[6] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives," *Pacific J. Math.*, pp. 1–3, 1966.
[7] D. G. Luenberger, *Linear and Nonlinear Programming*. New York: Addison-Wesley, 1989.

Automated Labeling for Unsupervised Neural Networks: A Hierarchical Approach

Roberto Tagliaferri, Nicola Capuano, and Giorgio Gargiulo

***Abstract*— In this paper a hybrid system and a hierarchical neural-net approaches are proposed to solve the automatic labeling problem for unsupervised clustering. The first method consists in the application of nonneural clustering algorithms directly to the output of a neural net; the second one is based on a multilayer organization of neural units. Both methods are a substantial improvement with respect to the most important unsupervised neural algorithms existing in literature. Experimental results are shown to illustrate clustering performance of the systems.**

***Index Terms*—Automated labeling, clustering, hierarchical neural networks, hybrid neural networks.**

I. Introduction

Unsupervised neural nets (NN's) divide the patterns belonging to a training set into subsets called clusters by using specific similarity measures in such a way that patterns of the same clusters are very similar while patterns belonging to different clusters are very dissimilar. After the learning process each neuron represents a single cluster composed by all the patterns enclosed in the Voronoi region identified by the neuron itself. Several times the so obtained partition of the input space is more detailed than required because the net, to correctly learn, need more neurons than the class number. In this case we have some neurons representing the same class and some neurons positioned between the classes without representing any specific one.

These facts lead us to a next step, where, by using the *a priori* knowledge of an expert, we label each cluster with the name of a class in such a way that we can use the NN as a classifier. On the other way, if the net has as many outputs as the class number then it is very easy to label the output units, by using just an element for each class. Unfortunately, the neuron number for the classical one layer unsupervised NN's need to be much greater than the number of output classes to avoid a great error during the learning. In fact, during the learning, in many models, when a neuron wins the net adapts not only its weights but also the weights of its neighbors (soft-max adaptation) to avoid the fall in local minima of the error function. With a low number of neurons, when presenting in input a new pattern, we strongly modify also the weights related to other classes. This implies that classes with a greater number of input patterns have more than one neuron for representing them, while other classes have not even one neuron. In this case we say that the NN is under-dimensioned and it has an insufficient generalization capacity.

As a consequence, we need to use many neurons with respect to the class number and to have a complex labeling phase with several class representatives distributed over the output layer. To label every neuron, we need to use a trial and error technique, by giving in input to the net more patterns for each class and then by seeing the activated units. The main risk is to activate only a subset of the available neurons. It is easy to see that such a method is boring and not reliable and need many well classified patterns. In this case it is better to use supervised NN's which work better than unsupervised NN's. Aim of this paper is to solve in an automatic manner and without the help of an expert the labeling phase of any unsupervised NN, by using two different approaches: the former, called hybrid, is based on the application of a classical clustering algorithm to the neuron weights; the latter, called hierarchical neural net, is based on the multilayer organization of the net on ever smaller layers from the input to the output. In the next section we illustrate the neural models, i.e., unsupervised NN's, Hybrid models and hierarchical unsupervised NN's. In Section III we show the behavior of the nets on some data sets used as example.

Manuscript received April 23, 1998; revised October 16, 1998.

R. Tagliaferri is with DMI, Università di Salerno and INFM unità di Salerno, 84081 Baronissi, Salerno, Italy.
N. Capuano is with Facoltà di Scienze, Università di Salerno, 84081 Baronissi, Salerno, Italy.
G. Gargiulo is with IIASS "E. R. Caianiello," 84019 Vietri s/m, Salerno, Italy.

Publisher Item Identifier S 1045-9227(99)01159-5.

II. The Neural-Network Models

A. Unsupervised Neural Nets

Kohonen self organizing maps (SOM's) [4], [5] are composed by a neuron layer structured in a rectangular grid. When a pattern x is presented to the net each neuron i receives the components and computes the distance from its weight vector. The unit k which has the minimum distance from the input pattern will be the winner. The adaptation step consists in the modification of the weights of the neurons in the following way:

$$w_i^{(t+1)} = w_i^{(t)} + \varepsilon^{(t)} \cdot h_{\sigma(t)}(d(i,k)) \cdot (x - w_i^{(t)})$$

where $\varepsilon^{(t)}$ is a gain term $(0 \square \varepsilon^{(t)} \square 1)$ decreasing in time, $h_{\Sigma(t)}(x)$ is a unimodal function with variance $\Sigma^{(t)}$ decreasing with x and $d(i,k)$ is the distance in the grid between the i and the k neurons.

