# Pattern Classification Based On Multi-Hyperellipsoid Clustering.

Yao Cai

# Pattern Classification Based On Multi-Hyperellipsoid Clustering

A Thesis Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment of the Requirement for the Degree

Master of Science

University of Nebraska at Omaha

by

Yao Cai

July 1995

# THESIS ACCEPTANCE

Acceptance for the faculty of the Graduate College, University of Nebraska, in partial fulfillment of the requirements for the degree of Master of Science, University of Nebraska at Omaha.

Committee

| Name | Department/School |
|---|---|
| Stanley A. Wileman Jr. | Computer Science/Arts & Science |
| [illegible] | Computer Science / Arts & Scien |
| Steve From | Math / Arts & Sciences |
| | |

Chairperson [illegible]

Date 7/21/95

# ACKNOWLEDGEMENTS

My deepest thanks go to my advisor, Dr. Qiuming Zhu, for his support, patience, encouragement, comment, and inspiration. I feel very fortunate to have him as my advisor. I learned a lot from his class and in the process of doing this thesis.

My sincere appreciation goes to Professor Stanley Wileman, Dr. Zhengxin Chen, and Dr. Steven From for graciously serving on my committee. I greatly appreciate their suggestions, time, and support.

Finally, I want to express my special thanks to my wife, Meiyu. Without the opportunity she made for me a few years ago, I would not possibly do this thesis here in UNO. Moreover, her love, support, and encouragement are very important for the accomplishment of this thesis.

# ABSTRACT

Traditional model-based pattern classification is based on the assumption that the distribution of the training samples of each pattern class can be formulated by a single statistical function. It is difficult to make an accurate classification by the traditional method when the training samples of different classes do not bind to this assumption. The main contribution of this research is the development of a new clustering technique, called Multi-Hyperellipsoid Clustering, that is able to handle any irregular pattern distributions. The new method uses a supervised maximum likelihood estimation to derive a set of distribution functions for the training samples of each class, and then uses an improved Bayesian probability decision model to partition the pattern space. The new classifier achieved a higher rate of correct classification than the traditional method, with respect to some rather complex pattern distributions in a number of test examples.

*Key terms:*

Pattern classification, multi-hyperellipsoid clustering, Gaussian distribution, maximum likelihood estimation, Bayes decision rule.

# 1. Introduction

Suppose there are $w$ classes of samples, i.e. $S = \bigcup_{i=1}^{w} S_i$, in an n-dimensional space; and there are $q_i$, $i = 1,2,\cdots,w$, samples in each class, i.e. $S_i = \{s_1, s_2, \cdots, s_{q_i}\}$. A Pattern Classifier can be viewed as a mapping or decision-making mechanism that partitions an n-dimensional feature space into $w$ subspaces. For any given feature vector **x** in this space, the classifier will provide a class-label as its output. That is, if $\pi_i(\mathrm{x}), i = 1,2,\cdots,w$, is a set of discriminant functions for the classification, then the classifier will assign a class-label $\omega_k$ to vector **x** such that $\pi_k(\mathrm{x})$ has the maximum value. This decision-making procedure can be represented as a mapping machine in the following figure.

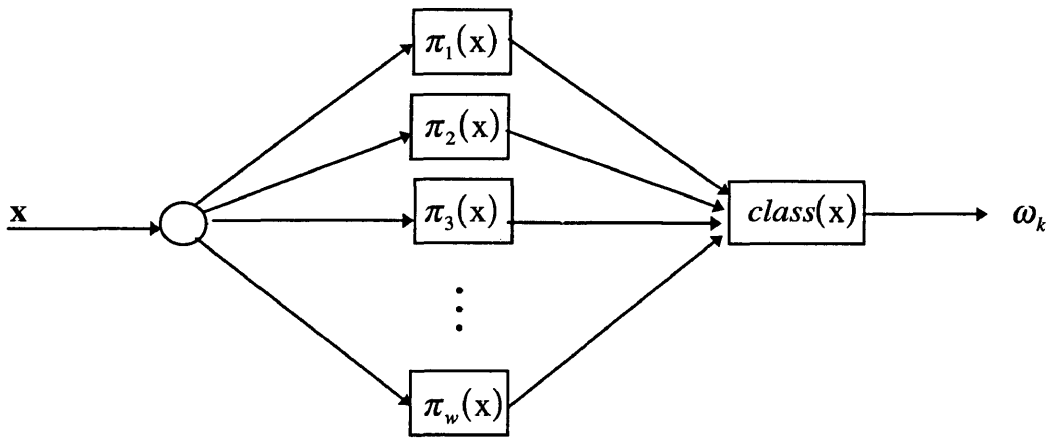


**Figure 1.1 Mapping Machine of Pattern Classification**

Pattern classification is a very important technique used in many computer applications such as image processing, speech recognition, disease diagnosis, biological slide analysis, weather forecasting, machine vision, etc.

Traditionally, statistical approach is a fundamental technique used in pattern classification. The technique works in this fashion: (1) by assuming a form of the class-conditional probability distribution functions and using parameter estimations on the given training set, a set of decision (or say, discriminant) functions is found; (2) based on these discriminant functions, a classification model is constructed to partition the feature space into *w* subspaces such that each subspace corresponds to a class; (3) any vector in the feature space should locate in a specific subspace. In practical problems, however, the form of the desired class-conditional probability distribution functions are often unknown. Therefore, the theoretically ideal classifier is hardly applied in practical problems without loss of statistical precision.

The conventional classification models are based on the assumptions of Gaussian normal distribution in general. A Gaussian normal distribution function can be regarded as a hyperellipsoid in an n-dimensional feature space. Therefore, in conventional parameter estimation, there is a single hyperellipsoid corresponding to the pattern distribution for each class. In this research, a model based on unions of multiple pattern distributions is developed. The models can be in Gaussian normal distribution or others. There is a major difference between our model and the conventional approach. In our approach, samples of each class are modeled by multi-hyperellipsoids, instead of a single hyperellipsoid. The advantage of this modeling technique is that it can be used in case where the distribution of the training samples is not in single Gaussian distribution.

Moreover, compared with the traditional model, our model provides a better performance in classification when the training samples are in complex and irregular distributions.

There are lots of studies been done in pattern classification and parameter estimation. Basically, they can be categorized as four techniques: 1) Statistical Approaches; 2) Neural Network Approaches; 3) Fuzzy Clustering Approaches; and 4) Linear Programming Approaches. R. O. Duda and P. E. Hart [28] presented the classic work on statistical theory for the combined classification and parameter estimation. Similar discussions about the statistical approaches on the classification problem can be found in [5], [18] and [30]. Clustering techniques with unsupervised learning was presented as a popular approach for pattern analysis in [30] and [24]. The popularity of clustering has spawned a library of clustering algorithms (R. K. Blashfield et. al.[25], and A. K. Jain and R. C. Dubes [2]). Neural network approaches for clustering and classification are shown in [30] and [33]. R. Schalkoff [27] covered the statistical, structural and neural-net approaches in pattern classification area. Many significant and broad studies in nearest neighbor techniques within the fields of pattern classification are presented in [4]. Fuzzy clustering was studied by V. D. Gesu [31], I. Gath and A. B. Geva[11], H. Ishibuchi et. [10], D. E. Gustafson and W. C. Kessel [6]. Multisurface method was a famous linear programming approach ([19], [20], and [21]).

This thesis is organized as follows : In Section 2, we summarize the basic processes for a general probability-based classifier and indicate its limitations in solving certain classification problems. In Section 3, we describe the multi-hyperellipsoid clustering algorithm which performs supervised clustering on the labeled training sample set. In Section 4, we illustrate the construction of the Multi-HyperEllipsoid Classifier (M-HEC) which is derived from the clustering algorithm described in Section 3. In Section 5, we present the comparison of the M-HEC with the conventional classifier, neural-network based classifiers, and fuzzy-clustering based classifiers. Section 6 contains conclusion remarks.

## 2. Model-Based Classifications

### *2.1 Bayes Decision Rule*

A pattern classification process starts with a set of measurements that can be viewed as points in a vector space. That is, the measurement is a feature vector **x** in an n-dimensional space, $\mathrm{x} = \left[x_1, x_{2,} \cdots, x_n\right]$.

Statistically, an optimal classifier is the one which minimizes the probability of overall decision error, i.e. mis-classification, on the samples in the vector space. For a given observation vector **x** of unknown class membership, if class-conditional distributions $p(\mathrm{x}|\omega_k)$ and prior probabilities $P(\omega_k)$ for the class $\omega_k$, $k = 1,2,\cdots,w$, are provided, then a posterior probability $P(\omega_k|\mathrm{x})$ can be computed by **Bayes Rule**:

$$P(\omega_k|\mathrm{x}) = \frac{p(\mathrm{x}|\omega_k)P(\omega_k)}{p(\mathrm{x})}, \qquad (2.1)$$

where

$$p(\mathbf{x}) = \sum_{i=1}^{w} p(\mathbf{x}|\omega_i)P(\omega_i). \qquad (2.2)$$

Bayes decision method finds the membership label *class*(**x**) for the observation feature vector **x** such that the overall probability of classification-error, in the decision theoretic terminology, the *expected loss* (or risk), is minimized. Suppose for the given observation vector **x**, we are going to make decision $\delta_j$. If the true state of nature is $\omega_k$, we will incur a loss $\tau(\delta_j|\omega_k)$. Since $P(\omega_k|\mathbf{x})$ is the probability that the true state of nature is $\omega_k$, the expected loss (or risk) associated with decision $\delta_j$ is

$$R(\delta_j|\mathbf{x}) = \sum_{k=1}^{w} \tau(\delta_j|\omega_k)P(\omega_k|\mathbf{x}). \qquad (2.3)$$

Then, the overall expected loss is given by

$$R(\delta(\mathrm{x})) = \int R[\delta(\mathrm{x})|\mathrm{x}]p(\mathrm{x})d\mathrm{x} \qquad (2.4)$$

where $\delta(\mathrm{x})$ is the decision function for vector **x** , which assumes one of the decision among the decision set $\{\delta_1, \delta_2, \cdots, \delta_d\}$. The integral extends over the entire feature space. Clearly, if $\delta(\mathrm{x})$ is chosen so that $R[\delta(\mathrm{x})|\mathrm{x}]$ is as small as possible for every vector **x**, then the overall expected loss will be minimized. Therefore, to minimize the overall loss, we need to compute the conditional risk

$$R(\delta_j|\mathbf{x}) = \sum_{k=1}^{w} \tau(\delta_j|\omega_k)P(\omega_k|\mathbf{x}) \qquad (2.5)$$

for $j = 1,2,\cdots,d$ and select the decision $\delta_j$ for which $R(\delta_j|\mathbf{x})$ is minimum. This is what we called the *Bayes Decision Rule.*

In a typical classification problem, each sample is usually associated with a unique member of the $w$ classes, and the $\delta_j$ is usually interpreted as the decision that the true state of nature of the sample is $\omega_j$. If decision $\delta_j$ is made and the true state of nature is $\omega_k$, then the decision is correct if $k = j$, and is error if $k \neq j$. To avoid decision errors, it is nature to seek a decision rule that minimizes the overall probability of error. In certain cases, we can define the loss function as

$$\tau(\delta_j|\omega_k) = \begin{cases} 0, & j = k \\ 1, & j \neq k \end{cases} \qquad j,k = 1,2,\ldots,w \tag{2.6}$$

This loss function assigns no lose to a correct decision, and assigns a unit loss to any error. The risk corresponding to this loss function is precisely the overall probability of error, since the conditional risk is

$$R(\delta_k|\mathrm{x}) = \sum_{i=1}^{w} \tau(\delta_k|\omega_i)P(\omega_i|\mathrm{x}) = \sum_{i\neq k} P(\omega_i|\mathrm{x}) = 1 - P(\omega_k|\mathrm{x}) \tag{2.7}$$

and $P(\omega_k|\mathbf{x})$ is the conditional probability that decision $\delta_k$ is correct.

From Eq.(2.7), we can see that the class probability functions, $P(\omega_k|\mathbf{x})$ s, dominate the computation of conditional risks. They describe the class boundaries in the n-dimensional feature space which are the decision bases of many model-based classifiers. Generally, they are referred as Bayes' discriminant functions.

## *2.2 Maximum Likelihood Estimation*

From the above discussion, we learned that the Bayes classifier is determined primarily on the class probability functions $P(\omega_k|\mathrm{x})$. If the prior probabilities $P(\omega_k)$ of all classes are equal (this is the general assumption for many pattern classification problems), then, according to Eq. (2.1), the class conditional densities $p(\mathrm{x}|\omega_k)$ uniquely determine a set of discriminant functions. Decisions are made based on these discriminant functions such that if $p(\mathrm{x}|\omega_i) > p(\mathrm{x}|\omega_j), \forall j \neq i$, then $\mathrm{x} \in \omega_i$.

Of the various probability density functions that have been investigated, none has received more attention than the multivariate normal density. The general multivariate normal density in Gaussian distribution is defined as

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{\left[-\frac{1}{2}(\mathbf{x}-\mu)^t \Sigma^{-1}(\mathbf{x}-\mu)\right]} \tag{2.8}$$

where **x** is an n-dimensional vector, $\mu$ is an n-component mean vector, $\Sigma$ is an $n \times n$ covariance matrix, $(\mathbf{x}-\mu)^t$ is the transpose of $(\mathbf{x}-\mu)$, $\Sigma^{-1}$ is the inverse of $\Sigma$, and $|\Sigma|$ is the determinant of $\Sigma$. Formally, $\mu$ is the expectation of **x** and $\Sigma$ is the expectation of $(\mathbf{x}-\mu)(\mathbf{x}-\mu)^t$. That is,

$$\mu = Expect[\mathbf{x}] = \int_{-\infty}^{+\infty} \mathbf{x}p(\mathbf{x})d\mathbf{x} \tag{2.9}$$

and

$$\Sigma = Expect\left[(\mathrm{x}-\mu)(\mathrm{x}-\mu)^t\right] = \int_{-\infty}^{+\infty}\left[(\mathrm{x}-\mu)(\mathrm{x}-\mu)^t\right]p(\mathrm{x})d\mathrm{x} \qquad (2.10)$$

where $p(\mathrm{x})$ is the density function. The expected value of a vector or a matrix is found by taking the expected values of its components.

