the learning-window and does not need overall weight normalization as required in most traditional Hebbian learning algorithms.

Previous Results and Open Questions. In [5] Natschläger & Ruf showed that artificially constructed clusters of inputs firing within the encoding interval are correctly clustered in an unsupervised manner, but the type of clusters they consider severely limits applicability. For N input neurons, a cluster C in [5] is defined of dimensionality $M \leq N$ with M-dimensional location $\{T_1, \ldots T_M\}$, T_i being the spike-time of input neuron i. For such a setup it was found that the RBF neurons converged reliably to the centers of the clusters, also in the presence of noise and randomly spiking neurons.

In practice, problems arise when applying this scheme to more realistic data. A first issue concerns the *coding* of input: following the aforementioned method, we were not able to successfully cluster data containing significantly more clusters than input-dimensions, especially in the case of low dimensionality. This problem is associated with the minimum width β of the learning function $L(\Delta t)$, leading to a fixed minimal spatial extend of a learned cluster, potentially (much) larger than the actual cluster size. In fact, for 2-dimensional input containing more than two clusters, the above algorithm failed in our experiments for a wide range of parameters. Furthermore, the finite width of the learning rule effectively inhibits the detection of multiple nearby clusters of smaller size relative to the width of the learning function, requiring advance knowledge of the effective cluster-scale. Hence, to achieve practical applicability, it is necessary to develop an encoding that is scalable in terms of cluster capacity and precision and that is also efficient in terms of the number of input-neurons required. In the following section, we present improvements to the architecture that address these issues.

3. Encoding continuous input variables in spike-times

To extend the encoding precision and clustering capacity, we introduce a method for encoding input-data into temporal spike-time patterns by population coding. Although our encoding is simple and elegant, we are not aware of any previous encoding methods for transforming continuous data into spike-time patterns and describe the method in detail.

As a means of population coding, we use multiple local receptive fields to distribute an input variable over multiple input neurons. Such a population code where input variables are encoded with graded and overlapping activation functions is a well-studied method for representing real-valued parameters (e.g.: [11; 12; 13; 14; 15; 16]). In these studies, the activation function of an input-neuron is modeled as a local receptive field that determines the firing rate. A translation of this paradigm into relative firing-times is straightforward: an optimally stimulated neuron fires at $t = 0$, whereas a value up to say $t = 9$ is assigned to less optimally stimulated neurons (depicted in figure 2).

For actually encoding high-dimensional data in the manner described above, a choice has to be made with respect to the dimensionality of the receptive-fields of the neurons. We observe that the least expensive encoding in terms of neurons is to independently encode the respective input variables: each input-dimension is encoded by an array of 1-dimensional receptive fields. Improved representation accuracy for a particular variable can then be obtained by sharpening the receptive fields and increasing the number of neurons [15]. Such coarse coding has been shown to be statistically bias-free [12] and in the context of spike-time patterns we have applied it successfully to supervised pattern classification in spiking neural networks [8].

In our experiments, we determined the input ranges of the data, and encoded each input variable with neurons covering the whole data-range. For a range $[I^n_{min} \ldots I^n_{max}]$ of a variable n, m neurons

were used with Gaussian receptive fields. For the i^{th} neuron coding for variable n, the center of the Gaussian was set to $I^n_{min}+(i-1)\cdot\frac{\{I^n_{max}-I^n_{min}\}}{m-2}$ and the width $\sigma=\beta\frac{\{I^n_{max}-I^n_{min}\}}{m-2}$. For β, values between 1.0 and 2.0 were tried, and for the reported experiments a value of 1.5 was used as it produced the best results. For each input pattern, the response values of the neurons encoding the respective variables were calculated, yielding N x $m(n)$ values between 0 and 1 (N: dimension of data, $m(n)$: number of neurons used to encode dimension n). These values were then converted to delay times, associating $t=0$ with a 1, and increasingly later times up to $t=10$ with lower responses. The resulting spike-times were rounded to the nearest internal time-step, and neurons with responses larger than $t=9$ were coded to not fire, as they were considered to be insufficiently excited. The encoding of a single input-value a by receptive field population coding is depicted in figure 2.

