FEATURE EXTRACTION AND CLASSIFICATION ALGORITHMS FOR HIGH DIMENSIONAL DATA

Chulhee Lee
David Landgrebe

TR-EE 93-1
January, 1993

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907-1285

This work was sponsored in part by NASA under Grant NAGW-925

Table of Contents

ABSTRACT

In this research, feature extraction and classification algorithms for high dimensional data are investigated. Developments with regard to sensors for Earth observation are moving in the direction of providing much higher dimensional multispectral imagery than is now possible.

In analyzing such high dimensional data, processing time becomes an important factor. With large increases in dimensionality and the number of classes, processing time will increase significantly. To address this problem, a multistage classification scheme is proposed which reduces the processing time substantially by eliminating unlikely classes from further consideration at each stage. Several truncation criteria are developed and the relationship between thresholds and the error caused by the truncation is investigated.

Next a novel approach to feature extraction for classification is proposed based directly on the decision boundaries. It is shown that all the features needed for classification can be extracted from decision boundaries. A novel characteristic of the proposed method arises by noting that only a portion of the decision boundary is effective in discriminating between classes, and the concept of the effective decision boundary is introduced. The proposed feature extraction algorithm has several desirable properties: (1) it predicts the minimum number of features necessary to achieve the same classification accuracy as in the original space for a given pattern recognition problem (2) it finds the necessary feature vectors. The proposed algorithm does not deteriorate under the circumstances of equal means or equal covariances as some previous algorithms do. In addition, the decision boundary feature extraction algorithm can be used both for parametric and non-parametric classifiers.

Finally, we study some problems encountered in analyzing high dimensional data and propose possible solutions. We first recognize the increased importance of the second order statistics in analyzing high dimensional data. By investigating the characteristics of high dimensional data, we suggest the reason why the second order statistics must be taken into account in high dimensional data. Recognizing the importance of the second order statistics, there is a need to represent the second order statistics. We propose a method to visualize statistics using a color code. By representing statistics using color coding, one can easily extract and compare the first and the second statistics.

CHAPTER 1 INTRODUCTION

1.1 Background

Advances in sensor technology for Earth observation make it possible to collect multispectral data in much higher dimensionality. For example, the HIRIS instrument now under development for the Earth Observing System (EOS) will generate image data in 192 spectral bands simultaneously. In addition, multi-source data also will provide high dimensional data. Such high dimensional data will have several impacts on processing technology: (1) it will be possible to classify more classes; (2) more processing power will be needed to process such high dimensional data; and (3) feature extraction methods which utilize such high dimensional data will be needed.

In this research, three main subjects are studied: fast likelihood classification, feature extraction, and the characteristics of high dimensional data and problems in analyzing high dimensional data.

The analysis of remotely sensed data is usually done by machine oriented pattern recognition techniques. One of the most widely used pattern recognition techniques is classification based on maximum likelihood (ML) assuming Gaussian distributions of classes. A problem of Gaussian ML classification is long processing time. This computational cost may become an important problem if the remotely sensed data of a large area is to be analyzed or if the processing hardware is more modest in its capabilities. The advent of the future sensors will aggravate this problem. As a result, it will be an important problem to extract detailed information from high dimensional data while reducing processing time considerably.

Feature extraction has long been an important topic in pattern recognition and has been studied by many authors. Linear feature extraction can be viewed as finding a set of vectors which effectively represent the information content of an observation while reducing the dimensionality. In pattern recognition, it is desirable to extract features which are focused on discriminating between classes. Although numerous feature extraction/selection algorithms have been proposed and successfully applied, it is also true that there are some circumstances where the previous methods do not work well. In particular, if there is little difference in mean vectors or little difference in covariance matrices, some of the previous feature extraction methods fail to find a good feature set.

Although many feature extraction algorithms for parametric classifiers are proposed, relatively few feature extraction algorithms are available for non-parametric classifiers. Furthermore, few feature extraction algorithms are available which utilize the characteristics of a given non-parametric classifier. As use of non-parametric classifiers such as neural networks to solve complex problems increases, there is a great need for an effective feature extraction algorithm for non-parametric classifiers.

In dealing with high dimensional data, there will be problems which have not been encountered in analyzing relatively low dimensional data. In order to realize the full potential of high dimensional data, it is necessary to understand the characteristics of high dimensional data. One of these characteristics is the increased importance of the second order statistics. Although some classifiers, e.g., as a minimum distance classifier utilizing only first order statistics, often perform relatively well on low dimensional data, it is observed that classifiers utilizing only first order statistics show limited performance in high dimensional space. Further, information contained in the second order statistics plays an important role in discriminating between classes in high dimensional data. We will illustrate this problem and investigate the reasons for it by examining the characteristics of such high dimensional data.

More detailed background and related works on each of these subjects will be discussed at the beginning of each chapter.

1.2 Objective of Research

It is the objective of this research to better understand the characteristics of high dimensional data relative to the analysis process, and to create algorithms which increase the feasibility of its use.

In order to utilize the discriminating power of high dimensional data without increasing processing time significantly, a fast likelihood classification algorithm based on a multistage scheme is proposed. At each stage, unlikely classes are eliminated from further consideration, thus reducing the number of classes for which likelihood values are to be calculated at the next stage. Several truncation criteria are developed and the relationship between such truncation and the error increased is investigated.

Another objective of this research is to develop a feature extraction algorithm which better utilizes the potential of high dimensional data. The proposed feature extraction algorithm is based directly on the decision boundary. By directly extracting feature vectors from the decision boundary without assuming any underlying density function, the proposed algorithm can be used for both parametric and non-parametric classifiers. The proposed algorithm also predicts the minimum number of features needed to achieve the same classification accuracy as in the original space for a given problem and finds all the needed feature vectors. In addition, the proposed algorithm does not deteriorate under the circumstances of equal means or equal covariances as some previous algorithms do.

It is a further objective of this research to investigate and understand the characteristics of high dimensional data. Problems in applying to high dimensional data some analysis techniques which were primarily developed for relatively low dimensional data are studied. In particular, the increased role of second order statistics in analyzing high dimensional data is examined. Although most analysis and classification of data are conducted by machine, sometimes it is helpful and necessary for human to interpret and analyze data. However, as the dimensionality grows, it becomes increasingly difficult for human extraction of information from numerical values. In order to overcome this problem, a visualization method is proposed using a color coding scheme. In this method, the correlation matrix of a class is displayed using a color code along with the mean vector and the standard deviation. Each color represents a degree of correlation.

1.3 Research Organization

In Chapter 2, the fast likelihood classification algorithm is presented for high dimensional data. A method to avoid redundant calculations in multi-stage classification is proposed. Several truncation criteria are developed and the relationship between truncation and truncation error is investigated. Experimental results are presented and compared. In Chapter 3, after reviewing various feature extraction algorithms, the decision boundary feature extraction algorithm is developed. After several new concepts are defined, all the needed equations are derived. A decision boundary feature extraction procedure for parametric classifiers is proposed and experimental results are presented. In Chapter 4, the decision boundary feature extraction algorithm is extended to non-parametric classifiers. In Chapter 5, the decision boundary feature extraction algorithm is applied to a neural network. In Chapter 6, discriminant feature extraction, which is a generalization of the decision boundary feature extraction, is presented. In Chapter 7, problems encountered in analyzing high dimensional data are studied and the characteristics of high dimensional data are investigated. In Chapter 8, conclusions are summarized and suggestions for future work are presented. Proofs of theorems, color pictures, and programs are presented in appendices.

CHAPTER 2 FAST LIKELIHOOD CLASSIFICATION

2.1 Introduction

Earth observing systems such as the LANDSAT MSS and Thematic Mapper have played a significant role in understanding and analyzing the Earth resources by providing remotely sensed data of the Earth surface on a regular basis. The analysis of remotely sensed data is usually done by machine oriented pattern recognition techniques. One of the most widely used pattern recognition techniques is classification based on maximum likelihood (ML) assuming Gaussianly distributions of classes. A problem of ML Gaussian classification is long processing time. This computational cost may become an important problem if the remotely sensed data of a large area is to be analyzed or if the processing hardware is more modest in its capabilities. The advent of future sensors, for example HIRIS (High Resolution Imaging Spectrometer) (Goetz and Herring 1989), which is projected to collect data in many more spectral bands will aggravate this problem.

In this chapter, we propose a multistage classification procedure which reduces the processing time substantially while maintaining essentially the same accuracy. The proposed multistage classification procedure is composed of several stages, and at each stage likelihood values of classes are calculated using a fraction of the total features. This fraction increases as stages proceed. Classes which are determined to be unlikely candidates by comparing likelihood values with a threshold are truncated, i.e., eliminated from further consideration so that the number of classes for which likelihood values are to be calculated at the following stages is reduced. Depending on the number of features and the number of classes, the processing time can be reduced by the factor of 3 to 7.

2.2 Related Works and Background

Processing time has been an influential factor in designing classifiers for the analysis of remotely sensed data. Even with the data from the previous sensors such as MSS and TM, for which the numbers of spectral bands are 4 and 7, respectively, the cost of the analysis of even a moderate area was considerable. Future sensors such as HIRIS which will collect data in 192 spectral bands at 30 m spatial resolution, will aggravate this problem.

Efforts to reduce processing time have been pursued in various ways. By employing feature selection/extraction algorithms [(Muasher and Landgrebe,1983), (Duchene and Leclercq, 1988), (Riccia and Shapiro, 1983), (Eppler 1976) and (Merembeck and Turner, 1980)], the number of features can be reduced substantially without sacrificing significant information. Feature selection/extraction is generally done by removing redundant features or by finding new features in transformed coordinates. This reduction in the number of features has several advantages. First of all, higher accuracies can be achieved in cases where the number of training samples is low, due to the Hughes phenomenon (Hughes 1968). Since generally processing time increases with the square of the number of features, a benefit of feature selection/extraction is reduction in processing time.

Another possible approach to reduce computing time can be found in decision tree classifiers [(Swain and Hauska 1977), (Chang and Pavlidis 1977), and (Wang and Suen 1987)]. Though the decision tree classifier can have several advantages depending on the situation, one of the advantages is processing time. For instance, in an ideal binary decision tree classifier, the computing time will be proportional to ln(M) instead of M where M is the number of classes, assuming the same number of features is used at each node. However, how to find the optimum tree structure still remains a problem for the decision tree classifier, though many algorithms are proposed for the design of decision tree classifiers (Argentiero et al., 1982).

Feiveson (1983) proposed a procedure to reduce computing time by employing thresholding. In his algorithm, the most likely candidate class of a given observation is selected based on some prediction, and its probability density function is calculated. If the probability density function is greater than a threshold, calculation of the probability density functions for the other classes is omitted, resulting in reduction of computing time. If it is possible to make a good prediction, the computing time can be reduced significantly. But a problem of this method is that its performance depends on the accuracy of predictions, especially when many classes are involved.

Wald's sequential probability ratio test (SPRT) provides another perspective (Wald, 1947). In Wald's sequential probability ratio test, the sequential probability ratio

$$\lambda_n = \frac{p_n(X|\omega_1)}{p_n(X|\omega_2)}$$

is computed where $p_n(X|\omega_i)$ is the conditional probability density function of X for class ω_i and n denotes the number of features. Then the likelihood ratio, λ_n, is compared with two stopping boundary A and B. If

$\lambda_n \geq A$, then it is decided that $X \sim \omega_1$
$\lambda_n \leq B$, then it is decided that $X \sim \omega_2$

If $B < \lambda_n < A$, an additional feature will be taken and likelihood ratio will be compared with the additional feature. The error probabilities are related to the two stopping boundaries by the following expressions.

$$A = \frac{1 - e_{21}}{e_{12}} \qquad \text{and} \qquad B = \frac{e_{21}}{1 - e_{12}}$$

where e_{ij} is the probability of deciding $X \sim \omega_i$ when $X \sim \omega_j$ is true.

A sequential probability ratio test was applied to pattern classification by Fu [(Fu 1962) and (Chien and Fu 1966)]. When the cost of a feature measure is high or features are sequential in nature, the sequential classification proved useful. Although the sequential classifier achieves the desired accuracy with the minimum number of features, the processing time of the sequential classifier may not be reduced proportionally due to the repeated calculation of the probability density function.

Generally it is true that the processing time of a classifier increases as the number of features increase. In the Gaussian ML classifier, for instance, the processing time is proportional to the square number of features. Therefore it is possible to reduce the processing time considerably by exploiting the property of the SPRT that the decision is reached with the lowest possible number of features if the redundant computations caused by repeated calculation of likelihood values can be avoided. There is, however, another problem in the straightforward application of SPRT to pattern recognition where there are more than two classes. The general relationship between stopping boundaries and the optimum property of SPRT remains to be understood if there are more than two classes.

The SPRT does not take into account the separability of two classes. If the separability of classes is taken into account, a decision may be reached sooner. Considering two cases of a two-class classification problem (Figure 2.1), it is observed that errors in case I are smaller than errors in case II even though the same stopping boundaries are used for both classes. Thus for the same error tolerances, the stopping boundaries for case I can be less strict than case II.

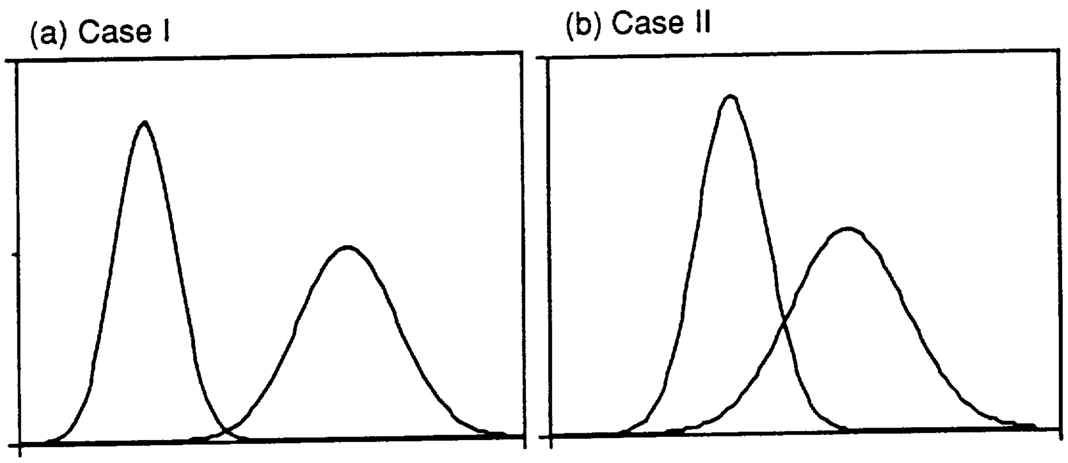

Figure 2.1 A hypothetical example where classes are more separable in case I than those in case II.

In this chapter, an algorithm is proposed which avoids the redundant calculations of SPRT so that the characteristics of SPRT which classifies with the lowest possible number of features can be exploited in such a way that the decision can be made with considerably less processing time [(Lee and Landgrebe 1990), (Lee and Landgrebe 1991-3)]. It is noted that the proposed multistage classifier is different from the Wald's sequential classifier in that the number of stages of the multistage classifier is considerably smaller than the number of features, while the sequential classifier has essentially the same number of stages as the number of features. Also the criteria for truncation are different. We also address the case where there are more than two classes. Though some error is inevitably introduced by truncation, the error is minimal and can be constrained within any specified range. Most of the samples which cause error are found to be outliers. The relationship between truncation and error caused by the truncation is investigated.

2.3 Multistage Gaussian Maximum Likelihood Classification

In the conventional Gaussian ML classifier, a discriminant function is calculated for all classes using the whole feature set and the class which has the largest value is chosen as the classification result of an observation **X**.

$$\mathbf{X} \in \omega_i \text{ if } g_i(\mathbf{X}) > g_j(\mathbf{X}) \text{ for all } j \neq i$$

where g_i is the discriminant function is given by

$$g_i(\mathbf{X}) = -\ln|\Sigma_i| - (\mathbf{X}-\mathbf{M}_i)^t\Sigma_i^{-1}(\mathbf{X}-\mathbf{M}_i) \tag{2.1}$$

where Σ_i is the covariance matrix of class ω_i and $\mathbf{M}_i$ is the mean vector of class ω_i. The discriminant function is essentially the log likelihood value.

In the proposed multistage classifier, at each intermediate stage only a portion of the features is used to calculate the discriminant function, and the classes whose discriminant function values are less than a threshold are truncated. At the final stage the whole feature set is used. The block diagram of the multistage classifier is shown in Figure 2.2.

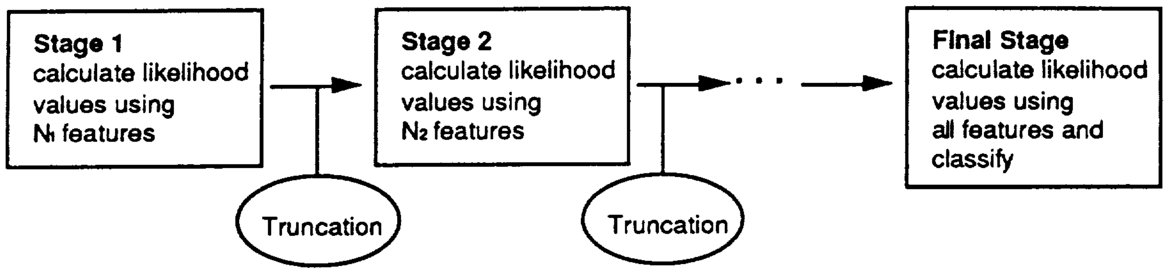
Stage 1
calculate likelihood values using N1 features
Stage 2
calculate likelihood values using N2 features
Final Stage
calculate likelihood values using all features and classify
Truncation
Truncation

Figure 2.2 Multistage classifier.

By truncating unlikely classes at early stages where only a small portion of the whole feature set is used, it is possible to reduce the number of classes for the later stages where more features are to be used. Therefore if it is possible to truncate a substantial number of classes at early stages, the processing time can be reduced substantially. However, there are several problems to be addressed. Since the discriminant function must be calculated repeatedly at each stage, additional calculations are inevitably introduced. Thus truncation alone does not guarantee less processing time. Another problem is to develop criteria for truncation. The successful application of the multistage classifier in reducing processing time depends on how accurately and early a class can be truncated with little risk of introducing truncation error.

2.3.1 Additional Calculations in a Multistage Classifier

Suppose an N-stage classifier where n features are used at the nth stage and N is the total number of features. A possibility to avoid unnecessary calculations is to use the discriminant function values from the current stage in calculating the discriminant function values for the next stage. The most time consuming part in calculating the discriminant function (equation 2.1) is matrix multiplication. Therefore to avoid the additional calculation, one would like to be able to use

$$(X_n - M_n)^t \Sigma_n^{-1} (X_n - M_n)$$

in calculating

$$(X_{n+1} - M_{n+1})^t \Sigma_{n+1}^{-1} (X_{n+1} - M_{n+1})$$

where the subscript denotes the number of features, and **X**, **M** are as in equation (2.1).

This can not be done easily since $\Sigma_{n+1}^{-1} \neq \begin{bmatrix} \Sigma_n^{-1} & \mathbf{p} \\ \mathbf{p}^t & a \end{bmatrix}$ where Σ_n is the covariance matrix of n features, **p** is a column vector, and Σ_{n+1} is the covariance matrix of (n+1) features even though $\Sigma_{n+1}= \begin{bmatrix} \Sigma_n & \mathbf{u} \\ \mathbf{u}^t & a \end{bmatrix}$ where **u** is a column vector. But it can be shown that if Σ_n is invertible and $(a-\mathbf{u}^t\Sigma^{-1}\mathbf{u})$ is not zero, then $\Sigma_{n+1}= \begin{bmatrix} \Sigma_n & \mathbf{u} \\ \mathbf{u}^t & a \end{bmatrix}$ is also invertible and

$$\Sigma_{n+1}^{-1}= \begin{bmatrix} \Sigma_n^{-1}+\alpha\Sigma_n^{-1}\mathbf{u}\mathbf{u}^t\Sigma_n^{-1} & -\alpha\Sigma_n^{-1}\mathbf{u} \\ -\alpha\mathbf{u}^t\Sigma_n^{-1} & \alpha \end{bmatrix} \text{ where } \alpha = \frac{1}{a-\mathbf{u}^t\Sigma^{-1}\mathbf{u}}$$

Without loss of generality, we can assume that the mean vector is zero. Then

$$(\mathbf{X}_{n+1}- \mathbf{M}_{n+1})^t \Sigma_{n+1}^{-1} (\mathbf{X}_{n+1}- \mathbf{M}_{n+1})$$

$$= [\mathbf{X}_n^t \text{ , } x_{n+1}] \begin{bmatrix} \Sigma_n^{-1}+\alpha\Sigma_n^{-1}\mathbf{u}\mathbf{u}^t\Sigma_n^{-1} & -\alpha\Sigma_n^{-1}\mathbf{u} \\ -\alpha\mathbf{u}^t\Sigma_n^{-1} & \alpha \end{bmatrix} \begin{bmatrix} \mathbf{X}^n \\ x_{n+1} \end{bmatrix}$$

$$= \mathbf{X}_n^t(\Sigma_n^{-1}+\alpha\Sigma_n^{-1}\mathbf{u}\mathbf{u}^t\Sigma_n^{-1})\mathbf{X}_n - 2\alpha x_{n+1}\mathbf{X}_n^t\Sigma_n^{-1}\mathbf{u} + \alpha x_{n+1}^2$$

$$= \mathbf{X}_n^t\Sigma_n^{-1}\mathbf{X}_n + \alpha(\mathbf{u}^t\Sigma_n^{-1}\mathbf{X}_n)(\mathbf{u}^t\Sigma_n^{-1}\mathbf{X}_n) - 2\alpha x_{n+1}\mathbf{u}\Sigma_n^{-1}\mathbf{X}_n^t + \alpha x_{n+1}^2$$

$$= \mathbf{X}_n^t\Sigma_n^{-1}\mathbf{X}_n + \alpha[(\mathbf{u}^t\Sigma_n^{-1}\mathbf{X}_n)\{(\mathbf{u}^t\Sigma_n^{-1}\mathbf{X}_n) - 2x_{n+1}\} + x_{n+1}^2] \qquad (2.2)$$

Considering equation (2.2), the value of $\mathbf{X}_n^t\Sigma_n^{-1}\mathbf{X}_n$ is known from the current stage and $\mathbf{u}^t\Sigma_n^{-1}$can be calculated once at the beginning and saved. Therefore the number of multiplications and additions required to calculate $(\mathbf{X}_{n+1}- \mathbf{M}_{n+1})^t$ Σ_{n+1}^{-1} $(\mathbf{X}_{n+1}- \mathbf{M}_{n+1})$ is (n+3) and (n+4), respectively. Thus the total number of multiplications and additions for a class which passes all the truncation tests and reaches the final stage are given by equation (2.3.1) and equation (2.3.2), respectively.

$$\sum_{n=1}^{N-1}(n+3) = \frac{1}{2}N^2+\frac{5}{2}N-3 \propto \frac{1}{2}N^2 \tag{2.3.1}$$

$$\sum_{n=1}^{N-1}(n+4) = \frac{1}{2}N^2+\frac{7}{2}N-3 \propto \frac{1}{2}N^2 \tag{2.3.2}$$

On the other hand, the total number of multiplications and additions of the conventional single stage Gaussian ML classifier are given by equation (2.4.1) and equation (2.4.2), respectively

$$\frac{N(N+1)}{2} + N = \frac{1}{2}N^2+ \frac{3}{2}N \propto \frac{1}{2}N^2 \tag{2.4.1}$$

$$\frac{N(N+1)}{2} = \frac{1}{2}N^2+\frac{1}{2}N \propto \frac{1}{2}N^2 \tag{2.4.2}$$

Comparing equations (2.3.1-2) and equations (2.4.1-2), it can be seen that the total number of multiplications and additions of both methods are about of the same order and the multistage classifier does not introduce significant additional calculations.

2.3.2 Truncation by Absolute Region

Since unlikely classes are to be truncated at each stage in the multistage classifier, a criterion for truncation must be developed. Along with the criteria for truncations, the relationship between truncation and error caused by truncation must also be understood and quantized.

One possible way for truncation is to find the smallest region, Ω_i, for class ω_i which contains a certain portion, P_t, of class ω_i, and to check whether a test sample is within that region. If the data classes are assumed to have Gaussian distributions, the smallest region for a class will be a hyperellipsoid which has its origin at the mean of the class and whose semi-axes are in the directions of the eigenvectors of the covariance of the class with lengths proportional to the corresponding eigenvalues. If a test sample is found outside region Ω_i, class ω_i can be truncated with the risk of error $1-P_t$. For example in Figure 2.3, class 1 can be truncated as an unlikely class with risk of error 0.001.

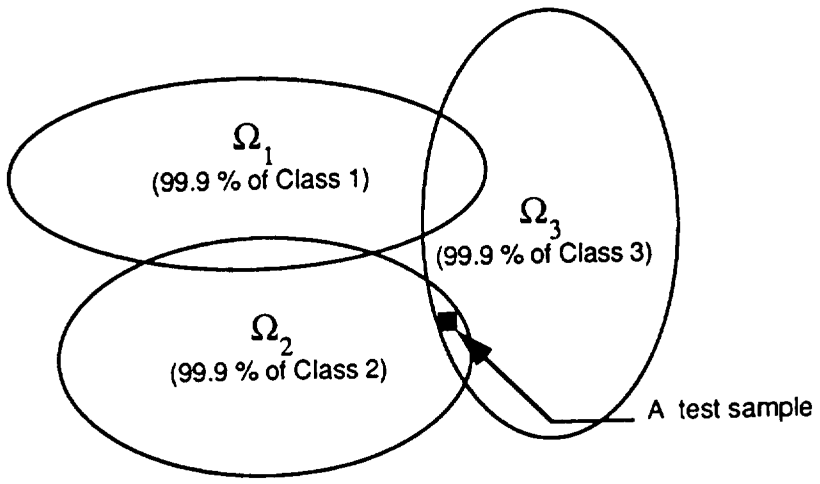

Figure 2.3 A hypothetical distribution of 3 classes.

Finding the smallest region Ω_i for class ω_i is equivalent to finding r_0^2 such that

$$Pr\{\mathbf{X} |\ (\mathbf{X} - \mathbf{M}_i)^t \Sigma_i^{-1} (\mathbf{X} - \mathbf{M}_i) \le r_0^2\} = P_t$$

where $\mathbf{M}_i$ and Σ_i are the mean vector and the covariance matrix of class ω_i.

The smallest region Ω_i is given by

$$\Omega_i = \{\mathbf{X} \mid (\mathbf{X} - \mathbf{M}_i)^t \Sigma_i^{-1} (\mathbf{X} - \mathbf{M}_i) \le r_0^2 \} \qquad (2.5)$$

The quantity $(\mathbf{X} - \mathbf{M}_i)^t \Sigma_i^{-1} (\mathbf{X} - \mathbf{M}_i)$ is the so-called Mahalanobis distance. It is noteworthy that r_0^2 does not depend on $\mathbf{M}$ and Σ but depends solely on n, the dimensionality.

$$Pr_n\{\mathbf{X} |\ (\mathbf{X} - \mathbf{M})^t \Sigma^{-1} (\mathbf{X} - \mathbf{M}) \le r_0^2\}$$

$$= \int\limits_{(X-M)^t\Sigma^{-1}(X-M)<r_0^2} \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma|}} \exp\{-\tfrac{1}{2}(X - M)^t\Sigma^{-1}(X - M)\}dX$$

where the subscript n in Pr_n denotes the number of elements in **X**.

The quantity $r = \sqrt{(X-M)^t\Sigma^{-1}(X-M)}$, is a *chi* statistic, and $Pr_n\{X|\ (X-M)^t\Sigma^{-1}(X-M) \le r_0^2\}$ is given by

$$Pr_n\{X|\ (X-M)^t\Sigma^{-1}(X-M) \le r_0^2\}$$

$$= Pr_n\{r|r^2 < r_0^2\} = C_n\frac{1}{(2\pi)^{\frac{n}{2}}}\int_0^{r_0} r^{n-1}e^{-\frac{1}{2}r^2}dr = P_t \qquad (2.6)$$

where $C_n = \dfrac{2\,\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2})}$ and $\Gamma(\frac{n}{2})$ is a gamma function.

Therefore for a given threshold probability P_t, one can find r_0 by solving equation (2.6) and the region Ω_i for class ω_i is given by equation (2.5). An advantage of the above method of truncation is that the truncation can be performed on an absolute basis. In other words, the truncation can be performed by calculating the likelihood value of a class, and no information about the other classes is required. It is noted that checking truncation by the above method does not require any additional computation. It can be performed as a part of calculating the discriminant function (equation 2.1). Figure 2.4 shows the flowchart of the multistage classifier where class ω_i is truncated if a test sample is found outside region Ω_i containing a prescribed portion of class ω_i.

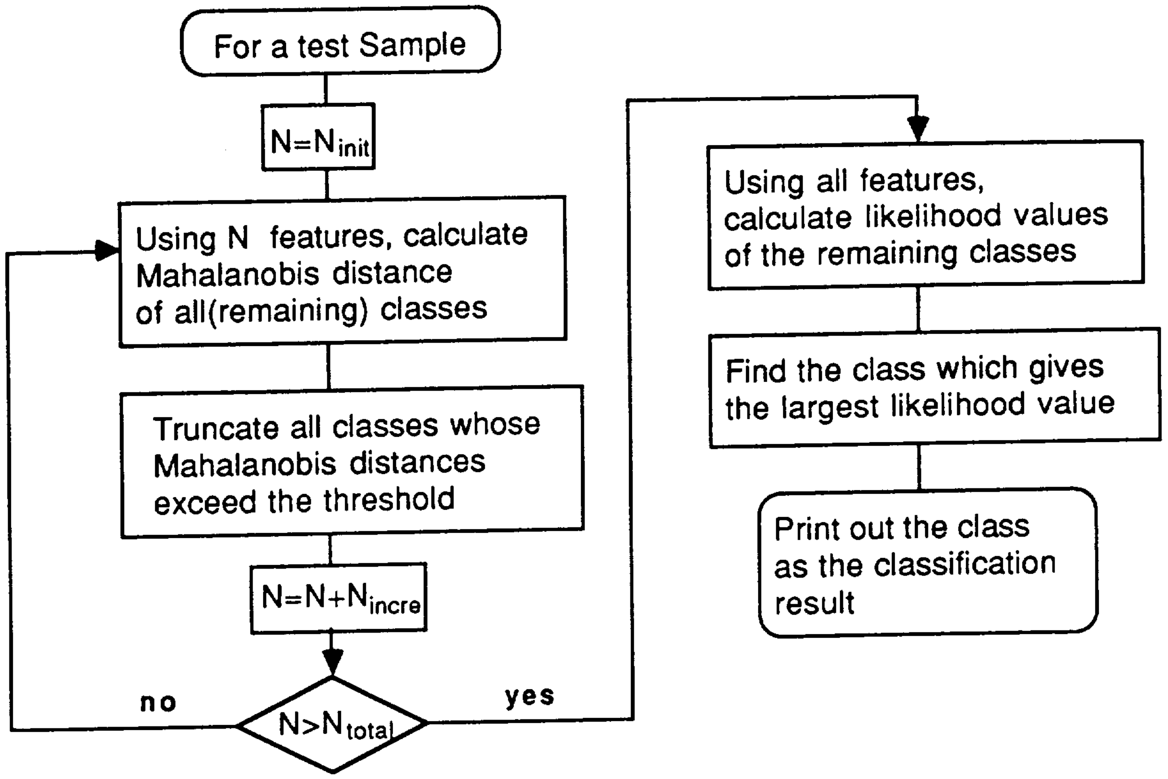
For a test Sample
N=N_init
Using N features, calculate Mahalanobis distance of all(remaining) classes
Truncate all classes whose Mahalanobis distances exceed the threshold
N=N+N_incre
N>N_total
no
yes
Using all features, calculate likelihood values of the remaining classes
Find the class which gives the largest likelihood value
Print out the class as the classification result

Figure 2.4 Flowchart of the multistage classifier.

2.3.3 Truncation by Likelihood Ratio

Another possibility for a truncation criterion is by likelihood ratio, or, equivalently, by difference of the log likelihood values. Since in likelihood classifiers classification is based upon the relative size of the likelihood values, classes which have relatively low likelihood values compared with class ω_L, the class having the largest likelihood value, when only a fraction of the whole feature set is used, would be expected to have lower likelihood values relative to class ω_L when the whole feature set is used. Thus such classes could be truncated at an intermediate stage with little risk of error. To be more precise, at each stage of the multistage classifier the discriminant function (equation 2.1) which is equivalent to the log likelihood value of each class is computed. If $g_i(x) < T$, then class ω_i is truncated, where T is a threshold and determined by

$$T = L - D \text{ where } L = \max(g_i(x), i = 1,m)$$

where m is the number of classes and D is a difference to be selected by the user.

In this case, it is important to understand the relationship between the threshold and the error increment caused by the truncations. We next derive an upper bound on the error increment caused by the truncation.

In a two class classification problem using a Bayes' decision rule with the [0,1] loss function case, the decision is made by the following rule (Fukunaga, 1990).

$$\begin{array}{ll} p(X|\omega_1)P(\omega_1) > p(X|\omega_2)P(\omega_2) & X \sim \omega_1 \\ p(X|\omega_1)P(\omega_1) < p(X|\omega_2)P(\omega_2) & X \sim \omega_2 \end{array}$$

The error probability is given by

$$\varepsilon = P(\omega_1)\varepsilon_1 + P(\omega_2)\varepsilon_2$$

$$\text{where} \quad \varepsilon_1 = \int_t^{\infty} p(h|\omega_1)dh \text{ and } \varepsilon_2 = \int_{-\infty}^{t} p(h|\omega_2)dh$$

$$t = \ln\{\frac{p(\omega_1)}{p(\omega_2)}\}$$

$$h(X) = \log\frac{p(X|\omega_2)}{p(X|\omega_1)}$$

The quantities ε_1 and ε_2 are bounded by

$$\begin{aligned} \varepsilon_1 &\le \exp[-\mu(s) - st] \int_t^{\infty} p_g(g = h|\omega_1)dh \\ &< \exp[-\mu(s) - st] \end{aligned}$$

$$\begin{aligned} \varepsilon_2 &\le \exp[-\mu(s) + (1-s)t] \int_{-\infty}^{t} p_g(g=h|\omega_1)dh \\ &< \exp[-\mu(s) + (1-s)t] \end{aligned}$$

$$\text{where } p_g(g=h|\omega_1) = \exp[sh+\mu(s)]p(h|\omega_1)$$

$$\mu(s) = -\ln \varphi_1(s) = -\ln \int_{-\infty}^{+\infty} \exp(sh)p(h|\omega_1)dh$$

$$t = \ln\{\frac{p(\omega_1)}{p(\omega_2)}\}$$

$\mu(s)$ is obtained by taking the minus logarithm of $\varphi_1(s)$ which is the moment generating function of $h(X)$ for ω_1, and $p_g(g=h|\omega_1)$ is a probability density function. In the case of normal distributions, an explicit mathematical expression for $\mu(s)$ can be obtained.

$$\mu(s) = \frac{s(1-s)}{2}(\mathbf{M}_2 - \mathbf{M}_1)^t [s\Sigma_1+(1-s)\Sigma_2]^{-1}(\mathbf{M}_2-\mathbf{M}_1) + \frac{1}{2}\ln\frac{|s\Sigma_1+(1-s)\Sigma_2|}{|\Sigma_1|^s+|\Sigma_2|^{1-s}}$$

The term $\mu(\frac{1}{2})$ is called the Bhattacharyya distance (Fukunaga, 1990) and is used as a measure of the separability of two classes. The Bhattacharyya distance gives an upper bound for the Bayes' error in the case of normal distributions. By moving the decision boundary, one can reduce the omission error arbitrarily for a specific class even though the overall error may increase.

In the similar way, an upper bound on incremental error of the multistage classification with likelihood value truncation can be obtained. Assume class ω_l has the largest log likelihood value L at the nth stage. Class ω_i is truncated if

$$\ln p^n(X|\omega_i) + \ln\{\frac{p(\omega_i)}{p(\omega_l)}\} + D_{il}^n < \ln p^n(X|\omega_l).$$

The truncation error of class ω_i is bounded by

$$\varepsilon_i^n \leq \exp[\, -\mu_{il}^n(s) - st_{il}^n) \,] \text{ where } t_{il}^n = \ln\{\frac{p(\omega_i)}{p(\omega_l)}\}+D_{il}^n \qquad (2.9)$$

where $\mu_{il}^{n}(s)$ is the J-M distance of class ω_i and class ω_l and D_{il}^{n} is an offset value of class ω_i and class ω_l and superscript n denotes the number of features.

It is noteworthy that the truncation boundary is moved so that truncation error is reduced. Therefore by adjusting D_{il}^{n} which depends on classes ω_i, ω_l and the number of features, the errors caused by truncations can be constrained within any given error limit ε_0.

Figure 2.5 shows the flowchart of the multistage classifier where truncation is done by the differences of log likelihood values. In this example, the number of features is increased by the same amount from stage to stage.

2.3.4 Upper Bound on the Total Incremental Error Probability of Multistage Classifier

Assume that there are M classes and N stages without counting the final stage. The total error increment, ε_{incre}, caused by the truncations can be viewed as the accumulation of truncation error at each stage and can be formulated as

$$\varepsilon_{incre} \leq \sum_{i=1}^{M} p(\omega_i) \sum_{j=1}^{N} T_{ij} \tag{2.10}$$

where

T_{ij} :	Probability that class ω_i is truncated at jth stage.
M :	The number of classes.
N :	The number of stages without counting the final stage.

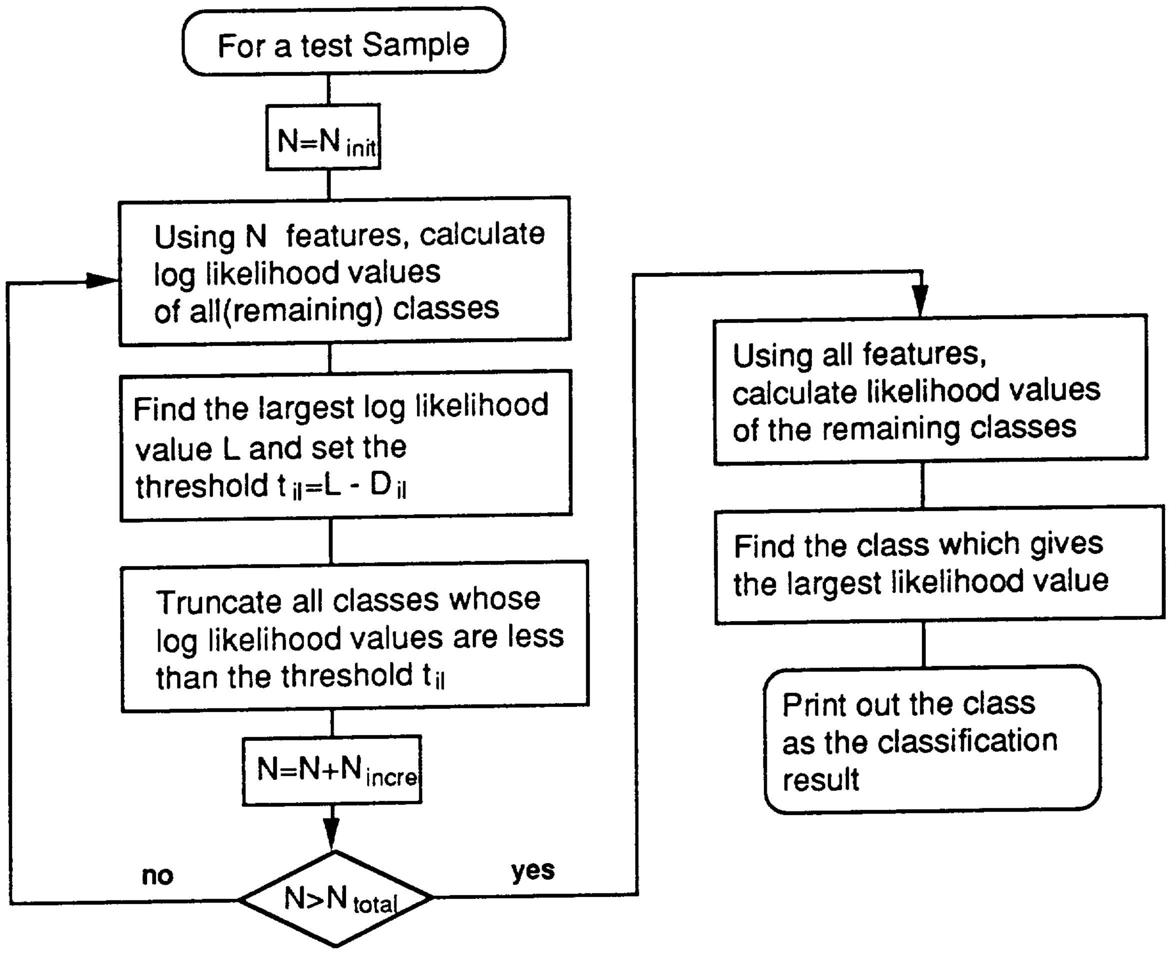
For a test Sample
N=N init
Using N features, calculate log likelihood values of all(remaining) classes
Find the largest log likelihood value L and set the threshold t il=L - D il
Truncate all classes whose log likelihood values are less than the threshold t il
N=N+N incre
N>N total
no
yes
Using all features, calculate likelihood values of the remaining classes
Find the class which gives the largest likelihood value
Print out the class as the classification result

Figure 2.5 Flowchart of the multistage classifier where the truncation is done by the differences of log likelihood values.

If the truncation is done by absolute regions and P_t is the threshold probability, T_{ij} does not depend on the number of classes and is given by

$$T_{ij} = (1-P_t)R_{ij}$$

where R_{ij} is the probability that the samples of class ω_i which are truncated at the j^{th} stage have not been truncated until $(j-1)^{th}$ stage.

From the definition of R_{ij}, it can be easily seen that R_{i1} is 1. In the desirable case, R_{ij} would be zero except R_{i1}, i.e., all classes to be truncated are truncated at the first stage. And the total error increment is given by

$$\begin{aligned}\varepsilon_{\text{incre}} &\le \sum_{i=1}^{M} p(\omega_i) \sum_{j=1}^{N} (1-P_t)R_{ij} \\ &= (1-P_t) \sum_{i=1}^{M} p(\omega_i) = (1-P_t)\end{aligned}$$

In the worst case, R_{ij} would be 1. In other words, the truncation error of class ω_i at each stage is accumulated without any overlap. Then the total error increment is given by

$$\begin{aligned}\varepsilon_{\text{incre}} &\le \sum_{i=1}^{M} p(\omega_i) \sum_{j=1}^{N} (1-P_t)R_{ij} \\ &= N(1-P_t) \sum_{i=1}^{M} p(\omega_i) = N(1-P_t)\end{aligned}$$

Thus even in the worst case which is very unlikely, the total error increment is bounded by

$$\varepsilon_{\text{incre}} \le N(1-P_t)$$

A typical number of intermediate stages in a multistage classifier would be 3 to 5. Thus by carefully choosing P_t, it is possible that the total error increment due to truncation can be constrained within any specified range while achieving a substantial reduction in processing time. In practice, the average value of R_{ij} would be much less than 1. In addition, since most of the samples truncated in intermediate stages would be misclassified at the final stage, the actual error increment due to truncation would be much smaller.

If the truncation is done by likelihood ratio and ε_0 is an error limit, T_{ij} depends on the number of classes and can be formulated as

$$T_{ij} = R_{ij} \sum_{k=1\ k\neq i}^{M} \varepsilon_0 \, Q_{ij}^{k}$$

where R_{ij} is the probability that the samples of class ω_i which are truncated at jth stage have not truncated until the (j−1)th stage, and Q_{ij}^k is the probability that the samples of class ω_i which are truncated by class ω_k at the jth stage have not truncated by other classes at the jth stage.

In the worst case, which is very unlikely, Q_{ij}^k and R_{ij} would be 1. In other words, the truncation error of class ω_i by the other classes are accumulated without any overlap at the jth stage, and the truncation error of class ω_i at each stage is also accumulated without any overlap. In the worst case, the total error increment is bound by

$$\varepsilon_{incre} \leq \sum_{i=1}^{M} p(\omega_i) \sum_{j=1}^{N} R_{ij} \sum_{k=1\ k\neq i}^{M} \varepsilon_0\, Q_{ij}^k \leq (M-1)N\varepsilon_0$$

Therefore, even in the worst case, it is possible that the total error increment due to truncation can be constrained within any specified range by adjusting ε_0. In addition, it is observed that even a significant difference in ε_0 results in a minor difference in computing time. Moreover, in real data, the average values of Q_{ij}^k and R_{ij} would be much less than 1, though the values depend on the characteristics of data. In addition, since most of the samples truncated in intermediate stages would be misclassified at the final stage, the actual error increment due to truncation would be much smaller. Thus by carefully choosing ε_0, it is possible that the total error increment due to truncation can be constrained within any specified range while achieving a substantial reduction in processing time.

2.4 Experiments and Results

Tests were conducted using FSS (Field Spectrometer System) data which has 60 spectral bands (Biehl et al., 1982) The major parameters of FSS are shown in Table 2.1. The data are multi-spectral and multi-temporal.

Table 2.1 Parameters of Field Spectrometer System.

Number of Bands	60
Spectral Cover	0.4 - 2.4 μm
Altitude	60 m
IFOV(ground)	25 m

Figure 2.6 shows the relationship between accuracy and the number of features for a 40-class classification using a conventional Gaussian ML classifier. A total of 13,033 data points were used. Half of the data were used for training and the other half were used for test. From Figure 2.6, it can be seen that accuracies increase as the number of features increases. Though this demonstrates clearly the discriminating power of high dimensional data, the computation cost is also high. The proposed multistage classifier can be successfully employed in such circumstances, in particular for high dimensional and numerous class cases.

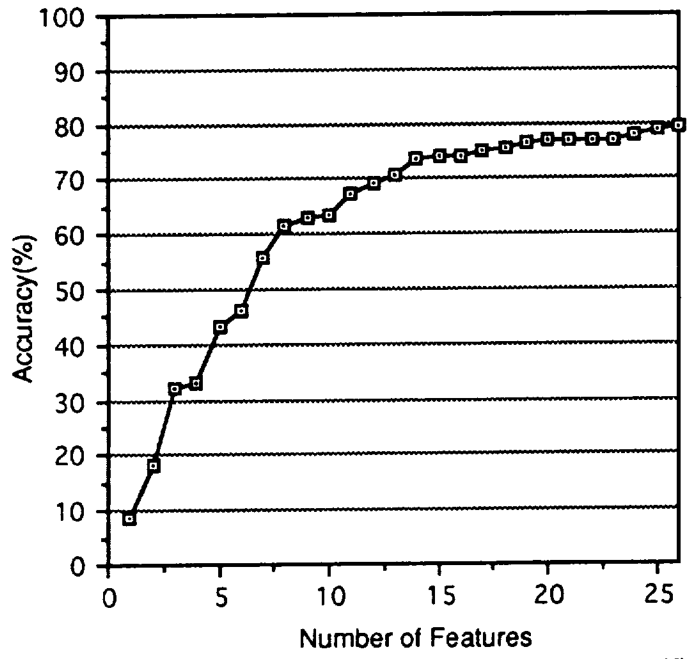

Figure 2.6 Accuracy vs. number of features in a 40-class classification problem.