Suppose there is a training set with $k$ samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k\}$. By applying Maximum Likelihood Estimation on this sample set, the $\mu$ and $\Sigma$ can be obtained by (*see* R. O. Duda and P. E. Hart [28], p.49),

$$\mu = \frac{1}{k}\sum_{i=1}^{k}\mathbf{x}_i \qquad (2.11)$$

and

$$\Sigma = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{x}_i-\mu)(\mathbf{x}_i-\mu)^t \qquad (2.12)$$

Thus, the maximum likelihood estimations for the mean vector and the covariance matrix are the sample mean and the arithmetic average of the n matrices $(\mathbf{x}_i-\mu)(\mathbf{x}_i-\mu)^t$, respectively.

The multivariate normal density is completely determined by $n+\frac{n(n+1)}{2}$ parameters : 1) the elements of the mean vector $\mu$ ; and 2) the independent elements of the covariance matrix $\Sigma$. Geometrically, samples drawn from a normal population tend to fall in a single cluster. The center of the cluster is determined by the mean vector, and the shape of the cluster is determined by the covariance matrix. It follows from Eq.(2.8) that the loci of points of constant density are hyperellipsoids for which the quadratic form

$(\mathbf{x}-\mu)^t\Sigma^{-1}(\mathbf{x}-\mu)$ is constant. The principal axes of these hyperellipsoids are given by the eigenvectors of $\Sigma$ and the lengths of these axes are determined by the eigenvalues. The quantity

$$r=\sqrt{(\mathbf{x}-\mu)^t\Sigma^{-1}(\mathbf{x}-\mu)} \tag{2.13}$$

is called the *Mahalanobis Distance* from **x** to $\mu$. Thus, the contours of constant density are hyperellipsoids with a constant Mahalanobis Distance to the point $\mu$. The volume of these hyperellipsoids measures the scatter of the samples about the mean.

Suppose that there is a set S of labeled training samples in which each sample is associated with a specific class, i.e., $S=\bigcup_{i=1}^{w}S_i$ , $S_i\bigcap S_j=\Phi, \mathit{for}\quad i\neq j$. Moreover, for each $\mathbf{x}\in S$ , there exists a *k*, $(k=1,2,\cdots,w)$, such that $\mathbf{x}\in S_k$. A pattern classifier can be formed by applying the Maximum Likelihood estimation in a learning process that uses the training samples. The general supervised learning based classifier works as follows: (1) *independently* assuming a functional form of $p(\mathrm{x}|\omega_k)$ for each class; (2) a parameter estimation procedure, e.g. Maximum likelihood estimation, is used to determined the parameters in $p(\mathrm{x}|\omega_k)$; (3) the probability density functions $p(\mathrm{x}|\omega_k)$ form a set of discriminant functions; (4) based on these functions, a set of decision rules are established for a classifier. More discussions on maximum likelihood estimation and multivariate normal density can be found in books by Charles W. Therrien [5] and R. O. Duda and P. E. Hart [28].

## *2.3 Problems in the Maximum Likelihood Classifier*

In real world applications, the functional form of probability distribution for the training samples is not predetermined. Furthermore, the training samples of different classes are not completely independent. For example, in Figure 2.1, we have a training sample set which includes samples in two classes in a 2-dimensional space.

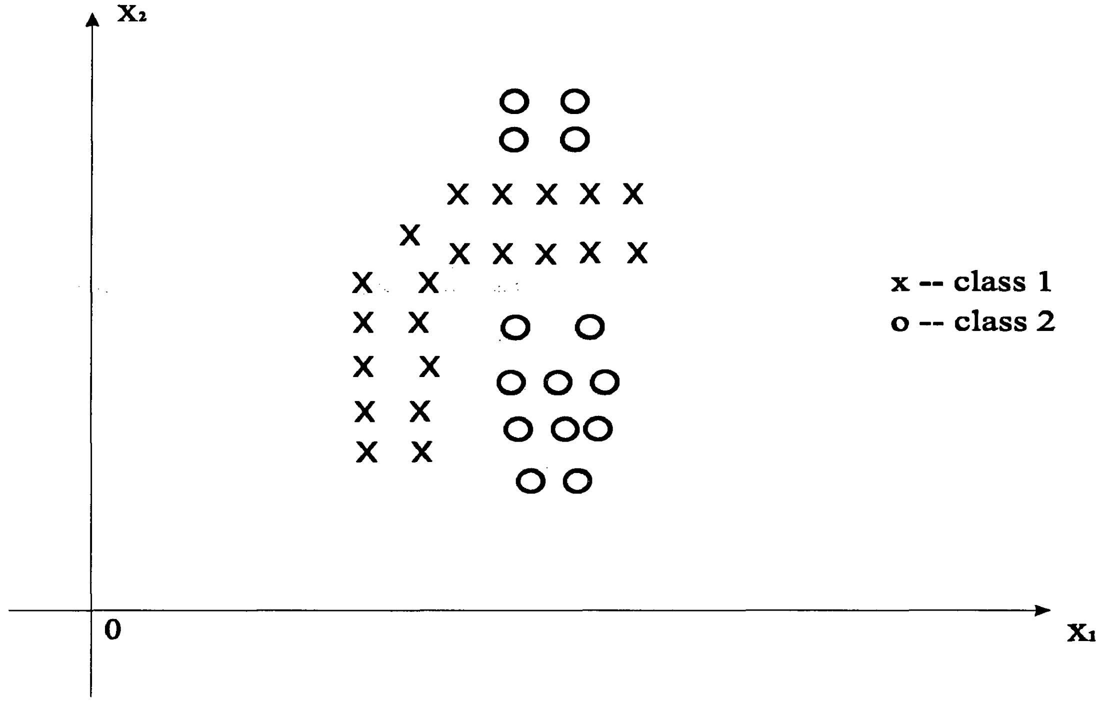


**Figure 2.1 Training Samples**

The class-conditional probability distributions $p(\mathrm{x}|\omega_1)$ and $p(\mathrm{x}|\omega_2)$, which are computed by the maximum likelihood estimation based on the assumptions of Gaussian distribution, are depicted in Figure 2.2.

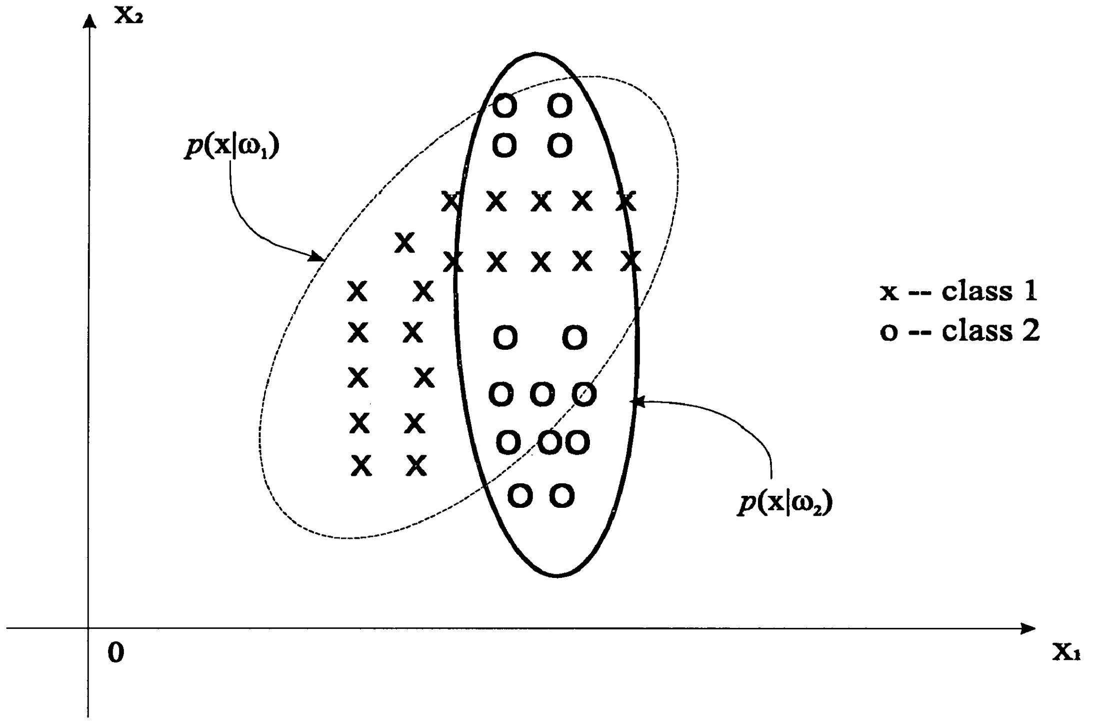


**Figure 2.2 Result of Gaussian Clustering**

It is obvious that the Gaussian distribution assumption does not fit to the nature of the distribution of the samples in class 1 or both classes. The use of a single Gaussian distribution for $p(\mathrm{x}|\omega_1)$ and $p(\mathrm{x}|\omega_2)$ causes a big error between the real distribution and the estimated one. Even though the samples in class 2 seems in Gaussian distribution, the resulting estimation of $p(\mathrm{x}|\omega_2)$ may not be good enough to be used as a discriminant function against $\omega_1$. Therefore, it even be not advantageous to simply estimate the $p(\mathrm{x}|\omega_2)$ independent of the class 1 samples using the Gaussian distribution functional form.

To solve these problems, we developed a new model in this research to compute the sample density distribution function $p(\mathrm{x}|\omega_k)$ for each class $\omega_k$. We call this approach Multi-Hyperellipsoid Clustering. Instead of assuming a single $p(\mathrm{x}|\omega_k)$ for each class, a set of locally-bound normal density functions are used in our approach to find the $p(\mathrm{x}|\omega_k, \varepsilon_{ki})$ from the training samples in $S_k$. In the procedure of finding $p(\mathrm{x}|\omega_k, \varepsilon_{ki})$, the influences of samples in other classes are considered. Though these local sample normal density functions could be modeled in Gaussian distribution by themselves, they should be distinctive enough that no significant interference should happen with the distributions of other classes. Thus, sample normal density function $p(\mathrm{x}|\omega_k)$ for class $\omega_k$ is computed by the combination of these multiple local normal density functions $p(x|\omega_k, \varepsilon_{ki})$. By this way, the sample normal density functions of different classes will not interfere significantly with each other. As a result, they can be used as the discriminant functions to partition the feature space in classes. This method is based on the assumption that even the distribution of a sample set is not in Gaussian distribution, the subset of it can be treated as Gaussian distributions.

For the example mentioned above, the proposed model has the clustering result as shown in Figure 2.3.

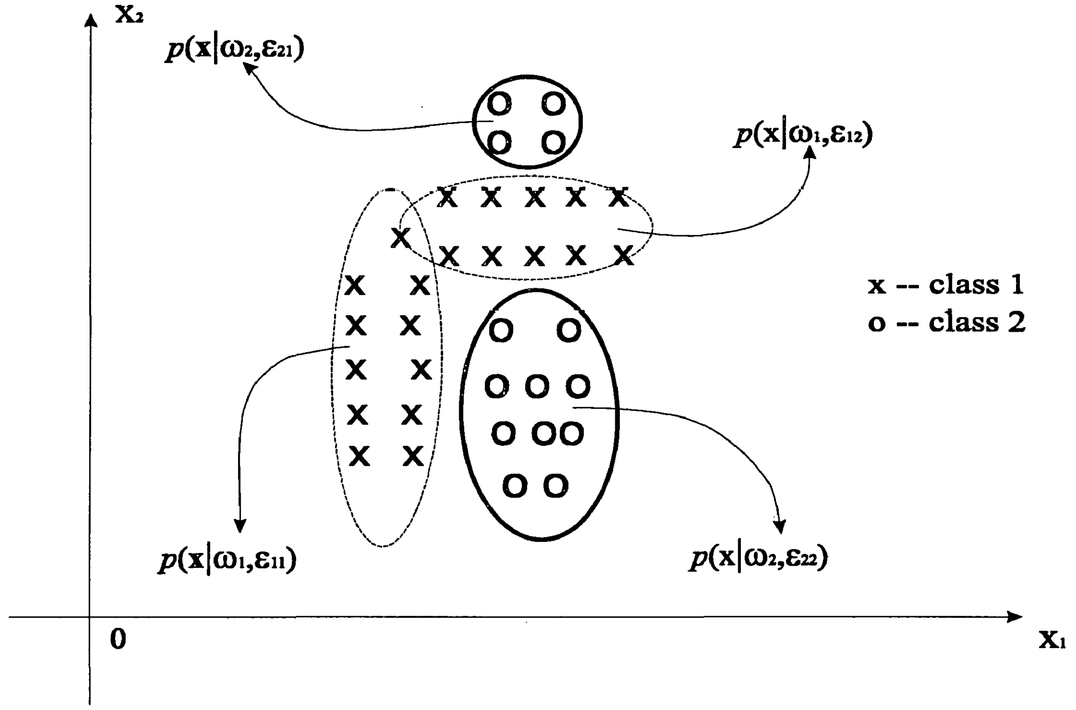


**Figure 2.3 Result of Multi-Hyperellipsoid Clustering**

According to the resulting figure, we have four local distributions : $p(\mathrm{x}|\omega_1,\varepsilon_{11})$ and $p(\mathrm{x}|\omega_1,\varepsilon_{12})$ for class 1, $p(\mathrm{x}|\omega_2,\varepsilon_{21})$ and $p(\mathrm{x}|\omega_2,\varepsilon_{22})$ for class 2. Then,

$$p(\mathrm{x}|\omega_1)=\frac{p(\mathrm{x}|\omega_1,\varepsilon_{11})+p(\mathrm{x}|\omega_1,\varepsilon_{12})-p(\mathrm{x}|\omega_1,\varepsilon_{11})p(\mathrm{x}|\omega_1,\varepsilon_{12})}{p(\mathrm{x})}, \tag{2.14}$$

and

$$p(\mathrm{x}|\omega_2)=\frac{p(\mathrm{x}|\omega_2,\varepsilon_{21})+p(\mathrm{x}|\omega_2,\varepsilon_{22})-p(\mathrm{x}|\omega_2,\varepsilon_{21})p(\mathrm{x}|\omega_2,\varepsilon_{22})}{p(\mathrm{x})}, \tag{2.15}$$

where

$$p(\mathrm{x})=p(\mathrm{x}|\omega_1,\varepsilon_{11})+p(\mathrm{x}|\omega_1,\varepsilon_{12})+p(\mathrm{x}|\omega_2,\varepsilon_{21})+p(\mathrm{x}|\omega_2,\varepsilon_{22})-p(\mathrm{x}|\omega_1,\varepsilon_{11})p(\mathrm{x}|\omega_1,\varepsilon_{12})-p(\mathrm{x}|\omega_2,\varepsilon_{21})p(\mathrm{x}|\omega_2,\varepsilon_{22}).$$

The $p(\mathrm{x}|\omega_1)$ and $p(\mathrm{x}|\omega_2)$ obviously constitute non-intersected and separable class boundaries between these two classes. They can be used as discriminant functions for the classifier.

