LETTER Communicated by Alexander Loskutov

Hebbian Self-Organizing Integrate-and-Fire Networks for Data Clustering

Florian Landis*
flandisf@ethz.ch
Thomas Ott
thomas.ott@zhaw.ch
Ruedi Stoop
Ruedi@ini.phys.ethz.ch
Institute of Neuroinformatics, University of Zurich and ETH Zurich, CH-8057, Zurich, Switzerland

We propose a Hebbian learning-based data clustering algorithm using spiking neurons. The algorithm is capable of distinguishing between clusters and noisy background data and finds an arbitrary number of clusters of arbitrary shape. These properties render the approach particularly useful for visual scene segmentation into arbitrarily shaped homogeneous regions. We present several application examples, and in order to highlight the advantages and the weaknesses of our method, we systematically compare the results with those from standard methods such as the *k*-means and Ward's linkage clustering. The analysis demonstrates that not only the clustering ability of the proposed algorithm is more powerful than those of the two concurrent methods, the time complexity of the method is also more modest than that of its generally used strongest competitor.

1 Introduction

In the analysis of sensory data, we often deal with huge amounts of data. It is then useful to partition such data sets into classes (clusters) of items that share similar properties. This fact is illustrated by the visual scene analysis task, where a large set of individual pixels is segmented into (usually few) objects. The problem of finding appropriate groupings in the data set is referred to as the clustering problem (Jain, Murty, & Flynn, 1999; Xu & Wunsch, 2005). In everyday life, humans cluster sensory data without any perceivable effort. Seen from a neural network perspective, clustering

*Florian Landis is now at the Center for Energy Policy and Economics, ETH Zurich, CH-8032 Zurich, Switzerland, and Thomas Ott is now at the Institute of Applied Simulation, Zurich University of Applied Science, CH-8401, Winterthur, Switzerland.

is the most general formulation of a perception task. Whereas the classical backpropagation network requires a teacher to convey the correct association between input and output, in unsupervised learning, we use prototype examples to which we associate the examples from the data set. In the clustering context, where prototypes are no longer given, we proceed in a fully self-organized way. The prototypes, if desired, could then be derived from the obtained clusters. Simple examples, some of them exhibited in our contributions, however, demonstrate that for general problems, the selection of appropriate prototypes is generally difficult or even impossible.

Despite the existence of a plethora of standard algorithms (Jain et al., 1999), the technical implementation of human clustering skills is still a challenging task. Two classical, and therefore benchmark, approaches are *k*-means (Steinhaus, 1956) and hierarchical clustering (Johnson, 1967; D'Andrade, 1978). In these methods, the number of clusters either has to be prespecified or remains undetermined, which can be a severe drawback. Moreover, both algorithms proceed by assigning each data item to a cluster. Such an assignment is not always appropriate, as we sometimes deal with noisy data, where we have independent singletons that should not be attributed to clusters. Finally, most clustering algorithms bias the shape of the resulting clusters. For example, the clusters found by *k*-means tend to be of spherical shape. It goes without saying that such forms of bias should be avoided since it will generally severely hamper the correct outcome of the clustering process. In the Hebbian learning clustering (HLC) that we introduce here, we avoid most of these problems (see Figure 1). The algorithm is based on dynamic processes occurring in integrate-and-fire networks, making the clusters arise out of the interplay between neural activity and changes in the network connectivity. Our algorithm is inspired by the rich tradition of self-organizing clustering methods, such as the self-organizing map (SOM; see, e.g., Kohonen, 1997, or Samsonova, Kok, & Ijzerman, 2006, for novel developments), as well as by spin system methods, such as the Hopfield network (e.g., Hopfield & Tank, 1986) or superparamagnetic clustering (Blatt, Wiseman, & Domany, 1996, 1997).

Figuratively, our algorithm proceeds by dwelling on slight initial inhomogeneities in the network structure from which the final clusters emerge. The local rule nature of the processing renders it ideal for extracting clusters without shape bias. The algorithm does not require any prior information about the number of clusters and has the capability of recognizing background singletons. In the following section we describe the HLC algorithm and discuss typical results in detail. The algorithm is applied to various test data. By comparing the performance of the algorithm to *k*-means and Ward's linkage method (Ward, 1963), we exhibit both its advantages and potential drawbacks.