Two tests were conducted to evaluate the performances of the proposed algorithm. The machine used was a CCI 3/32. The number of classes were 12 and 40 and the numbers of data were 6668 and 13,033, respectively. Half of the data was again used for training and the other half for test. The number of features was reduced to 28 and 26, respectively, using the algorithm proposed by Chen and Landgrebe (1989). Six classifiers were tested for each data set in order to evaluate the performances of the proposed multistage classifiers. The first one was the conventional single stage Gaussian ML classifier. The next two are the multistage classifiers where truncation is done by absolute region Ω_i. Two threshold probabilities, 99.9% and 99%, were tested. The last three were multistage classifiers where truncation was done by the difference of the discriminant function values for ε_0=0.001, 0.005 and 0.01. The number of stages of the tested multistage classifier was 4 in all cases and the number of features used at the first stages was 5, 10 and 15, respectively. The whole feature set was used at the final stage.

Figure 2.7 shows the performance comparison for the case of 12 classes. The computing time of the conventional single stage Gaussian ML classifier, C1, was 117 seconds with an accuracy of 95.2%. The computing time of the multistage classifier, C2, where truncations were done by absolute region Ω_i with the threshold probability, P_t=99.9% was about 31 seconds with an accuracy of 94%; the computing time of the multistage classifier, C3, with the threshold probability, P_t= 99% was 25 seconds with the accuracy of 92.7%. Comparing classifier C1 with classifier C2 and C3, the processing times of multistage classifiers C1 and C2 were 21-27% of that of the single stage classifier C1 with error increased by 1.2% and 2.5%, respectively.

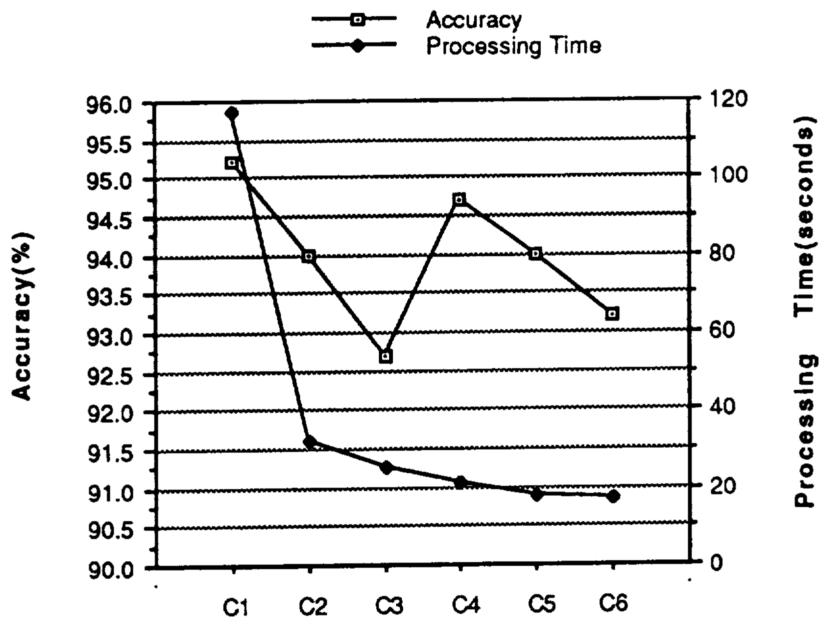

Figure 2.7 Classifier performance comparison for the 12-class case.

The classifiers are as follows:

C1	Single Stage Gaussian ML Classifier.
C2	Multistage Classifier. Truncation by absolute region. P_t=99.9%.
C3	Multistage Classifier. Truncation by absolute region. P_t=99%.
C4	Multistage Classifier. Truncation by the difference of the discriminant function values. ε_0=0.001.
C5	Multistage Classifier. Truncation by the difference of the discriminant function values. ε_0=0.005.
C6	Multistage Classifier. Truncation by the difference of the discriminant function values. ε_0=0.01.

On the other hand, the computing times of the multistage classifiers where truncation was done by the difference of the discriminant function values for ε_0=0.001, 0.005 and 0.01 were 21, 18 and 17 seconds, respectively with accuracies of 94.7%, 94% and 93.2%. It is observed that the processing times were reduced by the factor of 5.6 to 6.9 while errors increased by 0.5%, 1.2% and 2%, respectively. Table 2.2 shows accuracies and error increments for individual classes due to the truncation. It can be seen that the error increments due to the truncation are evenly distributed and no particular class is sacrificed.

Table 2.2 Accuracies and error increments for individual classes due to truncation. Minus signs in the "Error Incre." rows indicate that accuracies increased.

Classifier	Cl.1	Cl.2	Cl.3	Cl.4	Cl.5	Cl.6	Cl.7	Cl.8	Cl.9	Cl.10	Cl.11	Cl.12	Ave.
C1 Acc.	97.4	99.1	96.7	96.7	96.3	72.6	98.1	99.2	91.0	98.3	98.7	97.3	95.2
C2 Acc.	95.4	97.6	96.1	95.1	96.6	73.7	96.1	96.6	89.7	96.1	97.4	94.5	94.0
C2 Err Incre.	2.0	1.5	0.6	1.5	-0.3	-1.2	1.9	2.5	1.3	2.2	1.3	2.7	1.2
C3 Acc.	93.3	96.5	93.7	93.3	95.3	76.8	94.6	95.4	88.5	94.0	94.7	94.5	92.7
C3 Err Incre.	4.1	2.7	3.0	3.3	0.9	-4.2	3.5	3.8	2.6	4.3	4.0	2.7	2.5
C4 Acc.	96.2	98.2	95.8	96.4	96.3	77.2	96.1	97.9	89.7	97.4	98.7	94.5	94.7
C4 Err Incre.	1.2	0.9	0.9	0.3	0.0	-4.6	1.9	1.3	1.3	0.9	0.0	2.7	0.5
C5 Acc.	95.9	97.0	93.7	93.3	96.3	82.2	94.6	97.5	88.0	96.1	97.8	93.6	94.0
C5 Err Incre.	1.4	2.1	3.0	3.3	0.0	-9.7	3.5	1.7	3.0	2.2	0.9	3.6	1.2
C6 Acc.	95.1	97.0	92.7	90.0	96.6	85.3	91.9	96.2	87.6	95.7	97.8	91.8	93.2
C6 Err Incre.	2.3	2.1	3.9	6.7	-0.3	-12.7	6.2	3.0	3.4	2.6	0.9	5.5	2.0

Figure 2.8 shows the performance comparison for the classification with 40 classes. The computing time of the conventional single stage Gaussian ML classifier, C1, was 655 seconds with an accuracy of 79.4% while the computing times of the multistage classifier, C2 and C3 was 188 and 155 seconds with accuracies of 78.3% and 76.8%, respectively. Comparing classifier C1 with multistage classifiers C2 and C3, the processing times of multistage classifiers C1 and C2 were 24% and 29% of that of the single stage classifier C1 with an error increase of 1.1% and 2.6%.

On the other hand, the computing times of multistage classifiers, C4, C5 and C6, were 123, 103 and 93 seconds with accuracies of 78.7%, 77.9% and 77.4%, respectively. It is observed that the processing times were reduced by factor of 5.3 to 7.0 while errors increased by 0.7%, 1.5% and 2%, respectively. In particular, comparing C1 and C4, the processing time for 40 classes was reduced from 652 seconds to 123 seconds, a factor of more than 5, while the accuracy decreased from 79.4% to 78.7%.

In most applications, such error increments due to truncations would be acceptable. It is also observed that most of the test samples which caused truncation error are found at boundaries and may be truncated if a *chi* threshold is applied. In other words, the results for such test samples are not reliable nor critical. In addition, the error tolerance can be adjusted depending on the requirement of the application.

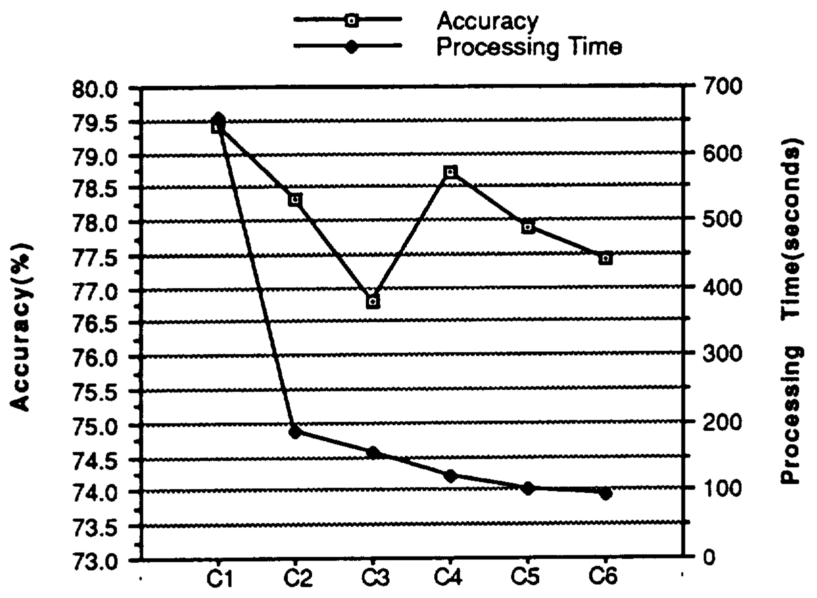

Figure 2.8 Classifier performance comparison for the 40-class case.

2.5 Conclusion

It is shown that the computing time can be reduced by a factor of 3 to 7 using the proposed multistage classification while maintaining essentially the same accuracies when the Gaussian ML classifier is used. Although the proposed algorithm was developed on the assumption of Gaussian ML classifier, the relationship between threshold and the error increments are derived without the assumption of Gaussian ML classification. Thus the proposed algorithm can be used for other classification algorithms if an algorithm to avoid the repeated calculation is developed. Therefore after selecting features which depend on an accuracy requirement, the processing time could be reduced substantially without losing any significant accuracy by employing the multistage classifiers, particularly for high dimensional data.

CHAPTER 3 DECISION BOUNDARY FEATURE EXTRACTION

3.1 Introduction

Linear feature extraction can be viewed as finding a set of vectors that represent an observation while reducing the dimensionality. In pattern recognition, it is desirable to extract features that are focused on discriminating between classes. Although a reduction in dimensionality is desirable, the error increment due to the reduction in dimensionality must be constrained to be adequately small. Finding the minimum number of feature vectors which represent observations with reduced dimensionality without sacrificing the discriminating power of classifiers along with finding the specific feature vectors has been one of the most important problems of the field of pattern analysis and has been studied extensively.

In this chapter, we address this problem and propose a new algorithm for feature extraction based directly on the decision boundary. The algorithm predicts the minimum number of features to achieve the same classification accuracy as in the original space; at the same time the algorithm finds the needed feature vectors. Noting that feature extraction can be viewed as retaining informative features or eliminating redundant features, we define the terms "discriminantly informative" feature and "discriminantly redundant" feature. This reduces feature extraction to finding discriminantly informative features. We will show how discriminantly informative features and discriminantly redundant features are related to the decision boundary and can be derived from the decision boundary. We will need to define several terms and derive several theorems and, based on the theorems, propose a procedure to find discriminantly informative features from the decision boundary.

3.2 Background and previous works

Most linear feature extraction algorithms can be viewed as linear transformations. One of the most widely used transforms for signal representation is the Karhunen-Loeve transformation. Although the Karhunen-Loeve transformation is optimum for signal representation in the sense that it provides the smallest mean square error for a given number of features, quite often the features defined by the Karhunen-Loeve transformation are not optimum with regard to class separability (Malina 1987). In feature extraction for classification, it is not the mean square error but the classification accuracy that must be considered as the primary criterion for feature extraction.

Many authors have attempted to find the best features for classification based on criterion functions. Fisher's method finds the vector that gives the greatest class separation as defined by a criterion function (Duda and Hart 1973). Fisher's linear discriminant can be generalized to multiclass problems. In canonical analysis (Richards 1986), a within-class scatter matrix Σ_W and a between-class scatter matrix Σ_b are used to formulate a criterion function and a vector **d** is selected to maximize

$$\frac{\mathbf{d}^t\Sigma_b\mathbf{d}}{\mathbf{d}^t\Sigma_w\mathbf{d}}$$

where

$$\Sigma_W = \sum_i P(\omega_i)\Sigma_i \qquad \text{(within-class scatter matrix)}$$

$$\Sigma_b = \sum_i P(\omega_i)(\mathbf{M}_i - \mathbf{M}_0)(\mathbf{M}_i - \mathbf{M}_0)^t \quad \text{(between-class scatter matrix)}$$

$$\mathbf{M}_0 = \sum_i P(\omega_i)\mathbf{M}_i$$

Here $\mathbf{M}_i$, Σ_i, and $P(\omega_i)$ are the mean vector, the covariance matrix, and the prior probability of class ω_i, respectively. Although the vector found by canonical analysis performs well in most cases, there are several problems with canonical analysis. First of all, if there is little or no difference in mean vectors, the feature vector selected by canonical analysis is not reliable. Second, if a class has a mean vector very different from the mean vectors of the other classes, that class will be dominant in calculating the between-class scatter matrix, thus resulting in ineffective feature extraction.

Fukunaga recognized that the best representational features are not necessarily the best discriminating features and proposed a preliminary transformation (Fukunaga and Koontz 1970). The Fukunaga-Koontz method first finds a transformation matrix **T** such that,

$$\mathbf{T}[\mathbf{S}_1 + \mathbf{S}_2]\mathbf{T}^{-t} = \mathbf{I}$$

where $\mathbf{S}_i$ is the autocorrelation matrix of class ω_i.
Fukunaga showed that $\mathbf{TS}_1\mathbf{T}^{-t}$ and $\mathbf{TS}_2\mathbf{T}^{-t}$ have the same eigenvectors and all the eigenvalues are bounded by 0 and 1. It can be seen that the eigenvector with the largest differences in eigenvalues is the axis with the largest differences in variances. The Fukunaga-Koontz method will work well in problems where the covariance difference is dominant with little or no mean difference. However, by ignoring the information of mean difference, the Fukunaga-Koontz method is not suitable in the general case and could lead to irrelevant results (Foley and Sammon 1975).

Kazakos proposed a linear scalar feature extraction algorithm that minimizes the probability of error in discriminating between two multivariate normally distributed pattern classes (Kazakos 1978). By directly employing the probability of error, the feature extraction method finds the best single feature vector in the sense that it gives the smallest error. However, if more than one feature is necessary, it is difficult to generalize the method.

Heydorn proposed a feature extraction method by deleting redundant features where redundancy is defined in terms of a marginal distribution function (Heydorn 1971). The redundancy test uses a coefficient of redundancy. However, the method does not find a redundant feature vector unless the vector is in the direction of one of the original feature vectors even though the redundant feature vector could be detected by a linear transformation.

Decell et al. developed an explicit expression for the smallest compression matrix such that the Bayes classification regions are preserved (Decell et al. 1981). Young et al. extended the method to a general class of density functions know as θ-generalized normal densities (Young et al. 1985) and Tubbs et al. discussed the problem of unknown population parameters (Tubbs et al. 1982).

Feature selection using statistical distance measures has also been widely studied and successfully applied [(Swain and King 1973), (Swain and Davis 1978), and (Kailath 1967)]. However, as the dimension of data increases, the combination of bands to be examined increases exponentially, resulting in unacceptable computational cost. Several procedures to find a sub-optimum combination of bands instead of the optimum combination of bands have been proposed with a reasonable computational cost (Devijver and Kittler 1982). However, if the best feature vector or the best set of feature vectors is not in the direction of any original feature vector, more features may be needed to achieve the same performance.

Depending on the characteristics of the data, it has been shown that the previous feature extraction/selection methods can be applied successfully. However, it is also true that there are some cases in which the previous methods fail to find the best feature vectors or even good feature vectors, thus resulting in difficulty in choosing a suitable method to solve a particular problem. Although some authors addressed this problem [(Malina 1981) and (Longstaff 1987)], there is still another problem. One must determine, for a given problem, how many features must be selected to meet the requirement. More fundamentally, it is difficult with the previous feature extraction/selection algorithms to predict the intrinsic discriminant dimensionality, which is defined as the smallest number of features needed to achieve the same classification accuracy as in the original space for a given problem.

In this chapter, we propose a different approach to the problem of feature extraction for classification. The proposed algorithm is based on decision boundaries directly. The proposed algorithm predicts the minimum number of features needed to achieve the same classification accuracy as in the original space for a given problem and finds the needed feature vectors, and it does not deteriorate when mean or covariance differences are small.

3.3 Feature Extraction and subspace

3.3.1 Feature Extraction and Subspace

Let **X** be an observation in the N-dimensional Euclidean space E^N. Then **X** can be represented by

$$\mathbf{X} = \sum_{i=1}^{N} a_i \alpha_i \quad \text{where } \{\alpha_1, \alpha_2, .., \alpha_N\} \text{ is a basis of } E^N$$

Then feature extraction is equivalent to finding a subspace, **W**, and the new features can be found by projecting an observation into the subspace. Let **W** be a M-dimensional subspace of E^N spanned by M linearly independent vectors, $\beta_1, \beta_2, .., \beta_M$.

$$\mathbf{W} = \text{Span}\{\beta_1, \beta_2, ..., \beta_M\} \text{ and } \dim(\mathbf{W}) = M \leq N$$

Assuming that β_i's are orthonormal, the new feature set in subspace **W** is given by

$$\{\mathbf{X}^t\beta_1, \mathbf{X}^t\beta_2, .., \mathbf{X}^t\beta_M\} = \{b_1, b_2, .., b_M\} \text{ where } b_i = \mathbf{X}^t\beta_i$$

Now let $\hat{X} = \sum_{i=1}^{M} b_i \beta_i$. Then $\hat{X}$ will be an approximation to **X** in terms of a linear combination of $\{\beta_1, \beta_2, .., \beta_M\}$ in the original N-dimensional space.

3.3.2 Bayes' Decision Rule for Minimum Error

Now consider briefly Bayes' decision rule for minimum error, which will be used later in the proposed feature extraction algorithm. Let **X** be an observation in the N-dimensional Euclidean space E^N under hypothesis H_i: $\mathbf{X} \in \omega_i$ i=1,2. Decisions will be made according to the following rule.

$$\text{Decide } \omega_1 \text{ if } P(\omega_1)P(\mathbf{X}|\omega_1) > P(\omega_2)P(\mathbf{X}|\omega_2)$$
$$\text{else } \omega_2$$

where $P(\mathbf{X}|\omega_i)$ is a conditional density function and $P(\omega_i)$ is a priori probability of class ω_i.

Let $h(\mathbf{X}) = -\ln\frac{P(\mathbf{X}|\omega_1)}{P(\mathbf{X}|\omega_2)}$ and $t = \ln\frac{P(\omega_1)}{P(\omega_2)}$. Then

$$\begin{array}{ll} \text{Decide } \omega_1 & \text{if } h(\mathbf{X}) < t \\ \text{else } \omega_2 & \end{array}$$

Feature extraction has been used in many applications, and the criteria for feature extraction can be different in each case. If feature extraction is directed specifically at classification, a criterion could be to maintain classification accuracy. As a new approach to feature extraction for classification, we will find a subspace, **W**, with the minimum dimension M and the spanning vectors $\{\beta_i\}$ of the subspace such that for any observation **X**

$$(h(\mathbf{X}) - t)(h(\hat{\mathbf{X}}) - t) > 0 \qquad (3.1)$$

where $\hat{\mathbf{X}}$ is an approximation of **X** in terms of a basis of subspace **W** in the original N-dimensional space. The physical meaning of (3.1) is that the classification result for $\hat{\mathbf{X}}$ is the same as the classification result of **X**. In practice, feature vectors might be selected in such a way as to maximize the number of observations for which (3.1) holds with a constraint on the dimensionality of subspaces. In this chapter, we will propose an algorithm which finds the minimum dimension of a subspace such that (3.1) holds for all the given observations and which also finds the spanning vectors $\{\beta_i\}$ of the subspace. In the next section, we define some needed terminology which will be used in deriving theorems later.

3.4 Definitions

3.4.1 Discriminantly Redundant Feature

Feature extraction can be performed by eliminating redundant features, however, what is meant by "redundant" may be dependent on the application.

For the purpose of feature extraction for classification, we will define a "discriminantly redundant feature" as follows.

Definition 3.1 We say the vector β_k is discriminantly redundant if for any observation **X**

$$(h(\mathbf{X}) - t)(h(\hat{\mathbf{X}}) - t) > 0 \qquad (3.1)$$

In other words,

if $h(\mathbf{X}) > t$, then $h(\hat{\mathbf{X}}) > t$ or

if $h(\mathbf{X}) < t$, then $h(\hat{\mathbf{X}}) < t$

where $\mathbf{X} = \sum_{i=1}^{N} b_i\beta_i$ and $\hat{\mathbf{X}} = \sum_{i=1, i\neq k}^{N} b_i\beta_i$

The physical meaning of (3.1) is that the classification result for $\hat{\mathbf{X}}$ is the same as the classification result of **X**. Figure 3.1 shows an example of a discriminantly redundant feature. In this case even though **X** is moved along the direction of vector β_k, the classification result will remain unchanged. This means vector β_k makes no contribution in discriminating between classes, thus vector β_k is redundant for the purpose of classification.

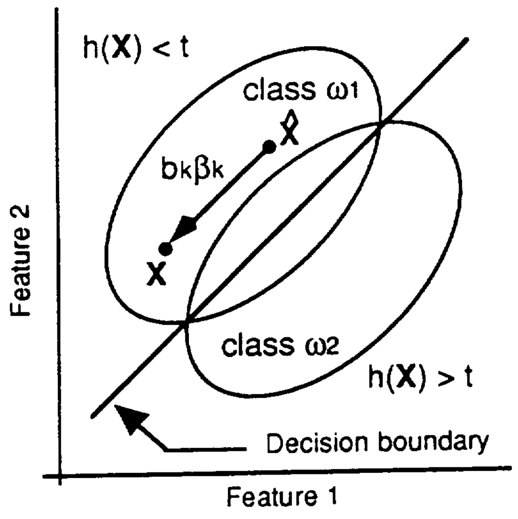

Figure 3.1 An example of a discriminantly redundant feature.

3.4.2 Discriminantly Informative Feature

In a similar manner, we define a discriminantly informative feature.

Definition 3.2 We say that β_k is discriminantly informative if there exists <u>at least one observation</u> $\mathbf{Y}$ such that

$$(h(\mathbf{Y}) - t)(h(\hat{\mathbf{Y}}) - t) < 0 \tag{3.2}$$

In other words,

$$h(\mathbf{Y}) > t \text{ but } h(\hat{\mathbf{Y}}) < t \text{ or}$$

$$h(\mathbf{Y}) < t \text{ but } h(\hat{\mathbf{Y}}) > t$$

$$\text{where } \mathbf{Y} = \sum_{i=1}^{N} b_i\beta_i \text{ and } \hat{\mathbf{Y}} = \sum_{i=1,\, i\neq k}^{N} b_i\beta_i$$

The physical meaning of (3.2) is that there exists an observation $\mathbf{Y}$ such that the classification result of $\hat{\mathbf{Y}}$ is different from the classification result of $\mathbf{Y}$. It is noted that (3.2) need not hold for all observations. A vector will be discriminantly informative if there exists at least one observation whose classification result can be changed as the observation moves along the direction of the vector. Figure 3.2 shows an example of a discriminantly informative feature. In this case, as $\mathbf{Y}$ is moved along the direction of vector β_k, the classification result will be changed.

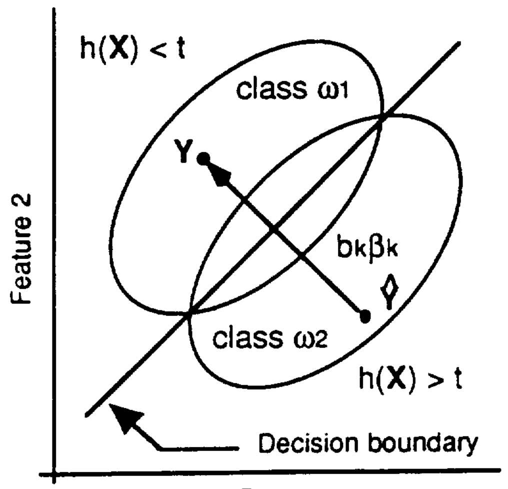

Figure 3.2 An example of a discriminantly informative feature.

3.4.3 Decision Boundaries and Effective Decision Boundaries

The decision boundary of a two-class problem is a locus of points on which a posteriori probabilities are the same. To be more precise, we define a decision boundary as follows:

Definition 3.3 A decision boundary is defined as

$$\{ \mathbf{X} \mid h(\mathbf{X}) = t \}$$

A decision boundary can be a point, line, curved surface or curved hyper-surface. Although a decision boundary can be extended to infinity, in most cases some portion of the decision boundary is not significant. For practical purposes, we define the effective decision boundary as follows:

Definition 3.4 The effective decision boundary is defined as

$$\{ \mathbf{X} \mid h(\mathbf{X}) = t \text{ , } \mathbf{X} \in R_1 \text{ or } \mathbf{X} \in R_2 \}$$

where R_1 is the smallest region which contains a certain portion, $P_{threshold}$, of class ω_1 and R_2 is the smallest region which contains a certain portion, $P_{threshold}$, of class ω_2.

The effective decision boundary may be seen as an intersection of the decision boundary and the regions where most of the data are located. Figures 3.3 and 3.4 show some examples of decision boundaries and effective decision boundaries. In these examples, the threshold probability, $P_{threshold}$, is set to 99.9%. In the case of Figure 3.3, the decision boundary is a straight line and the effective decision boundary is a straight line segment, the latter being a part of the former. In Figure 3.4, the decision boundary is an ellipse and the effective decision boundary is a part of that ellipse.

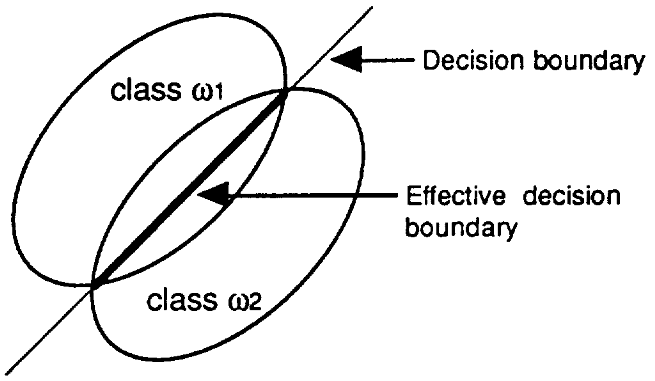

Figure 3.3 $M_1 \neq M_2$, $\Sigma_1 = \Sigma_2$. The decision boundary is a straight line and the effective decision boundary is a line segment coincident to it.

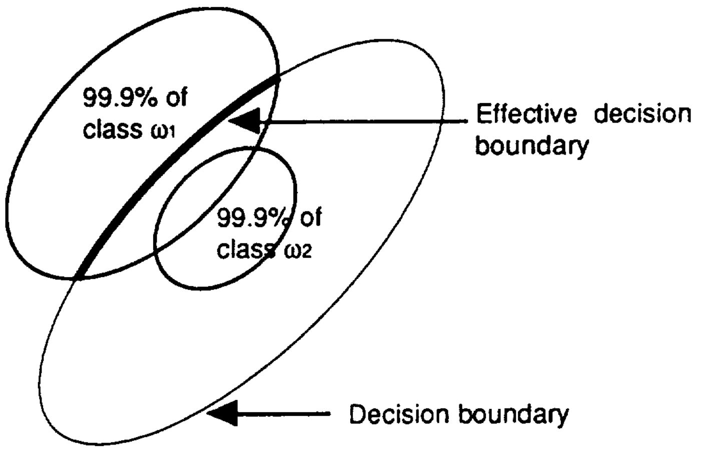

Figure 3.4 $M_1 \neq M_2$, $\Sigma_1 \neq \Sigma_2$. The decision boundary and the effective decision boundary.

3.4.4 Intrinsic Discriminant Dimension

One of the major problems of feature extraction for classification is to find the minimum number of features needed to achieve the same classification accuracy as in the original space. To be more exact, we define the term, "intrinsic discriminant dimension".

Definition 3.5 The Intrinsic discriminant dimension for a given problem is defined as the smallest dimension of a subspace, **W**, of the N-dimensional Euclidean space E^N such that for any observation **X** in the problem,

$$(h(\mathbf{X}) - t)(h(\hat{X}) - t) > 0$$

$$\text{where } \hat{X} = \sum_{i=1}^{M} a_i \beta_i \in \mathbf{W} \text{ and } M \leq N.$$

The intrinsic discriminant dimension can be seen as the smallest dimensional subspace wherein the same classification accuracy can be obtained as could be obtained in the original space.

The intrinsic discriminant dimension is related to the discriminantly redundant feature vector and the discriminantly informative feature vector. In particular, if there are M linearly independent discriminantly informative feature vectors and L linearly independent discriminantly redundant feature vectors, then it can be easily seen that

$$N = M + L$$

where N is the original dimension and the intrinsic discriminant dimension is equal to M. Figure 3.5 shows an example of the intrinsic discriminant dimension. In the case of Figure 3.5, the intrinsic discriminant dimension is one dimension. In the case of Figure 3.5, the intrinsic discriminant dimension is one even though the original dimensionality is two. If V_2 is chosen as a new feature vector, the classification accuracy will be the same as in the original 2-dimensional space.

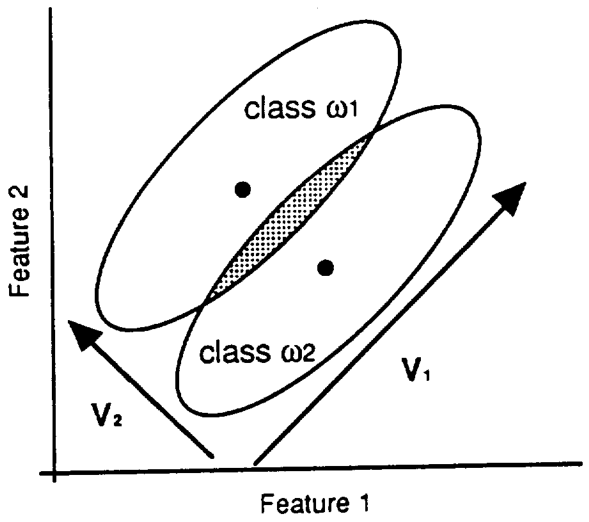

Figure 3.5 $\Sigma_1=\Sigma_2$. In this case the intrinsic discriminant dimension is one even though the original space is two dimensional.

3.5 Feature Extraction Based on the Decision Boundary

3.5.1 Redundancy Testing Theorem

From the definitions given in the previous section, a useful theorem can be stated which tests whether a feature vector is a discriminantly redundant feature or a discriminantly informative feature.

Theorem 3.1 If a vector is parallel to the tangent hyper-plane to the decision boundary at every point on the decision boundary for a pattern classification problem, the vector contains no information useful in discriminating between classes for the pattern classification problem, i.e., the vector is discriminantly redundant.

Proof. Let $\{\beta_1,\beta_2,..,\beta_N\}$be a basis of the N-dimensional Euclidean space E^N, and let β_N be a vector that is parallel to the tangent hyper-plane to the decision boundary at every point on the decision boundary. Let **W** be a subspace spanned by N-1 spanning vectors, $\beta_1,\beta_2,..,\beta_{N-1}$, i.e.,

$$\mathbf{W} = \mathrm{Span}\{\beta_1,\beta_2,..,\beta_{N-1}\} \text{ and } \dim(\mathbf{W}) = N\text{-}1$$

If b_N is not a discriminantly redundant feature, there must exist an observation **X** such that

$$(h(\mathbf{X}) - t)(h(\hat{X}) - t) < 0$$

$$\text{where } \mathbf{X} = \sum_{i=1}^{N} b_i\beta_i \text{ and } \hat{X} = \sum_{i=1}^{N-1} c_i\beta_i$$

Without loss of generality, we can assume that the set of vectors $\beta_1,\beta_2,..,\beta_N$ is an orthonormal set. Then $b_i = c_i$ for i=1,N-1. Assume that there is an observation **X** such that

$$(h(\mathbf{X}) - t)(h(\hat{X}) - t) < 0$$

This means **X** and $\hat{X}$ are on different sides of the decision boundary. Then the vector

$$\mathbf{X}_d = \mathbf{X} - \hat{X} = b_N\beta_N$$

where b_N is a coefficient, must pass through the decision boundary. But this contradicts the assumption that β_N is parallel to the tangent hyper-plane to the decision boundary at every point on the decision boundary. Therefore if β_N is a vector parallel to the tangent hyper-plane to the decision boundary at every point on the decision boundary, then for all observations **X**

$$(h(\mathbf{X}) - t)(h(\hat{\mathbf{X}}) - t) > 0$$

Therefore β_N is discriminantly redundant. Figure 3.6 shows an illustration of the proof.

Q.E.D.

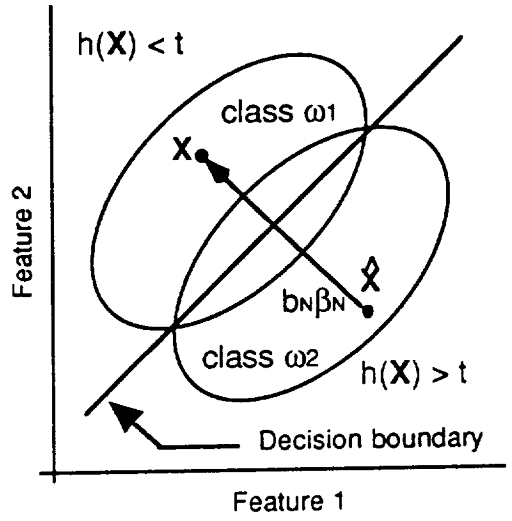

Figure 3.6 If two observations are on the different sides of the decision boundary, the line connecting the two observations will pass through the decision boundary.

It is noted that we did not make any assumption on the number of classes in proving Theorem 3.1. In other words, Theorem 3.1 holds for any number of classes. From the theorem, we can easily derive the following lemmas which are very useful in finding discriminantly informative features.

Lemma 3.1 If vector **V** is orthogonal to the vector normal to the decision boundary at every point on the decision boundary, vector **V** contains no information useful in discriminating between classes, i.e., vector **V** is discriminantly redundant.

Lemma 3.2 If a vector is normal to the decision boundary at at least one point on the decision boundary, the vector contains information useful in discriminating between classes, i.e., the vector is discriminantly informative.

3.5.2 Decision Boundary Feature Matrix

From the previous theorem and lemmas, it can be seen that a vector normal to the decision boundary at a point is a discriminantly informative feature, and the effectiveness of the vector is roughly proportional to the area of the decision boundary which has the same normal vector. Now we can define a DECISION BOUNDARY FEATURE MATRIX which is very useful to predict the intrinsic discriminant dimension and find the necessary feature vectors.

Definition 3.6 The decision boundary feature matrix (DBFM): Let **N(X)** be the unit normal vector to the decision boundary at a point **X** on the decision boundary for a given pattern classification problem. Then the decision boundary feature matrix Σ_{DBFM} is defined as

$$\Sigma_{DBFM} = \frac{1}{K} \int_S \mathbf{N}(\mathbf{X})\mathbf{N}^t(\mathbf{X})p(\mathbf{X})d\mathbf{X}$$

where $p(\mathbf{X})$ is a probability density function, $K= \int_S p(\mathbf{X})d\mathbf{X}$, and S is the decision boundary, and the integral is performed over the decision boundary.

We will show some examples of the decision boundary feature matrices next. Even though the examples are in 2-dimensional space, the concepts can be easily extended to higher dimensional spaces. In all examples, a Gaussian Maximum Likelihood classifier is assumed.

Example 3.1 The mean vectors and covariance matrices of two classes are given as follows:

$$\mathbf{M}_1 = \begin{bmatrix}1\\2\end{bmatrix}, \quad \Sigma_1 = \begin{bmatrix}1 & 0.5\\0.5 & 1\end{bmatrix} \quad \mathbf{M}_2 = \begin{bmatrix}2\\1\end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix}1 & 0.5\\0.5 & 1\end{bmatrix}$$

$$P(\omega_1) = P(\omega_2) = 0.5$$

These distributions are shown in Figure 3.7 as "ellipses of concentration." In a two-class, two-dimensional pattern classification problem, if the covariance matrices are the same, the decision boundary will be a straight line and the intrinsic discriminant dimension is one. This suggests that the vector normal to the decision boundary at any point is the same. And the decision boundary feature matrix will be given by

$$\Sigma_{DBFM} = \frac{1}{K}\int_S \mathbf{N}(\mathbf{X})\mathbf{N}^t(\mathbf{X})p(\mathbf{X})d\mathbf{X} = \frac{1}{K}\mathbf{N}\mathbf{N}^t \int_S p(\mathbf{X})d\mathbf{X} = \mathbf{N}\mathbf{N}^t$$

$$\Sigma_{DBFM} = \frac{1}{\sqrt{2}}(-1,1)^t \frac{1}{\sqrt{2}}(-1,1) = \frac{1}{2}\begin{bmatrix}1 & -1\\-1 & 1\end{bmatrix}$$

$$\text{Rank}(\Sigma_{DBFM}) = 1$$

It is noted that the rank of the decision boundary feature matrix is one which is equal to the intrinsic discriminant dimension and the eigenvector corresponding to the non-zero eigenvalue is the desired feature vector which gives the same classification accuracy as in the original 2-dimensional space.

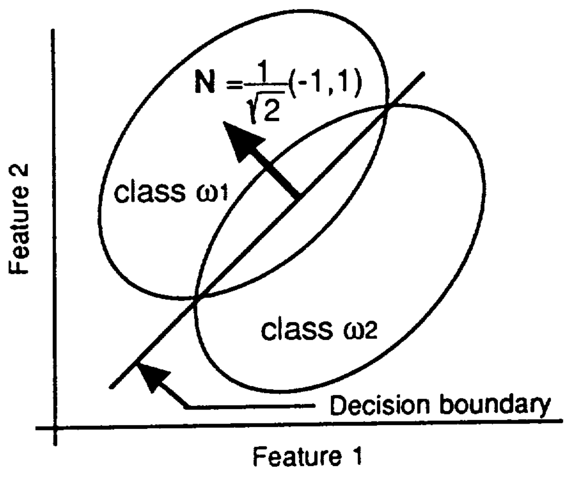

Figure 3.7 An example where the covariance matrices of two classes are the same and the decision boundary is a straight line.

Example 3.2 The mean vectors and covariance matrices of two classes are given as follows:

$$\mathbf{M}_1 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \quad \Sigma_1 = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \qquad \mathbf{M}_2 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$P(\omega_1) = P(\omega_2) = 0.5$$

The distributions of the two classes are shown in Figure 3.8 as "ellipses of concentration." In this example, the decision boundary is a circle and symmetric, and $\frac{1}{K}p(\mathbf{X})$ is a constant given by $\frac{1}{2\pi r}$ where r is the radius of the circle. The decision boundary feature matrix will be given by

$$\Sigma_{DBFM} = \int_0^{2\pi} \frac{1}{2\pi r}[\cos\theta \ \sin\theta]^t[\cos\theta \ \sin\theta] \, r \, d\theta$$

$$= \frac{1}{2\pi}\int_0^{2\pi} \begin{bmatrix} \cos\theta\cos\theta & \cos\theta\sin\theta \\ \sin\theta\cos\theta & \sin\theta\sin\theta \end{bmatrix} d\theta$$

$$= \frac{1}{2\pi}\begin{bmatrix} \pi & 0 \\ 0 & \pi \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathrm{Rank}(\Sigma_{DBFM}) = 2$$

From the distribution of data, it is seen that two features are needed to achieve the same classification accuracy as in the original space. This means that the intrinsic discriminant dimension is 2 in this case. It is noted that the rank of the decision boundary feature matrix is also 2, which is equivalent to the intrinsic discriminant dimension.

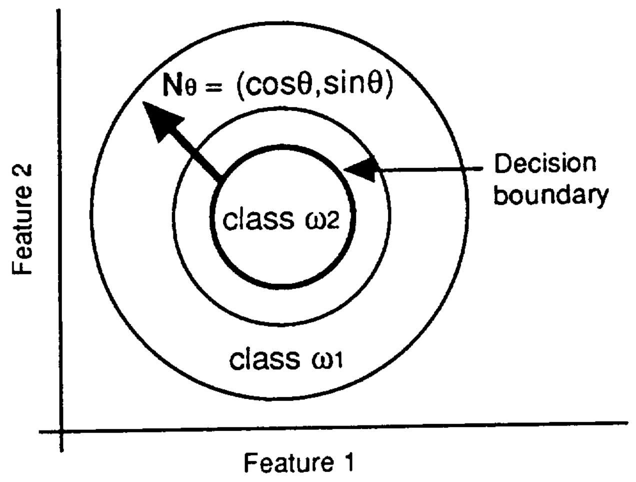

Figure 3.8 The decision boundary feature matrix for equal means and different covariances.

In a similar way, we define an EFFECTIVE DECISION BOUNDARY FEATURE MATRIX. The effective decision boundary feature matrix is the same as the decision boundary feature matrix except that only the effective decision boundary instead of the entire decision boundary is considered.

Definition 3.7 The effective decision boundary feature matrix (EDBFM): Let $\mathbf{N(X)}$ be the unit normal vector to the decision boundary at a point $\mathbf{X}$ on the effective decision boundary for a given pattern classification problem. Then the effective decision boundary feature matrix Σ_{EDBFM} is defined as

$$\Sigma_{EDBFM} = \frac{1}{K'} \int_{S'} \mathbf{N(X)N^t(X)}p(\mathbf{X})d\mathbf{X}$$

where $p(\mathbf{X})$ is a probability density function, $K' = \int_{S'} p(\mathbf{X})d\mathbf{X}$, and S' is the effective decision boundary as defined in Definition 3.4, and the integral is performed over the effective decision boundary.

3.5.3 Properties of Decision Boundary Feature Matrix

In this section, some properties of the decision boundary feature matrix will be discussed.

Property 3.1 The decision boundary feature matrix is a real, symmetric matrix.

Proof: It can be shown that $\Sigma_{DBFM} = (\Sigma_{DBFM})^t$ as follows:

$$
\begin{aligned}
(\Sigma_{DBFM})^t &= \{\frac{1}{K} \int_S N(\mathbf{X})N^t(\mathbf{X})p(\mathbf{X})d\mathbf{X}\}^t \\
&= \frac{1}{K} \int_S \{N(\mathbf{X})N^t(\mathbf{X})\}^t p(\mathbf{X})d\mathbf{X} \\
&= \frac{1}{K} \int_S N(\mathbf{X})N^t(\mathbf{X})p(\mathbf{X})d\mathbf{X} \\
&= \Sigma_{DBFM}
\end{aligned}
$$

Property 3.2 The eigenvectors of the decision boundary feature matrix are orthogonal.

Proof: Since the decision boundary feature matrix is a real symmetric matrix, the eigenvectors of the decision boundary feature matrix are orthogonal (Cullen 1972).

Property 3.3 The decision boundary feature matrix is positive semi-definite.

Proof: Let **N** be a real column vector. Then the matrix, $\mathbf{NN}^t$, is positive semi-definite (Cullen 1972). Let λ be an eigenvalue of Σ_{DBFM} and $\varphi \neq 0$ an associated eigenvector. Then

$$\Sigma_{DBFM}\,\varphi=\lambda\varphi$$

And

$$\varphi^t\Sigma_{DBFM}\,\varphi=\varphi^t\lambda\varphi$$

$$\lambda = \frac{\varphi^t\Sigma_{DBFM}\varphi}{\varphi^t\varphi}$$

$$=\frac{1}{\varphi^t\varphi}\varphi^t\{\frac{1}{K}\int_S \mathbf{N}(\mathbf{X})\mathbf{N}^t(\mathbf{X})p(\mathbf{X})d\mathbf{X}\}\varphi$$

$$=\frac{1}{\varphi^t\varphi}\frac{1}{K}\int_S \varphi^t\mathbf{N}(\mathbf{X})\mathbf{N}^t(\mathbf{X})\varphi p(\mathbf{X})d\mathbf{X} \geq 0$$

where $\varphi^t\mathbf{N}(\mathbf{X})\mathbf{N}^t(\mathbf{X})\varphi \geq 0$ for any **X**,

$p(\mathbf{X}) \geq 0$ since $p(\mathbf{X})$ is a probability density function,

$K= \int_S p(\mathbf{X})d\mathbf{X} \geq 0,$

$\varphi^t\varphi > 0.$

Thus, the decision boundary feature matrix is also positive semi-definite.

Property 3.4 The decision boundary feature matrix of the whole decision boundary can be expressed as a summation of the decision boundary feature matrices calculated from segments of the whole decision boundary if the segments are mutually exclusive and exhaustive.

Proof: Let S be the whole decision boundary. Let $S_1 \cup S_2=S$ and $S_1 \cap S_2=\varnothing$. Then

$$\Sigma_{DBFM} = \frac{1}{K}\int_{S} N(X)N^t(X)p(X)dX$$

$$= \frac{1}{K}\int_{S_1} N(X)N^t(X)p(X)dX + \frac{1}{K}\int_{S_2} N(X)N^t(X)p(X)dX$$

$$= \Sigma^{S_1}_{DBFM} + \Sigma^{S_2}_{DBFM}$$

where $\Sigma^{S_i}_{DBFM}$ is the decision boundary feature matrix calculated from the segment decision boundary S_i.

Figure 3.9 shows an illustration. The decision boundary of Figure 3.9 is a circle. Let S_1 be the upper half of the circle and S_2 the lower half of the circle. Then the decision boundary feature matrix can be expressed as a summation of the decision boundary feature matrix calculated from S_1 and the decision boundary feature matrix calculated from S_2.