# 3. Multi-Hyperellipsoid Clustering Algorithm

Based on the preceding discussion, in this section, we introduce an algorithm which can be used to cluster the given training samples. With this algorithm, any labeled training sample set in an n-dimensional feature space can be clustered into different multi- hyperellipsoid sets. Samples in each class is modeled by its corresponding multi-hyperellipsoid set.

## *3.1 Algorithm Description*

The basic procedure of the algorithm is :

1) let every sample in the training sample set be a hyperellipsoid;

2) merge the closest two hyperellipsoids in same class under the constraint that the resulting hyperellipsoid will not intersect with any hyperellipsoid in other classes;

3) repeat step 2) until that no two hyperellipsoids in a class can be merged without intersecting with hyperellipsoids in other classes.

To simplify the description, we define the following notations:

$w$ ---- the number of classes in discussion,

$q_i$ ---- the number of samples in class $i$ , $i = 1,2,\cdots w$.

$S_i$ ---- a subset of sample set $S$ , which contains the samples in class $i$ , $i = 1,2,\cdots w$.

$\underline{s}$ ---- a vector in n-dimensional space. $\underline{s} = [s_1, s_2, \ldots, s_n]$, $s_i$ is the $i^{th}$ value in vector $\underline{s}$.

$\varepsilon, \rho$ ---- hyperellipsoids in n-dimension space.

$H_i$ ---- the set of hyperellipsoids for class $i$ , $i = 1,2,\cdots w$.

$\|H_i\|$ ---- the number of hyperellipsoids in $H_i$, $i = 1,2,\cdots w$.

$\Phi$ ---- an empty set.

$H\_Merge(\varepsilon_1, \varepsilon_2)$ ---- algorithm returning a hyperellipsoid which is the result of merging two hyperellipsoids $\varepsilon_1$ and $\varepsilon_2$.

**Algorithm** 3.1: Multi-HyperEllipsoid-Clustering Algorithm (M-HEC Algorithm)

```
    Input:       $w$, $q_{i,(i=1,2,\cdots,w)}$, $S_{i,(i=1,2,\cdots,w)}$, C = 1
    Output:      $\{H_i\}$, $1 = 1,2,\cdots,w$

Step 1:          for each class $i$ $(i = 1,2,\cdots,w)$ do begin
Step 1.1:            $H_i \leftarrow \Phi$, $\|H_i\| \leftarrow 0$
Step 1.2:            for each $\underline{s} \in S_i$ do begin
Step 1.2.1:              $\varepsilon \leftarrow H\_Merge(\Phi, \underline{s})$
Step 1.2.2:              $H_i \leftarrow H_i \cup \{\varepsilon\}$, $\|H_i\| \leftarrow \|H_i\| + 1$
                     end;
                 end;
Step 2:          Repeat:
Step 2.1:            find a pair $(\varepsilon_j, \varepsilon_k), \varepsilon_j, \varepsilon_k \in H_i, j,k \in \{1,2,\cdots,\|H_i\|\}$, $j \neq k$ and
                     Distance$(\varepsilon_j, \varepsilon_k)$ is the minimum among all pairs of hyperellipsoids in the
                     hyperellipsoid set $H_t$ $(t = 1,2,\cdots,w)$.
Step 2.2:            $\varepsilon \leftarrow H\_Merge(\varepsilon_j, \varepsilon_k)$
Step 2.3:            if NOT($\varepsilon$ intersect with any $\rho$, $\rho \in H_t$, $1 \le t \le w$ and $t \neq i$) then
                     begin
Step 2.3.1:              remove $\varepsilon_j, \varepsilon_k$ from $H_i$
Step 2.3.2:              $H_i \leftarrow H_i \cup \{\varepsilon\}$, $\|H_i\| \leftarrow \|H_i\| - 1$
                     end;
Step 2.4:        Until there is no such pair $(\varepsilon_j, \varepsilon_k)$ which satisfies the condition in Step 2.1.
Step 3:          Return $\{H_i\}$, $(i = 1,2,\cdots,w)$
```

**Figure 3.1 M-HEC Algorithm**

This algorithm does the following:

1) In Step 1, we let all the hyperellipsoid set , each one corresponding to a class, be a empty set initially. Then, we let each sample be a hyperellipsoid. That is, each hyperellipsoid has only one sample in it. The hyperellipsoid has a center at the position of the sample and zero in all its principal axes. Put these hyperellipsoids into the hyperellipsoid sets which are corresponding to their classes.

2) In Step 2, we merge the hyperellipsoids in each set of hyperellipsoids of the same class such that the merged hyperellipsoid does not intersect with any hyperellipsoid in other classes. This procedure will find out $w$ hyperellipsoid sets, each set corresponding to a class, which cover the given training samples in an n-dimensional space without intersection with hyperellipsoids in other classes.

3) In Step 3, we return all sets of hyperellipsoids.

## *3.2 Properties of the Algorithm*

This Multi-Hyperellipsoid Clustering Algorithm has following properties:

*i*) After the algorithm terminates, there is no intersection between any two hyperellipsoids of different classes;

$$\text{i.e. } \varepsilon_i \cap \varepsilon_i = \Phi, \qquad for(\varepsilon_i \in H_u)\&(\varepsilon_j \in H_v)\&(u \neq v).$$

Step 2.3 in the M-HEC Algorithm assures this property.

*ii*) After the algorithm terminates, each hyperellipsoid set $H \in \{H_1, H_2, \cdots, H_w\}$ is the minimum set with the property *i*);

That is,

$$\forall \varepsilon_i \forall \varepsilon_j \big( (\varepsilon_i \in H_u) \& (\varepsilon_j \in H_u) \big) \Rightarrow$$
$$\exists \varepsilon_k \Big\{ (\varepsilon_k \in H_v) \& (u \neq v) \& \big( H\_Merge(\varepsilon_i, \varepsilon_j) \cap \varepsilon_k \neq \Phi \big) \Big\}$$

This property is obvious, since if there exists $\varepsilon = H\_Merge(\varepsilon_i, \varepsilon_j)$ which does not intersect with any hyperellipsoid in the other class, then in Step 2.3, this hyperellipsoid will substitute $\varepsilon_i$ and $\varepsilon_j$ as a new hyperellipsoid in $H_k$. Therefore, at the termination of M-HEC Algorithm, property *ii)* persists.

*iii)* This algorithm converges in a finite number of hyperellipsoid merging operations.

Without loss of generality, to simplify the description, we assume that the number of samples in every class is identical, that is, $q_1 = q_2 = \cdots = q_w = q$.

At the very beginning of M-HEC Algorithm, every sample corresponds to a trivial hyperellipsoid. So, at the Step 1, we have $O(wq)$ hyperellipsoids.

In Step 2, the Algorithm tries to merge every pair of hyperellipsoids which are in the same class. There are at most $O(q^2)$ possible combinations of hyperellipsoid pairs for every class. Totally, there are at most $O(wq^2)$ possible hyperellipsoid pairs in the training sample set. Since each time a successful hyperellipsoid merging takes place, the number of hyperellipsoids in that class decreases. After checking all possible $O(wq^2)$ hyperellipsoid pairs for the training sample set, M_HEC Algorithm should stop. At each merging operation, the algorithm needs to check whether the merged hyperellipsoid intersects with the hyperellipsoid in the other class or not. At worst case, this will take $O((w-1)q) = O(wq)$ times to run. Totally, there are at most $O(wq)O(wq^2) = O(wq^3)$ operations of checking hyperellipsoid intersections.

That is, at worst case, there are at most $O(wq^2)$ operations of hyperellipsoid-pair emerging and at most $O(wq^3)$ operations of checking hyperellipsoid intersections in this algorithm. Therefore, the M-HEC Algorithm guarantees converging in a finite number of algorithmic operation steps.

## *3.3 Algorithm Discussions*

(1) In this algorithm, a set of samples, say $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k\}$, uniquely determine a hyperellipsoid by equation

$$(\mathbf{x}-\mu)^t \Sigma^{-1} (\mathbf{x}-\mu) \le C \tag{3.1}$$

where

$$\mu = \frac{1}{k}\sum_{i=1}^{k} \mathbf{x}_i \tag{3.2}$$

and

$$\Sigma = \frac{1}{k}\sum_{i=1}^{k} (\mathbf{x}_i-\mu)(\mathbf{x}_i-\mu)^t = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \dots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_{nn} \end{bmatrix} \tag{3.3}$$

and $\sigma_{uv} = \frac{1}{k}\sum_{i=1}^{k} \left(\mathbf{x}_i^{(u)} - \mu^{(u)}\right)\left(\mathbf{x}_i^{(v)} - \mu^{(v)}\right)^t, \quad u, v = 1, 2, \cdots, n$

C is a constant to determine the scale of the hyperellipsoid.

Therefore, a hyperellipsoid $\varepsilon$ can be denoted as $\varepsilon \sim (\mathrm{x}-\mu)^t \Sigma^{-1}(\mathrm{x}-\mu) \leq C$.

The parameter C should be chosen such that hyperellipsoids properly cover the samples in the set. In our research, we choose C = 1.

(2) Algorithm $H\_Merge(\varepsilon_1, \varepsilon_2)$ is a simple algorithm to merge two hyperellipsoids $\varepsilon_1$ and $\varepsilon_2$ into a hyperellipsoid $\varepsilon$. What this algorithm does is simply calculating the mean $\mu$ and covariance matrix $\Sigma$ for $\varepsilon$ from those training samples which are either in $\varepsilon_1$ or in $\varepsilon_2$.

There are two sub-algorithms which significantly affect the time complexity of M-HEC algorithm. One is the algorithm for finding the closest pair of vectors in an n-dimensional space among the given vector set. The other is the algorithm to determine whether two hyperellipsoids intersect with each other or not.

Jon Louis Bentley [16] developed an efficient algorithm for finding the closest pair of vectors in 1976. This algorithm only takes $O(N \lg^{k-1} N)$ time to run for N vectors in k-space. The basic design clue about this algorithm can also be found in [17].

To determine if two hyperellipsoids $\varepsilon_1$ and $\varepsilon_2$ intersect with each other, it is needed to solve the following co-equation:

$$\begin{cases} (\mathrm{x}-\mu_1)^t \Sigma_1^{-1}(\mathrm{x}-\mu_1) \leq C_1 \\ (\mathrm{x}-\mu_2)^t \Sigma_2^{-1}(\mathrm{x}-\mu_2) \leq C_2 \end{cases} \tag{3.4}$$

Analytic solution for these equations is not necessary in our problem. We only need to know if a solution exist or not. If there exists a solution, then $\varepsilon_1$ and $\varepsilon_2$ intersect with each other, otherwise, they are apart. It is not easy to solve Eq.(3.4) when the dimension of feature space is more than two. This is the main barrier to use M-HEC to cluster training samples in higher dimensional feature spaces.

(3) If the covariance matrix $\Sigma$ is non-singular, then there are n linearly independent *eigenvectors* $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$ such that

$$e_i^t \Sigma e_j = e_i^t\left(\lambda_j e_j\right) = \lambda_j e_i^t e_j = \begin{cases} \lambda_i, & \text{if} \quad i = j \\ 0, & \text{if} \quad i \neq j \end{cases} \tag{3.5}$$

Therefore,

$$E^t \Sigma E = \begin{bmatrix} \leftarrow e_1^t \rightarrow \\ \leftarrow e_2^t \rightarrow \\ \vdots \\ \leftarrow e_n^t \rightarrow \end{bmatrix} \Sigma \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ e_1 & e_2 & \cdots & e_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix} = \Lambda \tag{3.6}$$

where $\lambda_i, i = 1,2,\cdots,n,$ are called *eigenvalues*.

Then, after transforming vector $\mathbf{x}$ into $\mathbf{x}'$ by linear transformation $\mathbf{x}' = E^t\mathbf{x}$ and $\mu$ into $\mu'$ by $\mu' = E^t\mu$ respectively, the hyperellipsoid $\varepsilon \sim (\mathrm{x} - \mu)^t \Sigma^{-1}(\mathrm{x} - \mu) \leq C$ can be represented by the following equation

$$(\mathbf{x}' - \mu')^t \Lambda^{-1}(\mathbf{x}' - \mu') \leq C \tag{3.7}$$

This equation represents the co-ordinate rotation of hyperellipsoid from the original co-ordinate system to the orthogonal co-ordinate system determined by the eigenvectors. By using this equation, it is much easier to compute the Mahalanobis Distance from vector **x** to the mean $\mu$ of the hyperellipsoid $\varepsilon$. The way is to perform the linear transform $\mu' = E^t\mu$ and $\mathbf{x}' = E^t\mathbf{x}$ first, and then compute $r^2 = (\mathbf{x}'-\mu')^t \Lambda^{-1}(\mathbf{x}'-\mu')$. Since $\Lambda$ is a diagonal matrix, its inverse matrix is a diagonal one with $1/\lambda_i$, $(i = 1,2,\cdots,n)$, as its diagonal elements. This will save lot of computation to compute the inverse matrix for $\Sigma$.

## 4. Multiple Hyperellipsoid Classifier

A general pattern classifier can be viewed as a mapping function in terms of a set of discriminant functions $\pi_k(\mathrm{x})$, $k = 1,2,\cdots,w$, which selects the class corresponding to the largest discriminant value. Thus, the classifier is said to assign an observation feature vector **x** to a class $\omega_k$ if $\pi_k(\mathrm{x}) > \pi_j(\mathrm{x}), \textit{for} \quad \textit{all} \quad j \neq k.$

From this point of view, the effect of any decision rule in the classifier is to divide the feature space into $w$ decision regions, $\Re_1, \Re_2, \cdots, \Re_w$. If $\pi_k(\mathrm{x}) > \pi_j(\mathrm{x})$ for all $j \neq k$, then **x** is in $\Re_k$, and the decision rule assigns **x** to class $\omega_k$. The regions are separated by decision boundaries, i.e. surfaces in feature space where ties occur between a pair of discriminant functions.