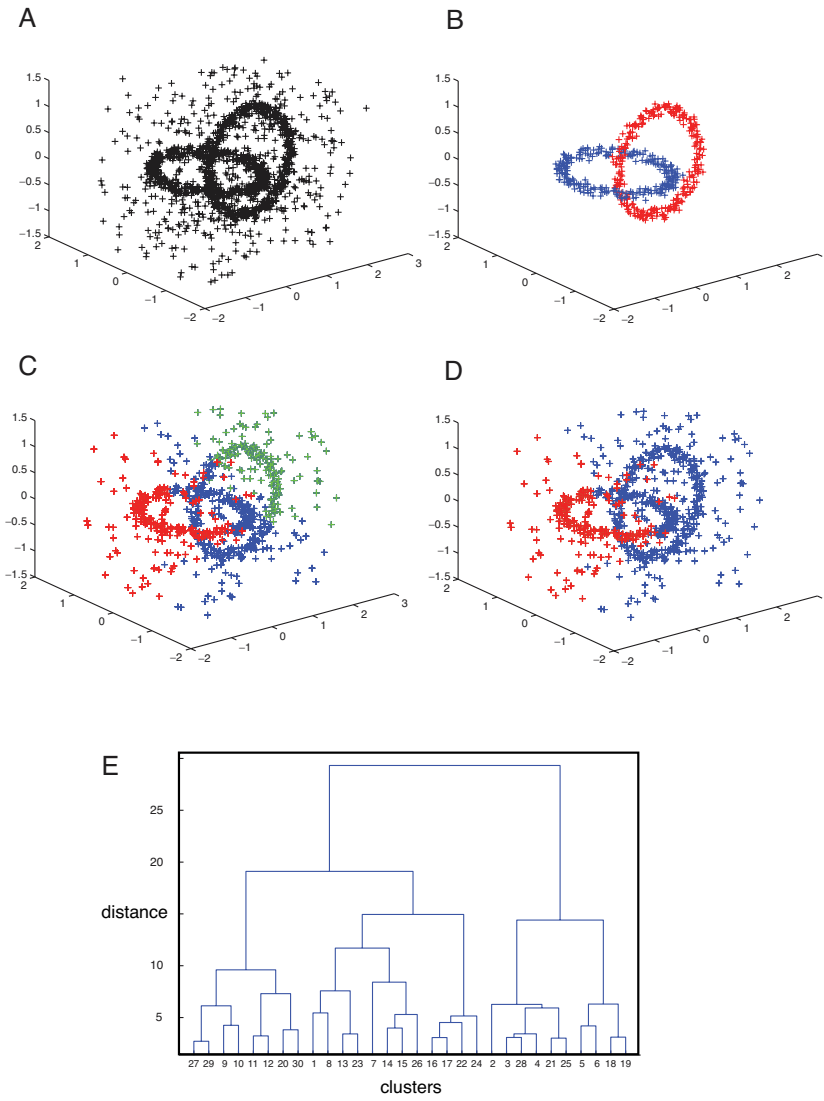

Figure 1: Clustering of two noisy rings in dimension 3 (A), performed by HLC clustering with ease (B). The *k*-means clustering method is inherently unable to perform this task. Ward clustering also runs in severe problems (C–D), even when going to the optimal region of the corresponding dendrogram (E).

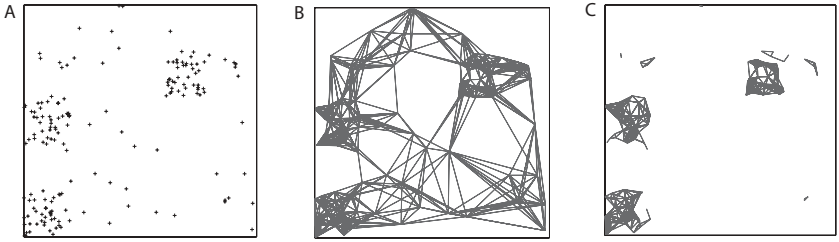

Figure 2: (A) Two-dimensional data distribution with $N = 170$ points, consisting of three clusters and background points. (B) Basic (initial) graph structure, involving the $k_{nn} = 10$ next neighbors. (C) Evolved connectivity graph by HLC, clearly exposing three main clusters and some small accidental structures. Singletons are not reported.

2 Hebbian Clustering HLC

It is convenient to partition our clustering approach into four steps.

In the first step, the network structure that will represent the data needs to be created. Given a data set S with N data items as in Figure 2A, using an appropriate distance function d, the pairwise similarities or distances d_{ij} between items i and j are calculated. For each item, we then determine the k nearest neighbors k_{nn}. By identifying each item with a node and connecting each node with its k nearest neighbors, we obtain the basic graph structure (see Figure 2B). Each node is now identified with an integrate-and-fire (I&F) neuron; each edge stands for a symmetric synaptic connection w_{ij} of initial strength

$$w_{ij} = w_{ji} = e^{-d_{ij}^2/d_0^2}. \tag{2.1}$$

d_0 denotes a local parameter initially chosen proportional to the average distance $\bar{d} =< d_{i,j} >|_{i,j\,\text{connected}}$ between connected nodes. Whereas within the context of Potts-Spin systems that comotivated our approach (see, e.g., Ott et al., 2004; Ott, Kern, Steeb, & Stoop, 2005; Ott & Stoop, 2006), $\bar{d}$ has a precise meaning; the precise value of the proportionality constant embodied in d_0 is of lesser importance. In the context here, it is sufficient to know that a value $\frac{d_0}{\bar{d}} \simeq \frac{1}{4}$ will be fine (for related issues, see section 3).

