A Self-Organizing Network for HyperEllipsoidal Clustering (HEC) *

Jianchang Mao and Anil K. Jain
Department of Computer Science
Michigan State University
East Lansing, MI 48824, USA

Abstract

We propose a self-organizing network (HEC) for HyperEllipsoidal Clustering. The HEC network performs a partitional clustering using the regularized Mahalanobis distance. This regularized Mahalanobis distance measure is proposed to deal with the problems in estimating the Mahalanobis distance when the number of patterns in a cluster is less than (ill-posed problem) or not considerablely larger than (poorly-posed problem) the dimensionality of the feature space in clustering multidimensional data. This regularized distance also achieves a tradeoff between hyperspherical and hyperellipsoidal cluster shapes so as to prevent the HEC network from producing unusually large or unusually small clusters. The significance level of the Kolmogrov-Smirnov test on the distribution of the Mahalanobis distances of patterns in a cluster to the cluster center under the multivariate Gaussian assumption is used as a measure of cluster compactness. The HEC network has been tested on a number of artificial data sets and real data sets. Experiments show that the HEC network gives better clustering results compared to the well-known K-means algorithm (and other partitional clustering algorithms and competitive learning networks) with the Euclidean distance metric. Our results on real data sets also indicate that hyperellipsoidal shaped clusters are often encountered in practice.

1 Introduction

A data clustering problem can be formulated as follows. Given a set of n patterns in a d-dimensional space, $\{\mathbf{x}_i \mid \mathbf{x}_i \in \mathcal{R}^d,\ i = 1, 2, \cdots, n\}$, the objective of a partitional clustering algorithm is to determine an assignment (or a partition) $\{M_{ij} | M_{ij} \in \{0, 1\}; i = 1, 2, \cdots, n;\ j = 1, 2, \cdots, K\}$, such that some cost function is minimized, where $M_{ij} = 1$ if pattern $\mathbf{x}_i$ is assigned to the j^{th} cluster, and $M_{ij} = 0$ otherwise; K is the number of clusters which is usually specified by the user. For the unique assignment (non-overlapping partition), $\sum_{j=1}^{K} M_{ij} = 1,\ i = 1, 2, \cdots, n$. In the "soft" clustering approaches, such as fuzzy c-means [1], M_{ij} can be any continuous value in $[0, 1]$. Let $D(\mathbf{x}_i, \mathbf{x}_j)$ denote a distance metric between two pattern vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. The cost function can be defined as $E_K(M) = \sum_{i=1}^{n} \sum_{j=1}^{K} M_{ij} D(\mathbf{x}_i, \mathbf{m}_j)$, where $\mathbf{m}_j = \sum_{i=1}^{n} M_{ij}\mathbf{x}_i / \sum_{i=1}^{n} M_{ij}$. Vector $\mathbf{m}_j$ is called the center (or prototype) of the j^{th} cluster, $j = 1, 2, \cdots, K$. The most commonly used distance metric is the Euclidean metric: $D_E(\mathbf{x}_i, \mathbf{m}_j) = (\mathbf{x}_i - \mathbf{m}_j)^T(\mathbf{x}_i - \mathbf{m}_j)$.

Various clustering or unsupervised learning algorithms have been proposed in the literature [1, 4, 5]. In spite of the numerous research efforts, data clustering remains a difficult and essentially an unsolved problem [5]. Each clustering algorithm imposes its own structure on the data. A clustering algorithm with the Euclidean distance metric favors hyperspherically-shaped clusters of equal size. This has the undesirable effect of splitting large as well as elongated clusters under some circumstances [3]. Simple feature normalization schemes can not solve this problem [3]. It is easy to find examples of data sets in real applications which do not have spherically-shaped clusters of equal size. For example, in the well-known IRIS data[1] all the three clusters are nonspherically shaped (Figure 2(a)). The nonspherical nature of the clusters can be easily observed by examining their covariance matrices.

