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

Sander M Bohte, Han La Poutré, Joost N Kok

Abstract— **We demonstrate that spiking neural networks encoding information in the timing of single spikes 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 can induce 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 the induced synchronization of neurons within early RBF layers allows for the subsequent detection of complex clusters.**

Keywords— **Spiking neurons, unsupervised learning, high-dimensional clustering, complex clusters, Hebbian-learning, synchronous firing, sparse coding, temporal coding, coarse coding.**

I. Introduction

It is well known that cortical neurons produce all-or-none action potentials, or spikes, but the timing of these pulses has only recently been recognized as a possible means of neuronal information coding. As the biological evidence has been mounting, e.g. [1], it has been shown theoretically that coding with the timing of single spikes 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], [7], [8], [9]. In particular, Hopfield [10] 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. However, as we argue in detail in section II, this method is limited in both cluster capacity as well as in precision.

Sander Bohte and Han La Poutré are with the Netherlands Center for Computer Science and Mathematics (CWI), Amsterdam, The Netherlands. E-mail: {S.M.Bohte, Han.La.Poutre}@cwi.nl. Joost N. Kok is with the Leiden Institute of Advanced Computer Science (LIACS), Leiden University, Leiden, The Netherlands. E-mail: joost@liacs.nl

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 from 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 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 [11], these results show that single spike-time coding is a viable means for neural information processing on real-world data within the novel paradigm of artificial spiking neural networks.

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

II. Networks of delayed spiking neurons

In this section, we describe the spiking neural network as introduced for unsupervised clustering in [5], as well as 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, the "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 [12], where the time-varying impact of a spike on a post-synaptic neuron is described by a spike-response function, also referred to as Post-Synaptic Potential (PSP). 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 $\varepsilon(t, t_i)$ weighted by the synaptic efficacy w_{ij}:

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

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, inset). 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 when the post-synaptic potential resulting from terminal k starts rising. This PSP is then weighted by the synaptic efficacy w_{ij}^k. The input to a neuron j becomes:

$$x_j(t) = \sum_{i \in \Gamma_j} \sum_{k=1}^{m} w_{ij}^k \varepsilon(t - t_i - d^k). \tag{2}$$

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 neuron [5]. For unsupervised learning, a Winner-Take-All learning rule modifies the weights between the input neurons and the neuron first to fire in the output layer using a time-variant of Hebbian learning: if the start of a PSP at a synapse slightly precedes a spike in the output neuron, the weight of this synapse is increased, as it had 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 ouput neuron's spike time. For a weight with delay d^k from neuron i to neuron j we use

$$\Delta w_{ij}^k = \eta L(\Delta t) = \eta(1-b)e^{-\frac{(\Delta t - c)^2}{\beta^2}} + b, \tag{3}$$

after [5] (depicted in figure 1b), where the parameter b determines the effective size of the integral over the entire learning window (usually negative), β sets the width of the positive part of the 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 synaptic terminal and the time of the spike generated in the winning output neuron. The weight of a single terminal is limited by a minimum and maximum value, respectively 0 and w_{max}, where learning drives the individual weights of the synaptic terminals to one of the extremes. For a single connection, the minimal number of consecutive delayed synaptic weights driven to w_{max} is determined by the width parameter β: if an input neuron were to precede the firing of the output-neuron by a fixed amount Δt_{ij}, the set of connecting delayed terminals that is positively reinforced is determined by the width of the positive part of the learning window. This thus results in a minimal value for the efficacy between an input neuron that codes for part of a cluster, and the corresponding output neuron, both in length of time, as well as in size. Larger efficacies can be learned when the size of a cluster extends over a larger temporal width (i.e., Δt_{ij} varies), and more weights are thus driven to the maximal value. If the temporal variation becomes too large, the average delayed weight adjustment due to (3) becomes negative, as the integral over the learning-window is then negative, and all weights converge to zero, thus ignoring input neurons that only contribute "noise" (see also [5]). This dynamic recruitment of delayed terminals negates the need for overall weight normalization (see also the delay selection in [13]).