In the second step, we implement a leaky I&F neuron site dynamics. In order to eventually be able to implement the leaky I&F neurons most naturally and efficiently in hardware, they are modeled here directly as RC circuits, driven by an input current I. Including a leaky term, the basic equation for the electrical potential $u(t)$ of a single I&F neuron therefore is

$$\frac{\partial u(t)}{\partial t} = -\frac{u(t)}{RC} + \frac{I}{C}. \tag{2.2}$$

I can be divided into I_{ext} and I_{int}. I_{ext} is a driving current from an external source. We assume I_{ext} to be constant for all times and neurons. I_{int} is made up of the electric signals (spikes) from the connected neurons. A spike is an instantaneous pulse that is carried from neuron j to neighboring neurons i. In our setting, the strength of the spike will be proportional to the synaptic strength w_{ij}. For unconnected neurons, $I = I_{ext}$, we observe that potentials smaller than $I_{ext}R$, are driven toward the value $I_{ext}R$. At a threshold value $\theta < I_{ext}R$, however, u is reset: at times t^k defined by $u(t^k) = \theta$, the neuron emits a spike, and u is set to 0. In this way, we obtain the network-specific I&F equations,

$$\frac{\partial u_i(t)}{\partial t} = -\frac{u_i(t)}{RC} + \frac{I_{ext}}{C} + \overbrace{\sum_{t_j^k,\ j\in N_i} w_{ij}\delta\left(t - t_j^k\right)}^{I_{int,i}} \cdot \frac{1}{C}, \tag{2.3}$$

where N_i, $i = 1, \ldots, N$, is the set of indices of neurons connected to neuron i.

Equation 2.2 allows the exact integration of the action potentials between spikes: Given the potential $u_i(t_0)$ at t_0, the time until $u_i(t)$ reaches the threshold θ can be expressed as

$$T_i(t_0) = RC \ln \frac{u_i(t_0) - I_{ext}R}{\Theta - I_{ext}R}. \tag{2.4}$$

The time until the next spike in the network occurs can therefore be calculated as $T_k = \min_i\{T_i\}$, where k indexes the neuron that fires next. The action potentials of all the neurons are then updated according to

$$u_i(t') = I_{ext}R\left(1 - \exp\left(\frac{-T_k}{RC}\right)\right) + u_i(t)\exp\left(\frac{-T_k}{RC}\right), \tag{2.5}$$

where $t' = t + T_k$. Instant spike emission and transmission are finally calculated according to

$$u_i(t') \to 0 \quad \text{and} \quad i \in \mathcal{F}(t'), \qquad \text{if } u_i(t') > \theta, \tag{2.6}$$

$$u_i(t') \to u_i(t') + \sum_{j\in\mathcal{F}(t')} w_{ij}R \quad \text{otherwise}, \tag{2.7}$$

where $\mathcal{F}(t')$ is the set of all neurons that fire at time t'. Note that the updates 2.6 and 2.7 have to be repeated as long as equation 2.7 produces action potentials that exceed the threshold θ. Every neuron, however, can spike only once at $t = t'$, where $u(t') = 0$.

Finally, the synapse dynamics needs to be defined. Two antagonistic mechanisms dynamically determine the synaptic strength. Synapses between neurons that tend to fire synchronously are strengthened likewise to the Hebbian learning rule. To have a practical criterion for synchronicity, a time window τ is defined. Learning is then based on the following simple rule: If the time difference between spikes of two connected neurons i and j is smaller than τ, the two spikes are considered to be co-instantaneous, and the synaptic weight $w_{ij} = w_{ji}$ is doubled. A maximal synaptic weight is implemented by means of a cutoff threshold at $w_{ij} = 1$.

This simple rule gives rise to a self-organization and self-amplification mechanism by means of which clusters defined as ensembles of synchronously firing neurons emerge. For preventing uncontrolled strengthening of the synaptic connections, a constant decay of the synaptic weights is implemented as well. This ingredient also prevents weakly connected background neurons from clustering and supports the emergence of well-defined clusters. A balance between increasing and decreasing weights is achieved by means of the decay law,

$$w_{ij}(t) = w_{ij}(0) \cdot 2^{\frac{-2\tau}{T_{ext}^2}t}, \tag{2.8}$$

where

$$T_{ext} = RC \ln \frac{-I_{ext}R}{\Theta - I_{ext}R} \tag{2.9}$$

is the firing period of unconnected neurons.

