Learning and Nonlinear Models – Revista da Sociedade Brasileira de Redes Neurais, Vol. 2, No. 2, pp. 99-106, 2004
© Sociedade Brasileira de Redes Neurais

A NEURAL NETWORK BASED ALGORITHM FOR COMPLEX PATTERN CLASSIFICATION PROBLEMS

Allan de Medeiros Martins, Adrião Duarte Dória Neto, Jorge Dantas de Melo
UFRN, Departamento de Engenharia de Computação e Automação
allan@dca.ufrn.br, adriao@dca.ufrn.br, jdmelo@dca.ufrn.br

Abstract - This work presents an application of neural networks in pattern classification. A new algorithm for automatic classification of data is presented. The algorithm makes use of a competitive neural network to aid the classification process. The algorithm gets a data set *D* and segments it into clusters. The only prior given information is a number of auxiliary centers and a threshold distance. The algorithm uses the *Mahalanobis* metrics to cluster the data and find itself the number of classes. Some tests were made in artificially generated data sets with complex distributions and compared to standard classification methods that use Euclidian distance as its metrics.

Index Terms – Pattern Classification, Vector Quantization, Neural Networks, *Mahalanobis* distance

Introduction

Many research areas like data mining, image segmentation, pattern recognition and statistical data analysis make use of pattern classification to accomplish a task that is part of a process. The main objective of a good pattern classification algorithm is to separate classes that are distributed arbitrarily in the data space, in a non-supervised way. Some techniques have been developed to achieve this task, and a great number of them are based on several heuristics. The most simple way to classify patterns consists in the use of vector quantization techniques, such as k-means[1] or pure competitive neural networks[2] to find the centers that represent the clusters. The similarity measure used in most of those techniques is the Euclidian distance between the point and the center of its class. Recently many papers have been published with new approaches and modifications of classical k-means and competitive networks. In [3][4] the authors use a different metric (in fact a non-Euclidian metric), which is used in classical algorithm to improve the positioning of the clusters, modeling the clusters as hyperellipsoids. Those techniques improve the classical vector quantization (VQ) allowing more complex clusters and more sophisticated decision boundaries. However, as can be seen in this paper, some data sets are even more complex and simple hyperellipsoids does not fit the data very well. Non-Euclidian metrics is also used in this paper and is the main idea behind the proposed method. More elaborate techniques, using Kohonen(SOM) maps[5] or Fuzzy k-means[6] have been developed. That techniques still model the data set using Euclidian metrics and even with more precise center positioning, the generated model is still far from fitting complex data sets. Some modifications of the usual SOM algorithm that is based in the segmentation of the output map had also been developed[7][8] and had given good results, but need more complex computations. Because of the segmentation of the Kohonen map, some heuristics must be used. As a result, the algorithm became more complex, requiring heavy computation efforts. The idea is to quantify the data using the SOM algorithm and, after that, generate the U matrix[5] with the distances between each center. Because of the topology preservation in the SOM map, points of the data set that are close, fall into the same region in the U matrix. Therefore, if we segment the U matrix, we can also segment the data set. All this process is hard to compute and no model is generated, requiring that we use a test point to the neural net to verify which class those points belong to.

Neural approach for modeling the classifier using neurons with non-Euclidian metrics is also proposed in literature[9]. That approach consists in composing a neural network with high-order neurons that can model any shape in space. Regular (one or two order) neurons can model hyperspheres or hyperellipsoids. With high-order neurons, the authors propose that a more complex shape can be achieved. Again, the computational cost to train the model and the choice of the order of the model for the neurons limits the algorithm to some applications. In most of those techniques, the number of classes that exist in the data set must be given previously. In some cases this information is not available, so it is important to develop algorithms that perform the automatic classification without this information. Another problem in the pattern classification is the complexity of the spatial distribution of the data set. This kind of problem is hard to solve because of the complexity of the data model. The statistical data model in those complex problems is not Gaussian and therefore some more sophisticated models must be used. The proposed algorithm uses a simple competitive neural network and a linking heuristic of auxiliary centers to cluster similar regions using the *Mahalanobis* distance[10] as metric of similarity between points and its centers. The incorporation of the spatial statistics of the data gives us a good measure of the distribution of the points, making the classification of very complex data set possible. The only prior information on the data set given to the algorithm is the number of auxiliary centers and a threshold distance. The use of threshold in competitive neural networks is an old idea and it was first proposed by Grossberg[13] based on the leader-joiner approach proposed by Späth[14] to avoid having to provide the number of centers. The threshold used in this work also avoids having to provide the number of centers, but it is not used in competitive training, it is used, as we will see, to link the auxiliary centers.