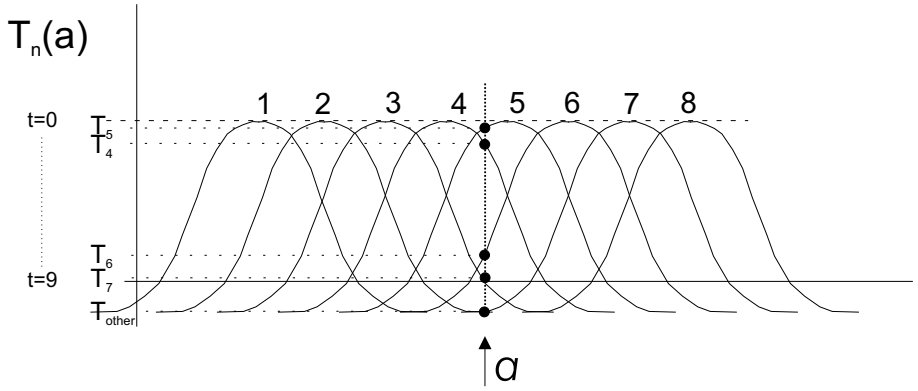

Figure 2: Encoding with overlapping Gaussian receptive fields. An input value a is translated into firing times for the input-neurons encoding this input-variable. The highest stimulated neuron (5), fires at a time close to 0, whereas less stimulated neurons, as for instance neuron 7, fire at increasingly later times.

The temporal encoding of input-variables thus obtained has two important properties: by assigning firing times to only a small set of significantly activated neurons we achieve a sparse coding, allowing for an efficient event-based network simulation as in [6]. Also, by encoding each variable independently, we achieve a coarse coding where each variable can be encoded by an optimal number of neurons while maintaining an efficient use of neurons.

4. Clustering with Receptive Fields

The encoding described above enhances capacity and precision as compared to the original architecture in [5]. In this section, we demonstrate this for a number of artificial and real-world datasets, both for low- as well as for high-dimensional input. In section 4.1, we show examples of improved capacity and precision, in section 4.2 a method for enhanced scale-sensitivity is shown, and in section 4.3 an example of a real-world clustering task is given.

4.1 Capacity

When clustering input consisting of two separately encoded variables, we found that a network with 24 input neurons (each variable encoded by 12 neurons) was easily capable of correctly classifying 17 evenly distributed clusters, demonstrating a significant increase in the clustering capacity (figure 3a). After presenting 750 randomly chosen data-points, all 1275 cluster points were correctly classified. In figure 3b the correct clustering of less regular input is shown. In general, we found that for such single layer RBF networks, capacity was only constrained by cluster separability. By decreasing the width of the receptive fields while increasing the number of neurons, increasingly narrowly separated clusters could be distinguished, as predicted by the results from [15].

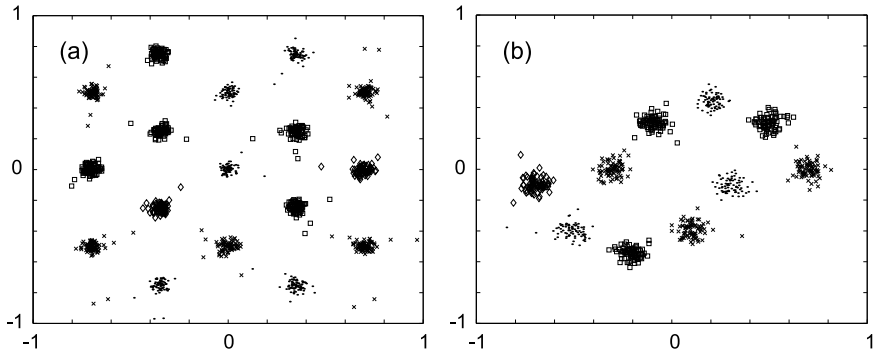

Figure 3: (a) Some 17 clusters in 2-d space, represented by two one-dimensional input variables, each variable encoded by 12 neurons (5 broadly tuned, 7 sharply tuned).(b) Classification of 10 irregularly spaced clusters. For reference, the different classes as visually extractable were all correctly clustered, as indicated by the symbol/graylevel coding.

