Proceedings of the International Conference
on Computational and Mathematical Methods
in Science and Engineering, CMMSE2009
Gijón, Spain, June 30, July 1-3 2009

A new clustering algorithm applying a hierarchical method neural network

Juan F. De Paz, Sara Rodríguez, Javier Bajo and Juan M. Corchado

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

emails: {fcofds, srg, jbajope, corchado}@usal.es

Abstract

Clustering is a branch of multivariate analysis that is used to create groups of data. While there are currently a variety of techniques that are used for creating clusters, many require defining additional information, including the actual number of clusters, before they can be carried out. The case study of this research 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.

Key words: Clustering, SOM, hierarchical clustering, PAM, Dendrogram,

1. Introduction

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 including bioinformatics [9] [19] and surveillance [14] [15]. The methods used for clustering differ considerably according to the type of data and the amount of available information. Clustering techniques are typically broken down into the following categories [18] [19] **¡Error! No se encuentra el origen de la referencia.**: hierarchical, which include dendrograms [6], agnes [8], Diana [8], Clara []; neural networks such as self-organized maps [2], GCS [3], ESOINN [1]; methods based on minimizing objective functions, such as k-means [10] and PAM [8] (Partition around medoids); or probabilistic-based models such as EM [7] (Expectation-maximization) and fanny [8]. Traditionally the different

methods try to minimize 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. In certain cases, neural networks allow 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.

This research presents the new Self Organized Dynamic Tree neural network (SODTNN) which allows data to be grouped automatically, without having to specify the number of existing clusters. Additionally, the SODTNN eliminates the expansion phase before dividing and interconnecting the neurons, thus avoiding one of the most costly phases of the algorithm. The SODTNN uses algorithms to detect low density zones and graph theory procedures in order to establish a connection between elements. This would allow connections to be established dynamically, thus avoiding the need for the network to expand and adjust the data surface. 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 SODTNN 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 that 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 different clustering alternatives, section 3 describes the SODTNN, and section 4 presents the results and conclusions.

2. Clustering techniques

The techniques used for creating clusters vary according to the type of problem at hand. Because of the ease in performing calculations, the most common clustering techniques used with bioinformatics are either hierarchical or based on minimizing objective functions. Hierarchical methods such as dendrograms [6] 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 [7]. The k-means algorithm presents 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. This is due to the fact that the algorithms tend to associate each element according to the distances from a pre-determined element. This implies that they are not able to correctly detect the cluster, as indicated in the results. Artificial neural networks (ANN) [16], [17] can adapt to the data surface, although they usually require additional time to do so. The self-organized Kohonen maps (SOM) [2], have variants of learning methods that base their behaviour on methods similar to the Neural Gas (NG) [4]. 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. Growing Cell Structure (GCS) [3] 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 ANN such as SOINN [5] and ESOINN [1] (Enhanced self-organizing incremental neural network). 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 [20] 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. SODTNN

This study proposes the SODTNN, which can detect the number of existing groups or classes and, by using the Kruskal algorithm [11], create clusters based on the connections taken from the minimum spanning tree. As opposed to the ESOINN or GCS networks, the SODTNN 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 neighboring 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 phase ~~is~~ 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 neighboring 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.

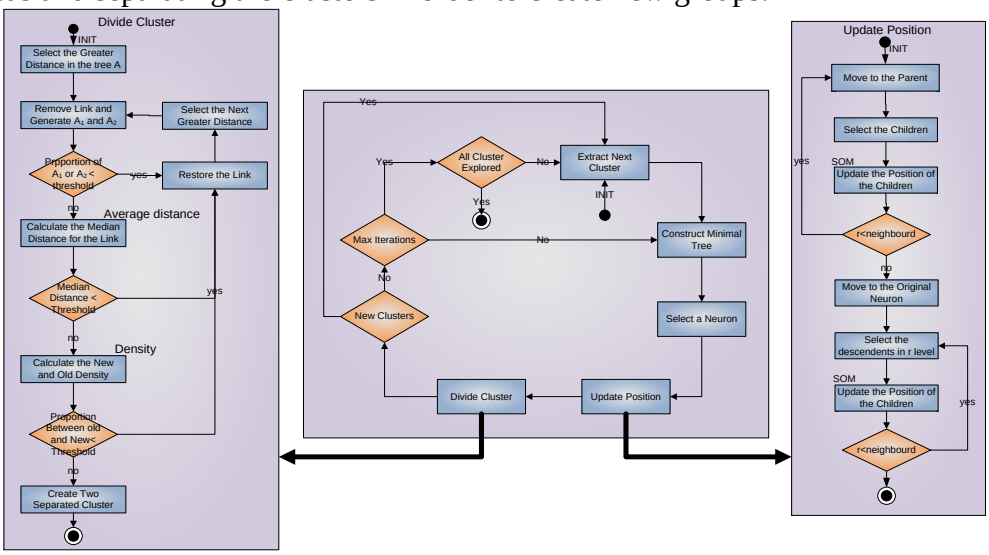

Fig. 1. Algorithm for the SODTNN

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 Density: block 3

One of the main problems when assigning individuals into groups is knowing which divisions cause a significant rise in the density of the resulting clusters. ANN such as SOINN or ESOINN study 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 SODTNN 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.