The initial weight distribution of the network contains weights of different sizes that are dynamically adjusted according to the laws described above. The network simulation can be stopped once the dynamics have become stationary. Weights within a cluster will then be close to $w_{ij} = 1$, as the small loss in weight due to the decay is compensated by the Hebbian learning, whereas the weights of neurons that do not participate in a cluster will approach zero. As a consequence, synchronization clusters can be identified as components of the connectivity graph that have the maximal number of internal connections. The learning process can therefore be considered to have come to an end once very few connection weights w with $s_{\min} < w < 1$ remain. In practice, by means of accidental learning, some weights may cross from $[0, s_{\min}[$ to $[s_{\min}, 1]$. In the long run, this could lead to unwanted maximal connection strengths all over the network. It is therefore advisable to stop the learning process soon after the learning in the highly connected groups of neurons has terminated, before the singletons cluster. This can be achieved by observing the ratio

$$r_{learn} = n_{learn}/n_1,$$

where $n_{learn} = |\{w_{ij} | s_{\min} < w_{ij} < 1\}|$ and $n_1 = |\{w_{ij} | w_{ij} = 1\}|$. This ratio compares the present learning activity (number of neurons engaged in

learning) versus the already accomplished learning processes. Obviously, after r_{learn} has crossed an empirical learning threshold r_Θ from above, the learning can be stopped. The precise value of the empirical learning threshold is not too critical (for a numerical value, see section 3). Arguments as to why this is the case and why there are no learning oscillations can be found in Ott et al. (2004).

For the identification of the clusters after termination, all connections with a weight smaller than $\Theta = 0.8$ are canceled, so that the connectivity graph subdivides into a number of strongly connected components that represent the clusters (see Figure 2C). The exact choice of the threshold Θ is not critical, as most synaptic weights are either close to zero or close to one, indicating a robust self-organizing process at work.

3 Choice of Parameters and Initial Conditions

For the I&F neuron parameters, we chose biologically inspired values $I_{ext}R = 25$ mV, $\theta = 16$ mV, $RC = 8$ ms, and $R = 1$ MΩ. From this, the time T it takes the action potential to rise from resting potential $u_r = 0$ to the threshold value θ in the absence of input from its next neighbors can be measured. Another measurable quantity is the average distance between two connected neurons, $\bar{d}$, which depends on the data to be clustered. When choosing the parameters d_0 and τ, it should be kept in mind that larger values entail quicker synchronization. Our standard choices of these parameters were $\tau = \frac{T}{4}$ and $d_0 = \frac{\bar{d}}{4}$. Throughout, the learning threshold was set to $r_\Theta = 0.1$. The neuron neighborhood order was $k_{nn} = 10$ if it was not specified differently by the nature of the data. These values are motivated by simulations of toy models, and some of them have been corroborated by analytical arguments (Ott et al., 2004, 2005).

4 Applications

With the simple application shown in Figure 1, we demonstrated that a clustering task that inherently cannot be solved by most other clustering algorithms is solved by the proposed approach with ease. Note that no supervised learning has taken place, yet the individualities of the two rings have been worked out perfectly.

For several benchmark problems, we compared the proposed algorithm with the k-means and the Ward clustering (Ward, 1963). The quality of performance is usually determined by comparing an obtained clustering c_f to a given optimal clustering c_g and by measuring the run time needed for completing the process. For the first comparison, we evaluated the following quantities: The Jaccard index (Wild & Blankley, 2000),

$$J(c) = \frac{a(c_f, c_g)}{a(c_f, c_g) + e(c_f, c_g) + e(c_g, c_f)}, \tag{4.1}$$

measures how similar two clusterings c_f, c_g are (the "goodness of clustering" if the comparison to the perfect clustering can be made). Function $a(c_1, c_2)$ counts the pairs of data that are in the same cluster in clustering c_1 and c_2, whereas function $e(c_1, c_2)$ counts the pairs of data that are in the same cluster in the clustering c_2, but not in c_1. In this way, the Jaccard index reports the fraction of pairs that cluster together in both clusterings c_f and c_g, relative to those pairs that cluster together in at least one clustering. Therefore, if the clusterings have a relatively similar pair membership, the Jaccard index should be close to one.

Measures that concentrate on the errorness of a clustering are

$$E_1 = \frac{e(c_f, c_g)}{\max_{c \in C} e(c, c_g)}, \tag{4.2}$$

and, similarly,

$$E_2 = \frac{e(c_g, c_f)}{\max_{c \in C} e(c_g, c)}. \tag{4.3}$$

They are based on the count of pairs of data items that were put into the same cluster, although in the ideal clustering, they are in different classes, scaled by the worst possible case. Therefore, for a decent clustering, these values should be close to zero.

4.1 Example 1: Two-Dimensional Toy Set. As an introductory example and an illustration of the first beneficial property of HLC, we discuss the results of our clustering for the simple two-dimensional problem shown in Figure 2A. Figure 2B shows the final strong connectivity components provided by HLC, which reflect the detected clusters. Clearly, three main clusters are correctly identified. Moreover, HLC discriminates between actual clusters and background points. The background distribution yields small, temporally unstable clusters, whereas the remaining points are classified as singletons.

