International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:4, No:9, 2010

Hierarchical Clustering Analysis with SOM Networks

Diego Ordóñez, Carlos Dafonte, Minia Manteiga, Bernardino Arcay

Abstract—This work presents a neural network model for the clustering analysis of data based on Self Organizing Maps (SOM). The model evolves during the training stage towards a hierarchical structure according to the input requirements. The hierarchical structure symbolizes a specialization tool that provides refinements of the classification process. The structure behaves like a single map with different resolutions depending on the region to analyze. The benefits and performance of the algorithm are discussed in application to the Iris dataset, a classical example for pattern recognition.

Keywords—Neural networks, Self-organizing feature maps, Hierarchical systems, Pattern clustering methods.

I. Introduction

THE input of one SOM([9], [8]) can be taken from the output of another. The input can also be formed by several output vectors from many SOMs. This kind of structure is referred to as a hierarchical SOM. The topmost SOM in the structure clusters the outputs and provides a means of monitoring the operations of the underlying SOMs. As the SOM hierarchy is followed upwards, the information becomes increasingly abstract.

The classical idea of a hierarchical structure for self-organizing maps is generally based on the connectivity of several self-organizing layers that form a larger network. Each layer or map of this structure represents a higher level of abstraction with regard to preceding layers in the hierarchy ([2], [5], [3], [1]). The usual structure of hierarchical SOM networks contains two layers ([2], [1]). In this sense, [4] proposes a structure for the self-organized map that starts with a single SOM network; the map is labeled using training data and nodes that do not produce a single output are replaced with a submap, resulting in a map with irregular connections.

More recent works suggest a structure of self-organizing maps arranged in a tree ([6]): these structures allow us to adapt the topology of each hierarchy layer to the characteristics of the training set. The training of this type of structure occurs in two stages: the first stage consists in the construction of the hierarchical structure of self-organizing maps, the second stage provides information about the classes at each level of the tree. The present paper proposes a new technique to train this type of structure, a criterion for stopping the expansion of the tree, the combination of classes provided by the unsupervised training algorithm, and quality criteria of the obtained classifications.

Diego Ordóñez, Carlos Dafonte, Bernardino Arcay are with the Department of Information and Comunications Technologies, Faculty of Computer Science, University of A Coruña, 15071, A Coruña, Spain

Minia Manteiga is with the Department of Navigation and Earth Sciences, University of A Coruña, 15011, A Coruña, Spain, e-mail: manteiga@udc.es

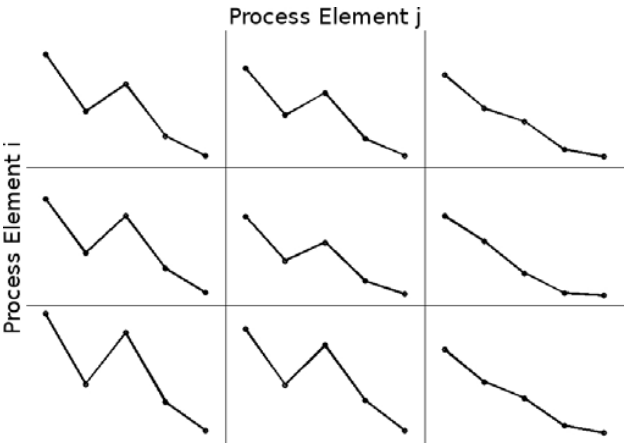

Fig. 1. Models represented by each of the neurons. Each signal represents what was stored by a neuron from an input determined by the weights of the process elements at the network input for the example of Iris dataset.

Self Organizing Maps (SOM) are an effective tool to visualize data with large dimensions. They transform the data that belong to the inputs space by means of an information compression process that preserves the most important topological relationships of the data and results in the abstraction of the inputs. The relationships between the input data are established through the topological bounds stored in the connections at the network output, which represents models of subsets of the input patterns (see Figure 1).

After the learning process, we must analise the training result of the SOM to apply it to a concrete domain. This task depends on the application domain and focuses in many cases on detecting regions in an output map. These regions form object groupings that represent a taxonomy of the input vectors. The process elements of the map represent models of subsets or samples of the inputs ([8]), and through the neighbourhood relationships the map relates similar models that represent the same categorisation or subcategories.