4.2 Scale sensitivity

Encoding input variables with local receptive fields incorporates an inherent choice of spatial scale sensitivity by fixing the width of the Gaussian; using a mix of neurons with varying receptive field widths proved to significantly enhance the range of detectable detail. In experiments, we found that the use of a mixture of receptive field sizes increased the range of spatial scales by more than an order of magnitude on a number of artificially generated datasets, and in general the clustering reliability improved.

The multi-scale encoding was implemented by assigning multiple sets n_i of neurons to encode a single input dimension n, with $n_i < n_j$. As a set of neurons is evenly spread out over the data range, the width of the receptive field scales inversely with the number of neurons, achieving multi-scale sensitivity as illustrated in the clustering example in figure 4. Data consisted of one large (upper left) and two small (upper right) Gaussian clusters. The input variables were encoded with 15 respectively 10 input neurons with a mixture of receptive field widths (depicted in the side panels). The network successfully centered the output-neurons on the clusters, even though the large cluster is almost an order of magnitude larger than the two small clusters combined. When using a uniform receptive field size, the same size network often failed for this example, placing two neurons on the large cluster and one in between the two small clusters. Similar configuration is other datasets showed the same behavior, demonstrating increased spatial scale sensitivity when encoding the input with multiple sets of receptive field sizes.

In an unsupervised setting, scale is typically not, or not well, known (e.g. [17]). Encoding the input with a mixture of receptive widths thus adds multi-scale sensitivity while maintaining the network architecture and learning rule.

4.3 Clustering of realistic data

Good results were also obtained classifying more realistic and higher dimensional data. In a number of experiments the spiking RBF network correctly classified high-dimensional artificial input (10-D+). The clustering of the 4-dimensional iris data-set, encoded in 4x8 input neurons, yielded 90 to 97% correct classification. The latter result is comparable to k-nearest neighbor results.

To demonstrate the viability of the clustering with spiking neural networks on a more realistic unsupervised clustering task, we tested the network on the classification of remote sensing data. This task is a more realistic example of unsupervised clustering in the sense that the data consists of a

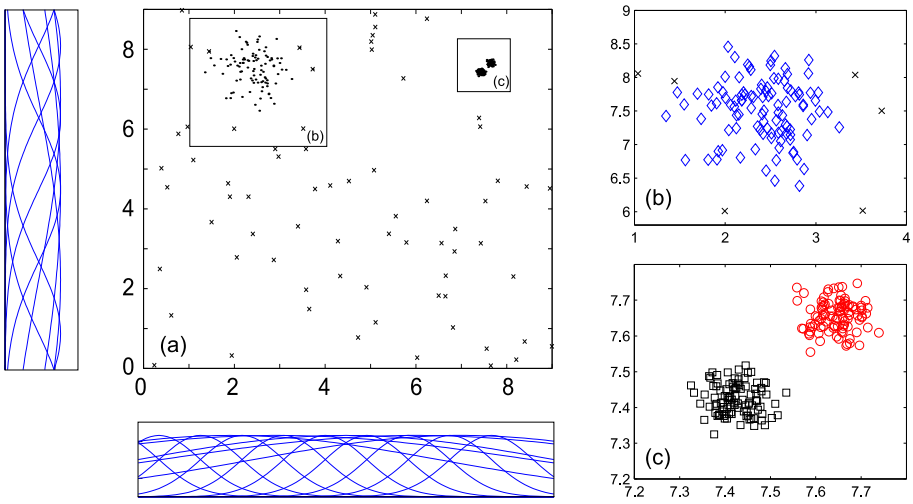

Figure 4: (a) Three clusters (upper left and upper right) of different scale with noise (crosses). (b,c) Insets: actual classification. Respective classes are marked with diamonds, squares, and circles. Noise outside the boxes or points marked by x's did not elicit a spike and were thus not attributed to a class. Side panels: receptive fields used.

large number of data-points, has ill-balanced, non-Gaussian classes and probably contains considerable noise. As an example, we took a 103x99 area (10197 datapoints) of the full 3-band RGB image as shown in figure 5a, and compared the classification obtained by the RBF network of spiking neurons to the advanced clustering algorithm for remote sensing: UNSUP [18]. Shown are three classifications of the indicated area: the first two are obtained by UNSUP (fig. 5b,c) and the third by the spiking neural network (fig. 5d). Figure 5c shows the classification of the area with UNSUP with high class separation. In figure 5d the results of classifying the same image with our RBF network is shown. The input was encoded by 3x24 = 72 neurons and at most 17 output classes were detected (output neurons). When comparing the RBF results to UNSUP running on the limited area (fig 5c), much more detail is extracted. The spiking neural network shows similarly detailed class separation as the results of running UNSUP on the entire image (fig 5b). In this case, approximately the same classes are detected, although some clusters detected by the RBF network are due to neurons centered on the same class.