An input (data-point) to the network is coded by a pattern of firing times within a coding interval T_Δ and each input neuron is allowed to fire at most once during this interval. In our experiments, we set T_Δ to $[0-9]$ ms and delays d^k to $\{1, \ldots, 15\}$ [ms] in 1 ms intervals ($m = 16$), after [5]. For the experiments, the parameter values for the learning function $L(\Delta t)$ are set to: $b = -0.2$, $c = -2.85$, $\beta = 1.67$, and $\eta = 0.0025$. To model the (strictly excitatory) post-synaptic potentials, we used an α-function:

$$\varepsilon(t) = \frac{t}{\tau} e^{(1-\frac{t}{\tau})}, \tag{4}$$

with $\tau = 3.0$ ms, implementing leaky-integrate-and-fire spiking neurons. The parameter values are taken from [5]; deviations from these defaults in experiments are noted.

To clarify the rationale behind the selection of the respective parameters, we briefly discuss their effects. Contrary to the experiments in [5], the majority of the input-neurons in our network does not fire: we found that a larger value of c was required to select a stable subset of synaptic terminals. Values between approximately 2 and 3(ms) yielded stable results. For smaller and in particular larger values, the selected delayed terminals tended to drift either to zero or out of range, effectively negating the connection. In the experiments, any value of β that selected a minimum of three consecutive delayed terminals typically yielded better results than other settings. Provided that the value c results in stable weight selection, the values of b and β determine the minimal extent of the clusters learned. Despite this minimal extend, we do not lose generality with these fixed parameters, provided that we use the input-encoding as outlined in Section III.

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 limits applicability. For N input neurons, a cluster C in [5] is of dimensionality $M \leq N$, with M-dimensional location $\{s_1, \ldots s_M\}$, s_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 extent 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.

III. Encoding continuous input variables in spike-times

To improve the encoding precision and clustering capacity, we introduce a method for encoding input-data by population coding. The aim of this encoding is to increase the temporal distance between the temporal input-patterns associated with respective (input) data-points. Since we use delayed terminals with a resolution of 1 ms, the discriminatory power of the unsupervised learning rule is naturally limited to approximately this resolution. The encoding increases the temporal distance between points, and thus the separability of clusters. 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 therefore, we 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.: [14], [15], [16], [17], [18], [19]). 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 [18]. Such coarse coding has been shown to be statistically bias-free [15] and in the context of spike-time patterns we have applied it successfully to supervised pattern classification in spiking neural networks [11].

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 $I^n_{min} + \frac{2i-3}{2} \cdot \frac{\{I^n_{max} - I^n_{min}\}}{m-2}$ ($m > 2$), positioning one input neuron outside the data-range at both ends. The width was set to $\sigma = \frac{1}{\gamma} \frac{\{I^n_{max} - I^n_{min}\}}{m-2}$ (with $m > 2$). For γ, a range of values was tried, and, unless stated otherwise, for the 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 $N \times m(n)$ 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.

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, enabling us to process only a list of "active" connections, instead of all connections (event-based network simulation, e.g. [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.

IV. Clustering with Receptive Fields

We investigate the clustering capabilities of spiking neural networks where the input is encoded with receptive fields. With such encoding, each data-point is translated into a multi-dimensional vector of spike-times (spike-time vector). Clustering relies on a single output neuron firing earlier then the other output neurons for data-points from a single cluster. The optimal activation of such an output neuron is achieved when the spikes of input neurons arrive at the output neuron simultaneously. This is what the Hebbian learning-rule (3) accomplishes, provided that the input lies within a particular cluster. If the distance between clusters is sufficient, the winner-takes-all competition between output neurons tunes these output neurons to the spike-time vectors associated with the centers of the respective clusters. The activation of a neuron for a given pattern then depends on the distance between the optimal and actual spike-time vector, resulting in increasingly later firing times (or none) with increasing distance from the cluster-center. We will use this diverging temporal firing pattern later for subsequent multi-layer clustering.