Two input data belonging to the same object class activate process elements in a SOM that are nearby in the topological disposition of the map. This idea intuitively lead us towards an initial analysis that considers the division of the map in regions that could represent subcategories in the input domain. This analysis could be recursively applied resulting in a hireachical structure. A common problem consists in discerning the frontiers between the different taxonomies and doing so automatically.

There are usually no abrupt transitions among the regions

that represent the categories in the map, but there are process elements that represent something intermediate between two or more object classes, i.c. transition elements, which lead us to apply some type of fuzzy technique to fuzzify the borders. This problem can be solved by segmenting the SOM with a fuzzy logic technique applied to the weights of the connections with the output elements, the Fuzzy C-Means technique (FCM) ([11],[12]). The Dunn-Bezdeck algorithm provides us with the framework to define the membership levels of the process elements to the clusters and to identify the process elements that represent some of the object categories of the problem as well as those that are intermediate between various different types of the taxonomy.

Once trained, the network generates a series of models that try to represent the input vectors. Each model is characterized, through the weights of the output neurons, by the inputs. Figure 1 shows an example of these models. We can analyse the output of the network by taking the input models as points of reference, and applying a clustering technique to them. This process allows us to find out which categories can be obtained in the map (map analysis). In the course of a SOM analysis, we find areas with process elements that clearly represent some of the categories recognised for a particular problem. However, certain process elements will correspond to transition regions, areas in the map that are intermediate representations between different categories. In general, these regions fill up the voids left by other areas that do represent specific categories. The technique used to explore their nature consists in refining the represented models with the neurons of those map areas, as will be described in Section IV. This approach will also be used to deal with the case that arises when a model represents a combination of different types of categories of objects.

The following sections provide a detailed description of the algorithm, opportunities, and application examples. Sections II and III separately present the tools that were used to develop the algorithm, i.c. the SOMs and the FCM algorithm. Sections IV and V represent the steps to construct the information structure that supports the algorithm:

- Section IV: automatic generation of the hierarchical structure
- Section V: segmentation knowledge extraction by means of a labelling process

Section VI describes how to quantify the quality of the I accept the terms of the following Honor Code and Plagiarism Statement for Paper Submission, and that the paper is original research contribution with the references properly cited in the manuscript. I accept the terms of the following Copyright Agreement, and on submitting the full text paper for possible publication to World Academy of Science, Engineering and Technology; I confirm, acknowledge and agree expressly to the terms and conditions of this copyright agreement. This copyright agreement prevails and is binding to the contributing author(s). classification by evaluating the membership function with regard to each of the nodes followed during the classification process. Section VII presents an example of application, as well as the obtained results. Finally, Section VIII comments upon the most relevant features of this work.

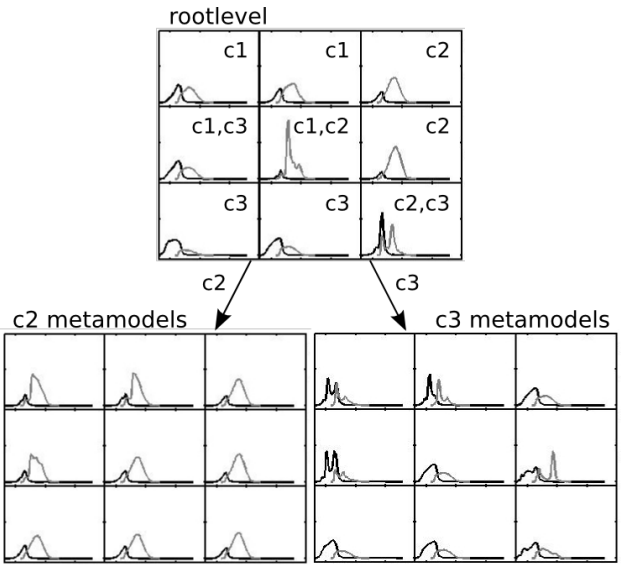

Fig. 2. Hierarchical decomposition of the classification in more detailed maps

II. Architecture

The proposed architecture uses a tree structure in which each node represents a SOM network. The hierarchical disposition of the nodes indicates different levels of finetuning in the classification (see Figure 2), going from general models to specific models, increasing the detail level as we descend through the hierarchy. The hierarchical structure of the SOMs is a problem that was tackled in certain studies ([13], [14]). As mentioned above, this work focuses on the problem from a perspective that combines the hierarchical structure and fuzzy logic techniques ([15], [11]). Fuzzy logic is the support tool that allows us to descend in the hierarchical structure that is built around the SOMs.

