Application of Neural Networks in Cluster Analysis

Mu-Chun Su*, Nicholas DeClaris+, and Ta-Kang Liu*
***Department of Electrical Engineering, Tamkang University, Taiwan, R.O.C.**
+School of Medicine and College of Engineering, University of Maryland in Baltimore and College Park, U.S.A.
E-mail: muchun@ee.tku.edu.tw

ABSTRACT

How to efficiently specify the "correct" number of clusters from a given multidimensional data set is one of the most fundamental and unsolved problems in cluster analysis. In this paper, we propose a method for automatically discovering the number of clusters and estimating the locations of the centroids of the resulting clusters. This method is based on the interpretation of a self-organizing feature map (SOFM) formed by the given data set. The other difficult problem in cluster analysis is how to choose an appropriate metric for measuring the similarity between a pattern and a cluster centroid. The performance of clustering algorithms greatly depends on the chosen measure of similarity. Clustering algorithms utilizing the Euclidean metric view patterns as a collection of hyperspherical-shaped swarms. Actually, genetic structures of real data sets often exhibit hyperellipsoidal-shaped clusters. In the second part of this paper we present a method of training a single-layer neural network composed of quadratic neurons to cluster data into hyperellipsoidal- and/or hyperspherical-shaped swarms. Two data sets are utilized to illustrate the proposed methods.

1. INTRODUCTION

Cluster analysis is one of the basic tools for exploring the underlying structure of a given data set and is being applied in a wide variety of engineering and scientific disciplines such as medicine, psychology, biology, society, pattern recognition, and image processing. The primary objective of cluster analysis is to partition a given data set of multidimensional vectors (patterns) into so-called homogeneous clusters such that patterns within a cluster are more similar to each other than patterns belonging to different clusters. Cluster seeking is very experiment-oriented in the sense that cluster algorithms that can deal with all situations are not yet available. Extensive and good overview of clustering algorithms can be found in [1]-[3]. Perhaps the best known and most widely used member of the family is the K-means algorithm or the Isodata algorithm [4]. Lately neural networks, for example, competitive-learning networks [5], self-organizing feature maps [6], and adaptive resonance theory (ART) networks [7], [8] also often have been used to cluster data.

The performance of most clustering algorithms is greatly influenced by the number of clusters which can not always be defined *a priori*, the choice of initial cluster centroids, and the geometrical properties (e.g. shapes and distributions) of the data. Generally, there are two approaches to specifying the number of clusters. The first approach involves increasing the number of clusters, computing some certain performance measures in each run, until partition into optimal number of clusters is obtained [9]-[10]. This approach requires extensive computation. The second approach focuses on finding a good projection algorithm which maps a set of multi-dimensional pattern onto a two-dimensional space so as to allow one to cluster data directly by eye [11]-[14]. The price paid for the possibility of visual examination of clusters is that we can not automate the specification of clusters.

While it is easy to consider the idea of a data cluster on a rather informal basis, it is very difficult to give a formal and universal definition of a cluster. In order to mathematically identify clusters in data, it is usually necessary to first define a measure of similarity or proximity which will establish a rule for assigning patterns to the domain of a particular cluster centroid. As it is to be expected, the measure of similarity is problem dependent. The most popular similarity measure is the Euclidean distance. The smaller the distance, the greater the similarity. By using Euclidean distance as a measure of similarity, hyperspherical-shaped clusters of equal size are usually detected. This measure is useless or even undesirable when clusters tend to develop along principal axes. Actually, genetic structures of real data sets often exhibit hyperellipsoidal-shaped clusters. To take care of hyperellipsoidal-shaped clusters, the Mahalanobis distance from $\underline{x}$ to $\underline{m}$, $D(\underline{x},\underline{m}) = (\underline{x}-\underline{m})^T \Sigma^{-1} (\underline{x}-\underline{m})$, is one of the popular choices. The matrix Σ is the covariance matrix of a pattern population, $\underline{m}$ is the mean vector, and $\underline{x}$ represents a pattern. One of the major difficulties associated with using the Mahalanobis distance as a similarity measure is that we have to recompute the inverse of the sample covariance matrix every time a pattern changes its cluster domain, which is computational expensive. A self-organizing network for hyperellipsoidal clustering (HEC) using the regularized

0-7803-4053-1/97/$10.00 © 1997 IEEE

Mahalanobis distance was proposed to reduce the computational requirements [15]. The HCE network consists of two layers. The first layer employs a number of principal component analysis (PCA) subnetworks which are used to estimate the hyperellipsoidal shapes of currently formed clusters. The second layer then performs a competitive learning using the cluster shape information provided by the first layer. The approach does not need to compute the inverse of the covariance matrix, however, the PCA subnetworks may take time to converge if the learning parameters are not properly chosen.