The encoding described in Section III enhances capacity and precision as compared to the original architecture in [5]. In this Section, we show this for a number of artificial and real-world datasets, both for low- as well as for high-dimensional input. In section IV-A, we show examples of improved capacity and precision, in section IV-B a method for enhanced scale-sensitivity is shown, and in section IV-C a examples of real-world clustering tasks are given.

A. Capacity

In this section, we report on experiments that show how the outlined encoding allows for increased capacity, e.g. by encoding variables with more neurons, many different clusters can be separated.

In experiments, we cluster input consisting of two separately encoded variables, and 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 (just as predicted by theoretical work on the properties of receptive field encodings, e.g. [18]).

B. 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 of neurons to encode a single input dimension n. For different scales, say I and J, each scale was encoded with increasingly less neurons, scale I encoded with n_i neurons and scale J with n_j neurons, 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 proportional to 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 neurons for the variable along the x-axis, and 10 input neurons for the y-variable. These neurons were given a mixture of receptive field widths, 3 broad and 7 tight Gaussians for the y-variable, and 5 broad and 10 tight Gaussians for the x-variable (depicted in the side panels). The width σ_t of the tight Gaussians was set to $\frac{1}{\gamma}(I_{max} - I_{min})/(m-2)$, with $\gamma = 1.5$. The width σ_b of the broad Gaussians was set to $\frac{1}{\gamma}(I_{max} - I_{min})/(m+1)$, with $\gamma = 0.5$. This results in widths of respectively $\sigma_b = 4.5$ and $\sigma_t = 1.2$ (y-axis), and $\sigma_b = 3$ and $\sigma_t = 0.5$ (x-axis). The tight Gaussians were distributed along the respective axes as outlined in Section III, the broad Gaussians were all evenly placed with their centers *inside* the respective data-ranges, with center i placed at $I^n_{min} + i \cdot \frac{\{I^n_{max} - I^n_{min}\}}{m+1}$. Note that the width of the small clusters is still substantially smaller than the receptive field sizes of the tight Gaussians. As the spike-time vectors for a particular data-point are derived from the activation values of the population of neurons, the spike-time vectors corresponding to the respective cluster centers are still sufficiently distant to make discrimination possible.

The learning-rule 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 configurations with 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. [20]). Encoding the input with a mixture of receptive widths thus adds multi-scale sensitivity while maintaining the network architecture and learning rule.

C. Clustering of realistic data

Good results were also obtained when classifying more realistic and higher dimensional data. As an example of relatively simple but realistic data, we clustered Fisher's 4-dimensional Iris data-set. The input was encoded in 4×8 input neurons, classification yielded 92.6 ± 0.9 % correct classification (over 10 runs, with 1 failing clustering removed and with parameter settings as outlined in section II). Alternative clustering methods, like k-means[1] and a Self-Organizing Map (SOM)[2], yielded somewhat worse results, see table **??**. Since our SOM and k-Means methods can probably be improved upon, this result indicates that the clustering capability of the RBF network is at least competitive with similar methods.

To assess the feasibility of using the RBF network for clustering in high-dimensional data, a number of artificial data-sets (10-D+) were generated (not shown). In all experiments, the spiking RBF network correctly classified these datasets.