The neural-gas NN's have a learning algorithm [7] which works better than the preceding one: in fact, it is quicker and it reaches a lower average distortion value.[1] It uses a soft-max adaptation of the weights and it classifies the neurons in an ordered list $(i_1, i_2, \cdots, i_n)$ following their distance form the input pattern. The weight adaptation depends on the position *rank*(i) of the i neuron in the list in the following manner:

$$w_i^{(t+1)} = w_i^{(t)} + \varepsilon^{(t)} \cdot h_\sigma(\mathrm{rank}(i)) \cdot (x - w_i^{(t)}).$$

The algorithm applies the gradient descent technique to the error function

$$E_{ng} = \frac{1}{2C(\sigma)} \sum_{i=1}^{m} \int P(x) \cdot h_\sigma(\mathrm{rank}(x)) \cdot (x - w_i)^2 \, d^n x,$$

$$\text{with } C(\sigma) = \sum_{k=0}^{m-1} h_\sigma(k).$$

[1] Let $P(x)$ be the pattern probability distribution over the set $V \subseteq \Re^n$ and let $w_{i(x)}$ be the weight vector of the neuron which classifies the pattern x, therefore we define average distortion as

$$E = \int P(x)(x - w_{i(x)})^2 \, d^n x.$$

The NN is composed by a linear layer of neurons.

The growing cell structure (GCS) [2] is a NN which is able to change its structure depending on the data set. Aim of the net is to map the pattern space into a two-dimensional discrete structure A in such a way that similar patterns are represented by topological neighbor elements. The structure A is a two-dimensional (2-D) simplex where the vertices are the neurons and the edges attain the topological information. Every modification of the net always maintains the simplex properties. The learning algorithm starts with a simple three node simplex and tries to obtain an optimal network by a controlled growing process: for each x pattern of the training set the winner k and the neighbors weights are adapted as follows:

$$w_k = w_k + \varepsilon_b \cdot (x - w_k); \qquad w_i = w_i + \varepsilon_n \cdot (x - w_i)$$
$$\forall i \text{ connected to } k$$

where ε_b and ε_n are constants which determine the adaptation strength for the winner and for the neighbors, respectively. The insertion of a new node is made after a fixed number λ of adaptation steps. The new neuron is inserted between the unit which has won more times than the others and the farthest of its topological neighbors. The algorithm stops when the network reaches a predefined number of elements.

A simpler algorithm is the K-means clustering algorithm [9] in its on-line release which applies the gradient descent directly to the average distortion function above defined as follows:

$$w_i^{(t+1)} = w_i^{(t)} + \varepsilon^{(t)} \cdot (x - w_i^{(t)}).$$

The main limitation of this technique is that the error function presents many local minima which stops the learning before reaching the optimal configuration.

The last unsupervised learning algorithm is the maximum entropy [6] which applies the gradient descent with soft-max adaptation of the weights to the error function

$$E_{me} = -\frac{1}{\beta} \int P(x) \cdot \ln \sum_{j=1}^{m} e^{-\beta \cdot (x - w_j)^2} \, d^n x$$

and adaptation step

$$w_i^{(t+1)} = w_i^{(t)} + \varepsilon^{(t)} \cdot \frac{e^{-(\beta \cdot d_i)}}{\sum_{k=1}^{m} e^{-(\beta \cdot d_k)}} \cdot (x - w_i^{(t)})$$

where β is the inverse temperature and takes value increasing in time.

B. Hybrid Neural Nets

Hybrid NN's are composed by an unsupervised single layer NN and a clustering algorithm that uses the information derived by the NN learning algorithm. After the learning of the net, we must apply the clustering algorithm to have a neuron partition in subsets. Their number is equal to the number of the output classes. Furthermore, we want that neurons with similar weight vectors will be in the same class, while neurons with very distant weight vectors will be in different classes. The best strategy is clearly to apply the clustering algorithm directly to the weight vectors of the unsupervised NN after the learning.