In this paper, we propose to use two classes of neural networks to cluster multidimensional data. First, we use the data set to form a self-organizing feature map (SOFM). Although the SOFM is originally intended to visualize metric-topological relationships of input patterns, however, it should be emphasized that the interpretation of an obtained map is not as straightforward as it appears to be. In order to extract clustering information from a trained SOFM, we first transform the accumulated responses of the array neurons into a 2-D digital image where brightness of each pixel is proportional to the accumulated response of the respective neuron and then use a 3×3 mask to search the peaks of the image. Each peak corresponds to a pixel of which gray level is brighter then its 8-neighbors. The number of peaks of the image estimates the number of clusters of the given data set. Besides, the locations of the peaks also provide us with the estimates of the locations of cluster centroids. After we have estimated the clustering information from the SOFM, we train a single-layer neural network composed of quadratic neurons to cluster data. The number of the quadratic neurons is determined according to the number of the peaks of the image. The values of the synaptic weights are initialized by the coordinates of the locations of the peaks. The inherent characteristics of quadratic neurons were reported in [16]-[18]. The neural network is trained in an unsupervised manner to cluster the data into hyperspherical-shaped or hyperellipsoidal-shaped swarms according to the underlying structure of the data.

The remaining of the paper is organized as follows. Section 2 briefly introduces the characteristics of SOFMs and the proposed technique to extract clustering information from a trained SOFM. We discuss the characteristics of quadratic neurons and the proposed unsupervised training algorithm for the neural networks consisted of quadratic neurons in Section 3. Simulation results of two data sets are provided in Section 4. Finally, a few concluding remarks are given to conclude this paper.

2. INTERPRETATION OF SELF-ORGANIZING FEATURE MAPS

The principal goal of self-organizing feature maps is to transform patterns of arbitrary dimensionality into the responses of one- or two-dimensional arrays of neurons, and to perform this transform adaptively in a topological ordered fashion. The transformation makes topological neighborhood relationship geometrically explicit in low-dimensional feature maps. The essential constituents of SOFMs are as follows [6] :

- an array of neurons that compute simple output functions of incoming inputs of arbitrary dimensionality,
- a mechanism for selecting the neuron with the largest output, and
- an adaptive mechanism that updates the weights of the selected neuron and its neighbors.

One typical network structure is shown in Fig. 1. Note that each component of the input pattern $\underline{x} = (x_1, \cdots, x_n)^T$ is simultaneously connected to each of an N×N array of neurons. The output of the jth neuron is defined as

$$Out_j^{(t+1)}(\underline{x}(t+1)) = f\left(\sum_{i=1}^{n} w_{ji} x_i(t+1) + \sum_{l \in L_j} w_{jl} Out_l^{(t)}(\underline{x}(t))\right) \quad (1)$$

where w_{ji} denotes the connection weight from the ith input component to the jth neuron, w_{jl} denotes the lateral feedback weight from neuron l to neuron j, L_j denotes the subset that contains the neurons having lateral feedback weights connected to neuron j, f is a suitable activation function, t denotes a discrete time index, and $\underline{x}(t) = [x_1(t), \cdots, x_n(t)]^T$ represents the tth input pattern. The training algorithm for forming a SOFM is summarized as follows :

Step 1: Initialization: Choose random values for the initial weights $\underline{w}_j(0)$.

Step 2: Winner Finding: Find the winning neuron j^* at time t, using the minimum-distance Euclidean criterion:

$$j^* = \arg\min_j \|\underline{x} - \underline{w}_j\|, \qquad j = 1, \cdots, N^2 \quad (2)$$

where $\|\cdot\|$ indicates the Euclidean norm.

Step 3: Updating: Adjust the weights of the winner and its neighbors, using the following rule:

$$\underline{w}_j(t+1) = \begin{cases} \underline{w}_j(t) + \eta(t)(\underline{x}(t) - \underline{w}_j(t)) & \text{if } j \in N_{j^*}(t) \\ \underline{w}_j(t) & o.w. \end{cases} \quad (3)$$

where $\eta(t)$ is a positive constant and $N_{j^*}(t)$ is the topological neighborhood set of the winner neuron j^* at time t. Practically, we usually start with a wide range for $N_{j^*}(t)$ and a large $\eta(t)$ and then reduce both the range of $N_{j^*}(t)$ and the value of $\eta(t)$ gradually as learning proceeds.