The paper is organized as follows: in section 1 the competitive neural networks used to position the auxiliary centers used in the method will be described. The training process and the structure of the neural network are detailed. In section 2, it will be described the *Mahalanobis* metric used to achieve complex data modeling. We will define the square distance and how to estimate the covariance matrix. In section 3, it will be illustrated the proposed algorithm and some metrics details. A brief summary of the algorithm is showed and the linkage heuristics are detailed. In section 4, some results are presented with some artificially generated classical examples and a generalization of some examples to test the robustness of the method. In section 5, some conclusions and comments about the proposed method are presented.

1 - Competitive neural networks

In the proposed algorithm, auxiliary centers that are distributed in a uniform manner through the data set are used. This distribution is made using a conventional competitive neural network. A competitive neural network is a self-organizing neural network that achieves vector quantization in a given data set. The network has an input layer and an output layer as shown in figure 1.

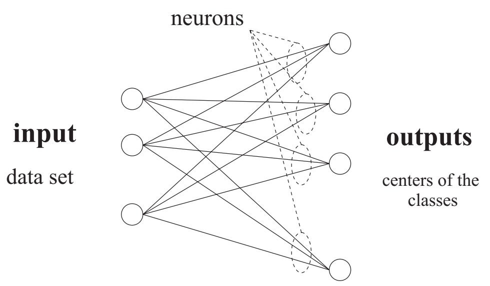

Figure 1: Competitive neural network topology

The input layer of the network takes the training patterns, each of them corresponding to a point in the data set. Each neuron in the network will represent a center of each class that will quantify the total number of points in the data set. The training process of a competitive neural network is based on the update of weights of the winner neuron, that is, the neuron with smallest distance to a given input pattern. In the competitive training, only one neuron is updated for each input pattern, this approach is called *winner takes all*. The similarity criterion is the Euclidian distance between the input pattern and the weights of the neuron. The weights are updated as presented in the equation (1)[2]

$$\begin{aligned}\mathbf{w}_c(n) &= \mathbf{w}_c(n) + \Delta\mathbf{w}_c(n)\\ \Delta\mathbf{w}_c(n) &= \alpha\eta(n)(\mathbf{x}_n - \mathbf{w}_c(n))\end{aligned} \tag{1}$$

where $\mathbf{w}_c(n)$ are the weights of the winner neuron in the iteration n, $\Delta\mathbf{w}_c(n)$ is the difference that will be added to the winner neuron, α is the learning parameter and η is a convergence parameter. $\eta(n)$ is updated at each iteration having its value decreased to avoid oscillations in the convergence. Generally we take $\eta(n+1) = \eta(n)\tau$, where τ is a constant. $\mathbf{x}_n$ is the input pattern. After some iterations, the weights of the network's neurons converge to the centers that represent the entire data set.

The next step is to choose a metrics that make possible the correct link of each center to represent a class. The main idea is to choose a metrics that can incorporate the local statistics of the data. By taking the local statistics, we can model the data with a statistical model, which is very important in the decision process.