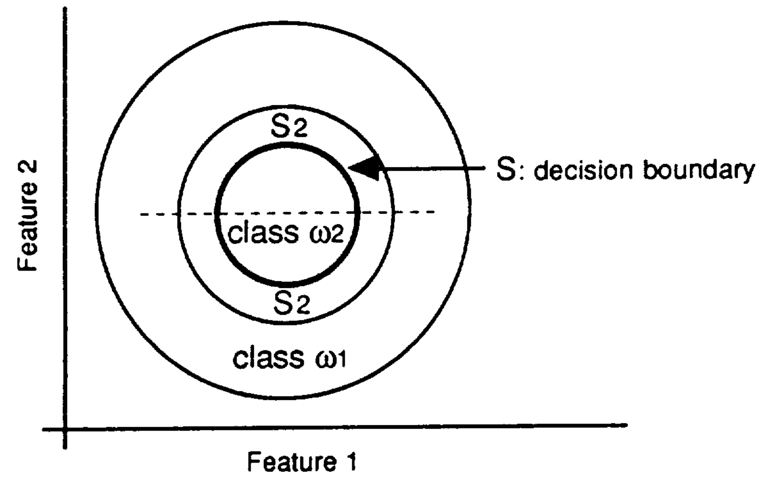

Figure 3.9 The decision boundary feature matrix can be calculated by segments.

From Property 3.4, we can calculate the decision boundary feature matrix of a multiclass problem by summing up the decision boundary feature matrices of each pair of classes. Figure 3.10 shows an example. The decision boundary feature matrix of the 3-class problem can be calculated as follows:

$$\Sigma_{DBFM} = \frac{1}{K}\int_{S} N(X)N^t(X)p(X)dX$$

$$= \frac{1}{K}\int_{S_{12}} N(X)N^t(X)p(X)dX + \frac{1}{K}\int_{S_{13}} N(X)N^t(X)p(X)dX$$

$$+ \frac{1}{K}\int_{S_{23}} N(X)N^t(X)p(X)dX$$

$$= \Sigma_{DBFM}^{S_{12}} + \Sigma_{DBFM}^{S_{13}} + \Sigma_{DBFM}^{S_{23}}$$

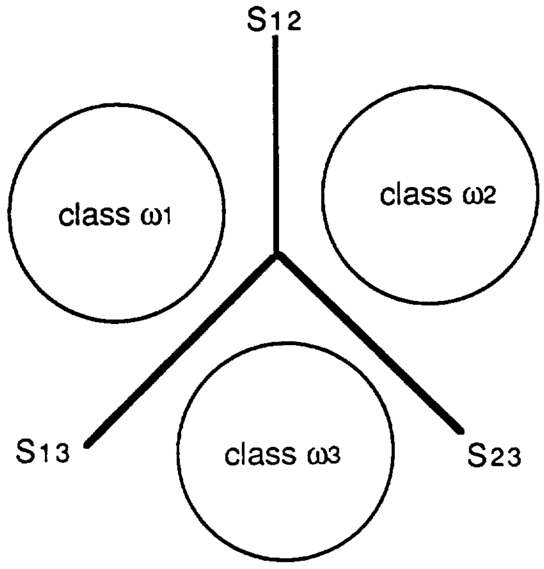

Figure 3.10 The decision boundary feature matrix of a multiclass problem can be calculated from the decision boundary feature matrices from each pair of classes.

3.5.4 Decision Boundary Feature Matrix for Finding the Intrinsic Discriminant Dimension and Feature Vectors

From the way the decision boundary feature matrix is defined and from the examples, one might suspect that the rank of the decision boundary feature matrix will be the intrinsic discriminant dimension, and the eigenvectors of the decision boundary feature matrix of a pattern recognition problem corresponding to non-zero eigenvalues are the required feature vectors to achieve the same classification accuracy as in the original space. In this regard we state the following two theorems which are useful in predicting the intrinsic discriminant dimension of a pattern classification problem and finding the feature vectors.

Theorem 3.2 The rank of the decision boundary feature matrix Σ_{DBFM} (Definition 3.6) of a pattern classification problem will be the intrinsic discriminant dimension (Definition 3.5) of the pattern classification problem.

Proof: Let **X** be an observation in the N-dimensional Euclidean space E^N under the hypothesis H_i: $\mathbf{X} \in \omega_i$ $\{i = 1,...,J\}$ where J is the number of classes. Let Σ_{DBFM} be the decision boundary feature matrix as defined in Definition 3.6. Suppose that

$$\text{rank}(\Sigma_{DBFM}) = M \leq N.$$

Let $\{\phi_1, \phi_2,.., \phi_M\}$ be the eigenvectors of Σ_{DBFM} corresponding to non-zero eigenvalues. Then a vector normal to the decision boundary at any point on decision boundary can be represented by a linear combination of ϕ_i, i=1,..,M. In other words, for any normal vector V_N to the decision boundary

$$V_N = \sum_{i=1}^{M} a_i \phi_i$$

Since any linearly independent set of vectors from a finite dimensional vector space can be extended to a basis for the vector space, we can expand $\{\phi_1, \phi_2,.., \phi_M\}$ to form a basis for the N-dimension Euclidean space. Let $\{\phi_1, \phi_2,.., \phi_M, \phi_{M+1},.., \phi_N\}$ be such a basis. Without loss of generality, we can assume $\{\phi_1, \phi_2,.., \phi_M, \phi_{M+1},.., \phi_N\}$ is an orthonormal basis. One can always find an orthonormal basis for a vector space using the Gram-Schmidt procedure (Cullen 1972). Since the basis is assumed to be orthonormal, it can be easily seen that the vectors $\{\phi_{M+1}, \phi_{M+2},.., \phi_N\}$, are orthogonal to any vector V_N normal to the decision boundary. This is because for i = M+1,..,N

$$\phi_i^t V_N = \phi_i^t \sum_{k=1}^{M} a_k \phi_k$$

$$= \sum_{k=1}^{M} a_k \phi_i^t \phi_k = 0 \qquad \text{since } \phi_i^t \phi_k = 0 \text{ if } i \neq k$$

Therefore, since the vectors $\{\phi_{M+1}, \phi_{M+2}, \ldots, \phi_N\}$ are orthogonal to any vector normal to the decision boundary, according to Lemma.1, the vectors $\{\phi_{M+1}, \phi_{M+2}, \ldots, \phi_N\}$ are discriminantly redundant. Therefore the number of discriminantly redundant features is N – M, and the intrinsic discriminant dimension is M which is the rank of decision boundary feature matrix Σ_{DBFM}.

Q.E.D.

It is noted that we did not make any assumption on the number of classes in proving Theorem 3.2. In other words, Theorem 3.2 holds for any number of classes. From Theorem 3.2, we can derive the following theorem which is useful to find the feature vectors needed to achieve the same classification accuracy as in the original space.

Theorem 3.3 The eigenvectors of the decision boundary feature matrix of a pattern recognition problem corresponding to non-zero eigenvalues are the feature vectors needed to achieve the same classification accuracy as in the original space for the pattern recognition problem.

Proof: In the proof of Theorem 3.2, it was shown that the eigenvectors of Σ_{DBFM} corresponding to non-zero eigenvalues are the only discriminantly informative feature vectors. Thus by retaining the eigenvectors of Σ_{DBFM} corresponding to non-zero eigenvalues, it is possible to achieve the same classification accuracy as in the original space.

Q.E.D.

3.5.5 Procedure to Find the Decision Boundary Feature Matrix

Assuming a Gaussian ML classifier is used, the decision boundary will be a quadratic surface if the covariance matrices are different. In this case, the rank of the decision boundary feature matrix will be the same as the dimension of the original space except for some special cases. However, in practice, only a small portion of the decision boundary is significant. Therefore if the decision boundary feature matrix is estimated using only the significant portion of the decision boundary or the effective decision boundary, the rank of the decision boundary feature matrix, equivalently the number of features, can be reduced substantially while achieving about the same classification accuracy.

More specifically, the significance of any portion of the decision boundary is related to how much accuracy can be achieved by utilizing that portion of the decision boundary. Consider the case of Figure 3.11 which shows the two regions which contain 99.9% of each Gaussianly distributed class, along with the decision boundary and the effective decision boundary of 99.9%. Although in this example the threshold probability, $P_{threshold}$, is set to 99.9% arbitrarily, it can be set to any value depending on the application (See Definition 3.4). If only the effective decision boundary, which is displayed in bold, is retained, it is still possible to classify 99.9% of data from class ω_1 the same as if the whole decision boundary had been used, since the effective decision boundary together with the boundary of the region which contains 99.9% of class ω_1 can divide the data of class ω_1 into two groups in the same manner as if the whole decision boundary is used; less than 0.1% of data from class ω_1 may be classified differently.

Therefore, for the case of Figure 3.11, the effective decision boundary displayed as a bold line plays a significant role in discriminating between the classes, while the part of the decision boundary displayed as a non-bold line does not contribute much in discriminating between the classes. On the other hand, other portions of the decision boundary, displayed as a dotted line, would be very rarely used.

It is noted, however, that even though only the effective decision boundary is used for feature extraction, this does not mean that the portion outside of the effective regions does not have a decision boundary. The actual decision boundary is approximated by the extension of the effective decision boundary as shown in Figure 3.11. As shall be seen, feature extraction based on the effective decision boundary instead of the complete decision boundary will result in fewer features while achieving nearly the same classification accuracy.

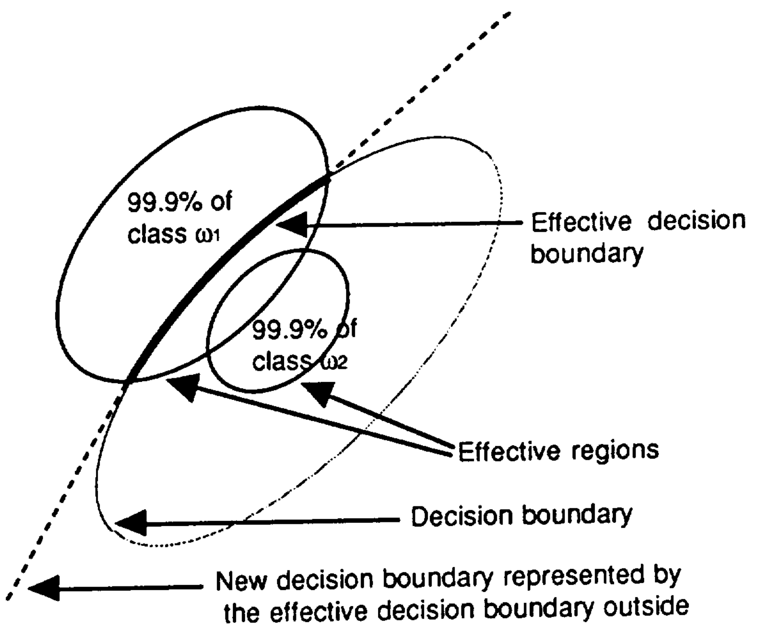
99.9% of class ω1
99.9% of class ω2
Effective decision boundary
Effective regions
Decision boundary
New decision boundary represented by the effective decision boundary outside the effective regions

Figure 3.11 An example of a decision boundary and an effective decision boundary.

Next we propose a procedure for calculating the effective decision boundary feature matrix numerically.

Numerical Procedure to Find the Effective Decision Boundary Feature Matrix
(2 pattern classes)

1. Let $\hat{M}_i$ and $\hat{\Sigma}_i$ be the estimated mean and covariance of class ω_i. Classify the training samples using full dimensionality. Apply a chi-square threshold test to the correctly classified training samples of each class and delete outliers. In other words, for class ω_i, retain **X** only if

$$(\mathbf{X} - \hat{M}_i)^t \hat{\Sigma}_i^{-1} (\mathbf{X} - \hat{M}_i) < R_{t1}$$

In the following STEPs, only correctly classified training samples which passed the chi-square threshold test will be used. Let $\{X_1, X_2, \ldots, X_{L_1}\}$ be such training samples of class ω_1 and $\{Y_1, Y_2, \ldots, Y_{L_2}\}$ be such training samples of class ω_2.

2. Apply a chi-square threshold test of class ω_1 to the samples of class ω_2 and retain $\mathbf{Y}_j$ only if

$$(\mathbf{Y}_j - \hat{\mathbf{M}}_1)^t \hat{\Sigma}_1^{-1} (\mathbf{Y}_j - \hat{\mathbf{M}}_1) < R_{t2}$$

   If the number of the samples of class ω_2 which pass the chi-square threshold test is less than L_{min} (see below), retain the L_{min} samples of class ω_2 which gives the smallest values.
3. For $\mathbf{X}_i$ of class ω_1, find the nearest sample of class ω_2 retained in STEP 2.
4. Find the point $\mathbf{P}_i$ where the straight line connecting the pair of samples found in STEP 3 meets the decision boundary.
5. Find the unit normal vector, $\mathbf{N}_i$, to the decision boundary at the point $\mathbf{P}_i$ found in STEP 4.
6. By repeating STEP 3 through STEP 5 for X_i, $i=1,..,L_1$, L_1 unit normal vectors will be calculated. From the normal vectors, calculate an estimate of the effective decision boundary feature matrix (Σ^1_{EDBFM}) from class ω_1 as follows:

$$\Sigma^1_{EDBFM} = \frac{1}{L_1} \sum_i^{L_1} \mathbf{N}_i \mathbf{N}_i^t$$

   Repeat STEP 2 through STEP 6 for class ω_2
7. Calculate an estimate of the final effective decision boundary feature matrix as follows:

$$\Sigma_{EDBFM} = \frac{1}{2} (\Sigma^1_{EDBFM} + \Sigma^2_{EDBFM})$$

The chi-square threshold test in STEP 1 is necessary to eliminate outliers. Otherwise, outliers may give a false decision boundary when classes are well separable. The chi-square threshold test to the other class in STEP 2 is necessary to concentrate on effective decision boundary (Definition 4). Otherwise, the decision boundary feature matrix may be calculated from an insignificant portion of decision boundary, resulting in ineffective features. In the experiments, L_{min} in STEP 2 is set to 5 and R_{t1} is chosen such that

$$\Pr\{\mathbf{X} \mid (\mathbf{X} - \hat{\mathbf{M}}_i)^t \hat{\Sigma}_i^{-1} (\mathbf{X} - \hat{\mathbf{M}}_i) < R_{ti}\} = 0.95,\ i=1,2, \text{ and } R_{t1} = R_{t2}$$

The threshold probability is taken as 0.95. In an ideal case assuming a Gaussian distribution, the threshold probability can be larger, i.e., 0.999. However, for real data, if the threshold probability is set too large, some outliers could be included, causing some inefficiency in calculating the decision boundary feature matrix.

Figure 3.12 shows an illustration of the proposed procedure. For each sample, the nearest sample classified as the other class is found and the two samples are connected by a straight line. Then a vector normal to the decision boundary is found at the point where the straight line connecting the two points meets the decision boundary. From these normal vectors, Σ_{EDBFM} is estimated.

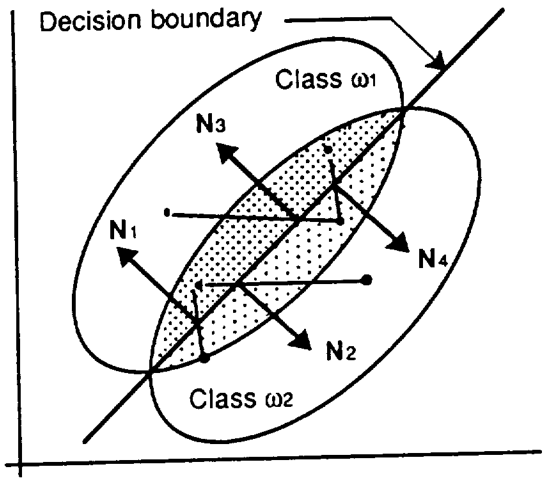

Figure 3.12 Illustration of the procedure to find the effective decision boundary feature matrix numerically.

If we assume a Gaussian distribution for each class and the Gaussian ML classifier is used, h(**X**) in equation (3.1) is given by

$$h(\mathbf{X}) = -\ln\frac{P(\mathbf{X}|\omega_1)}{P(\mathbf{X}|\omega_2)} = \ln P(\mathbf{X}|\omega_2) - \ln P(\mathbf{X}|\omega_1)$$
$$= \frac{1}{2}(\mathbf{X} - \mathbf{M}_1)^t \Sigma_1^{-1} (\mathbf{X} - \mathbf{M}_1) + \frac{1}{2}\ln|\Sigma_1| - \frac{1}{2}(\mathbf{X} - \mathbf{M}_2)^t \Sigma_2^{-1} (\mathbf{X} - \mathbf{M}_2) - \frac{1}{2}\ln|\Sigma_2|$$

The vector normal to the decision boundary at $\mathbf{X}_0$ is given by (See Appendix A)

$$N= \nabla h(\mathbf{X})|_{\mathbf{X}=\mathbf{X}_0} =(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{X} + (\Sigma_2^{-1} \mathbf{M}_2 - \Sigma_1^{-1}\mathbf{M}_1) \qquad (3.3)$$

If $\mathbf{P}_1$ and $\mathbf{P}_2$ are on different sides of decision boundary $h(\mathbf{X}) = t$ assuming that the Gaussian ML classifier is used, the point $\mathbf{X}_0$ where the line connecting $\mathbf{P}_1$ and $\mathbf{P}_2$ passes through the decision boundary is given by (Appendix A)

$$\mathbf{X}_0=u\mathbf{V}+ \mathbf{V}_0 \qquad (3.4)$$

where $\mathbf{V}_0 = \mathbf{P}_1$

$\mathbf{V} = \mathbf{P}_2 - \mathbf{P}_1$

$u =\dfrac{t - c'}{b}$ if $a = 0$,

$u =\dfrac{-b \pm \sqrt{b^2 - 4a(c' - t)}}{2a}$ and $0 \le u \le 1$ if $a \neq 0$,

$a = \frac{1}{2}\mathbf{V}^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V}$,

$b = \mathbf{V}_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V} - (\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})\mathbf{V}$,

$c' = \frac{1}{2}\mathbf{V}_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V}_0 - (\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})\mathbf{V}_0 + c$,

$c = \frac{1}{2}(\mathbf{M}^t\Sigma_1^{-1}\mathbf{M}_1 - \mathbf{M}_2^t\Sigma_2^{-1}\mathbf{M}_2) + \frac{1}{2}\ln\dfrac{|\Sigma_1|}{|\Sigma_2|}$

Equation (3.4) can be used to calculate the point on the decision boundary from two samples classified differently and equation (3.3) can be used to calculate a normal vector to the decision boundary.

3.5.6 Decision Boundary Feature Matrix for Multiclass Problem

If there are more than two classes, the total decision boundary feature matrix can be defined as the sum of the decision boundary feature matrices of each pair of classes. If prior probabilities are available, the summation can be weighted. In other words, if there are M classes, the total decision boundary feature matrix can be defined as

$$\Sigma_{DBFM} = \sum_{i}^{M} \sum_{j,\, j\neq i}^{M} P(\omega_i)P(\omega_j)\Sigma_{DBFM}^{ij} \tag{3.5}$$

where Σ_{DBFM}^{ij} is the decision boundary feature matrix between class ω_i and class ω_j and $P(\omega_i)$ is the prior probability of class ω_i if available. Otherwise let $P(\omega_i)=1/M$.

It is noted that Theorem 3.2 and Theorem 3.3 still hold for the multiclass case and the eigenvectors of the total decision boundary feature matrix corresponding to non-zero eigenvalues are the necessary feature vectors to achieve the same classification accuracy as in the original space. In practice, the total effective decision boundary feature matrix can be calculated by repeating the procedure for each pair of classes.

3.5.7 Eliminating Redundancy in Multiclass Problem

The total decision boundary feature matrix defined in equation (3.5), can be made more efficient. Consider the following example situation. Suppose Table 3.1 shows eigenvalues for the 2 pattern class problem of Table 3.6. Table 3.1 also shows proportions of the eigenvalues, classification accuracies, and normalized classification accuracies obtained by dividing the classification accuracies by the classification accuracy obtained using all features. With just one feature, the classification accuracy is 93.4% which is 97.9% of the classification accuracy obtained using all features. Thus, in this 2 class problem, if this level of accuracy is deemed adequate, just one feature is necessary to be included in calculating the total decision boundary feature matrix. The other 19 features contributes little in improving classification accuracy and can be eliminated in calculating the total decision boundary feature matrix. In addition, feature vectors from other pairs of classes will improve the classification accuracy.

Table 3.1 Eigenvalues and classification accuracies of the 2 class problem.

	Eigenvalues	Proportion of Eigenvalues (%)	Classification Accuracy (%)	Normalized Classification Accuracy (%)
1	0.994	49.6	93.4	97.9
2	0.547	27.3	94.3	98.8
3	0.167	8.3	94.4	99.0
4	0.133	6.6	95.0	99.6
5	0.066	3.3	95.1	99.7
6	0.041	2.1	94.9	99.5
7	0.020	1.0	94.9	99.5
8	0.012	0.6	94.8	99.4
9	0.008	0.4	95.0	99.6
10	0.007	0.3	95.3	99.9
11	0.005	0.2	95.3	99.9
12	0.001	0.1	95.7	100.3
13	0.001	0.0	95.5	100.1
14	0.001	0.0	95.4	100.0
15	0.000	0.0	95.3	99.9
16	0.000	0.0	95.6	100.2
17	0.000	0.0	95.5	100.1
18	0.000	0.0	95.5	100.1
19	0.000	0.0	95.4	100.0
20	0.000	0.0	95.4	100.0

To eliminate such redundancy in multiclass problems, we define the decision boundary feature matrix of P_t ($\Sigma_{DBFM(Pt)}$) as follows:

Definition 3.8 Let L_t be the number of eigenvectors corresponding to largest eigenvalues needed to obtain P_t of the classification accuracy obtained using all features. Then the decision boundary feature matrix of P_t ($\Sigma_{DBFM(Pt)}$) is defined as

$$\Sigma_{DBFM(Pt)} = \sum_{i=1}^{L_t} \lambda_i \varphi_i \varphi_i^t$$

where λ_i and φ_i are eigenvalues and eigenvectors of the decision boundary feature matrix.

The total decision boundary feature matrix of P_t in a multiclass problem can be defined as

$$\Sigma_{DBFM(Pt)} = \sum_{i=1}^{M} \sum_{j=1\ j\neq i}^{M} P(\omega_i)P(\omega_j)\Sigma^{ij}_{DBFM(Pt)}$$

where $\Sigma^{ij}_{DBFM(Pt)}$ is the decision boundary feature matrix of P_t between class ω_i and class ω_j and $P(\omega_i)$ is the prior probability of class ω_i if available. Otherwise let $P(\omega_i)=1/M$.

From Definition 3.8, we can calculate the decision boundary feature matrix of 0.95 of Table 3.1 as follows:

The classification accuracy using full dimensionality (assume it is 20) is 95.4%. The number of features needed to achieve a classification accuracy of 92.5%(=95.4*0.95) is 1. Therefore, the decision boundary feature matrix of 0.95 of Table 3.1 is given by

$$\Sigma_{DBFM(0.95)} = \sum_{i=1}^{1} \lambda_i \varphi_i \varphi_i^t = \lambda_1 \varphi_1 \varphi_1^t$$

where λ_i's are eigenvalues of Σ_{DBFM} sorted in descending order and φ_i's are the corresponding eigenvectors.

Figure 3.13 shows a performance comparison for various values of P_t. By eliminating feature vectors which contribute little to improvement of the classification accuracy, it is possible to improve classification accuracy up to 1.5% in this example. The experiment showed P_t between 0.95 and 0.97 would be reasonable.

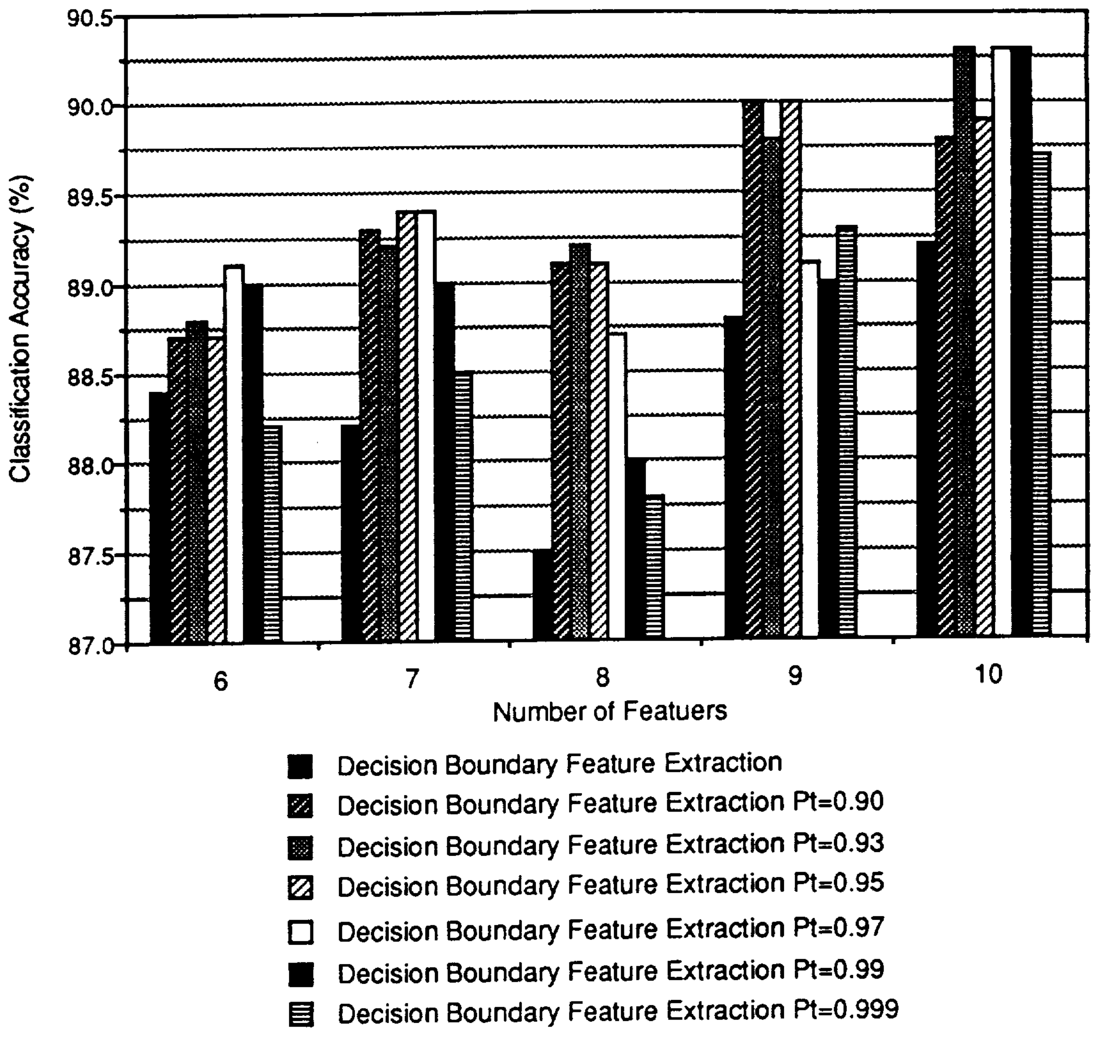

Figure 3.13 Performance comparison for various P_ts.

3.6 Experiments and Results

3.6.1 An experiment with generated data

To evaluate closely how the proposed algorithm performs under various circumstances, tests are conducted on data generated with given statistics assuming Gaussian distributions. In all examples, a Gaussian ML classifier is used and the same data are used for training and test. In each example, the Foley & Sammon method (Foley and Sammon 1975) and the Fukunaga & Koontz method (Fukunaga and Koontz 1970) are discussed. In particular, classification accuracies of the decision boundary feature extraction method and the Foley & Sammon method are compared.

Example 3.3 In this example, data are generated for the following statistics.

$$M_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \qquad M_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$P(\omega_1) = P(\omega_2) = 0.5$$

300 samples are generated for each class and all samples are used for training and test. Since the covariance matrices are the same, it can be easily seen that the decision boundary will be a straight line and just one feature is needed to achieve the same classification accuracy as in the original space. The eigenvalues λ_i and the eigenvectors ϕ_i of Σ_{EDBFM} are calculated as follows:

$$\lambda_1 = 0.99995\ ,\ \lambda_2 = 0.00005 \qquad \phi_1 = \begin{bmatrix} 0.71 \\ -0.70 \end{bmatrix}, \quad \phi_2 = \begin{bmatrix} 0.70 \\ 0.71 \end{bmatrix}$$

Since one eigenvalue is significantly larger than the other, it can be said that the rank of Σ_{EDBFM} is 1. That means only one feature is needed to achieve the same classification accuracy as in the original space. Considering the statistics of the two classes, the rank of Σ_{EDBFM} gives the correct number of features to achieve the same classification accuracy as in the original space. Figure 3.14 shows the distribution of the generated data and the decision boundary found by the proposed procedure. Since class mean differences are dominant in this example, the Foley & Sammon method will also work well. However, the Fukunaga & Koontz method will fail to find the correct feature vector. Table 3.2 shows the classification accuracies of Decision Boundary Feature Extraction and Foley & Sammon method. With two features, the classification accuracy is 95.8% and both methods achieve the same accuracy with just one feature.

Table 3.2 Classification accuracies of Decision Boundary Feature Extraction and the Foley & Sammon method of Example 3.3.

No. Features	Decision Boundary Feature Extraction	Foley & Sammon Method
1	95.8 (%)	95.8 (%)
2	95.8 (%)	95.8 (%)

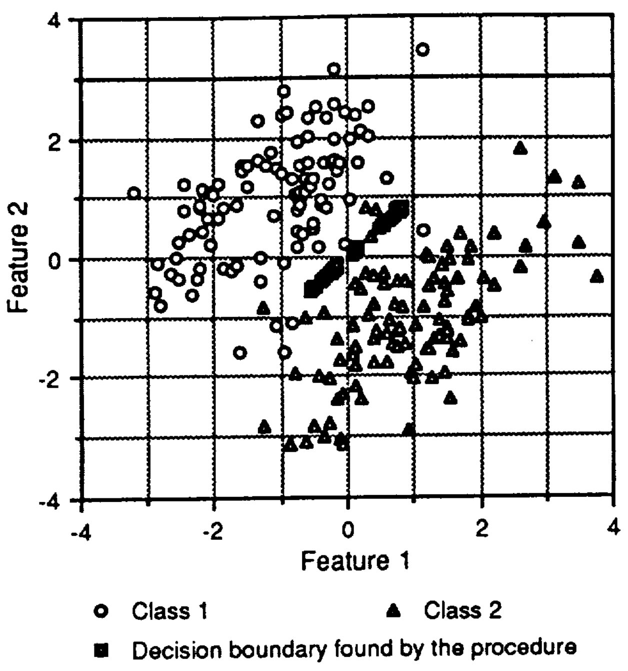

Figure 3.14 The distribution of data for the two classes in Example 3.3. The decision boundary, found by the proposed algorithm, is also shown.

Example 3.4 In this example, data are generated with the following statistics.

$$M_1 = \begin{bmatrix} 0.01 \\ 0 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \qquad M_2 = \begin{bmatrix} -0.01 \\ 0 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

$$P(\omega_1) = P(\omega_2) = 0.5$$

300 samples are generated for each class and all samples are used for training and test. In this case, there is almost no difference in the mean vectors and there is no correlation between the features for each class. The variance of feature 1 of class ω_1 is equal to that of class ω_2 while the variance of feature 2 of class ω_1 is larger than that of class ω_2. Thus the decision boundary will consist of hyperbolas, and two features are needed to achieve the same classification accuracy as in the original space. However, the effective decision boundary could be approximated by a straight line without introducing significant error. Figure 3.15 shows the distribution of the generated data and the decision boundary obtained by the proposed procedure. The eigenvalues λ_i and the eigenvectors ϕ_i of Σ_{EDBFM} are calculated as follows:

$$\lambda_1 = 0.92421, \lambda_2 = 0.07579 \qquad \phi_1 = \begin{bmatrix} 0.06 \\ 1.00 \end{bmatrix}, \qquad \phi_2 = \begin{bmatrix} -1.00 \\ 0.06 \end{bmatrix}$$

Since the rank of Σ_{EDBFM} is 2, two features are required to achieve the same classification accuracy as in the original space. However, λ_2 is considerably smaller than λ_1, even though λ_2 is not negligible. Therefore, nearly the same classification accuracy could be achieved with just one feature.

Since there is a very small difference in the mean vectors in this example, the Foley & Sammon method will fail to find the correct feature vector. On the other hand, the Fukunaga & Koontz method will find the correct feature vector. Table 3.3 shows classification accuracies. Decision Boundary Feature Extraction achieves the same accuracy with one feature as can be obtained with two features while the Foley & Sammon method fails to find the right feature in this example.

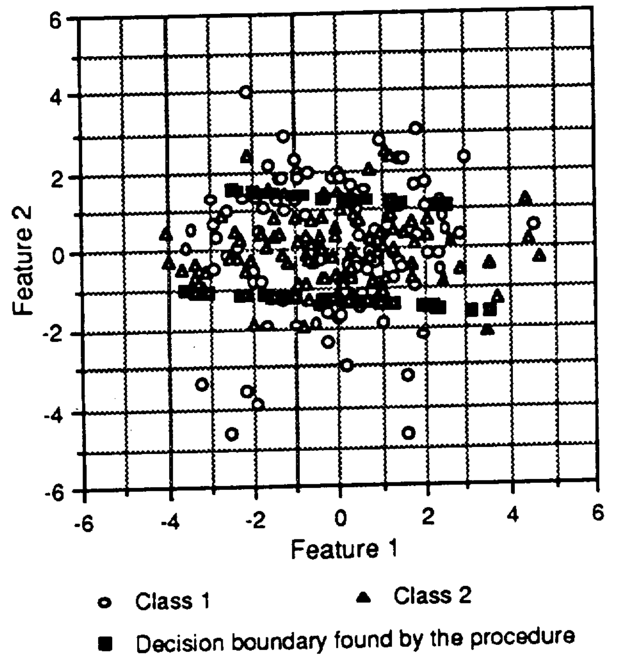

Figure 3.15 Distribution of data from the two classes in Example 3.4. The decision boundary found by the proposed algorithm is also shown.

Table 3.3 Classification accuracies of Decision Boundary Feature Extraction and the Foley & Sammon method of Example 3.4.

No. Features	Decision Boundary Feature Extraction	Foley & Sammon Method
1	61.0 (%)	52.5 (%)
2	61.0 (%)	61.0 (%)

Example 3.5 In this example, we generate data for the following statistics.

$$\mathbf{M}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{M}_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P(\omega_1) = P(\omega_1) = 0.5$$

200 samples are generated for each class and all samples are used for training and test. In this case, there is no difference in the mean vectors and there are variance differences in only two features. It can be seen that the decision boundary will be a right circular cylindrical surface of infinite height and just two features are needed to achieve the same classification accuracy as in the original space. Eigenvalues λ_i and eigenvectors ϕ_i of Σ_{EDBFM} are calculated as follows:

$$\lambda_1 = 0.57581\ , \lambda_2 = 0.42032, \lambda_3 = 0.00387$$

$$\phi_1 = \begin{bmatrix} 0.86 \\ -050 \\ 001 \end{bmatrix}, \phi_2 = \begin{bmatrix} 0.49 \\ 0.84 \\ 0.21 \end{bmatrix}, \phi_3 = \begin{bmatrix} -0.21 \\ -0.18 \\ 0.98 \end{bmatrix}$$

$$\text{Rank}(\Sigma_{EDBFM}) \approx 2$$

Since the rank of Σ_{EDBFM} is roughly 2, it can be said that two features are required to achieve the same classification accuracy as in the original space, which agrees with the data. Since there is no difference in the mean vectors in this example, the Foley & Sammon method will fail to find the correct feature vectors. On the other hand, the Fukunaga & Koontz method will find the correct feature vector. Table 3.4 shows the classification accuracies. Decision Boundary Feature Extraction finds the two effective feature vectors, achieving the same classification accuracy as in the original space.

Table 3.4 Classification accuracies of Decision Boundary Feature Extraction and the Foley & Sammon method of Example 5.

No. Features	Decision Boundary Feature Extraction	Foley & Sammon Method
1	65.0 (%)	62.3 (%)
2	70.0 (%)	60.5 (%)
3	70.0 (%)	70.0 (%)

From the experiments with generated data for given statistics, it is noted that the proposed feature extraction algorithm based on the decision boundary performs well even if there is no mean difference (Examples 3.4, 3.5) or no covariance difference (Example 3.3) without any deterioration. On the other hand, the Foley & Sammon method fails if there is no mean difference (Examples 3.4, 3.5) and the Fukunaga & Koontz method would fail if there is no covariance difference (Example 3.3) or significant mean difference (Foley and Sammon 1975). In Chapter 5, the decision boundary feature extraction algorithm is applied to a 3-class problem (generated data).

3.6.2 Experiments with real data

3.6.2.1 FSS Data and Preprocessing

In the following experiments, tests are conducted using multispectral data which was collected as a part of the LACIE remote sensing program (Biehl et al. 1982) and major parameters are shown in Table 3.5.

Table 3.5 Parameters of Field Spectrometer System.

Number of Bands	60
Spectral Coverage	0.4 - 2.4 μm
Altitude	60 m
IFOV(ground)	25 m

If estimation of statistics is not accurate, using more features does not necessarily increase classification accuracy. The so-called Hughes phenomenon occurs in practice when the number of training samples is not enough for the number of features (Swain and Davis 1978). Figure 3.16 shows a graph of classification accuracy vs. number of features. There are 6 classes and Table 8.1 in Chapter 8 provides information on the 6 classes. The original 60 dimensional data are reduced to different numbers of feature sets using a simple band combination procedure which will be referred as Uniform Feature Design. For example, if the number of features is to be reduced from 60 to 30, every two consecutive bands are combined to form a new feature. In other words, the i-th feature of a new feature set is given by

$$Y_i = X_{2*i-1} + X_{2*i}$$

Where the number of features desired is not evenly divisible into 60, the nearest integer number of bands is used. For example, for 9 features, the first 6 original bands were combined to create the first feature, then the next 7 bands were combined to create the next feature, and so on.

In the test, 100 training samples are used to estimate the statistics and the rest are used for test data. As can be seen, the classification accuracy peaked at about 29 features. After 29 features, adding more features decreases the classification accuracy. In fact, the classification accuracy is saturated at about 17-20 features. As a result, in the following experiments using the FSS data, the original 60 dimensional data are reduced to 17--20 dimensional data using Uniform Feature Design. Then various feature extraction/selection methods are applied to the reduced data set.

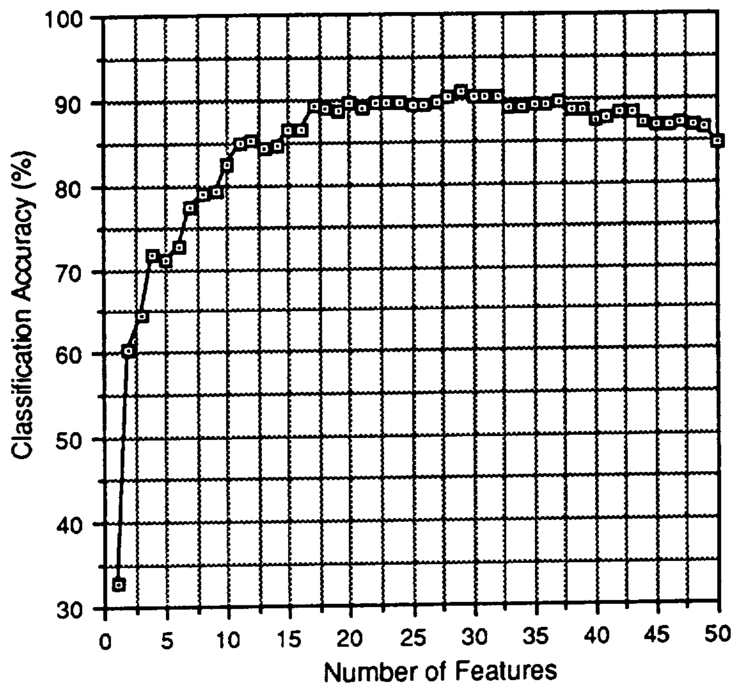

Figure 3.16 Classification accuracy vs. number of features.

3.6.2.2 Experiments and Results

Along with the proposed Decision Boundary Feature Extraction, five other feature selection/extraction algorithms, Uniform Feature Design, Principal Component Analysis (the Karhunen-Loeve transformation) (Richards 1986), Canonical Analysis (Richards 1986), feature selection using a statistical distance measure, and the Foley & Sammon method (Foley and Sammon 1975) are tested to evaluate and compare the performance of the proposed algorithm. In the feature selection using a statistical distance measure, Bhattacharyya distance (Fukunaga 1990) is used. Feature selection using the statistical distance measure will be referred as Statistical Separability. The Foley & Sammon method is based on the generalized Fisher criterion (Foley and Sammon 1975). For a two class problem, the Foley & Sammon method is used for comparison. If there are more than 2 classes, Canonical Analysis is used for comparison.

In the following test, two classes are chosen from the data collected at Finney Co. KS. in May 3, 1977. Table 3.6 shows the number of samples in each of the two classes. In this test, the covariance matrices and mean vectors are estimated using 400 randomly chosen samples from each class and the rest of the data are used for test. Figure 3.17 shows the mean graph of the two classes. There is a relatively large difference in the mean vectors between the two classes.

Table 3.6 Class description of data collected at Finney Co. KS.

SPECIES	No. of Sample
WINTER WHEAT	691
UNKNOWN CROPS	619

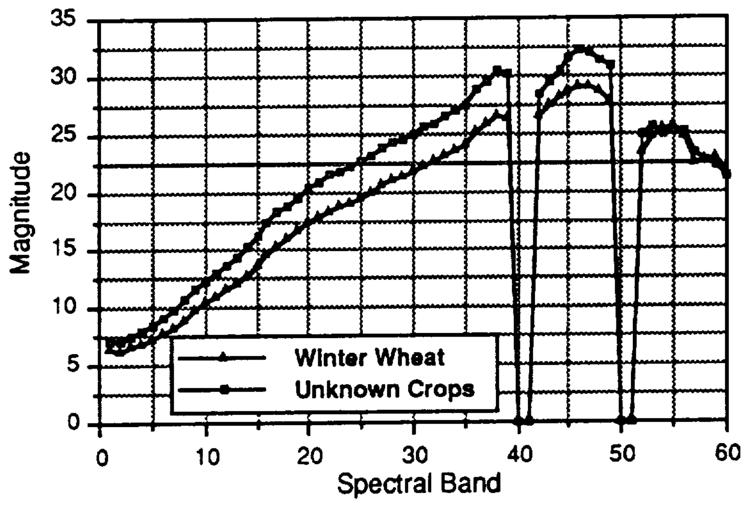

Figure 3.17 Mean graph of the two classes of Table 3.6.

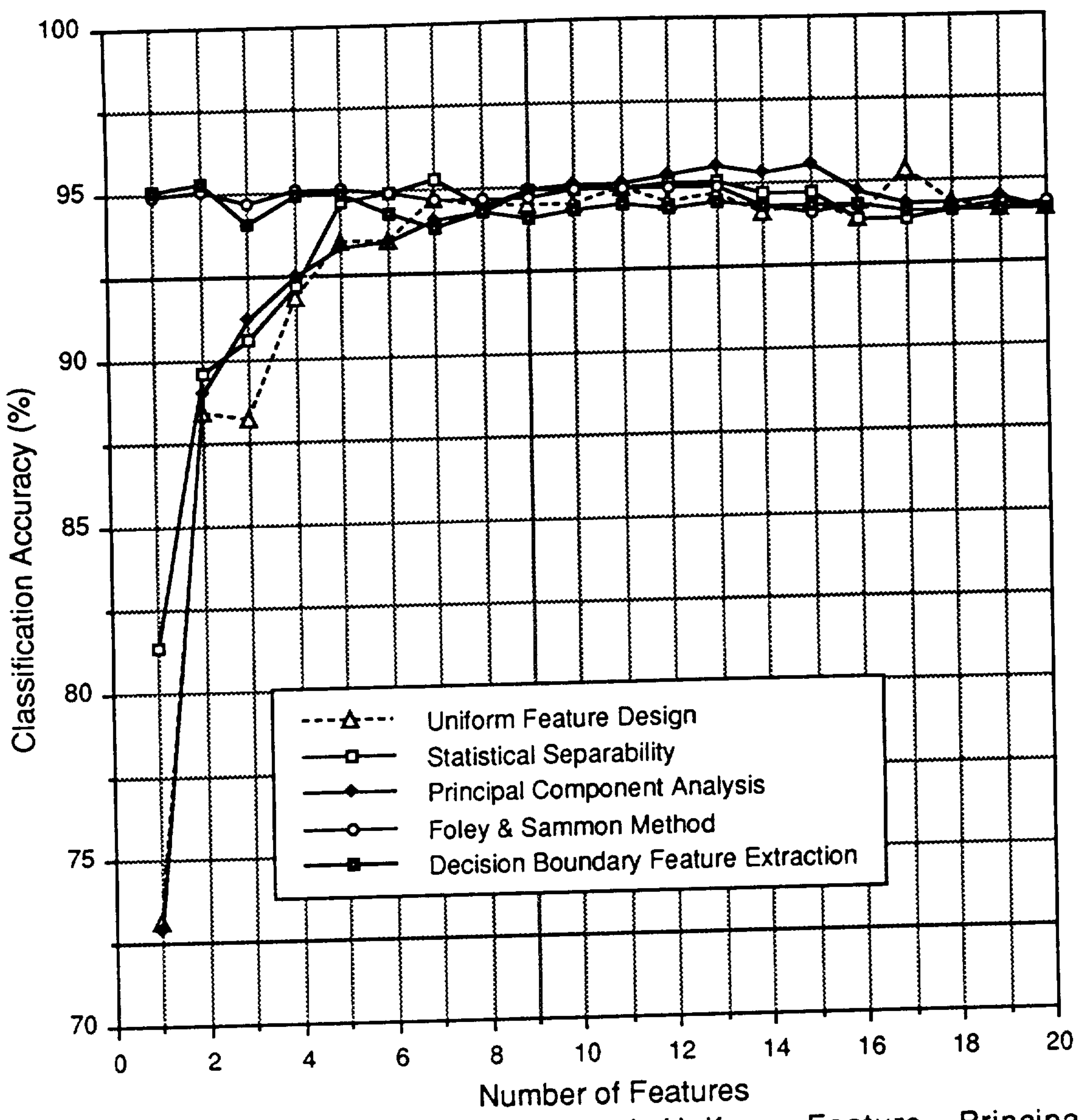

Figure 3.18 Performance comparison of Uniform Feature, Principal Component Analysis, the Foley & Sammon method, Statistical Separability, and Decision Boundary Feature Extraction.