The global topological ordering of the weight vector $\underline{w}_j(t)$ take place during the initial phase of the learning process. The remaining iterations are needed

principally for the fine tuning of the feature map. It should be emphasized that the success of the map formation is critically dependent on how the main parameters (i.e. $\eta(t)$ and $N_{j^*}(t)$) are selected, initial values of the weight vectors, and the number of iterations. Although the SOFM are originally intended to visualize metric-topological relationships of input patterns, the interpretation of a SOFM is not as straightforward as it appears to be. A trained SOFM has to be calibrated by supervised labeling of array neurons in response to a specific known vector from the training set. Such labeling is usually achieved by the so-called "voting method" (i.e. a neuron is labeled class k if it responds to input patterns belonging to class k as a majority within the whole data set). After this, we are able to analyze the meaning of the labeled map. As we can see, if no category information is available, the inspection of the map does not reveal any information about the clustering characteristics of the data set. In fact, the absence of category labels distinguishes cluster analysis from pattern recognition (and discriminate analysis), therefore, an unlabeled SOFM is not of much help in cluster analysis.

Based on above discussions, we propose a method to interpret an unlabeled SOFM so as to provide us with information of both the number of clusters and the locations of the cluster centroids. The method is given as follows:

1. **Map forming.** The whole training data set is used to form a SOFM.
2. **Response accumulation.** The responses of each neuron are accumulated according to the following equations:

$$h_j = \sum_{t=1}^{M} Out_j(\underline{x}_t) \qquad j = 1,2,\cdots,N^2 \tag{4}$$

and

$$Out_j(\underline{x}_t) = e^{-\left\|\underline{x}_t - \underline{w}_j\right\|^2} \qquad j = 1,2,\cdots,N^2 \tag{5}$$

where M denotes the number of training patterns.

3. **Peak searching.** We can view the accumulated responses of the neurons as an N×N digital image. The digital image is a matrix whose row and column indices identify a neuron in the array and the corresponding matrix element value identifying the gray level at that point is proportional to the value of the accumulated response of the corresponding neuron. The image provides a global structure of the given data set. Pixels with relatively bright gray levels are the potential centroids of clusters. A 3×3 mask shown in Fig. 2 can be used to automate the specification of such pixels. We scan the image pixel by pixel, from top to bottom and from left to right. Let p_0 denote the pixel at any step in the scanning process and let $p_1,\ldots,p_8$ be the 8-neighbors of p_0. A peak exists at the pixel p_0 if the following condition is satisfied:

$$h_{p_0} > h_{p_i} \qquad i = 1,2,\cdots,8 \tag{6}$$

3. NEURAL NETWORKS WITH QUADRATIC NEURONS

In many practical situations data tend to cluster. A reasonable approach to cluster data would be in the form of a hyperspherical or hyperellipsoidal swarm of patterns. This assumption brought about the class of neural networks using quadratic junctions reported earlier. The inherent architecture of the networks was discussed in [16], [17]. In addition, it was shown that any Gaussian classifier can be mapped into a neuron using quadratic junctions [18]. Here a variety of the quadratic-type junctions we consider in this paper is described by the following equations :

$$y_{jk}(\underline{x}) = \sum_{i=1}^{n} w_{jki} x_i \tag{7}$$

$$net_j(\underline{x}) = \sum_{k=1}^{n} (y_{jk}(\underline{x}) - b_{jk})^2 \tag{8}$$

and

$$Out_j(\underline{x}) = e^{-s_j^2 net_j(\underline{x})} \tag{9}$$

where b_{jk}, s_j, and w_{jki} are adjustable weights, $\underline{x} = (x_1,\cdots,x_n)^T$ is an input pattern, $\underline{y}_j = (y_{j1},\ldots,y_{jn})^T$ is the transformed version of the input pattern $\underline{x}$, and $Out_j(\underline{x})$ is the output function of neuron j.

