A new clustering algorithm applying a hierarchical method neural network

Javier Bajo, Juan F. De Paz[*], Sara Rodríguez and Angélica González

Department of Computer Science and Automation, University of Salamanca Plaza de la Merced s/n, 37008, Salamanca, Spain

Abstract

Clustering is a branch of multivariate analysis that is used to create groups of data. Most of the existing clustering techniques require defining additional information, including the actual number of clusters, before they can be carried out. This article presents a novel neural network that is capable of creating groups by using a combination of hierarchical clustering and self-organizing maps, without requiring the number of existing clusters to be specified beforehand. The self-organized cluster automatic detection neural network is described in detail, focusing on the density, the average distance, the division algorithm, the update algorithm and the training phase. Three case studies have been carried out in this research in order to evaluate the performance of the neural network, and the results obtained are presented within this article.

Keywords: Clustering, self-organized maps, hierarchical clustering, partition around medoids, dendrogram.

1 Introduction

The techniques used for creating clusters vary according to the type of problem at hand. Cluster analysis is a branch of multivariate statistical analysis that is used for detecting patterns in the classification of elements. Cluster analysis is used in a wide variety of fields [5, 8] including bioinformatics [6, 18] and surveillance [4, 11]. The most common clustering techniques used with bioinformatics are either hierarchical or based on minimizing objective functions because of the ease in performing calculations. The methods used for clustering differ considerably according to the type of data and the amount of available information [7]. Clustering techniques are typically broken down into the following categories: [18] hierarchical, which include dendrograms [19], agnes [13], Diana [13], Clara [13]; neural networks [16, 17] such as self-organized maps (SOMs) [14], neural gas (NG) [15], growing cell structure (GCS) [9], enhanced self-organizing incremental neural network (ESOINN) [10, 20]; methods based on minimizing objective functions, such as k-means [12] and partition around medoids (PAM) [13]; or probabilistic-based models such as expectation-maximization (EM) [22] and fanny [13].

Cluster methods are generally aimed at minimizing the distance that exists between the individuals and the groups. For certain algorithms, this assumes the need to either establish the number of clusters beforehand, or set the number once the algorithm has been completed. This is a problem requiring novel solutions, since it is necessary to provide methods to automatically determine the number of clusters. In certain cases, neural networks allow

[*]E-mail: fcofds@usal.es

Vol. 19 No. 2, © The Author 2010. Published by Oxford University Press. All rights reserved.
For Permissions, please email: journals.permissions@oup.com
doi:10.1093/jigpal/jzq030 Advance Access published 12 July 2010

the number of clusters to be selected automatically based on the existing elements. The networks typically require a previous adaptation phase for the neurons and the initial data that generates the connections among the neurons. Some neural networks may also require establishing the level of connectivity for the neurons beforehand.

The study presented in this article focuses on obtaining a method that allows data to be grouped automatically, without having to specify the number of existing clusters. Our approach is the new self-organized cluster automatic detection neural network (SOCADNN), a self-organized neural network that incorporates a minimal tree. The SOCADNN incorporates algorithms to detect low-density zones and graph theory procedures to establish a connection between elements, which would allow connections to be established dynamically. One of the advantages of the SOCADNN is that it eliminates the expansion phase of a NG to adjust to the data surface before dividing and interconnecting the neurons, thus avoiding one of the most costly phases of the algorithm. Additionally, the connections would continue to adapt throughout the learning process, reducing the high-density neuron areas and separating them from the low-density areas.

The neural network presented within this article is a hybrid approach that integrates techniques from hierarchical and density-based models that allow the grouping and division of clusters according to the changes in the densities that are detected. The hierarchical process is based on the Kruskal algorithm, which creates a minimum spanning tree containing data for the problem at hand. Based on the information obtained from the minimum spanning tree, low-density areas are detected by using a distance matrix for each cluster. The low-density areas will allow the clusters to be separated iteratively. Furthermore, the minimum spanning tree determines the network structure and connections so that learning can take place according to the tree's distribution.

This article is divided as follows: Section 2 describes related studies, Section 3 describes the SOCADNN proposed within the framework of this study, and Sections 4 and 5 present the results and conclusions obtained after evaluating the SOCADNN in two case studies.