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 $f^T(D) = \sum_{i,j} d_{ij}, d_{ij} \in D$
3. Calculate the final density $f^D(C,D) = f^T(D) / f^A(C,D)$

3.2 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 centered 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.

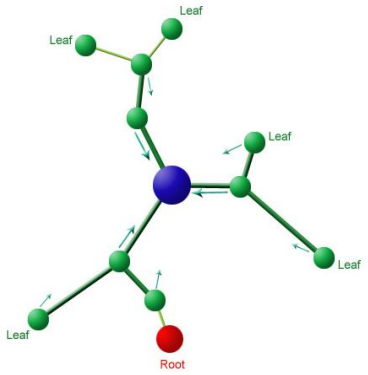

Fig. 2. Subtree for neuron falling within 2 links

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, that is defined by
   $f^p : A \to A$
   $a_i \to f^p(a_i) = a_s$ Where $c_{si} = 1$ 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 $A_{a_i}^e \subseteq A$ as follows $A_{a_i}^e = \{a_j \in A / \exists r \in \mathrm{N}, a_r = (\overbrace{f^p \circ \ldots \circ f^p}^{r \le 2e})(a_j)\}$
4. Calculate the average distance for the node a_i $f^m(A_{a_i}^e, D) = \dfrac{\sum_{a_j \in A_{a_i}^e} d_{sj} \cdot c_{sj}}{\# A_{a_i}^e}$

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, 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 $d_{a_k}^m = f^m(A_{a_i}^e, D)$
7. Determine if the distance from tree node a_k and its parent is less than the average distance $d_{sk} \le d_{a_k}^m \cdot \beta$ 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 ak 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 neighboring neurons closer in. The neuron is 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 neighborhood 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 neighboring 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 $A^1_{a_t}$
4. For each instance $a_j \in A^1_{a_t}$ update the coordinates for the neuron by following the equation for self-organizing maps
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 neighboring function $\eta(t)$ the learning rate [12].

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

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, $A_{a_k}^1 \ldots A_{a_k}^r$

3.5 Entrenamiento

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 self-organized maps, and neural networks such as GCS or ESOINN.

1. Initiate the entire cluster with the inicial 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 **¡Error! No se encuentra el origen de la referencia.**
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 and Conclusions

In order 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 was created and distributed homogeneously along a determined surface, and non-homogeneously along the surface of the data distribution. The data from the test was generated in 2D in order to facilitate the graphical representation of the results.

In order to confirm that the proposed SODTNN functioned properly, the clustering process was compared with other statistical techniques traditionally used for unsupervised clustering. In the first case, we selected a test case generated with fictitious data. Figure 4 shows the data that was generated manually for creating the clusters. A total of 6 clusters were used to create the groups. It is easy to see how elements from the different clusters overlap, from

which it can be easily assumed that the error rate will not be low for the methods that are applied. For example, cluster 4 contains some data in the lower range that are considerably closer to the elements in cluster 6 than to many of the elements in their own cluster. Practically any clustering technique would associate those elements with group 6.

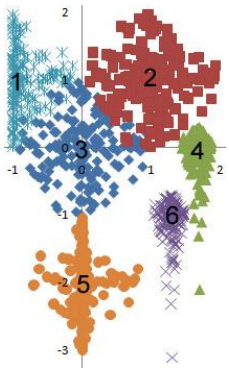

Fig. 4. Sequence of clusters generated by the ANN. Total number of points 1080. Each colour represents a different cluster that was found.

The images in Figure 5 show the classification process as carried out by the SODTNN network for the given test. The five images show the sequence of the clusters detected with successive iterations. The last image shows the different groups that were obtained with the neural network as the final classification. We can see how the elements from cluster 4 were incorrectly classified, as we expected. However, the classification performed with the SODTNN method is very similar to the original clustering of the data and corresponds in large part to what could be done manually.