This type of quadratic junctions is capable of achieving hyperellipsoidal discriminants that can be varied in sizes and in locations. The reasons are as follows. First, the rotation and scaling transformation are easily accomplished to transform the X-space into the Y-space via Eq.(7). Then a hypersphere is defined in the transformed space (i.e. Y-space) by Eq.(8). Combining these two equations, we are able to define a hyperellipsoid whose principal axes may be oblique to the coordinate axes.

From our previous works [16], [17], we have developed a supervised training algorithm for training the networks. Here we present an unsupervised training algorithm for the networks. Instead of sequentially computing principal components to define a hyperellipsoid, we propose to adjust the weights of the winning quadratic neuron in the direction of gradient of the output function so as to increase the chances of winning by the jth neuron for the repetition of the same input pattern. That is, the weight adaptation rule tends to enhance the same responses to a sufficiently similar subsequent input. The weights are updated using the following rules:

$$b_{jk}(t+1) = b_{jk}(t) + \eta_b \frac{\partial Out_j(\underline{x})}{\partial b_{jk}(t)} \tag{10}$$

$$= b_{jk}(t) + \eta_b (2 s_j^2 Out_j(\underline{x})(y_{jk}(\underline{x}) - b_{jk}(t))),$$

$$s_j(t+1) = s_j(t) + \eta_s \frac{\partial Out_j(\underline{x})}{\partial s_j(t)} \tag{11}$$

$$= s_j(t) + \eta_s(-2s_j Out_j(\underline{x}) net_j(\underline{x})),$$

and

$$w_{jki}(t+1) = w_{jki}(t) + \eta_w \frac{\partial Out_j(\underline{x})}{\partial w_{jki}(t)} \tag{12}$$

$$= w_{jki}(t) + \eta_w(-2s_j^2 Out_j(\underline{x})(y_{jk}(\underline{x}) - b_{jk}(t))x_i)$$

where η_b, η_s, and η_w are positive learning rates. The whole competition learning algorithm for a single-layer neural network composed of quadratic neurons can be stated as follows:

1. **Initialization:** Set $\underline{b}_j(0)$ to be the coordinates of the estimated locations of cluster centroids, set $w_{jki}(0) = 1$ if $k = i$, and zero otherwise, and set $s_j = 1$, for j=1, 2,···, C, where C represents the estimated number of clusters. That is, we let the C clusters start from being C hyperspheres.
2. **Activation:** At time t, activate the C quadratic neurons by applying input pattern $\underline{x}(t)$.
3. **Winner Finding:** Find the winning neuron j^* at time t, using the maximum-value criterion:
   $j^* = \arg\max_j Out_j(\underline{x}), \quad j = 1,\ldots,C.$
4. **Updating:** Adjust the synaptic weight vector of the winning neuron, using Eqs. (10)-(12).
5. **Continuation:** Continue with Step 2 until no noticeable changes in the weight vectors are observed or the same result (*i.e.* the cluster membership of input patterns) as in the previous iteration is obtained.

4. SIMULATION RESULTS

Two data sets are used to illustrate the effectiveness of the proposed method. For the artificial data set the comparison is performed between a single-layer neural network composed of quadratic neurons and the K-means algorithm with Euclidean distance. The performance was also demonstrated on the well-known Iris data. For the Iris data set, the reported results of [15] and [19] are also used for the comparison purpose in the simulations. Note that since the behaviors of the K-means algorithm and the networks with quadratic junctions are influenced by the choice of initial cluster centroids or initial weights, the best results in ten trials are reported in this paper. For each data set, a 5×5 network was trained to form a SOFM.

Example 1: Artificial data set

In order to test the performance of the proposed method, we generated a mixture of spherical and ellipsoidal clusters which are shown in Fig. 3. As can be seen, there is no clear border between clusters. Fig. 4 and Fig. 5. show the trained SOFM and the image of the accumulated responses, respectively. For a more distinct illustration, we flip the gray levels such that darker regions represent peaks instead of valleys. By using the 3×3 mask shown in Fig. 2. to scan the image, we found there are three peaks located on the coordinates (2,4), (3,1), and (4,4), respectively. Therefore, there exist three clusters in the data set. We then use these information to initialize a single-layer neural networks with 3 quadratic neurons. After training, the resulting clusters are shown in Fig. 6. For the comparison purpose, we use the K-means algorithm associated with the Euclidean metric to cluster the same data set. The resulting three clusters are shown in Fig. 7. This figure tells us there are several patterns which are not appropriately clustered.