2 Related studies

The most common clustering techniques used with bioinformatics are either hierarchical or based on minimizing objective functions. Hierarchical methods such as dendrograms [19] do not require a number of clusters up front, since they use a graphical representation to determine the number. Partition-based methods, which optimize specific objective functions, have the disadvantage of requiring the number of clusters up front [22]. The k-means algorithm present problems with atypical points. The PAM method resolves this problem by assigning an existing element as the centroid. Methods that are either hierarchical or minimize specific objective functions present certain deficiencies when it comes to recognizing groupings of individuals. Artificial neural networks [17, 21] can adapt to the data surface, although they usually require additional time to do so. The SOM [14], have variants of learning methods that base their behaviour on methods similar to the NG [15]. They create a mesh that is adjusted automatically to a specific area. The greatest disadvantage, however, is that both the number of neurons that are distributed over the surface and the degree of proximity are set beforehand. GCS [9] do not set the number of neurons or the degree of connectivity, but they do establish the dimensionality of each mesh. This complicates the separation phase between groups once it is distributed evenly across the surface. There are other artificial neural network (ANN) such as SOINN [20] and ESOINN [10]. Unlike the SOINN, ESOINN consists of a single layer, so it is not necessary to determine the manner in which the training of the first layer changes to the second. Nevertheless, with the ESOINN network it is necessary to adjust the neurons to the surface for the data that needs to be grouped. To this end, a phase is required to adjust the previous division of the neurons. The ART networks can be considered as an alternative. They are unsupervised learning networks that facilitate the automatic detection of clusters and, in their latest versions, allow the incorporation of continuous patterns. The major disadvantage of these networks is the selection of the monitoring parameter [1] to determine the number of clusters. Another disadvantage is that the knowledge extraction is more complicated than in mesh-based networks, so learning is less evident.

3 SOCADNN

This study proposes the SOCADNN, which can detect the number of existing groups or classes and, by using the Kruskal algorithm [3], create clusters based on the connections taken from the minimum spanning tree. As opposed to the ESOINN or GCS network, the SOCADNN does not distinguish between the original data and the neurons—during the initial training phase, the latter correspond to the position for each element. This makes it possible to eliminate the expansion phase for a NG to adjust to the surface. However, this step can be applied in situations where the number of elements for carrying out the clustering process needs to be reduced. As each neuron is updated, it can draw closer to neighbouring neurons, thus facilitating the detection of clusters and the separation from other elements. The learning phase for the network is illustrated in Figure 1: the main loop of the learning

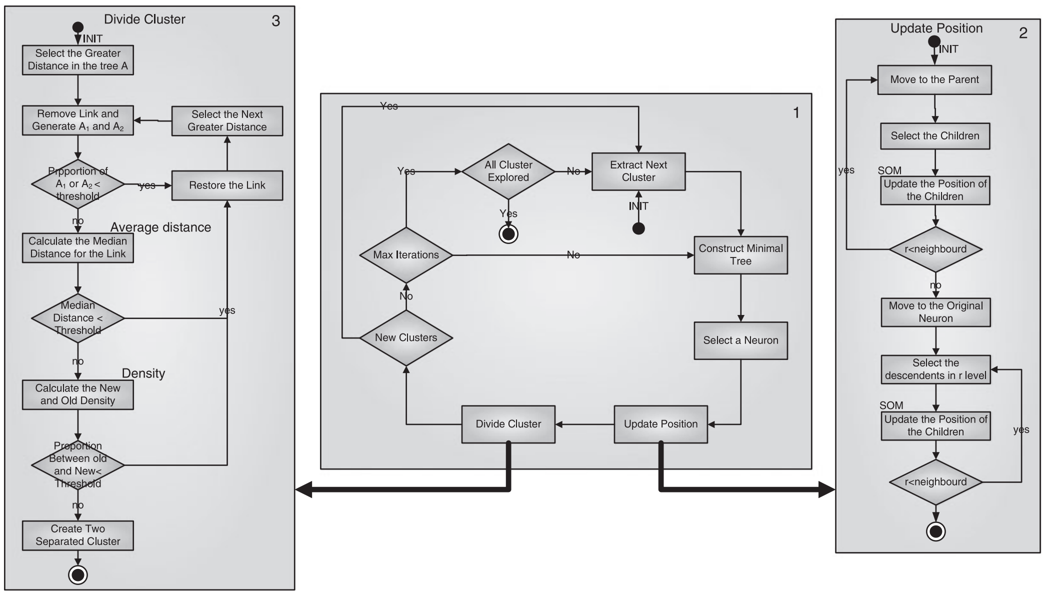
Divide Cluster
3
INIT
Select the Greater Distance in the tree A
Remove Link and Generate A_1 and A_2
Select the Next Greater Distance
Proportion of A_1 or A_2 < threshold
yes
Restore the Link
no
Average distance
Calculate the Median Distance for the Link
Median Distance < Threshold
yes
no
Density
Calculate the New and Old Density
Proportion Between old and New< Threshold
no
Create Two Separated Cluster
1
Yes
All Cluster Explored
No
Extract Next Cluster
Yes
INIT
Max Iterations
No
Construct Minimal Tree
No
New Clusters
Select a Neuron
Divide Cluster
Update Position
Update Position
2
INIT
Move to the Parent
Select the Children
yes
SOM
Update the Position of the Children
r<neighbourd
no
Move to the Original Neuron
Select the descendents in r level
SOM
Update the Position of the Children
yes
r<neighbourd