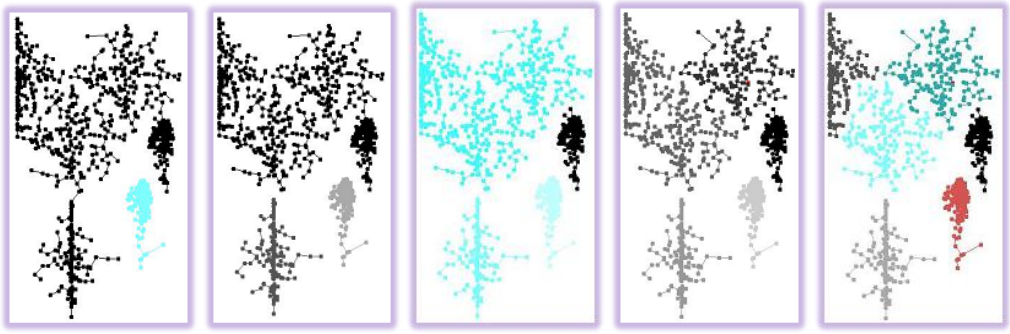

Fig. 5. Sequence of clusters generated by the ANN. Total number of points 1080. Each colour represents a different cluster that was found.

Figure 6 illustrates the final classification obtained by the PAM, dendrogram, k-means, fanny, diana and clara clustering methods. For each of these methods, it was necessary to determine the number of clusters that we wanted to obtain, for which it was necessary to provide more information than with the SODTNN.

According to the images, the classification of groups 4 and 6 is quite inaccurate for each of the different methods. This is because the methods tend to be based on models within the clusters that associate the elements with different groups. The classification of the remaining groups is more or less similar, given that the distribution of the calculated data favors assigning clusters through the selection of a particular model. With certain cases, such as Diana, K-means or dendrograms, the resulting classification is quite erroneous. PAM, fanny, agnes and Clara were capable of detecting different clusters, although they were not entirely accurate in detecting the limits of each one.

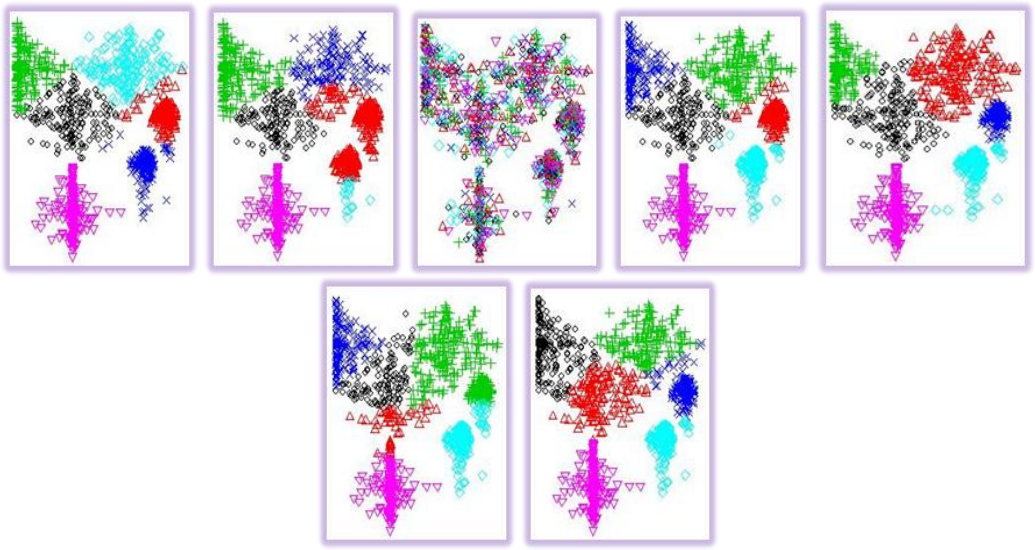

Fig. 6. Sequence of clusters generated by PAM, dendrograms, k-means, fanny, agnes, Diana, Clara in this order.

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. Clusters 4 and 6 were placed such 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. Secondly, we studied a real case from the UC Irvine Machine Learning Repository [13] regarding data for wines. The data within the range [0-1] was 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%, 71.35% for the SODTNN, PAM, dendrogram, k-means, agnes 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. In order 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 multidimensional 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.

The results obtained with the SODTNN are promising. Nevertheless, we have detected several deficiencies in the case of elements that are distributed along very close parallel lines. Occasionally, the SODTNN 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. In order 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.

Acknowledgements. This development has been supported by the projects SA071A08 and SIAAD-TSI-020100-2008-307.