2 - Mahalanobis metrics

The *Mahalanobis* metrics is a similarity measure that consider spatial statistics of the points where the measure takes place[11]. The distance between two points, $\mathbf{p}_1$ and $\mathbf{p}_2$ inside a space, where $\mathbf{C}$ is the covariance matrix of the distribution of the probability that represents the spatial statistics, is given by:

$$d_m(\mathbf{p}_1, \mathbf{p}_2, \mathbf{C}) = (\mathbf{p}_1 - \mathbf{p}_2)^t C^{-1}(\mathbf{p}_1 - \mathbf{p}_2) \tag{2}$$

Figure 2 shows the effect of spatial probability in the distance of two points. In that figure, even though the distance d_1 seems to be greater (in terms of Euclidian distance) than the distance d_2, it is not. That difference exists because the covariance matrix of the spatial distribution gives a greater weight to the distances that are not in the direction of the distribution.

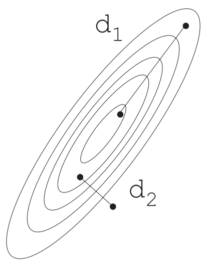

Figure 2: Distance measurement in a space with non-uniform probability distribution

The *Mahalanobis* distance is important in problem classification since the information about spatial distribution of the points is incorporated in the metrics. The spatial distribution can be represented by the statistics of the data set where we wish to measure, in some way, the distance between two points.

We can consider the Euclidian distance, a special case of the *Mahalanobis* distance, where the data would be uniformly distributed and the covariance matrix is an identity. That case corresponds to a distribution where the covariance matrix is a diagonal matrix. In that sense, the *Mahalanobis* metrics become a general case of distance measurements and suitable to be used in problem classification.

3 - The algorithm

Figure 3 shows a set of bidimensional points where exists two clusters of data representing two regions of interest. The goal of the pattern classification is to separate this distribution in two regions (two classes) and, given a point out of the data set, to say if that point belongs to a class or another.

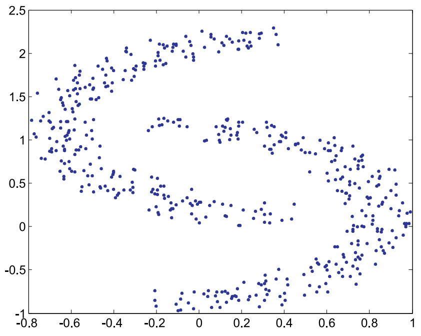

Figure 3: Data set example

At first, the algorithm finds N_a auxiliary centers that separate the data set in N_a regions according to the Euclidian distance. For this task, a competitive neural network that locates the auxiliary centers is used. The number of auxiliary centers must be chosen in order to divide the region as much as possible, without slowing the algorithm. Usually, a choice based on a percentage of total number of points is suitable.

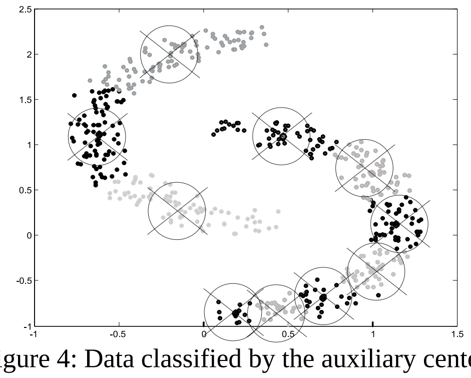

Figure 4: Data classified by the auxiliary centers

After the choice of auxiliary centers, the data set is divided in N_a regions that still not correspond to the real clusters of the data set, as shown in the figure 4. To cluster the regions, for each center, the *Mahalanobis* distance is measured between that center and all others. Each distance is compared with a threshold distance. If that distance is smaller than the threshold, the two centers are linked. The covariance matrix used for the *Mahalanobis* distance between the auxiliary centers is an estimate of the covariance matrix of the distribution formed by the regions classified for both auxiliary centers. This is done in the following way: let $x_1, x_2, \ldots, x_n$ be points that belong to the region of one of the two centers where we want to calculate the *Mahalanobis* distance. The mean point of the data set is represented by $\mathbf{m}$. The covariance matrix $\mathbf{C}$ can be estimated as showed in the equation (3)[12].

$$\hat{\mathbf{C}} = \frac{1}{n-1}\sum_i (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^t \tag{3}$$

Using the equation (3) it is possible to estimate the covariance matrix of the data where the auxiliary centers are in. With this, it is possible to calculate the *Mahalanobis* distance between each pair of auxiliary centers. The figure 5 shows the calculation of the distance between two centers whose regions where used to estimate the covariance matrix.