FIG. 1. Algorithm for the SOCADNN.

phase shown in the centre represents the selection loop for the neurons and establishes the links for the subsequent updating and division of meshes; block ‘Update position’ is the algorithm that updates the position of the selected neuron and the neighbouring neurons connected by the minimal tree; and block ‘Divide Cluster’ is in charge of locating the low-density areas and separating the clusters in order to create new groups.

The following sections describe the steps that are carried out during the learning phase for the ANN. The nomenclature defines T as the set of neurons to be classified, A as the minimum spanning tree that contains all of the nodes from T where matrix C defines the connections between the nodes where element $c_{ij} = 1$ if node $i \epsilon T$ is connected with element $j \epsilon T$, D the distance matrix for T.

3.1 Average distance: block 3

Selecting the links for finding low-density areas can be done by considering the distance of one neuron with respect to its parent in tree A, and the average distance surrounding the neuron. The calculation of the latter distance is based on the distance that exists for each link of the subtree, where the depth is equal to the surrounding distance and centred on the neuron in question, and the number of neurons that exist in the subtree. Figure 2 illustrates the subtree, highlighted in the thicker lines to indicate the neuron that falls within 2 links. The root node of the tree is shown in red, while blue indicates the central node where the average distance with which it is related to the subtree will be calculated. The arrows represent the direction and magnitude of the position of the contiguous neurons. The magnitude depends on both the number of intermediate nodes to arrive at the blue node, and the respective distances.

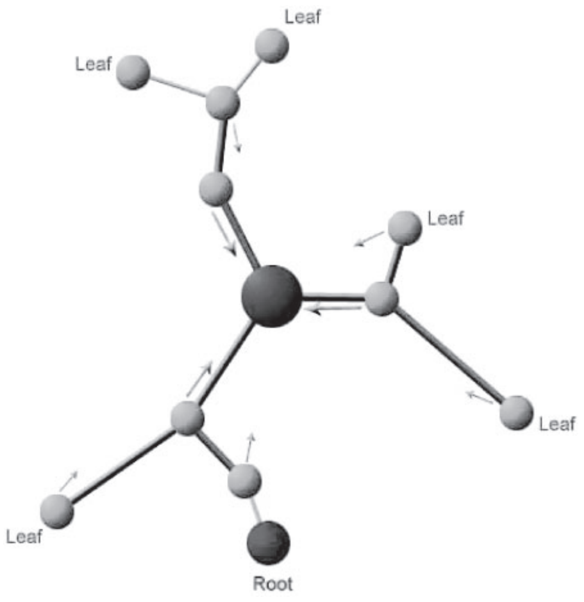

FIG. 2. Subtree for neuron falling within 2 links.