To show the viability of clustering with spiking neural networks on a more "real-life" unsupervised clustering task, we trained the network to classify a set of remote sensing data. This task is a more realistic example of unsupervised clustering in the sense that the data consists of a large number of data-points, has non-Gaussian classes, and probably contains considerable noise. The distribution of the data-points over the classes is also ill-balanced: some classes have many data-points and others only a few (e.g. grasslands vs. houses). As an example, we took a $103 \times 99 = 10197$ data-points of the full 3-band RGB image shown in figure 5a, and compared the classification obtained by the RBF network of spiking neurons to that of a SOM-network, both for the detection of 17 classes. As a benchmark, we use the results obtained by the UNSUP clustering algorithm for remote sensing [21] on the entire image (figure 5b). Figure 5c shown the classification of the area with a SOM-network, and figure 5d shows the classification by the spiking neural network. Note that both methods required approximately the same amount of computer runtime. When comparing figures 5c and 5d to the UNSUP classification, visual inspection shows that the respective classifications do not differ much, although some clusters detected by the RBF network are due to multiple neurons centered on the same class: both RBF and SOM classifications seem reasonable. Although the correctness of remote sensing classifications is notoriously difficult to determine due to lack of ground evidence (labeled data), the results show that our RBF network is robust with respect to ill-balanced, non-Gaussian and noisy real-world data.

Summary. The experiments show that capacity and precision in spiking RBF networks can be enhanced such that they can be used in practice. The simulation of spiking neurons in our implementation is quite computationally intensive as compared to the optimized clustering by UNSUP (minutes vs. seconds), but takes approximately the same amount of time as SOM methods, and is only somewhat slower than k-Means (though run in Matlab). Possible speedups could be accomplished by using more computationally efficient spiking neural network models, for instance by taking a spike as an "event" and interpolating all deterministic effects between these events, e.g. the time-evolution of the membrane-potential under a set of preceding PSP's [22].

V. Hierarchical clustering in a multi-layer network

With a few modifications to the original network, we can create a multi-layer network of spiking neurons that is capable of hierarchical clustering based on temporal cluster distance. Cluster boundaries in real data are often subjective, and hierarchical classification is a natural approach to this ambiguity, e.g. [23]. 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 monotonic decreasing function of spatial separation, e.g. the further a data-point lies from the center of a neuron, the later the neuron fires. This could serve as a cluster-distance measure.

To achieve such hierarchical clustering, we created a multi-layer network of spiking neurons. Given a suitable choice of neurons within the layers, respective layers yield the classification of a data-point 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 cay 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.

To implement hierarchical clustering in such a fashion, we added a second RBF layer to the network as 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 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

[1] from SOMToolbox at `www.cis.hut.fi/projects/somtoolbox/`.
[2] from Matlab R12.

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.

In unsupervised learning, the determination of the number of classes present in the dataset is a well-known problem in competitive winner-take-all networks, as it effectively is determined a-priori by the number of output neurons, e.g. [24]. In the hierarchical clustering example, 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 to be one neuron.

VI. 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.

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 another part of the same cluster 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 the UNSUP clustering algorithm [21]. The situation is depicted in figure 7. The analogous problem exists when discriminating different clusters that are nearby. In both cases, when such clusters are covered by multiple neurons that are concatenated in a next layer, they might suffer from the fact that some of these neurons belonging to different clusters are in fact closer together than to other neurons in the same cluster (and thus fire closer together).

We present a SOM-like addition to the network to overcome this problem: by adding excitatory lateral connections to an 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 synchronize their respective firing-times when a data-point lying within a cluster-structure is presented to the network, the temporal response from the first RBF layer enables a correct classification in the second layer.

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:

$$L(\Delta t) = e^{-\Delta t^2/b^2}\{(1-c)e^{-\Delta t^2/\beta^2} + c\}, \qquad (5)$$

with $b = 4.5$, $c = -0.2$, $\beta = 0.8$. The "Mexican hat" learning functions defines the temporal difference for which connections 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 is 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.

VII. Discussion and 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 "real-life" 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 learning 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 relatively 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.