From Equation(2.7), we know that the optimal discriminant function corresponds to the minimum conditional risk. A Bayes classifier is naturally represented as

$\pi_k(\mathrm{x}) = -R(\delta_k|\mathrm{x}) = -(1 - P(\omega_k|\mathrm{x})) = P(\omega_k|\mathrm{x}) - 1$. To simplify this expression, we can let $\pi_k(\mathrm{x}) = P(\omega_k|\mathrm{x})$, that is, the optimal discriminant function corresponding to the one that maximizes a posterior probability. This form of the decision rule emphasizes the role of the a posteriori probabilities. By using Eq. (2.1), we can express the rule in terms of the conditional and a priori probabilities. Note that term $p(\mathrm{x})$ in Eq.(2.1) is not important for making a decision. It is just a scale factor to assure that $\sum_{\mathrm{k=1}}^{\mathrm{w}} P(\omega_k|\mathrm{x}) = 1$. In the real application problem, we generally assume that the states of nature are equally likely a priori, i.e. $P(\omega_k)$ are identical for all $k = 1,2,\cdots,w$. In this case, the decision is entirely based on $p(\mathrm{x}|\omega_k)$. $p(\mathrm{x}|\omega_k)$ is called the *likelihood function* of $\omega_k$ with respect to **x**. The bigger $p(\mathrm{x}|\omega_k)$ is, the bigger $P(\omega_k|\mathrm{x})$ is. That is, the optimal discriminant function corresponding to the one that *maximizes* the *likelihood function* $p(\mathrm{x}|\omega_k)$. Therefore, to achieve the minimum probability of error, the decision rule in our modeling is : for a given vector **x**, choosing class label $\omega_k$ such that *the likelihood value* $p(\mathrm{x}|\omega_k)$ is the maximum for all $\omega_k$, $k = 1,2,\cdots,w$.

To the given sample set $S_k \subset S$ for class $\omega_k$, $k = 1,2,\cdots,w$, from Eq.(2.8), we have the density probability functions

$$p(\mathrm{x}|\omega_k) = \frac{1}{(2\pi)^{n/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}r^2} \qquad k = 1,2,\ldots,w \qquad (4.1)$$

where $r^2 = (\mathbf{x} - \mu_k)^t \Sigma_k^{-1} (\mathbf{x} - \mu_k)$ is the squared *Mahalanobis Distance* from vector $\mathbf{x}$ to mean vector $\mu_k$, $\mu_k$ and $\Sigma_k$ is the mean vector and covariance matrix corresponding to sample set $S_k$, respectively. It is easy to see that : 1) the smaller $r^2$ is, thus the closer $\mathbf{x}$ to $\mu_k$ is, the bigger the $p(\mathrm{x}|\omega_k)$ has; 2) the small is the $|\Sigma_k|$, the bigger is the $p(\mathrm{x}|\omega_k)$.

## *4.1 Multiple Hyperellipsoid Classification*

By applying the foregoing Multi-hyperellipsoid Clustering algorithm onto the training sample set $S$, we have a set $\{H_1, H_2, \cdots, H_w\}$ as its output, where $H_k$ is a set of clustering hyperellipsoids for class $\omega_k$, $k = 1,2,\cdots,w$. That is, $H_k = \{\varepsilon_{k1}, \varepsilon_{k2}, \cdots, \varepsilon_{kv(k)}\}$, $v(k) = \|H_k\|$, where $\varepsilon_{kj}, j = 1,2,\cdots,v(k)$, is a clustering hyperellipsoid for class $\omega_k$. Therefore, hyperellipsoid $\varepsilon_{kj}$ describes a local class-conditional distribution for class $\omega_k$, denoted as $p(\mathrm{x}|\omega_k, \varepsilon_{kj})$. The total class-conditional distribution of class $\omega_k$ is

$$p(\mathrm{x}|\omega_k) = \frac{p(\mathrm{x}|\omega_k, \varepsilon)}{p(\mathrm{x})} \tag{4.2}$$

where

$$p(\mathrm{x}|\omega_k, \varepsilon) = \sum_{j=1}^{\|H_k\|} (-1)^{j-1} \mathrm{Z}_j(\mathrm{x}|\omega_k, \varepsilon) \tag{4.3}$$

$$\mathrm{Z}_1(\mathrm{x}|\omega_k, \varepsilon) = \sum_{\mathrm{t}=1}^{\|\mathrm{H_k}\|} p(\mathrm{x}|\omega_k, \varepsilon_{\mathrm{kt}}), \tag{4.3.1}$$

$$\mathrm{Z}_2(\mathrm{x}|\omega_k, \varepsilon) = \sum_{1 \le t < u \le \|H_k\|} p(\mathrm{x}|\omega_k, \varepsilon_{\mathrm{kt}}) p(\mathrm{x}|\omega_k, \varepsilon_{\mathrm{ku}}), \tag{4.3.2}$$

$$Z_3(\mathrm{x}|\omega_k,\varepsilon)=\sum_{1\le t<u<v\le\|H_k\|} p(\mathrm{x}|\omega_k,\varepsilon_{\mathrm{kt}})p(\mathrm{x}|\omega_k,\varepsilon_{\mathrm{ku}})p(\mathrm{x}|\omega_k,\varepsilon_{\mathrm{kv}}), \qquad (4.3.3)$$

$$\cdots,\quad Z_{\|H_k\|}(\mathrm{x}|\omega_k,\varepsilon)=\prod_{j=1}^{\|H_k\|} p(\mathrm{x}|\omega_k,\varepsilon_{\mathrm{kj}}). \qquad (4.3.4)$$

and

$$p(\mathrm{x})=\sum_{\mathrm{i}=1}^{w} p(\mathrm{x}|\omega_i,\varepsilon) \qquad (4.4)$$

Thus, those hyperellipsoids in $H_k$ characterize class-conditional distribution for class $\omega_k$.

In Eq. (4.2), $p(\mathrm{x})$ is unimportant for making the classification decision since it is just a scale factor to assure $\sum_{\mathrm{k}=1}^{w} p(\mathrm{x}|\omega_k)=1$. Together with the preceding conclusion, to minimize the overall decision risk in assigning the class membership to the observation feature vector **x**, we should categorize **x** into class $\omega_k$ such that $p(\mathrm{x}|\omega_k)$, i.e. $p(\mathrm{x}|\omega_k,\varepsilon)$, has the *maximum likelihood value* for all $k=1,2,\cdots,w$. Thus, the Decision Rule of the classifier is

1) computing values $p(\mathrm{x}|\omega_k,\varepsilon)$ for all classes according to Eq. (4.3), and

2) categorizing x into class $\omega_k$ which has the maximum $p(\mathrm{x}|\omega_k,\varepsilon)$ value for $k=1,2,\cdots,w$.

## *4.2 Implementation of the Classifier : Parallel-distributed Processing*

To compute the value $p(\mathrm{x}|\omega_k,\varepsilon)$ for class $\omega_k$, it is needed to compute all values

$p\left(\mathrm{x}\middle|\omega_k,\varepsilon_{kj}\right)$, $j=1,2,\cdots,\|H_k\|$. Since $p\left(\mathrm{x}\middle|\omega_k,\varepsilon_{kj}\right)=\frac{1}{(2\pi)^{n/2}\left|\Sigma_{\varepsilon_{kj}}\right|^{1/2}}e^{-\frac{1}{2}\left(\mathrm{x}-\mu_{\varepsilon_{kj}}\right)^t\Sigma_{\varepsilon_{kj}}^{-1}\left(\mathrm{x}-\mu_{\varepsilon_{kj}}\right)}$,

to compute $p\left(\mathrm{x}\middle|\omega_k,\varepsilon_{kj}\right)$, we need to compute the determinant $\left|\Sigma_{\varepsilon_{kj}}\right|$ of covariance matrix $\Sigma_{\varepsilon_{kj}}$ and the squared Mahalanobis Distance $\left(\mathbf{x}-\mu_{\varepsilon_{kj}}\right)^t\Sigma_{\varepsilon_{kj}}^{-1}\left(\mathbf{x}-\mu_{\varepsilon_{kj}}\right)$ from $\mathbf{x}$ to the mean $\mu_{\varepsilon_{kj}}$ of hyperellipsoid $\varepsilon_{kj}$. From Section 3, we knew that hyperellipsoid $\varepsilon_{kj}$ is determined by $\mu_{\varepsilon_{kj}}$ and $\Sigma_{\varepsilon_{kj}}$. After diagonalizing the covariance matrix $\Sigma_{\varepsilon_{kj}}$ by the orthonormal eigenvectors transformation, hyperellipsoid $\varepsilon_{kj}\sim\left(\mathrm{x}-\mu_{\varepsilon_{kj}}\right)^t\Sigma_{\varepsilon_{kj}}^{-1}\left(\mathrm{x}-\mu_{\varepsilon_{kj}}\right)\le C_{\varepsilon_{kj}}$ can be represented as $\varepsilon_{kj}\sim\left(\mathbf{x}'-\mu'_{\varepsilon_{kj}}\right)^t\Lambda_{\varepsilon_{kj}}^{-1}\left(\mathbf{x}'-\mu'_{\varepsilon_{kj}}\right)\le C_{\varepsilon_{kj}}$, where $\mathbf{x}'=E_{\varepsilon_{kj}}^t\mathbf{x}$ is the linear transformation of vector $\mathbf{x}$ by matrix $E_{\varepsilon_{kj}}$ whose columns are the eigenvectors, and $\Lambda_{\varepsilon_{kj}}$ is a diagonalized matrix with eigenvalues as its diagonal elements. Since squared Mahalanobis Distance $r^2=\left(\mathbf{x}'-\mu'_{\varepsilon_{kj}}\right)^t\Lambda_{\varepsilon_{kj}}^{-1}\left(\mathbf{x}'-\mu'_{\varepsilon_{kj}}\right)$ is much easier to compute than to compute $r^2=\left(\mathbf{x}-\mu_{\varepsilon_{kj}}\right)^t\Sigma_{\varepsilon_{kj}}^{-1}\left(\mathbf{x}-\mu_{\varepsilon_{kj}}\right)$, the later one involves an inverse operation of an $n\times n$ matrix. We prefer to do linear transformation $\mathbf{x}'=E_{\varepsilon_{kj}}\mathbf{x}$ and $\mu'=E^t\mu$ first, and then compute the squared Mahalanobis Distance by

$$r^2=\left(\mathbf{x}'-\mu'_{\varepsilon_{kj}}\right)^t\Lambda_{\varepsilon_{kj}}^{-1}\left(\mathbf{x}'-\mu'_{\varepsilon_{kj}}\right)=\sum_{i=1}^{n}\frac{\left(x'-\mu'^{(i)}_{\varepsilon_{kj}}\right)^2}{\lambda_{\varepsilon_{kj}}^{(i)\,2}} \tag{4.5}$$

This computation procedure corresponds to the following functional link neural-network:

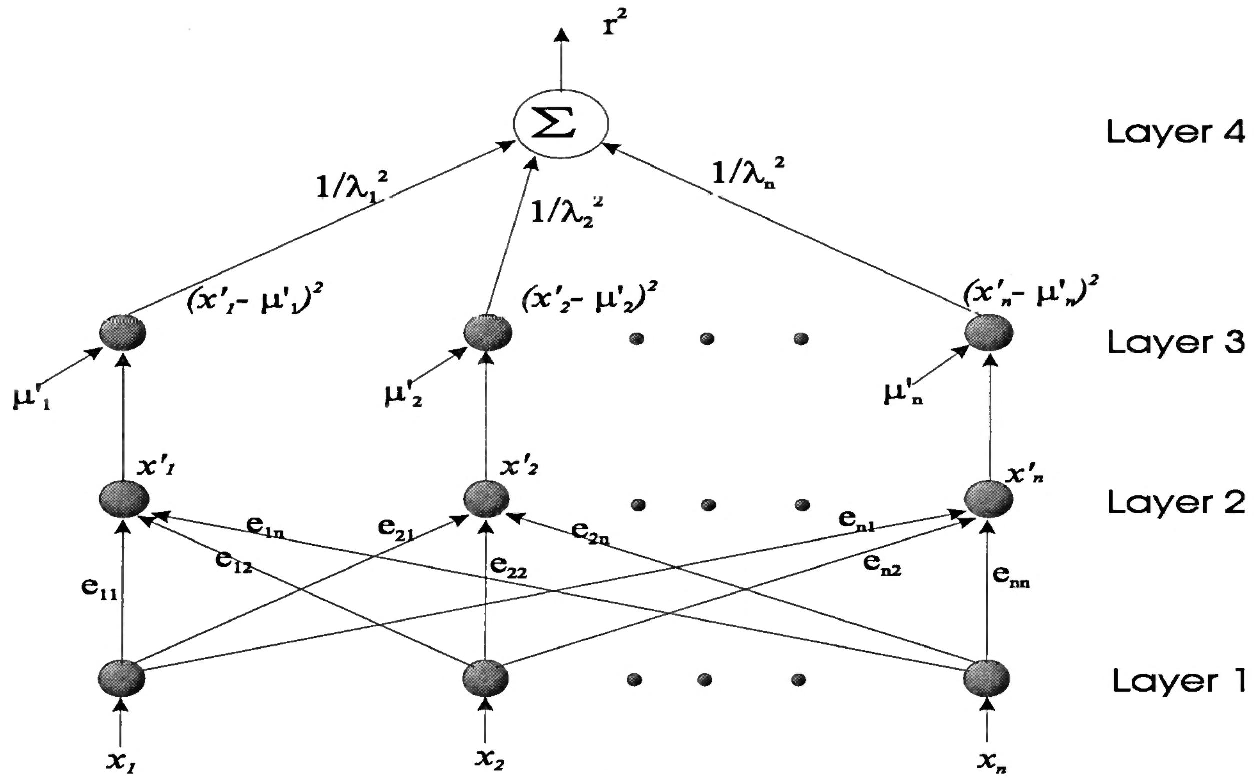
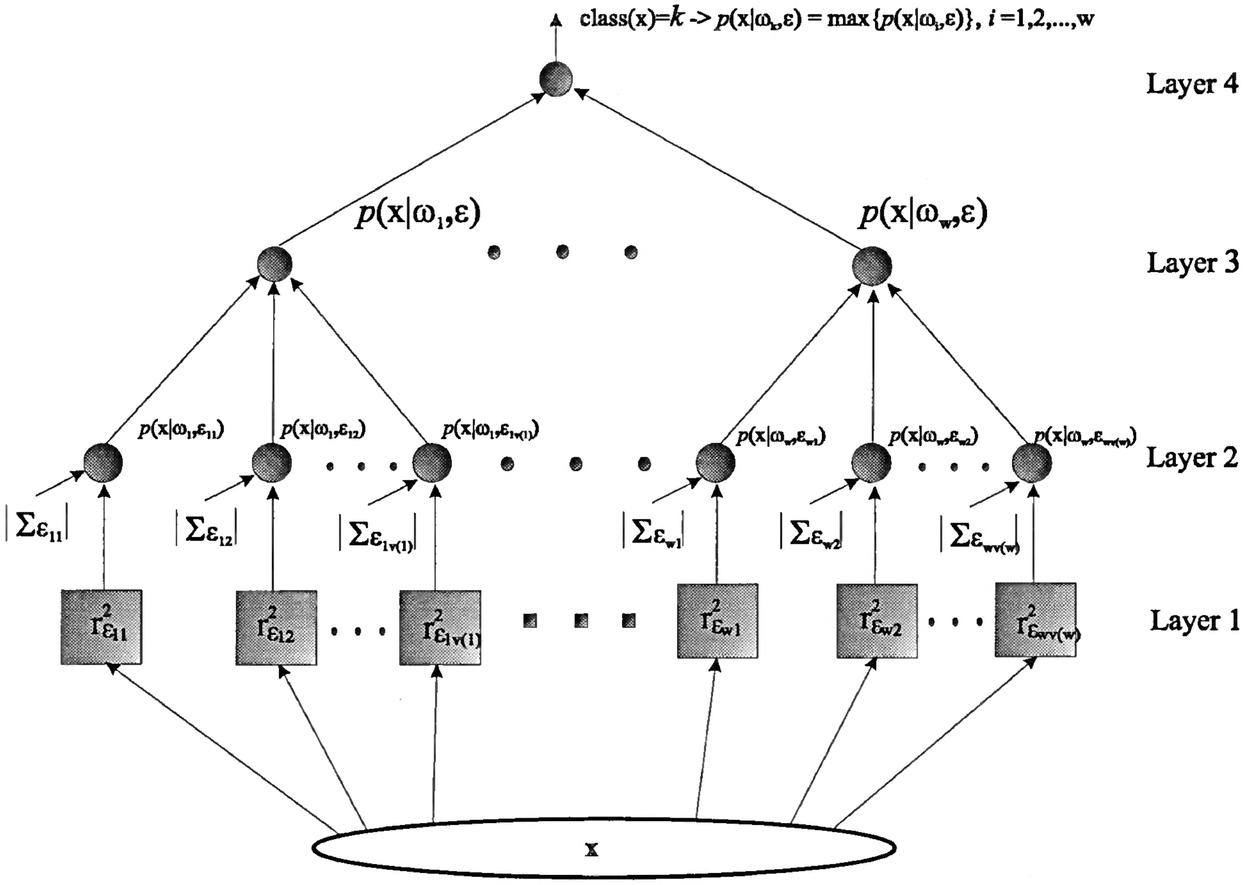