The algorithm is described as follows:

1. Given a_i is the neuron for the tree for which the average distance needs to be calculated, with $i \epsilon T$ where $f^p(a_i)$ is the function that determines the parent node for a_i, which is defined by
$$\begin{aligned} f^p:\ & A \to A \\ & a_i \to f^p(a_i) = a_s \end{aligned} \qquad \text{Where } c_{si}=1 \text{ and } c_{si} \in C$$
2. Apply f^p recursively and select the root node a_r from subtree $a_r = \overbrace{(f^p \circ \ldots \circ f^p)}^{e}(a_i)$
3. Establish the group of nodes within the surrounding area of a_i $A^e_{a_i} \subseteq A$ as follows: $A^e_{a_i} = \{a_j \in A / \exists r \in \mathrm{N}, a_r = \overbrace{(f^p \circ \ldots \circ f^p)}^{r \leq 2e}(a_j)\}$
4. Calculate the average distance for the node $a_i f^m(A^e_{a_i}, D) = \frac{\sum_{a_j \in A^e_{a_i}} d_{sj} \cdot c_{sj}}{\# A^e_{a_i}}$

3.2 Density: block 3

One of the main problems when assigning individuals into groups is by knowing which divisions cause a significant rise in the density of the resulting clusters. ANN such as SOINN or ESOINN studies the length of the links in order to determine if the length is different within the subgroup for each individual. This process requires the creation of subclasses within each cluster, which is done by using a set of functions that determines the threshold on which the creation of the subclasses is based. The SOCADNN searches for cut-off points in areas that produce a significant rise in density. It does so by using the relationship between the total distance calculated from the distance matrix, and the distance from the minimum spanning tree. The matrix for distances D is calculated by a determined measure of distance. In this case, the Euclidean distance was selected so that it would coincide with the measure used in different techniques. However, other measures of distance have been added to the network. Initially, a definition of f^T independent from C was considered, and the connections between neurons were not taken into account for the distance calculations in this measure. That condition was then modified in order to be able to recognize surfaces that do not contain homogeneous distributions of the data.

1. Distance from tree $f^A(C, D) = \sum_{i,j} d_{ij}$ where $c_{ij} = 1, c_{ij} \in C, d_{ij} \in D$
2. Distance between neurons in the tree $f^T(C, D) = \sum_{s \in S} d(f^p(a_s), a_s) + \sum_{t \in T} d(f^p(a_t), a_t)$ Where
$$s = \{\overbrace{(f^p \circ \ldots \circ f^p)}^{m}(a_s), \overbrace{(f^p \circ \ldots \circ f^p)}^{m-1}(a_s) \ldots, f^p(a_s), a_s\} \text{ with } \#s = m_y$$
$t = \{\overbrace{(f^p \circ \ldots \circ f^p)}^{n}(a_t), \overbrace{(f^p \circ \ldots \circ f^p)}^{n-1}(a_t) \ldots, f^p(a_t), a_t\}$ with $\#t = n$, n and m are selected so that there cannot exist any value for n or m $\overbrace{(f^p \circ \ldots \circ f^p)}^{m}(a_s) = \overbrace{(f^p \circ \ldots \circ f^p)}^{n}(a_s)$
$$d(a_k, a_s) = d_{ks} \text{ with } d_{ks} \in D$$
3. Calculate the final density $f^D(C, D) = f^T(D) / f^A(C, D)$

3.3 Division algorithm: block 3

The division algorithm is responsible for finding the connections between the low-density neurons in order to separate the cluster. It considers the distance between the neurons and the resulting changes in density for the potential divisions. The process is described as follows:

1. Determine the cut-off point for the elements α, and the cut-off points for distance β.
2. Initiate $i = 1$.
3. Select the greatest distance i for $d_{jk} \in D / c_{jk} = 1$ and remove the node from the tree $a_k \in A$.
4. Given A_1 and A_2 are the remaining trees alter eliminating a_k and the connection with the parent node $f^p(a_k)$ where $T_1 = \{s \in T / a_s \in A_1\}$ and $T_2 = \{s \in T / a_s \in A_2\}$ with $T = T_1 \cup T_2$, $T_1 \cap T_2 = \phi$, C_1, C_2, D_1, D_2 for the corresponding link and distance matrixes.
5. If $\#T_1/\#T$ or $\#T_2/\#T$ is less than α go to step 13.
6. Calculate the average distance from the node for the tree a_k following the average distance algorithm $d^m_{a_k} = f^m(A^e_{a_i}, D)$.
7. Determine if the distance from tree node a_k and its parent is less than the average distance $d_{sk} \le d^m_{a_k} \cdot \beta$ where $s \in T$ and $a_s = f^p(a_k)$ go to step 13.
8. The density for T, T_1 and T_2 following the density algorithm $f^D(C, D)$, $f^D(C_1, D_1)$, $f^D(C_2, D_2)$
9. Calculate the new density threshold $\delta(t+1) = f^D(C_1, D_1) + f^D(C_2, D_2)$ and the previous $\delta(t) = f^D(C, D)$
10. If the value $\delta(t)/\delta(t+1) < 1/(\delta(0)/\delta(1) \cdot \rho)$ where ρ is constant, go to 12.
11. Finish
12. Re-establish the connection a_k with its parent node
13. If $i < \#T$ calculate the value of $i = I + 1$ and go to step 2.

3.4 Update algorithm: block 2

The neurons from the network that define the clusters are periodically updated in a way similar to the kohonen SOM. By updating automatically, the positions and connections of the neurons can be readjusted in order complete the division of the clusters. The network randomly selects an initial neuron and brings neighbouring neurons closer in. The neurons are updated according to the hierarchy of the tree. The arrows in Figure 2 indicate the direction and strength with which the neurons are brought closer to the selected neuron. The magnitude of the vector and the direction depend on the distance and neighbourhood as indicated in the following algorithm:

1. Given $k \in T$ with $a_k \in A$ is the selected neuron, set the value of the neighbouring radius r
2. Begin $i = 1$, $a_s = a_k$
3. Calculate the parent node from the current node $a_t = f^p(a_s)$, obtain all the sons from a_t which are defined as $A^i_{a_t}$
4. For each instance $a_j \in A^i_{a_t}$, update the coordinates for the neuron by following the equation for SOMs
5. $x_j(t+1) = x_j(t) + \eta(t) \cdot g(i, t) \cdot (x_s(t) - x_j(t))$

6. Where $g(i,\, t)$ represents the neighbouring function $\eta(t)$ the learning rate [2].

$$g(i,t)=\exp\left[-\frac{i}{N}\frac{\sqrt{(x_{j1}-x_{s1})^2+\ldots+(x_{jn}-n_{sn})^2}}{\max\limits_{i,j}\{d_{ij}\}}-\lambda\frac{i\cdot t}{\beta N}\right]\eta(t)=\mathrm{Exp}\left[-\sqrt[4]{\frac{t}{\beta N}}\right]$$

7. Where t is the iteration, N the number of elements from group $\#A$, n is the dimension of the coordinates, x_{ij} coordinate j for the neuron $i\in T$, with $a_i\in A$, λ and β the constants established for 1 and 5, respectively.
8. If $i<r$ set $a_s=a_t$ and increase i
9. Use the same procedure to update the descendents $a_k\in A$ until reading depth r, $A^1_{a_k}\ldots A^r_{a_k}$.

3.5 Training: block 1

The training for the neural network is performed iteratively, as shown in Figure 1. Each of the clusters from the network is chosen in a sequential manner. The network training has an initial phase that is similar to SOMs, and neural networks such as GCS or ESOINN.

1. Initiate the entire cluster with the initial group $G=\{P\}$ where P contains all of the nodes from the network.
2. Establish the entire group to be analyzed $i=1$.
3. If $i\geq\#G$ go to 11.
4. Select the group to be analyzed $T=g_i$
5. Establish $j=1$.
6. Apply the Kruskal algorithm to create a minimal tree A, distance matrix D and connection matrix C
7. Select a neuron from $T$and execute the update algorithm on the element T
8. Apply the division algorithm on T. If T divided $i=i-1$.
9. If $j<\beta N$, where β is a constant and N the number of terms for T, increase the value of j and go to 8.
10. If $i<\#T$ increase the value of i and return to 3.
11. Finish

4 Results

To conclude the tests, both real data and fictitious self-generating data were used. The fictitious data was generated from various data distributions with different characteristics and homogeneity. Data were created and distributed homogeneously along a determined surface, and non-homogeneously along the surface of the data distribution. The data were generated by following different functions so that the groups would overlap and have different distributions, thus complicating the clustering process. The data from the test were generated in 2D in order to facilitate the graphical representation of the results.

