Self Organized Dynamic Tree Neural Network

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

Departamento de Informática y Automática, Universidad de Salamanca
Plaza de la Merced s/n, 37008, Salamanca, España
{fcofds,srg,jbajope,corchado,vivian}@usal.es
Department of Computer Science and Automation, University of Salamanca Plaza de la Merced s/n, 37008, Salamanca, Spain

Abstract. Cluster analysis is a technique used in a variety of fields. There are currently various algorithms used for grouping elements that are based on different methods including partitional, hierarchical, density studies, probabilistic, etc. This article will present the SODTNN, which can perform clustering by integrating hierarchical and density-based methods. The network incorporates the behavior of self-organizing maps and does not specify the number of existing clusters in order to create the various groups.

Keywords: Clustering, SOM, hierarchical clustering, PAM, Dendrogram.

1 Introduction

The assignment of a set of objects into clusters is a widely spread problem that has been the object of investigation in various scientific branches including bioinformatics [10], surveillance [15], [16], [17]. Although occasionally the number of groups is known beforehand, clustering data requires an additional step for identifying the existing groups. There are currently different methods for creating clusters, most notably those based on partitioning, such as k-means [11], and PAM [9] (Partition around medoids), which work by minimizing the error function. Other widely accepted methods are the hierarchical methods which include dendrograms [7], agnes [9], and Diana [9]. In addition to the hierarchical methods, there are others that use density-based models, or probabilistic-based models such as EM [8] (Expectation-maximization) and fanny [9].

This research presents the new Self Organized Dynamic tree neural network which allows data to be grouped automatically, without having to specify the number of existing clusters. 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

J. Cabestany et al. (Eds.): IWANN 2009, Part I, LNCS 5517, pp. 220–227, 2009.

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 problem of clustering is far reaching, and there have been various proposals for its resolution: i) Partition based methods have the disadvantage of requiring the number of clusters up front [8]. The k-means algorithm presents problems with atypical points. The PAM method resolves this problem by assigning an existing element as the centroid. ii) Hierarchical methods such as dendrograms [7] do not require a number of clusters up front since they use a graphical representation to determine the number. iii) Probability based methods such as EM define an algorithm of probabilities that determines the probability that a point belongs to a cluster. iv) Finally, there are the methods that use changes in density in order to separate clusters. Included in these methods are the artificial neural networks (ANN) [18], [19], which estimate the surface of the point distribution by using a mesh of neurons that can be automatically adjusted to the surface. There are also other networks such as ART [5] (Adaptive Resonance Theory) that can form clusters, although it does not function based on meshes. Our research will concentrate on the mesh-based neural networks.

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 [6] 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.

3 SODTNN

This study proposes the SODTNN, which can detect the number of existing groups or classes and, by using the Kruskal algorithm [12], 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

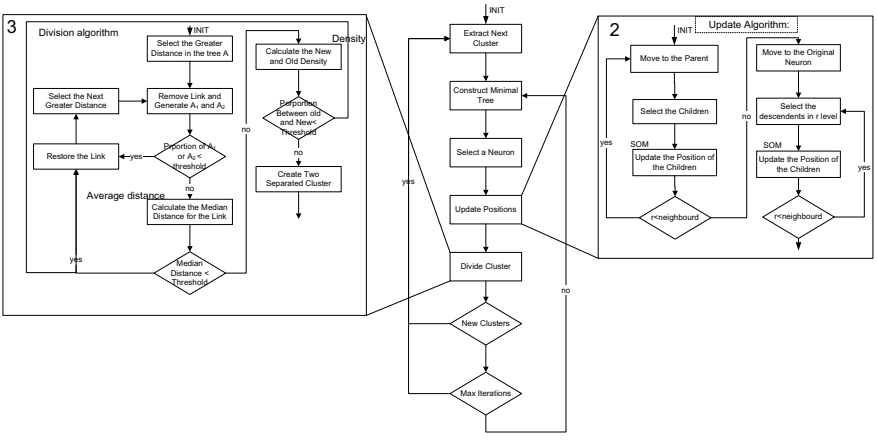

Fig. 1. Algorithm for the SODTNN

adjust to the surface. 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 center, block 2 is the algorithm that updates the position of the neurons, and block 3 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 $i \in$ T is connected with element $j \in T$,, 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.

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 gray to indicate the neuron that falls within 2 links.

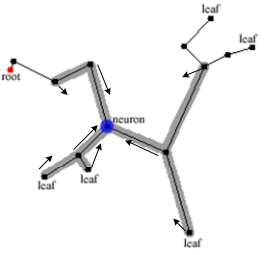

Fig. 2. Subtree for neuron falling within 2 links

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

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 $d^m_{a_k} = f^m(A^e_{a_i}, D)$
7. Determine if the distance from tree node a_k and its parent is less than the average distance $d_{sk} \leq d^m_{a_k} \cdot \beta$ 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 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

   $$x_j(t+1) = x_j(t) + \eta(t) \cdot g(i,t) \cdot (x_s(t) - x_j(t))$$

   Where $g(i,t)$ represents the neighboring function $\eta(t)$ the learning rate [13].