Figure 3.18 shows the performance comparison of test data of the 5 feature selection/extraction algorithms for different numbers of features. With 20 features, the classification accuracy is about 94.1%. Decision Boundary Feature Extraction and the Foley & Sammon method achieve approximately the maximum classification accuracy with just one feature while the other feature selection/extraction algorithms need 7-8 features to achieve about the same classification accuracy.

Table 3.7 shows the eigenvalues of the decision boundary feature matrix along with proportions and accumulations. The eigenvalues are sorted in the decreasing order. The classification accuracies obtained using the corresponding eigenvectors are also shown along with the normalized classification accuracies obtained by dividing the classification accuracies by the classification accuracy obtained using all features. The rank of the decision boundary feature matrix(Σ_{DBFM}) must be decided. Although it is relatively easy to decide the rank for low dimensional generated data, it becomes less obvious for high dimensional real data. One may add eigenvalues until the accumulation exceeds 95% of the total sum and set that number of the eigenvalues as the rank of the Σ_{DBFM}. Defined in this way, the rank of the Σ_{DBFM} would be 5. Alternatively, one may retain the eigenvalues greater than one tenth of the largest eigenvalue. In this way, the rank of the Σ_{DBFM} would be 4. We will discuss more about this problem later.

Table 3.7 Eigenvalues of the Decision Boundary Feature Matrix of the 2 classes of Table 3.6 along with proportions and accumulations. Ev.:Eigenvalue, Pro. Ev.:Proportion of Eigenvalue, Acc. Ev.: Accumulation of Eigenvalues, Cl. Ac.: Classification Accuracy, N. Cl. Ac.:Normalized Classification Accuracy.

	Ev.	Pro. Ev. (%)	Acc. Ev. (%)	Cl. Ac. (%)	N. Cl. Ac. (%)
1	0.994	49.6	49.6	93.4	97.9
2	0.547	27.3	77.0	94.3	98.8
3	0.167	8.3	85.3	94.4	99.0
4	0.133	6.6	91.9	95.0	99.6
5	0.066	3.3	95.2	95.1	99.7
6	0.041	2.1	97.3	94.9	99.5
7	0.020	1.0	98.3	94.9	99.5
8	0.012	0.6	98.8	94.8	99.4
9	0.008	0.4	99.2	95.0	99.6
10	0.007	0.3	99.6	95.3	99.9
11	0.005	0.2	99.8	95.3	99.9
12	0.001	0.1	99.9	95.7	100.3
13	0.001	0.0	99.9	95.5	100.1
14	0.001	0.0	100.0	95.4	100.0
15	0.000	0.0	100.0	95.3	99.9
16	0.000	0.0	100.0	95.6	100.2
17	0.000	0.0	100.0	95.5	100.1
18	0.000	0.0	100.0	95.5	100.1
19	0.000	0.0	100.0	95.4	100.0
20	0.000	0.0	100.0	95.4	100.0

In the following test, two classes are chosen from the data collected at Hand Co. SD. on May 15, 1978. Table 3.8 shows the number of samples in each of the two classes. Figure 3.19 shows the mean graph of the two classes. As can be seen, the mean differences are relatively small. In this test, all data are used for training and test since the number of available samples is very limited.

Table 3.8 Class description of data collected at Hand Co. SD.

SPECIES	No. of Sample
WINTER WHEAT	223
SPRING WHEAT	474

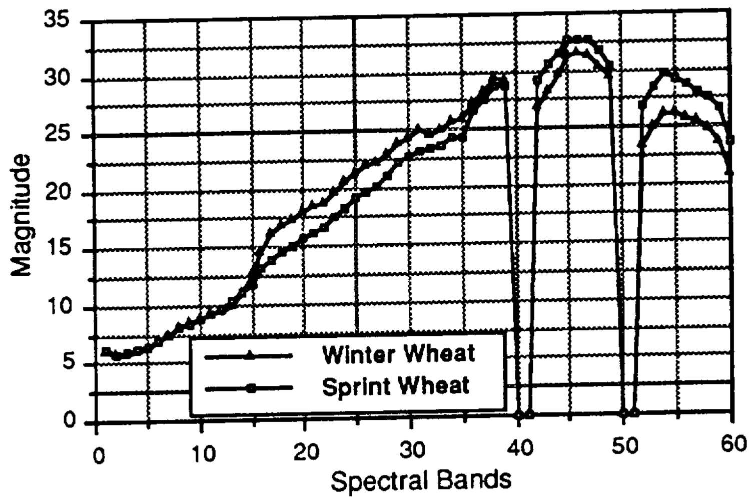

Figure 3.19 Mean graph of the two classes of Table 3.8.

Figure 3.20 show the performance comparison of the 5 feature selection/extraction algorithms for different numbers of features. With 20 features, the classification accuracy is 91.1%. In this case, the Foley & Sammon method performs less well due to the small class mean difference. Statistical Separability performs similarly. However, Decision Boundary Feature Extraction out-performs all other methods. Decision Boundary Feature Extraction achieves approximately 90% classification accuracy with 9 features while the other feature selection/extraction algorithms need 15-18 features to achieve 90% classification accuracy.

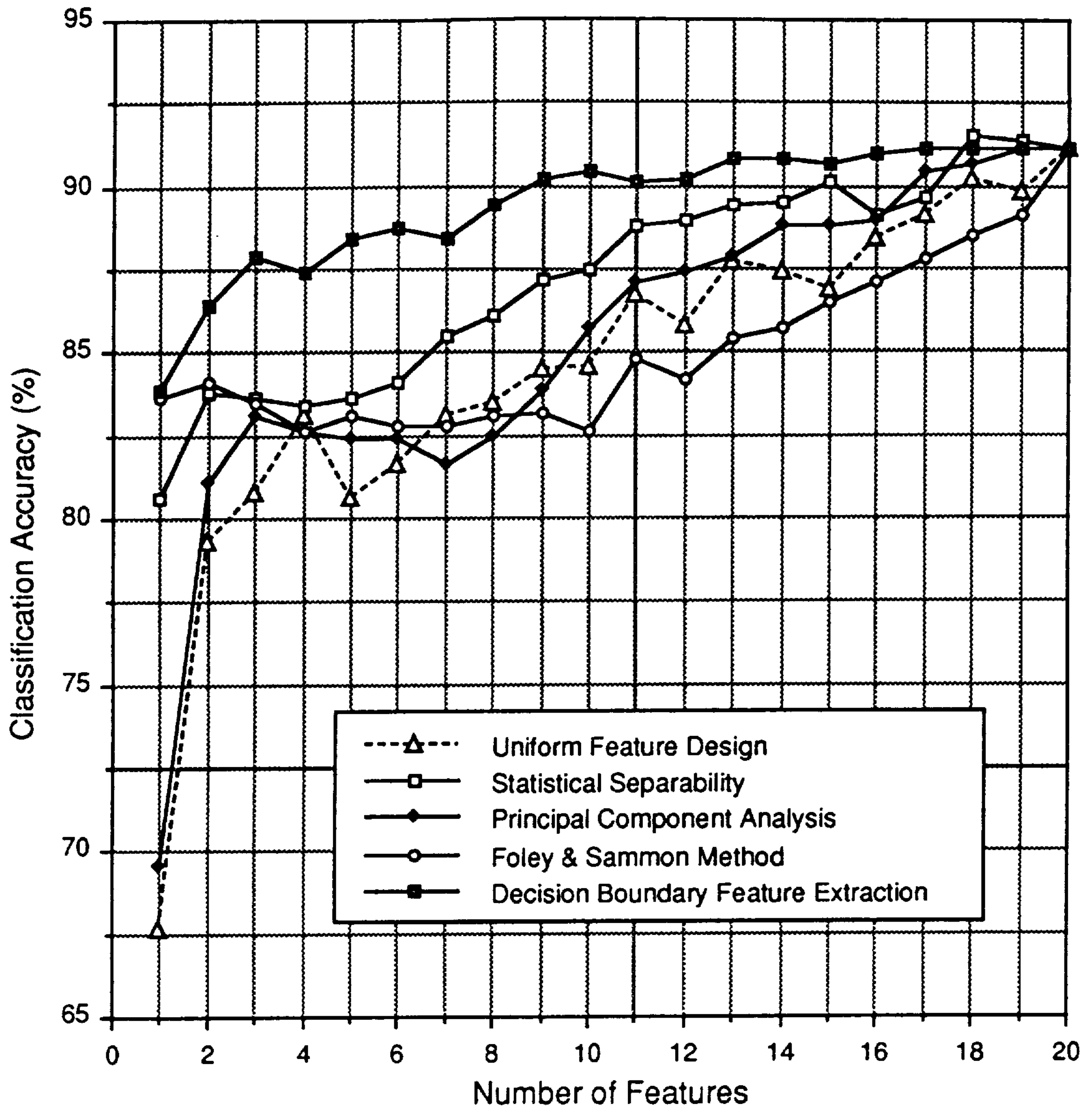

Figure 3.20 Performance comparison of Uniform Feature, Principal Component Analysis, Canonical Analysis, Statistical Separability, and Decision Boundary Feature Extraction.

In the following test, 4 classes are chosen from the FSS data. Table 3.9 provides data on the 4 classes. Figure 3.21 shows the mean graph of the 4 classes. In this test, 300 randomly selected samples are used for training and the rest are used for test.

Table 3.9 Class description.

SPECIES	DATE	No. of Samples
Winter Wheat	May 3, 1977	657
Unknown Crops	May 3, 1977	678
Winter Wheat	March 8, 1977	691
Unknown Crops	March 8, 1977	619

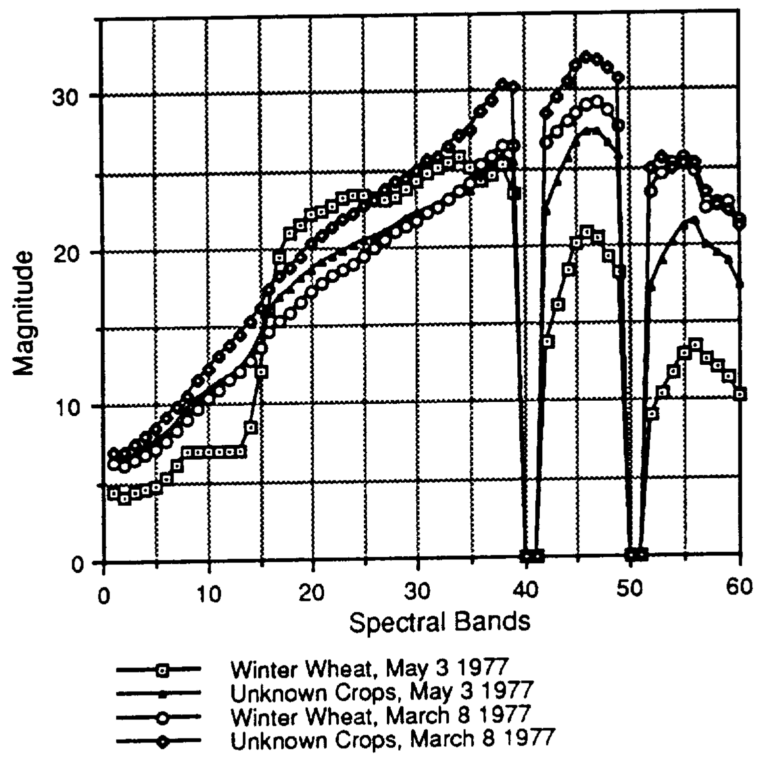

Figure 3.21 Mean graph of the two classes of Table 3.9.

Figure 3.22 shows the performance comparison of the 5 feature selection/extraction algorithms for different numbers of features. Decision Boundary Feature Extraction achieves approximately 90% classification accuracy with 3 features while Canonical Analysis achieves about 87.5% classification accuracy with 3 features. On the other hand, Statistical Separability achieves about 87.5% with 5 features. Both Uniform Feature Design and Principal Component Analysis perform poorly.

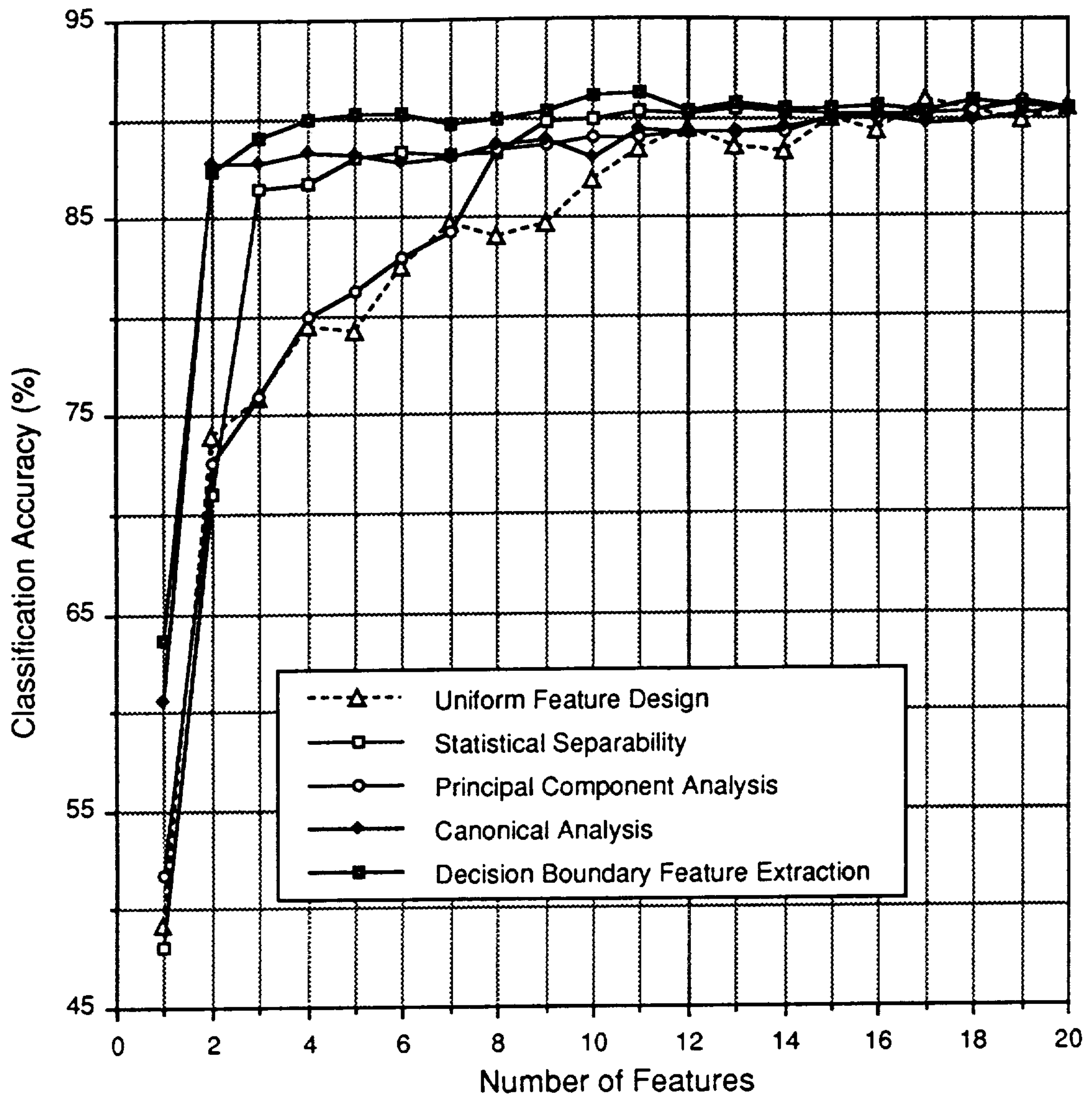

Figure 3.22 Performance comparison of Uniform Feature, Principal Component Analysis, Canonical Analysis, Statistical Separability, and Decision Boundary Feature Extraction.

In the following test, 4 classes are chosen from the data collected at Hand Co. SD. on May 15, 1978. Table 3.10 shows the number of samples in each of the 4 classes. Figure 3.23 shows the mean graph of the 4 classes. As can be seen, the mean difference is relatively small among some classes. In this test, all data are used for training and test.

Table 3.10 Class description.

Species	Date	No. of Samples
Winter Wheat	May 15, 1978	223
Native Grass Pas	May 15, 1978	196
Oats	May 15, 1978	163
Unknown Crops	May 15, 1978	253

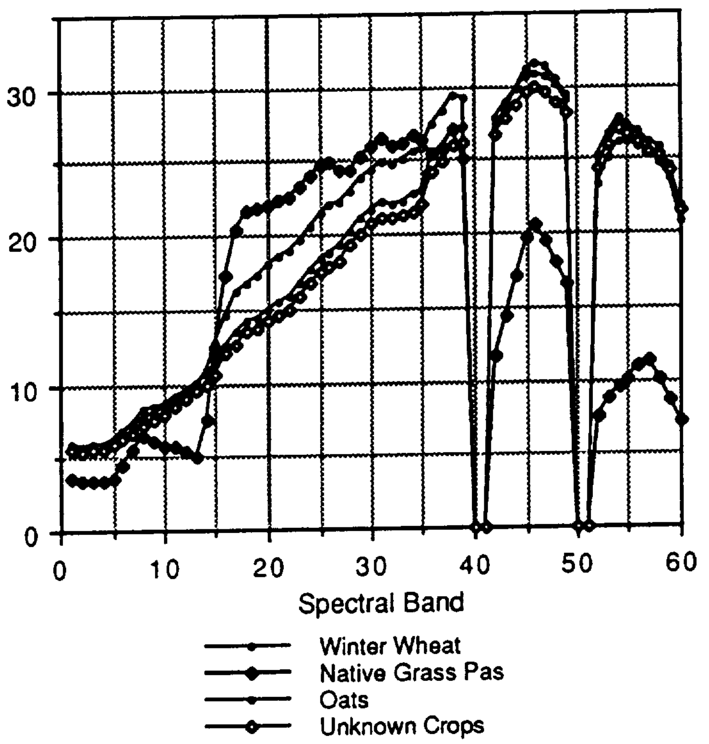

Figure 3.23 Mean graph of the two classes of Table 3.10.

Figure 3.24 shows the performance comparison of the 5 feature selection/extraction algorithms for different numbers of features. The classification accuracy using 20 features is about 88%. In this case, Canonical Analysis performs less well since class mean differences are relatively small. The performance of Decision Boundary Feature Extraction is much better than those of the other methods. Decision Boundary Feature Extraction achieves approximately 87.5% classification accuracy with 11 features while the other methods need 17-20 features to achieve about the same classification accuracies.

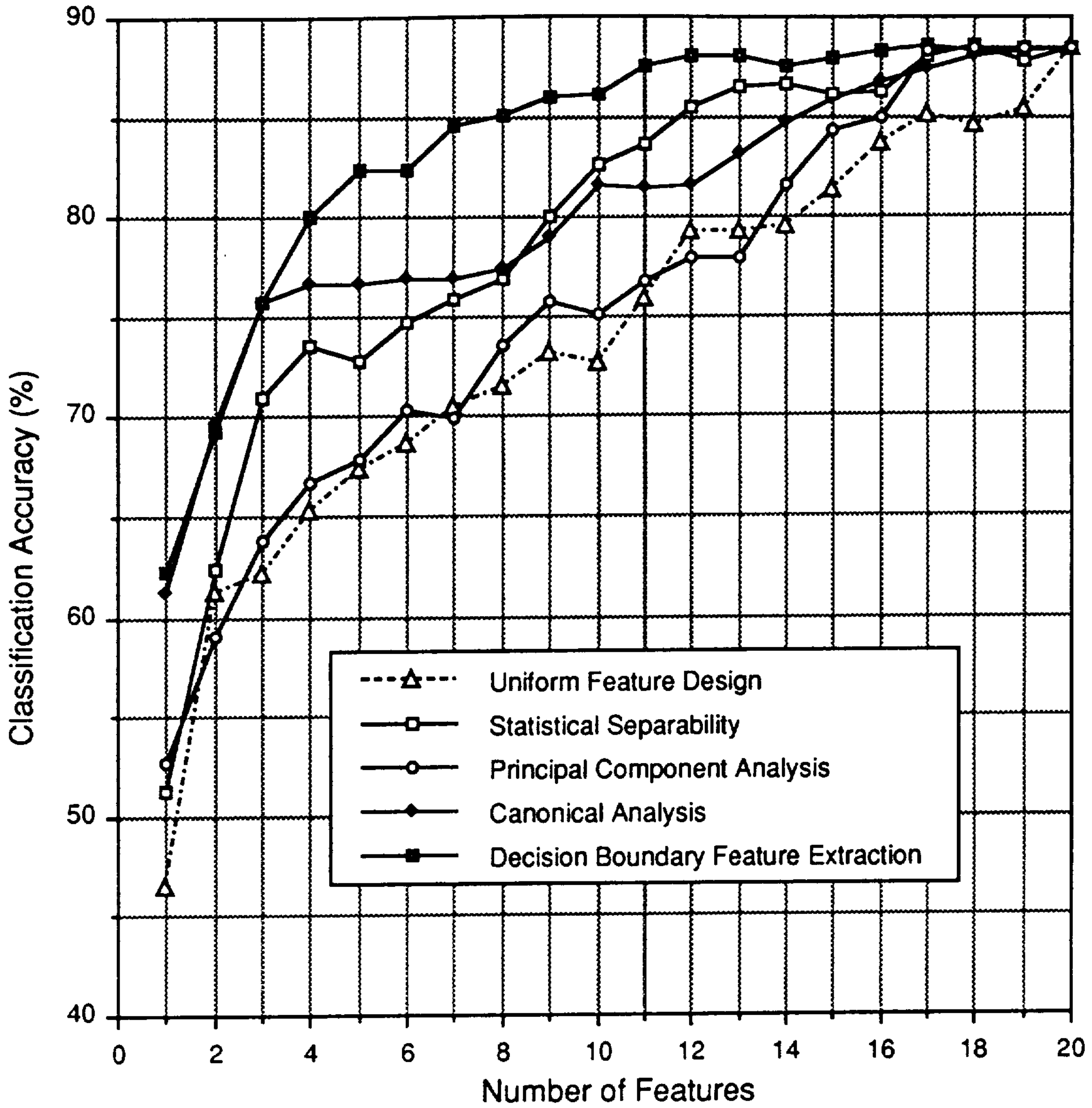

Figure 3.24 Performance comparison of Uniform Feature Design, Principal Component Analysis, Canonical Analysis, Statistical Separability, and Decision Boundary Feature Extraction.

In the following test, 4 classes are chosen from the data collected at Hand Co. SD. on August 16, 1978. Table 3.11 shows the number of samples in each of the 4 classes. Figure 3.25 shows the mean graph of the 4 classes. In this test, 100 randomly selected samples are used for training and the rest are used for test.

Table 3.11 Class description.

SPECIES	DATE	No. of Samples
Other Crops	August 16, 1978	199
Native Grass Pas	August 16, 1978	212
Oats	August 16, 1978	165
Summer Fallow	August 16, 1978	216

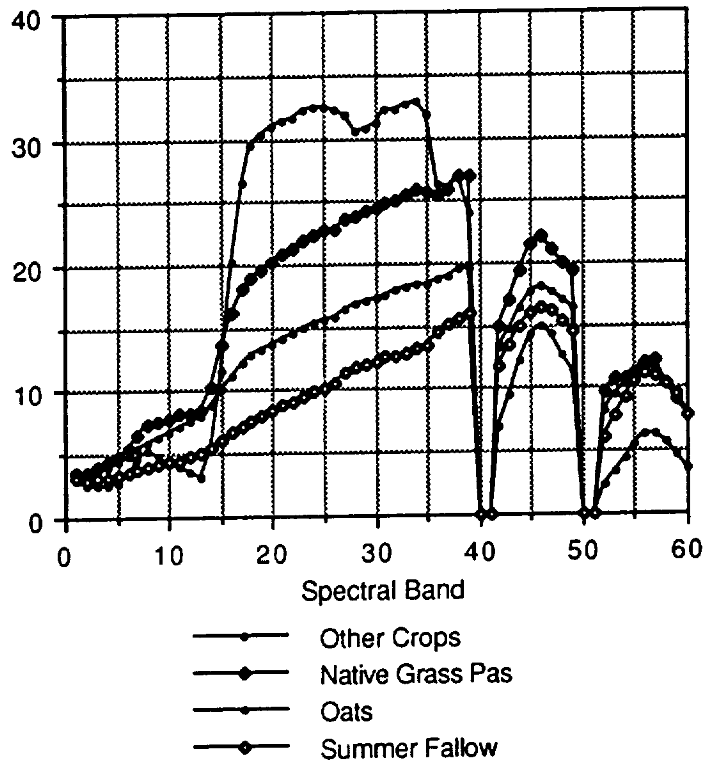

Figure 3.25 Mean graph of the two classes of Table 3.11.

Figure 3.26 show the performance comparison of the 5 feature selection/extraction algorithms for different numbers of features. Decision Boundary Feature Extraction achieves 95% classification accuracy with 4 features while the classification accuracy of Canonical Analysis with 3 features is 93%. The performances of Uniform Feature Design and Principal Component Analysis are poor compared with the other methods.

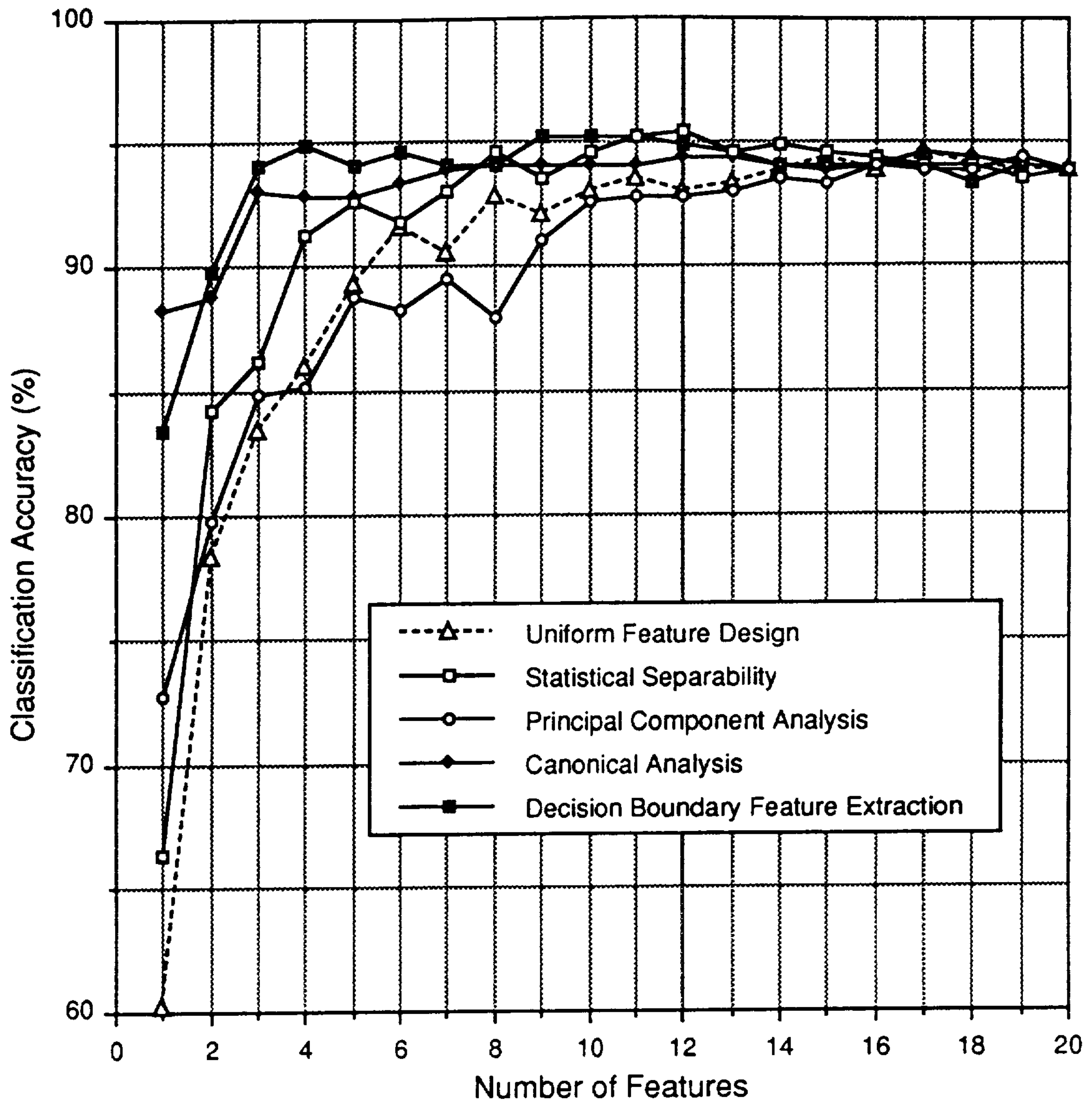

Figure 3.26 Performance comparison of Uniform Feature, Principal Component Analysis, Canonical Analysis, Statistical Separability, and Decision Boundary Feature Extraction.

In the following test, 6 are classes chosen from the FSS data. Table 3.12 provides information on the 6 classes. Figure 3.27 shows the mean graph of the 6 classes. In this test, 300 samples are used for training and the rest are used for test.

Table 3.12 Class description of the multi-temporal 6 classes.

Date	Location	Species	No. Sample
770308	Finney CO. KS.	Winter Wheat	691
770626	Finney CO. KS.	Winter Wheat	677
771018	Hand CO. SD.	Winter Wheat	662
770503	Finney CO. KS.	Winter Wheat	658
770626	Finney CO. KS.	Summer Fallow	643
780726	Hand CO. SD.	Spring Wheat	518

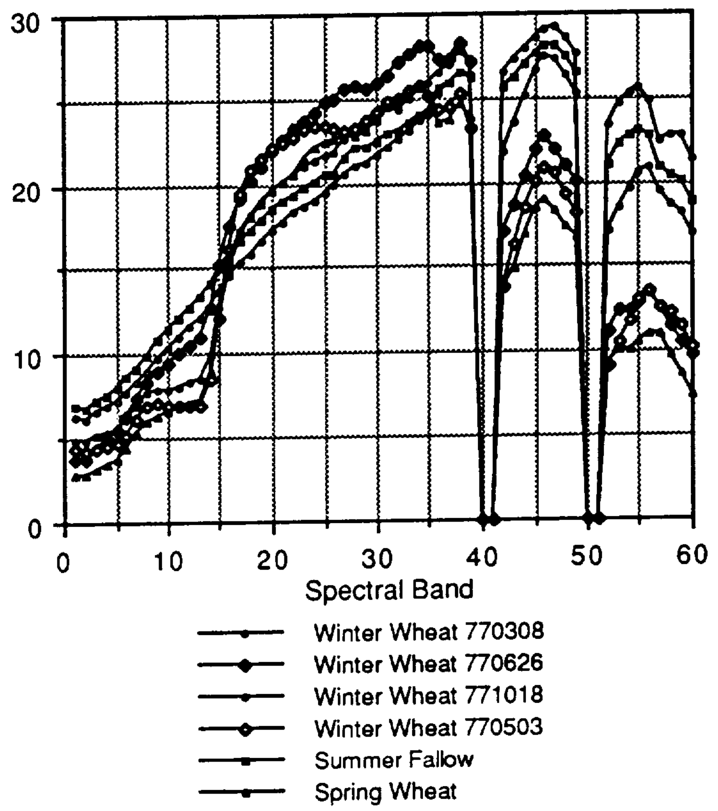

Figure 3.27 Mean graph of the two classes of Table 3.12.

Figure 3.28 shows the performance comparison of the 5 feature selection/extraction algorithms for different numbers of features. The classification accuracy using all features is 96.2%. Decision Boundary Feature Extraction achieves 94.2% classification accuracy with 5 features while the classification accuracy of Canonical Analysis with 5 features is 92.2%. Statistical Separability needs 11 features to achieve 94.2%. The performances of Uniform Feature Design and Principal Component Analysis are poor.

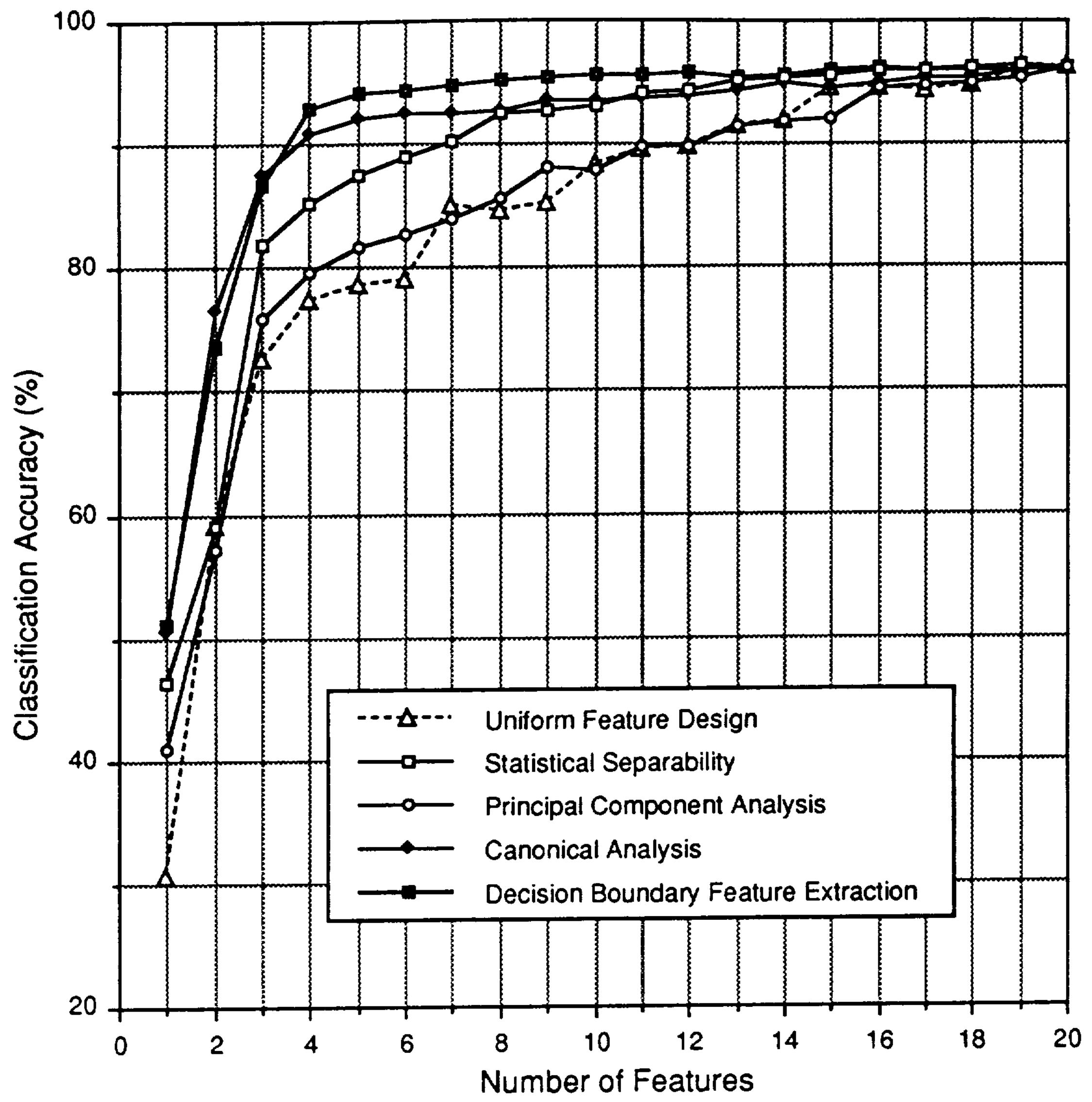

Figure 3.28 Performance comparison of Uniform Feature, Principal Component Analysis, Canonical Analysis, Statistical Separability, and Decision Boundary Feature Extraction.

In the following test, 12 classes are chosen from the FSS data. Table 3.13 shows the number of samples in each of the 12 classes. The data is multi-temporal.

Table 3.13 Class description of the multi-temporal 12 classes.

Date	Location	Species	No. Sample
770308	Finney CO. KS.	Winter Wheat	691
770626	Finney CO. KS.	Winter Wheat	677
771018	Hand CO. SD.	Winter Wheat	662
770503	Finney CO. KS.	Winter Wheat	658
770626	Finney CO. KS.	Summer Fallow	643
780726	Hand CO. SD.	Spring Wheat	518
780602	Hand CO. SD.	Spring Wheat	517
780515	Hand CO. SD.	Spring Wheat	474
780921	Hand CO. SD.	Spring Wheat	469
780816	Hand CO. SD.	Spring Wheat	464
780709	Hand CO. SD.	Spring Wheat	454
781026	Hand CO. SD.	Spring Wheat	441

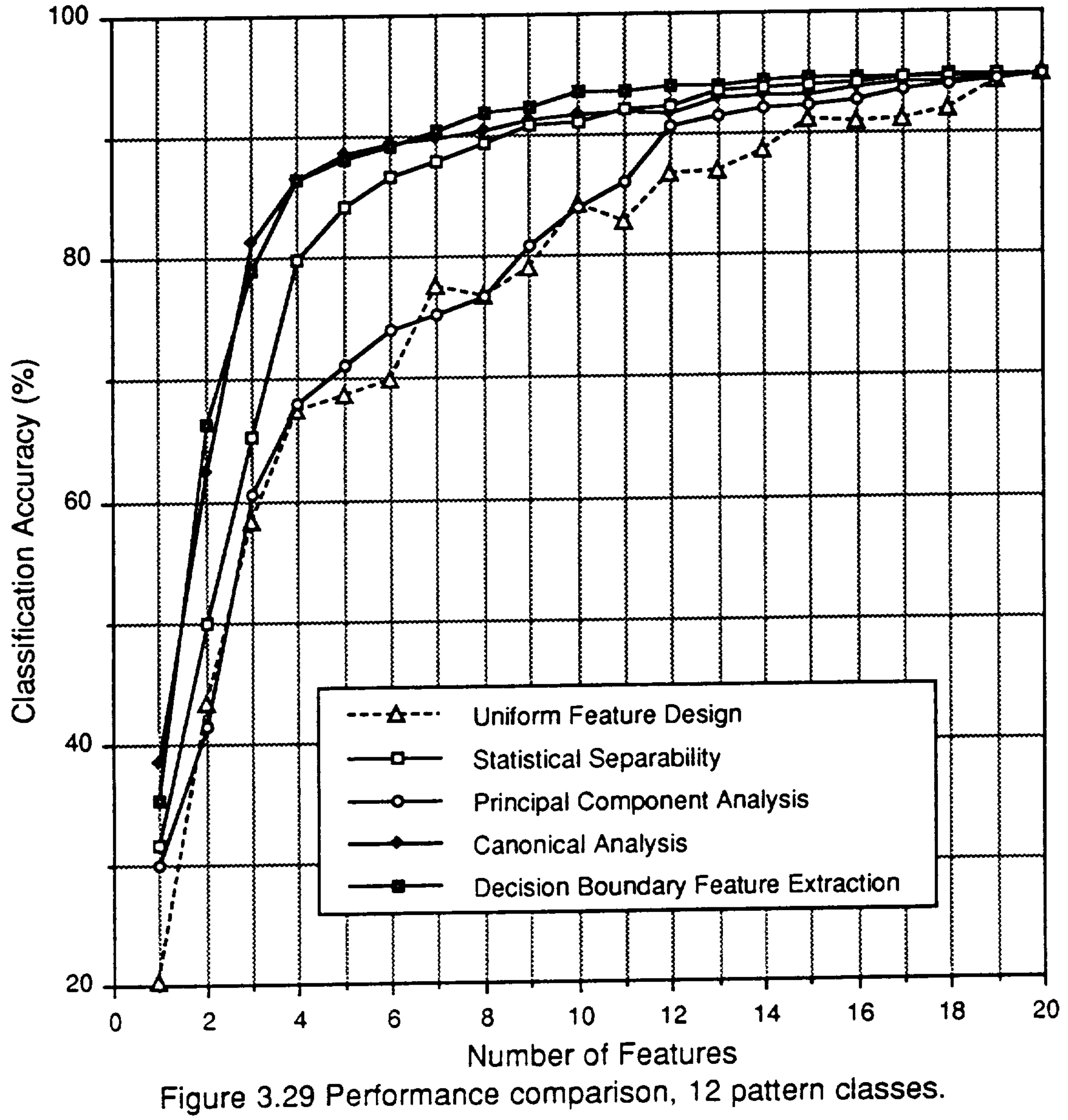

Figure 3.29 Performance comparison, 12 pattern classes.

Figure 3.29 show the performance comparison of the 5 feature selection/extraction algorithms for different numbers of features. In this case, Decision Boundary Feature Extraction and Canonical Analysis show comparable performances, although Decision Boundary Feature Extraction shows a little better performance than Canonical Analysis when more than 8 features are used. Statistical Separability shows a relatively good performance. It is noted that, as more features are used, the performances of 5 feature selection/extraction algorithms continue to improve.

In the next test, 40 classes are chosen from the FSS data. Table 3.14 provides information on the 40 classes. The data is multi-temporal. Figure 3.30 shows the performance comparison of the 5 feature selection/extraction algorithms for different numbers of features. In this case, Canonical Analysis, Statistical Separability and Decision Boundary Feature Extraction show essentially equivalent performances. In addition, as more features are used, the classification accuracies of the 5 feature selection/extraction algorithms continue to improve, suggesting that, for a large number of classes, a large number of features are also needed to discriminate between classes. In such a large number of classes, the fast classification algorithm in Chapter 2 can be employed.

Table 3.14 Class description of the multi-temporal 40 classes.

Date	Location	Species	No. of Data
770308	Finney Co. KS	Winter Wheat	691
770626	Finney Co. KS	Winter Wheat	677
771018	Hand Co. SD	Winter Wheat	662
770503	Finney Co. KS	Winter Wheat	658
770626	Finney Co. KS	Summer Fallow	643
780726	Hand Co. SD	Spring Wheat	518
780602	Hand Co. SD	Spring Wheat	517
780515	Hand Co. SD	Spring Wheat	474
780921	Hand Co. SD	Spring Wheat	469
780816	Hand Co. SD	Spring Wheat	464
780709	Hand Co. SD	Spring Wheat	454
781026	Hand Co. SD	Spring Wheat	441
760928	Finney Co. KS	Summer Fallow	414
781026	Hand Co. SD	Winter Wheat	393
771018	Hand Co. SD	Spring Wheat	313
770920	Hand Co. SD	Winter Wheat	292
780921	Hand Co. SD	Winter Wheat	292
770308	Finney Co. KS	Grain Sorghum	279
760928	Finney Co. KS	Grain Sorghum	277
780602	Hand Co. SD	Oats	259
780921	Hand Co. SD	Pasture	225
780515	Hand Co. SD	Winter Wheat	223
780726	Hand Co. SD	Native Grass Pas	217
781026	Hand Co. SD	Pasture	217
780816	Hand Co. SD	Summer Fallow	216
780602	Hand Co. SD	Native Grass Pas	214
780816	Hand Co. SD	Native Grass Pas	212
770503	Finney Co. KS	Summer Fallow	211
780726	Hand Co. SD	Summer Fallow	204
780515	Hand Co. SD	Native Grass Pas	196
780709	Hand Co. SD	Summer Fallow	190
771018	Hand Co. SD	Native Grass Pas	183
780921	Hand Co. SD	Oats	182
780726	Hand Co. SD	Oats	177
780709	Hand Co. SD	Native Grass Pas	170
780816	Hand Co. SD	Oats	165
780515	Hand Co. SD	Oats	163
780709	Hand Co. SD	Oats	163
771018	Hand Co. SD	Oats	161
770626	Finney Co. KS	Grain Sorghum	157

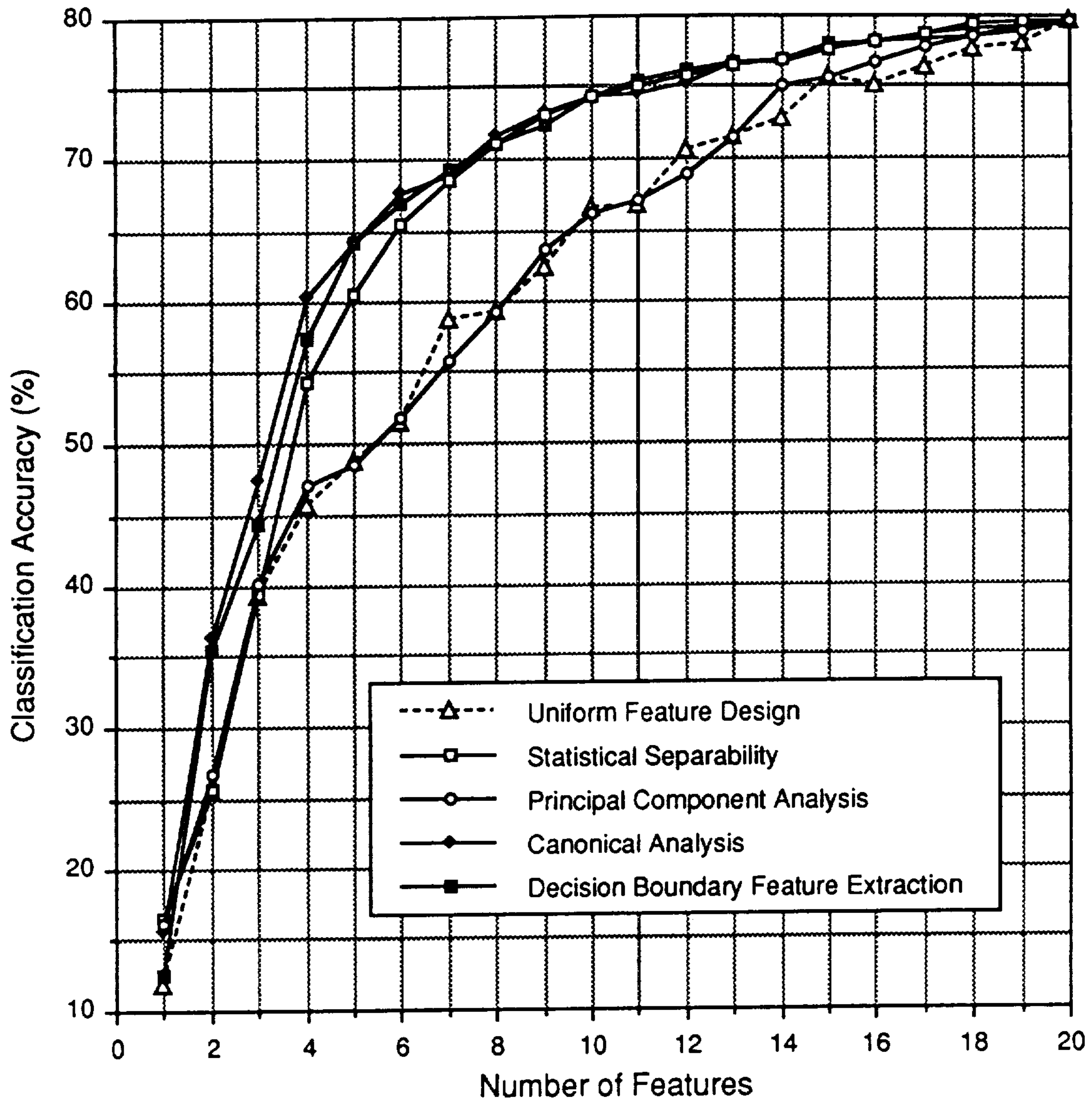

Figure 3.30 Performance comparison, 40 pattern classes.