Example 2: Iris data set

The Iris data set has three subsets (i.e. Iris Setosa, Iris Versicolor, and Iris Virginica), two of which are overlapping. The Iris data are in a four-dimensional space and there are total 150 patterns in the set. Fig. 8. shows the image of the accumulated responses. For a more distinct illustration, we flip the gray levels such that darker regions represent peaks instead of valleys. The image tells us there are three peaks located on the coordinates (1,3), (3,1), and (4,4), respectively. We then used 3 quadratic neurons to cluster the Iris data set. The learning rates were set to be 0.01, 0.001, and 0.04 for η_b, η_s, and η_w, respectivly. The training procedure terminated at the 60th iterations. The performance results are tabulated in Table I. Note that the K-means algorithm with Euclidean distance misclassified 16 patterns, the HEC network misclassified 5 patterns, the fuzzy clustering algorithm [19] misclassified 4 patterns, and the quadratic-junction network misclassified 4 patterns.

5. CONCLUSIONS

In this paper, we first propose a method of estimating both the number of clusters and the locations of the cluster centroids. The method is based on interpreting an unlabeled SOFM. We then use these estimates to initialize a single-layer neural network with quadratic neurons.The network is updated in an unsupervised manner to cluster data into hyperellipsoidal-shaped or hyperspherical-shaped clusters according to the underlying structure of the data set. Simulation results are very encouraging.

REFERENCES

[1] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentic Hall, New Jersey 1988.

[2] R. O. Duda and P. E. , *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.

[3] J. Bezdek, *Pattern Recognition with Fuzzy Objective*

Function Algorithms, New York: Plenum, 1981.

[4] G. H. Ball and D. I. Hall, "Some fundamental concepts and synthesis procedures for pattern recognition preprocessors," *in Proc. of Int. Conf. Microwaves, Circuit Theory, and Information Theory, Tokyo, Japan*, pp. 281-297, Sep, 1964.

[5] T. Kohonen, "The 'Neural' Phonetic Typewritter," *IEEE Computer*, vol. 27, no. 3, pp. 11-12, 1988.

[6] T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed. New York, Berlin: Springer-Verlag, 1989.

[7] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Proc*, vol. 37, pp. 54-115, 1987.

[8] G. A. Carpenter and S. Grossberg, "ART2: self-organization of stable category recognition codes for analog input patterns," *Appl. Optics*, vol. 26, no. 23, pp. 4919-4930, Dec. 1987.

[9] J.-C. Lin, "Multi-class clustering by analytical two-class formulas," *Pattern Recognition and Artifical Intelligence*, vol. 10, no.4, pp. 307-323, 1996.

[10] R. P. Li and M. Mukaidono, "A new approach to rule learning based on fusion of fuzzy logic and neural networks," *IEICE Trans. Inf. & Syst.*, vol. E78-D, No. 11, November, 1995.

[11] W. C. Chang, "On using principal components before separating a mixture of two multivariate normal distribution," *Applied Statistics*, vol. 32, pp. 267-275, 1983.

[12] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Trans. on Computers*, vol 18, pp. 401-409, 1969.

[13] C. E. Pykett, "Improving the efficiency of Sammon's nonlinear mapping by using clustering archetypes," *Electronics Letters* 14, 799-800, 1978.

[14] R. C. T. Lee, J. R. Slagle, and H. Blum, "A triangulation method for the sequential mapping of points from N-space to two-space," *IEEE Trans. on Computers*, vol. 26, pp. 288-292, 1977.

[15] J. Mao and A. K. Jain, "A self-organizing network for hyperellipsoidal clustering (HCE)," *IEEE Trans. on Neural Networks*, vol. 7, no. 1, pp. 16-29, Jan. 1996.

[16] M. C. Su, *Network Training Using Quadratic Neural-Type Junctions*, Master's thesis, Univ. of Maryland, College, Park, Aug. 1990.

[17] N. DeClaris and M. C. Su, "A novel class of neural networks with quadratic junctions," *IEEE Int. Conf. on Systems, Man, and Cybernetics*, 1991, pp. 1557-1562 (Received the IEEE 1992 Franklin V. Taylor Award).

[18] N. DeClaris and M. C. Su, "Introduction to the theory and application of neural networks with quadratic junctions," *IEEE Int. Conf. on Systems, Man, ,and Cybernetics*, pp. 1320-1325, 1992.

[19] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Trans. Pattern Anal. Machine Intell*, vol. 11, no. 7, pp. 773-781, July. 1989.