The intuitive approach to within-layer synchronization as an aide for clustering is inspired by efforts to implement image-segmentation in neural networks via dynamic synchronization of spiking neurons that code for those parts of an image that are part of the same object, e.g. [7], [8], [9]. Clustering entails the classification of a data-point in terms of other data-points with "similar" properties in some, potentially high-dimensional, input-space, and is not necessarily concerned with the spatial organization of the data (e.g. the UNSUP remote sensing method used for figure 5). As such, clustering is essentially a different problem. For clustering, it is important that the number of neurons involved scales moderately with increasing dimensionality of the data, whereas image-segmentation is inherently two or three dimensional and is not, or less, subject to this restriction. However, our results lend further support for the use of precise spike timing as a means of neural computation and provide common ground in terms of the coding paradigm for these different problems.

We want to 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 [11] strengthen this case and contribute to establishing that spike-time coding is a viable neural information processing strategy.

Acknowledgments

The authors would like to acknowledge the helpful suggestions of the referees, both for this paper, as well as the referees of a version of this paper presented at IJCNN 2000.

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] P. König and T. B. Schillen, "Stimulus-dependent assembly formation of oscillatory responses: I.Synchronization," *Neural Computation*, vol. 3, no. 2, pp. 155–166, 1991.

[8] K. Chen and D.L. Wang, "Perceiving without learning: from spirals to insideoutside relations," in *Advances in Neural Information Processing Systems*, M.J. Kearns, S.A. Solla, and D. Cohn, Eds. 1999, vol. 11, The MIT Press.

[9] S.R. Campbell, D.L. Wang, and C. Jayapraksh, "Synchrony and desynchrony in integrate-and-fire oscillators," *Neural Computation*, vol. 11, pp. 1595–1619, 1999.

[10] J.J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representation," *Nature*, vol. 376, pp. 33–36, 1995.

[11] S.M. Bohte, J.N. Kok, and H. La Poutré, "Spike-prop: error-backprogation in multi-layer networks of spiking neurons," *Neurocomputing*, to appear, An abstract has appeared in the proceedings of ESANN'2000.

[12] W. Gerstner, "Time structure of the activity in neural network models," *Phys. Rev. E*, vol. 51, pp. 738–758, 1995.

[13] W. Gerstner, R. Kempter, J.L. van Hemmen, and H. Wagner, "A neuronal learning rule for sub-millisecond temporal coding," *Nature*, vol. 383, pp. 76–78, 1996.

[14] C. W. Eurich and S. D. Wilke, "Multi-dimensional encoding strategy of spiking neurons," *Neural Comp.*, vol. in press, 2000.

[15] 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.

[16] H.P. Snippe and J.J. Koenderink, "Information in channel-coded systems: correlated receivers," *Biol. Cybern.*, vol. 67, no. 2, pp. 183–190, 1992.

[17] K.-C. Zhang, I. Ginzburg, B.L. McNaughton, and T.J. Sejnowski, "Interpreting neuronal population activity by reconstruction: Unified framework with application to hippocampal place cells," *J. Neurophysiology*, vol. 79, pp. 1017–1044, 1998.

[18] K. Zhang and T.J. Sejnowski, "Neuronal tuning: To sharpen or broaden?," *Neural Comp.*, vol. 11, no. 1, pp. 75–84, 1999.

[19] 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.

[20] 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.

[21] 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.

[22] M. Mattia and P. Del Giudice, "Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses," *Neural Computation*, vol. 12, no. 10, pp. 2305–2329, 2000.

[23] J.J. Koenderink, "The structure of images," *Biol. Cybern.*, vol. 50, pp. 363–370, 1984.

[24] J.M. Zurada, *Introduction to Artifical Neural Systems*, St. Paul, MN: West, 1992.

Iris clustering	
method	error training-set
Spiking RBF	92.6% ± 0.9%
k-Means	88.6% ± 0.1%
SOM	85.33% ± 0.1%

TABLE I

Unsupervised clustering of Fisher's Iris-dataset. The k-Means method was set to $k = 3$, SOM was run with 3 output neurons.