Figure 2 shows a graphic simplification of the structure. We dispose of a map of a higher hierarchical level (rootlevel) where the neurons represent general models. After identifying the map areas with the analysis of the neuron connections, we segment the general map and finetune the models so as to give space to the metamodels of the inferior level. This process is applied recursively and represents a selective increase of the resolution in certain areas of the map. For this task we rely on a fuzzy logic algorithm that allow us to automatically identify the areas of interest of the map, as shall be shown in Section III.

III. Self-organized maps and Dunn-Bezdeck algorithm

In order to carry out the automatic analysis of the map, we need a technique that allows us to classify, in an unsupervised way, the models that were obtained during the network training. Section II mentioned the fact that there will be regions in the map that represent categories of the problems taxonomy, as well as groups of neurons that represent intermediate (transitional) categories. In such a scenario, it would be highly

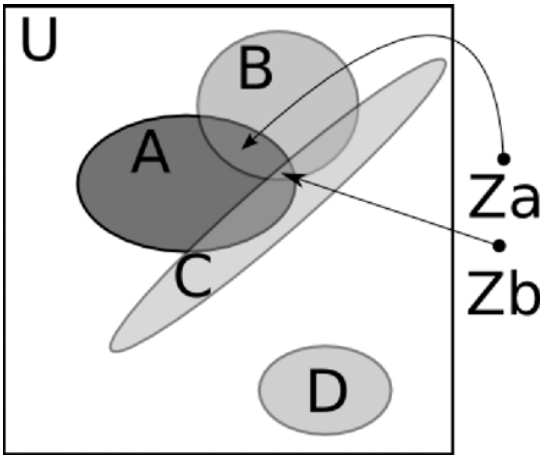

Fig. 3. Metamodel intersections. There are several zones on the map where cluster action radious overlaps. U stands for all models

interesting to know the category to which a certain process element of the network belongs as well as its membership level. For this analysis task we rely on the FCM technique.

This technique consists in a clustering method that makes it possible for an input example to belong to more than one grouping. The model was developed by Dunn in 1973 ([12]) and improved by Bezdek in 1981 ([11]), and is frequently used for pattern recognition. It is based on the minimisation of the following objective function:

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} U_{ij}^{m}\|x_i - c_j\|^2 \quad (1)$$

where m is any real number above 1, U_{ij} is the membership level of x_i to the cluster j, x_i is the ith example of the patterns that must be grouped, c_j is the centre of the cluster of class j, and $\|*\|$ is any norm function that expresses the similarity between the observed data and a cluster center.

The fuzzy partition is carried out with an iterative optimisation of the objective function shown in 1, through the update of the membership function U_{ij}, and the cluster centers.

Consequently, the FCM evaluates the membership level to each category of every network model generated during the training stage. These categories do not correspond $1:1$ to those of the problem in question, since we are working with unsupervised techniques. Section V will comment upon the mapping of the unsupervised categories to those of the problem in the course of a post-processing phase.

When applying this technique, the transition elements are those that do not have an acceptable membership level for any of the clusters. The criterion for the membership of a model to a cluster is based on a threshold (λ). The choice of an adequate threshold makes the algorithm more efficient because, as will be seen in the example of Section VII, less finetuning is required to reach the envisaged result.

Figure 3 shows one of the problems that appear during thresholding: some areas on the map cannot be assigned with certainty to one of the groups, because the membership is distributed among two or more groups. In this case, the proposed solution consists in generating new metamodels that surge from the combination of others; so if we configurated the FCM algorithm with 3 clusters, we would have clusters C_1, C_2, and C_3; but also, depending on the situation, we would have combinations of two clusters (C_1,C_2), (C_1,C_3), (C_2,C_3), and even the combination of all of them (C_1,C_2,C_3). Therefore, the finetuning process of a hierarchical node could, potentially, generate $2^n - 1$ nodes in the following level of the hierarchical structure; the more restrictive the threshold, the more nodes will be generated.

IV. Construction of the Hierarchical Structure

Previous sections separately presented the tools that will be used to developed the algorithm: the SOMs and the FCM algorithm. This section describes the automatic generation of the structure.