To confirm that the proposed SOCADNN functioned properly, the clustering process was compared with other statistical techniques traditionally used for unsupervised clustering. A series of data were generated and organized in different clusters resulting in an overlap among the groups. Figure 3a and b shows the clusters obtained in the experiments. In the

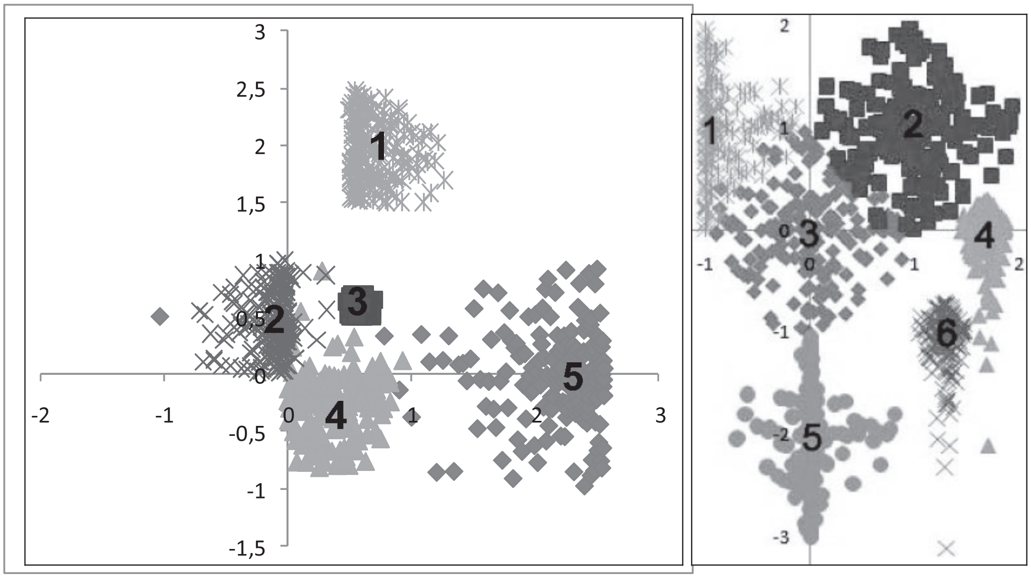

FIG. 3. Sequence of clusters generated by the ANN. Total number of points (**a**) 1080 and (**b**) 900. Each symbol represents a different cluster where the data are assigned.

first example, five clusters were created and, as can be seen in Figure 3a, Cluster 5 presented a high variability (although there is a zone in Cluster 5 where the density is bigger). In the second example, the aim was to obtain elongated clusters (specifically Clusters 4 and 6), in order to compare the performance of the SOCADNN to the alternative existing techniques.

Figure 4 shows the results obtained by the SOCADNN network for the two examples. In the first example, the neural network had the ability to automatically determine the number of clusters for the data. The clusters are similar to those obtained in Figure 4a. In the second example, Cluster 5 is the one that presents more errors, mainly due to the variability of the data.

Finally, the SOCADNN was compared with traditional techniques used for clustering purposes and, as shown in Figure 5, the neural network presents the most accurate results for the examples taken into consideration for this research. In the first example, the improvement is less noticeable, while in the second example, the improvement is more evident. The reason is that in the second case the data do not follow normal distributions.

It is evident that the neural network is more adept at detecting the different forms and the changes in density than the traditional methods. Additionally, it eliminates the expansion phase that the traditional neural networks generally require. In the second case study, Clusters 4 and 6 were placed in order to ensure that certain elements would be closer to the centroids of other clusters, regardless of where the centroid was actually located. As a result, these methods are not capable of making a correct classification, while the neural network is in fact capable of adapting to these circumstances. Second, we studied a real case from the UC Irvine Machine Learning Repository [21] regarding data for wines. The data within the range [0–1] were normalized in order to eliminate the scale factor and units. The classification process was then carried out. The percentage of success was as follows: 91.01%, 90.45%, 93.26%, 94.94%, 33.71% and 71.35% for the SOCADNN, PAM, dendrogram, k-means, agnes