3.6.3 Eigenvalues of Decision Boundary Feature Matrix and Classification Accuracy

Theoretically, the eigenvectors of the decision boundary feature matrix corresponding to non-zero eigenvalues will contribute to improvement of classification accuracy. However, in practice, a threshold must be set to determine the effectiveness of eigenvectors by the corresponding eigenvalues, especially for high dimensional real data. Figure 3.31 shows the relationship between the accumulation of eigenvalues of the decision boundary feature matrix and the normalized classification accuracies obtained by dividing the classification accuracies by the classification accuracy obtained using all features. There is a nearly linear relationship between normalized classification accuracy and accumulation of eigenvalues up to x≈95 where x is the accumulation of eigenvalues. As the accumulation of eigenvalues approaches 100 percent, the linear relationship between the normalized classification accuracy and the accumulation of eigenvalues does not hold; care must be taken to set the threshold. More experiments are needed to obtain a better understanding on the relationship between the normalized classification accuracy and the accumulation of eigenvalues.

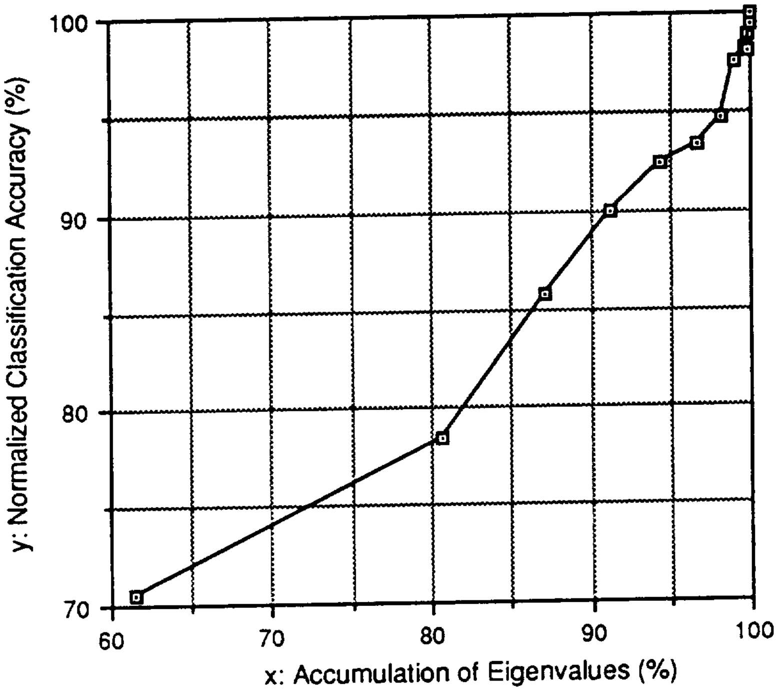

Figure 3.31 Relationship between the normalized classification accuracy (see text) and the accumulation of eigenvalues.

3.6.4 Decision Boundary Feature Extraction Method and the Foley & Sammon Method in High Dimensional Space

The Foley & Sammon method will find an optimum feature set if there is a reasonable class mean difference. However, the Foley & Sammon method fails if the class mean differences are small. Another problem with the Foley & Sammon method is that it does not take full advantage of information contained in the second order statistics. In the Foley & Sammon method (Foley and Sammon 1975), a new feature vector **d** is found to maximize R(**d**)

$$R(\mathbf{d}) = \frac{(\mathbf{d}^t\Delta)^2}{\mathbf{d}^t\mathbf{A}\mathbf{d}}$$

where **d** = column vector on which the data are projected;
$\Delta = \mathbf{M}_1 - \mathbf{M}_2$ and $\mathbf{M}_i$ is estimated mean of class w_i.
$\mathbf{A} = c\Sigma_1 + (1-c)\Sigma_2$ and $0 \le c \le 1$ and Σ_i is estimated covariance of class ω_i.

By using the lumped covariance **A** in the criterion, the Foley & Sammon method may lose some information contained in the difference of the class covariances. In a high dimensional space, information contained in the second order statistics play a significant role in discriminating between classes as shown in Chapter 7.

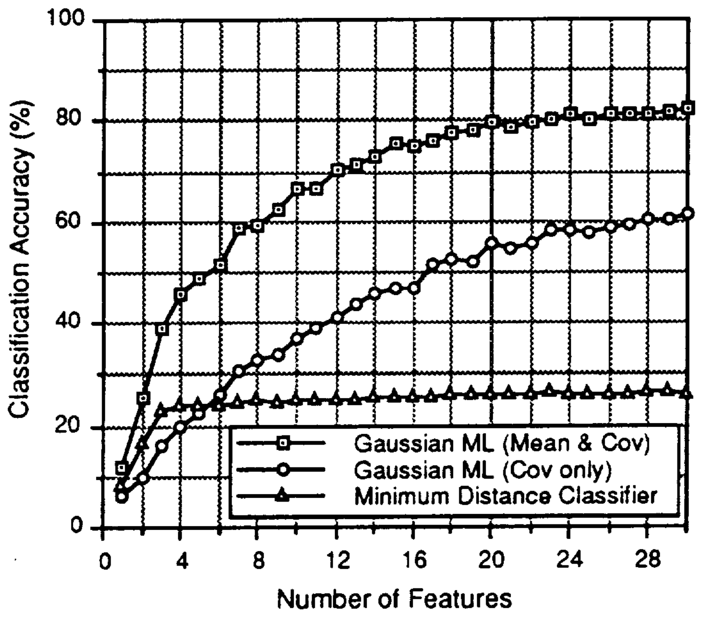

Figure 3.32 Performance comparison of the Gaussian ML classifier, the Gaussian ML classifier with zero mean, and the minimum distance classifier, tested on 40 multi-temporal classes.

Figure 3.32 shows an example. Three classifiers are tested on different numbers of features. The first classifier is the Gaussian ML classifier which utilizes both class mean and class covariance information. In the second test, the mean vectors of all classes were made zero and the Gaussian ML classifier was applied to the zero mean data. In other words, the second classifier, which is a Gaussian ML classifier, is constrained to use only covariance differences among classes. The third classifier is a conventional minimum distance classifier (Richards 1986) which utilizes only the first order statistics. It is noteworthy that the classifier using only first order statistics outperformed that using only second order statistics when the dimensionality was low. However, saturation soon set in, while performance of the classifier using only covariance information improved as more features were used. The implication seems to be that at low dimensionality the relative location of class distributions in feature space dominates in importance, but at higher dimensionality, the relative shape of the distribution dominates and in the long run is more significant to class separation.

In order to evaluate the performances of the Foley & Sammon method and Decision Boundary Feature Extraction for various mean differences in high dimensional space, the following test is done. Two classes are selected from FSS data. Table 3.15 shows the data on the two classes. In this test, all data are used for training and test.

Table 3.15 Class description.

SPECIES	No. of Sample	Date
1: WINTER WHEAT	658	May 3, 1977
2: WINTER WHEAT	393	Oct. 26, 1978

In the test, the mean of one class is moved relative to the mean of the other class. And performances of the Foley & Sammon method and Decision Boundary Feature Extraction are evaluated for various mean difference ($0.5\sigma \le \Delta = |\mathbf{M}_1 - \mathbf{M}_2| \le 5\sigma$) where σ is the average standard deviation, i.e.,

$$\sigma = \frac{1}{2N} \sum_{i=1}^{2} \sum_{j=1}^{N} \sigma_j^i$$

where N is the number of features and σ_j^i is j-th feature standard deviation of class ω_i.

Figures 3.33 and 3.34 shows the performances of the Foley & Sammon method and Decision Boundary Feature Extraction for various mean differences. First, it is noted that even when there is small mean difference ($\Delta = 0.5\sigma$), classification accuracy can be almost 100%. This shows again that information contained in the second order statistics plays a significant role in discriminating between classes in high dimensional space. As can be seen in Figure 3.33, the Foley & Sammon method fails to find a good feature set if the mean differences are relatively small ($\Delta \leq 2.5\sigma$). After there are sufficient mean differences ($\Delta \geq 3\sigma$), the Foley and Sammon method begins to find a good feature set. On the other hand, Decision Boundary Feature Extraction works well even when the mean differences are small and finds a good feature set utilizing the covariance differences as can be seen in Figure 3.34.

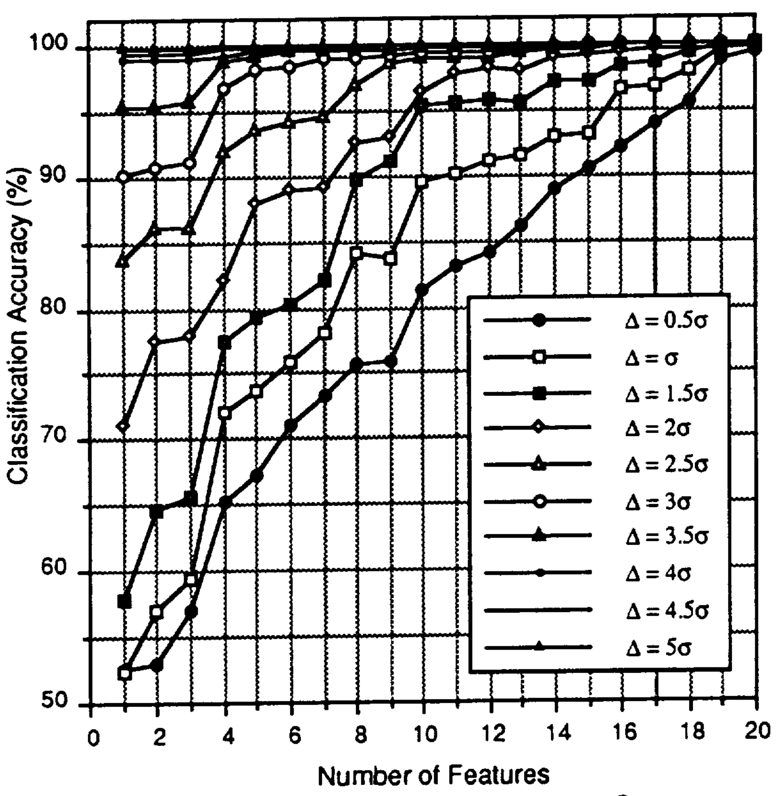

Figure 3.33 Performance comparison of the Foley & Sammon method for various mean differences.

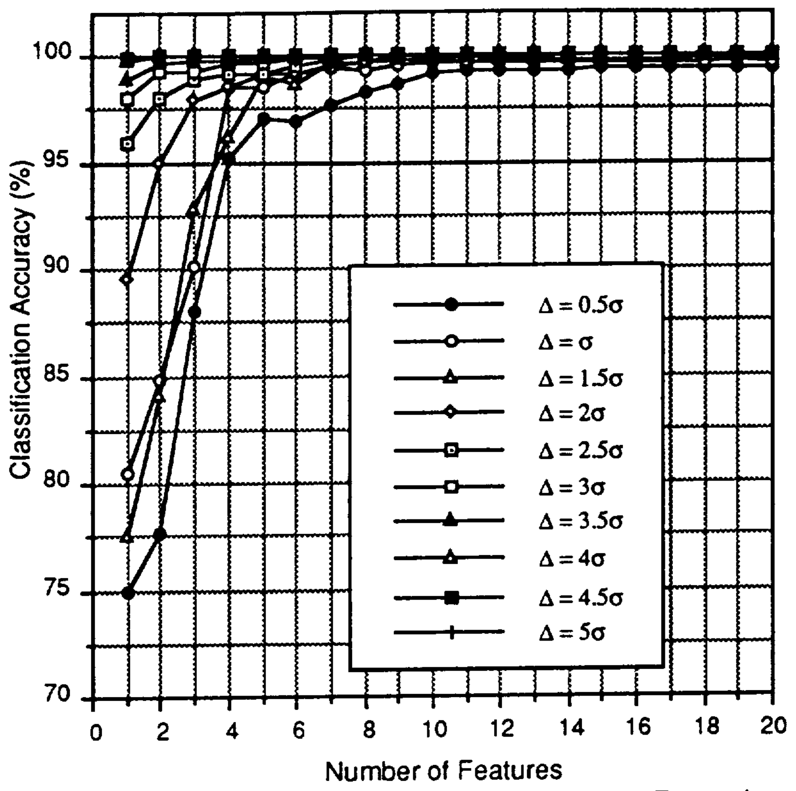

Figure 3.34 Performance comparison of Decision Boundary Feature Extraction for various mean differences.

Table 3.16 shows the number of features needed to achieve over 97% of the classification accuracy obtained using all features for the various mean differences. If $\Delta \geq 4\sigma$, both methods find a proper feature sets, achieving over 97% of the classification accuracy obtained using all features with one feature. For $3\sigma \leq \Delta \leq 3.5\sigma$, Decision Boundary Feature Extraction achieves over 97% of the classification accuracy obtained using all features with just one feature while the Foley & Sammon method needs 4-5 features. When $\Delta \leq 2.5\sigma$, the Foley & Sammon method performs poorly while Decision Boundary Feature Extraction achieves over 97% of the classification accuracy obtained using all features with 2, 3, 5, 4, and 5 features for $\Delta = 2.5\sigma$, 2σ, 1.5σ, σ, and 0.5σ, respectively.

Table 3.16 Number of features needed to achieve over 97% of the classification accuracy obtained using all features for the various mean differences. σ=average standard deviation.

Mean Difference	0.5σ	1σ	1.5σ	2σ	2.5σ	3σ	3.5σ	4σ	4.5σ	5σ
Foley & Sammon	19	18	14	11	9	5	4	1	1	1
Decision Boundary	5	4	5	3	2	1	1	1	1	1

3.7 Conclusion

We have proposed a new approach to feature extraction for classification based on decision boundaries. We defined discriminantly redundant features and discriminantly informative features for the sake of feature extraction for classification and showed that the discriminantly redundant features and the discriminantly informative features are related to the decision boundary. By recognizing that normal vectors to the decision boundary are discriminantly informative, the decision boundary feature matrix was defined using the normal vectors. It was shown that the rank of the decision boundary feature matrix is equal to the intrinsic discriminant dimension, and the eigenvectors of the decision boundary feature matrix corresponding to non-zero eigenvalues are discriminantly informative. We then proposed a procedure to calculate empirically the decision boundary feature matrix.

Except for some special cases, the rank of decision boundary feature matrix would be the same as the original dimension. However, it was noted that in many cases only a small portion of the decision boundary is effective in discriminating among pattern classes, and it was shown that it is possible to reduce the number of features by utilizing the effective decision boundary rather than the complete boundary.

The proposed feature extraction algorithm based on the decision boundary has several desirable properties. The performance of the proposed algorithm does not deteriorate even when there is little or no mean difference or covariance difference. In addition, the proposed algorithm predicts the minimum number of features required to achieve the same classification accuracy as in the original space for a given problem. Experiments show that the proposed feature extraction algorithm finds the right feature vectors even in cases where some previous algorithms fail to find them, and the performance of the proposed algorithm compares favorably with that of several previous algorithms.

Developments with regard to sensors for Earth observation are moving in the direction of providing much higher dimensional multispectral imagery than is now possible. The HIRIS instrument now under development for the Earth Observing System (EOS), for example, will generate image data in 192 spectral bands simultaneously. In order to analyze data of this type, new techniques for all aspects of data analysis will no doubt be required. The proposed algorithm provides such a new and promising approach to feature extraction for classification of such high dimensional data.

Even though the experiments are conducted using Gaussianly distributed data or assuming a Gaussian distribution, all the developed theorems hold for other distributions or to other decision rules as well. In addition, it will be shown in the next chapter how the proposed algorithm can be also applied for non-parametric classifiers if the decision boundary can be found numerically.

CHAPTER 4 DECISION BOUNDARY FEATURE EXTRACTION FOR NON-PARAMETRIC CLASSIFICATION

4.1 Introduction

Although many authors have studied feature extraction for parametric classifiers (Decell and Guseman 1979), relatively few algorithms are available for non-parametric classifiers. The lack of practical feature extraction algorithms for the non-parametric classifier is mainly due to the nature of a non-parametric classifier. Without an assumption about the underlying density functions, feature extraction for non-parametric classifiers is often practically not feasible or very time consuming in many cases.

Some general feature extraction methods could be used for non-parametric classifiers. Muasher and Landgrebe (1983) proposed a method to base feature extraction on the statistics of the whole data set. Although this is not optimal in a theoretical sense, it can be used even when underlying class densities are unknown, or precise estimates of them are not possible. In addition, such methods can be used for both parametric and non-parametric classifiers. Since, in many cases, it may be difficult to obtain enough training samples, feature extraction methods based on the whole data set may be a good and useful solution.

In discriminant analysis (Fukunaga 1990), a within-class scatter matrix Σ_W and a between-class scatter matrix Σ_b are used to formulate a criterion function. A typical criterion is

$$J_1 = tr(\Sigma_W^{-1}\Sigma_b) \tag{4.1}$$

where Σ_w is the within-class scatter matrix and Σ_b is the between-class scatter matrix as defined in Section 3.2. New feature vectors are selected to maximize the criterion. Fukunaga proposed a non-parametric discriminant analysis which is based on non-parametric extensions of commonly used scatter matrices (Fukunaga and Mantock 1983). Patrick proposed a non-parametric feature extraction process where a non-quadratic distance function defined between classes is used to define the best linear subspace (Patrick and Fischer II 1969).

Features can be selected under a criterion which is related to the probability of error. The Bhattacharyya distance is a measure of statistical separability and is defined as follows (Fukunaga 1990):

$$\mu(\tfrac{1}{2}) = -\ln \int_S [\, p(\mathbf{X}/\omega_1) p(\mathbf{X}/\omega_2)\,]^{\frac{1}{2}} d\mathbf{X} \tag{4.2}$$

Although theoretically it is possible to calculate equation (4.2) for a non-parametric classifier such as Parzen density estimator, in practice, it is frequently not feasible due to a prohibitively long computing time, particularly for high dimensional data.

Short and Fukunaga showed that, by problem localization, most pattern recognition problems can be solved using simple parametric forms, while global parametric solution may be untractable (Fukunaga and Short 1978). Short and Fukunaga also proposed a feature extraction algorithm using problem localization (Short and Fukunaga 1982). They considered feature extraction as a mean-square estimation of the Bayes risk vector. The problem is simplified by partitioning the distribution space into local subregions and performing a linear estimation in each subregion.

Though the computation cost of non-parametric classifiers is often much larger than that of parametric classifiers, there are some cases where the use of non-parametric classifiers is desirable. For instance, if underlying densities are unknown or problems involve complex densities which cannot be approximated by the common parametric density functions, use of a non-parametric classifier may be necessary. However, for high dimensional data and multi-source data, the computation cost of non-parametric classifiers can be very large. As a result, there is a greater need for a practical feature extraction algorithm which can take a full advantage of non-parametric classifiers which can define an arbitrary decision boundary.

In this chapter, we extend the decision boundary feature extraction method in Chapter 3 to non-parametric cases (Lee and Landgrebe 1991-1). The method is based directly on the decision boundary. Instead of utilizing distributions of data, we explore the decision boundary which the employed classifier defines. It has been shown that all feature vectors which are helpful in discriminating between classes can be obtained from the decision boundary (Lee and Landgrebe 1991-2). Thus, by extracting features directly from the decision boundary which a non-parametric classifier defines, one can fully explore the advantage of the non-parametric classifier. Since the decision boundary can not be expressed analytically in the non-parametric case, the proposed algorithm finds points on the decision boundary numerically. From these points, feature vectors are extracted. The proposed algorithm predicts the minimum number of features to achieve the same classification accuracy as in the original space while at the same time finding the needed feature vectors.

4.2 Decision Boundary Feature Extraction for Non-Parametric Classification

4.2.1 Effective Decision Boundary in Non-Parametric Classifiers

In Chapter 3, we defined the effective decision boundary for parametric classifiers as follows (see Definition 3.4):

Definition 3.4 The effective decision boundary is defined as

$$\{ \mathbf{X} \mid h(\mathbf{X}) = t\,,\ \mathbf{X} \in R_1 \text{ or } \mathbf{X} \in R_2 \}$$

where R_1 is the smallest region which contains a certain portion, $P_{\text{threshold}}$, of class ω_1 and R_2 is the smallest region which contains a certain portion, $P_{\text{threshold}}$, of class ω_2.

Also, the effective decision boundary feature matrix was defined as follows:

Definition 3.7 The effective decision boundary feature matrix (EDBFM): Let **N**(**X**) be the unit normal vector to the decision boundary at a point **X** on the effective decision boundary for a given pattern classification problem. Then the effective decision boundary feature matrix Σ_{EDBFM} is defined as

$$\Sigma_{EDBFM} = \frac{1}{K'} \int_{S'} N(X)N^t(X)p(X)dX$$

where p(**X**) is a probability density function, $K' = \int_{S'} p(X)dX$, and S' is the effective decision boundary as defined in Definition 3.4, and the integral is performed over the effective decision boundary.

In parametric classifiers, assuming Gaussian distributions, the above definitions gives a proper meaning. However, in non-parametric classifiers, the above definitions may not give a correct effective decision boundary when the problem involves outliers or some special multimodal cases.

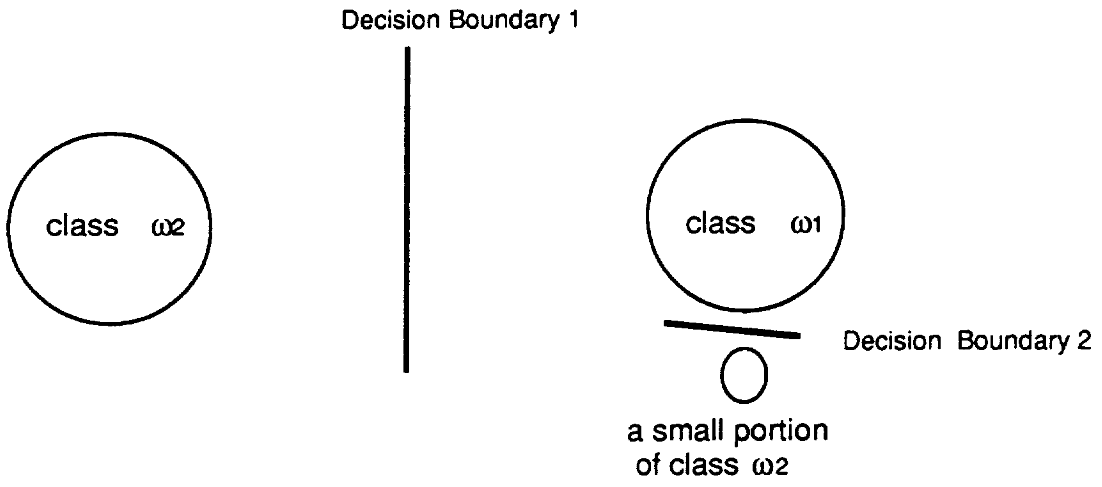

Figure 4.1 Effective decision boundary in non-parametric classifiers.

Figure 4.1 illustrates such an problem. In Figure 4.1, Decision Boundary 1 should be the effective decision boundary to be considered in calculating the decision boundary feature matrix. However, according the Definition 3.7, Decision Boundary 2 will be more heavily weighted. As a result, inefficiency will be introduced in the calculated decision boundary feature matrix.

To overcome such a problem in non-parametric classifiers, the definition of the effective decision boundary (Definition 3.4) needs to be generalized. We can define the effective decision boundary as the portion of the whole decision boundary which separates most of the data in the same way as the whole decision boundary separates. To be more precise, we generalize the definition of the effective decision boundary as follows:

Definition 4.1 The effective decision boundary of $P_{portion}$ is defined as the portion of the whole decision boundary which separates $P_{portion}$ of the data in the same way as the whole decision boundary separates.

It is noted that Definition 4.1 holds for parametric and non-parametric classifiers and gives a proper physical meaning. It can be viewed that Definition 3.4 is a special case of Definition 4.1 assuming Gaussian distribution. With the effective decision boundary as in Definition 4.1, the definition of the effective decision boundary feature matrix (Definition 3.7) will give a relevant result for non-parametric classifiers even when the problem involves outliers. However, as will be seen, it is more difficult to locate the effective decision boundary in non-parametric classifiers than in parametric classifiers. We will discuss this problem in detail later.

4.2.2 Parzen Density Estimation and Selection of Kernel Size

A non-parametric classifier with Parzen density estimation will be used to test the proposed feature extraction algorithm for non-parametric classification; thus we will briefly discuss Parzen density estimation. Parzen density estimation with kernel φ is defined as (Duda and Hart 1973)

$$p(X) = \frac{1}{Nh^N}\sum_{i=1}^{n} \varphi\left(\frac{X - X_i}{h}\right)$$

where N is the dimensionality of the data, and h is the window size, and n is the number of training samples. The kernel φ must be non-negative and satisfy the following condition:

$$\frac{1}{h^N}\int_{R^N}\varphi(X)\,dX = 1$$

Although many authors have studied the problem of determining the value of the Parzen scale parameter h, no theoretical value of h gives consistently optimum results (Fukunaga and Hummels 1987). As a result, we determined the best h experimentally in our experiments. Figure 4.2 shows the classification results for various h. The peak performance occurs when h is between 0.5 and 0.7 in this case.

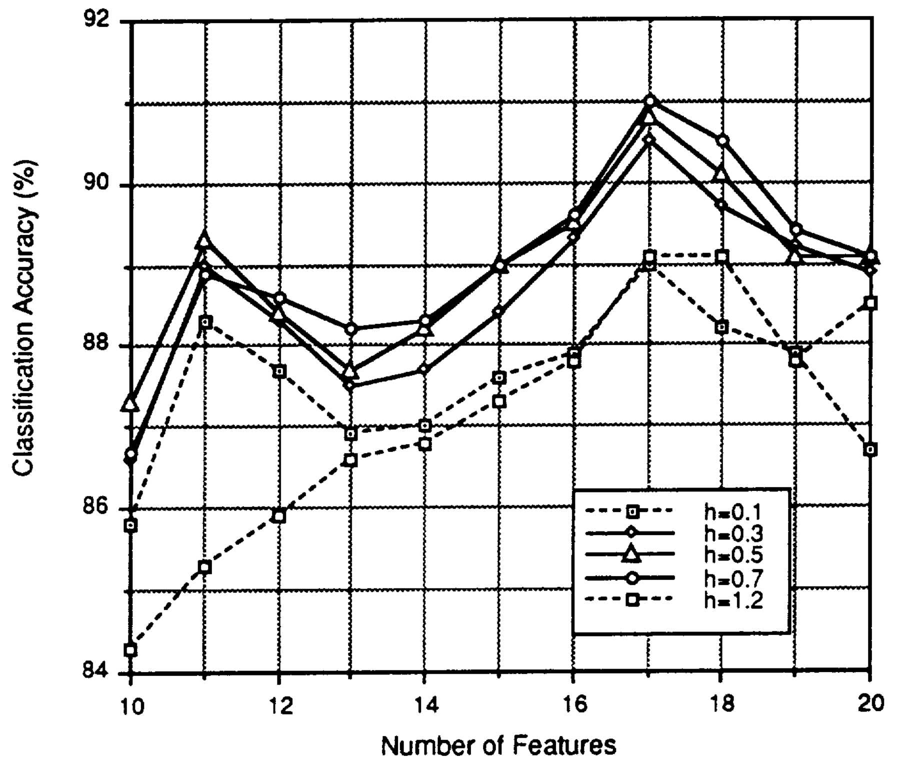

Figure 4.2 Determining the best h experimentally.

4.2.3 Determining the Decision Boundary and Finding Normal Vectors to the Decision Boundary for Non-Parametric Classifiers

In order to extract feature vectors from the decision boundary of a given classifier, we need to calculate the decision boundary feature matrix Σ_{DBFM} as given in Definition 3.6. Then Theorem 3.2 and Theorem 3.3 tell us that the eigenvectors of Σ_{DBFM} corresponding to non-zero eigenvalues of Σ_{DBFM} are all the feature vectors needed for discriminating between the classes for the given classifier as shown in Chapter 3. In order to calculate the decision boundary feature matrix Σ_{DBFM}, the decision boundary must be found. However, in general, a non-parametric classifier defines an arbitrary decision boundary which may not be expressed in analytic form. Therefore the decision boundary for non-parametric classifiers must be calculated numerically.

In section 3.4.3, we defined the decision boundary as follows (Definition 3.3):

$$\{\mathbf{X} \mid h(\mathbf{X})=t\}$$

$$\text{where} \quad h(\mathbf{X}) = -\ln\frac{P(\mathbf{X}|\omega_1)}{P(\mathbf{X}|\omega_2)} \tag{4.3}$$

$$t = \ln\frac{P(\omega_1)}{P(\omega_2)} \tag{4.4}$$

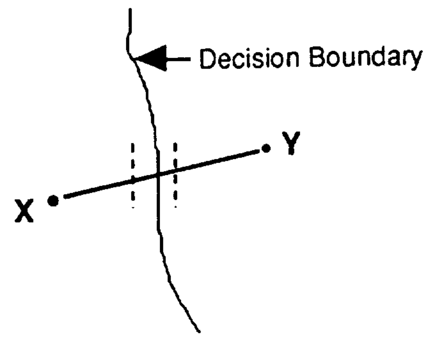

Figure 4.3 Finding decision boundary numerically for non-parametric classifiers.

Consider an example in Figure 4.3. Assuming **X** and **Y** are classified differently, the line connecting **X** and **Y** must pass through decision boundary. Although, by moving along the line, we can find a point **Z** at which h(**Z**)=t, there is no guarantee that the point **Z** is exactly on the true decision boundary, even though h(**Z**)=t. Figure 4.4 shows an example. In the example, data are generated for the following statistics.

$$\mathbf{M}_1 = \begin{bmatrix} -0.6 \\ 0.6 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \qquad \mathbf{M}_2 = \begin{bmatrix} 0.6 \\ -0.6 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

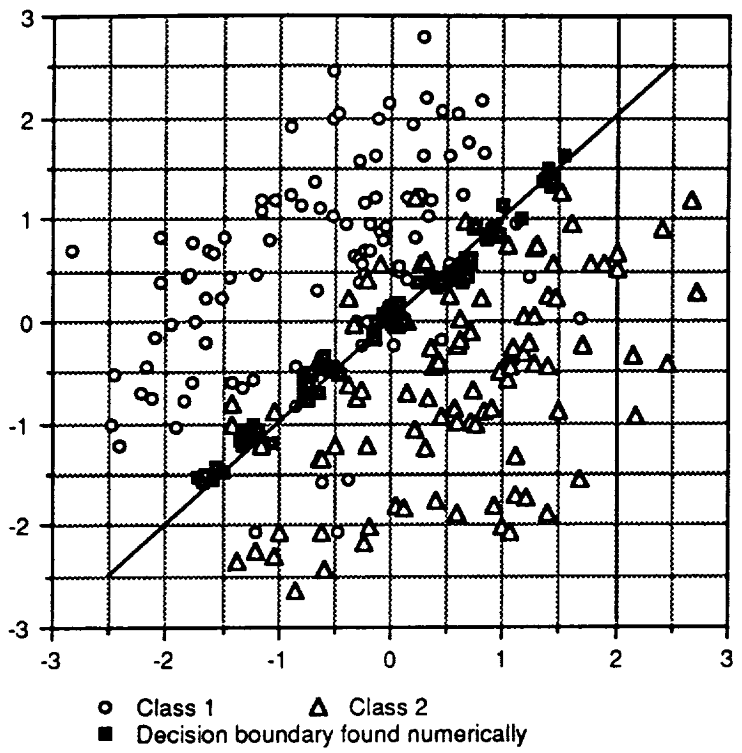

Figure 4.4 Finding decision boundary numerically.

The points of the decision boundary found numerically are shown along with the true decision boundary plotted as a straight line in Figure 4.4. As can be seen, the points of the numerically found decision boundary are distributed along the true decision boundary. However, the points are not exactly on the true decision boundary. The problem that the numerically found decision boundary does not match exactly the true decision boundary becomes more apparent when training samples are limited or the Parzen scale parameter h is small. However, in our experiments, we found that inaccurate estimation of the decision boundary has relatively little impact on the performance of the decision boundary feature extraction method for non-parametric classifiers if the estimated decision boundary is in the vicinity of the true decision boundary. We will discuss this problem more in the experiments.

A normal vector to the decision boundary at **X** is given by

$$\nabla h(\mathbf{X}) = \frac{\partial h}{\partial x_1}\mathbf{x}_1 + \frac{\partial h}{\partial x_2}\mathbf{x}_2 + \cdots + \frac{\partial h}{\partial x_n}\mathbf{x}_n \qquad (4.5)$$

However, in non-parametric classifiers, the decision boundary can not be expressed analytically and equation (4.5) can not be used. Instead, we may estimate the normal vector as follows:

$$\nabla h(\mathbf{X}) \approx \frac{\Delta h}{\Delta x_1}\mathbf{x}_1 + \frac{\Delta h}{\Delta x_2}\mathbf{x}_2 + \cdots + \frac{\Delta h}{\Delta x_n}\mathbf{x}_n \tag{4.6}$$

A problem of estimating a normal vector numerically is that the nearest samples have often much influence on the estimation of normal vectors. This problem becomes more apparent when training samples are limited or the Parzen scale parameter h is small. As a result, care must be taken in selecting the Parzen scale parameter h, particularly in a high dimensional space. We will discuss this problem more in the experiments.

4.2.4 Decision Boundary Feature Extraction Procedure for Non-Parametric Classification

Now we propose the following procedure to find decision boundary numerically and calculate the decision boundary feature matrix for non-parametric classifiers.

Procedure for Feature Extraction for Non-Parametric Classifier
Utilizing the Decision Boundary
(2 pattern class case)

STEP 1: Classify the training data using full dimensionality.

STEP 2: For each sample correctly classified as class ω_1, find the nearest sample correctly classified as class ω_2. Repeat the same procedure for the samples correctly classified as class ω_2.

STEP 3: Connect the pairs of samples found in STEP 2. Since a pair of samples are classified differently, the line connecting the pair of samples must pass through the decision boundary. By moving along the line, find the point on the decision boundary or near the decision boundary within a threshold.

STEP 4: At each point found in STEP 3, estimate the unit normal vector $\mathbf{N}_i$ by

$$\mathbf{N}_i = \nabla h(\mathbf{X}) \,/\, |\nabla h(\mathbf{X})|$$

where $\nabla h(\mathbf{X}) \approx \frac{\Delta h}{\Delta x_1} \mathbf{x}_1 + \frac{\Delta h}{\Delta x_2} \mathbf{x}_2 + \cdots + \frac{\Delta h}{\Delta x_n} \mathbf{x}_n$

$h(\mathbf{X}) = -\ln \frac{P(\mathbf{X}|\omega_1)}{P(\mathbf{X}|\omega_2)}$ assuming Bayes' decision rule for minimum error is used.

STEP 5: Estimate the decision boundary feature matrix using the normal vectors found in STEP 4.

$\Sigma_{EDBFM} = \frac{1}{L}\sum_{i}^{L} \mathbf{N}_i \mathbf{N}_i^t$ where L is the number of points found on the decision boundary.

STEP 6: Select the eigenvectors of the decision boundary feature matrix as new feature vectors according to the magnitude of corresponding eigenvalues.

Euclidean distance is used to find the nearest sample in STEP 2 in our experiments. Figure 4.5 shows an illustration of the proposed procedure. Although the proposed procedure does not find the decision boundary where data are sparsely distributed, this is an advantage, not a disadvantage of the procedure. By concentrating on the decision boundary where most of data are distributed, the feature extraction can be more efficient as shown in Chapter 3. The classification error increase resulting from not considering the decision boundary in the region where data are sparsely distributed will be minimal since there will be very little data in that region.

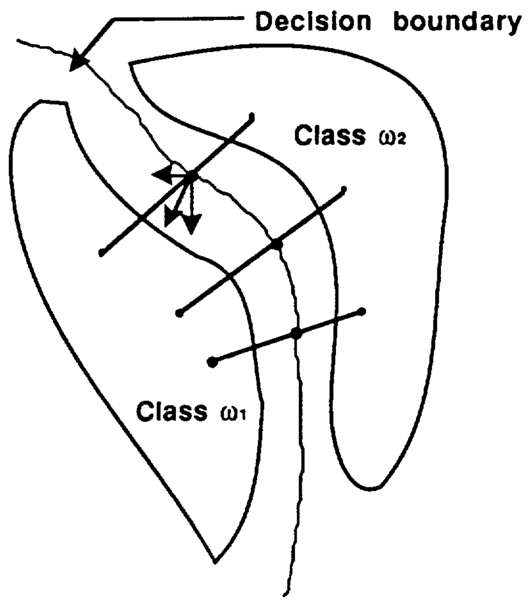

Figure 4.5 Illustration of the procedure feature extraction for a non-parametric classifier utilizing decision boundary.

4.2.5 Outlier Problem

If the two classes are very separable and one class has some outliers as shown in Figure 4.6, the proposed procedure will calculate the decision boundary feature matrix with more weight on the decision boundary between the outliers of class ω_2 and class ω_1 (Decision Boundary 2) than the decision boundary between the main portion of class ω_2 and class ω_1 (Decision Boundary 1), assuming the outliers are correctly classified. Although such a case as in Figure 4.6 will not occur frequently in real applications, such outliers will make the proposed procedure less efficient since Decision Boundary 1 is the effective decision boundary in that case. However, it is not a fundamental problem of the decision boundary feature extraction algorithm, but a procedural problem of how to find the effective decision boundary (Definition 4.1). In parametric classifiers, such outliers could be eliminated using the chi-square threshold test. In non-parametric classifiers, it is more difficult to eliminate such outliers.

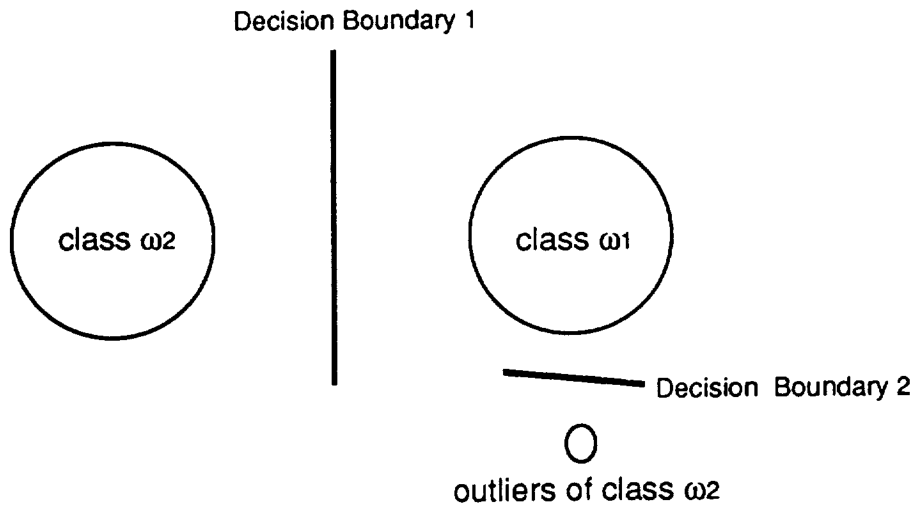

Figure 4.6 Outlier problem.

To overcome the outlier problem, STEP 2 in the proposed procedure can be modified as follows:

STEP 2a: For each sample correctly classified as class ω_1, select randomly a sample correctly classified as class ω_1. Repeat the same procedure for the samples correctly classified as class ω_2.

By randomly selecting a sample classified as the other class, the decision boundary which separates the main portion of classes will be weighted more heavily, and inefficiency caused by outliers, when classes are very separable, will be eliminated.

However, if data are distributed as shown in Figure 4.7, STEP 2a will cause the inclusion of some ineffective decision boundary in calculating the decision boundary feature matrix, while STEP 2 can concentrate on the effective decision boundary. Thus, in the case of Figure 4.7, which is a more typical case in real data, STEP 2 will be more efficient than STEP 2a. The problem can be summarized as how to find the effective decision boundary even when there exists outliers. So STEP 2 in the proposed procedure can be modified as follows:

STEP 2b: For each sample correctly classified as class ω_1, find the L nearest samples correctly classified as class ω_2. From the L nearest samples, select randomly a sample. Repeat the same procedure for the samples correctly classified as class ω_2.

By increasing L, one can eliminate the outlier problem. By decreasing L, one can concentrate on the effective decision boundary. Thus, there will be a tradeoff between eliminating the outlier problem and concentrating on the effective decision boundary. As pointed out previously, the problem is how to find the effective decision boundary. If one can exactly locate the effective decision boundary, the decision boundary feature extraction algorithm will be more effective.

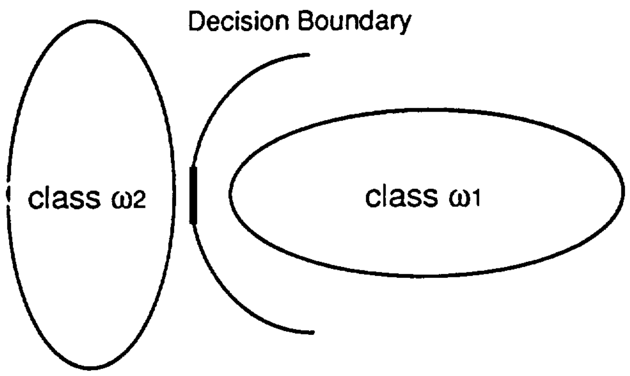

Figure 4.7 A more typical data distribution and its decision boundary.

4.2.6 Non-Parametric Classifiers Not Defining Probability Densities

Some non-parametric classifiers such as the kNN classifier do not define class probability densities. If the employed non-parametric classifier does not define class probability densities, h(**X**) in equation (4.3) can not be calculated. In such a case, normal vectors can not be estimated. In that case, one might find a vector along which the classification result changes most rapidly. For example, let **X** be a point on the decision boundary. Then find the smallest Δx_i such that the classification result of $\mathbf{X}+\Delta x_i \mathbf{x}_i$ is different from that of **X**. We may then estimate a unit vector **N** along which the classification result changes most rapidly as follows:

$$\mathbf{N} = \mathbf{V} / |\mathbf{V}| \text{ where } \mathbf{V} \approx \frac{1}{\Delta x_1}\mathbf{x}_1 + \frac{1}{\Delta x_2}\mathbf{x}_2 + \cdots + \frac{1}{\Delta x_n}\mathbf{x}_n$$

4.2.7 Multiclass Cases

If there are more than two classes, the procedure can be repeated for each pair of classes, and the total effective decision boundary feature matrix can be calculated by averaging the effective decision boundary feature matrices which are calculated for each pair of classes. If prior probabilities are available, the summation can be weighted. In other words, if there are M classes, the decision boundary feature matrix can be calculated as

$$\Sigma_{DBFM} = \sum_{i}^{M} \sum_{j,\, j\neq i}^{M} P(\omega_i)P(\omega_j)\Sigma_{DBFM}^{ij}$$

where Σ_{DBFM}^{ij} is the decision boundary feature matrix between class ω_i and class ω_j and $P(\omega_i)$ is the prior probability of class ω_i if available. Otherwise let $P(\omega_i)=1/M$.

4.3 Decision Boundary Feature Extraction and Problem Localization

By problem localization, Short and Fukunaga showed that most pattern recognition problems can be solved using simple parametric forms (Fukunaga and Short 1978). In (Short and Fukunaga 1982), Short and Fukunaga proposed a feature extraction method using problem localization. In their method, the original space is subdivided into a number of subregions and a linear estimation is performed in each subregion. A modified clustering algorithm is used to find the subregions. To a certain extent, the decision boundary feature extraction method parallels the problem localization approach. In problem localization, Short and Fukunaga recognized that a parametric discriminant function can be used in each subregion (Fukunaga and Short 1978). In the decision boundary feature extraction method, we recognized that only a small portion of the decision boundary plays a significant role in discriminating between classes.

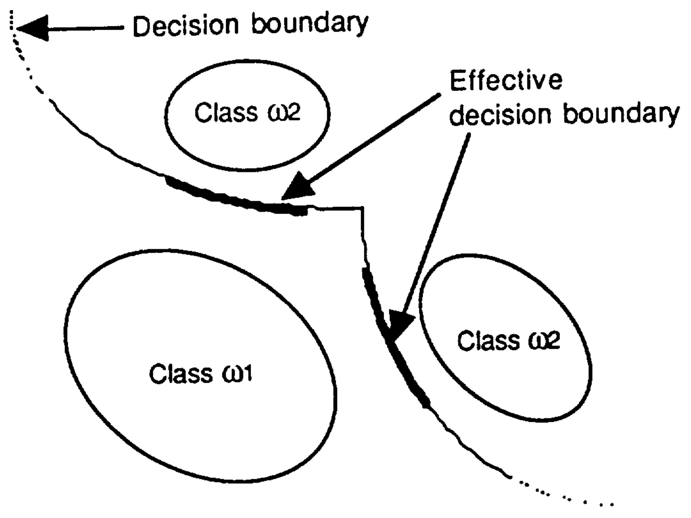

Figure 4.8 Decision boundary and effective decision boundary.

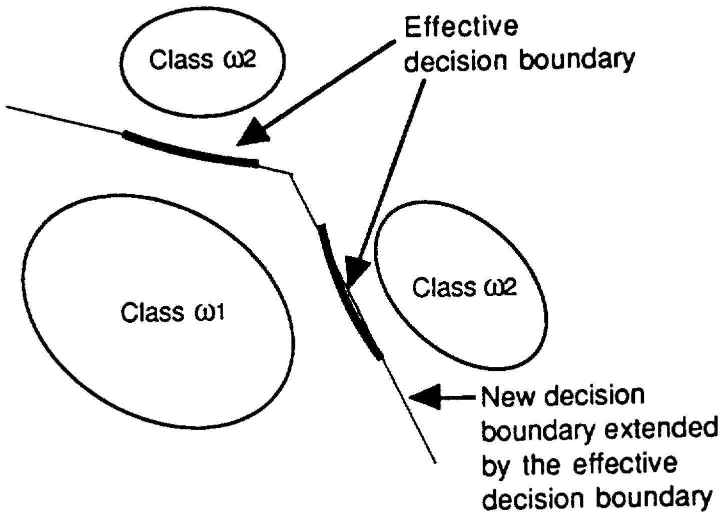
Effective decision boundary
Class ω2
Class ω1
Class ω2
New decision boundary extended by the effective decision boundary

Figure 4.9 Effective decision boundary and new decision boundary extended by the effective decision boundary.