Starting from the representation of a dataset that belongs to a specific domain $P = x_1, x_2, \ldots, x_n$, where each x_i belongs to $\Re^n$, we train a SOM network with the Kohonen algorithm ([10]). The training of this initial network results in the representative models of the input in the shape of weights vectors. These weights vectors (models) will serve as an input for the FCM clustering algorithm that will build the so-called metamodels. That information is used to segment the set of input vectors by trying to identify the metamodel that best represents each vector. Once the segmentation is terminated, a new SOM network is generated for each subset of the inputs, and the training is carried out. On the basis of the set of vectors in P, we generate the sets P_1, P_2,... P_m, so that P_1 intersection P_2 is the empty set.

This is an iterative process in which each level represents a finetuning stage of the classification. This approach has two consequences:

- In each node, the neural networks are reduced in size (considering complex problems that require a map with many nodes), which means that the search of the winning element is faster in each level of the tree. The search time of a winner increases exponentially with the size of the map.
- By gradually specializing the training, there will be less and less transition process elements that represent no subset of vectors at the input.

The more levels the structure has, the more detail we obtain; the question is to know how many levels we need for a specific problem. The proposed halting criterion considers the fact that the specialization that takes place in the models when they descend the hierarchical levels makes those models become increasingly similar, and as a consequence the variance becomes smaller. By calculating the variance in each dimension of the input, we have a measure of how much the input vectors resemble each other in each dimension. When one of the iterative finetunings generates a node for which the classified patterns generate a vector of variancies, and none of the dimensions has a significant variability (the patterns are practically identical), continuing the expansion of the tree will not entail any benefit whatsoever. This halting criterion requires the definition of a set of threshold values while

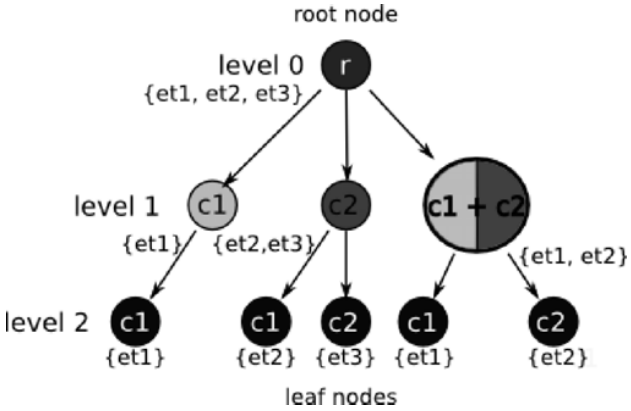

Fig. 4. Hirearchy, nodes and labels

considering the dimensions of the problem that allows us to take the decision. This value will determine the detail level that we wish to achieve.

Section III described one of the problems that arise during thresholding: there will be examples for which none of the clusters has a fuzzy value above that threshold. In that case, we opt for adding new nodes that represent combinations of clusters. Naturally, the maximum value will always be reached by the combination of all the clusters, but what really interests us is to find the minimal set of clusters that exceeds the threshold.

V. Labeling

Since both the generated structure and the SOMs are unsupervised, we have to train the network and subsequently analyse it to extract the knowledge of the segmentation. Once the structure is built according to the steps that were traced in Section IV, it would be interesting to know what is represented by each group (nodes of the hierarchical structure or metamodels) and map the unsupervised categories to those of the problem. Domain knowledge can be added to the structure by means of a labeling process. Labeling consists in classifying a known set of objects with the generated structure and assigning the labels associated to each object to the groups that classify it through the different tree levels, from the roots to the leaves.

It is preferable for a labeling algorithm to maintain a structural coherence of the labels. A father node must contain the labels of all its descendents down to the leaves, so that each descendent node represents some type of specialization of the father node. This way, we provide not only a single classification but also a process that discards object types level after level until it reaches the leaves.

We may find a situation in which one leaf has more than one label: this would mean that there exist two types of objects that are so similar that they cannot be distinguished. This type of classification minimizes the false positives, because it only provides discriminating information when the objects are significantly different.

The labeling algorithm is based on a set of input vectors for which the classification is known in advance. This set can be the same as the one used for the training of SOM networks or a different one. Each example follows a series of steps:

1) Start from the root node.
2) Determine the winning node with the SOM network associated to the current node. Add the label associated to the input to the labels set of the current node.
3) Select the metamodel associated to this process element while taking into account the fuzzy values associated to that neuron through the U matrix of the FCM algorithm.
4) Transit to the node indicated by the metamodel.
5) Iterate steps 2 to 4 until the selected node is a leaf node.