*Research supported by NSF grant IRI 9002087

[1]The IRIS data set consists of 150 4-dimensional patterns from 3 classes. It contains 4 measurements on 50 flowers from each of the 3 species of the iris flower (setosa, versicolor, and virginica).

Other distance measures, such as the Mahalanobis distance, can also be used in the clustering criterion to take care of hyperellipsoidal-shaped clusters. The Mahalanobis distance between pattern vector $\mathbf{x}_i$ and the center of the j^{th} cluster $\mathbf{m}_j$ is defined as $D_M(\mathbf{x}_i, \mathbf{m}_j) = (\mathbf{x}_i - \mathbf{m}_j)^T \Sigma_j^{-1} (\mathbf{x}_i - \mathbf{m}_j)$, where Σ_j^{-1} is the inverse of the $d \times d$ covariance matrix of the patterns belonging to the j^{th} cluster. However, there are a number of difficulties associated with incorporating the Mahalanobis distance in a clustering method: (i) The Mahalanobis distance requires computation of the inverse of the sample covariance matrix every time a pattern changes its cluster category (sequential mode), which is computationally expensive. (ii) If the number of patterns in a cluster is small compared to the input dimensionality d, then the $d \times d$ sample covariance matrix of the cluster may be singular. (iii) The K-means clustering algorithm with the Mahalanobis distance metric tends to produce unusually large or unusually small clusters. As a result, the squared-error clustering method with Euclidean metric is the most commonly used partitional clustering method in practice.

We propose a neural network (HEC) for hyperellipsoidal clustering with a regularized Mahalanobis distance which can adaptively estimate the hyperellipsoidal shape of each cluster. Introducing the regularized Mahalanobis distance can resolve the singularity of estimating the inverse covariance matrices when the number of patterns in a cluster is less than or not significantly larger than the input dimensionality, and can prevent the clustering algorithm from producing unusually large or unusually small clusters.

2 Network for Hyperellipsoidal Clustering (HEC)

In the previous section, we mentioned the three difficulties associated with integrating the Mahalanobis distance metric into the K-means clustering algorithm. The proposed regularized Mahalanobis distance metric can overcome the second and the third difficulties. However, the first difficulty dealing with the computational issue still remains unsolved. Real-time implementation of the K-means algorithm with the Mahalanobis distance metric may be pursued only through hardware implementation. Which architecture to use is an important issue. In this section, we present a self-organizing network (HEC) for hyperellipsoidal clustering.

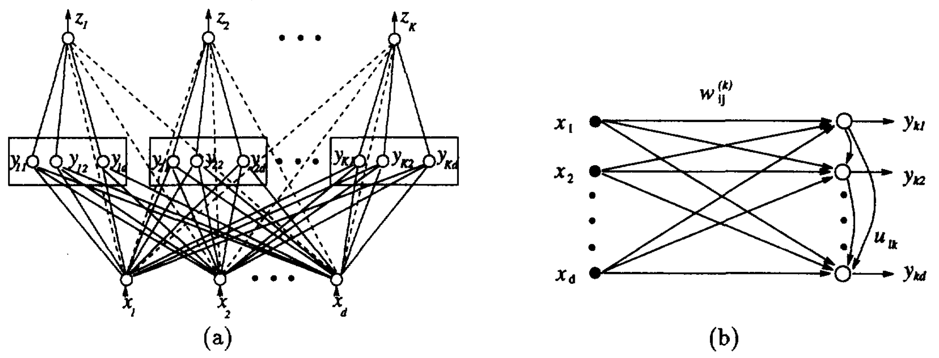

Figure 1: (a) Architecture of the neural network (HEC) for hyperellipsoidal clustering. (b) PCA network proposed by Rubner et al.