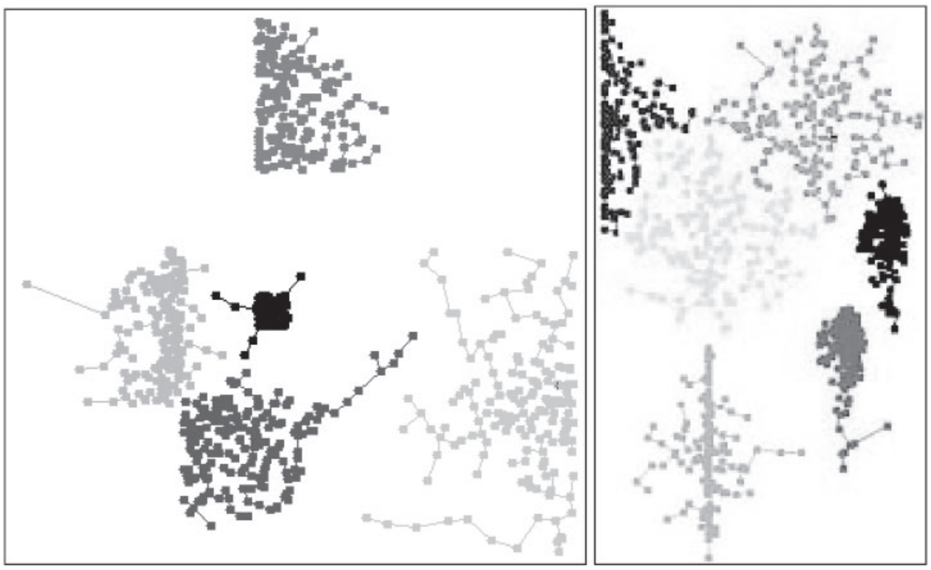

FIG. 4. Sequence of clusters generated by the ANN. Total number of points 1080. Each colour represents a different cluster that was found.

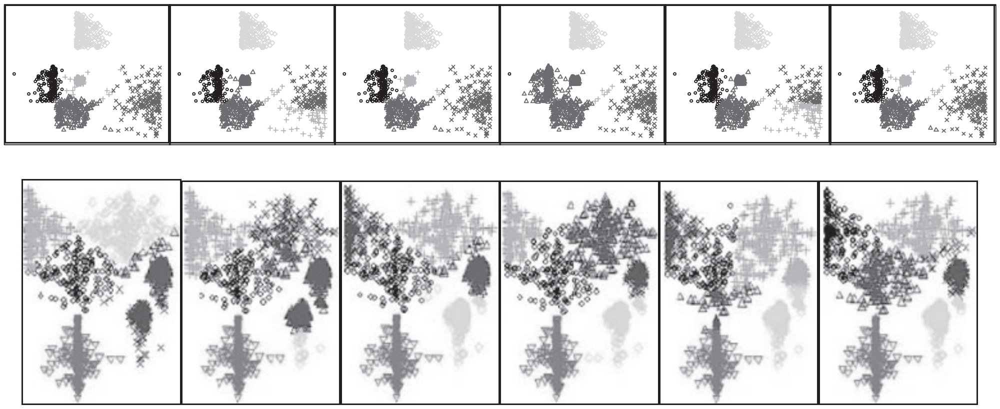

FIG. 5. Sequence of clusters generated by PAM, dendrograms, fanny, agnes, Diana, Clara in this order.

and diana, respectively. Fanny did not produce any results since it included data approaching 0. As we can see, the network provided results similar to the PAM, dendrograms and k-means methods, while the others provided worse results. To analyze the elements that the ANN and PAM classified incorrectly as compared to the dendograms and k-means, we created a 3D representation and applied a multi-dimensional scaling process to reduce the dimensionality. The results demonstrated that the errors were atypical elements that were located outside of both clusters that would have been eliminated with a filtering phase.

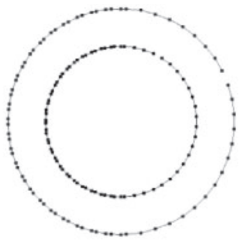

Fig. 6. Figures identified for a set of data with close point distribution.