A nonneural agglomeration clustering algorithm that divides the pattern set (in this case the weights of the neurons) $W = \{w_1, \cdots, w_m\}$ in l clusters $C_1, \cdots, C_l$ (with $l < m$) can be briefly summarized as follows.

1) We initially divide W in m clusters $C_1, \cdots, C_m$ such that $C_i = \{w_i\}$.
2) We compute the distance matrix D such that $D_{ij} = d(C_i, C_j)$.
3) We find the smallest element D_{ij} of the matrix D and we unify the clusters C_i and C_j in a new one $C_{ij} = C_i \cup C_j$.
4) If the number of clusters is greater than l then go to Step 2) else stop.

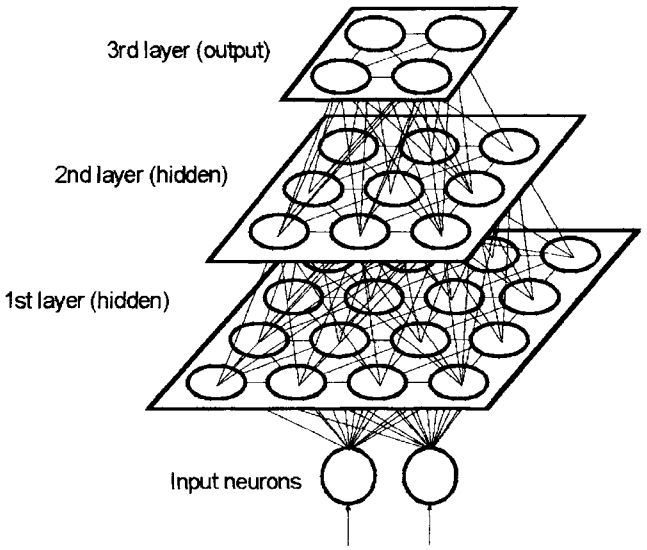

Fig. 1. Structure of a hierarchical neural network.

This is the shared basis of many algorithms appeared in literature [1]. The only difference is the distance function. For example, three different choices can be:

- $d(C_i, C_j) = \min_{w_{ik} \in C_i \text{ and } w_{jl} \in C_j} \|w_{ik} - w_{jl}\|$ (nearest neighbor algorithm);
- $d(C_i, C_j) = \|(1/|C_i|) \sum_{w_{ik} \in C_i} w_{ik} - (1/|C_j|) \sum_{w_{jl} \in C_j} w_{jl}\|$ (barycentre method);
- $d(C_i, C_j) = (1/mn) \sum_{1 \le k \le n, 1 \le l \le m} \|w_{ik} - w_{jl}\|$ (average between groups).

The output of the clustering algorithm will be a labeling of the patterns (in this case neurons) in l different classes.

C. Unsupervised Hierarchical Neural Nets

An alternative approach is to use a new unsupervised single layer NN instead of a clustering algorithm. In this way the second-layer NN learns from the weights of the first NN and clusters the neurons on the basis of a similarity measure or a distance. If we apply this process several times then we obtain the unsupervised hierarchical NN's. The number of neurons at each layer decreases form the first to the output layer, and, as a consequence, the net takes a pyramidal aspect as illustrated in Fig. 1.

The net takes as input a pattern x and then the first layer finds the winner neuron. The second layer takes the first layer winner weight vector as input and finds the second layer winner neuron and so on until the top layer. The activation value of the output layer neurons is one for the winner unit and zero for all the others. Briefly, the learning steps of a s layer hierarchical NN with training set X are the following.