The viability of the spiking RBF network for unsupervised clustering of real-world data is clearly demonstrated by the good correspondence of the identified clusters with both visual inspection of the source and the clustering obtained in figure 5b. Although exact quality comparisons have some essentially subjective aspects in remote sensing, the results show that the RBF network is robust with respect to ill-balanced, non-Gaussian and noisy real-world data.

Summarizing the results shown in this section: the examples demonstrate that capacity and precision in spiking RBF networks can be enhanced to the extent of being of practical value. The actual simulation of spiking neurons in our implementation is quite computationally intensive as compared to the optimized clustering by UNSUP (hours vs. minutes), however recent work has demonstrated however that the "event" nature of spikes in such networks allows for very efficient implementations [6].

5. Hierarchical clustering in a multi-layer network

With a few modifications to the original network, we can also create a multi-layer network of spiking neurons that is capable of hierarchical clustering based on temporal cluster distance. Cluster bound-

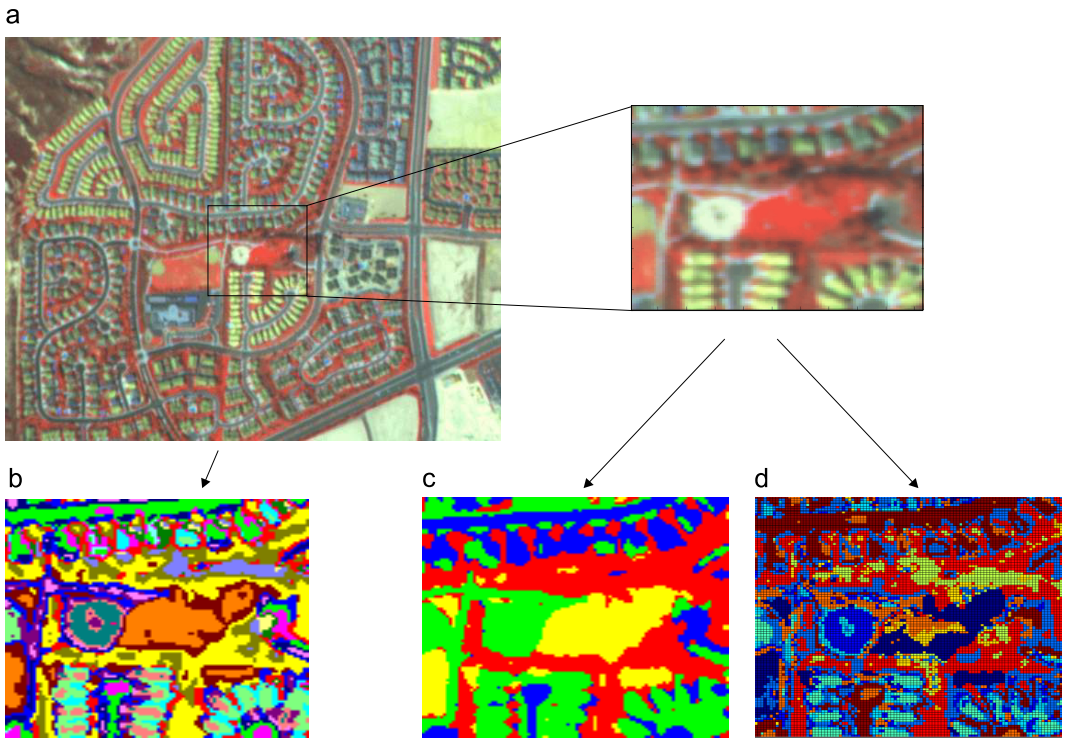

Figure 5: (a) The full image. (b) Image cutout actually clustered. (c) Classification of the cutout as obtained by clustering the entire image with UNSUP. (d) Classification of the cutout as obtained by clustering only the cutout area with UNSUP at very high separation values. (e) Spiking neural network RBF classification of the cutout image after learning from 70,000 randomly drawn datapoints from the 103x99 image.