Figure 1(a) shows the diagram of the proposed network architecture for hyperellipsoidal clustering. The input layer has d input nodes, where d is the number of features. The hidden layer is composed of $K \times d$ linear nodes which are grouped into K subnetworks with d nodes each, where K is the number of clusters. Each subnetwork is designed to perform a principal component analysis of a cluster consisting of those input patterns which activate the corresponding output node. We choose the PCA network proposed by Rubner et al. [9] as a building block in our network for hyperellipsoidal clustering. Each PCA subnetwork (as shown in Figure 1(b)) consists of d input nodes and d output nodes. Each input node i is connected to each output node j with connection strength $w_{ij}^{(k)}$ (index k here indicates the k^{th} subnetwork). All the output nodes are hierarchically organized in such a way that the output node i is connected to the output node j with connection strength $u_{ij}^{(k)}$ if and only if $i < j$ (not shown in Figure 1(a)). The weights on connections between the input nodes and the output nodes of the PCA subnetwork are adjusted according to the Hebbian rule. The lateral synaptic weights are updated according to the anti-Hebbian rule. Rubner and Tavan [9] proved that if the learning parameters are properly chosen, then all the lateral weights will rapidly vanish and the network will converge to a state in which the d weight vectors associated with the d output nodes are the d eigenvectors of the covariance matrix of input patterns with eigenvalues, $\lambda_{k1} \geq \lambda_{k2} \cdots \geq \lambda_{kd}$. Therefore, the weights on the connections between input layer and the k^{th} subnetwork will converge to the eigenvector matrix, Φ_k, of the k^{th} cluster, $k = 1, 2, \cdots, K$. To produce this result, the implicit requirement in Rubner's PCA algorithm is that the centroid of each cluster should be subtracted from the input pattern vectors. These centroids are stored in the connections between the input nodes and the output nodes.

Let h_{kj} be the output of the j^{th} unit in the k^{th} PCA subnetwork and $Var\{h_{kj}|z_k = 1\}$ be the variance of h_{kj} based on only those transformed input pattern vectors which are assigned to the k^{th} cluster by the output layer. Then, $\lambda_{kj} = Var\{h_{kj}|z_k = 1\}$ is the j^{th} eigenvalue of the covariance matrix of the k^{th} cluster. If we scale h_{kj} by s_{kj}, where $s_{kj} = 1/\sqrt{Var\{h_{kj}|z_k = 1\} + \varepsilon}$, $k = 1, 2, \cdots, K$, $j = 1, 2, \cdots, d$, and ε is a small positive number to prevent a zero denominator (we use $\varepsilon = 0.000001$), then each subnetwork "whitens" its corresponding cluster [8].

The output layer has K nodes corresponding to K clusters, and is basically a "winner-take-all" type of network. Competitive learning is performed in the output layer. The input layer is fully connected to the output layer (by dashed lines in Figure 1(a)). However, all the output nodes in the k^{th} subnetwork are connected only to the k^{th} node in the output layer. Let m_{ik} denote the weight on the connection between the i^{th} input node and the k^{th} node in the output layer, and v_{jk} be the weight on the connection between the j^{th} output node of the k^{th} subnetwork and the k^{th} output node in the output layer. Each output node computes the weighted squared-Euclidean distance between its inputs (both the input pattern and outputs of the connected subnetwork) and the stored pattern on the connections. Let D_k denote the total signal intensity (potential value) which the k^{th} output node receives from the input layer and hidden layer. Then,

$$D_k = (1 - \lambda) \sum_{j=1}^{d} (v_{jk} - y_{kj})^2 + \lambda \sum_{i=1}^{d} (m_{ik} - x_i)^2, \; k = 1, 2, \cdots, K, \tag{1}$$

where $0 \leq \lambda \leq 1$ is a regularization parameter. The output values, z_k, $k = 1, 2, \cdots, K$, are determined by competitive learning. The node with the *smallest* potential value D_{k^*} will be the winner.