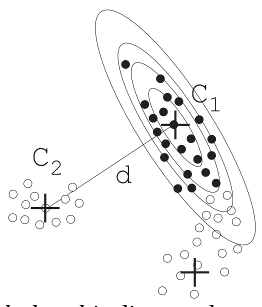

Figure 5: *Mahalanobis* distance between 2 centers

In figure 6, only the sets marked with crosses where used to estimate the covariance matrix used in the calculation of the *Mahalanobis* between $\mathbf{c}_1$ and $\mathbf{c}_2$. In this way, points whose data are "aligned" will have a smaller distance than points where the data are not in the variation of the whole set. According to the measured distances between each center and the others, the concept of linking between centers is established. If the measured *Mahalanobis* distance is smaller than a threshold d_t, the centers are linked, and then, the two regions becomes one. If a previously linked center is to be linked again, the data of the 3 centers involved become a single region and so on. In the end of this process, the set will be divided in *N* regions, each one defined by one or more linked centers. The figure 6 shows two regions in the end of the linking process.

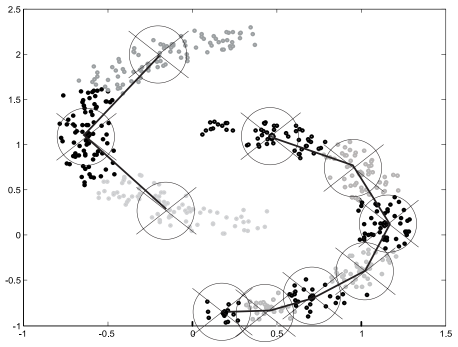

Figure 6: Result of the linkage of centers based on the threshold distance d_t

The number of classes found by the algorithm will be equal to the number of sets of linked auxiliary centers. In the worst case, the number of classes will be equal to the number of auxiliary centers. To classify a point out of the data set, we find which auxiliary center it belongs to (using the Euclidian metrics) and then which cluster that center belongs to. That cluster will be the cluster which that point belongs to. The outline of the algorithm is shown as follows:

1. *Choose N_a and d_t for the data set that we want to classify*

2. *Calculate the N_a centers using a competitive neural network.*

3. *For all centers* $\mathbf{c}_i$*, compute the distance* $d_{ij} = d_m(\mathbf{c}_i, \mathbf{c}_j, \hat{\mathbf{C}})$ *from itself to all others (*$i \neq j$*), where* $\hat{\mathbf{C}}$ *is the estimate of the covariance matrix of the data of the class where the center is* $\mathbf{c}_i$*, mixed with data where the center is* $\mathbf{c}_j$*.*

4. *If for a pair of centers* $\mathbf{c}_i$ *and* $\mathbf{c}_j$*, the distance* d_{ij} *is less than* d_t*, link* $\mathbf{c}_i$ *with* $\mathbf{c}_j$*.*

As we can see in the presented algorithm, the final result depends on the parameters N_a and d_t. The choice of the number of auxiliary centers N_a is less critical because it is sufficient to put a large number of centers until it represent the spatial tendency of the data set. The implication of choosing a very large number of auxiliary centers is the computational cost since, for each center, the *Mahalanobis* distance to all others centers have to be measured. The threshold distance d_t is an empirical parameter and it depends on the spatial distribution of the data set. A good way to choose the threshold is to normalize the data set before its classification.