aries in real data are often subjective, and hierarchical classification is a natural approach to this ambiguity, e.g. [19]. By classifying data with increasing or decreasing granularity based on a cluster-distance measure, multiple "views" of a dataset can be obtained. To enable hierarchical clustering in spiking RBF neurons, we observe that the differential firing times of output neurons are a monotonous function of spatial separation and could hence serve as a cluster-distance measure.

To achieve hierarchical clustering, we created a multi-layer network of spiking neurons. Given a suitable choice of neurons within the layers, a single layer yields the classification of a datapoint at a decreasing level of granularity as compared to the classification in a previous layer. The combined classification of all layers then effectively achieves hierarchical classification. To enable hierarchical classification with decreasing granularity, the neural population decreases for subsequent layers, and all n neurons within a layer are allowed to fire such that the next layer with m neurons can extract up to m clusters from "input" n neurons firing, with $m < n$. The clustering mechanism is maintained by only modifying the weights for the winning neuron within a layer. The implementation of such a winner-take-all (WTA) mechanism can be considered an abstract implementation of the more complicated WTA learning rule as explored in [20].

To implement hierarchical clustering, we added a second RBF layer to the network in the fashion described above, and successfully trained this network on a multitude of hierarchically structured datasets. An example is shown in figure 6: the data contained two clusters each consisting of two components. The winner-take-all classification found in the first RBF layer is shown in figure 6a, and correctly identifies the respective components of the two clusters. For a configuration as in figure 6a, the receptive field of any RBF neuron extends over the accompanying component. In this case,

the evaluation of a single data point elicits a response from both neurons in the cluster, albeit one somewhat later than the other. The neurons centered on the other cluster are insufficiently stimulated to elicit a response. This disparate response is sufficient for the second layer to concatenate the neurons in the first layer into two clusters (figure 6b). Thus, as we extend the network with subsequent RBF layers comprising of fewer neurons in effect we achieve hierarchical clustering with decreasing granularity: nearby components are compounded in the next layer based on relative spatial proximity as expressed in their temporal distance.

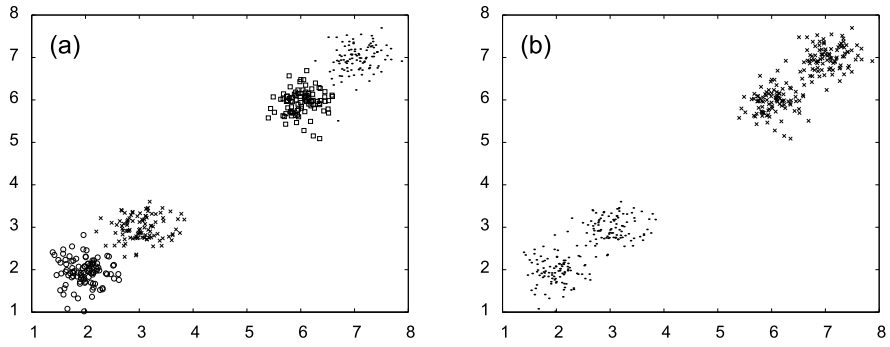

Figure 6: Hierarchical clustering in a 2 layer RBF network. (a) Clustering in the first layer consisting of 4 RBF neurons. Each data-point is labeled with a marker designating the winning neuron (squares, circles, crosses, and dots). (b) Clustering in the second layer, consisting of 2 RBF neurons. Again each data-point is labeled with a marker signifying the winning neuron (crosses and dots).