5. References

[1] FURAO, S., OGURA, T., HASEGAWA, O.: AN ENHANCED SELF-ORGANIZING INCREMENTAL NEURAL NETWORK FOR ONLINE UNSUPERVISED LEARNING. NEURAL NETWORKS, VOL. 20, 893--903 (2007)

[2] KOHONEN, T.: SELF-ORGANIZED FORMATION OF TOPOLOGICALLY CORRECT FEATURE MAPS. BIOLOGICAL CYBERNETICS, 59--69 (1982)

[3] FRITZKE, B.: A GROWING NEURAL GAS NETWORK LEARNS TOPOLOGIES.. G. TESAURO, D. S. TOURETZKY AND T. K. LEEN, EDS, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 7, 625--632 (1995)

[4] MARTINETZ, T., AND SCHULTEN, K.: A NEURAL-GAS NETWORK LEARNS TOPOLOGIES. ARTIFICIAL NEURAL NETWORKS, VOL 1. 397--402 (1991)

[5] SHEN, F.: AN ALGORITHM FOR INCREMENTAL UNSUPERVISED LEARNING AND TOPOLOGY REPRESENTATION. PH.D. THESIS. TOKYO INSTITUTE OF TECHNOLOGY (2006)

[6] SAITOU, N. AND NIE, M.: THE NEIGHBOR-JOINING METHOD: A NEW METHOD FOR RECONSTRUCTING PHYLOGENETIC TREES. MOL. BIOL. VOL. 4, 406--425 (1987)

[7] XU, L.: BAYESIAN YING–YANG MACHINE, CLUSTERING AND NUMBER OF CLUSTERS PATTERN RECOGNITION LETTERS. VOL. 18 (11-13) 1167--1178 (1997)

[8] KAUFMAN, L., AND ROUSSEEUW, P.J.: FINDING GROUPS IN DATA: AN INTRODUCTION TO CLUSTER ANALYSIS. WILEY, NEW YORK (1990)

[9] CORCHADO, J.M., DE PAZ, J.F., RODRÍGUEZ, S., BAJO J.: MODEL OF EXPERTS FOR DECISION SUPPORT IN THE DIAGNOSIS OF LEUKEMIA PATIENTS. ARTIFICIAL INTELLIGENCE IN MEDICINE. IN PRESS.

[10] HARTIGAN, J. A. AND WONG, M. A.: A K-MEANS CLUSTERING ALGORITHM. APPLIED STATISTICS 28, 100--108 (1979)

[11] CAMPOS, R. AND RICARDO M.: A FAST ALGORITHM FOR COMPUTING MINIMUM ROUTING COST SPANNING TREES, VOL. 52 (17) 3229--3247 (2008)

[12] BAJO J., DE PAZ J.F., DE PAZ Y. AND CORCHADO J.M.: INTEGRATING CASE-BASED PLANNING AND RPTW NEURAL NETWORKS TO CONSTRUCT AN INTELLIGENT ENVIRONMENT FOR HEALTH CARE. EXPERT SYSTEMS WITH APPLICATIONS VOL. 36 (3) (2009)

[13] UC IRVINE MACHINE LEARNING REPOSITORY HTTP://ARCHIVE.ICS.UCI.EDU/

[14] CARBÓ, J., MOLINA, J.M., DÁVILA, J.: FUZZY REFERRAL BASED COOPERATION IN SOCIAL NETWORKS OF AGENTS", AI COMMUNICATIONS. VOL. 18, (1) 1--13 (2005)

[15] GARCÍA, J., BERLANGA, A., MOLINA, J.M., CASAR, J.R.: METHODS FOR OPERATIONS PLANNING IN AIRPORT DECISION SUPPORT SYSTEMS, APPLIED INTELLIGENCE. VOL. 22 (3) 183--206 (2005)

[16] PAVÓN, J., ARROYO, M., HASSAN, S., SANSORES, C.: AGENT-BASED MODELLING AND SIMULATION FOR THE ANALYSIS OF SOCIAL PATTERNS. PATTERN RECOGNITION LETTERS 29 1039--1048 (2008)

[17] PAVÓN, J., GÓMEZ, J., FERNÁNDEZ, A., VALENCIA, J.: DEVELOPMENT OF INTELLIGENT MULTI-SENSOR SURVEILLANCE SYSTEMS WITH AGENTS ROBOTICS AND AUTONOMOUS SYSTEMS, VOL. 55 (12) 892--903 (2007)

[18] RUNG-WEI PO, YUH-YUAN GUH, AND MIIN-SHEN YANG, A NEW CLUSTERING APPROACH USING DATA ENVELOPMENT ANALYSIS, EUROPEAN JOURNAL OF OPERATIONAL RESEARCH, VOL. 199 (1) 276--284 (2009)

[19] G. KERR, A, , H.J. RUSKINA, M. CRANEA AND P. DOOLAN TECHNIQUES FOR CLUSTERING GENE EXPRESSION DATA, COMPUTERS IN BIOLOGY AND MEDICINE, VOL. 38 (3) 283--293 (2008)

[20] A. AKHBARDEH, NIKHIL, P.E. KOSKINENB AND O. YLI-HARJAA TOWARDS THE EXPERIMENTAL EVALUATION OF NOVEL SUPERVISED FUZZY ADAPTIVE RESONANCE THEORY FOR PATTERN CLASSIFICATION, PATTERN RECOGNITION LETTERS, 29 (8) 1082--1093 (2008)