The consequence of this process is that each root node contains all the possible labels and that the classification gradually pinpoints nodes until it reaches the leaf nodes that represent the concrete classifications.

VI. Quality of the Classification and Generation of the Most Reliable Solution

Section III described the fuzzy algorithm for model classification. As mentioned above, the models are linked to each cluster by means of a membership function that represents the fuzzy behaviour of the algorithm. Implementing this behaviour requires us to build an appropriate fuzzy matrix called U (using the FCM algorithm) and represent the level of membership to each of the cluster centers. As a result, if a neuron (model) possesses a membership level close to 1, it provides us with the knowledge that it is clearly different from the other cluster centers. This means that if we follow a node route in the hierarchical structure, where in each node the example has a high level of membership to the cluster in which it was classified, we will consider the classification to be more reliable, whereas if we follow a node route through a sequence of nodes with low membership level, the classification is less reliable.

The advantage of this procedure is that it provides not only a classification but also a quality criterion for the classification, indicating its level of reliability.

Since each node generates a fuzzy value, we need a way to combine them and provide a final membership value. The fuzzy operator of the intersection is minimal ([15]), because two nodes whose sequence represents a path of the tree represent the fact that we have determined the example to belong simultaneously to both fuzzy sets, which is the definition of the intersection. So if $n_1, n_2, \ldots, nk$ represents the nodes sequency and f_1, f_2, f_k are the fuzzy values provided in each node, the confidence criterion in the classification is minimal (f_1,f_2,$\ldots$, f_k).

The classification is obtained by searching in each level for the node that exceeds the search threshold. We must find a way to make sure that that value is the highest among the clusters combinations of all the paths that start from the root node. The resulting fuzzy membership value is maximal. Indeed, if we did not work with clusters combinations, and λ were the membership treshold selected for the algorithm, each node would have a (specifically) associated path from the root node where the accumulated fuzzy value is the smallest of the fuzzy values in each ancestral node and the present value. For $\lambda >$

0.5 and the fuzzy values belong to the $[0, 1]$ interval, we find that by selecting the path of the highest probability and always following the highest value cluster that exceeds the threshold:

1) We reach the end of the nodes sequence and the minimum will never return a value below the threshold, i.e. we always obtain a value above λ;
2) If all the fuzzy values of the individual clusters add up to 1 and $\lambda > 0.5$, the sum of all the other values separately gives a value below the threshold.
3) Taking into account that the minimal function is applied to the path, and the results of points 1 and 2, no other individual node path will exceed the threshold.

We therefore observe that the returned value is the highest of all the path combinations from the root node until the leaves.

This result was obtained under the supposition that there are no clusters combinations, but that could not be the case in a real situation. Apart from obtaining the maximum fuzzy value, we are interested in minimizing the number of combined clusters in order to reach a solution in each level. We are obliged to combine clusters because no other node that represents an individual cluster reaches the threshold value. This is why the demonstration is the same in both cases: when the path that starts from the node has the highest fuzzy value without considering clusters combination of a higher order, and when the nodes represent individual clusters.

VII. Application example: The Iris dataset

In literature, the Iris dataset is probably the best known dataset for pattern recognition. The article by Fisher ([16]) is a classic in the field and references to his work are frequent ([17]). This dataset contains 3 classes of 50 examples each, each class referring to a type of Iris plant. The Setosa class can be separated linearly from the other two (Virgina, Versicolor), which, in turn, cannot.

The attributes that describe each example are represented by four parameters:

1) Sepal length in cm
2) Sepal width in cm
3) Petal length in cm
4) Petal width in cm

The measurements are extracted from each example and used as input vectors for the algorithm to compose the patterns set. Starting from this set of a total of $50x3 = 150$ examples, we arbitrarily separate 10 examples from each class and keep $3x10 = 30$ patterns that will be used to validate the algorithm and 120 patterns to build the structure and train the SOM networks.

The configuration of the algorithm in each node of the hierarchical structure is carried out with SOM networks of $5x5$ (25) process elements disposed in a rectangular map. This will provide us, during the application of the FCM algorithm to the resulting models, with a matrix of $25xN$ fuzzy values, in which each row of N values adds op to 1. The number N is determined by the number of clusters with which the FCM algorithm was configured. The threshold that was defined to determine the membership of a model to a cluster in the algorithm is $\lambda = 0.7$, a value that was chosen empirically after various tests and is able to generate efficient structures for this example with regard to the number of nodes and the tree levels. More restriction for this parameter would make most examples group into clusters combinations; less restriction would increase the probability of grouping object types of a different nature into one and the same cluster.