In unsupervised learning, the determination of the number of classes present in the dataset is well known problem in competitive winner-take-all networks, as it effectively is determined a-priori by the number of output neurons, e.g. [21]. In the hierarchical clustering example shown, we tuned the number of neurons to the number of components and clusters. In an RBF layer with more units than clusters or components, typically multiple output-neurons will become centered on the same cluster, especially when clusters consisted of multiple components. Correct classification is only reliably achieved when the number of RBF neurons matches the number of clusters, see also [5]. However, in the case of more neurons than components/clusters the same hierarchical clustering principle holds, as multiple neurons centered on the same component are identifiable by their strong synchrony. Hence the relative synchronization of nearby neurons is an important clue when reading the classification from a layer, as well as an effective means of coding for further (hierarchical) neuronal processing. Note that the problem is rather one of extraction than of neuronal information processing, as multiple synchronized neurons are effectively indiscriminable downstream and can hence be considered one.

6. Complex clusters.

In this section, we show how temporal synchrony can be further exploited for separating interlocking clusters by binding multiple correlated RBF neurons via the addition of reciprocal excitatory lateral connections to the first RBF-layer, thus enhancing the network clustering capabilities.

As remarked, cluster boundaries in real data are often subjective. Hierarchical clustering is only part of the solution, as some measure for grouping components into subsequent clusters has to be implemented. For complex clusters, separate parts of the same cluster can easily be spatially separated to the point where the neuronal receptive fields no longer overlap: a neuron coding for one part will no longer respond when a data-point belonging to the other part is presented. Another issue relates to the measure for concatenating components into clusters: only those components that have a certain density of data points "in between" should be concatenated, as implemented for instance in [18] in an unsupervised clustering algorithm using such a "separation density" measure. The situation is sketched in figure 7.

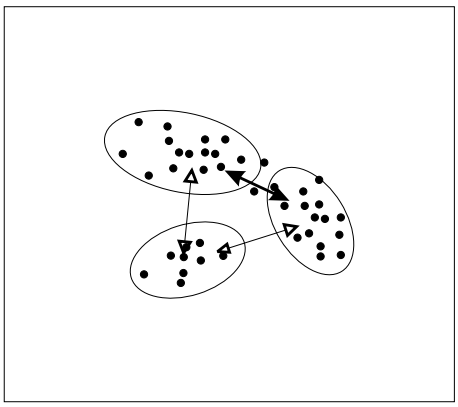

Figure 7: Weights connecting different part of a single cluster. Given two clusters of datapoints (solid circles) classified by three RBF neurons (elliptic receptive fields), datapoints between two RBF neurons strengthen the mutual lateral connections (solid arrows), whereas the connections to the equidistant third RBF neuron are not due to the lack of points "in between".

By adding excitatory lateral connections to the first RBF-layer and using a competitive SOM-like rule for modifying connections, nearby neurons become tightly coupled and are in effect bound together as they synchronize their firing times. As only the weights between temporally proximate neurons are augmented, ultimately neurons centered on the same cluster are synchronized due to the data points that lie "in between" neighboring neurons. These points elicit approximately the same time-response from the nearest neurons, strengthening their mutual connections. This process does not take place for neurons coding for different clusters, due to the relative lack of "in between" points (figure 7). As a set of neurons associated with a complex cluster synchronize their respective firing-times when a data-point lying within this structure is presented to the network, the temporal response from the first RBF layer enables a correct classification in the second layer.

Following the idea outlined above, we implemented such lateral connections in a multi-layer network and successfully classified a number of artificial datasets consisting of interlocking clusters. The lateral connections were modeled as the feedforward connections, albeit with only one delay $d^1 = 1$ ms. The lateral connections from the winning neuron are adapted using a "difference of Gaussians (DOG)" or "Mexican hat" learning function:

$$L(\Delta t) = e^{-\Delta t^2/b^2}\{(1-c)e^{-\Delta t^2/\beta^2} + c\}, \tag{6.1}$$

with $b = 4.5$, $c = -0.2$, $\beta = 0.8$. The "Mexican hat" learning functions defines the temporal difference for which connection are strengthened or weakened, where β determines the temporal width of the positive part of the learning function, and b determines the width of the weight depressing trough. During learning, the maximal allowed strength of the lateral connections was slowly increased from 0 to a value sufficiently strong to force connected neurons to fire. Experiments with these connections incorporated in the multi-layer network yielded the correct classification of complex, interlocking clusters, an example is shown in figure 8.

Summarizing, the addition of lateral excitatory connections with competitive SOM-learning synchronizes spatially correlated neurons within an RBF layer. This temporal property then enables the correct clustering of complex non-linear clusters in a multi-layer network, without requiring additional neurons.