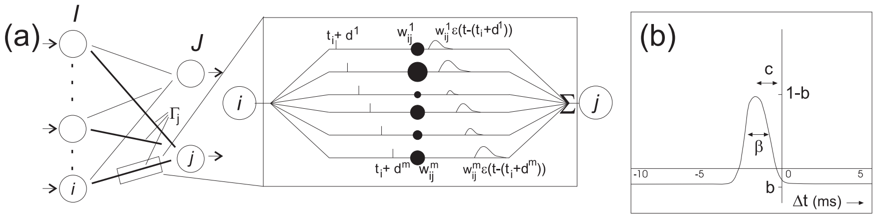

Fig. 1. (a) Network connectivity and a single connection composed of multiple delayed synapses. Neurons in layer J receive connections from neurons Γ_j in layer I. Inset: a single connection between two neurons consists of m delayed synaptic terminals. A synaptic terminal k is associated with a weight w^k_{ij}, and delay d^k. A spike from neuron i thus generates m delayed spike-response functions ($\varepsilon(t-(t_i+d^k))$, the sum of which generates the membrane-potential in neuron j. (b) Graph of the learning function $L(\Delta t)$. The parameter 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.

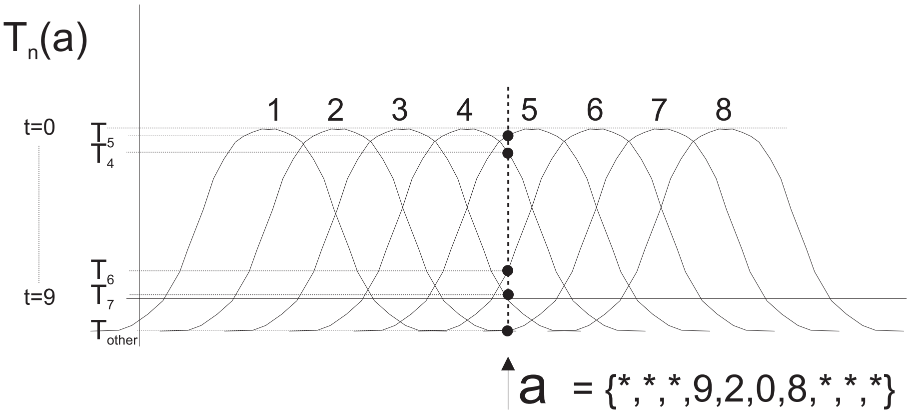

Fig. 2. Encoding with overlapping Gaussian receptive fields. An input value a is translated into firing times for the input-neurons encoding this input-variable. The highest stimulated neuron (5), fires at a time close to 0, whereas less stimulated neurons, as for instance neuron 7, fire at increasingly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

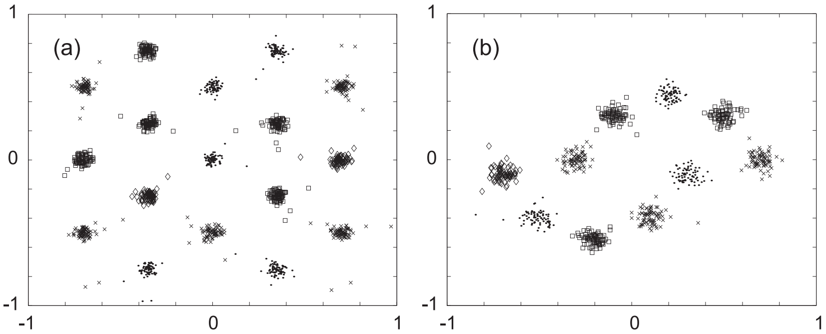

Fig. 3. (a) Some 17 clusters in 2-d space, represented by two one-dimensional input variables, each variable encoded by 12 neurons (5 broadly tuned, 7 sharply tuned).(b) Classification of 10 irregularly spaced clusters. For reference, the different classes as visually extractable were all correctly clustered,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