After the linkage of the centers, each class can be modeled with a statistical model. The data is modeled by using a Gaussian mixture for each class. A Gaussian mixture is composed by a weighed sum of single Gaussians as showed in equation (4).

$$P(\mathbf{x}) = \sum_i P_i N(\mathbf{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{4}$$

where $P(\mathbf{x})$ is the probability of $\mathbf{x}$ to belong to the class that is being modeled. P_i is the weigh of each Gaussian $N(\mathbf{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the mean vectors and the covariance matrixes that models the class. In that model, the gaussian's parameters are easily obtained from the linked centers. The mean vectors are the auxiliary centers. The covariance matrixes are obtained when calculating the *Mahalanobis* distance. The prior probabilities P_i of each Gaussian are obtained from the proportion of number of points assigned to each auxiliary center of the total number of centers in the class.

The main difference between this algorithm and the conventional statistical modeling by Gaussian mixtures is that in the conventional modeling, the entire data set is represented by one mixture. In the proposed algorithm, the data set is represented by several mixtures, corresponding to several clusters in the data set. Differently from conventional algorithms for Gaussian mixtures as the Expectation-Maximization[15], the proposed algorithm has a low computational cost, and find one mixture for each cluster in the data set.

In some clusters algorithms, the main objective is the optimization of functions of matrixes, known as within-class and between-class variance matrixes. Those matrixes can be calculated as follows[10].

$$\begin{aligned} W &= \frac{1}{n-k} \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)(x_{ij} - \bar{x}_i)^t \\ B &= \sum_{i=1}^{k} n_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^t \end{aligned} \tag{4}$$

where W is the within-class variance matrix and B is the between-class variance matrix. x_{ij} is the *j-th* data belonging to the class *i*, $\bar{x}_i$ is the center of the class and $\bar{x}$ is the total mean (center) of the whole data set. *n* and *k* are the total number of points and total number of class. In general, the algorithms cluster the data in order to minimize the trace of *W* corresponding to minimizing the sum of squared distances to the classes centers[10]. The use of that measure (trace of *W*) is well suited when the number of classes is known (as in algorithms like k-means). In this work, that approach is only useful as an estimate of variance, since the minimization of *W* will occur when all centers are unlinked.

4 - Results

The proposed algorithm was tested in many data sets that have a complex spatial distribution. The tests where made with bi and three-dimensional data sets, however the algorithm can be used in data sets with higher dimensions. For comparison purposes, the sets were also classified with pure competitive neural networks. The figures 7 to 16 shows the data sets and the results obtained with the regular neural network and with the proposed algorithm. The figure 7 shows the classification of a non-linear separable data set with the neural network, in which there is a wrong classification of some points because of Euclidian metrics. In figure 8 we have the classification with the proposed metrics, for $N_a = 10$ and $d_t = 100$. Figure 9 and 10 show the same procedure but with a data set more difficult to separate. In Figures 11 and 12 the algorithm is compared in a mostly interlaced data set. In figures 7 and 8 illustrate a non-linearly separable data set. As we can see, the classification provided by the classical cluster with competitive neural network is mistaken. With the proposed algorithm, the data is separated correctly and the numbers of classes are not given previously. Figures 9 and 10 show one more example with a slight non-linearity. Note that even with a small non-linearity the classical method does not work. Finally in figures 11 and 12 a high non-linearly separable data set is tested and the classification is obtained correctly.

The choice of the Na is not critical. The algorithm works fine with a wide range of values, using values close to 2% or 5% of the total number of points worked well in all tests. The choice of dt is more critical than the choice of Na. The heuristic used to choose dt was the stability of the number of classes found. As the threshold is increased, the number of classes found by the algorithm is changed. The value chosen is the one that produces the most stable number of classes found in a wide range of values. That heuristic worked well for all data sets tested.