**Figure 4.1 Neural-network to compute the Mahalanobis Distance**

In this network, layer 1 and layer 2 together accomplish the functional computation for the linear transformation $\mathbf{x}' = E_{\varepsilon_{kj}}\mathbf{x}$. At layer 3, $\mu_i'$ is a bias input of the i th neuron which can be calculated by $\mu_i' = E_{kj}^i \mu_i$. The activation function in the i th neuron performs the calculation of $(x_i' - \mu'_i)^2$. Then, with the weight $\frac{1}{\lambda_i^2}$ connecting the i th neuron in layer 3 to the output neuron in layer 4, the activation function of the neuron in layer 4 provides the summation of $r^2 = \sum_{i=1}^{n} \frac{\left(x' - \mu'^{(i)}_{\varepsilon_{kj}}\right)^2}{\lambda^{(i)\,2}_{\varepsilon_{kj}}}$ as its output. Simplifying the network in Figure 4.1 as

$$\begin{tikzcd}
{} \\
\boxed{r^2_{\varepsilon_{kj}}} \arrow[u] \\
\mathbf{x} \arrow[u]
\end{tikzcd}$$

we have the functional-link neural network shown as follows which is equivalent to the classifier we developed in the previous section.



class(x)=$k$ -> $p(\mathbf{x}|\omega_k,\varepsilon) = \max\{p(\mathbf{x}|\omega_i,\varepsilon)\}$, $i$ =1,2,...,w
Layer 4
$p(\mathbf{x}|\omega_1,\varepsilon)$
$p(\mathbf{x}|\omega_w,\varepsilon)$
Layer 3
$p(\mathbf{x}|\omega_1,\varepsilon_{11})$
$p(\mathbf{x}|\omega_1,\varepsilon_{12})$
$p(\mathbf{x}|\omega_1,\varepsilon_{1v(1)})$
$p(\mathbf{x}|\omega_w,\varepsilon_{w1})$
$p(\mathbf{x}|\omega_w,\varepsilon_{w2})$
$p(\mathbf{x}|\omega_w,\varepsilon_{wv(w)})$
Layer 2
$|\Sigma\varepsilon_{11}|$
$|\Sigma\varepsilon_{12}|$
$|\Sigma\varepsilon_{1v(1)}|$
$|\Sigma\varepsilon_{w1}|$
$|\Sigma\varepsilon_{w2}|$
$|\Sigma\varepsilon_{wv(w)}|$
$r^2_{\varepsilon_{11}}$
$r^2_{\varepsilon_{12}}$
$r^2_{\varepsilon_{1v(1)}}$
$r^2_{\varepsilon_{w1}}$
$r^2_{\varepsilon_{w2}}$
$r^2_{\varepsilon_{wv(w)}}$
Layer 1
x

**Figure 4.2 Functional-link Neural Network for M-HEC**

In this classifier network, Layer 1 is a functional-link layer to calculate the Mahalanobis Distances from the given vector **x** to the means of all hyperellipsoids. Then, together with the determinants of covariance matrices of hyperellipsoids, the neurons in Layer 2 use the Eq. (4.1) as their activation functions to calculate the class-conditional probability value $p(\mathrm{x}|\omega_i, \varepsilon_{ij})$ for every hyperellipsoid $\varepsilon_{ij}$ in class $\omega_i$, $i = 1,2,\cdots,w$. At Layer 3, the neurons use Eq.(4.3) as their activation functions to calculate the class-conditional probability value $p(\mathrm{x}|\omega_i, \varepsilon)$ for every class $\omega_i$, $i = 1,2,\cdots,w$. Finally, the neuron in Layer 4 simply returns the class label $k$ which has the maximum value $p(\mathrm{x}|\omega_k, \varepsilon)$ among all $p(\mathrm{x}|\omega_i, \varepsilon), i = 1,2,\cdots w$.

If multi-processing computer system is available, the parallel-distributed processing ability inhered in neural network will make it possible to substantially speed up the classification process for the real world applications.

## *4.3 Simulations*

In this research, we designed a simulation program embedded with both M-HEC classifier and Gaussian classifier. By using MS-Visual C++ and object-oriented programming technique, we implemented the program within an integrated graphical interface in MS-Windows environment. In Figure 4.3 ~ 4.5, we present snapshots of the program running under Windows® 95.

After a user activates the program, a graphical interface is shown up as in Figure 4.3. The user can interactively create a training sample set by using the mouse to choose different pattern styles from the Tool Bar and click on different co-ordinate points in the window.

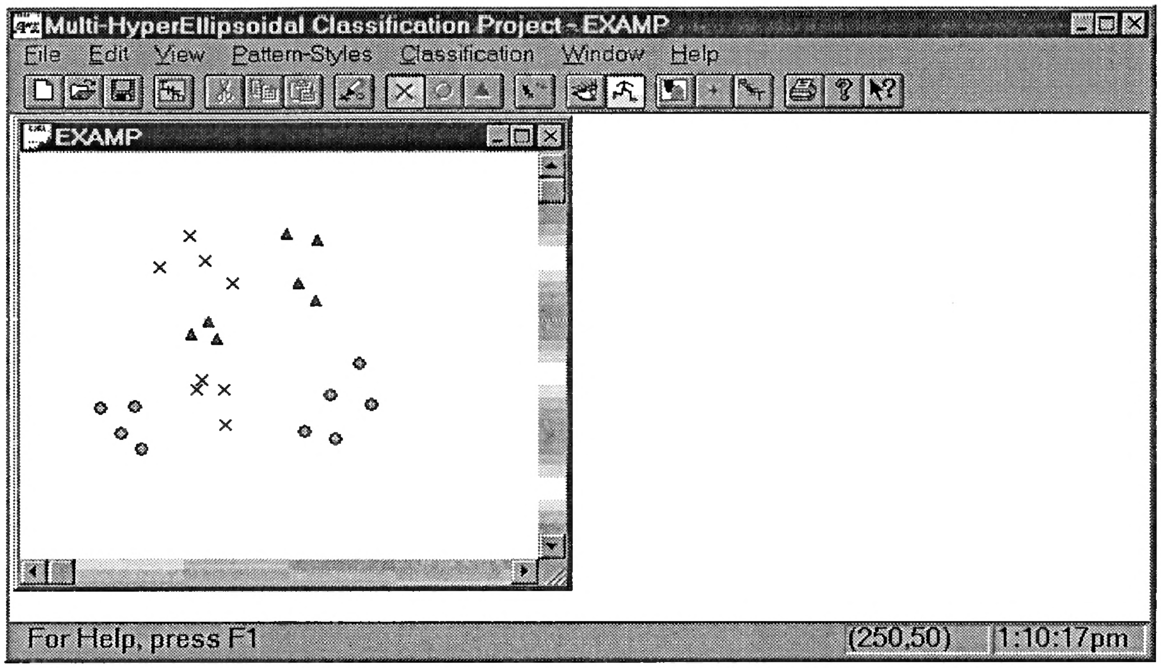

Multi-HyperEllipsoidal Classification Project - EXAMP
File Edit View Pattern-Styles Classification Window Help
EXAMP
For Help, press F1
(250,50)
1:10:17pm

**Figure 4.3 Interface of the Simulation Program**

After creating a training sample set, the user can use either M-HEC clustering or conventional Gaussian parameter estimation to cluster the training sample set. The clustering result is directly shown in the window, as in Figure 4.4.

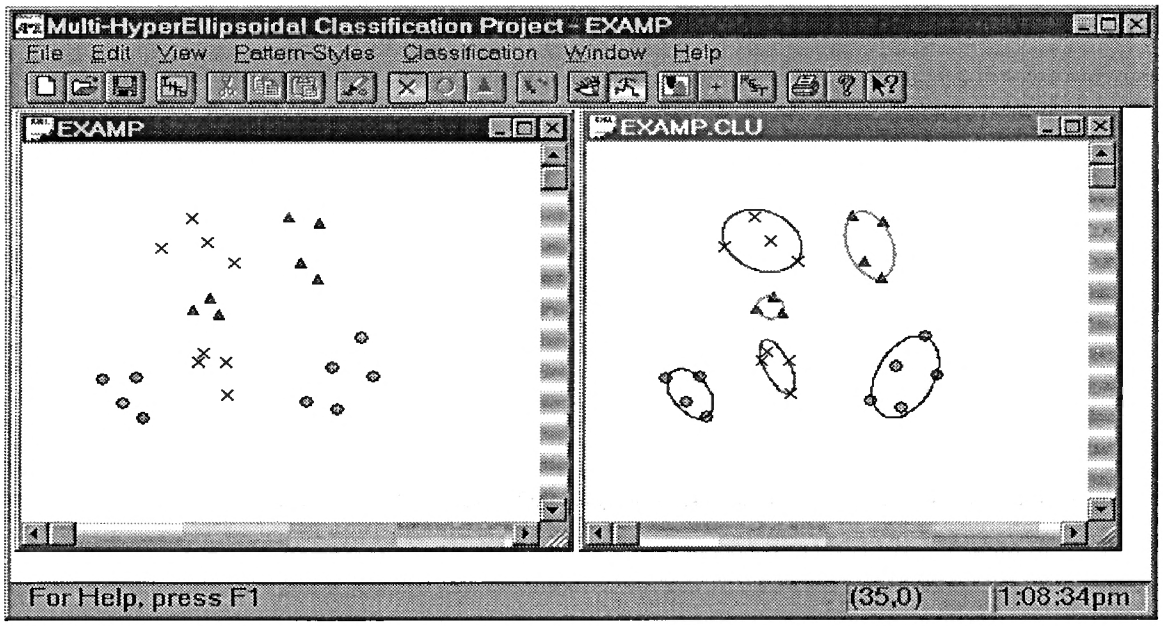

Multi-HyperEllipsoidal Classification Project - EXAMP
File Edit View Pattern-Styles Classification Window Help
EXAMP
EXAMP.CLU
For Help, press F1
(35,0)
1:08:34pm

**Figure 4.4 M-HEC Clustering Result of the Simulation Program**

Once the clustering procedure terminates, the program provides different ways for the user to see the classification result. For example, the program can show the feature space partition result as in Figure 4.5. The program can also provide the information of the rate of correct classification on the training samples. Moreover, if a user randomly picks up a point in the feature space by clicking the mouse, the program can tell what class this point belongs to.

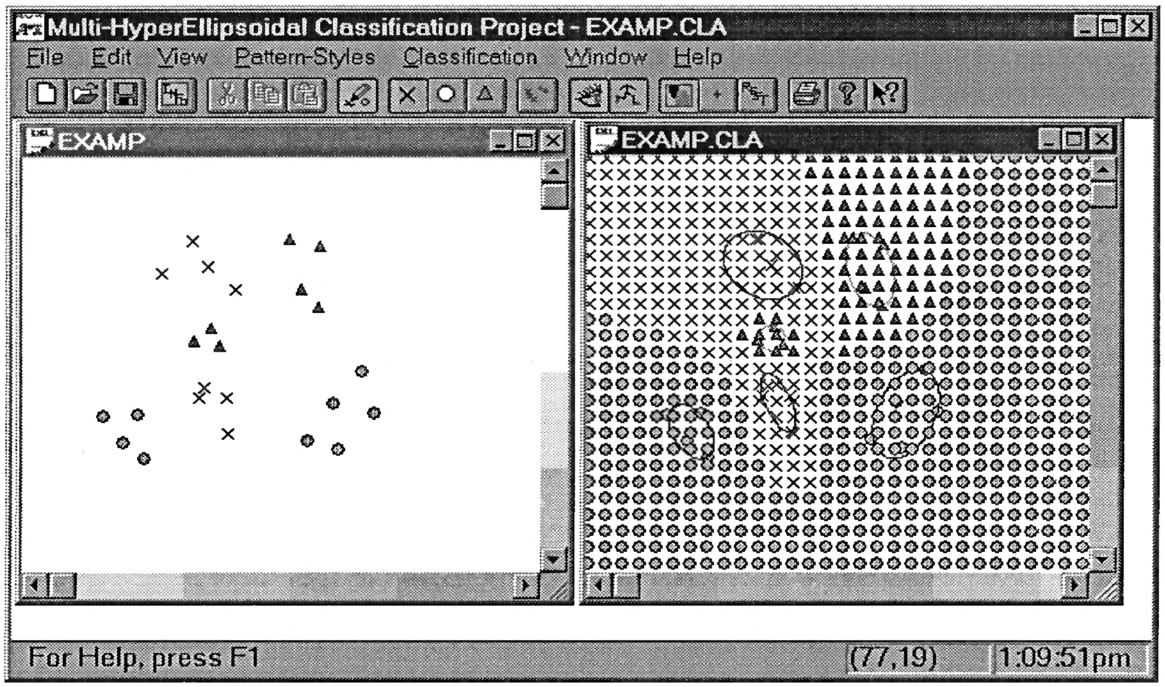

Multi-HyperEllipsoidal Classification Project - EXAMP.CLA
File Edit View Pattern-Styles Classification Window Help
EXAMP
EXAMP.CLA
For Help, press F1
(77,19)
1:09:51pm

**Figure 4.5 Feature Space Partition Result of The Simulation Program**

The simulation results with performance comparison between M-HEC classifier and conventional Gaussian classifier are presented in Section 5.1.