- The first layer is trained on the patterns of X with one of the previous learning algorithms.
- The second layer is trained by using the same algorithm or one of the other previous ones on the elements of the set X_2 which is composed by the weight vectors of the first layer winner units.
- By using the same algorithm or one of the others, we iterate the process to the ith layer NN ($i > 2$) on the training set which is

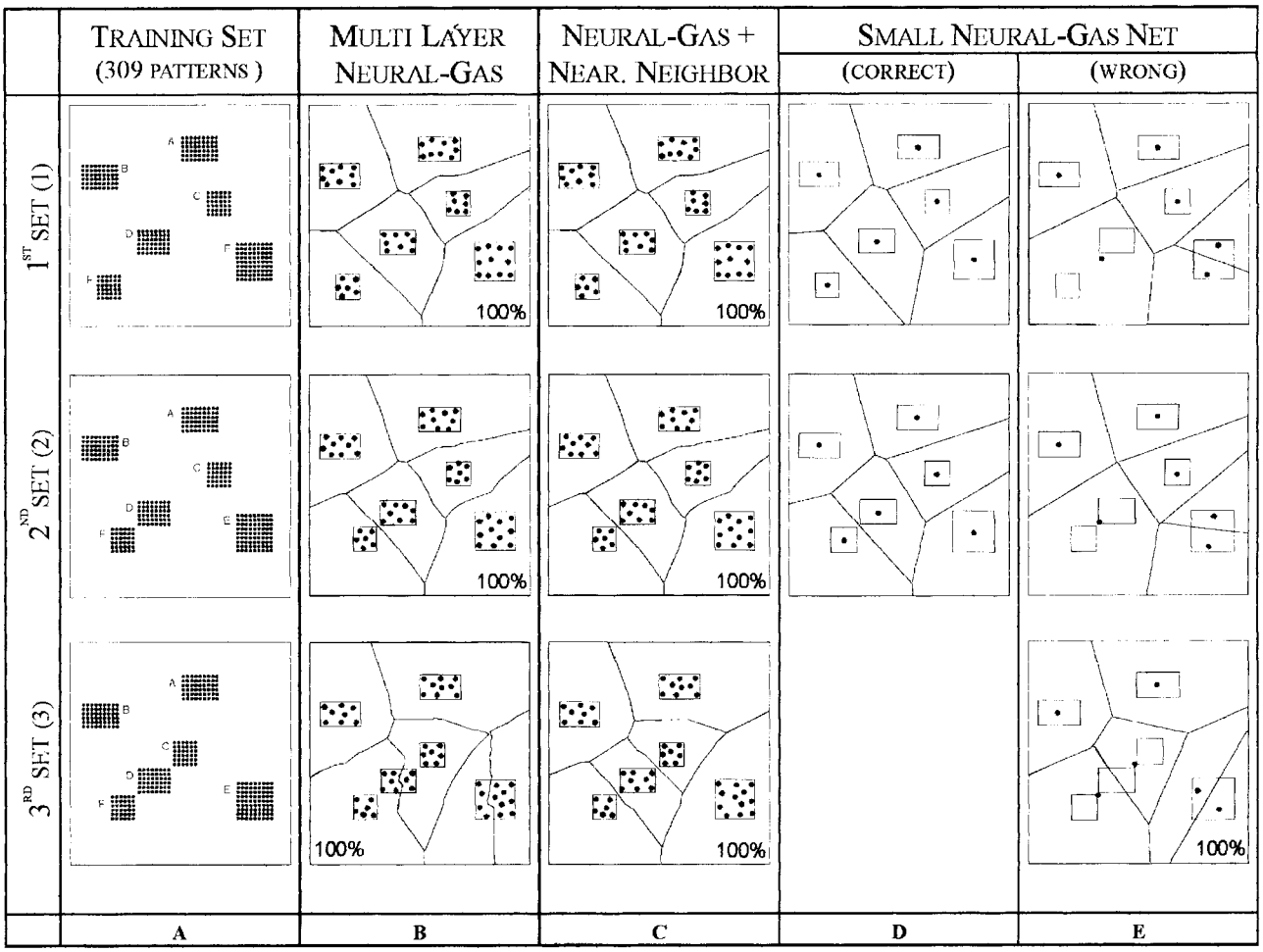

Fig. 2. Partitioning performance of the neural-gas algorithm.

composed by the weight vectors of the winner neurons of the $i-1$th layer when presenting X to the first layer NN, X_2 to the second layer and so on.

By varying the learning algorithms of the layers we obtain different NN's with different properties and abilities. For instance, by using only SOM's we have a *multilayer SOM* (ML-SOM) [3] where every layer is a 2-D grid. We can easily obtain *ML-Neural-Gas*, *ML-Maximum-Entropy* or *ML-K-means* organized on a hierarchy of linear layers. The *ML-GCS* has a more complex architecture and has at least three units for layer. We can think to have hierarchical NN's where different layers have different learning algorithms so that we can take advantage from the properties of each model (for example since we cannot have a ML-GCS with two output units, then we can use another NN in the output layer).