Consider the case of Figure 4.8. The effective decision boundary which is plotted in bold, plays a significant role in discriminating between classes. Even if the effective decision boundary is used, the data still can be classified in almost the same manner as when the whole decision boundary is used as shown in Figure 4.9. On the other hand, parts of the decision boundary, which are plotted as plain lines, play relatively little role in discriminating between classes while some part of the decision boundary, plotted as a dotted line, are rarely used. Therefore, we recognized that by concentrating on the effective decision boundary, the feature extraction can be more efficient. It is noted that the effective decision boundary need not be linear or be represented by a parametric form.

However, the decision boundary feature extraction method differs from the problem localization in several ways. First, the decision boundary feature extraction method does not divide the pattern space into subregions. Dividing the pattern space into subregions is not an easy task when the number of subregions is unknown. This problem becomes apparent particularly in a multiclass problem with real, high dimensional data. Secondly, the decision boundary feature extraction method finds a global feature set while a local feature set is found in the problem localization. Thirdly, in the problem localization, Short and Fukunaga take advantage of the fact that class boundaries are likely to be more nearly linear in each subregions while the decision boundary feature extraction method does not assume that the effective decision boundary is nearly linear or can be represented in a parametric form. In the decision boundary feature extraction method, the effective decision boundary can be of any shape. Finally the decision boundary feature extraction method has the capability to predict the minimum number of features needed to achieve the same classification accuracy as in the original space.

4.4 Experiment and Result

4.4.1 Experiments with generated data

In order to evaluate closely how the proposed algorithm performs under various circumstances, tests are conducted on generated data with given statistics. The non-parametric classifier was implemented by Parzen density estimation using a Gaussian kernel function (Silverman 1986). In each example, classification accuracies of the decision boundary feature extraction method and the discriminant analysis using equation (4.1) as a criterion function are compared. We will refer the decision boundary feature extraction method as Decision Boundary Feature Extraction, and discriminant analysis using equation (4.1) as Discriminant Analysis.

Example 4.1 In this example, class ω_1 is normal with the following statistics:

$$M_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \Sigma_1 = \begin{bmatrix} 3 & 0.5 \\ 0.5 & 3 \end{bmatrix}$$

Class ω_2 is equally divided between two normal distributions with the following statistics:

$$M_2^1 = \begin{bmatrix} -3 \\ 3 \end{bmatrix} \quad \Sigma_2^1 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix} \quad \text{and} \quad M_2^2 = \begin{bmatrix} 3 \\ -3 \end{bmatrix} \quad \Sigma_2^2 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix}$$

200 samples are generated for each class. Figure 4.10 shows the distribution of the data along with the decision boundary found by the proposed procedure numerically. Eigenvalues λ_i and eigenvectors ϕ_i of Σ_{EDBFM} are calculated as follows:

$$\lambda_1 = 0.98338, \ \lambda_2 = 0.01662 \qquad \phi_1 = \begin{bmatrix} 0.69 \\ -0.72 \end{bmatrix}, \quad \phi_2 = \begin{bmatrix} 0.72 \\ 0.69 \end{bmatrix}$$

Since one eigenvalue is significantly larger than the other, it can be said that the rank of Σ_{EDBFM} is 1. That means only one feature is needed to achieve the same classification accuracy as in the original space. Considering the statistics of the two classes, the rank of Σ_{EDBFM} gives the correct number of features needed to achieve the same classification accuracy as in the original space. Table 4.1 shows the classification accuracies of Decision Boundary Feature Extraction and Discriminant Analysis. Decision Boundary Feature Extraction finds the right features achieving the same classification accuracy with one feature while Discriminant Analysis performs significantly less well in this example since class means are the same.

Table 4.1 Classification accuracies of Decision Boundary Feature Extraction and Discriminant Analysis in Example 4.1.

Number of Features	Discriminant Analysis	Decision Boundary Feature Extraction
1	54.5 (%)	92.8 (%)
2	91.8 (%)	92.0 (%)

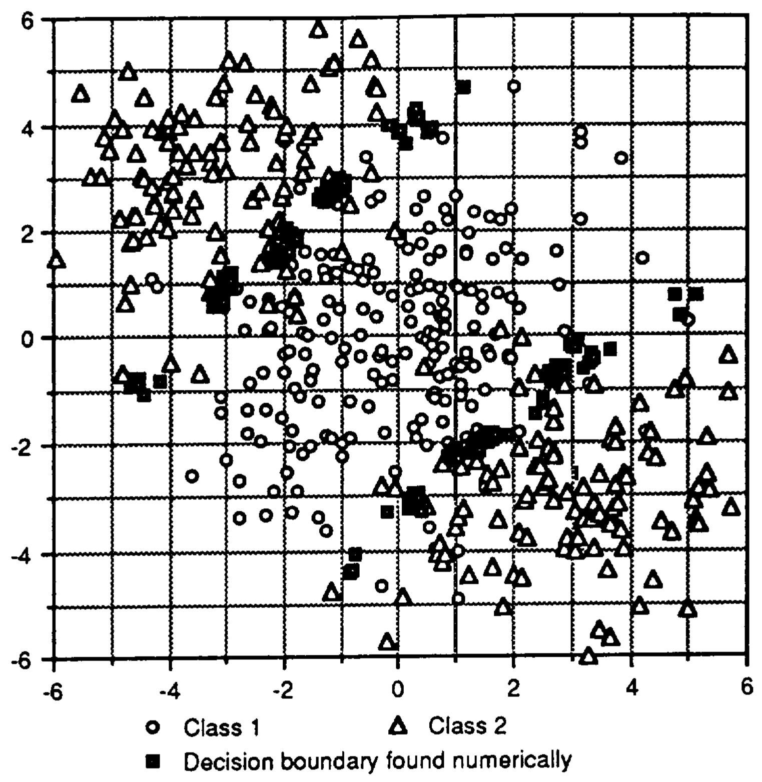

Figure 4.10 Data distribution of Example 4.1. The decision boundary found by the proposed procedure is also shown.

Example 4.2 In this example, class ω_1 is normal with the following statistics:

$$\mathbf{M}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad \Sigma_1 = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

Class ω_2 is equally divided between two normal distributions with the following statistics:

$$\mathbf{M}_2^1 = \begin{bmatrix} 3 \\ 0 \\ 0.1 \end{bmatrix} \quad \Sigma_2^1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 9 \end{bmatrix} \quad \text{and} \quad \mathbf{M}_2^2 = \begin{bmatrix} -3 \\ 0 \\ 0.1 \end{bmatrix} \quad \Sigma_2^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

200 samples are generated for each class. From the statistics, it can be seen that the decision boundary approximately consists of two cylindrical surfaces. Figure 4.11 shows the distribution of the data in the x1-x2 plane. The decision boundary found by the proposed procedure numerically is also shown. Eigenvalues λ_i and eigenvectors ϕ_i of Σ_{EDBFM} are calculated as follows:

$$\lambda_1 = 0.58993,\ \lambda_2 = 0.33985,\ \lambda_3 = 0.07021$$

$$\phi_1 = \begin{bmatrix} 0.25 \\ -0.97 \\ 0.00 \end{bmatrix},\ \phi_2 = \begin{bmatrix} 0.95 \\ 0.24 \\ 0.17 \end{bmatrix},\ \phi_3 = \begin{bmatrix} -0.17 \\ -0.04 \\ 0.98 \end{bmatrix}$$

$$\text{Rank}(\Sigma_{EDBFM}) \approx 2$$

It can be said that the rank of Σ_{EDBFM} is approximately 2. Thus two features are needed to achieve the same classification accuracy as in the original space, which agrees with the data. Table 4.2 shows the classification accuracies of Decision Boundary Feature Extraction and Discriminant Analysis. Decision Boundary Feature Extraction finds the correct features achieving about the same classification accuracy with two features while Discriminant Analysis performs significantly less well, since there is no class mean difference.

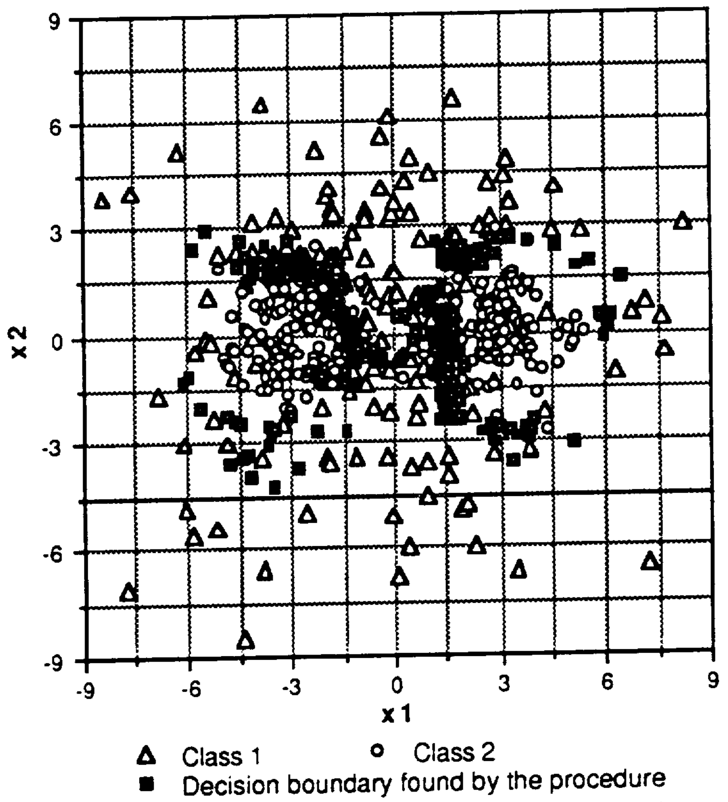

Figure 4.11 Data distribution of Example 4.2. The decision boundary found by the proposed procedure is also shown.

Table 4.2 Classification accuracies of Decision Boundary Feature Extraction and Discriminant Analysis in Example 4.2.

Number of Features	Discriminant Analysis	Decision Boundary Feature Extraction
1	61.5 (%)	68.8 (%)
2	67.8 (%)	76.3 (%)
3	76.0 (%)	76.3 (%)

4.4.2 Experiments with real data

Real data sets were selected from a high dimensional multispectral remote sensing data base of agricultural areas. The data were collected by the Field Spectrometer System (FSS), a helicopter-mounted field spectrometer, as a part of the LACIE program (Biehl et. al 1982). Table 4.3 shows the major parameters of FSS.

Table 4.3 Parameters of Field Spectrometer System (FSS).

Number of Bands	60
Spectral Coverage	0.4 - 2.4 μm
Altitude	60 m
IFOV(ground)	25 m

Along with the proposed algorithm, three other feature extraction algorithms, Uniform Feature Design, the Karhunen-Loeve transformation (Principal Component Analysis) (Duda and Hart 1973), and the discriminant analysis using equation (4.1) as a criterion function (Fukunaga 1990) are tested to evaluate and compare the performance of the proposed algorithm. Uniform Feature Design is a simple band combination procedure. For example, if the number of features is to be reduced to 30, every two consecutive bands are combined to form a new feature. Where the number of features desired is not evenly divisible into 60, the nearest integer number of bands is used. For example, for 9 features, the first 6 original bands were combined to create the first feature, then the next 7 bands were combined to create the next feature, and so on. Uniform Feature Design is used as a baseline means to evaluate efficiencies of the other feature extraction methods. The discriminant analysis using equation (4.1) is referred as Discriminant Analysis.

In the first test, 4 classes are chosen from the FSS data. Table 4.4 provides information on the 4 classes. Figure 4.12 shows the mean graph of the 4 classes. As can be seen, there are reasonable mean differences among the classes. In this test, 400 randomly selected samples are used for training and the rest are used for test.

Table 4.4 Class description.

SPECIES	DATE	No. of Samples
Winter Wheat	May 3, 1977	657
Unknown Crops	May 3, 1977	678
Winter Wheat	March 8, 1977	691
Unknown Crops	March 8, 1977	619

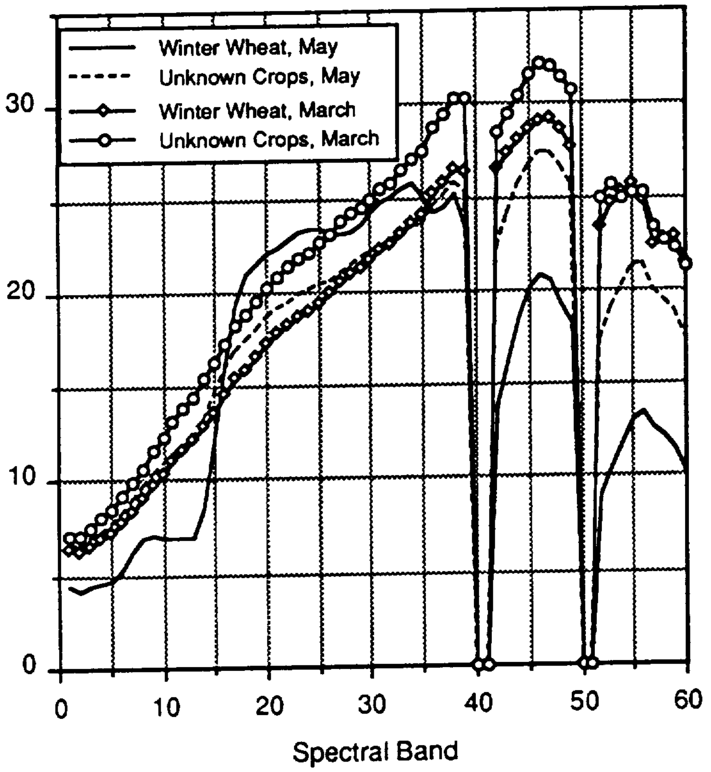

Figure 4.12 Mean graph of the two classes of Table 4.4.

Figure 4.13 shows a performance comparison. First the original 60 dimensional data is reduced to 17 dimensional data using Uniform Feature Design. And then Decision Boundary Feature Extraction, Discriminant Analysis, and Principal Component Analysis are applied to the 17 dimensional data. With 17 features, the classification accuracy is about 90.0%. In low dimensions (number of features ≤ 3), Discriminant Analysis performs better than the other methods.

When more than 3 features are used, Decision Boundary Feature Extraction starts to performs better than the other methods.

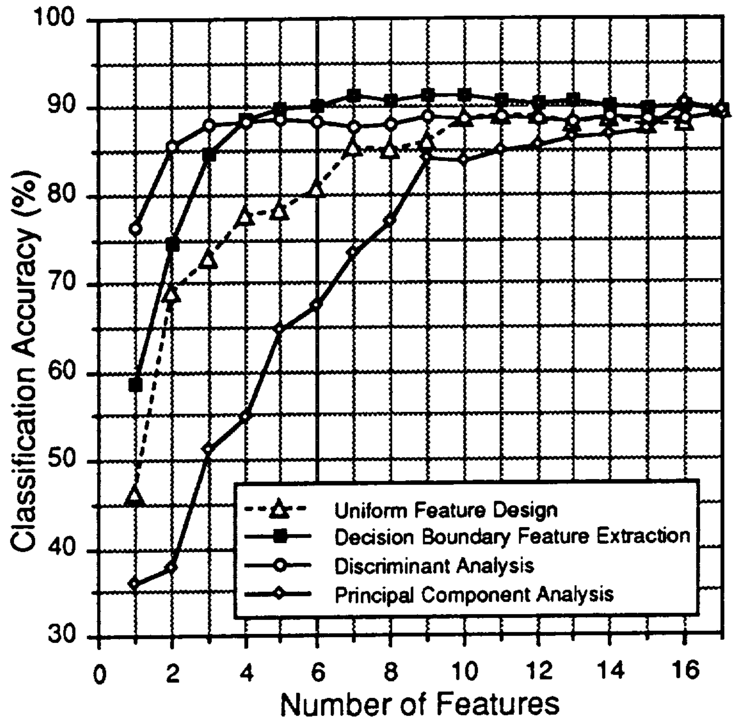

Figure 4.13 Performance comparison of Uniform Feature Design, Decision Boundary Feature Extraction, Discriminant Analysis, and Principal Component Analysis.

In the next test, there are 3 classes and each class has 2 subclasses. In other words, 2 subclasses were combined to form a new class. By purposely combining data from different classes, the data are made to be multi-modal. Table 4.5 provides information on the classes. Figure 4.14 shows a mean value graph of the 6 subclasses, and Figure 4.15 shows a mean value graph of the 3 classes each of which has 2 subclasses. 500 randomly selected samples from each classes are used as training data and the rest are used for test.

Table 4.5 Class description.

Class	Subclass	No. of Samples	Total No. of Sample
Class ω_1	Winter Wheat March 8, 1977	691	1209
	Spring Wheat July 26, 1978	518	
Class ω_2	Winter Wheat June 26, 1977	677	1146
	Spring Wheat Sep. 21, 1978	469	
Class ω_3	Winter Wheat Oct. 18, 1977	662	1103
	Spring Wheat Oct. 26, 1978	441	

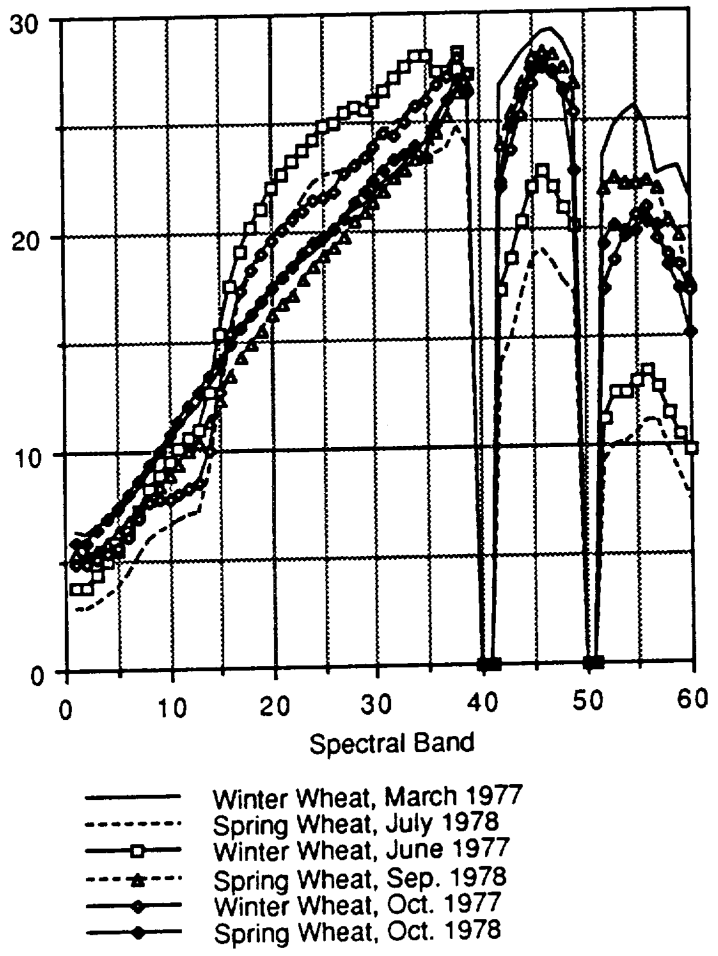

Figure 4.14 Mean graph of the 6 sub-classes of Table 4.5.

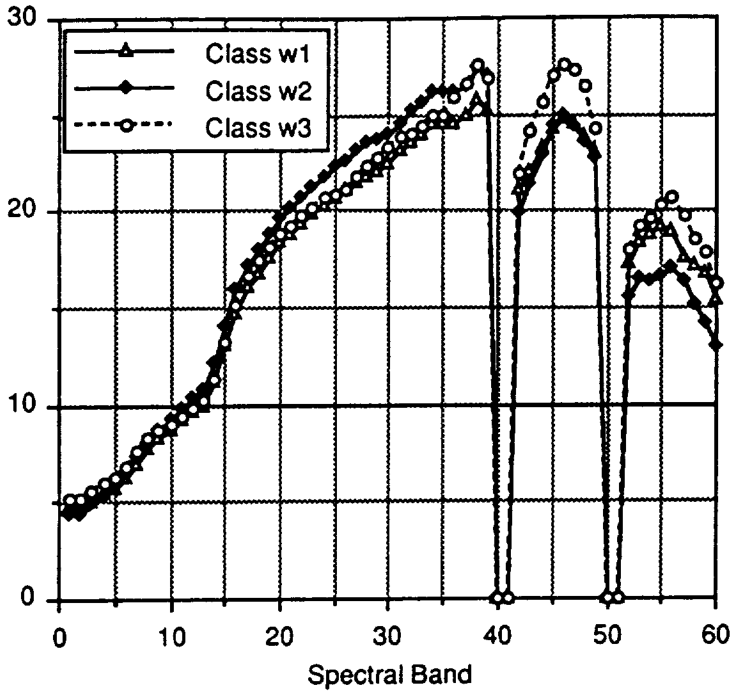

Figure 4.15 Mean graph of the 3 classes of Table 4.5.

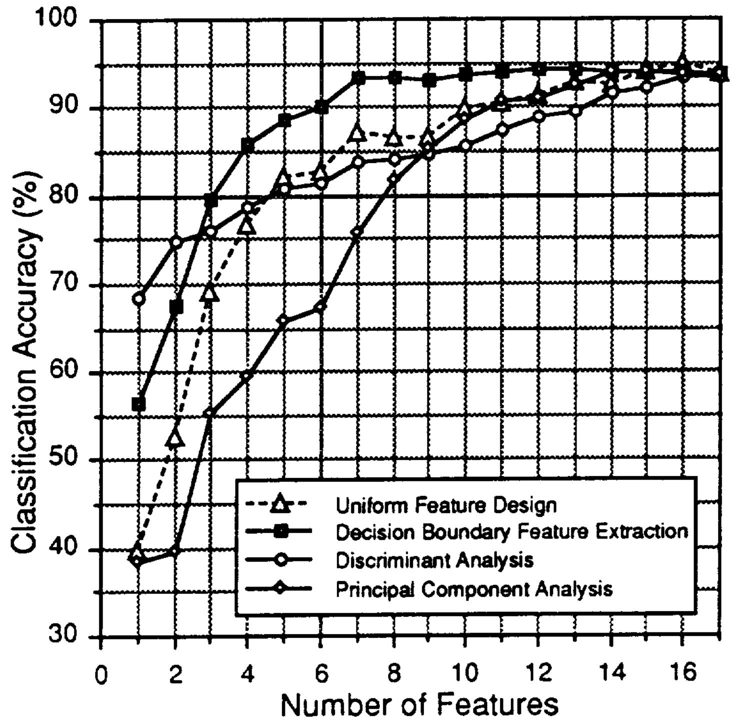

Figure 4.16 Performance comparison of Uniform Feature Design, Decision Boundary Feature Extraction, Discriminant Analysis, and Principal Component Analysis of the data of Table 4.5 (test data).

Figure 4.16 shows a performance comparison. With 17 features, the classification accuracy is about 89%. Discriminant Analysis shows the best performances until 3 features are used. However, the classification accuracies are much lower than the maximum possible classification accuracy and the comparison seems to be irrelevant. Decision Boundary Feature Extraction shows consistently better performances when more than 3 features are used. Decision Boundary Feature Extraction achieves about 89% classification accuracy with 7 features while all other methods needs 13-17 features to achieve about the same classification accuracy.

In the following test, there are 3 classes and each class has 2 subclasses. In other words, 2 subclasses were combined to form a new class. By purposely combining data from different classes, the data are made to be multi-modal. Table 4.6 provides information on the classes. 500 randomly selected samples from each classes are used as training data and the rest are used for test.

Table 4.6 Class description.

Class	Subclass	No. of Samples	Total No. of Sample
Class ω_1	Winter Wheat May 3, 1977	658	1340
	Unknown Crops May 3, 1977	682	
Class ω_2	Winter Wheat March 8, 1977	691	1310
	Unknown Crops March 8, 1977	619	
Class ω_3	Winter Wheat June 26, 1977	677	1320
	Summer Fallow June 26, 1977	643	

Figures 4.17-18 show the performance comparison. First the original 60 dimensional data was reduced to 17 dimensional data using Uniform Feature Design. And Decision Boundary Feature Extraction, Discriminant Analysis, and Principal Component Analysis were applied to the 17 dimensional data. With the 17 features, the classification accuracies of training data and test data are 96.5% and 95.7%, respectively. In low dimensionality ($N \leq 2$), Discriminant Analysis shows the best performances, though the difference between Discriminant Analysis and the decision boundary feature extraction method is small. However, when more than 2 features are used, the decision boundary feature extraction method outperforms all other methods.

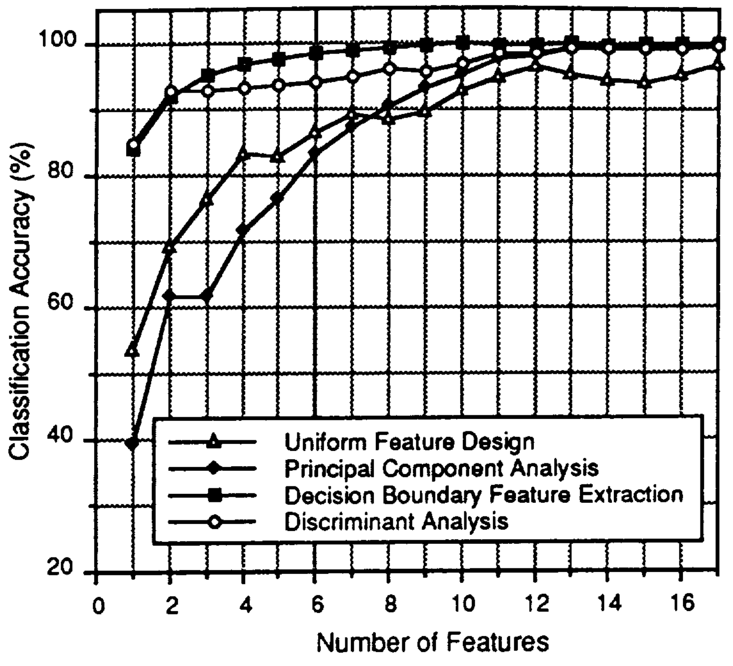

Figure 4.17 Performance comparison (train data).

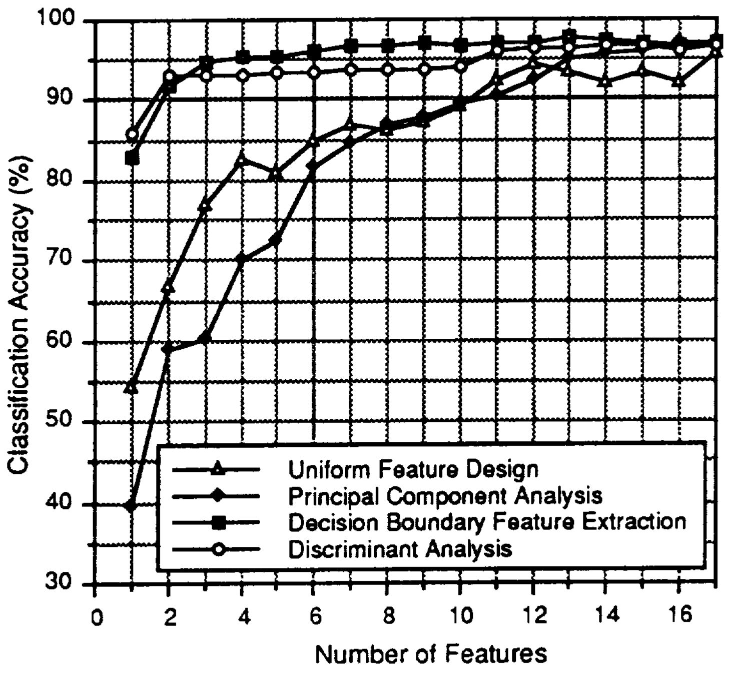

Figure 4.18 Performance comparison (test data).

In the following test, there are 3 classes and each class has 2 subclasses. In other words, 2 subclasses were combined to form a new class. By purposely combining data from different classes, the data are made to be multi-modal. Table 4.7 provides information on the classes. 500 randomly selected samples from each classes are used as training data and the rest are used for test.

Table 4.7 Class description.

Class	Subclass	No. of Samples	Total No. of Sample
Class ω_1	Spring Wheat July 9, 1978	454	972
	Spring Wheat July 26, 1978	518	
Class ω_2	Winter Wheat June 26, 1977	677	1339
	Winter Wheat Oct. 18, 1977	662	
Class ω_3	Spring Wheat Oct. 26, 1978	441	910
	Spring Wheat Sep. 21, 1978	469	

Figures 4.19-20 show the performance comparison. First the original 60 dimensional data was reduced to 17 dimensional data using Uniform Feature Design. Next Decision Boundary Feature Extraction, Discriminant Analysis, and Principal Component Analysis were applied to the 17 dimensional data. With the 17 features, the classification accuracies of training data and test data are 99.5% and 96.9%, respectively. In low dimensionality ($N \leq 2$), Discriminant Analysis shows the best performances, though the difference between Discriminant Analysis and the decision boundary feature extraction method is small. However, when more than 2 features are used, the decision boundary feature extraction method outperforms all other methods. With 5 features, the decision boundary feature extraction method achieves about 96.4% classification accuracy for test data while Principal Component Analysis, Discriminant Analysis and Uniform Feature Design achieve about 90.5%, 92.2%, and 87.9%, respectively.

It can be said that when class mean differences are reasonably large and classes are uni-modal, Discriminant Analysis finds a good feature set. However, when classes are multi-modal, Discriminant Analysis does not often find a good feature set. On the other hand, Decision Boundary Feature Extraction finds a good feature set even when classes are multi-modal.

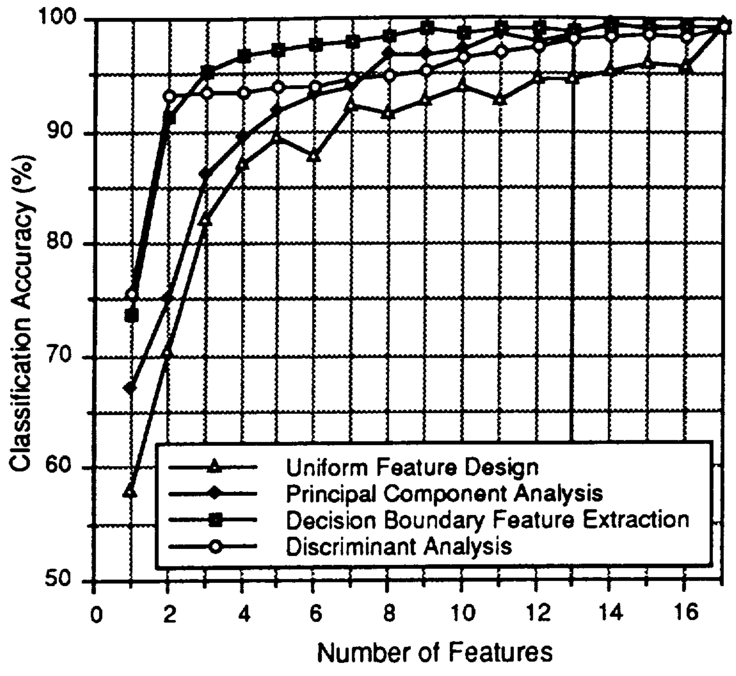

Figure 4.19 Performance comparison (train data).

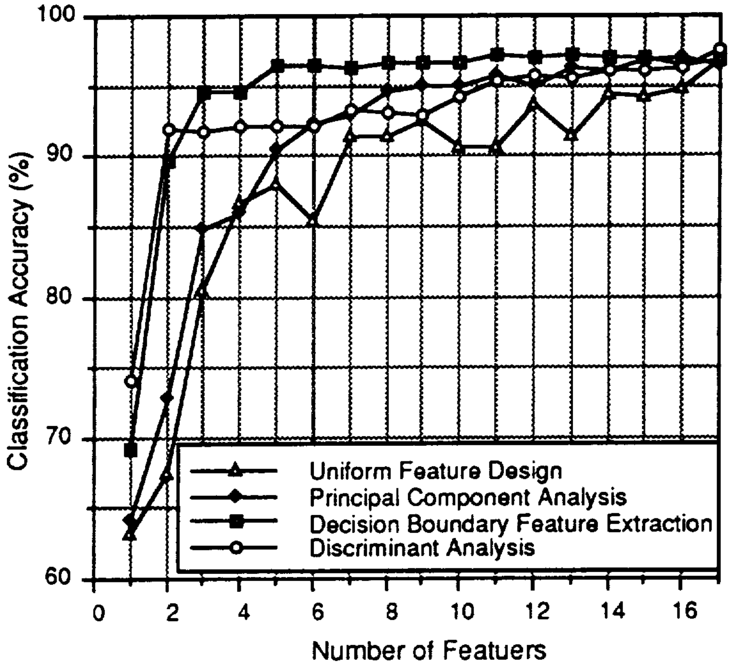

Figure 4.20 Performance comparison (test data).

4.4.3 Eigenvalues of Decision Boundary Feature Matrix and Classification Accuracy

Table 4.8 lists the eigenvalues of the decision boundary feature matrix of the 17 dimensional data, along with proportions and accumulations. It also shows classification accuracies and normalized classification accuracies obtained by dividing classification accuracies with the classification accuracy obtained using the whole feature set.

The rank of the decision boundary feature matrix (Σ_{DBFM}) must be decided upon, and in this case, somewhat arbitrarily so. Theoretically, the classification result obtained using all the eigenvectors of the decision boundary feature matrix corresponding to non-zero eigenvalues are the same as the classification result obtained using the whole feature set. However, for real data, eigenvalues of the decision boundary feature matrix are seldom zero, even though some eigenvalues are very close to zero, and there are large differences among the eigenvalues. As a result, although it is relatively easy to decide the rank of the decision boundary feature matrix for low dimensional generated data, it becomes less obvious for high dimensional real data. In non-parametric classification, it would be more difficult since the decision boundary and normal vectors are estimated. One may add eigenvalues until the accumulation exceeds 95% of the total sum and set that number of the eigenvalues as the rank of the Σ_{DBFM}. Defined in this way, the rank of the Σ_{DBFM} would be 9. Alternatively, one may retain the eigenvalues greater than one tenth of the largest eigenvalue. In this way, the rank of the Σ_{DBFM} would be 6. As can be seen of Table 4.8, the normalized classification accuracy increases monotonically as the accumulation of eigenvalues increases up to 5 features. After 5 features, the classification accuracy is almost saturated and adding more features does not improve classification accuracy. Figure 4.21 shows the relationship between the accumulations of eigenvalues and the normalized classification accuracies. More experiments are needed to obtain a better understanding on the relationship between the normalized classification accuracy and the accumulation of eigenvalues.

Table 4.8 Relationship between eigenvalues of the decision boundary feature matrix and classification accuracy.
(Ev: Eigenvalues, Pro: Proportion, Accu: Accumulation, Cl. Ac: Classification Accuracy, N. Cl. Ac: Normalized Classification Accuracy)

	Ev	Pro (%)	Accu (%)	Cl. Ac (%)	N.Cl.Ac (%)
1	995.2	34.5	34.5	57.1	63.2
2	556.4	19.3	53.8	84.7	93.7
3	446.4	15.5	69.3	87.9	97.2
4	293.3	10.2	79.4	88.5	97.9
5	138.5	4.8	84.2	89.8	99.3
6	120.5	4.2	88.4	89.8	99.3
7	88.6	3.1	91.5	90.1	99.7
8	55.8	1.9	93.4	90.1	99.7
9	50.8	1.8	95.2	90.5	100.1
10	46.2	1.6	96.8	90.2	99.8
11	34.0	1.2	97.9	90.1	99.7
12	21.4	0.7	98.7	90.2	99.8
13	14.1	0.5	99.2	90.3	99.9
14	11.3	0.4	99.6	90.4	100.0
15	5.8	0.2	99.8	90.4	100.0
16	4.5	0.2	99.9	90.4	100.0
17	2.3	0.1	100.0	90.4	100.0

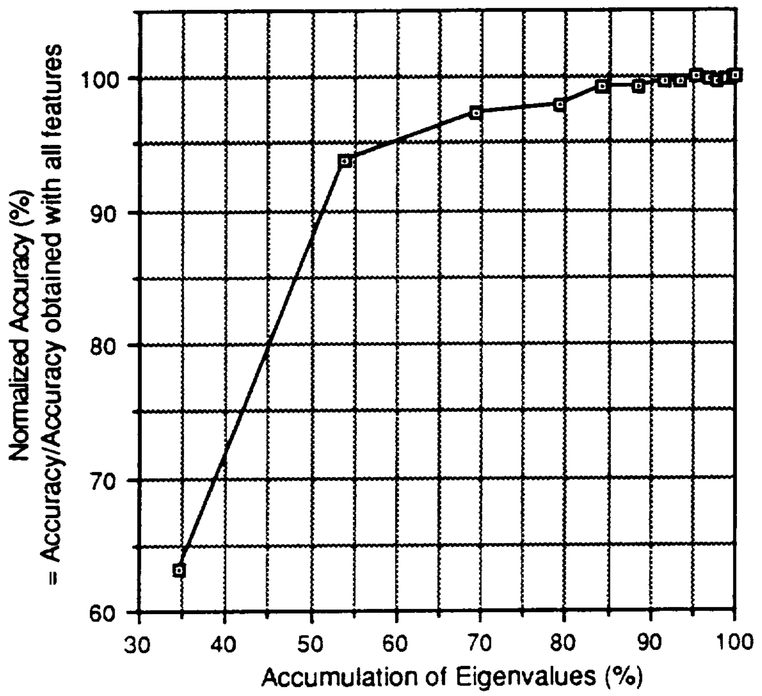

Figure 4.21 Relationship between Accumulations of Eigenvalues and Normalized Classification Accuracies.

4.5 Estimation of the Decision Boundary and Normal Vector

Since non-parametric classifiers do not define the decision boundary in analytic form, it must be estimated numerically. Then, from the estimated decision boundary, normal vectors are estimated as follows:

$$\nabla h(\mathbf{X}) \approx \frac{\Delta h}{\Delta x_1}\mathbf{x}_1 + \frac{\Delta h}{\Delta x_2}\mathbf{x}_2 + \cdots + \frac{\Delta h}{\Delta x_n}\mathbf{x}_n$$

Next we will investigate the effect of inaccurate estimation of the decision boundary and normal vectors on the performance of the proposed decision boundary feature extraction.

4.5.1 Effect of Inaccurate Estimation of the Decision Boundary

In the proposed procedure, we found a point on the decision boundary by moving along the line connecting two differently classified samples. In other words, by moving along the line, we try to find a point **X** such that

$$h(\mathbf{X})=t$$

When the difference between the decision boundary and an estimated decision boundary is smaller than a threshold, the searching procedure stopped. In other words, if

$$(h(\mathbf{X}) - t)(h(\mathbf{X}') - t) < 0 \text{ and } |\mathbf{X} - \mathbf{X}'| < \varepsilon$$

we take either **X** or **X'** as a point on the decision boundary. To investigate the sensitivity of the decision boundary feature extraction method, it was applied to the 17 dimensional data with various thresholds, $\varepsilon=0.01\sigma$, 0.05σ, 0.1σ, 0.5σ, 1σ and 2σ, where σ is the average standard deviation, i.e.,

$$\sigma = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\sigma_j^i$$

where N is the number of features, M is the number of classes, and σ_j^i is j-th feature standard deviation of class ω_i.

With 17 features, the classification accuracy is 90.4%. Figure 4.22 shows the performance comparison for the first 5 features. For 1 feature, there is not much difference. For 2 features, the classification accuracy decreases as the threshold increases. If more than 2 features are considered, the performances are essentially the same. When 3 features are used, all thresholds achieve about 89% classification accuracy. From the experiments, it appears that the threshold between 0.05σ and 0.5σ would be reasonable, and the performance of the decision boundary feature extraction method does not appear to be very sensitive to inaccurate estimation of the decision boundary if the estimated decision boundary is in the vicinity of the true decision boundary. Furthermore, there is no guarantee that a smaller threshold always results in a more accurate estimation of the decision boundary (section 4.2.3).

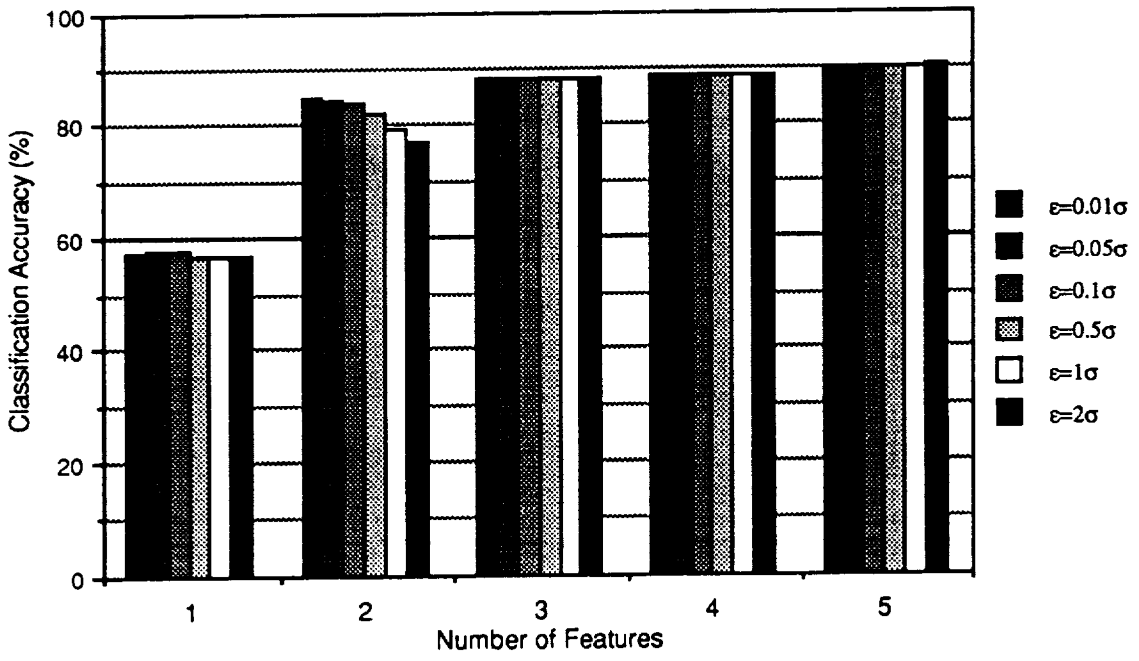

Figure 4.22 Effect of inaccurate estimation of decision boundary on the performance of the decision boundary feature extraction method.

4.5.2 Effect of the Parzen Scale Parameter h in Estimating Normal Vectors

Since normal vectors are estimated using equation (4.6), the Parzen scale parameter h will affect the estimation of normal vectors. Since normal vectors are used to estimate the decision boundary feature matrix, the Parzen scale parameter will affect the performance of the decision boundary feature extraction method. In the following test, we estimated the normal vectors using various Parzen scale parameters and investigate the effect of the Parzen scale parameter on the performance of the decision boundary feature extraction method. The decision boundary feature extraction method is applied to 18 dimensional data. With 18 features the classification accuracy is 92.9%. Figure 4.23 shows the performance comparison for various Parzen scale parameters in estimating normal vectors. When h=0.3, 0.5, 0.7, and 1.0, the classification accuracies with 3 features are 92.6%, 92.3%, 92.2%, and 92.1%, respectively. As larger Parzen scale parameters are used (h ≥ 2), classification accuracies decrease, though the decreasing rate is relatively small. However, if the Parzen scale parameter is too small (h=0.1), the classification accuracy decreases considerably. Overall, the Parzen scale parameters between 0.5 and 1.0 give best results in this case. Although the performance of the decision boundary feature extraction method does not seem to be very sensitive to the variation of the Parzen scale parameter, care must be taken that the Parzen scale parameter should not be too small or too large for a given data.

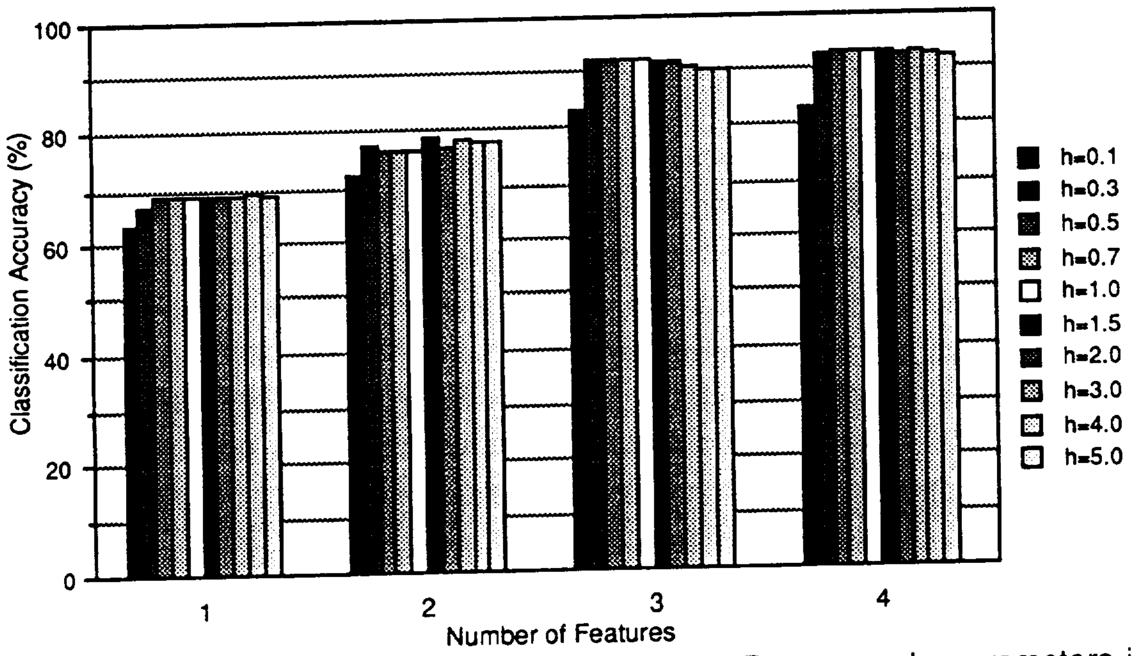

Figure 4.23 Performance comparison for various Parzen scale parameters in estimating normal vectors.