4.2 Example 2: Ring with Central Cluster. The second beneficial property of HLC is demonstrated in this example. Many clustering methods implicitly presume a certain cluster shape. Consequently, it is particularly hard for them to solve problems involving clusters that do not conform to the favored geometry. As an example, *k*-means, favoring spherical clusters, fails to detect the ring cluster depicted in Figure 3. HLC, however, easily finds an optimal solution. For a summary of the obtained results, clearly exhibiting this feature, see Table 1.

4.3 Example 3: Iris Data Set. The Iris data set (Anderson, 1935) includes measurements of 150 plants, dividing into 50 specimens of the three species

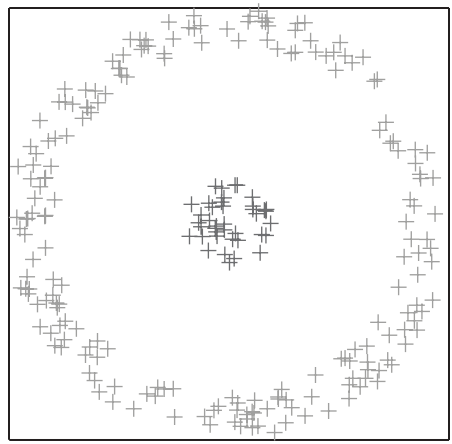

Figure 3: Circle surrounded by another circle. The fact that the two clusters cannot be described by detached spherical shapes makes this a difficult problem to solve for many clustering methods. HLC has no problem with this situation.

Table 1: Performance of the Different Algorithms for the Data of Figure 3.

Ring Cluster	HLC	k-Means	Ward Clustering
E_1	0.000	0.500	0.483
E_2	0	0.502	0.480
Jaccard index	0.990	0.419	0.436
Time [s]	0.917	0.0043	0.0473

Iris setosa, *Iris versicolor*, and *Iris virginica*. For each plant, the data set provides petal width, petal length, sepal width, and sepal length (Anderson, 1935). Figure 4 illustrates the data set by means of two two-dimensional projections. In the picture on the left, it is clearly visible how the data for the *Iris setosa* form a clearly distinct cluster at the left bottom, whereas the clusters of *Iris virginica* and *Iris versicolor* overlap and cannot be separated by a straight line in either of the two projections. Nonetheless, the nonconvex forms of the clusters can be expected to enhance the performance of k-means and Ward clustering. For this data set, the results obtained by the different clustering methods are shown in Table 2. For k-means and Ward's method, the correct number of clusters had been given as a prior information. As the clusters of *Iris virginica* and *Iris versicolor* partially overlap, HLC exhibits a relatively large error of the type highlighted by E_2. In terms of the Jaccard index, the algorithm does not do much worse than the other algorithms, although the number of clusters is not provided. The a priori knowledge of the numbers of clusters would significantly reduce the risk

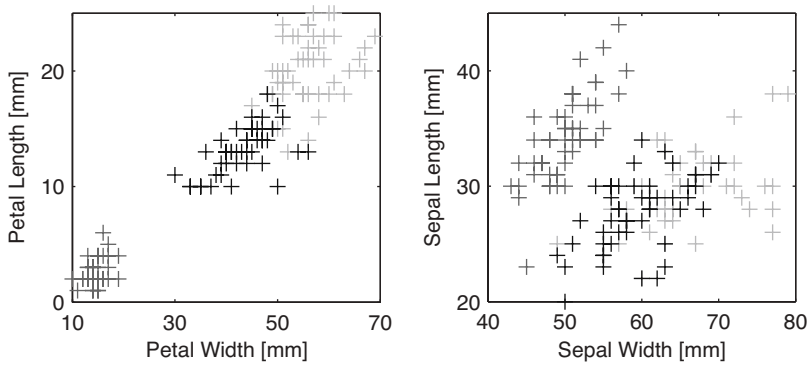

Figure 4: Iris data set. (Left) The distribution with respect to petal width and petal length. (Right) The coordinates are sepal width and sepal length.

Table 2: Performance of Different Algorithms on the Iris Data Set.

Iris Data Set	HLC	k-Means	Ward Clustering
E_1	0.097	0.198	0.176
E_2	0.243	0.108	0.102
Jaccard index	0.552	0.657	0.682
Time [s]	0.511	0.0026	0.0147

for errors of type 2, since in the present setting, they originate from binding the two overlapping species into one cluster. Using the a priori information, k-means and Ward clustering deliberately separate this cluster into two clusters. Although a classical example of clustering, from the projections, it could be disputed whether the iris data set is a good clustering data set at all, given the characteristics used for characterization of the data items.