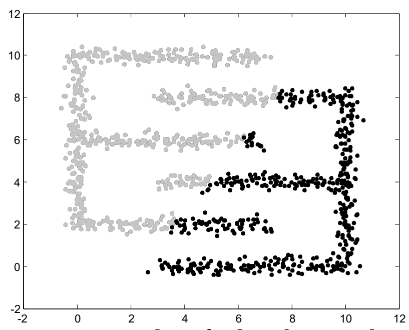

Figure 7: Data set classified with *neural network*

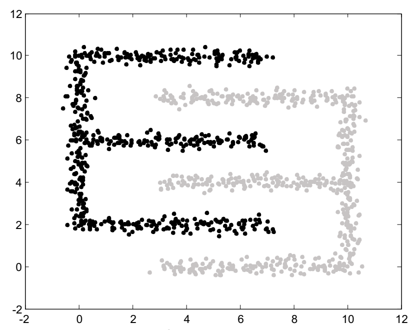

Figure 8: Data set classified with the proposed algorithm ($N_a = 10$ and $d_t = 100$)

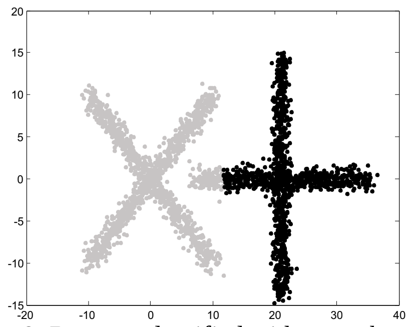

Figure 9: Data set classified with *neural network*

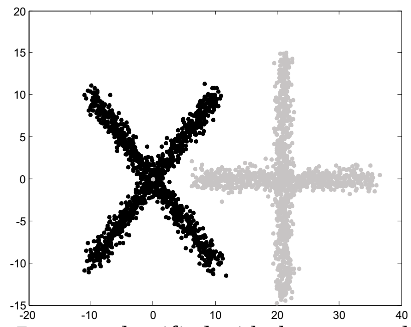

Figure 10: Data set classified with the proposed algorithm ($N_a = 15$ and $d_t = 100$)

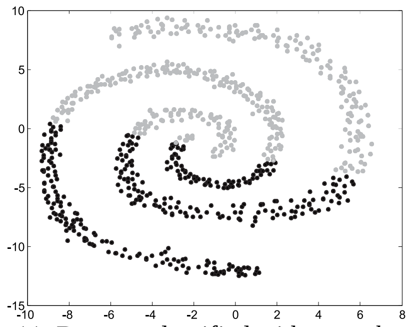

Figure 11: Data set classified with *neural network*

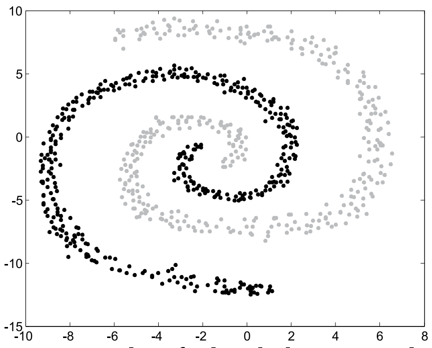

Figure 12: Data set classified with the proposed algorithm ($N_a = 60$ and $d_t = 100$)

For illustration purpose, the algorithm was also tested in three-dimensional data sets as complex as the one presented in figures 13 to 16. As can be seen, the algorithm classifies the data correctly, also finding the number of classes. The rings example is a generalization of the classical "two rings" data set. This specific data set is very used to test classification algorithms[7] since it is non-linearly separable. In figure 15, is illustrated a 3D high non-linearly separable data set composed by two classes. Again the algorithm separated the classes correctly.