# 5. Comparison and Analysis of Experiment Results

There are many techniques in pattern classification, such as Statistical Modeling using Gaussian Probability Distribution, Neural Network Clustering, and Fuzzy Clustering. We will compare our Multi-HyperEllipsoid Classifier with these classifiers in this section. To simplify the explanation, we denote Multi-HyperEllipsoid Classifier as M-HEC.

## *5.1 Comparison with Gaussian classifier*

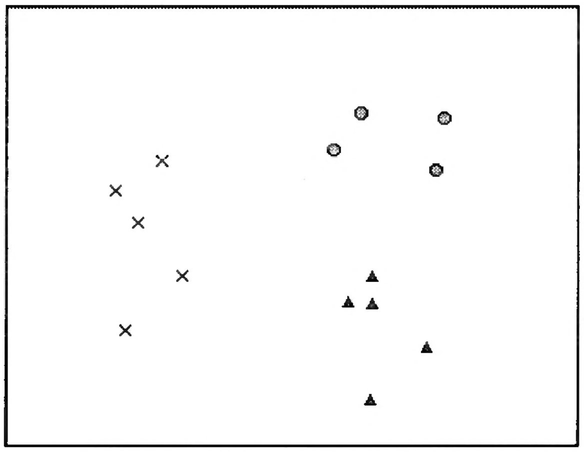
Conventional Gaussian classifier can be treated as a classifier which uses a Single HyperEllipsoid to cluster a given training sample set in a class. Every class will have its own single clustering hyperellipsoid. We denote this classifier as S-HEC.

In this section, we use the simulation program mentioned in section 4.3 to compare the classification performance of M-HEC with that of S-HEC. This program simulates classifications in 2-dimensional feature space.

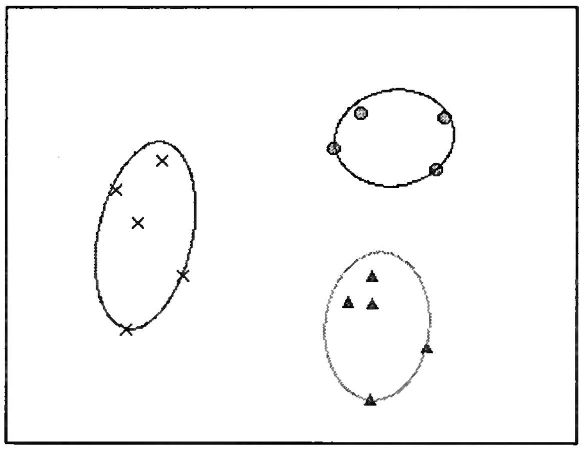
In all examples shown in this section, X stands for a pattern in the first class, O a pattern in the second class, and $\Delta$ a pattern in the third class.

(1) Example 1

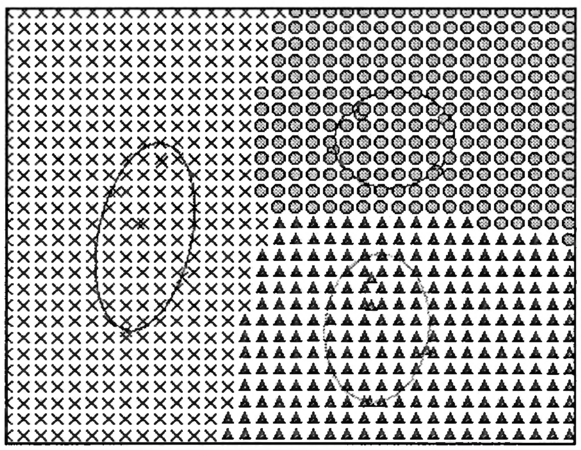
If the training samples can be clustered by hyperellipsoids such that each class has only a single corresponding hyperellipsoid and there is no intersection between any two hyperellipsoids of different classes, then both M-HEC and S-HEC will result in an identical classifier. We have an example shown in Figure 5.1(a),(b),(c) below.



(a) Training Sample Set



(b) Clustering Result by M-HEC and S-HEC



(c) Feature Space Partitioned by M-HEC and S-HEC (an identical partition of feature space is resulted from both classifiers)

**Figure 5.1 Simulation Example 1**

The classification results on the training sample set are shown in Table I below. From Table I, we can see that both M-HEC and S-HEC correctly classify the training samples.

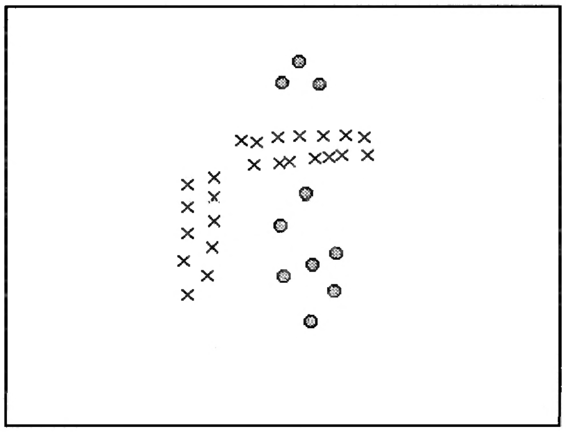
**Table I** Classification Result On The Training Sample Set of Example 1

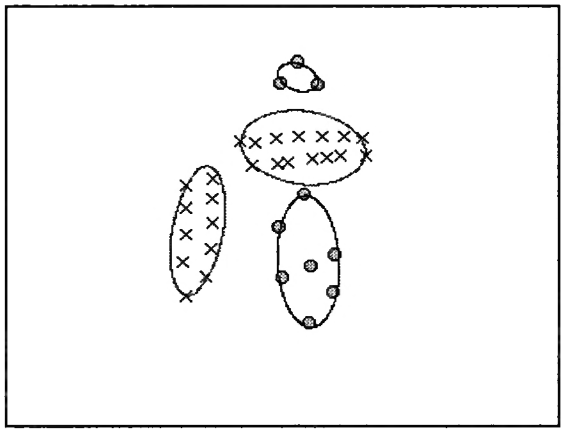
| | | M-HEC | | S-HEC | |
|---|---|---|---|---|---|
| Class | Samples # | Correct Classified # | Correct Ratio % | Correct Classified # | Correct Ratio % |
| 1 | 5 | 5 | 100.00 % | 5 | 100.00 % |
| 2 | 4 | 4 | 100.00 % | 4 | 100.00 % |
| 3 | 5 | 5 | 100.00 % | 5 | 100.00 % |
| Total | 14 | 14 | 100.00 % | 14 | 100.00 % |

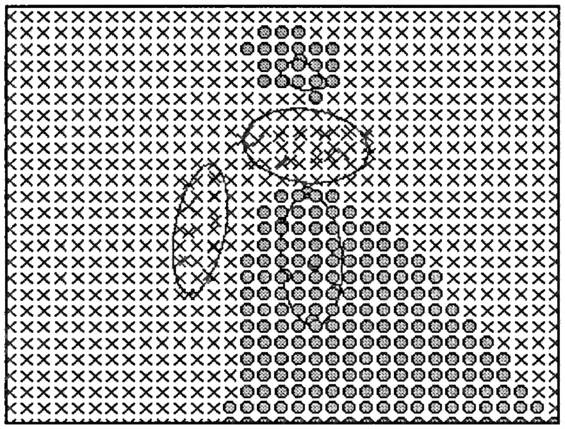
For those situations in which the training samples can not be separately clustered by S-HEC, M-HEC uses more than one hyperellipsoid to cluster each class. For S-HEC, since there are intersections among the clustering hyperellipsoids, those training samples located in the intersection regions will not be correctly classified by S-HEC. But for M-HEC, there is no intersection between any two clustering hyperellipsoids of different classes. The clustering hyperellipsoid dominates the probability calculation for patterns in the subspace covered by itself in the feature space. Therefore, theoretically, M-HEC guarantees 100 percent correct classification on the training sample set. We have examples with results which are consistent with this conclusion shown in Figure 5.2~5.5.

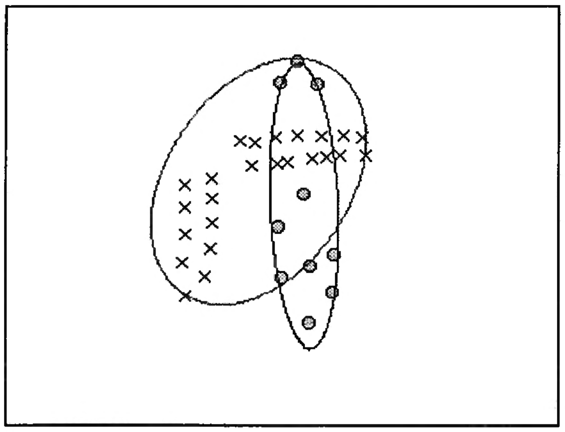
(2) Example 2

Figure 5.2(a) is an example with a training sample set in two classes. The samples in the first class does not look like a Gaussian normal distribution. The samples in the second class are separated by the samples in the first class. The clustering results and feature space partitioning results by M-HEC and S-HEC are presented in Figure 5.2(b) ~ (e), respectively.


(a) Training Sample Set


(b) Clustering Result by M-HEC


(c) Feature Space Partitioned by M-HEC


(d) Clustering Result by S-HEC

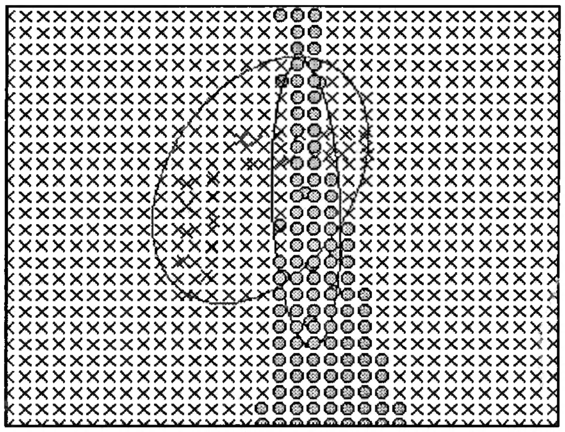

(e) Feature Space Partitioned by S-HEC

**Figure 5.2 Simulation Example 2**

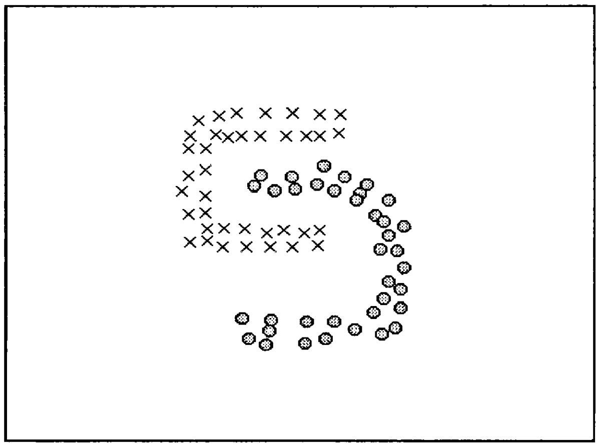
The classification results on the training sample set by M-HEC and S-HEC are in Table II. In this example, M-HEC correctly classifies the training sample set, but S-HEC has 5 mis-classified samples among 24 samples in class 1 and 3 mis-classified samples among 10 samples in class 2.

**Table II** Classification Result On The Training Sample Set of Example 2

| | | M-HEC | | S-HEC | |
|---|---|---|---|---|---|
| Class | Samples # | Correct Classified # | Correct Ratio % | Correct Classified # | Correct Ratio % |
| 1 | 24 | 24 | 100.00 % | 19 | 79.00 % |
| 2 | 10 | 10 | 100.00 % | 7 | 70.00 % |
| Total | 34 | 34 | 100.00 % | 26 | 76.00 % |

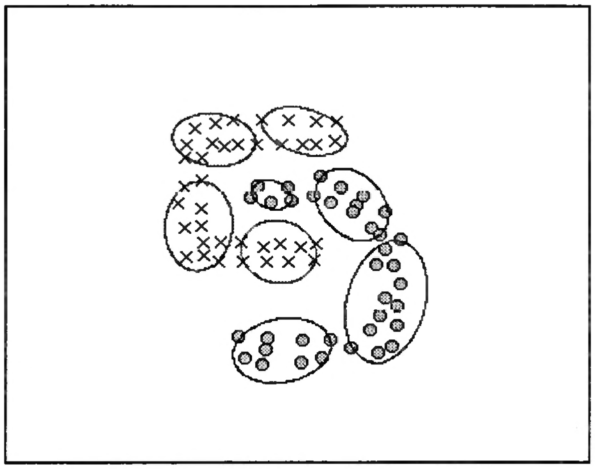
(3) Example 3

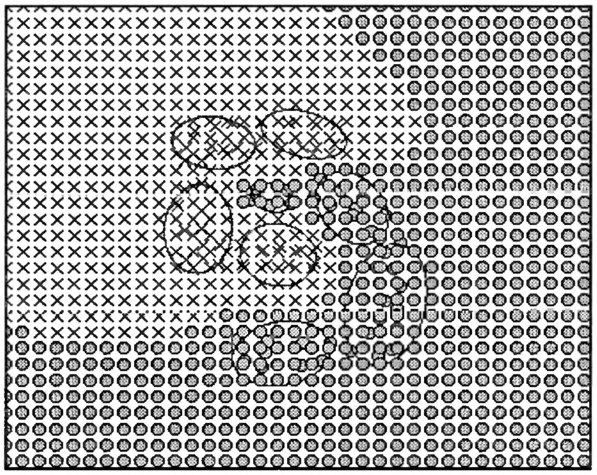
Figure 5.3(a) is an example with training samples in two classes. The samples of one class is partially surrounded by the samples in the other class. The clustering results and feature space partitioning results by M-HEC and S-HEC are presented in Figure 5.3(b) ~ (e).