The weight vector $\mathbf{m}_k = (m_{1k}, m_{2k}, \cdots, m_{dk})^T$ is designed to store the centroid (in the original input space) of the k^{th} cluster formed by the network. Kohonen's LVQ algorithm [7], for example, can be used to learn these weight vectors. The weight vector, $\mathbf{m}_k$, is used both in learning the weight matrix Φ_k (to subtract it from the input pattern vector) and in computing the distance D_k for competitive learning as a stabilizer. Similar to $\mathbf{m}_k$, the weight vector $\mathbf{v}_k = (v_{1k}, v_{2k}, \cdots, v_{dk})^T$ is designed to store the centroid in the rotated and whitened space by Φ_k and $\mathbf{s}_k$ of the k^{th} cluster currently formed by the network. Again, Kohonen's LVQ algorithm can be used to learn $\mathbf{v}_k$, $k = 1, 2, \cdots, K$.

Rewriting Equation (1) in the vector form, we obtain

$$D_k = (1 - \lambda)(\mathbf{y}_k - \mathbf{v}_k)^T(\mathbf{y}_k - \mathbf{v}_k) + \lambda(\mathbf{x} - \mathbf{m}_k)^T(\mathbf{x} - \mathbf{m}_k). \tag{2}$$

Since $s_{kj} = 1/\sqrt{\lambda_{kj} + \varepsilon}$, and $\mathbf{y}_k = \Lambda_k^{-1/2}\Phi_k\mathbf{x}$, where $\Lambda_k = \text{diag}\{\lambda_{k1}, \lambda_{k2}, \cdots, \lambda_{kd}\} + \varepsilon I$, then $\mathbf{v}_k = \Lambda_k^{-1/2}\Phi_k\mathbf{m}_k$. Substituting $\mathbf{y}_k$ and $\mathbf{v}_k$ into Equation (2) and using $\Phi_k^T\Lambda_k^{-1}\Phi_k = (\Sigma_k + \varepsilon I)^{-1}$, we obtain

$$D_k = (\mathbf{x} - \mathbf{m}_k)^T[(1 - \lambda)(\Sigma_k + \varepsilon I)^{-1} + \lambda I](\mathbf{x} - \mathbf{m}_k), \tag{3}$$

where Σ_k is the covariance matrix of the k^{th} cluster. Let $\Sigma_k^{-1*} = (1 - \lambda)(\Sigma_k + \varepsilon I)^{-1} + \lambda I$. We refer to Σ_k^{-1*} as the regularized inverse of the covariance matrix Σ_k, and D_k as the regularized Mahalanobis distance. When $\lambda = 0$ and $\varepsilon = 0$, D_k becomes the Mahalanobis distance metric. When $\lambda = 1$, D_k reduces to the Euclidean distance metric. When $0 < \lambda < 1$, D_k is a linear combination of the Mahalanobis distance and the Euclidean distance. Therefore, λ can be used as a parameter to control the degree that the distance metric deviates from the commonly used Euclidean distance. In situations where Σ_k^{-1} can not be reliably estimated or learned, a larger value of λ should be used. The role of ε is to convert a singular matrix (ill-posed problem) to a nonsingular matrix by adding a diagonal matrix with small diagonal elements.

The validity of the resulting partition (cluster validity problem) is an important yet difficult problem [5]. In this paper, the compactness of a cluster is measured by the significance level of the Kolmogorov-Smirnov test on the multivariate normality of clusters. Under the Gaussian cluster assumption, we can prove that the Mahalanobis distance satisfies the χ^2 distribution with number of degrees of freedom of d, which is the input dimensionality. Therefore, the test reduces to the Kolmogorov-Smirnov test on the distribution of the Mahalanobis distances of patterns to the their centroid with respect to the theoretical χ^2 distribution.

We summarize the learning algorithm for determining the weights and scaling factors in the HEC network as follows.