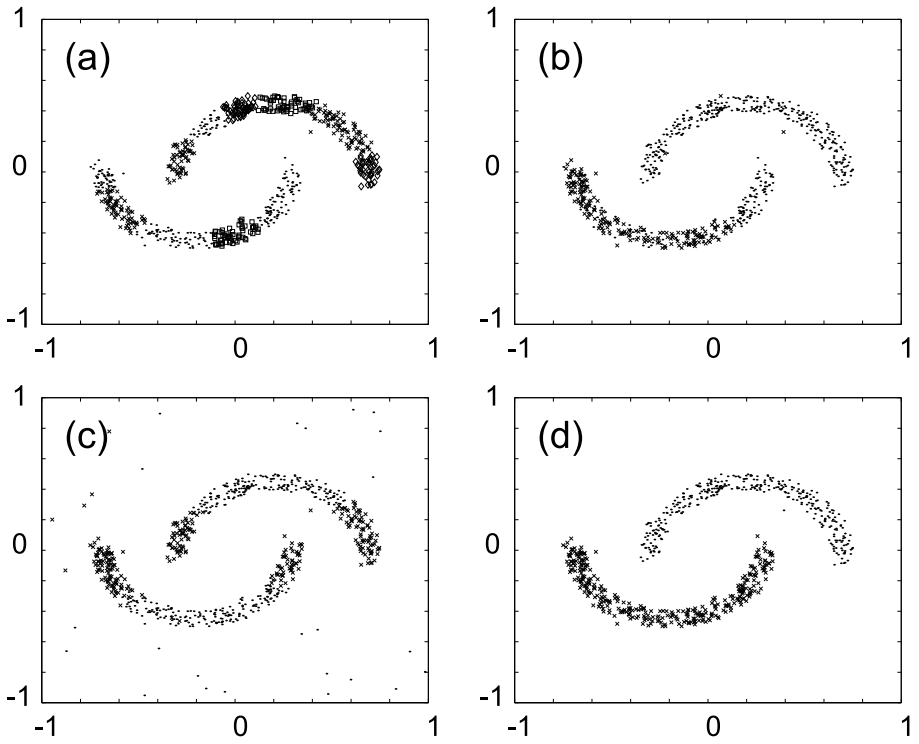

Figure 8: Clustering of two interlocking clusters in a multi-layer RBF network. (a) classification in the first layer: 11 outputs, the clusters are spread over 5 respectively 6 neurons (the respective classifications are denoted by different markers and graylevels). (b) Incorrect clustering in the second layer with two RBF neurons and input from (a), without lateral connections. (c) Incorrect classification as obtained in a single-layer network. (d) Correct classification (100%) in the second layer, with lateral connections. Each input variable was encoded by 12 input neurons (3 broadly and 9 sharply tuned).

7. Conclusions

We have shown that temporal spike-time coding in a network of spiking neurons is a viable paradigm for unsupervised neural computation, as the network is capable of clustering realistic and high-dimensional data. We investigated clustering for continuously valued input and found that our coarse coding scheme of the input data was effective and efficient in terms of required neurons. In a test on realistic data, our coarse coding approach proved to be effective on the unsupervised remote-sensing classification problem.

To detect non-globular or complex interlocking clusters we introduced an extension of the network to allow for multiple RBF-layers, enabling hierarchical clustering. When we added excitatory lateral connections we showed that a competitive SOM-like lateral learning rule enhances the weights between neurons that code for nearby, uninterrupted cluster-parts. This leads to synchronization of neurons coding for the same cluster and was shown to enable the correct classification of larger cluster-like structures in a subsequent layer. Hence the combination of multi-layer RBF and competitive SOM-like lateral learning adds considerably to the clustering capabilities, while the number of neurons required remains small. Also, we demonstrated how a local Hebbian learning-rule can both induce and exploit synchronous neurons resulting in enhanced unsupervised clustering capabilities, much as theorized in neurobiology.

We want emphasize that the main contribution of this paper lies in the demonstration that neural networks based on the alternative and possibly biologically more plausible coding paradigm are

effective and efficient for unsupervised clustering tasks when using the encoding and architectural strategies we described. Our results with supervised learning in spike-time based neural networks ([8]) strengthen this case. Depending on the application area, spiking neural networks could contribute to efficient pulse-stream VLSI implementations, or to novel neural network architectures for higher-level functions such as dynamic binding.