4.4 Example 4: Scene Segmentation. An important and advanced task for clustering algorithms is the analysis of visual scenes (Gdalyahu, Weinshall, & Werman, 2001). In the following, we report on the clustering analysis of two two-dimensional natural images, given as a 50×50 array of pixels. This low resolution has been chosen for demonstration, although due to its local coupling between neighboring pixels, it affects HLC unfavorably. For HLC, the mapping of a pixel image to a neural network is done by identifying each pixel with a neuron and connecting two neurons if their respective pixels lie adjacent to each other. The similarity measure d_{ij} is chosen to be the Euclidean distance in the RGB color space. For performance evaluation, HLC was compared to k-means. In order to circumvent the bias of k-means to convex shapes, for k-means the pixels were directly

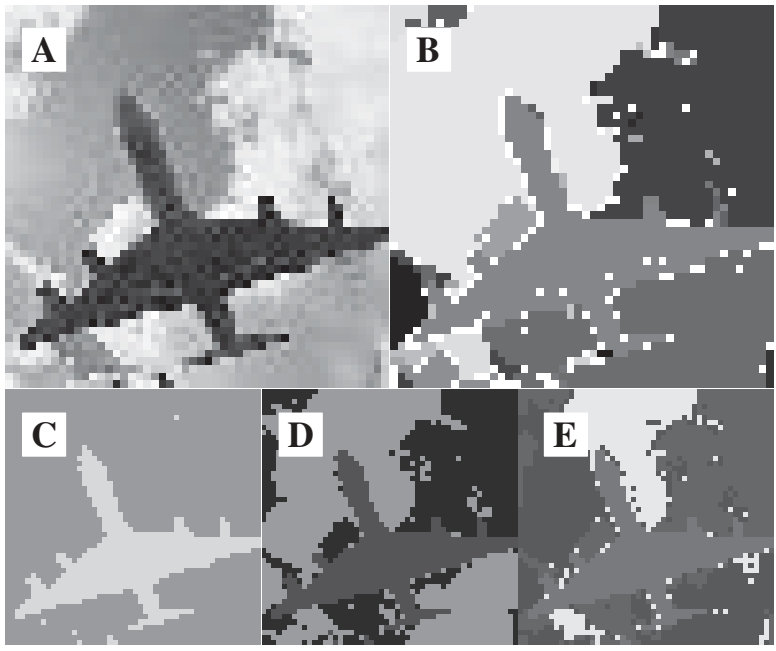

Figure 5: (A) Initial problem. (B) HLC solution. (C–E) *k*-means solution in color space (with $k = 2, 3, 4$, respectively), which circumvents the *k*-means convexity bias. See the text.

translated into points in RGB color space, that is, spatial information of the pixels was not taken into account (Bishop, 2006). As no well-defined optimal clustering solution was determined for the images, the Jaccard index was not evaluated. Instead, we will rely on a visual assessment of the results and provide a subsequent qualitative discussion exhibiting the beneficial properties of HLC even in this case.

A plane in a cloudy sky (see Figure 5A) is the input to our first scene segmentation task. The HLC results displayed in Figure 5B demonstrate that the shape of the plane is reproduced fairly well. Fractions of the clouds are detected as single objects, as they are separated by the plane. Figures 5C to 5E show the solutions for *k*-means clustering with $k = 2, 3, 4$, respectively. $k = 2$ yields a perfect shape of the plane against the sky but fails to distinguish between sky and clouds. The solution for $k = 3$ achieves this distinction. Parts of certain clouds, however, are now attributed to the sky. Setting $k = 4$ finally results in partitioning the sky above the plane into two parts. *k*-means clustering uses the a priori information (which may hold or may not be true) that pixels that have the same color belong to the same object. Our algorithm is more prudent. By repeating the basic clustering