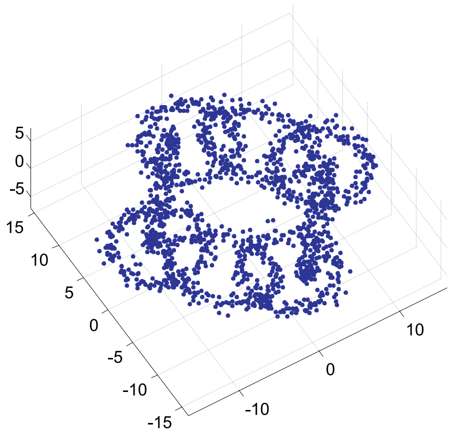

Figure 13: Original data set (eight set of points forming eight rings chaining each other)

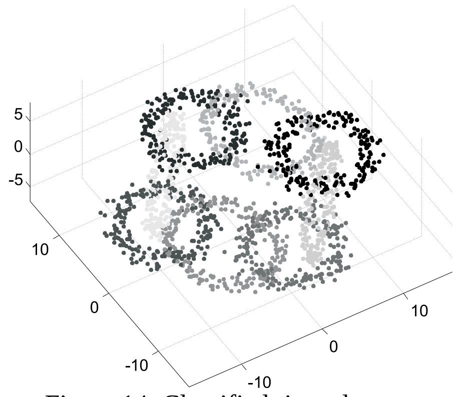

Figure 14: Classified rings data set

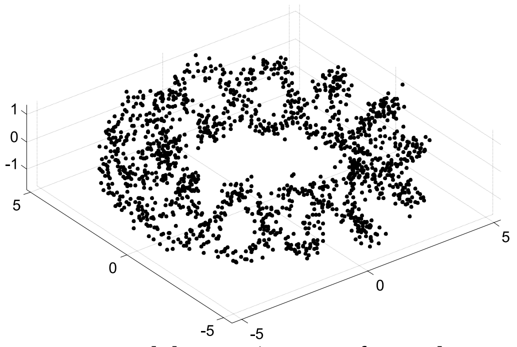

Figure 15: Original data set (two set of tentacles crossing each other)

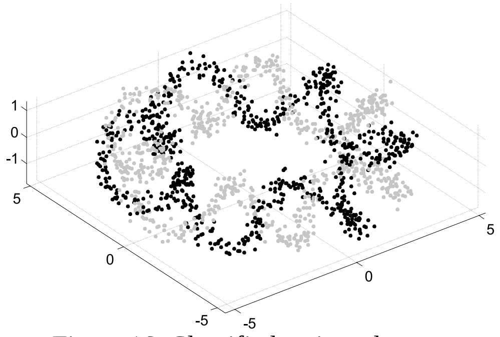

Figure 16: Classified springs data set

Figures 17 and 18 illustrate the statistical model obtained by the algorithm. The color gradient represents the probability of belonging to each class. Different gradient colors represent different classes. In figure 17 a non-linearly separable data set composed by two classes is presented. The algorithm generates two probability distributions (two gradient colors) and each point in the data set is tested. The contour, in which can be noticed high change in color, is exactly the decision boundary of the classifier. In figure 18, the same test, with more complex patterns in a high non-linearly separable data set, is executed. Again, the colors represent the classes and the gradient represents the probability. As can be seen, the decision boundary follows the distribution of the two classes.

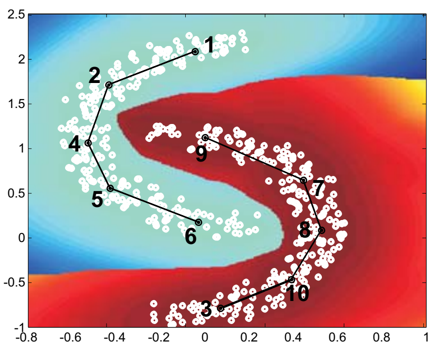

Figure 17: Statistical model obtained by the algorithm in a non-linearly separable data set

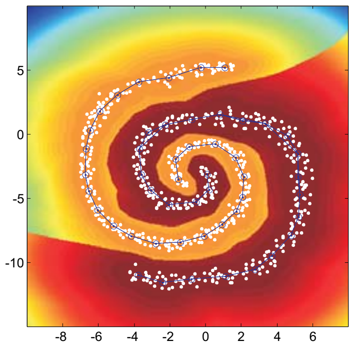

Figure 18: Statistical model obtained by the algorithm in a high non-linearly separable data set