1. Initialization: set weights $m_{ik} = w_{ik}$, $i = 1, 2, \cdots, d$, $k = 1, 2, \cdots, K$, to small random values; set all the weights, $w_{ij}^{(k)} = 1$ if $i = j$, and 0 otherwise; set all the scaling factors $s_{kj} = 1$. Set $\varepsilon = 0.000001$ and $\lambda_0 = 1.0$. Therefore, the network starts with a preference for hyperspherical shaped clusters.
2. Learn all the weights associated with the output nodes, $\mathbf{m}_k$, $k = 1, 2, \cdots, K$, using the LVQ algorithm with the activation function defined in Equation (1). (In our implementation, a batch-mode algorithm is used).
3. Learn all the weights in the subnetworks using the modified Rubner's PCA algorithm (with a momentum term and scaling factors). Note that upon presentation of a pattern to the network, only the weights in the winner PCA subnetwork are updated.
4. Decrease the value of λ using $\lambda_t = \max(\lambda_{min}, \lambda_{t-1} - \Delta\lambda)$.
5. Repeat steps 2-4 until the cluster membership of input patterns does not change, or the maximum number of cycles is reached.
6. Perform the Kolmogrov-Smirnov test on all the clusters.

Step 6 can also be performed for every cycle so that we can monitor the compactness of the currently-formed clusters. We can see that in the HEC learning algorithm, all the weights associated with a subnetwork and its corresponding output node are not updated unless the output node becomes the winner. However, some variations of this algorithm can be considered. In our implementation, the regularization parameter $\varepsilon = 0.000001$, and λ is a decreasing function of number of cycles (number of times that the algorithm performs Steps 2-4 in the HEC learning algorithm): $\lambda_t = \max(\lambda_{min}, \lambda_{t-1} - \Delta\lambda)$, with $\lambda_0 = 1.0$ and the values of λ_{min} and $\Delta\lambda$ specified by the user.

3 Simulations

We have applied the HEC network to a number of artificial and real data sets. In all our experiments, $\lambda_{min} = 0.0$, $\Delta\lambda = 0.2$, and the maximum number of cycles (Steps 2-4) is 10. Due to the limited space, we only report in this paper the results for the well-known IRIS data and texture segmentation. The batch-mode K-means algorithm is used for comparison purpose in all our experiments. Since the clustering solutions of the K-means algorithm and the HEC network depend on initial centroids or initial weights, we repeat all the experiments ten times with different initializations and the best solutions obtained by both the methods are reported. We should mention that only few distinct solutions were generated by both the algorithms and the probability of producing the best solution is much higher than producing other solutions.

Since the IRIS data are in a four-dimensional space, for the visualization purpose, we display only its eigenvector projection onto the 2-dimensional plane spanned by the first two principal eigenvectors. The position of each pattern in the map is determined by its eigenvector projection, while its label is obtained by using either the K-means algorithm or the HEC network operating in the original 4-dimensional space. For the 3-cluster solutions ($K = 3$) as shown in Figure 2(a)-(c), both the K-means and the HEC network correctly group patterns from the first class (setosa) into one cluster, because the patterns in the first class (setosa) are well separated from the rest of the data. However, since the second class (versicolor) and the third class (virginica) are slightly overlapped and both are not spherically shaped as we can see from Figure 2(a), the K-means algorithm produces a partition which does not correctly group patterns from these two classes; the compactness of the corresponding two clusters is also poor. On the other hand, the HEC network can recover the clusters which reflect the true classes very well. This can be observed by comparing Figure 2(c) with Figure 2(a). The numerical values of misclassification rate and compactness measure of the clusters generated by the K-means algorithm and the HEC network are shown in Table 1. Note that the