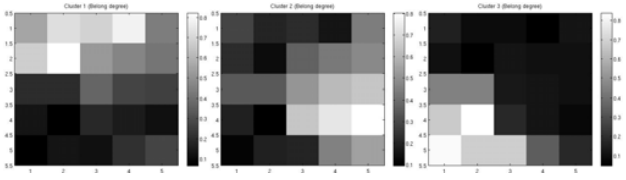

Fig. 5. Membership values for each process element of the root node and each cluster used to configure the Fuzzy C-Means algorithm for the case in which three clusters are considered.

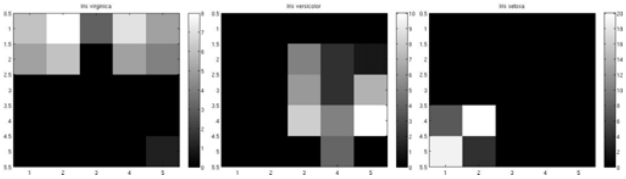

Fig. 6. Activation frequencies for each process element of the root node for each plant type of the Iris set.

The SOM networks used to configure each node were trained with a hextop topology ([18]); the number of training stages was 200, an amount that is sufficient for the present example because the training shows stabilisation before reaching that number. If the root node is considered to be level 1 of the tree, and the tree has 3 levels, we only have to descend a maximum of 2 levels to obtain the classification.

The following section shows various aspects of the algorithm and its usefulness in solving classification problems and checking the influence of some parameters on the global functioning and output. Table I shows the complexity of the tree and the efficiency of the global algorithm, depending on the number of clusters that configure the FCM algorithm that classifies the models. We can observe that the most efficient configuration with regard to the number of nodes is that of three groups, which coincides also with the number of different object types of Iris plants. Figures 5 and 6 show that the Fuzzy C-Means algorithm forms metamodels that resemble the projection made by the learning algorithm of the SOM for the different object types.

Another relevant aspect is that the number of groups formed by the Fuzzy C-Means does not affect (for this example) the success rate of the algorithm, which is 100% in all cases. Even though the configuration with two clusters provides for all cases one single correct label, the number of levels of the generated tree makes the structure very inefficient during classification.

Table I shows the success rate and the complexity of the tree according to the number of clusters (columns in the table) with which we configure the FCM algorithm after having labeled the structure. The % of single label successes represents the number of times the algorithm provides a single classification

TABLE I
ANALYSIS OF THE SUCCESS RATES AND THE COMPLEXITY OF THE TREE ACCORDING TO THE NUMBER OF CLUSTERS (N.C.).

	N.C. = 2	N.C. = 3	N.C. = 4
N. nodes	130	78	106
N. leaf	87	67	99
N. levels	20	5	4
% accuracy one class	100	100	100
% accuracy two classes	-	100	100
% accuracy global	100	100	100
% single class	100	90	93
% double class	0	10	7

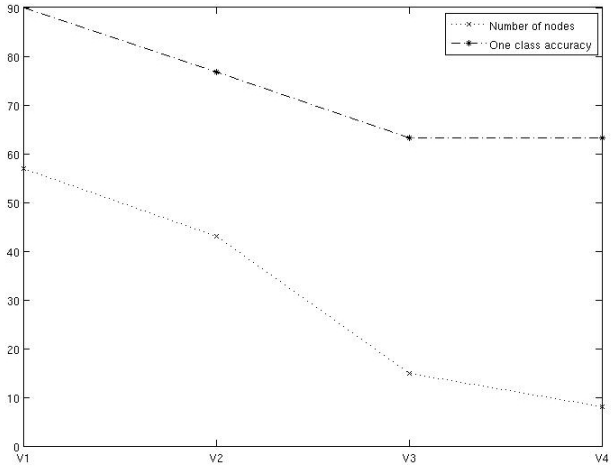

Fig. 7. Effect of the variance threshold in the number of nodes and the classification into one class

for a given input vector; the % of double label successes is the number of times the algorithm provides two different labels for an input example and one of them is correct.