4.6 Conclusion

Decision Boundary Feature Extraction is a new feature extraction technique which is derived from the fact that all the feature vectors needed in discriminating between classes for a given classifier can be obtained from the decision boundary defined by the given classifier. Instead of utilizing class mean differences or class covariance differences, the method utilizes the decision boundary directly. As a result, the method does not deteriorate under the circumstances of equal means or equal covariances, and can be used for both parametric and non-parametric classifiers. In this chapter we proposed a decision boundary feature extraction algorithm for non-parametric classifiers. By directly utilizing the decision boundary defined by an employed non-parametric classifier without any assumption about the distribution of data, the proposed feature selection algorithm can take advantage of the generality of the non-parametric classifier, which can define a complex decision boundary. The experiments show that the performance of the proposed algorithm is very promising. The importance of such algorithms is enhanced as the use of non-parametric classifiers such as neural networks continues to grow (Lee and Landgrebe 1992-2, Lee and Landgrebe 1992-3).

Compared with the conventional feature extraction/selection algorithms, the proposed algorithm predicts the minimum number of features to achieve the same classification accuracy as in the original space and at the same time finds the needed feature vectors which have a direct relationship with classification accuracy. Unlike some of the conventional extraction algorithms using the lumped covariance, the proposed algorithm takes full advantage of the information contained in class covariance differences by extracting new features directly from the decision boundary. Since the information contained in the second order statistics increases its importance in discriminating between classes in high dimensional data, the proposed algorithm also has potential for feature extraction for high dimensional data and multi-source data.

CHAPTER 5 DECISION BOUNDARY FEATURE EXTRACTION FOR NEURAL NETWORKS

5.1 Introduction

Although neural networks have been successfully applied in various fields [(Ersoy and Hong 1990) (McEliece et al. 1987) and (Fukushima and Wake 1991)], relatively few feature extraction algorithms are available for neural networks. A characteristic of neural networks is that they need a long training time but a relatively short classification time for test data. However, with more high dimensional data and multi-source data available, the resulting neural network can be very complex. Although once the networks are trained, the computational cost of neural networks is much smaller compared with other non-parametric classifiers such as the Parzen density estimator (Parzen 1962) and the kNN classifier (Cover and Hart 1967), the lack of efficient feature extraction methods inevitably will introduce some inefficient calculation into neural networks. For example, the number of multiplications needed to classify a test sample using a 2 layer feedforward neural network which has 20 input neurons, 60 hidden neurons (assuming that the number of hidden neurons is three times the number of input neurons), and 3 output neurons is given by

$$20*60 + 60*3 = 1,380$$

Figure 5.1 illustrates an example of the hardware implementation of the original data set.

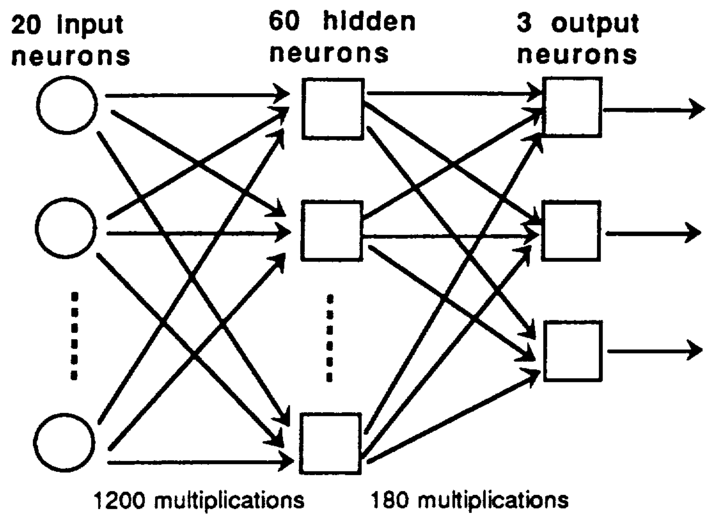

Figure 5.1 Hardware implementation of the original data set.

Assuming that it is possible to obtain about the same performance with 5 features selected by a good feature extraction method, the number of multiplications needed to classify a test sample can be reduced to

$$20*5 + 5*15 + 15*3 = 220$$

Figure 5.2 illustrates an example of the hardware implementation of the reduced data set. The first 100 (=20*5) multiplications are needed to calculated the 5 features from the original 20 dimensional data. In this example, the reduction ratio is 220/1380 ≈ 0.16. The reduction ratio will increase as the number of hidden layers and the number of hidden neurons increase. Thus, by employing a good feature extraction method, the resulting network can be much faster and simpler. If the neural network is to be implemented in a serial computer, classification time can be substantially reduced. If the neural network is to be implemented in hardware, the complexity of the hardware can be substantially reduced since the complexity of the hardware is proportional to the number of neurons and multiplications (connections between neurons). Hardware implementation of neural networks is an important topic [(Moonpenn et al. 1987), (Yasunaga et al. 1991), and (Fisher et al. 1991)]. In order to integrate a neural network on a single chip, it is important to reduce the number of neurons. The proposed method can be used in such a case, reducing the complexity of the network.

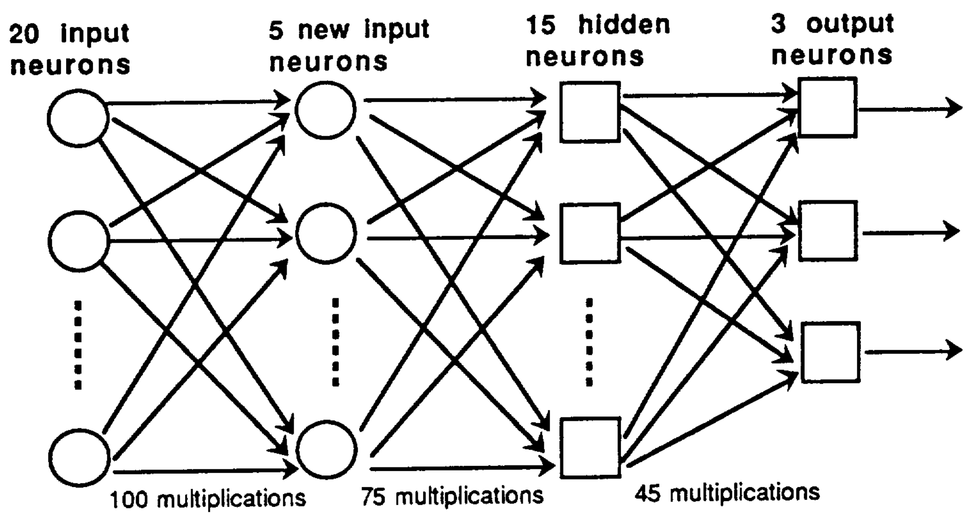

Figure 5.2 Hardware implementation of the reduced data set.

Neural networks are distribution free and can define arbitrary decision boundaries, and it is desirable that a feature extraction method for neural networks can preserve that characteristic. In this chapter, we apply the decision boundary feature extraction method to neural networks. First, we propose a feature extraction method for neural networks using the Parzen density estimator. In that method, we first select features using the Parzen density estimator employing the decision boundary feature extraction method. Then we use the selected features to train a neural network. Using a reduced feature set, we attempt to reduce the training time of a neural network and obtain a simpler neural network, further reducing the classification time for test data.

Finally, we apply directly the decision boundary feature extraction algorithm to neural networks (Lee and Landgrebe 1992-3). By directly applying the decision boundary feature extraction algorithm to neural networks, there will be no saving in training time. However, we will obtain a simpler network with better performance.

5.2 Neural Networks

5.2.1 Network Configurations

We will briefly discuss the neuron and the structure of a 2-layer feedforward neural network which will be used in the experiments. Backpropagation is used to train the network. Figure 5.3 shows an example of the neuron (Wasserman 1989).

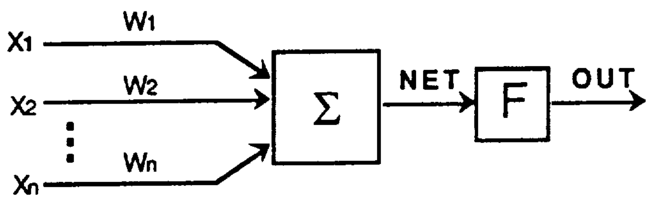

Figure 5.3 Artificial neuron with activation function.

A set of inputs each is multiplied by a weight, and the products are summed. Next an activation function F is applied to the summation, producing the signal OUT as follows:

$$OUT = F(NET) = \frac{1}{(1+e^{-NET})} \tag{5.1}$$

$$\text{where } NET = \sum_{i=1}^{n} x_i w_i$$

In the above example, the sigmoid function is used for the activation function. Figure 5.4 shows a 2 layer neural network (input layer, hidden layer, and output layer) with 2 outputs (OUT_1 and OUT_2). In Figure 5.4, let **X** be the input vector (1 by N) and let **Y** be the output vector (1 by M) of the hidden layer. Then

$$\mathbf{Y} = F(\mathbf{W}_i\mathbf{X}) \tag{5.2}$$

where **X** and **Y** are column vectors and $\mathbf{W}_i$ is a weight matrix (M by N) for the input vector.

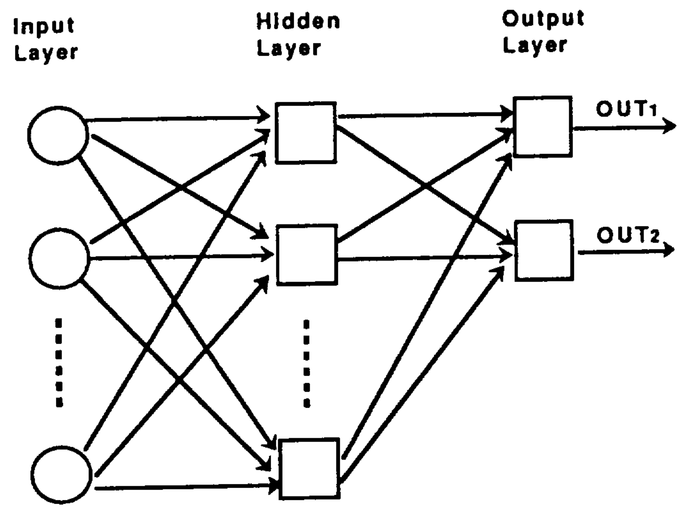

Figure 5.4 An example of 2 layer feedforward neural networks (2 pattern classes).

Then OUT1 and OUT2 can be expressed as follows:

$$\mathrm{OUT}_1(\mathbf{X}) = F(\mathbf{W}_h^1\mathbf{Y}) = F(\mathbf{W}_h^1 F(\mathbf{W}_i\mathbf{X})) \tag{5.3}$$
$$\mathrm{OUT}_2(\mathbf{X}) = F(\mathbf{W}_h^2\mathbf{Y}) = F(\mathbf{W}_h^2 F(\mathbf{W}_i\mathbf{X})) \tag{5.4}$$

where $\mathbf{W}_h^1$ and $\mathbf{W}_h^2$ are weight vector (M by 1) for the output vector of the hidden neurons. The decision rule is to select the class corresponding to the output neuron with the largest output (Lippmann 1987).

5.2.2 Backpropagation

The backpropagation algorithm is used to train the neural network [(Wasserman 1989) and (Hertz et al. 1991)] in the experiments. In the training phase, the weight changes are made by

$$\Delta w_{pq,k} = \eta\ \delta_{q,k}\ \mathrm{OUT}_{p,j}$$

where

η = learning rate

$\delta_{q,k}$ = the value of δ for neuron q in the layer k

$\mathrm{OUT}_{p,j}$ = the value of OUT for neuron p in the layer j.

The hidden layers are trained by propagating the output error back through the network layer by layer, adjusting weights at each layer.

However, it is noted that the decision boundary feature extraction algorithm can be used for neural networks regardless of training algorithms. Any other training algorithm can be employed.

5.3 Feature Extraction for Neural Networks Using the Parzen Density Estimator

5.3.1 Neural Networks and the Parzen Density Estimator

An advantage of non-parametric classifiers is that they can define arbitrary decision boundaries without any assumption on underlying densities. If underlying densities are unknown or problems involve complex densities which can not be approximated by common parametric density functions, use of a non-parametric classifier may be necessary. Some of the most widely used non-parametric classifiers include the Parzen density estimator, the kNN classifier, and neural networks. Recently, Neural network classifiers have been applied to various fields and demonstrated to be attractive alternatives to conventional classifiers (Benediktsson et al. 1990). One of the characteristics of neural networks is a long training time. However, once networks are trained, classification for test data can be done relatively fast.

In this section, we propose a feature extraction method for neural networks using the Parzen density estimator. We first select a new feature set using the decision boundary feature extraction algorithm for non-parametric classification in Chapter 4. By using the Parzen density estimator for feature extraction, we attempt to preserve the non-parametric characteristics of neural networks. Then the selected features are used to train neural networks. Using a reduced feature set, we attempt to reduce the training time of neural networks and obtain simpler neural network, further reducing the classification time for test data. Figure 5.5 shows an illustration of the proposed method.

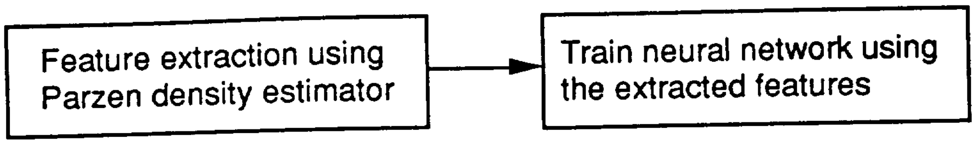
Feature extraction using Parzen density estimator
Train neural network using the extracted features

Figure 5.5 Feature extraction for neural networks using Parzen density estimation.

5.3.2 Experiments

5.3.2.1 Experiments with generated data

In order to evaluate closely how the proposed algorithm performs under various circumstances, tests are conducted on generated data with given statistics.

Example 5.1 In this example, class ω_1 is normal with the following statistics:

$$\mathbf{M}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \Sigma_1 = \begin{bmatrix} 3 & 0.5 \\ 0.5 & 3 \end{bmatrix}$$

Class ω_2 is equally divided between two normal distributions with the following statistics:

$$\mathbf{M}_2^1 = \begin{bmatrix} -3 \\ 3 \end{bmatrix} \quad \Sigma_2^1 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix} \quad \text{and} \quad \mathbf{M}_2^2 = \begin{bmatrix} 3 \\ -3 \end{bmatrix} \quad \Sigma_2^2 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix}$$

400 samples are generated for each class. Figure 5.6 shows the distribution of the data along with the decision boundary found by the proposed procedure numerically. Eigenvalues λ_i and eigenvectors ϕ_i of Σ_{EDBFM} are calculated as follows:

$$\lambda_1 = 0.98338, \quad \lambda_2 = 0.01662 \qquad \phi_1 = \begin{bmatrix} 0.69 \\ -0.72 \end{bmatrix}, \quad \phi_2 = \begin{bmatrix} 0.72 \\ 0.69 \end{bmatrix}$$

Since one eigenvalue is significantly larger than the other, it can be said that the rank of Σ_{EDBFM} is 1. That means only one feature is needed to achieve the same classification accuracy as in the original space. Considering the statistics of the two classes, the rank of Σ_{EDBFM} gives the correct number of features needed to achieve the same classification accuracy as in the original space. The selected features are used to train neural networks.

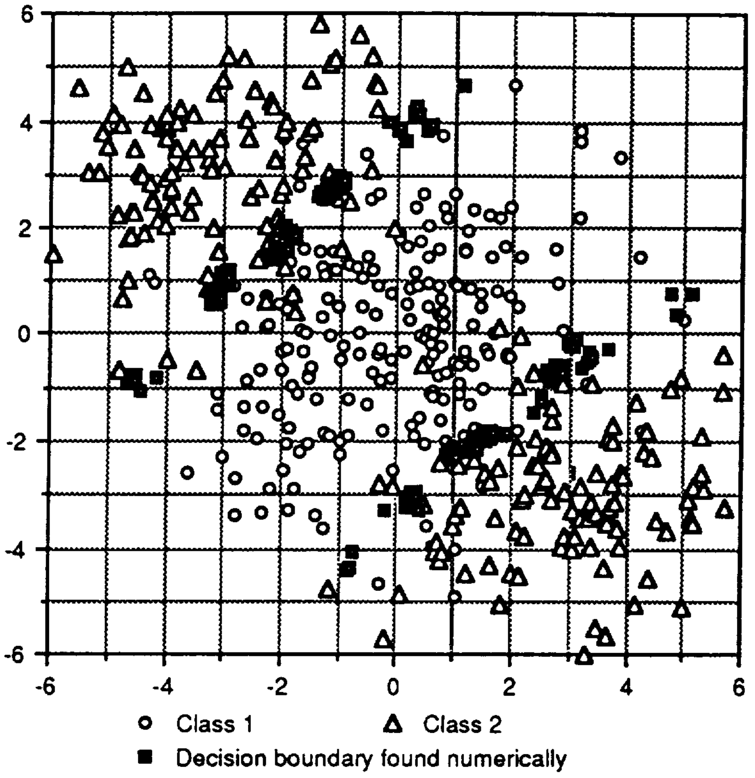

Figure 5.6 Data distribution of Example 5.1. The decision boundary found by the proposed procedure is also shown.

Table 5.1 shows the classification accuracies of the Parzen Density Estimator and neural networks. With one feature, the Parzen density estimator achieves about the same classification accuracy as could be obtained in the original 2-dimensional space. Likewise, the neural network achieves about the same classification accuracy with one feature selected by the proposed algorithm.

Table 5.1 Classification accuracies of the Parzen density estimator and neural networks.

Number of Features	Parzen Density Estimator	Neural Networks
1	91.4 (%)	91.6 (%)
2	91.6 (%)	90.9 (%)

Figure 5.7 shows a graph of the classification accuracies vs. the number of iterations. When one feature is used, the network converged after about 40 iterations. When two features are used, the network converged after about 70 iterations.

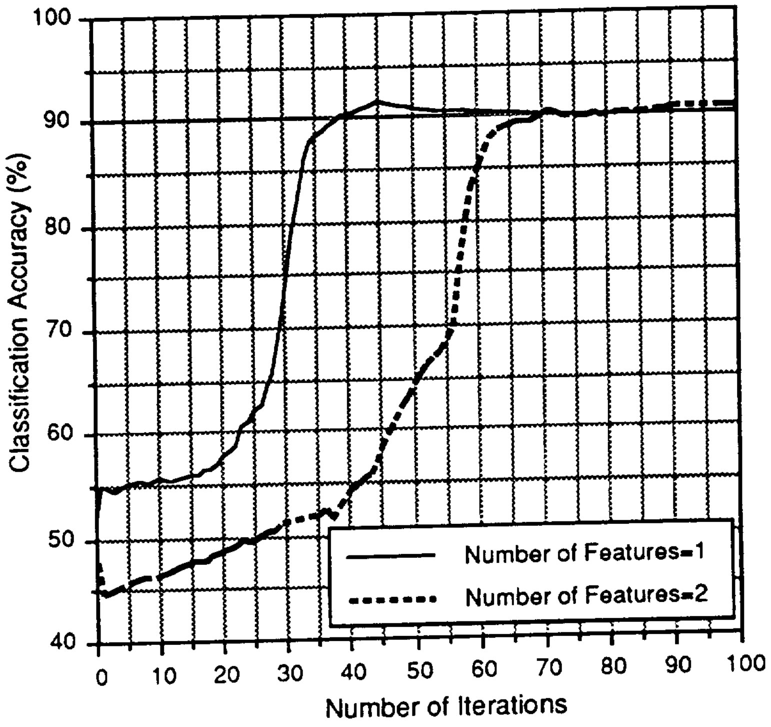

Figure 5.7 Classification accuracies vs. the number of iterations.

Example 5.2 In this example, there are 3 classes. Class ω_1 is normal with the following statistics:

$$\mathbf{M}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

Class ω_2 is equally divided between two normal distributions with the following statistics:

$$\mathbf{M}_2^1 = \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix} \quad \Sigma_2^1 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 9 \end{bmatrix} \text{ and } \mathbf{M}_2^2 = \begin{bmatrix} -5 \\ 0 \\ 0 \end{bmatrix} \quad \Sigma_2^2 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

And class ω_3 is equally divided between two normal distributions with the following statistics:

$$M_3^1 = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \Sigma_3^1 = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 9 \end{bmatrix} \text{ and } M_3^2 = \begin{bmatrix} 0 \\ -5 \\ 0 \end{bmatrix} \Sigma_3^2 = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

The distributions of these classes are shown as ellipses of concentration in Figure 5.8 in the x1-x2 plane.

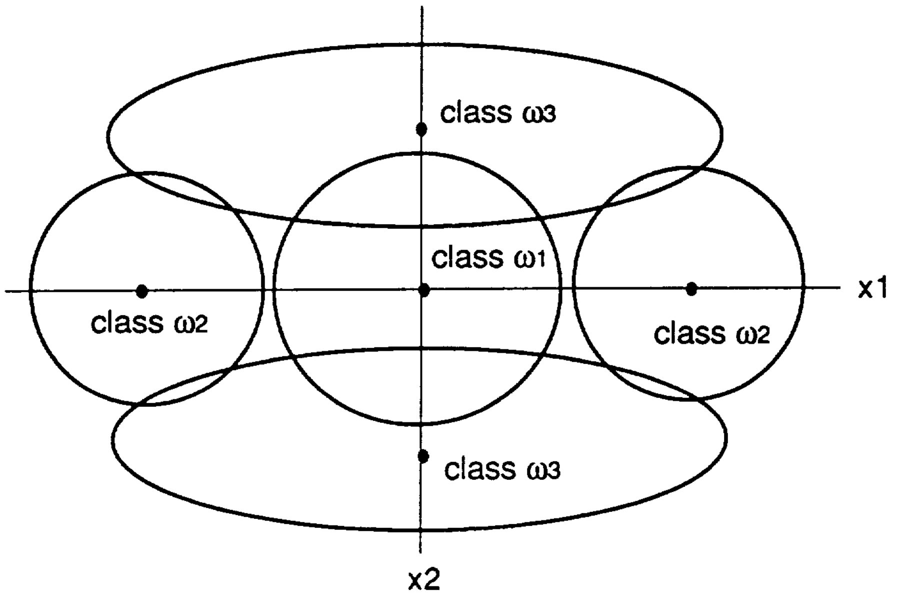

Figure 5.8 The distributions of Example 5.2 are shown as eclipse of concentrations.

Table 5.2 shows the classification accuracies of the Parzen density estimator and neural networks. With two features, the Parzen density estimator achieves about the same classification accuracy as could be obtained in the original 3-dimensional space. Table 5.2 also shows the classification accuracies of the neural network. The proposed feature extraction method for neural networks using the Parzen density estimator finds the correct 2 features, achieving about the same classification as could be obtained using the original 3-dimensional data.

Table 5.2 Classification accuracies of the Parzen density estimator and neural network.

Number of Features	Parzen Density Estimator	Neural Networks
1	65.0 (%)	64.8 (%)
2	84.8 (%)	84.3 (%)
3	84.8 (%)	84.0 (%)

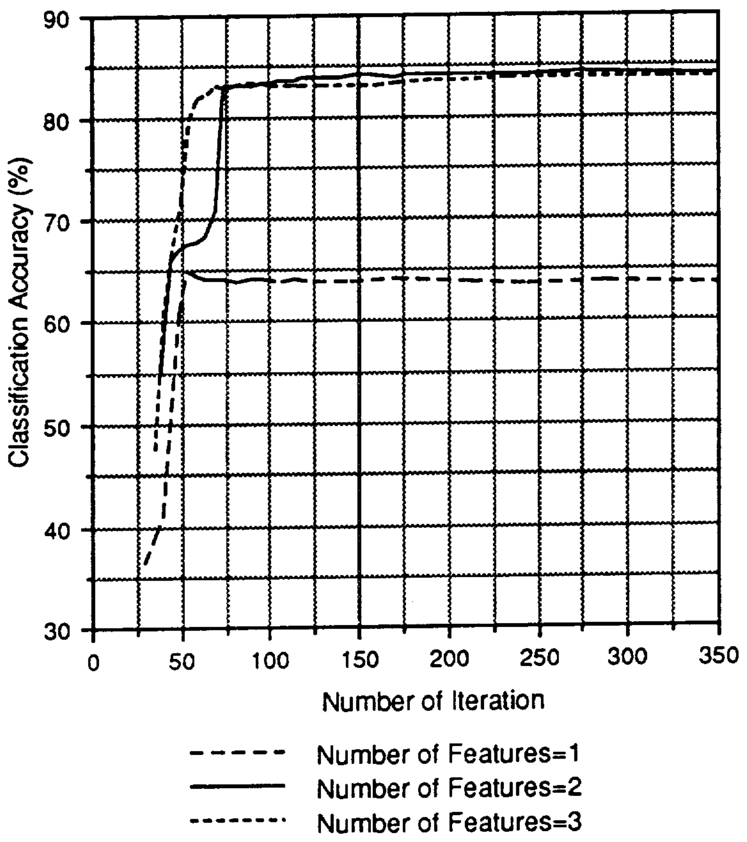

Figure 5.9 Classification accuracies vs. the number of iterations.

Figure 5.9 shows a graph of the classification accuracies vs. the number of iterations. When one feature is used, the network essentially converged after about 50 iterations. When two features are used, the classification accuracies are almost saturated after about 75 iterations. After 150 iterations, the classification accuracy is about 84%. When three features are used, the network converged after about 75 iterations, achieving about 84% classification accuracy.

5.3.2.2 Experiments with real data

Experiments were done using FSS (Field Spectrometer System) data which has 60 spectral bands (Biehl et al. 1982). To evaluate the performance of the proposed method, two other feature selection/extraction algorithms, Uniform Feature Design (see Section 3.6.2.1) and Principal Component Analysis (the Karhunen-Loeve transformation) are tested to evaluate and compare the performance of the proposed algorithm.

In order to test the performance in a multimodal situation, 3 classes with 2 subclasses were chosen. In other words, 2 subclasses were combined to form a new class, thus the data are purposely made multimodal. Table 5.3 provides information on the classes. In the experiment, 500 randomly selected samples from each class were used as training data and the rest were used as test data.

Table 5.3 Class description.

Class	Subclass	No. of Samples	Total No. of Sample
Class ω_1	Spring Wheat July 9, 1978	454	972
	Spring Wheat July 26, 1978	518	
Class ω_2	Winter Wheat June 26, 1977	677	1339
	Winter Wheat Oct. 18, 1977	662	
Class ω_3	Spring Wheat Oct. 26, 1978	441	910
	Spring Wheat Sep. 21, 1978	469	

First the original data are reduced to a 17 feature data set using Uniform Feature Design. Then, the Parzen density estimator is applied to the reduced data set and the decision boundary is calculated numerically. From the decision boundary, a decision boundary feature matrix is estimated and a new feature set is calculated from the decision boundary feature matrix.

Using the features selected by Parzen density estimator, neural networks are trained. In order to evaluate the performance of the proposed algorithm, two other feature sets selected by Uniform Feature Design and Principal Component Analysis are also used to train the network. The classification accuracies of training data and test data of the 1 through 10 dimensional data sets selected by the proposed algorithm are shown in Figures 5.10-11. As can be seen, the neural network using the proposed algorithm shows considerably better performances than the neural networks using Uniform Feature Design and Principal Component Analysis. In Figure 5.10, the 4-5 features selected by the proposed algorithm achieved about the same classification accuracy as can be obtained with the original 17 dimensional data.

Figures 5.12-13 show graphs of classification accuracy vs. number of iterations. From Figure 5.13, it can be said that the performances of the neural networks are saturated after about 100-200 iterations. The training time is proportional to the number of iterations and the square number of neurons. When the network is implemented in hardware, the complexity of the hardware will be proportional to the square number of neurons. As a result, by using the Parzen density estimator to select features for neural networks, one can reduce the training time and the complexity of the hardware implementation. For example, in Figure 5.13 (test data), the classification accuracy with 10 features is 96.0% and the classification accuracy with 4 features is 93.3%. The difference is 2.7%. If such a decrease in classification accuracy is acceptable, the training time can be reduced by the factor of 6.25. Furthermore, the classification time will be also reduced by the same factor when implemented in a serial computer. When implemented in hardware, the complexity of the hardware can be also reduced by the same factor.

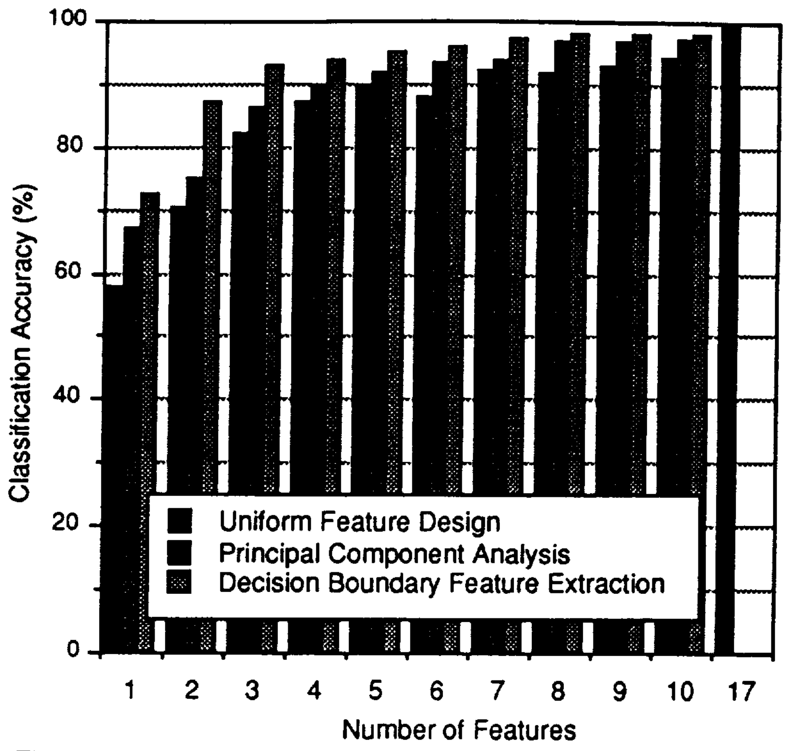

Figure 5.10 Performance comparison (training data).

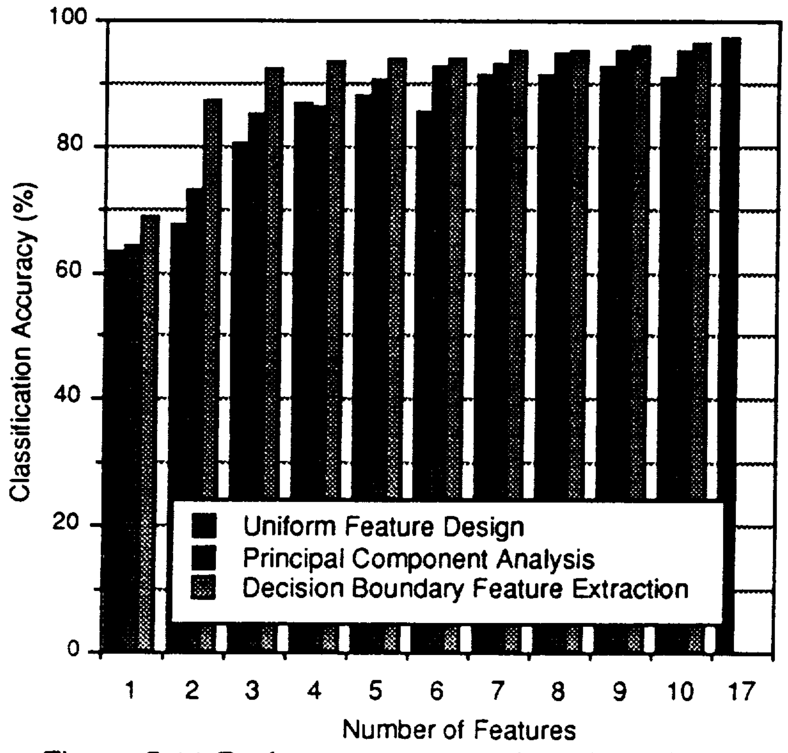

Figure 5.11 Performance comparison (test data).

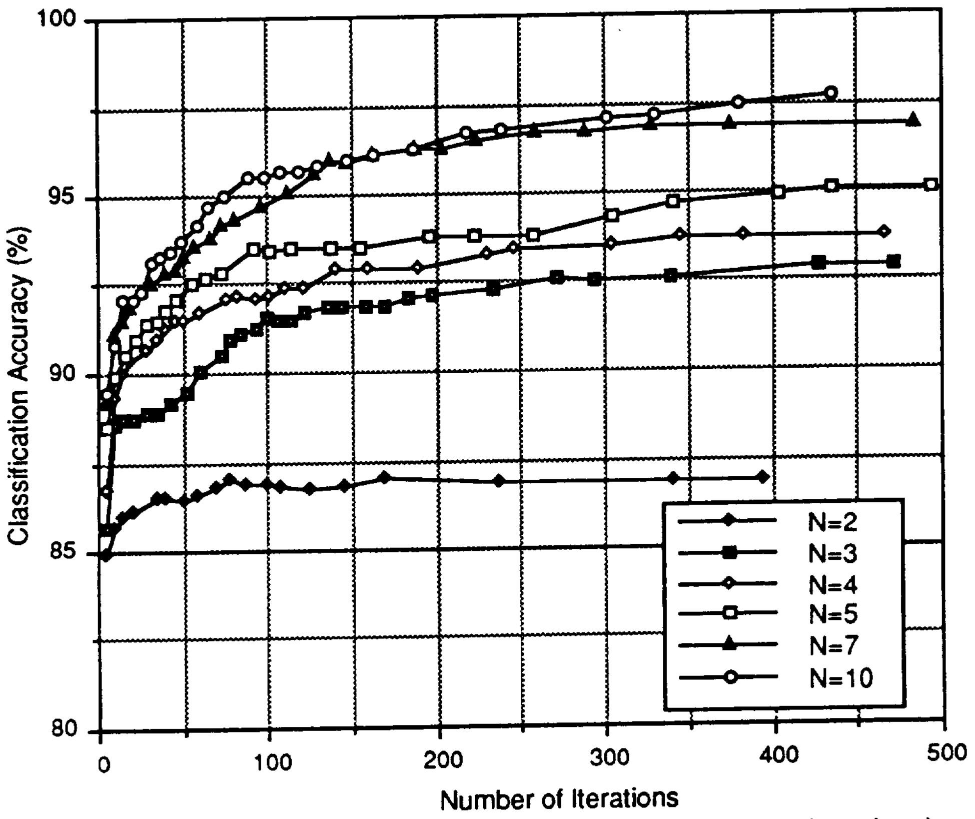

Figure 5.12 Iteration vs. classification accuracy (training data).

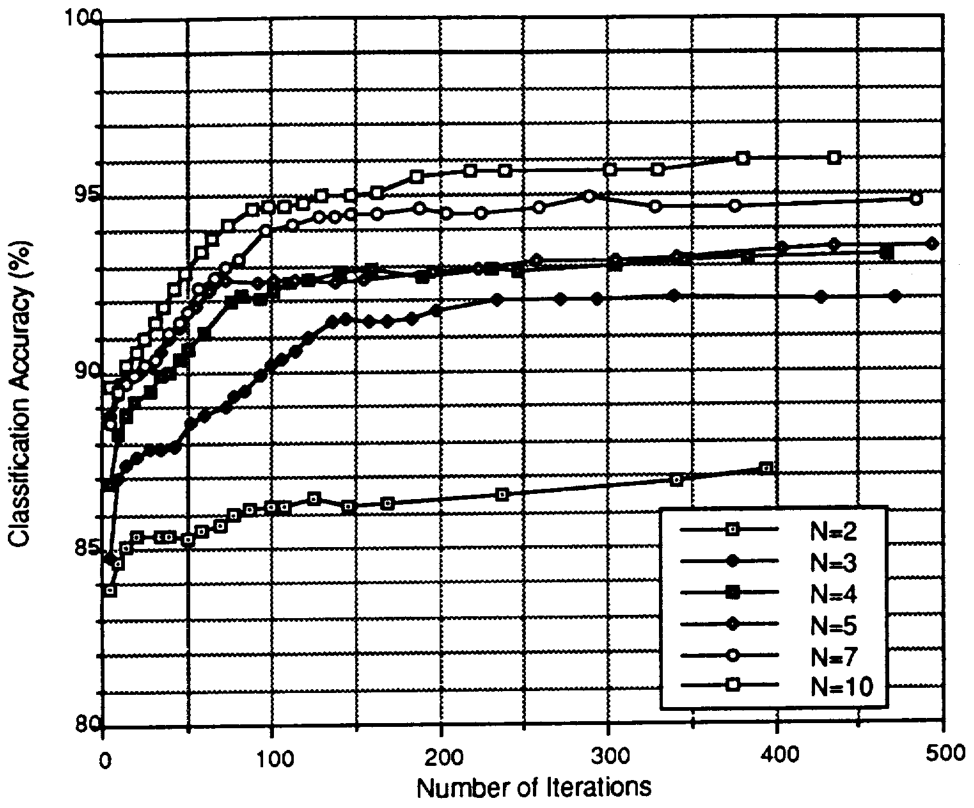

Figure 5.13 Iteration vs. classification accuracy (test data).

It is found that using the Parzen density estimator to select features for neural networks is not optimal in a sense that the performance can be improved if the decision boundary feature extraction method is directly applied to the neural network. In the following section, the decision boundary feature extraction method will be directly applied to the neural network. However, by directly applying the decision boundary feature extraction method to the neural network, there will be no saving in training time. On the other hand, using the Parzen density estimator to select features for neural networks results in reduction in training time.

5.4 Decision Boundary Feature Extraction for Neural Networks

5.4.1 Decision Boundaries in Neural Networks

In order to utilize the decision boundary feature extraction algorithm for neural networks, the decision boundary must be defined. We define the decision boundary in multi-layer feedforward neural networks as follows:

Definition 5.1 The decision boundary in a neural network for a two pattern class problem is defined as

$$\{ \mathbf{X} \mid OUT_1(\mathbf{X}) = OUT_2(\mathbf{X}) \} \text{ or} \tag{5.5}$$

where **X** is an input vector {See equations (5.3), (5.4), and (5.5)}.

In other words, the decision boundary of a two pattern class problem is defined as a locus of points on which $OUT_1(\mathbf{X}) = OUT_2(\mathbf{X})$ where **X** is an input vector. Let $h(\mathbf{X})=OUT_1(\mathbf{X}) - OUT_2(\mathbf{X})$ where **X** is an input vector to a neural network. Then the decision boundary can be defined as

$$\{ \mathbf{X} \mid h(\mathbf{X})=0 \} \tag{5.6}$$

The normal vector to the decision boundary at **X** will be given by

$$\nabla h(\mathbf{X}) = \frac{\partial h}{\partial x_1} \mathbf{x}_1 + \frac{\partial h}{\partial x_2} \mathbf{x}_2 + \cdots\cdots + \frac{\partial h}{\partial x_n} \mathbf{x}_n \tag{5.7}$$

Since the decision boundary in neural networks can not be expressed analytically, the term $\nabla h(\mathbf{X})$ must be calculated numerically as follows:

$$\nabla h(\mathbf{X}) \approx \frac{\Delta h}{\Delta x_1} \mathbf{x}_1 + \frac{\Delta h}{\Delta x_2} \mathbf{x}_2 + \cdots\cdots + \frac{\Delta h}{\Delta x_n} \mathbf{x}_n \tag{5.8}$$

5.4.2 Decision Boundary Feature Extraction Procedure for Neural Networks

Next we propose the following procedure for neural networks utilizing the decision boundary feature extraction algorithm.

Decision Boundary Feature Extraction Procedure for Neural Networks
(2 pattern class case)

STEP 1: Train the neural network using all features.

STEP 2: For each sample correctly classified as class ω_1, find the nearest sample correctly classified as class ω_2. Repeat the same procedure for the samples classified as class ω_2.

STEP 3: The lines connecting a pair of samples found in STEP 2 must pass through the decision boundary since the pair of samples are classified differently. By moving along the line, find the point on the decision boundary or near the decision boundary within a threshold.

STEP 4: At each point found in STEP 3, estimate the normal vector $\mathbf{N}_i$ by

$$\mathbf{N}_i = \nabla h(\mathbf{X}) / |\nabla h(\mathbf{X})|$$

where $\nabla h(\mathbf{X}) \approx \frac{\Delta h}{\Delta x_1} \mathbf{x}_1 + \frac{\Delta h}{\Delta x_2} \mathbf{x}_2 + \cdots\cdots + \frac{\Delta h}{\Delta x_n} \mathbf{x}_n$

$h(\mathbf{X}) = \mathrm{OUT}_1(\mathbf{X}) - \mathrm{OUT}_2(\mathbf{X})$ {See equation (5.5)}.

STEP 5: Estimate the decision boundary feature matrix using the normal vectors found in STEP 4.

$$\Sigma_{EDBFM} = \frac{1}{L} \sum_i \mathbf{N}_i \mathbf{N}_i^t$$

where L is the number of samples correctly classified

STEP 6: Select the eigenvectors of the decision boundary feature matrix as new feature vectors according to the magnitude of corresponding eigenvalues.

If there are more than 2 classes, the procedure can be repeated for each pair of classes after the network is trained for all classes. Then the total decision boundary feature matrix can be calculated by averaging the decision boundary feature matrix of each pair of classes. If prior probabilities are available, the summation can be weighted. That is, if there are M classes, the total decision boundary feature matrix can be calculated as

$$\Sigma_{DBFM} = \sum_{i}^{M} \sum_{j,\, j \neq i}^{M} P(\omega_i) P(\omega_j) \Sigma_{DBFM}^{ij}$$

where Σ^{ij}_{DBFM} is a decision boundary feature matrix between class ω_i and class ω_j and $P(\omega_i)$ is the prior probability of class ω_i if available. Otherwise let $P(\omega_i)=1/M$.

5.4.3 Experiments

5.4.3.1 Experiments with generated data

In order to evaluate closely how the proposed algorithm performs under various circumstances, tests are conducted on generated data with given statistics.

Example 5.3. In this example, class ω_1 is normal with the following statistics:

$$\mathbf{M}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma_1 = \begin{bmatrix} 3 & 0.5 \\ 0.5 & 3 \end{bmatrix}$$

And class ω_2 is equally divided between two normal distributions with the following statistics:

$$\mathbf{M}_2^1 = \begin{bmatrix} -3 \\ 3 \end{bmatrix} \Sigma_2^1 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix} \text{ and } \mathbf{M}_2^2 = \begin{bmatrix} 3 \\ -3 \end{bmatrix} \Sigma_2^2 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix}$$

200 samples are generated for each class. Figure 5.14 shows the distribution of the data along with the decision boundary found by the proposed procedure numerically. Eigenvalues λ_i and eigenvectors ϕ_i of Σ_{EDBFM} are calculated as follows:

$$\lambda_1 = 0.98105, \lambda_2 = 0.01895 \qquad \phi_1 = \begin{bmatrix} 0.72 \\ -0.69 \end{bmatrix}, \qquad \phi_2 = \begin{bmatrix} 0.69 \\ 0.72 \end{bmatrix}$$

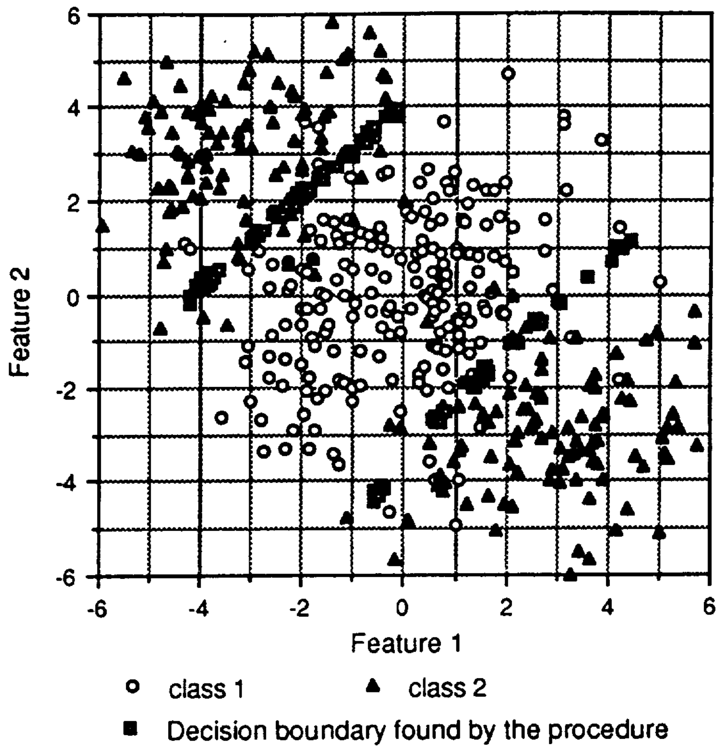

Figure 5.14 Data distribution and the decision boundary found by the proposed procedure.

Since one eigenvalue is significantly larger than the other, it can be said that the rank of Σ_{EDBFM} is 1. That means only one feature is needed to achieve the same classification accuracy as in the original space. Considering the statistics of the two classes, the rank of Σ_{EDBFM} gives the correct number of features needed to achieve the same classification accuracy as in the original space. With the original 2 features, the classification accuracy is about 90.8%. Table 5.4 shows the classification accuracies of the decision boundary feature extraction method. As can be seen, the decision boundary feature extraction method finds the right feature, achieving about the same classification accuracy with one feature. Figure 5.15 shows classification accuracies vs. number of iterations.

Table 5.4 Classification accuracies of Decision Boundary Feature Extraction of Example 5.3.

Number of Features	Classification Accuracy
1	91.6 (%)
2	90.9 (%)

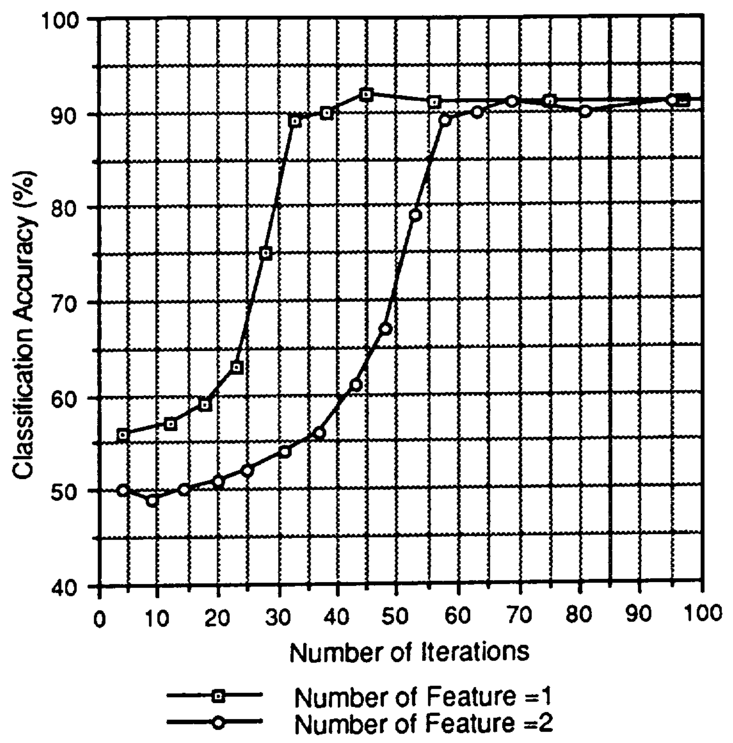

Figure 5.15 Iteration vs. classification accuracy.