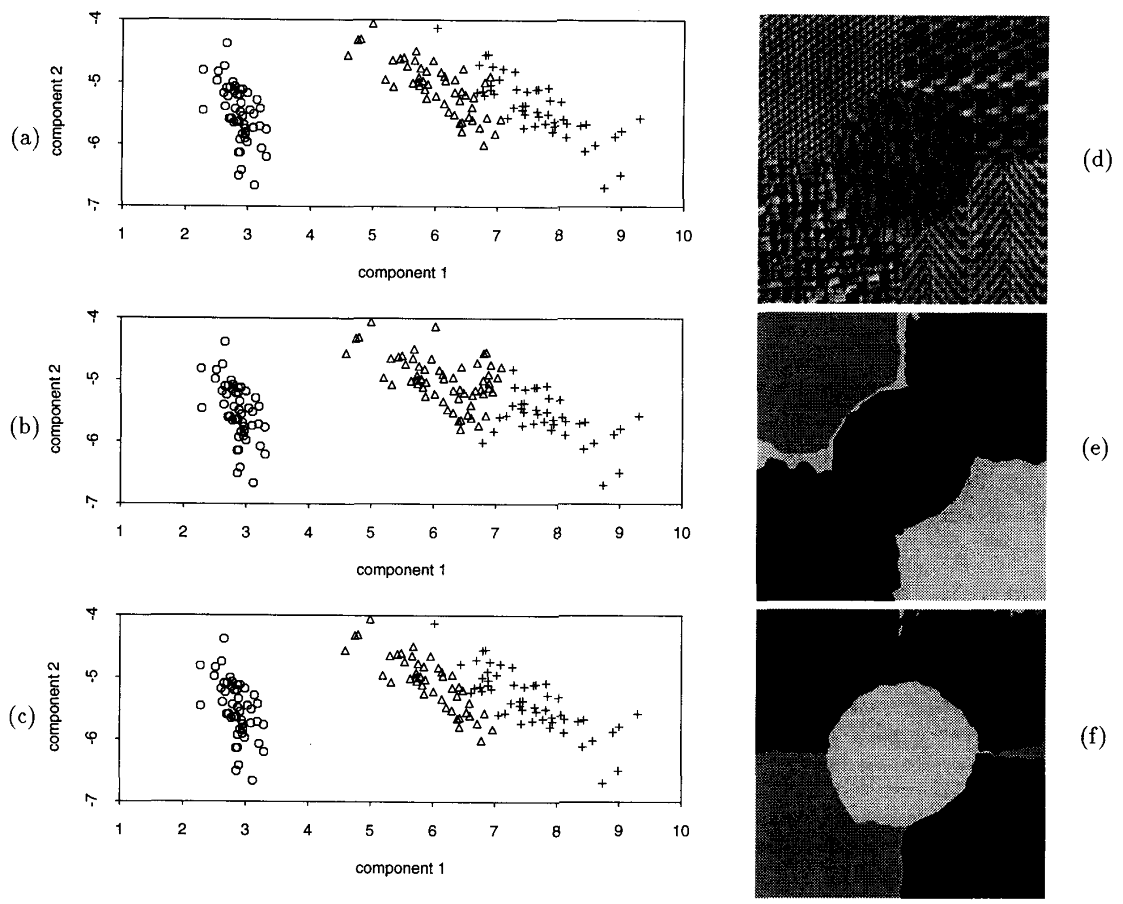

Figure 2: Two-dimensional projection maps (the first two principal components) of the IRIS data ((a)-(c)) and texture segmentation results ((d)-(f)) using the K-means clustering with the Euclidean distance and the HEC network. (a) The three true classes. (b)(c) 3-cluster solutions using the K-means clustering algorithm and the HEC network, respectively. (d) Original composite textured image. (e)(f) 5-cluster segmentations using the K-means clustering algorithm and the HEC network, respectively.

K-means algorithm misclassifies 16 patterns (compared to the true categories), while the HEC network has misclassified only five patterns. The compactness of the second cluster is also significantly improved by using the HEC network at the cost of slightly degrading the compactness of the third cluster. For the 2-cluster solutions, we notice that although the patterns in the first class (setosa) are well-separated from the rest of the data, due to the substantially different spreads of the two natural clusters in the feature space, there are still three patterns from the bigger cluster (mixture of classes 2 (versicolor) and 3 (virginica)) which are assigned to the smaller cluster by the K-means algorithm with the Euclidean distance metric. These three misclassified patterns significantly degrade the compactness of the two resulting clusters. On the other hand, the HEC network correctly produces the two natural clusters. Experiments on artificial data sets have shown even more significant improvements over the K-means algorithm with the Euclidean distance metric.