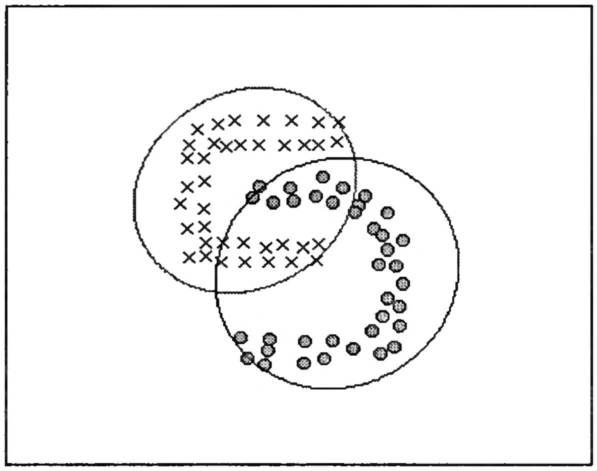


(a) Training Sample Set

**Figure 5.3 Simulation Example 3**


(b) Clustering Result by M-HEC


(c) Feature Space Partitioned by M-HEC


(d) Clustering Result by S-HEC

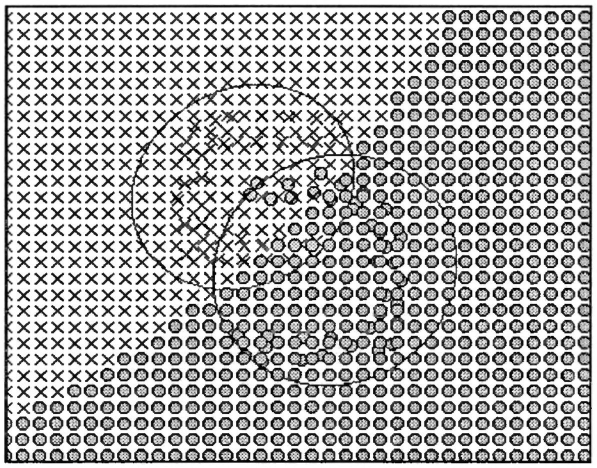

(e) Feature Space Partitioned by S-HEC

**Figure 5.3 Simulation Example 3 (Continued)**

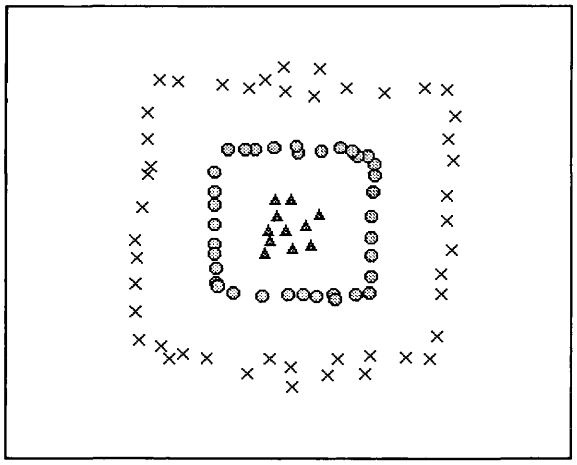
The classification results on the training sample set by M-HEC and S-HEC are in Table III. In this example, M-HEC correctly classifies the training sample set, but S-HEC has 6 and 7 mis-classified samples among 38 samples in class 1 and 37 samples in class 2 respectively.

**Table III** Classification Result On The Training Sample Set of Example 3

| | | M-HEC | | S-HEC | |
|---|---|---|---|---|---|
| Class | Samples # | Correct Classified # | Correct Ratio % | Correct Classified # | Correct Ratio % |
| 1 | 38 | 38 | 100.00 % | 32 | 84.00 % |
| 2 | 37 | 37 | 100.00 % | 30 | 81.00 % |
| Total | 75 | 75 | 100.00 % | 62 | 82.00 % |

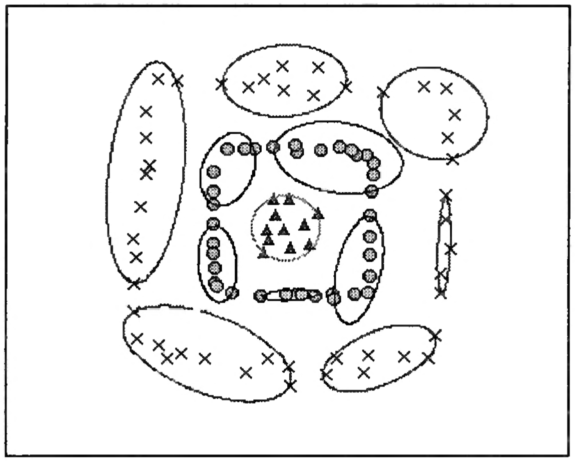
(4) Example 4

Figure 5.4(a) is an example with training samples in three classes. The samples in the first class besiege the samples in the second and third classes. The samples in the second class enclose the samples in the third class. The clustering results and feature space partitioning results by M-HEC and S-HEC are shown in Figure 5.4 (b) ~ (e).

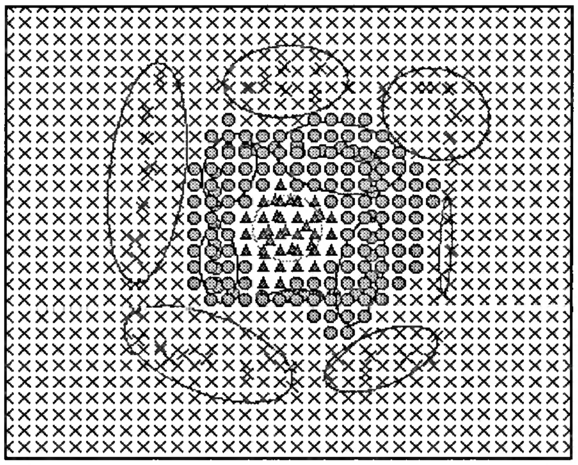


(a) Training Sample Set

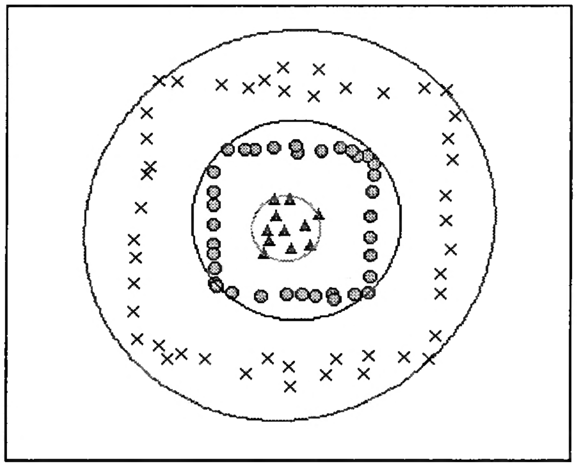
**Figure 5.4 Simulation Example 4**



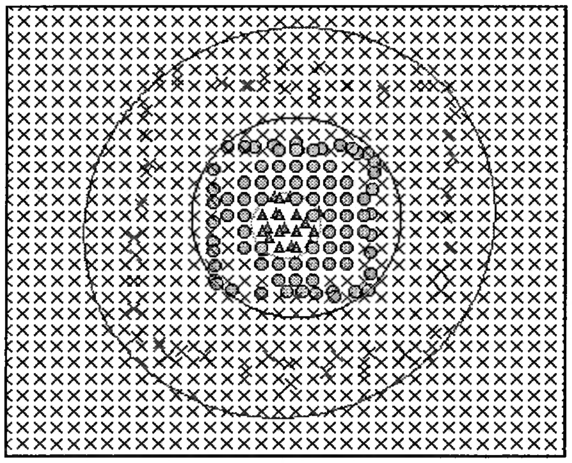
(b) Clustering Result by M-HEC



(c) Feature Space Partitioned by M-HEC



(d) Clustering Result by S-HEC



(e) Feature Space Partitioned by S-HEC

**Figure 5.4 Simulation Example 4 (Continued)**

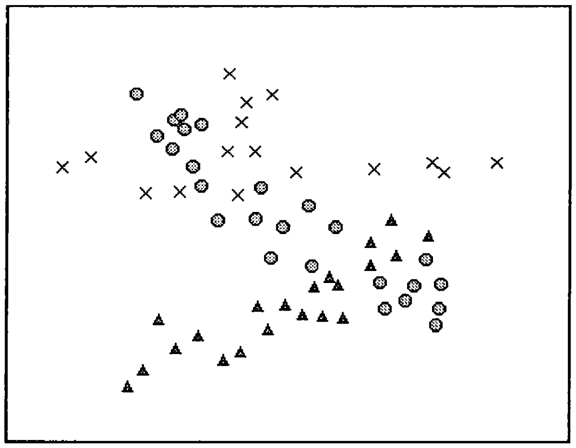
The classification results on the training sample set by M-HEC and S-HEC are in Table IV. In this example, M-HEC correctly classifies the training sample set. But S-HEC can not classify samples in class 2 and mis-classifies 4 samples among 6 samples in class 3.

**Table IV** Classification Result On The Training Sample Set of Example 4

| | | M-HEC | | S-HEC | |
|---|---|---|---|---|---|
| Class | Samples # | Correct Classified # | Correct Ratio % | Correct Classified # | Correct Ratio % |
| 1 | 35 | 35 | 100.00 % | 35 | 100.00 % |
| 2 | 18 | 18 | 100.00 % | 0 | 0.00 % |
| 3 | 6 | 6 | 100.00 % | 2 | 33.00 % |
| Total | 59 | 59 | 100.00 % | 37 | 62.00 % |

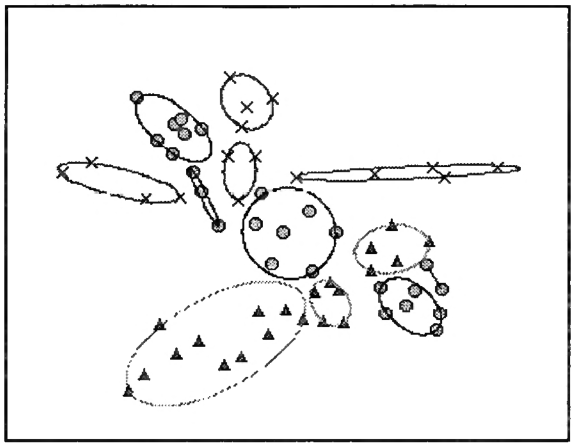
(5) Example 5

Figure 5.5 (a) is an example with training samples in three classes. In this example, the training samples of different classes mix up together. The clustering results and feature space partitioning results by M-HEC and S-HEC are presented in Figure 5.5(b) ~ (e).

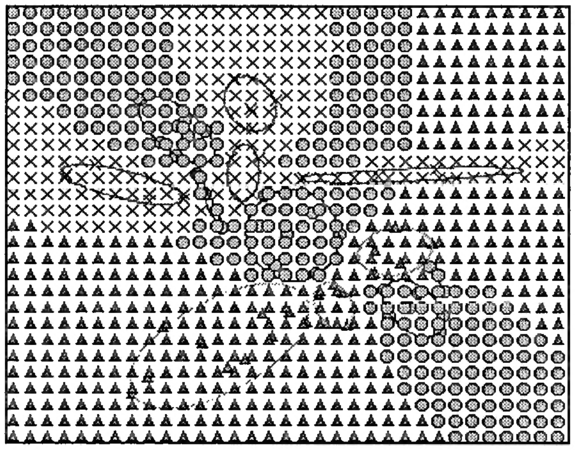


(a) Training Sample Set

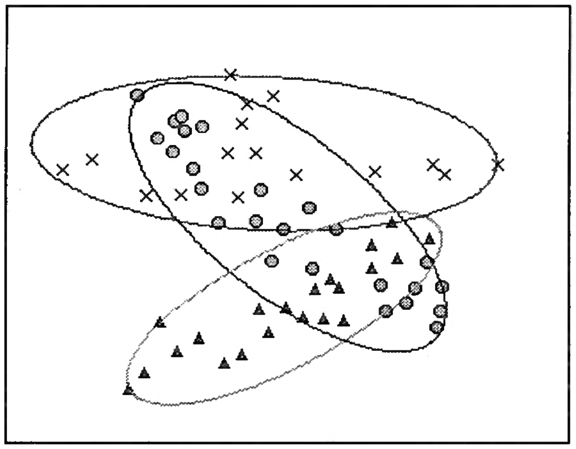
**Figure 5.5 Simulation Example 5**



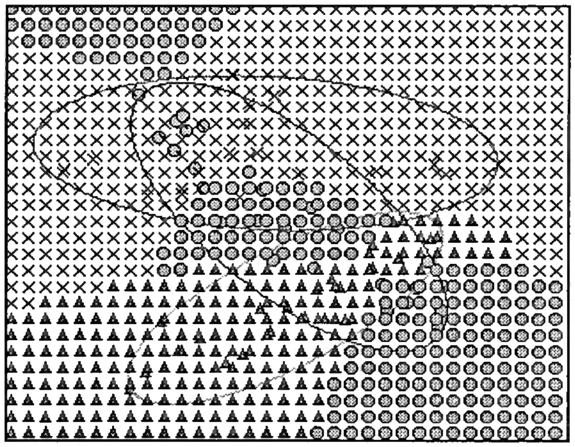
(b) Clustering Result by M-HEC



(c) Feature Space Partitioned by M-HEC



(d) Clustering Result by S-HEC



(e) Feature Space Partitioned by S-HEC

**Figure 5.5 Simulation Example 5 (continued)**

The classification results on the training sample set by M-HEC and S-HEC are in Table V. In this example, M-HEC correctly classifies the training sample set. But S-HEC has one mis-classified sample for both class 1 and class 3, and 11 mis-classified samples for class 2.

**Table V** Classification Result On The Training Sample Set of Example 5

| | | M-HEC | | S-HEC | |
|---|---|---|---|---|---|
| Class | Samples # | Correct Classified # | Correct Ratio % | Correct Classified # | Correct Ratio % |
| 1 | 16 | 16 | 100.00 % | 15 | 93.00 % |
| 2 | 25 | 25 | 100.00 % | 14 | 56.00 % |
| 3 | 21 | 21 | 100.00 % | 20 | 95.00 % |
| Total | 62 | 62 | 100.00 % | 49 | 79.00 % |

From experimental examples shown above, we can see that when the training samples of different classes can be separately clustered, S-HEC and M-HEC results in a equivalent classifier. But when the training samples of different classes are mixed together, M-HEC is more precise than S-HEC in partitioning the feature space into classes. That is, M-HEC covers and improves the functionality of S-HEC.