To solve our basic problem, we need to have a hierarchical NN with a number of output layer neurons equal to the number of the output classes. In this way the labeling becomes a simple problem without reducing the generalization capacity of the net. In fact, the first layer neurons are enough to correctly accomplish the distribution probability density of the input patterns. On the other hand, the number of neurons of a layer cannot rapidly decrease with respect to the number of units of the preceding layer, a hierarchical NN is slower than a single layer NN, and the computing time depends on the layer number. After the learning phase, it is simple to label in a unique way the input neurons depending on the corresponding output units. In this way we use single layer NN's in the computation on the test set.

III. Experimental Results

Following [7], we tested the models on training sets composed by 2-D patterns uniformly distributed over well-separated clusters. We generated three sets of 309 patterns divided into six clusters even nearer each other (see Fig. 2 column a).

We trained big one-layer NN's with 50 units and small NN's with six neurons on each training set by using the algorithms previously described in Section II.

Table I shows the learning parameters used by the nets (when the parameters are different for the big and small nets, the values related to the small ones are indicated in parenthesis). For the first two models we chose as adaptation function h_σ a gaussian function whose variance σ was exponentially decreased as shown in Table I. The GCS model starts with a trivial simplex with three neurons and then after 47 insertions (three in the case of small net) arriving to the final dimension of 50 units (three units for the small net). The λ parameter was set in such a way to obtain a number of adaptation steps very similar to those of the other nets.

First we focus on the neural-gas model (which reaches the best average distortion for the big net) and then we extend our attention to the other algorithms. Fig. 2 shows some partitioning examples of the feature space obtained with neural-gas nets. Column A illustrates

TABLE I
LEARNING PARAMETERS OF NEURAL NETWORKS (FIRST LAYER)

TRAINING ALG.	NODES	EPOCHS	GAIN TERM	PARAMETER
Kohonen	$7 \times 7(3 \times 2)$	50	$0.7 \rightarrow 0.05$	$\sigma = 5(3) \rightarrow 0.1$
Neural-Gas	50(6)	50	$0.7 \rightarrow 0.05$	$10(5) \rightarrow 0.1$
GCS	$3 \rightarrow 50(6)$	47(3)	$\epsilon_b = 0.2; \epsilon_n = 0.05$	$\lambda = 329(5150)$
K-means	50(6)	50	$0.7 \rightarrow 0.05$	
Max Entropy	50(6)	50	$0.7 \rightarrow 0.05$	$\beta = 5(10) \rightarrow 100$

TABLE II
PERFORMANCE OF NEURAL NETWORKS ON SYNTHETIC DATA

TRAINING ALGORITHM	AVE. DISTORTION		CORRECTNESS (PERCENT)								
	Big net	Small net	ML-big net			Big net + Near. neigh.			Small net		
			set 1	set 2	set 3	set 1	set 2	set 3	set 1	set 2	set 3
Kohonen	0.031	0.104	100	100	0	100	100	100	100	93	24
Neural-Gas	0.029	0.105	100	100	0	100	100	100	85	46	5
GCS	0.036	0.128	100	100	0	100	100	100	12	0	0
K-means	0.060	0.211	100	100	0	100	100	100	55	30	7
Max Entropy	0.072	0.127	100	100	0	100	100	100	13	0	0

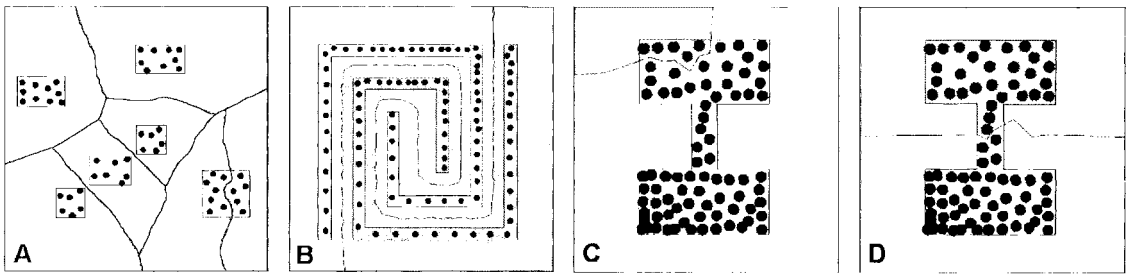

Fig. 3. Other neural-gas clustering examples.