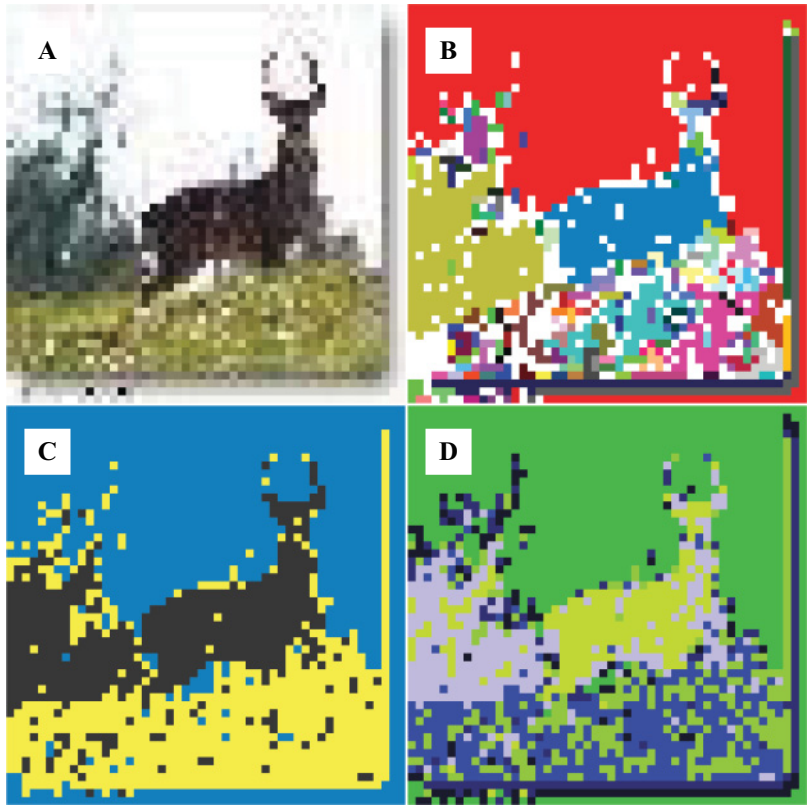

Figure 6: (A) Initial problem. (B) *HLC* solution. (C–D) *k*-means solution in color space, with $k = 3, 7$, respectively.

procedure on the achieved clustering or by using a coarse graining of the average colors of the obtained clusters, we would effortlessly come up with the optimal solution of *k*-means.

A scene with a stag in a meadow and a bush in the background (see Figure 6A) provides a second illustration of this observation. The three biggest clusters suggested by HLC (see Figure 6B) correspond the sky, the body of the deer, and the bush, respectively. Whereas *k*-means with $k = 3$ (see Figure 6C) appears to yield the most accurate picture of the body, it should be noted, however, that the bush is not separated from the deer. Increasing *k* does not solve this problem; before being recognized as two different objects, the deer body and the bush have already split into subclusters (see Figure 6D). Whereas due to its spatial nature, HLC is much more affected by the low resolution of the original picture than *k*-means, the example illustrates well the basic observation that for *k*-means, it is difficult to detect the coarsest unbiased clustering of a scene.

5 Time Performance

We find that the time complexity of the dynamics of HLC is $\mathcal{O}(N^2)$, where N is the size of the data file: Performing the spike updates is of order $O(N)$, and the subsequent checks for spike coincidences are of order $O(N^2)$. In terms of equations, the essential loop the program works sequentially through are equations 2.4 to 2.7, which implies $\mathcal{O}(N)$ (see 2.4, times calculation), $\mathcal{O}(N)$ (see equation 2.5, potentials calculation), and $\mathcal{O}(N^2)$ (see equations 2.6 and 2.7, spike coincidence), respectively. The learning and the stopping criterion are of local nature and sequentially implemented in the code. Locally, each neuron's horizon extends only to its k nearest neighbors, whereas checking for the termination of the process is of $\mathcal{O}(N)$. This analysis is corroborated by the observation that at a site, the number of time-update steps until learning termination is found to be insensitive to the size N of the data set. These observations imply that in terms of time complexity, HLC can compete with the well-established hierarchical Ward clustering even if the results in section 4 (see Tables 1 and 2) suggest a weaker time performance of HLC. Systematic trials with test files of the kind depicted in Figure 2 show, however, that for large test files, our nonoptimized implementation of HLC outperforms the Matlab implementation of Ward clustering (see Figure 7).

6 Conclusion

We presented the rationale, implementation, and results of a novel Hebbian learning spiking neural network clustering approach. The robust, locally mediated, self-organizing design makes it a genuinely nonparametric approach. As a consequence, unlike many standard methods such as k-means clustering, the algorithm does not require any information about the number or shape of clusters that it is supposed to find. This makes it a good choice for clustering problems where little a priori information on the nature of the clusters to be worked out is available. This is the case, as an important application field, in image segmentation. We demonstrated that our algorithm performs substantially better in the presence of nonconvex clusters. In these cases, frequently found in image segmentation, HLC generally comes up with useful solutions, while k-means intrinsically fails. Since our algorithm is able to separate clusters from background singletons, it is also more susceptible to density variations within the clusters to be found. This behavior is present in the chemical data investigated in detail in Ott et al. (2004), where we found an extremely strong cluster density variation. For these data, our algorithm cannot cope with our previously designed sequential superparamagnetic clustering SSC (Ott et al., 2004). The proposed algorithm, however, performed much better than k-means and better than Ward with automated Kelly criterion (Jaccard indices of 0.61 (HLC) versus 0.50 (Ward) for ISIS binary key and Euclidean distance

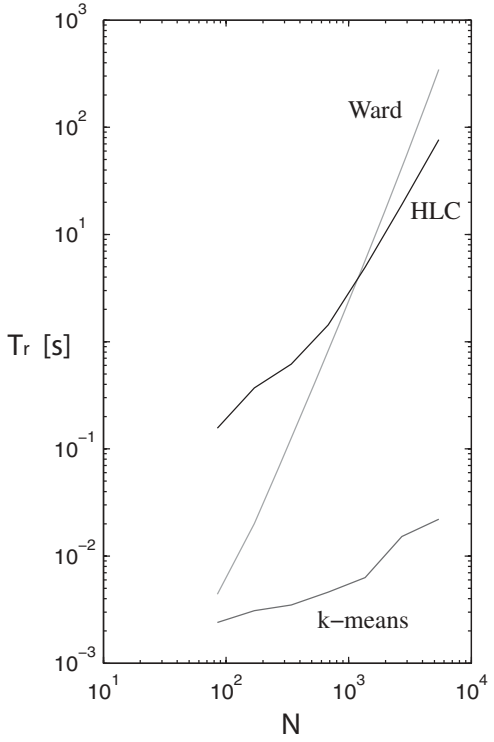

Figure 7: Run time T_r of *HLC*, of Ward's method and of *k*-means clustering as a function of the data size N.