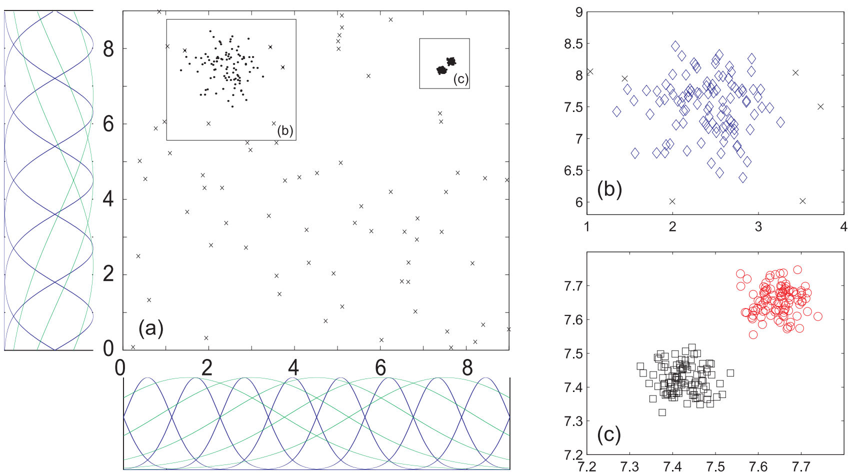

Fig. 4. (a) Three clusters (upper left and upper right) of different scale with noise (crosses). (b,c) Insets: actual classification. Respective classes are marked with diamonds, squares, and circles. Noise outside the boxes or points marked by x's did not elicit a spike and were thus not attributed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

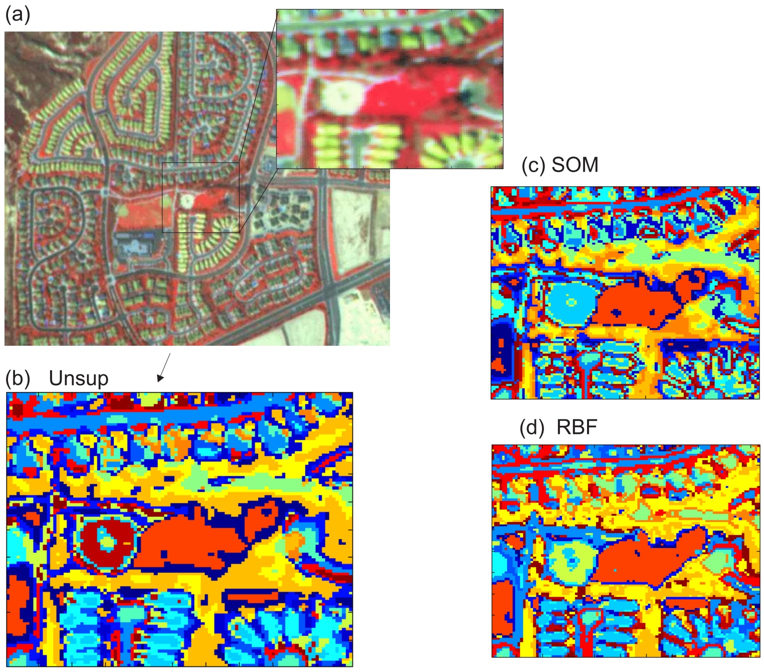

Fig. 5. (a) The full image. Inset: image cutout actually clustered. (b) Classification of the cutout as obtained by clustering the entire image with UNSUP. (c) Classification of the cutout as obtained by clustering with SOM algorithm. (d) Spiking neural network RBF classification of the cutout image after learning from 70,000 randomly drawn data-points from the 103x99 image.