The results of Table I were obtained by cutting the expansion of the tree when the variance of the input patterns assigned to a node does not reach the threshold variancies, which in this case are $V = (0.01, 0.01, 0.01, 0.01)$. These low values give us an idea of the algorithm s capacities when taken to a limit, i.e. when taken to expand until the patterns represented by a given node are partically identical. A more detailed analysis of this threshold will allow us to check how it affects the results and the configuration of the algorithm architecture. We show what happens when the variance threshold is made increasingly restrictive. The considered variance thresholds are the following:

1) $V1 = (0.05, 0.05, 0.05, 0.05)$
2) $V2 = (0.1, 0.1, 0.1, 0.1)$
3) $V3 = (0.2, 0.2, 0.2, 0.2)$
4) $V4 = (1.0, 1.0, 1.0, 1.0)$

Figure 7 shows the effect of the variance threshold, with great similarities between the successrate for a class and the number of nodes. The number of tree levels for each case also tends to descend $(5, 5, 3, 2)$.

A graphic example of this problem in the case of the Iris dataset can be seen in Figure 8: we see each of the training patterns with its two first dimensions (sepal length and sepal

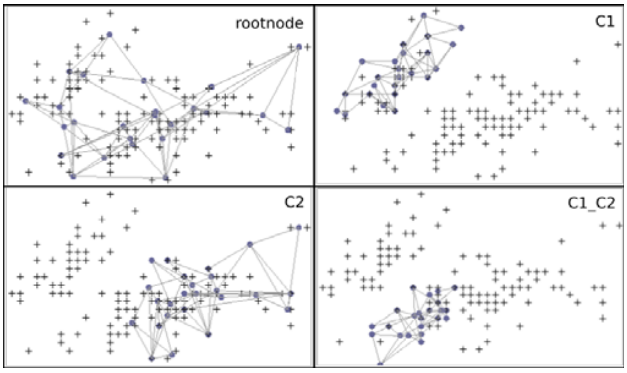

Fig. 8. SOM Topology distribution in over two dimensions of the input dataset (sepal length and sepal width)

width) and the disposition and topology of the SOM process elements in the same space. Figure 8 was splited in four parts, the upper left figure shows the network of the root element, this network try to cover the whole input space. The upper right and lower left show networks of the next level, wich specializes in different pap areas. Finally, the lower right Figure shows the network that is specialized in those input patterns for which the membership level to clusters 1 and 2 has not reached the threshold.

VIII. CONCLUSIONS

This work presents a methodology for the SOM-based classification. The analysis of the topological disposition of the models represented by the weight vectors allows us to segment and isolate iteratively the different object types. We subsequently label the clusters, adding supervised characteristics to the algorithm (concrete object types hidden by each group).

As a result, the algorithm gradually specializes with the levels of the tree structure that represents it. More depth is generated in the structure as more detail is needed to divide the patterns set. The complexity of the information structure on which the algorithm relies is controlled by various threshold values: the threshold variancies, the minimal level of group membership, and the number of groups formed by the FCM algorithm. The adequate selection of these parameters depends on the domain that is to be treated, the separation between the different object classes, and the number of examples of each class. The variancies and the membership threshold control the depth of the tree; less restrictions in this aspect imply less depth and less computational cost for the execution of the algorithm.

The number of clusters of the FCM algorithm has its impact on both the depth of the tree and the width of each level, increasing exponentially the global number of nodes with each new level. The number of SOM networks of the structure is not equal to the number of total nodes, since only one SOM is required for each node that expands. Table I shows that statistically most nodes of the structure are leaves, which means that the number of networks will be largely inferior to the number of nodes of the structure. The network is

necessary for the expansion of a node. The maximal number of steps needed to classify an object coincides with the maximal number of tree levels minus one. For the example with 4 clusters (Table I), a total of 106 nodes and 4 levels, classifying an object only requires computing the corresponding model of 3 levels of $5x5 = 25$ process elements in intermediate nodes and checking the membership levels in each level of the fuzzy matrix. In a network with a rectangular map of NxN process elements, the complexity of the search of the winning element, using the Big-O notation, will be $O(m*N^2)$, which means that the complexity grows quadratically with the map size. If the problem is sufficiently complex, the computational complexity grows quadratically according to the growing number of process elements in the output. In terms of complexity, it is therefore more efficient to chain smaller maps to reduce the number of floating point operations needed to carry out the classification.