5 Conclusions

The results obtained with the SOCADNN are promising. We initially detected several deficiencies in the case of elements that are distributed along very close parallel lines. However, we have solved this problem with the new density calculations presented in Section 3.2, which allow the network to better adapt itself to not homogeneous surfaces and to calculate the cut-off points. The network can now recognize figures as concentric circumferences, even if both conferences are very close together. Figure 6 shows the result obtained for an example of data with concentric circumferences surfaces.

Occasionally, the SOCADNN is incapable of calculating the correct cut-off point for dividing clusters, thus functioning as a hierarchical algorithm for which the user must interpret the results. The results can be interpreted according to the distances from the cut-off points and the changes in density. To resolve this problem, we are working on defining criteria for a cut-off point based on the calculation of the densities of the clusters.

Acknowledgement

This development has been supported by the projects SA071A08.

References

[1] A. Akhbardeh, P. E. Nikhil, Koskinenb and O. Yli-Harjaa. Towards the experimental evaluation of novel supervised fuzzy adaptive resonance theory for pattern classification. *Pattern Recognition Letters*, **29**, 1082–1093, 2008.

[2] J. Bajo, J. F. De Paz, Y. De Paz and J. M. Corchado. Integrating case-based planning and RPTW neural networks to construct an intelligent environment for health care. *Expert Systems with Applications,* **36**, 5844–5858, 2009.

[3] R. Campos and M. Ricardo. A fast algorithm for computing minimum routing cost spanning trees. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, **52**, 3229–3247, 2008.

[4] J. Carbó, J. M. Molina and J. Dávila. Fuzzy referral based cooperation in social networks of agents. *AI Communications*, **18**, 1–13, 2005.

[5] J. M. Corchado, J. F. De Paz, S. Rodríguez and J. Bajo. Model of experts for decision support in the diagnosis of leukemia patients. *Artificial Intelligence in Medicine*, **46**, 179–200, 2009.

[6] S. Das, A. Abraham and A. Konar. Automatic clustering using an improved differential evolution algorithm. *IEEE Transactions on Systems Man and Cybernetics*, **38**, 218–237, 2008.

[7] S. Das, A. Abraham and A. Konar. Automatic kernel clustering with Multi-Elitist Particle Swarm Optimization Algorithm. *Pattern Recognition Letters*, **29**, 688–699, 2008.

[8] S. Das, A. Abraham and A. Konar. *Metaheuristic Clustering, Studies in Computational Intelligence*, vol. 178. Springer, 2009.

[9] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems* 7, G. Tesauro, D. S. Touretzky and T. K. Leen, eds, pp. 625–632, 1995.

[10] S. Furao, T. Ogura, O. Hasegawa. An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks*, **20**, 893–903, 2007.

[11] J. García, A. Berlanga, J. M. Molina, J. R. Casar. Methods for operations planning in airport decision support systems. *Applied Intelligence*, **22**, 183–206, 2005.

[12] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics* **28**, 100–108, 1979.

[13] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990.

[14] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, **43**, 59–69, 1982.

[15] T. Martinetz and K. Schulten. A neural-gas network learns topologies. *Artificial Neural Networks*, **1**, 397–402, 1991.

[16] J. Pavón, M. Arroyo, S. Hassan and C. Sansores. Agent-based modelling and simulation for the analysis of social patterns. *Pattern Recognition Letters*, **29**, 1039–1048, 2008.

[17] J. Pavón, J. Gómez, A. Fernández and J. Valencia. Development of intelligent multi-sensor surveillance systems with agents. *Robotics and Autonomous Systems*, **55**, 892–903, 2007.

[18] R. –W. Po, Y. –Y. Guh and M. –S. Yang. A new clustering approach using data envelopment analysis. *European Journal of Operational Research*, **199**, 276–284, 2009.

[19] N. Saitou and M. Nie. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, **4**, 406–425, 1987.

[20] F. Shen. *An algorithm for incremental unsupervised learning and topology representation*. PhD Thesis, Tokyo Institute of Technology, 2006.

[21] UC Irvine Machine Learning Repository. Available at http://archive.ics.uci.edu/ (Last accessed date June 29, 2010).

[22] L. Xu. Bayesian Ying–Yang machine, clustering and number of clusters. *Pattern Recognition Letters*, **18**, 1167–1178, 1997.

Received 4 November 2009