References

1. W. Singer and C.M. Gray, "Visual feature integration and the temporal correlation hypothesis," *Annu. Rev. Neurosci.*, vol. 18, pp. 555–586, 1995.
2. W. Maass, "Fast sigmoidal networks via spiking neurons," *Neural Comp.*, vol. 9, no. 2, pp. 279–304, 1997.
3. Ch. von der Malsburg, "The what and why of binding: The modeler's perspective," *Neuron*, vol. 24, pp. 95–104, 1999.
4. D.V. Buonomano and M. Merzenich, "A neural network model of temporal code generation and position-invariant pattern recognition," *Neural Comp.*, vol. 11, no. 1, pp. 103–116, 1999.
5. T. Natschläger and B. Ruf, "Spatial and temporal pattern analysis via spiking neurons.," *Network: Comp. in Neural Syst.*, vol. 9, no. 3, pp. 319–332, 1998.
6. A. Delorme, J. Gautrais, R. VanRullen, and S.J. Thorpe, "Spikenet: A simulator for modeling large networks of integrate and fire neurons.," *Neurocomputing*, pp. 989–996, 1999.
7. J.J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representation," *Nature*, vol. 376, pp. 33–36, 1995.
8. S.M. Bohte, J.N. Kok, and H. La Poutré, "Spike-prop: error-backprogation in multi-layer networks of spiking neurons.," in *Proceedings of ESANN'2000*, M. Verleysen, Ed. 2000, pp. 419–425, D-Facto.
9. W. Gerstner, "Time structure of the activity in neural network models," *Phys. Rev. E*, vol. 51, pp. 738–758, 1995.
10. R. Kempter, W. Gerstner, and J.L. van Hemmen, "Intrinsic stabilization of output rates by spike-based hebbian learning," *submitted to Neural Comp.*, pp. XX–XX, 2000.
11. C. W. Eurich and S. D. Wilke, "Multi-dimensional encoding strategy of spiking neurons," *Neural Comp.*, vol. in press, 2000.
12. P. Baldi and W. Heiligenberg, "How sensory maps could enhance resolution through ordered arrangements of broadly tuned receivers," *Biol. Cybern.*, vol. 59, pp. 313–318, 1988.
13. H.P. Snippe and J.J. Koenderink, "Information in channel-coded systems: correlated receivers.," *Biol. Cybern.*, vol. 67, no. 2, pp. 183–190, 1992.
14. K.-C. Zhang, I. Ginzburg, B.L. McNaughton, and T.J. Sejnowski, "Interpreting neuronal popula-

tion activity by reconstruction: Unified framework with application to hippocampal place cells," *J. Neurophysiology*, vol. 79, pp. 1017–1044, 1998.

15. K. Zhang and T.J. Sejnowski, "Neuronal tuning: To sharpen or broaden?," *Neural Comp.*, vol. 11, no. 1, pp. 75–84, 1999.
16. A. Pouget, S. Deneve, J.-C. Ducom, and P.E. Latham, "Narrow versus wide tuning curves: What's best for a population code?," *Neural Comp.*, vol. 11, no. 1, pp. 85–90, 1999.
17. I.D. Guedalia, M. London, and M. Werman, "An on-line agglomerative clustering method for nonstationary data," *Neural Comp.*, vol. 11, no. 2, pp. 521–540, 1999.
18. C.H.M. van Kemenade, H. La Poutré, and R.J. Mokken, "Unsupervised class detection by adaptive sampling and density estimation," in *Spatial Statistics and Remote Sensing*, A. Stein and F. van der Meer, Eds. Kluwer, 1999.
19. J.J. Koenderink, "The structure of images," *Biol. Cybern.*, vol. 50, pp. 363–370, 1984.
20. K.P. Körding and P. König, "Learning with two sites of synaptic integration," *Network: Comput. Neural Syst.*, vol. 11, pp. 1–15, 2000.
21. J.M. Zurada, *Introduction to Artifical Neural Systems*, St. Paul, MN: West, 1992.