Figure 2(d) shows a 256 $\times$ 256 texture image. Sixteen Gabor filters [6] (4 highest radial frequencies and 4 different orientations per frequency) are applied to the texture image to obtain 16 256 $\times$ 256 feature images. A total number of 1,000 pixels are randomly chosen to form a set of "training" (without category labels) patterns. The K-means algorithm and the HEC network are applied only to this set of 1,000 patterns instead of all the 64K pixels in order to reduce the amount of computation. The minimum Euclidean distance classifier with the centroids obtained from the K-means algorithm and the trained HEC network are then used to classify every pixel in the image. Note that some improvement on the texture boundaries has been achieved by using the regularized Mahalanobis distance (Figure 2(f)) over the K-means algorithm with the Euclidean distance (Figure 2(e)).

We should point out that although the comparison is performed between the K-means algorithm with the Euclidean distance metric and the HEC network, these conclusions can also be applied to other partitional clustering algorithms and competitive networks such as, Kohonen's LVQ and ART models [2], which use the Euclidean distance metric.

method		K-means			HEC network		
3-cluster	#misclassified/n	16/150			5/150		
solution	compactness of the 3 clusters	0.559	0.676	0.857	0.559	0.897	0.814
2-cluster	#misclassified/n	3/150			0/150		
solution	compactness of the 2 clusters	0.271	0.603		0.559	0.720	

Table 1: The compactness of clusters and number of misclassified patterns by the K-means algorithm and the HEC network for the IRIS data. The eigenvector projections of the 3-cluster solutions are shown in Figures 2(b)-2(c). The larger (close to 1.0) is the value of compactness, the more similar is the cluster to a Gaussian cluster.

4 Conclusions

We have proposed a self-organizing network for hyperellipsoidal clustering. The network performs a partitional clustering using the proposed regularized Mahalanobis distance measure. This distance measure can deal with the ill-posed and poorly-posed problems in estimating the Mahalanobis distance and achieve a tradeoff between using the Euclidean distance and the Mahalanobis distance so as to prevent the HEC network from producing unusually large or unusually small clusters. Experiments on both artificial data and real data have shown that the HEC network provides better clusters than the K-means algorithm. Same improvement will be achieved over any other partitional clustering method which uses the Euclidean distance metric if the underlying hyperellipsoidal assumption made in the HEC network is valid. However, even if this hyperellipsoidal assumption is not true, the HEC network can be used to tune the results produced by the K-means algorithm. The Komogrov-Smirnov test on the distribution of the Mahalanobis distances between patterns in a cluster and the corresponding cluster center provides a compactness measure and validity test of the hyperellipsoidal assumption. In conclusion, the HEC network is an admissible clustering algorithm [5] which should be utilized along with other well-known clustering algorithm, to understand the structure of multidimensional patterns.

References

[1] J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* New York: Plenum, 1981.

[2] G. Carpenter and S. Grossberg. ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, 3:129–152, 1990.

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

[4] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[5] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Englewood Cliffs, NJ: Prentice Hall, 1988.

[6] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

[7] T. Kohonen. *Self Organization and Associative Memory.* Third edition, Springer-Verlag, 1989.

[8] J. Mao and A. K. Jain. Discriminant analysis neural networks. In *Proc. of IEEE Intl. Conf. on Neural Networks*, volume 1, pages 300–305, San Francisco, CA, March 1993.

[9] J. Rubner and P. Tavan. A self-organizing network for principal component analysis. *Europhysics Letters*, 10:693–698, 1989.