## *5.2 Comparison with Neural Network Based Classifier*

Neural network is a network with weighted connections on neurons in some specific architectures. Generally, neural networks have a training mechanism to help the network learn how to map a given input set into its corresponding output set, even though there is no explicitly mathematical relation between them. This makes it possible to use neural network as a classifier in pattern classification problems.

There are a lot of neural network models on which the training procedures are developed, such as Backpropagation Network (BPN) and Counterpropagation Network (CPN). After a neural network is created, a learning procedure should be applied to adjust the weights of neuron connections to map input into desired output. Different neural network architectures have different learning mechanisms and the architectures significantly affect the behaviors of the network. We call these kinds of networks as conventional neural networks. More detailed materials of neural network techniques can be found from text book [15]. More detailed studies about using neural network computing for pattern classification are available in H. Hey [9], H. G. C. Traven [8], Y. H. Pao [33], R. Schalkoff [27] and S.T. Bow[30].

Different from conventional neural networks, as we shown in previous section, a neural network classifier can be automatically generated by M-HEC. The architecture and the weights of the network are completely determined by the clustering result of the M-HEC. There are lots of advantages of M-HEC neural network classifier over conventional neural networks. They show in some respects as follows :

*1) Convergence*

Before a conventional neural network classifier can be used to do classification, it should be trained and converged on the training sample set. A converged neural network should have a very small total-error for mapping the training samples' inputs to their corresponding outputs. The convergence of a neural network depends on a number of factors, such as the architecture of the network, the training parameters in the training procedure, the distribution property of training samples, etc.. In conventional neural networks, there is no consistent criterion to determine these factors to guarantee that the training procedure results in a converged neural network.

By using the clustering result of M-HEC, i.e. the clustering hyperellipsoids, the architecture and the weights of the M-HEC neural network can be easily determined as shown in Section 4.2. From discussions in the above sections, M-HEC theoretically has the result with the minimum classification error on the training samples. That is, M-HEC guarantees that the neural network classifier constructed by the clustering hyperellipsoids is a converged neural network.

*2) Architecture of the Neural Network*

As discussed in the previous section, conventional neural network does not have a systematic criterion to determine a proper network architecture and weights which result in a converged neural network classifier in applying to the training samples. Since different neural network architecture comes with different functional properties, the non-deterministic property prevents the classifier from convergence. The non-converged neural network is useless in doing classification. This is the main reason why the conventional neural network hardly gets its wide application in the real classification problems.

For the M-HEC classifier, the neural network architecture is pre-determined by the clustering hyperellipsoids. Since the training sample set generally is a finite set, there should exist a finite number of clustering hyperellipsoids for any given training sample set. As discussed in preceding section, M-HEC guarantees resulting in a converged classifier.

*3) Training Time*

To get a converged neural network classifier, conventional neural network technique needs to take many iterations to train the network. The training time depends on many factors, such as the network architecture, the training parameters, the probability distribution of the training samples, etc. The conventional neural network techniques do not provide a systematic criterion to determine the proper training parameters which result in a converged network. Furthermore, as discussed in previous section, the functional property of a neural network is determined by its architecture. The conventional neural network technique does not provide a consistent way to get a proper architecture for different kinds of training samples. It is possible to get a wrong neural network architecture which will never converges, no matter how long in training. Therefore, the training time to get a converged classifier is non-deterministic. This non-deterministic property also prevents the conventional neural network techniques from applying in the real applications.

For the M-HEC, the neural network architecture is completely determined by the clustered hyperellipsoids. After the multiple hyperellipsoid clustering procedure, there is no time needed to train the neural network into a converged classifier. The time of M-HEC's clustering procedure is totally determined by the training samples. Since the training sample set generally is a finite set, the clustering time will be deterministic and finite. Therefore, it is predictable to get a converged classifier in a finite and deterministic training time.

To summarize, the non-deterministic property in architecture and training time of conventional neural networks prevents them from being widely applied in the real world applications. The pre-deterministic property of M-HEC make it possible to be used in many classification problems.

## *5.3 Comparison with Fuzzy Clustering*

Generally, fuzzy clustering approach is a way for finding a decision mechanism by using fuzzy set theory to establish a set of membership functions. Many algorithms for fuzzy clustering depend on initial guesses of cluster prototypes. The classification precision of fuzzy clustering depends on the priori assumption of the membership functions and the setting of the parameters. This kind of study can be found in [10] and [11]. There is no systematic way and consistent procedure to direct the selection of membership functions, due to the complexity and diversity of the pattern distribution in the real classification problems. Theoretically, it is impossible to get an idea form of membership functions and parameters to fit all the complex distribution situations occurring in training samples. Therefore, Y. Nakamori and M.Ryoke [32] proposed a fuzzy clustering model which allows people to evaluate the initial clustering result and interactively modify the parameters according to their intuition to get a satisfactory classification result.

By using M-HEC, we do not need to define the form of membership functions and select the parameters for the classifier. Also, we do not need to interact with the classifier to tune up its parameters to get a satisfactory classification result, since M-HEC provides a consistent procedure to establish the classifier from the given training samples. Theoretically, M-HEC will result in an idea classification for any given training sample set, which is difficult, if not impossible, by using the fuzzy clustering modeling.

## 6. Conclusions

Pattern classification is a process of finding a decision mechanism to partition the feature space into separated subspaces, such that each subspace corresponds to a specific class of patterns. There are many techniques for pattern classification, such as traditional modal-based Gaussian classifier, neural network classifier, and fuzzy clustering classifier. Among criteria to determine the performance of the classifier, the correct classification rate on the training sample set is the most important and significant one to be used in real application.

Traditional model-based Gaussian classifier relies on the statistic analysis, especially the pattern probability distribution, of the training samples. If the training samples of different classes are distinguishable by a set of single Gaussian distribution functions, then it will result in an idea classifier to partition the feature space in which the training sample set are completely separate. Otherwise, it is difficult to find a decision mechanism to precisely distinguish those training samples in the intersection regions. This results in a poorly classification rate on the training sample set.

Traditional neural-network approach can be used in classification problems with very complex mapping relationship between input and desired output. It can also be used in handling the classification with very complex distribution of training samples. But before the neural network can be applied to do classification, it needs a non-predetermined network architecture and non-deterministic training time to get a converged classifier. This make it difficult to be used in the real world applications.

Fuzzy clustering approach requires a priori assumption on the form of membership functions and parameters to set up the decision mechanism. Usually, fuzzy clustering requires a large amount of time to search the parameters for the membership functions. Some of the fuzzy clustering approaches even require people interactively using their intuition to tune up the parameters to get satisfactory classifications. The performance of fuzzy clustering classifier depends on the selection of uncertain membership functions and non-predetermined parameters.

In this research, we developed a pattern classification model based on multiple hyperellipsoid clustering. The decision rule in this classifier is based on the class-conditional probability distributions of classes. The difference between this classifier and the conventional model-based classifier is that the class-conditional probability distribution of every class is estimated by the combination of a set of local Gaussian probability distributions, instead of a single probability function. This overcomes the problems involved in the conventional model-based classifier in cases : (1) the parametric form of the probability distribution is unknown; (2) the training samples of different classes intersect with each other.

The main contributions of this research are :

1) using multiple hyperellipsoid modeling to cluster the training samples which overcomes the mis-classifying problem caused by the conventional model-based Gaussian classifier. Multi-hyperellipsoid clustering also provides a very precise classification result for any probable distribution of training samples.

2) combining the supervised learning procedure with clustering. Generally, clustering procedure works together with unsupervised learning parameter estimation. In this research, multi-hyperellipsoid clustering is a procedure based on supervised maximum likelihood estimation.

3) using multiple local Gaussian-probability-based distributions to estimate the whole probability distribution for the patterns in the feature space. Traditional model-based supervised classifier uses only one probability distribution function per class to estimate the pattern probability. In this research, a combining probability computation scheme which is based on the multiple local Gaussian-probability distributions is developed. This overcomes the problem caused by the conventional model-based classifier when the distribution form of training samples is unknown.

4) using neural network technique to implement the classifier. The potential parallel-distributed processing ability of neural network will substantially speed up the strenuous computation inhered in the clarification procedure.

5) being able to classify complex distributions with high accuracy.

6) guaranteeing convergence of the algorithms. Therefore, the behavior of the algorithm is predictable.

To improve M-HEC's running efficiency, further work on finding an efficient algorithm to solve co-equation (3.4) in more than 2 dimensional space is needed.

## Bibliography

[1] A. J. Scott and M. J. Symons, Clustering Methods Based on Likelihood Ratio Criteria, *Biometrics* **27**, 1971, pp.387~397

[2] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Englewood Cliffs, New Jessey, Prentice-Hall 1988.

[3] B. H. Juang and S. Katagiri, Discriminative Learning for Minimum Error Classification, *IEEE Trans. on Signal Processing*, Vol. 40, Dec. 1992, pp.3043~3054

[4] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press, 1991

[5] Charles W. Therrien, *Decision Estimation and Classification: An Introduction to Pattern Recognition and Related Topics*, New York: John Wiley & Sons, Inc.1989.

[6] D. E. Gustafson and W. C. Kessel, Fuzzy Clustering With a Fuzzy Covariance Matrix, *Proc. IEEE CDC*, San Diego, CA, January 1979, pp. 761~766

[7] D. Lowe and A. R. Webb, Optimized Feature Extraction and the Bayes Decision in Feed-Forward Classifier Networks, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.13, April 1991, pp. 355~364

[8] Hans G.C. Traven, A Neural Network Approach to Statistical Pattern Classification by "Semiparameteric" Estimation of Probability Density Functions, *IEEE Transactions on Neural Networks*, Vol.2, No.3, May 1991, pp. 366~377

[9] Hermann Hey, On the Probablistic Interpretation of Neural Network Classifiers and Discriminative Training Criteria, *IEEE Trans. On Pattern Analysis and Machine Intelligence*, Vol. 17, No. 2, Feb. 1995, pp. 107~119

[10] Hisao Ishibuchi, Ken Nozaki and Hideo Tanaka, Efficient Fuzzy Partition of Pattern Space for Classification Problems, *Fuzzy Sets and Systems* 59 (1993), pp. 295~304.

[11] I. Gath and A. B. Geva, Unsupervised Optimal Fuzzy Clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, July 1989, pp. 773~781

[12] J. Dunn, A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-separated Clusters, *J. Cybernetics*, Vol. 3, 1974, pp. 32~57

[13] Jeffrey D. Banfield and Adrian E. Raftery, Model-Based Gaussian and Non-Gaussian Clustering, *BIOMETRICS* 49, September 1993, pp. 803~821

[14] Jambu, M. and Lebeaux, M. O. , *Cluster Analysis and Data Analysis*, North-Holland, Amsterdam, 1983.

[15] James A. Freeman and David M. Skapura, *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison-Wesley Publishing Company 1991.

[16] Jon Louis Bentley, Divide and Conquer Algorithms for Closest Point Problems in Multidimensional Space, *Unpublished Ph.D. dissertation, Univ. of North Carolina*, Chapel Hill, N.C., 1976.

[17] Jon Louis Bentley, Multidimensional Divide-and-Conquer, *Communications of the ACM*, Vol. 23, No.4, April 1980, pp. 214~229

[18] Keinosuke Fukunaga, Statistical Pattern Recognition, *Handbook of Pattern Recognition and Computer Vision*, Eds. C.H. Chen, L.F. Pau and P. S. P. Wang, World Scientific Publishing Company 1993. pp. 33~60

[19] Kristin P. Bennett and O. L. Mangasarian, Robust Linear Programming Discrimination of Two Linearly Inseparable Sets, *Optimization Methods and Software*, 1992, Vol. 1, pp. 23~34

[20] M. Grotshel and Y. Wakabayashi, A Cutting Plane Algorithm for a Clustering Problem, *Mathematical Programming*, vol.45 Aug.1, 1989. pp.59

[21] O. L. Mangasarian, Multisurface Method of Pattern Separation, *IEEE Transactions on Information Theory* IT-14(6), 1968, pp. 801~807

[22] O. L. Mangasarian, R. Setiono, and W. H. Wolberg, Pattern Recognition via Linear Programming: Theory and Application to Medical Diagnosis, in: *Large-Scale Numerical Optimization*, Thomas F. Coleman and Yuying Li, (Eds.), SIAM, Philadelphia 1990, pp.22 ~30

[23] Panel on Discriminant Analysis, Classification, and Clustering, Discriminant Analysis and Clustering, *Statistical Science*, Vol. 4, No.1, 1989, pp. 34~69

[24] Richard C. Dubes, Cluster Analysis and Related Issues, *Handbook of Pattern Recognition and Computer Vision*, Eds. C.H. Chen, L.F. Pau and P. S. P. Wang, World Scientific Publishing Company 1993. pp. 3~32

[25] R. K. Blashfield, M. S. Aldenderfer, and L. C. Morey, Cluster Analysis Software, *Handbook of Statistics*, Vol.2, P. R. Krishnah and L. N. Kanal (eds.), North Holland, Amsterdam, The Netherlands, 1982, pp. 245~266.

[26] R.L. Cannon, J. V. Dave, and J. C. Bezdek, Efficient Implementation of the Fuzzy C-means Clustering Algorithms, *IEEE Trans. Pattern Anal, Machine Intell.*, Vol, PAMI-8, No. 2, 1986, pp. 248~255

[27] Robert Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches*, John Wiley & Sons, Inc. 1992

[28] R.O. Duda and P.E. Hart, *Pattern Recognition and Scene Analysis*, New York: John Wiley & Sons, Inc. 1973.

[29] Roy L. Streit and Tod E. Luginbuhl, Maximum Likelihood Training of Probabilistic Neural Network, *IEEE Transactions on Neural Networks*, Vol.5, No.5, September 1994, pp. 764~783

[30] Sing-Tze Bow, *Pattern Recognition and Image Preprocessing*, Marcel Dekker, Inc. 1992.

[31] Vito Di Gesu, Integrated Fuzzy Clustering, *Fuzzy Sets and Systems* 68 (1994) pp. 293~308.

[32] Y. Nakamori and M. Ryoke, Identification of Fuzzy Prediction Models Through Hyperellipsoidal Clustering, *IEEE Transactions On Systems, Man, And Cybernetics,* Vol. 24, No. 8, August 1994, pp. 1153~1173

[33] Yoh-Han Pao, Neural Net Computing for Pattern Recognition, *Handbook of Pattern Recognition and Computer Vision*, Eds. C. H. Chen, L. F. Pau and P. S. P. Wang, World Scientific Publishing Company 1993. pp. 125~162