distributions on the feature space of the three training sets (each set has six clusters with 54(A), 54(B), 36(C), 48(D), 81(E) e 36(F) patterns). Column B e C show the 50 weights position of the big neural-gas (the circles) in the feature space after the training phase on the three example sets (in this image, clusters are showed as boxes). To obtain an input space partition into six regions (one for each cluster) we must label the 50 neurons through one of the previously described algorithms. Column B shows the partitions obtained (in the three cases) by using a three-layer NN with the second and third Neural-Gas layers of 20 and six units, respectively (the barycentre method reach the same performance too). Column C instead shows the partitions obtained applying the nearest neighbor clustering algorithm to the trained net. Columns D and E illustrate correct and wrong partitions obtained by the small neural-gas net on the same data (for this net labeling is trivial because the output number is equal to the class number).

For what concern the first training set (easy case), all the labeling algorithms applied to the big neural-gas give correct partitions. The net with six neurons obtains a sufficient representation only in the 85% of the cases. In the remaining 15% we have the wrong representation of Fig. 2E1 where the biggest cluster absorbed two neurons (since there are few neurons, then the learning is unstable and the results strongly depend on the random weight initialization). By using the second training set (difficult case), the failures of the six units net increase (wrong classification of Fig. 2E2 in the 54% of the cases and correct classification of Fig. 2D2 in the 46%) while our neural nets (hierarchical and hybrids) continue to correctly work in the 100% of the experiments [Fig. 2(B) and 2(C)]. For what concern the third training set (very difficult case), both the six-unit NN's [Fig. 2(E)] and the hierarchical NN's [Fig. 2(B)] find a wrong representation. The big Neural-Gas + the nearest neighbor obtains the right representation shown in Fig. 2C3.

The NN's correctness percents are illustrated in Table II. It is worth of noting that the nearest neighbor applied to big nets always obtain the 100% of successful. The hierarchical NN's work well only on the first two training sets while on the third training set all of them obtain a failure. It is interesting to note that, if we add a single output neuron to the hierarchical nets (so dividing the feature space in seven clusters) then we obtain the quasicorrect partition showed in Fig. 3(a) (neural-gas case) where a small human intervention can lead to the correct one (it is sufficient to merge the two clusters that represent class E).

A second kind of experiments were accomplished by using the IRIS data set [8] to analyze the ability of the hierarchical and hybrid

TABLE III
PERFORMANCE OF NEURAL NETWORKS ON IRIS DATA SET

TRAINING ALGORITHM	CORRECTNESS (PERCENT) PER CLASS								
	ML-big net $10(3 \times 4) - 3$ nodes			Big net $10(3 \times 4)$ + nearest neighb.			Small net 3 nodes		
	Cl. 1	Cl. 2	Cl. 3	Cl. 1	Cl. 2	Cl. 3	Cl. 1	Cl. 2	Cl. 3
Kohonen	100	100	72	100	95	70	100	96	72
Neural-Gas	100	100	72	100	52	98	100	96	72
GCS	100	54	98	100	100	72	28	54	48
K-means	100	0	100	100	0	100	100	0	100
Max Entropy	100	98	90	100	96	72	100	96	72

NN's in handling overlapped clusters. IRIS data set is composed by three classes of 50 elements each; class 1 is linearly separable from the others, while classes 2 and 3 are strongly overlapped. The parameters of the models are the same as those of Table I except for the net sizes that are $10 - 3$ for hierarchical NN's and $3 \times 4 - 3$ for MLSOM, 10 for hybrid NN's and 3 (the class number) for small NN's. Experimental results are shown in Table III where we indicate the correctness percent of NN's to represent each class of the data set (for example, if we take ML-GCS, we see the third row and the first three columns: the first column indicates that all the elements of the first class are in the first cluster, the second column means that 27 elements of the second class are in the second cluster while the third column shows that 49 patterns of the third class are in the third cluster).