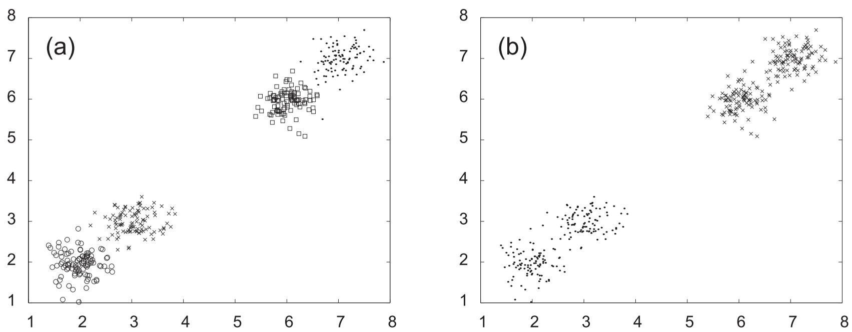

Fig. 6. Hierarchical clustering in a 2 layer RBF network. (a) Clustering in the first layer consisting of 4 RBF neurons. Each data-point is labeled with a marker designating the winning neuron (squares, circles, crosses, and dots). (b) Clustering in the second layer, consisting of 2 RBF neurons. Again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

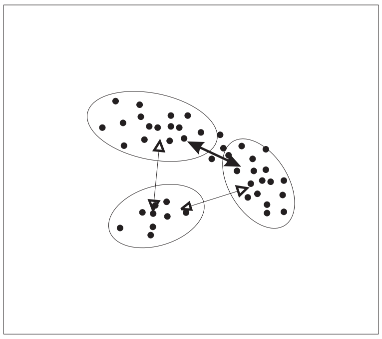

Fig. 7. Weights connecting different part of a single cluster. Given two clusters of data-points (solid circles) classified by three RBF neurons (elliptic receptive fields), data-points between two RBF neurons strengthen the mutual lateral connections (solid arrows), whereas the connections to the equidistant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

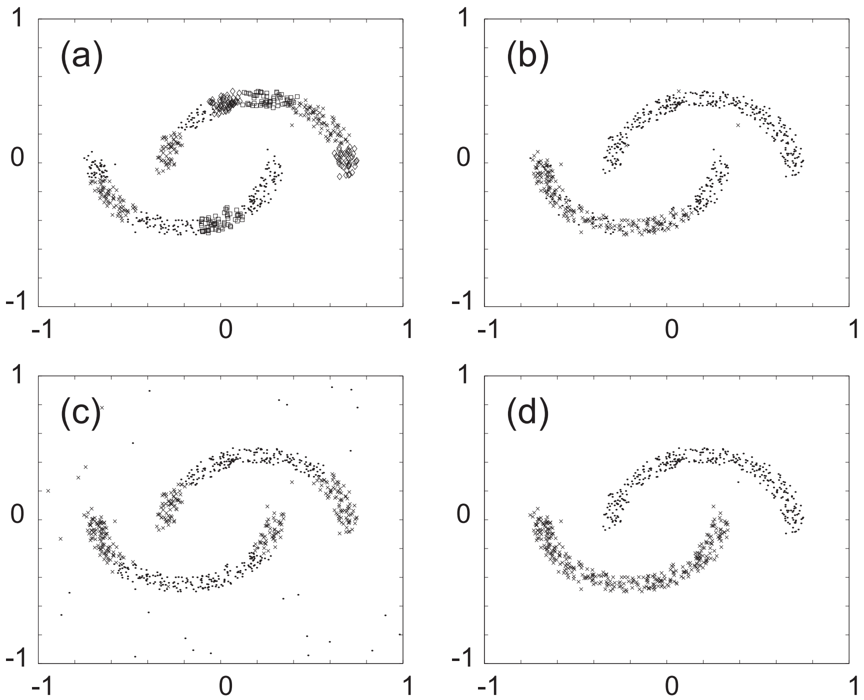

Fig. 8. Clustering of two interlocking clusters in a multi-layer RBF network. (a) classification in the first layer: 11 outputs, the two clusters are spread over respectively 5 (upper cluster) and 6 neurons (lower cluster). The respective classifications are denoted by different markers and gray levels. (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).