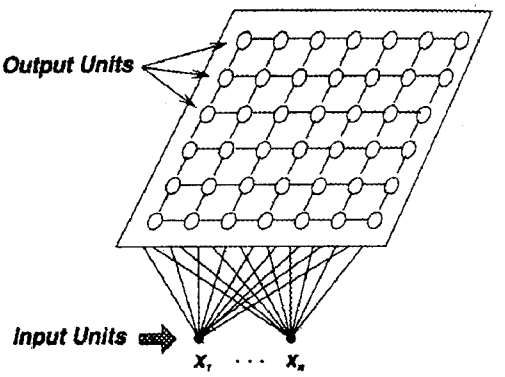

Fig. 1. One typical seif-organizing feature map network structure.

p_4	p_3	p_2
p_5	p_0	p_1
p_6	p_7	p_8

Fig. 2. A 3×3 mask for searching the peaks of an image.

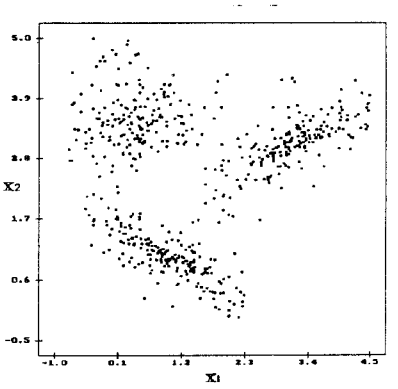

Fig. 3. The artificial data set: 579 data points.

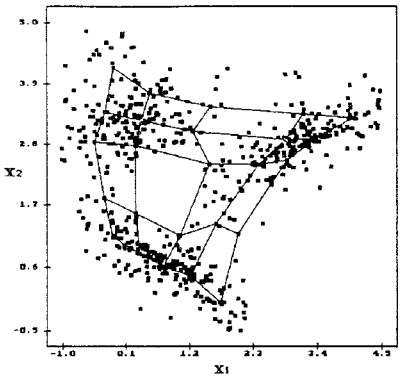

Fig. 4. The trained SOFM for the artificial data set.

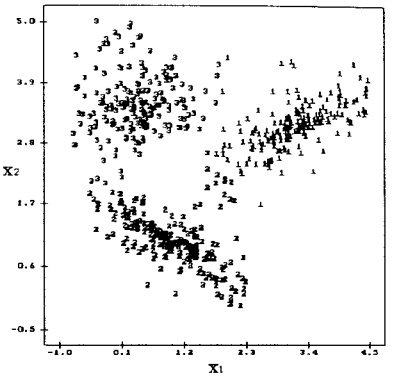

Fig. 7. Partition using the K-means algorithm with Euclidean distance.

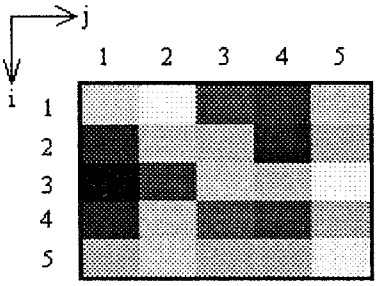

Fig. 5. The image of the accumulated responses for the artificial data set.

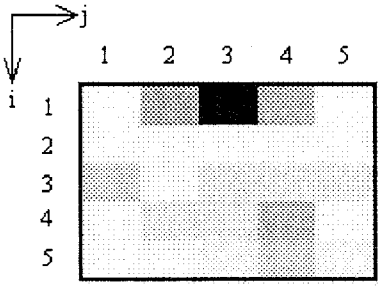

Fig.8. The image of the accumulated responses of Iris data set.

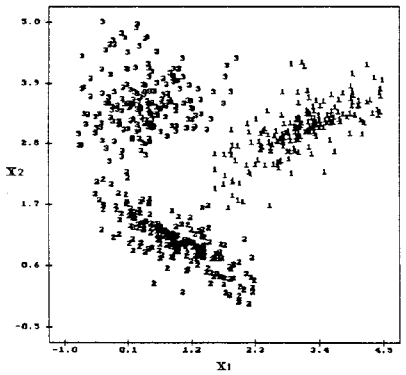

Fig. 6. Partitions using the network composed of three quadratic neurons.

Table I. The performance of clustering algorithms for the Irisdata

method	K-means Euclidean	HEC	fuzzy clustering	proposed network
misclassified patterns	16/150	5/150	4/150	4/150