(Ott et al., 2004)). We found earlier (Ott et al., 2004) that a non-Kelley, hand-sorted, fully problem-optimized Ward clustering was able to push the Jaccard index up to 0.66 for these data. We expect that a similar optimization procedure would also improve the performance of HLC. The comparison with the superparamagnetic clustering SC (Blatt et al., 1996), which is not sophisticated for the density issue, indicates that even with this tuning, the performance of SSC will not be attained (Jaccard index of 0.66 (SC) versus Jaccard index of 0.96 (SSC)). In order to achieve this, HLC would require a sequential sophistication, similar to the one used in SSC, which is possible only at the cost of significantly decreased performance. As it simulates a rather detailed learning process, for small data sets, the computation time of HLC lags somewhat behind those of the other two standard methods. For larger data sets, however, the Matlab implementation of our algorithm gains on Matlab's hierarchical clustering implementation, whereas *k*-means has the best immanent time performance (yielding, however, generally the worst results). We emphasize that our algorithm does

not yet take advantage of the parallel nature of the learning processes. A hardware implementation on a chip exploiting the close-to-hardware nature of the fundamental elements, as well as the highly local and parallel learning principles of HLC, will improve the time performance considerably. In this way, the time and computational resources that algorithms with similar beneficial features require (Blatt et al., 1996; Ott et al., 2004, 2005; Ott & Stoop, 2006), are strongly reduced by HLC, which will make many interesting problems and situations amenable to clustering-based sensory perception.

What clustering algorithm to choose for an application obviously depends a great deal on the nature of the data to be clustered. If the data are likely to contain nonconvex clusters and residual background items and if there are computation time constraints, the algorithm presented here may be an optimal candidate.

References

Anderson, E. (1935). The irises of the Gaspe peninsula. *Bulletin of the American Iris Society*, *59*, 2–5.

Bishop, C. M. (2006). *Pattern recognition and machine learning.* New York: Springer.

Blatt, M., Wiseman, S., & Domany, E. (1996). Superparamagnetic clustering of data. *Phys. Rev. Lett.*, *76*, 3251–3254.

Blatt, M., Wiseman, S., & Domany, E. (1997). Data clustering using a model granular magnet. *Neural Computation*, *9*, 1805–1842.

D'Andrade, R. (1978). U-statistic hierarchical clustering. *Psychometrika*, *4*, 58–67.

Gdalyahu, Y., Weinshall, D., & Werman, M. (2001). Self-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Trans. Pattern Anal. Mach. Intell.*, *23*, 1053–1074.

Hopfield, J. J., & Tank, D. W. (1986). Computing with neural circuits: A model. *Science*, *233*, 625–633.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, *31*, 275–279.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, *2*, 241–254.

Kohonen, T. (1997). *Self-organizing maps.* New York: Springer-Verlag.

Ott, T., & Stoop, R. (2006). Benefits and pitfalls of belief propagation-mediated superparamagnetic clustering. *Phys. Rev. E*, *74*(4), 042103.

Ott, T., Kern, A., Schuffenhauer, A., Popov, M., Acklin, P., Jacoby, E., & Stoop, R. (2004). Sequential superparamagnetic clustering for unbiased classification of high-dimensional chemical data. *J. Chem. Inf. Comp. Sci.*, *44*, 1358–1364.

Ott, T., Kern, A., Steeb, W.-H., & Stoop, R. (2005). Sequential clustering: Tracking down the most natural clusters. *Journal of Statistical Mechanics: Theory and Experiment*, P11014.

Samsonova, E. V., Kok, J. N., & Ijzerman, A.D. (2006). TreeSOM: Cluster analysis in the self-organizing map. *Neural Networks*, *19*(6–7), 935–949.

Steinhaus, H. (1956). Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci.*, *4*, 801–804.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *J. Am. Stat. Assoc.*, *58*, 236–244.

Wild, D. J., & Blankley, C. J. (2000). Comparison of 2D fingerprint types and hierarchy level selection methods for structural grouping using Wards clustering. *J. Chem. Inf. Comput. Sci.*, *40*, 155–162.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Trans. Neural Netw.*, *16*, 645–678.

Received December 18, 2008; accepted April 25, 2009.