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

   $$\eta(t) = Exp\left[-\sqrt[4]{\frac{t}{\beta N}} \right]$$

   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.
5. If $i < r$ set $a_s = a_t$ and increase i
6. Use the same procedure to update the descendents $a_k \in A$ until reading depth r, $A^1_{a_k} \ldots A^r_{a_k}$

4 Results and Conclusions

In order to conclude the tests, both real data and fictional self-generating data were used. Additionally, graphic representations are in 2D in order to facilitate the interpretation of the results. In order to confirm that the proposed SODTNN functioned properly, the clustering process was compared with other statistical techniques traditionally use for unsupervised clustering. In the first case, we selected a test case generated with fictional data. Figure 3 illustrates the classification process that was carried out for the given test. We can see the data and the sequence of the divisions in different colors that were made with the SODTNN until the algorithm finalize, where the last image shows the results obtained after implementing the algorithm.

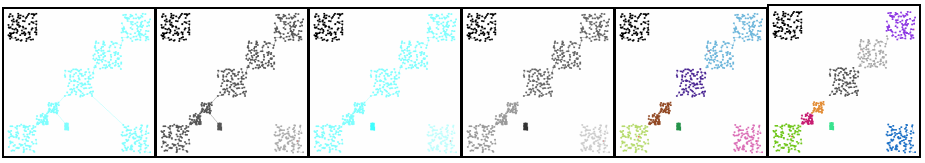

Fig. 3. Sequence of clusters generated by the ANN. Total number of points 774. Each colour represents a different cluster that was found.

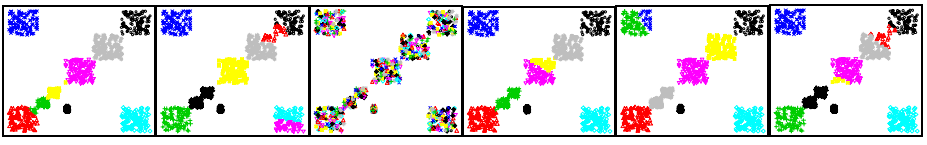

Fig. 4. Sequence of clusters generated by PAM, dendrograms, k-means, fanny, agnes, dian in this order

Figure 4 illustrates the final classification obtained by the PAM, dendograms, k-means, fanny, agnes and diana 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. Additionally, it is easy to see how only the PAM method is capable of obtaining results comparable to our network, although the green group includes elements from other clusters. The rest of the processes generate classifications that could be considered as erroneous given the distribution of the data.

Secondly, we studied a real case from the UC Irvine Machine Learning Repository [14] 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, dendograms 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.

References

[1] Furao, S., Ogura, T., Hasegawa, O.: An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Networks 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. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (eds.) Advances in Neural Information Processing Systems, vol. 7, pp. 625–632 (1995)
[4] Martinetz, T., Schulten, K.: A neural-gas network learns topologies. Artificial Neural Networks 1, 397–402 (1991)
[5] Carpenter, G.A., Grossberg, S.: The ART of adaptive pattern recognition by a self-organizing neural network. IEEE Trans. Computer, 77–88 (1987)
[6] Shen, F.: An algorithm for incremental unsupervised learning and topology representation. Ph.D. thesis. Tokyo Institute of Technology (2006)
[7] Saitou, N., Nie, M.: The neighbor-joining method: A new method for reconstructing phylogenetic trees. Mol. Biol. 4, 406–425 (1987)
[8] Xu, L.: Bayesian Ying–Yang machine, clustering and number of clusters. Pattern Recognition Letters 18(11-13), 1167–1178 (1997)
[9] Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York (1990)
[10] 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)
[11] Hartigan, J.A., Wong, M.A.: A K-means clustering algorithm. Applied Statistics 28, 100–108 (1979)
[12] Campos, R., Ricardo, M.: A fast algorithm for computing minimum routing cost spanning trees 52(17), 3229–3247 (2008)
[13] Bajo, J., De Paz, J.F., De Paz, Y., Corchado, J.M.: Integrating case-based planning and RPTW neural networks to construct an intelligent environment for health care. Expert Systems with Applications 36(3) (2009)
[14] UC Irvine Machine Learning Repository, `http://archive.ics.uci.edu/`
[15] Patricio, M.A., Carbó, J., Pérez, O., García, J., Molina, J.M.: Multi-Agent Framework in Visual Sensor Networks. EURASIP Journal on Advances in Signal Processing, special issue on Visual Sensor Networks, 21 (2007)
[16] Carbó, J., Molina, J.M., Dávila, J.: Fuzzy Referral based Cooperation in Social Networks of Agents. AI Communications 18(1), 1–13 (2005)
[17] García, J., Berlanga, A., Molina, J.M., Casar, J.R.: Methods for Operations Planning in Airport Decision Support Systems. Applied Intelligence 22(3), 183–206 (2005)
[18] 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)
[19] Pavón, J., Gómez, J., Fernández, A., Valencia, J.: Development of intelligent multi-sensor surveillance systems with agents. Robotics and Autonomous Systems 55(12), 892–903 (2007)