5 - Conclusions

The proposed algorithm separates data in an automatic way without having to provide the number of classes beforehand. This kind of problem hasn’t been exhaustively explored and doesn't have a completely efficient solution. As presented in the results section, the algorithm was efficient when applied to the classification of very complex data generated artificially. Good results still depend on the choice of suitable d_t and N_a, that can represent well the statistical features of the data, allowing the correct linking of the auxiliary centers. The choice of N_a is not critical because the number of classes is always overestimated and the algorithm will link the correct centers. The choice of d_t will cause differences in the number of classes if incorrect values are used. For the data sets tested, the heuristic for choosing d_t showed itself robust. Small changes (about 10% even more) in correct value do not affect the classification. Generally, the algorithm is an efficient alternative tool to be applied to problems where there is a spatially complex distributed data and several tests to d_t and N_a values can be performed. The examples used in the results section have two and tree dimensions only for illustration purposes, the algorithm can be used in high dimensions data sets normally. The results shown in figures 17 and 18 show that the statistical model obtained fits very well in the data set and, usually, the classification error is very small. Finally, the algorithm can be applied to several problems in several areas, such as data mining, pattern recognition, signal and image processing or any problem that involves cluster analysis and data classification.

6 - Acknowledgment

We want to thank Dr. Pablo Javier Alsina and Dr. Francisco das Chagas Motta for the precious comments and suggestions.

7 – References

[1] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics*, 1, pp. 281-297, AD 669871, Univ.of California Press, Berkeley, 1967.
[2] S. Haykin. *Neural Networks a Comprehensive Foundation*. Prentice Hall, second edition, 1999.
[3] Jianchang Mao e Anil K. Jain. *A self-organizing network for hiperellipsoidal clustering*. IEEE Transactions on neural networks, volume 7, 1996
[4] Jou, C, Quen-Zong W., Shuh-Chuan T., Yuh-Jiuan T. e Shih-Shien Y. *A hyperellipsoid neural network for pattern classification*. IEEE International Symposium on Circuits and Systems, volume 2, 1991
[5] T. Kohonen. *Self-Organization and Associative Memory*. Springer Verlag, 3 edition, 1989.
[6] Z. Huang and M. K. Ng. A fuzzy k-means algorithm for clustering categorical data. *IEEE Transactions on Fuzzy Systems*, 7, 1999.
[7] J. A. Costa. *Calssificação Automática e Análise de Dados por Redes Neurais Auto-organizáveis*. PhD thesis, UNICAMP, 1999.
[8] A.Ultsch e C.Vetter. *Self-Organizing-Feature-Maps versus Statistical Clustering Methods: A Benchmark*. FG Neuroinformatik & Künstliche Intelligenz, 1994
[9] H. Lipson e H. T. Siegelmann. *Clustering Irregular Shapes Using High-Order Neurons*. Neural Computation, volume 12, 1999
[10] B. S. Everitt. *Cluster Analysis*. Arnold, 1993.
[11] H. H. Bock, *Automatische Klassifikation*. Vandenhoeck & Ruprecht, 1974.
[12] A. Papoulis. *Probability, Random Variables and Stochastic Processes*. Mc Graw Hill, 3 edition, 1991.
[13] Grossberg, S. *Adaptive pattern classification and universal recording: I. Parallel development and coding of neural detectors.* Biological Cybernetics, vol. 23, pp. 121-134.
[14] Späth, H. *Cluster Analysis Algorithms: for data reduction and classification of objects.* Ellis Horwood: Chichester, West Sussex, England.
[15] A. P. Dempster, N. M. Laird and D. B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society Series B, vol. 39, no. 1, pp. 1--38, Nov. 1977.