As a consequence of the results shown in Table III, we can elicit that even with overlapped clusters, hierarchical NN's work better than the corresponding small nets and hybrid nets, but the GCS. More specifically, ML-Maximum Entropy has only six patterns not correctly clustered, ML-SOM and ML-Neural Gas are a bit worse with 14 patterns (all belonging to class 3) with the same performance of GCS + nearest neighbor. Hybrid NN's does not work so well as in the first experiment because although they are able to find nonellipsoidal clusters [see Fig. 3(b)], they do not work in a good way when dealing with well defined clusters but not well separated ones (overlapped clusters). In fact, in this case they create chains that lead to misclassification errors in the class separation. Effects of chaining are shown in Fig. 3(c) and (d) where we try to separate two clusters joined by a little "ribbon." In this case Neural-Gas + nearest neighbor fails while ML-Neural-Gas assures a good partition [Fig. 3(d)].

Preliminary results on astronomical real data in the context of star/galaxy separation have shown that hierarchical NN's can be successfully applied to image segmentation of stellar fields [10] in the detection and extraction of celestial objects from a noisy background (a highly overlapped clustering problem). Our approach (PCA + Hierarchical NN's + a simple deblending algorithm) has improved the detection performance from 74% (widely used nonneural method) to 94% (GCS + Neural Gas + Neural Gas, 50-20-6) of correctness percent. We needed six output classes for having a good result but it is much simpler both in terms of examples and of class separation than using a single-layer NN with 50 neurons. A bit worse results were obtained by ML-Neural Gas and by ML-SOM, while ML-Maximum Entropy was too much sensitive to noise; the best result reached by hybrid and nonhierarchical NN's is only 65% of correctness percent.

IV. CONCLUDING REMARKS

In this paper we have shown some experimental considerations on the automatic labeling by means of single layer, hierarchical and hybrid NN's. To obtain an automatic labeling, hierarchical NN's are to be preferred to single layer NN's. The hybrid barycentre method is equivalent to hierarchical NN's, while the hybrid nearest neighbor is somewhat different. This last method is the best technique for nonoverlapping clusters but, as shown above, it has some problems handling overlapping regions.

REFERENCES

[1] B. Everitt, *Cluster Analysis*, Social Science Research Council. London, U.K.: Heinemann, 1977.

[2] B. Fritzke, "Growing cell structures—A self-organizing network for unsupervised and supervised learning," in *Neural Networks*, vol. 7, no. 9, pp. 1441–1460, 1994.

[3] J. Koh, M. Suk, and S. Bhandarkar, "A multilayer self-organizing feature map for range image segmentation," in *Neural Networks*, vol. 8, no. 1, pp. 67–86, 1995.

[4] T. Kohonen, "Self-organized formation of topologically correct feature maps," in *Biol. Cybern.*, vol. 43, pp. 59–69, 1982.

[5] ——, *Self-Organization and Associative Memory*, 2nd ed. Berlin, Germany: Springer-Verlag, 1988.

[6] S. Lloyd, "Least squares quantization in PCM," in *IEEE Trans. Inform. Theory*, vol. IT-28, p. 2, 1982.

[7] T. Martinetz, S. Berkovich, and K. Schulten, "Neural-gas network for vector quantization and its application to time-series prediction," in *IEEE Trans. Neural Networks*, vol. 4, pp. 558–568, 1993.

[8] C. J. Mertz and P. M. Murphy, *UCI Repository of Machine Learning Databases*, http://www.ics.uci.edu/pub/machine-learning-data-bases.

[9] K. Rose, F. Gurewitz, and G. Fox, "Statistical mechanics and phase transition in clustering," in *Phys. Rev. Lett.*, vol. 65, no. 8, pp. 945–948, 1990.

[10] R. Tagliaferri, G. Longo, S. Andreon, S. Zaggia, N. Capuano, and G. Gargiulo, "Astronomical object recognition by means of neural networks," M. Marinaro and R. Tagliaferri, Eds., in *Proc. 10th Italian Wkshp. Neural Nets WIRN Vietri 98.* London: Springer-Verlag, pp. 169–178.