The structure is generated on the basis of a static model of neural network or SOM, but thanks to the iterative way of proceeding neither the initial size of the map nor the number of clusters of the fuzzy algorithm are decisive in obtaining good classification. The example shows how different choices in the number of clusters generate a more or less efficient structure in terms of memory and computational cost, but yielding good results in all cases.

The strategy proposed by this algorithm is very conservatory, because it only generates selective classifications for examples represented by models that are sufficiently different from each other. In any other case, the response of the algorithm will be the set of labels associated to input vectors that are sufficiently similar to be considered one single group. This criterion is selected by the user through the vector of threshold variancies.

Assuming a certain cost to maintain the entire structure in the memory, HSC is a robust algorithm that minimizes the error rates and, for a given classification, provides criteria to judge the reliability of the classification. It also provides hierarchic information on how the clusters and similarity measures between them are decomposed by means of specialization and generalization processes of the object groups. Labeling is a process that equips the algorithm with supervised clustering capacities, even though the SOM networks were not originally designed to that effect.

The Iris example VII shows that the applications of this algorithm can be of all kinds. It was specifically designed to provide information in cases with discriminating information that distinguishes one object from another, based on the information that is available in the training set, reducing to residual values the number of erroneous classifications. These statements depend on the presence of a domain-representative training set.

ACKNOWLEDGMENT

Spanish MEC project ESP2006-13855-CO2-02

REFERENCES

[1] M. Endo, M. Ueno, T. Tanabe, "A Clustering Method Using Hierarchical Self-Organizing Maps", *Journal of VLSI Signal Processing*, vol. 32, pp. 105-118, 2002.

[2] J. Lampinen, E. Oja, "Clustering Properties of Hierarchical Self-Organizing Maps", *Journal of Mathematical Imaging and vision*, vol 2, pp 261-272, 1992.

[3] P.N. Suganthan, "Pattern Classification Using Multiple Hierarchical Overlapped Self-Organising Maps", *Pattern Recognition*, vol. 34, pp. 2173-2179, 2001.

[4] S.B. Cho, "Neural Network Classifiers for Recognising Totally Unconstrained Handwritten Numerals", *IEEE Trans. Neural Networks*, vol. 8(1), pp. 43-53, 1997.

[5] S.P. Luttrell, "Image Compression Using a Multilayer Neural Network", *Pattern Recognition Letters*. vol. 10, pp. 1-7, 1989.

[6] A. Forti, G.L. Foresti, "Growing Hierarchical Tree SOM: An Unsupervised Neural Network with Dynamic Topology", vol. 19, pp 1568-1580, 2006.

[7] T. Kohonen, E. Oja, O. Simula, A. Visa and J. Kangas, "Engineering applications of the self-organizing map", *Proceedings of the IEEE*, vol. 84(10), pp. 1358-84, October 1996.

[8] T. Kohonen, "Self-organization and associative memory", *Springer-Verlag New York, Inc*, New York, 1989.

[9] T. Kohonen, "Analysis of a simple self-organizing process", *Biological Cybernetics*, vol. 44, pp. 135-140, July 1982.

[10] T. Kohonen, "Self Organizing Maps", *Springer*, Berlin, 1995 (Third, Extended Edition 2001).

[11] J.C. Bezdek, "Pattern Recognition with Fuzzy Objective Function Algorithms", *Kluwer Academic Publishers*, Norwell, MA, USA 1981.

[12] J.C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters", *Cybernetics and Systems*. vol. 3, pp. 32-57, 1973.

[13] P. Koikkalainen and E. Oja, "Self-organizing hierarchical feature maps", *International Joint Conference on Neural Networks*, vol. II, pp. 279-285, Piscataway, NJ 1990.

[14] P. Koikkalainen, "Progress with the tree-structured self-organizing map". *Proceedings of ECAI'94, 11th European Conference on Artificial Intelligence*, pp. 211-215, New York 1994

[15] L.A. Zadeh, "Fuzzy Sets", *Information and Control*, vol. 8, pp. 338-353, 1965.

[16] R.A. Fisher, "The use of multiple measurements in taxonomic problems", *Annual Eugenics*, vol. 7, pp. 179-188, 1936.

[17] R.O. Duda and P.E. Hart, "Pattern Classification and Scene Analysis", *John Wiley and Sons*, 1973.

[18] H. Demuth and M. Beale, "Neural Network Toolbox: For use with MATLAB: User's Guide", *The Mathworks*, 1993.