Example 5.4 In this example, there are 3 classes. Class ω_1 is normal with the following statistics:

$$M_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

And class ω_2 is equally divided between two normal distributions with the following statistics:

$$M_2^1 = \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix} \quad \Sigma_2^1 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 9 \end{bmatrix} \text{ and } M_2^2 = \begin{bmatrix} -5 \\ 0 \\ 0 \end{bmatrix} \quad \Sigma_2^2 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

And class ω_3 is equally divided between two normal distributions with the following statistics:

$$M_3^1 = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \quad \Sigma_3^1 = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 9 \end{bmatrix} \text{ and } M_3^2 = \begin{bmatrix} 0 \\ -5 \\ 0 \end{bmatrix} \quad \Sigma_3^2 = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

The distributions of these classes are shown in Figure 5.16 in the x1-x2 plane, along with the decision boundary found by the procedure. Eigenvalues λ_i and eigenvectors ϕ_i of Σ_{EDBFM} are calculated as follows:

$$\lambda_1 = 0.59957\ ,\ \lambda_2 = 0.40027,\ \lambda_3 = 0.00015$$

$$\phi_1 = \begin{bmatrix} 0.06 \\ 1.00 \\ -0.02 \end{bmatrix},\ \phi_2 = \begin{bmatrix} -1.00 \\ 0.06 \\ 0.02 \end{bmatrix},\ \phi_3 = \begin{bmatrix} 0.02 \\ 0.02 \\ 1.00 \end{bmatrix}$$

$$\text{Rank}(\Sigma_{EDBFM}) \approx 2$$

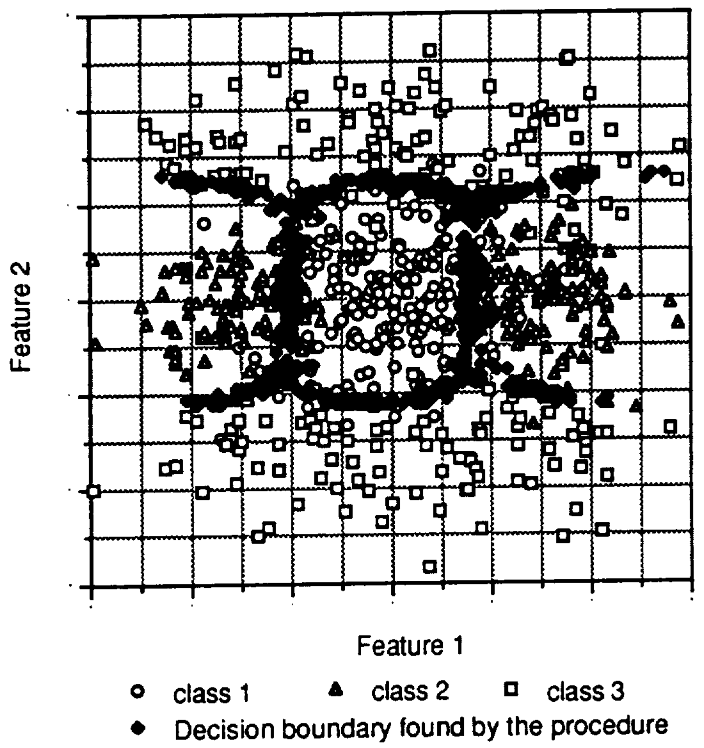

Figure 5.16 Data distribution and the decision boundary found by the proposed procedure.

With the original 3 features, the classification accuracy is about 85.7%. Table 5.5 shows the classification accuracies of the decision boundary feature extraction method. As can be seen, the decision boundary feature extraction method finds the right two features. Figure 5.17 shows a graph of the classification accuracies vs. the number of iterations.

Table 5.5 Classification accuracies of Decision Boundary Feature Extraction of Example 5.4.

No. Features	Classification Accuracy
1	62.8 (%)
2	85.7 (%)
3	85.8 (%)

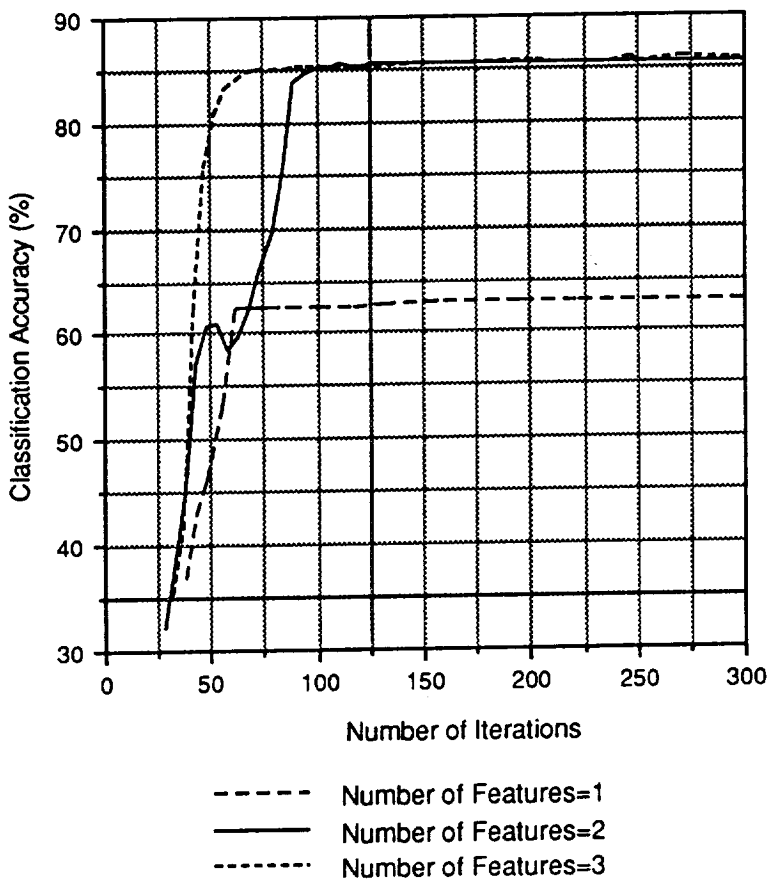

Figure 5.17 Iteration vs. classification accuracy.

5.4.3.2 Experiments with real data

Experiments were done using FSS (Field Spectrometer System) data which has 60 spectral bands (Biehl et al. 1982). Along with the proposed algorithm, three other feature extraction algorithms, Uniform Feature Design (see Section 3.6.2.1) and the Karhunen-Loeve transformation (Principal Component Analysis), and Discriminant Analysis (Fukunaga 1990) are tested to evaluate to evaluate the performance of the proposed algorithm.

In the following test, 4 classes were chosen from the FSS data. Table 5.6 provides information on the 4 classes. 400 randomly selected samples from each classes are used as training data and the rest are used for test.

Table 5.6 Class description.

SPECIES	DATE	No. of Samples
Winter Wheat	May 3, 1977	657
Unknown Crops	May 3, 1977	678
Winter Wheat	Mar. 8, 1977	691
Unknown Crops	Mar. 8, 1977	619

First the original 60 dimensional data was reduced to 17 dimensional data using Uniform Feature Design. And the decision boundary feature extraction method, Discriminant Analysis, and Principal Component Analysis were applied to the 17 dimensional data. Figures 5.18 and 5.19 show the classification results of training data and test data. With the 17 dimensional data, one can achieve about 97.6% classification accuracy for training data and about 94.4% classification accuracy for test data. The decision boundary feature extraction method achieves about the same classification accuracy for test data with just 3 features as can be seen in Figure 5.19. With 3 features, the decision boundary feature extraction method achieves about 92.2% classification accuracy for test data while Uniform Feature Design, Principal Component Analysis, and Discriminant Analysis achieve about 77.7%, 78.6%, 89.7%, respectively.

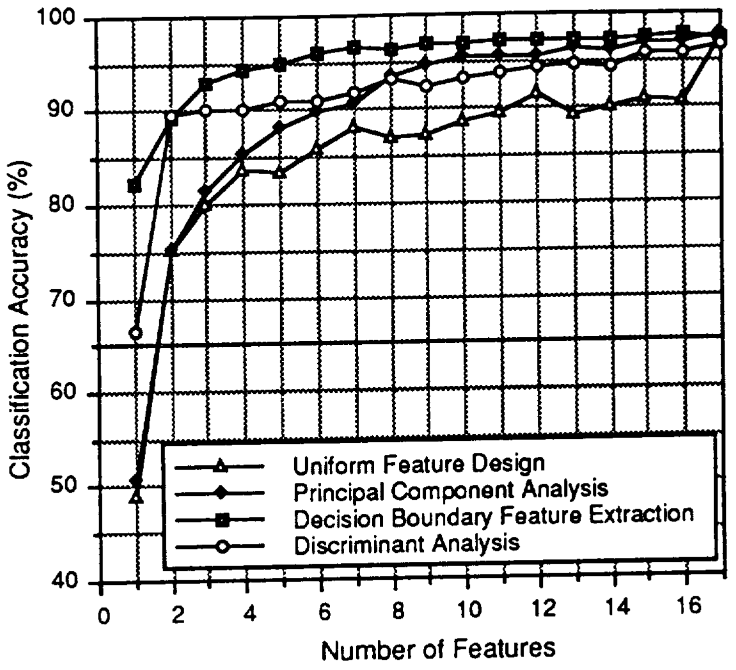

Figure 5.18 Performance comparison of the data of Table 5.6 (Train data).

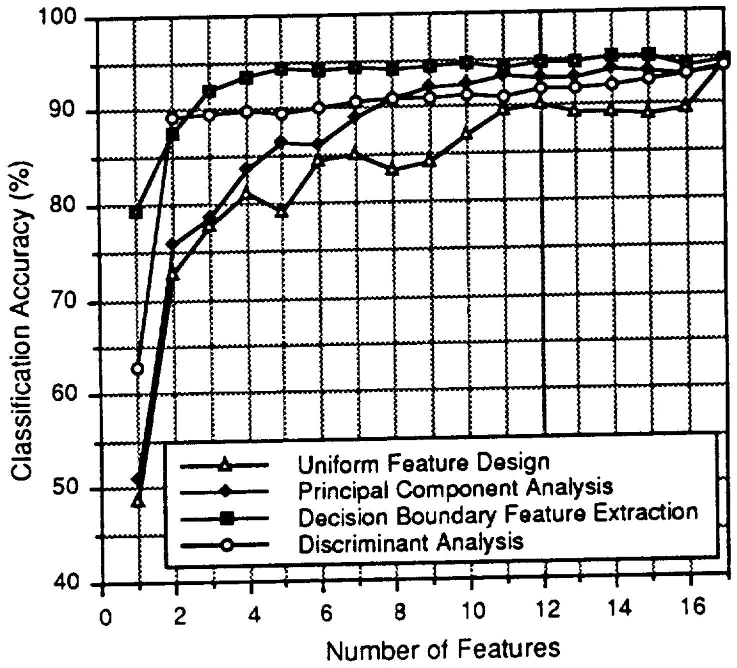

Figure 5.19 Performance comparison of the data of Table 5.6 (Test data).

In the next test, there are 3 classes and each class has 2 subclasses. In other words, 2 subclasses were combined to form a new class. By purposely combining data from different classes, the data are made to be multi-modal.

Table 5.7 provides information on the classes. Figure 4.13 (Chapter 4) shows a graph of the 6 subclasses and Figure 4.14 (Chapter 4) shows a graph of the 3 classes each of which has 2 subclasses. 500 randomly selected samples from each classes are used as training data and the rest are used for test.

Table 5.7 Class description.

Class	Subclass	No. of Samples	Total No. of Sample
Class ω_1	Winter Wheat March 8, 1977	691	1209
	Spring Wheat July 26, 1978	518	
Class ω_2	Winter Wheat June 26, 1977	677	1146
	Spring Wheat Sep. 21, 1978	469	
Class ω_3	Winter Wheat Oct. 18, 1977	662	1103
	Spring Wheat Oct. 26, 1978	441	

Figures 5.20-21 show the performance comparison. First the original 60 dimensional data was reduced to 17 dimensional data using Uniform Feature Design. With the 17 features, the classification accuracies of training data and test data are 99.9% and 95.6%, respectively. In low dimensionality ($N \leq 2$), Discriminant Analysis shows the best performances. However, the classification accuracies are much smaller than the maximum possible classification accuracies and the comparison seems irrelevant. When more than 2 features are used, the decision boundary feature extraction method outperforms all other methods. The decision boundary feature extraction method achieves about the same classification accuracy as could be obtained in the original 17-dimensional space with just 4 features. In particular, with 4 features, the classification accuracy of the decision boundary feature extraction method is about 92.4% while the classification accuracies of Uniform Feature Design, Principal Component Analysis, and Discriminant Analysis are 78.1%, 82.5%, and 82.3%, respectively.

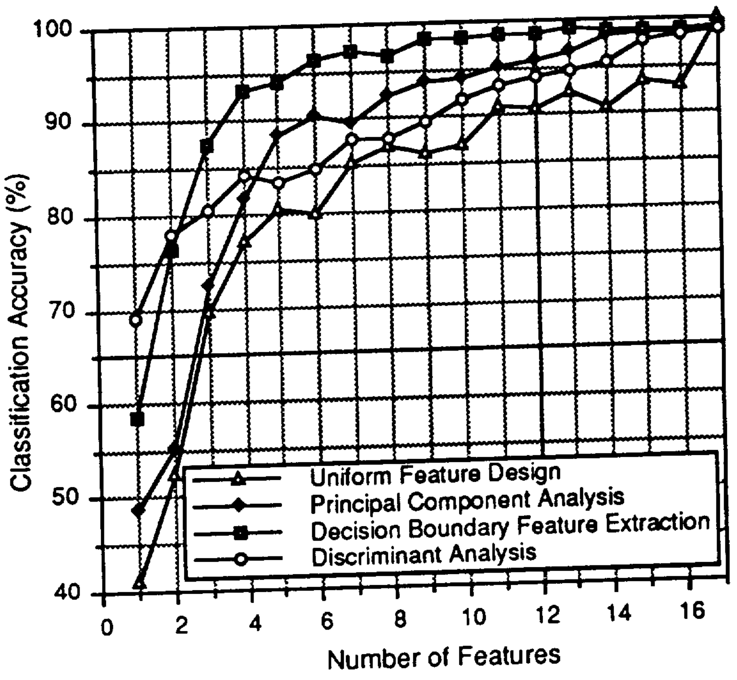

Figure 5.20 Performance comparison of the data of Table 5.7 (train data).

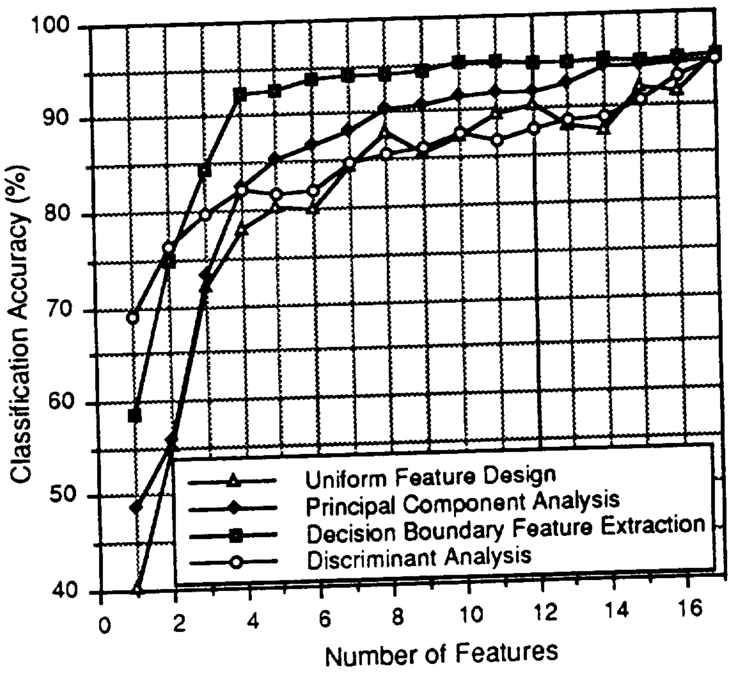

Figure 5.21 Performance comparison of the data of Table 5.7 (test data).

Figures 5.22-23 show graphs of classification accuracy vs. number of iterations of the decision boundary feature extraction method. As can be seen, the performances of neural networks are saturated after about 100 iterations. It can be also seen in Figure 5.23 that the performances are almost saturated when 4 features are used.

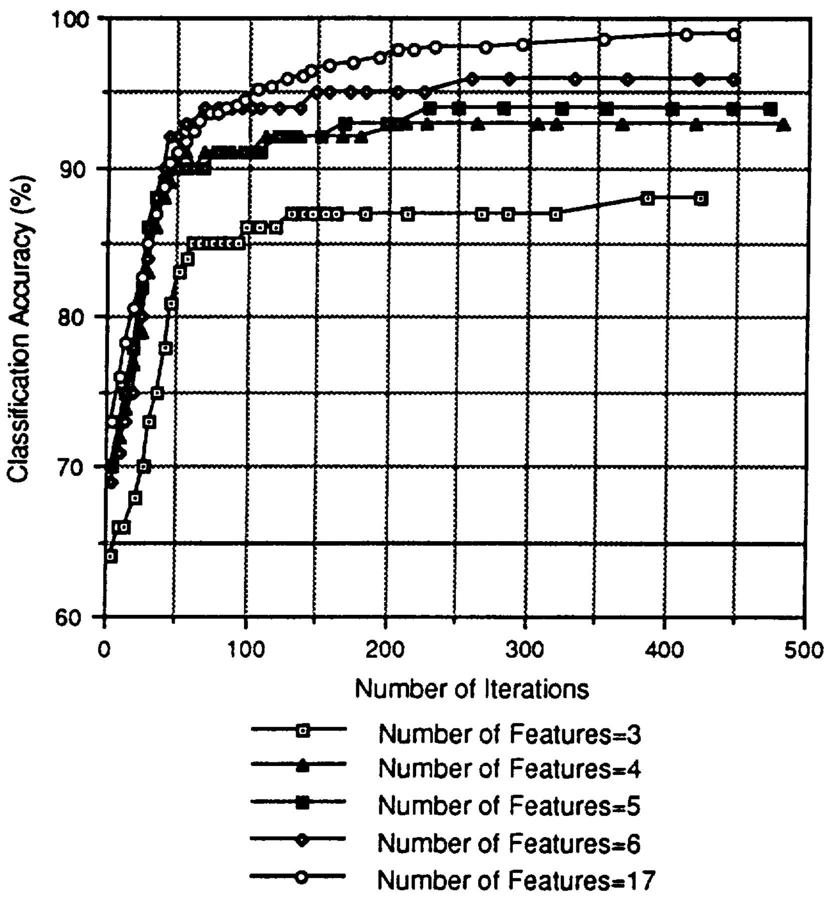

Figure 5.22 Iteration vs. classification accuracy (training data).

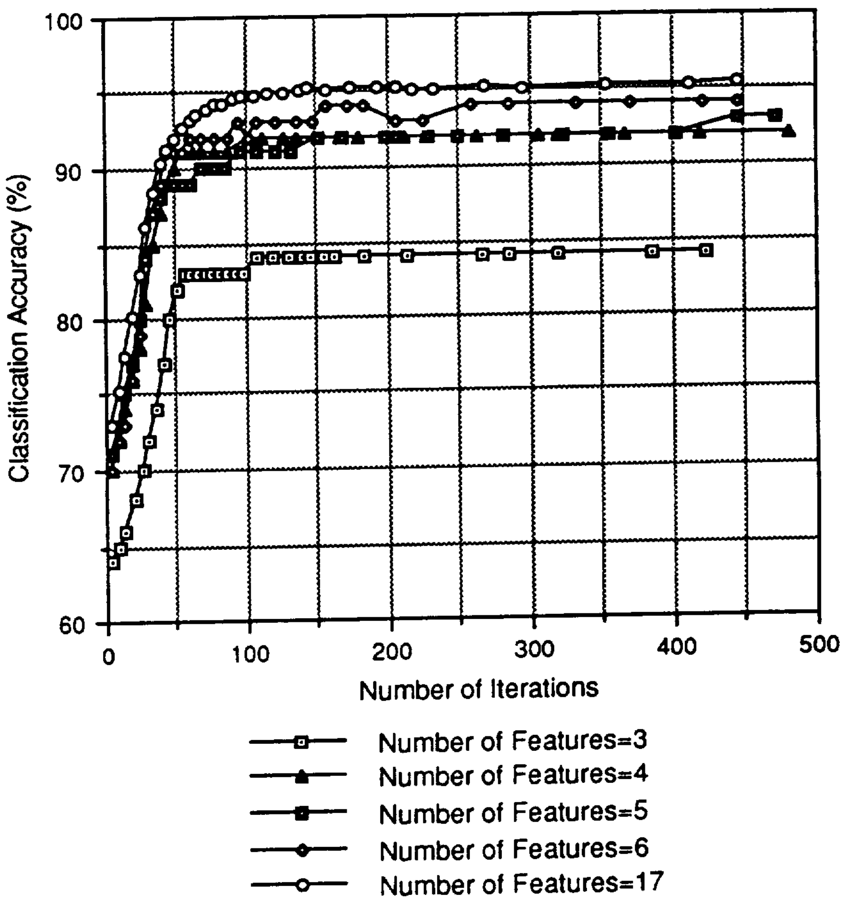

Figure 5.23 Iteration vs. classification accuracy (test data).

In the next test, there are 3 classes and each class has 2 subclasses. In other words, 2 subclasses were combined to form a new class. By purposely combining data from different classes, the data are made to be multi-modal. Table 5.8 provides information on the classes. 500 randomly selected samples from each classes are used as training data and the rest are used for test.

Table 5.8 Class description.

Class	Subclass	No. of Samples	Total No. of Sample
Class ω_1	Winter Wheat March 8, 1977	691	1145
	Spring Wheat July 9, 1978	454	
Class ω_2	Winter Wheat June 26, 1977	677	1195
	Spring Wheat July 26, 1978	518	
Class ω_3	Winter Wheat Oct. 18, 1977	662	1103
	Spring Wheat Oct. 26, 1978	441	

Figures 5.24-25 show the performance comparison. First the original 60 dimensional data was reduced to 17 dimensional data using Uniform Feature Design. And the decision boundary feature extraction method, Discriminant Analysis, and Principal Component Analysis were applied to the 17 dimensional data. With the 17 features, the classification accuracies of training data and test data are 97.3% and 96.7%, respectively. In low dimensionality ($N \leq 2$), Discriminant Analysis shows the best performances, though the difference between Discriminant Analysis and the decision boundary feature extraction method is small. However, when more than 2 features are used, the decision boundary feature extraction method outperforms all other methods. With 3 features, the decision boundary feature extraction method achieves about 95.6% classification accuracy for test data while Uniform Feature Design, Principal Component Analysis, and Discriminant Analysis achieve about 82.3%, 85.1%, and 90.8%, respectively.

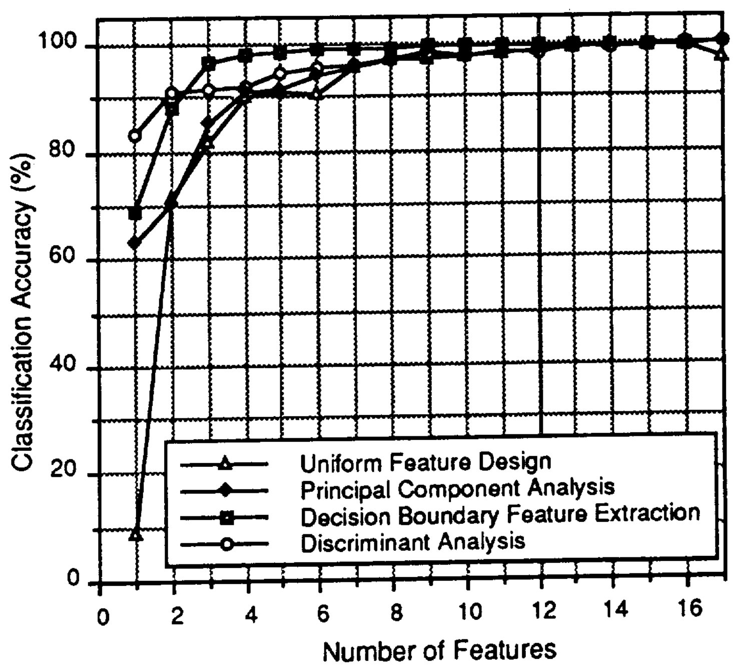

Figure 5.24 Performance comparison (training data).

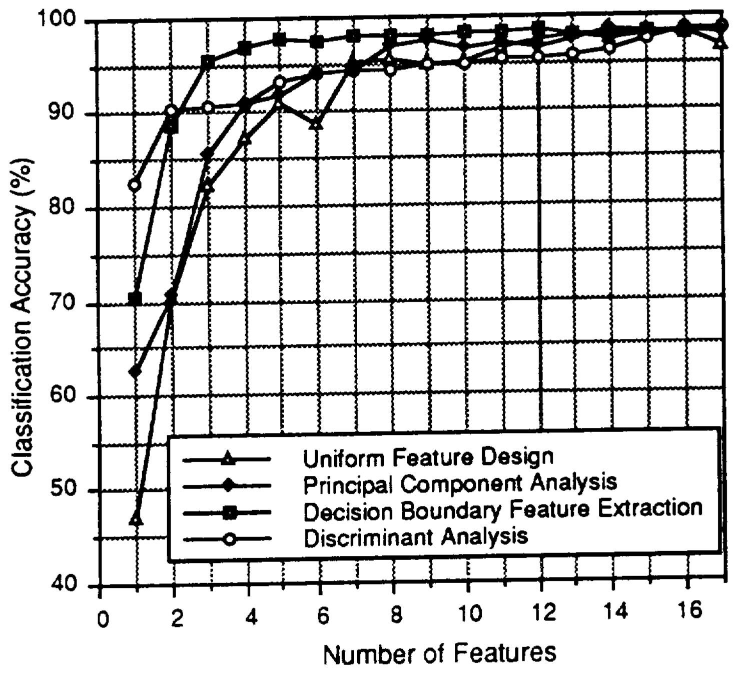

Figure 5.25 Performance comparison (test data).

5.5 Conclusion

In this chapter, we extended the decision boundary feature extraction method to neural networks. First we proposed the feature extraction algorithm for neural networks using Parzen density estimator (Figure 5.5). In this method, we first selected a new feature set using Parzen density estimator employing the non-parametric decision boundary feature extraction algorithm in Chapter 4. Then we used the reduced feature set to train neural networks. As a result, it would be possible to reduce training time and to obtain a simpler network because fewer features are used.

However, it is recognized that the characteristics of the Parzen density estimator and neural networks are not exactly the same. Thus, we applied the decision boundary feature extraction method directly to neural networks. We started by defining the decision boundary in a neural network. From the decision boundary, we estimated the normal vectors to the decision boundary, and the decision boundary feature matrix was calculated. From the decision boundary feature matrix, a new feature set was calculated. By directly applying the decision boundary feature extraction algorithm to neural networks, the performance was improved compared with using the Parzen density estimator for feature extraction. However, it is noted that by directly applying the decision boundary feature extraction algorithm to neural networks, there is no reduction in training time. In fact, the training time increased since we need to train two networks, one for the original feature set and the other for the reduced feature set.

The proposed algorithms preserve the nature of neural networks which can define a complex decision boundary and is able to take advantage of that nature. By employing the proposed algorithms, it is possible to reduce the number of features, and equivalently the number of neurons. This reduction results in much simpler networks and shorter classification time. When neural networks are to be implemented in hardware, the reduced number of neurons means a simpler architecture (Figures 5.1-2).

CHAPTER 6 DISCRIMINANT FEATURE EXTRACTION FOR PARAMETRIC AND NON-PARAMETRIC CLASSIFIERS

6.1 Introduction

In Chapter 3, the decision boundary feature extraction algorithm was developed where a new feature set is extracted from the decision boundary such that the classification result is preserved. The decision boundary feature extraction was applied to parametric classifiers (Chapter 3), to non-parametric classifiers (Chapter 4), and to neural networks (Chapter 5). In order to extract feature vectors from the decision boundary, the decision boundary feature matrix was defined which is constructed from the normal vectors to the decision boundary. In the decision boundary feature extraction techniques, we do not care whether the value of the discriminant function is changed or not, as long as the classification result remains the same.

In this chapter, the concept of decision boundary feature extraction algorithm is generalized such that feature extraction is considered as preserving the value of the discriminant function for a given classifier (Lee and Landgrebe 1992-4). And we consider feature extraction as eliminating features which have no impact on the value of the discriminant function and propose a feature extraction algorithm which eliminates those irrelevant features and retains only useful features. The proposed algorithm, referred as Discriminant Feature Extraction, can be used both for parametric and non-parametric classifiers and its performance does not deteriorate when there is no difference in mean vectors or no difference in covariance matrices.

Compared with the decision boundary feature extraction algorithms, Discriminant Feature Extraction will be less efficient for parametric classifiers where a good estimation of the decision boundary can be obtained. However, in some non-parametric classifiers, it is difficult or time-consuming to find the decision boundary. In such cases, Discriminant Feature Extraction could be a good alternative solution. Furthermore, by extracting features such that the value of interest is preserved, Discriminant Feature Extraction can be used not only for feature extraction for classification but also for feature extraction for any application. More detailed comparison will be made later.

6.2 Definitions and Theorems

We will briefly review Bayes' decision rule for minimum error. Let **X** be an observation in the N-dimensional Euclidean space E^N under hypothesis H_i: $\mathbf{X} \in \omega_i$ i=1,2. Decisions will be made according to the following rule (Fukunaga 1990):

$$\begin{array}{l}\text{Decide } \omega_1 \text{ if } P(\omega_1)P(\mathbf{X}|\omega_1) > P(\omega_2)P(\mathbf{X}|\omega_2) \\ \text{else } \omega_2\end{array}$$

Let $h(\mathbf{X}) = -\ln\frac{P(\mathbf{X}|\omega_1)}{P(\mathbf{X}|\omega_2)}$ and $t = \ln\frac{P(\omega_1)}{P(\omega_2)}$. Then the decision rule will be

$$\begin{array}{l}\text{Decide } \omega_1 \text{ if } h(\mathbf{X}) < t \\ \text{else } \omega_2\end{array}$$

$$\text{where} \quad h(\mathbf{X}) = -\ln\frac{P(\mathbf{X}|\omega_1)}{P(\mathbf{X}|\omega_2)} \tag{6.1}$$

$$t = \ln\frac{P(\omega_1)}{P(\omega_2)} \tag{6.2}$$

For the purpose of the proposed feature extraction, we start with defining "discriminantly irrelevant feature" as follows:[1]

[1] We distinguish "discriminantly irrelevant feature" from "discriminant redundant feature" (Definition 3.1) in that the discriminantly irrelevant feature does not change the value of the discriminant function while the discriminant redundant feature does not change the classification result.

Definition 6.1 We say the vector β_k is discriminantly irrelevant for any observation $\mathbf{X}$

$$h(\mathbf{X}) = h(\mathbf{X}+c\beta_k)$$

where c is a constant.

Since $h(\mathbf{X}) = h(\mathbf{X}+c\beta_k)$, the classification result for $\mathbf{X}+c\beta_k$ is the same as the classification result of $\mathbf{X}$. It can be easily seen that the discriminantly irrelevant feature does not contribute anything in discriminating between classes.

In a similar manner, we define "discriminantly relevant feature" as follows.

Definition 6.2 We say the vector β_k is discriminantly relevant if there exists at least one observation $\mathbf{X}$ such that

$$h(\mathbf{X}) \neq h(\mathbf{X}+c\beta_k)$$

where c is a constant.

From these definitions, it is clear that all discriminantly irrelevant features are features which have no impact on the value of the discriminant function and can be eliminated without increasing any classification error. Thus if it is possible to find all the discriminantly irrelevant features for a given classifier, it will be also possible to obtain the same classification accuracy as in the original space with a reduced number of features. To eliminate discriminantly irrelevant features for a given classifier, or equivalently to retain discriminantly relevant features, we define the discriminant feature matrix as follows:

Definition 6.3 The discriminant feature matrix (DFM): The discriminant feature matrix is defined as

$$\Sigma_{DFM} = \int N(\mathbf{X})N(\mathbf{X})^t p(\mathbf{X})d\mathbf{X} \tag{6.3}$$

where $N(\mathbf{X}) = \nabla h(\mathbf{X}) / |\nabla h(\mathbf{X})|$

$p(\mathbf{X})$ is a probability density function

$$\nabla h(\mathbf{X}) = \frac{\partial h}{\partial x_1}\mathbf{x}_1 + \frac{\partial h}{\partial x_2}\mathbf{x}_2 + \cdots + \frac{\partial h}{\partial x_n}\mathbf{x}_n$$

$$h(\mathbf{X}) = -\ln\frac{p(\mathbf{X}|\omega_1)}{p(\mathbf{X}|\omega_2)}$$

The property of the discriminant feature matrix is similar to that of the decision boundary feature matrix. The proof is identical to those in section 3.5.3.

Property 6.1 The discriminant feature matrix is a real symmetric matrix.

Property 6.2 The eigenvectors of the discriminant feature matrix are orthogonal.

Property 6.3 The discriminant feature matrix is positive semi-definite.

Property 6.4 The discriminant feature matrix of the whole space can be expressed as a summation of the discriminant feature matrices calculated from subspaces of the whole space if the subspaces are mutually exclusive and exhaustive.

Now we will show that all the eigenvectors of the discriminant feature matrix corresponding to zero-eigenvalues are discriminantly irrelevant and can be eliminated without increasing the classification error. In this regard, we state the following theorem.

Theorem 6.1 The eigenvectors of the discriminant feature matrix of a pattern classification problem corresponding to zero-eigenvalues are discriminantly irrelevant and can be eliminated without increasing any classification error.

Proof: We assume h(**X**) is continuous and differentiable for all **X**. Let Σ_{DFM} be the discriminant feature matrix as defined in Definition 6.3. Suppose that

$$\mathrm{rank}(\Sigma_{DFM}) = M < N.$$

Let $\{\phi_1, \phi_2, \ldots, \phi_M\}$ be the eigenvectors of Σ_{DFM} corresponding to non-zero eigenvalues. Then, for any **X**, $\nabla h(\mathbf{X})$ can be represented by a linear combination of ϕ_i, i=1,M. In other words,

$$\nabla h(\mathbf{X}) = \sum_{i=1}^{M} a_i\phi_i \text{ where } a_i \text{ is a coefficient}$$

Let φ be an eigenvector whose corresponding eigenvalue is zero. Then, φ is orthonormal to any eigenvector whose eigenvalue is not zero since eigenvectors of symmetric matrices are orthogonal to each other. It can be easily seen that the discriminant feature matrix is symmetric (Definition 6.3).

Thus, φ is orthogonal to $\nabla h(\mathbf{X})$ for any $\mathbf{X}$ since

$$\varphi\nabla h(\mathbf{X}) = \varphi\sum_{i=1}^{M} a_i\phi_i = \sum_{i=1}^{M} a_i\phi_i\varphi = 0$$

Assume that φ is not discriminantly irrelevant, i.e. discriminantly relevant. Then there exists at least one observation $\mathbf{Y}$ such that

$$h(\mathbf{Y}) \neq h(\mathbf{Y}+c\varphi) \text{ where c is a constant}$$

Let $h(\mathbf{Y})=t_0$, $h(\mathbf{Y}+c\varphi)=t_1$, and $t_0 \neq t_1$. Then there will be a point $\mathbf{Y'}$ between $\mathbf{Y}$ and $\mathbf{Y}+c\varphi$ such that

$$h(\mathbf{Y'})=\frac{t_0+t_1}{2}$$

Physically, $h(\mathbf{X})=(t_0+t_1)/2$ is a surface and $\nabla h(\mathbf{Y'})$ is a normal vector to the surface at $\mathbf{Y'}$. Then $\mathbf{Y}$ and $\mathbf{Y}+c\varphi$ must be on different sides of the surface $h(\mathbf{X})=(t_0+t_1)/2$. This means $c\varphi$ must pass through the surface $h(\mathbf{X})=(t_0+t_1)/2$ at $\mathbf{Y'}$. This contradicts the assumption that φ is orthogonal to $\nabla h(\mathbf{X})$ for any $\mathbf{X}$ including $\nabla h(\mathbf{Y'})$. Therefore if φ is an eigenvector of the discriminant feature matrix whose corresponding eigenvalue is zero, φ is discriminantly irrelevant and can be eliminated without increasing any classification error.

Q.E.D.

Figure 6.1 shows an illustration of the proof. It is impossible that $c\varphi$ passes through the surface $h(\mathbf{X})=(t_0+t_1)/2$ at $\mathbf{Y'}$ and is orthogonal to $\nabla h(\mathbf{Y'})$ at the same time.

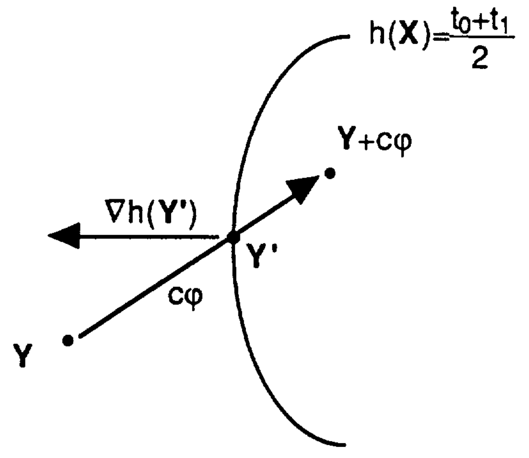

Figure 6.1 Illustration of the proof of Theorem 6.1.

From Theorem 6.1, it can be easily seen that, if the rank of the discriminant feature matrix is M, the minimum number of features needed to achieve the same classification accuracy as in the original space must be smaller than or equal to M. In particular, if the rank of the discriminant feature matrix is 1, only one feature is needed to achieve the maximum classification accuracy. This will happen when the covariance matrices of the two classes are the same assuming a Gaussian ML classifier is used. However, it is noted that all eigenvectors whose eigenvalues are not zero are not necessarily needed to achieve the same classification accuracy as could be obtained in the original space.

We will refer the feature extraction algorithm based on the Theorem 6.1 as Discriminant Feature Extraction. In practice, we will choose eigenvectors of the discriminant feature matrix according to the magnitude of the corresponding eigenvalues.

6.3 Discriminant Feature Extraction and Decision Boundary Feature Extraction

In Chapter 3, we introduced the decision boundary feature extraction algorithm. It was shown that all the needed feature vectors for classification can be extracted from the decision boundary. The decision boundary feature extraction algorithm was successfully applied to parametric classifiers in

Chapter 3, non-parametric classifiers in Chapter 4, and neural networks in Chapter 5. Now we will show that the discriminant feature extraction method is a generalized form of the decision boundary feature extraction method.

In the spectral decomposition of a matrix **A** (n by n), **A** can be represented by (Cullen 1972)

$$\mathbf{A}= \lambda_1 E_1 + \lambda_n E_n + \ldots + \lambda_n E_n \tag{6.4}$$

where the matrices E_i are called the projectors of **A** or the principal idempotents of **A**. In Chapter 3, the decision boundary feature matrix is defined as follows (Definition 3.6):

$$\Sigma_{DBFM} = \frac{1}{K} \int_S N(X)N^t(X)p(X)dX \tag{6.5}$$

There is a similarity between equations (6.4) and (6.5). In fact, the decision boundary feature matrix can be viewed as a matrix whose principal idempotents are constructed from normal vectors to the decision boundary. As a result, in the decision boundary feature extraction method, a new feature set is extracted so that the classification results are preserved.

On the other hand, in the discriminant feature extraction method, the discriminant feature matrix is defined as follows (Definition 6.3):

$$\Sigma_{DFM} = \int N(X)N(X)^t p(X)dX \tag{6.3}$$

The discriminant feature matrix can be viewed as a matrix whose principal idempotents are constructed from vectors which give changes to the value of the discriminant function. As a result, in the discriminant feature extraction method, a new feature set is extracted such that the value of the discriminant function for a given classifier is preserved. Consider the example in Figure 6.1. In the decision boundary feature extraction, the value of h(**X**) in (6.1) can be changed as long as the classification of **X** remains the same. In Discriminant Feature Extraction, the value of h(**X**) is preserved. As a result, the decision boundary feature extraction method will be more efficient if the decision boundary can be exactly located, which is a general case of parametric classifiers. However, when finding the decision boundary is difficult or time consuming as in some cases of non-parametric classifiers, the discriminant feature extraction method could be an alternative. Furthermore, since the discriminant feature extraction method finds all the vectors which give changes in the value of the discriminant function, such a generalization can be used for feature extraction of other applications such as density estimation, non-parametric regression, and etc.

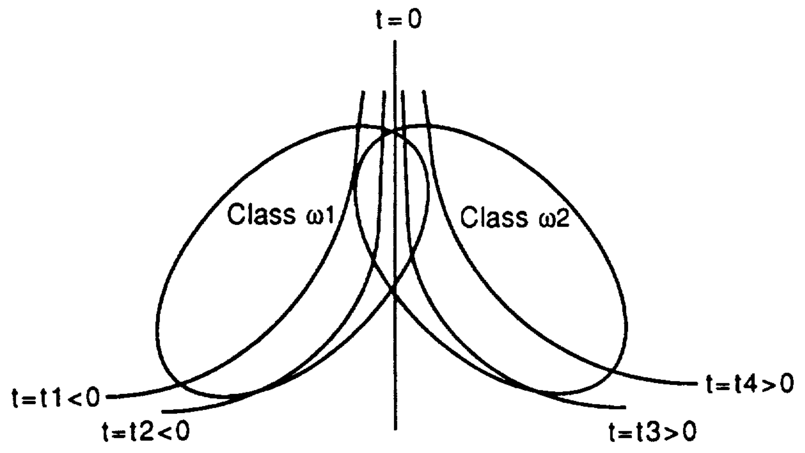

Figure 6.2 Decision Boundary Feature Extraction and Discriminant Feature Extraction.

6.4 Discriminant Feature Extraction

6.4.1 Discriminant Feature Extraction for Two Pattern Classes

Now we propose a procedure to calculate the discriminant feature matrix for parametric and non-parametric classifications.

Procedure for Discriminant Feature Extraction for
Parametric/Non-Parametric Classifications
(2 pattern class case)

1. Classify the training data using full dimensionality.

2. From correctly classified samples, estimate the discriminant feature matrix as follows:

$$\Sigma_{DFM} = \frac{1}{L}\sum_i N_i N_i^t$$

where L : the number of samples correctly classified

$N_i = \nabla h(\mathbf{X}) / |\nabla h(\mathbf{X})|$ and

$$\nabla h(\mathbf{X}) = \frac{\partial h}{\partial x_1} x_1 + \frac{\partial h}{\partial x_2} x_2 + \cdots + \frac{\partial h}{\partial x_n} x_n$$

For non-parametric classifications, estimate $\nabla h(\mathbf{X})$ as follows:

$$\nabla h(\mathbf{X}) \approx \frac{\Delta h}{\Delta x_1} x_1 + \frac{\Delta h}{\Delta x_2} x_2 + \cdots + \frac{\Delta h}{\Delta x_n} x_n$$

3. Select the eigenvectors of the decision boundary feature matrix as new feature vectors according to the magnitude of corresponding eigenvalues.

6.4.2 Discriminant Feature Extraction for Multiclass Case

If there are more than 2 classes, the procedure can be repeated for each pair of classes. The total discriminant feature matrix can be calculated by averaging the discriminant feature matrix of each pair of classes. If prior probabilities are available, the summation can be weighted. In other words, if there are M classes, the total discriminant feature matrix can be calculated as

$$\Sigma_{DFM} = \sum_{i}^{M} \sum_{j, j\neq i}^{M} P(\omega_i)P(\omega_j)\Sigma_{DFM}^{ij} \tag{6.6}$$

where Σ_{DFM}^{ij} is a discriminant feature matrix between class ω_i and class ω_j and $P(\omega_i)$ is the prior probability of class ω_i if available. Otherwise let $P(\omega_i)=1/M$.

6.4.3 Eliminating Redundancy in Multiclass Problems

The total discriminant feature matrix defined in equation (6.6), can be made more efficient. Consider the following example situation. Suppose Table

6.1 shows eigenvalues for a 2 pattern class problem. Table 6.1 also shows proportions of the eigenvalues, classification accuracies, and normalized classification accuracies obtained by dividing the classification accuracies by the classification accuracy obtained using all features. With just one feature, the classification accuracy is 91.6% which is 97.3% of the classification accuracy obtained using all features. Thus, in this 2 class problem, if this level of accuracy is deemed adequate, just one feature is necessary to be included in calculating the total discriminant feature matrix. The other 19 features contributes little in improving classification accuracy and can be eliminated in calculating the total discriminant feature matrix. In addition, feature vectors from other pairs of classes will improve the classification accuracy.

Table 6.1 Eigenvalues of the discriminant feature matrix.

Eigenvalue	Proportion of Eigenvalue (%)	Accumulation of Eigenvalue (%)	Classification Accuracy (%)	Normalized Classification Accuracy (%)
0.323	6.7	6.7	91.6	97.3
0.211	4.4	11.1	93.3	99.1
0.149	3.1	14.2	92.7	98.5
0.093	1.9	16.1	92.9	98.7
0.069	1.4	17.6	91.8	97.6
0.048	1.0	18.5	93.9	99.8
0.039	0.8	19.4	94.1	100.0
0.026	0.5	19.9	94.5	100.4
0.018	0.4	20.3	94.5	100.4
0.012	0.3	20.5	94.7	100.6
0.006	0.1	20.7	94.3	100.2
0.002	0.0	20.7	94.7	100.6
0.002	0.0	20.7	94.5	100.4
0.001	0.0	20.8	94.7	100.6
0.001	0.0	20.8	94.5	100.4
0.001	0.0	20.8	94.1	100.0
0.000	0.0	20.8	94.1	100.0
0.000	0.0	20.8	94.1	100.0
0.000	0.0	20.8	94.1	100.0
0.000	0.0	20.8	94.1	100.0

To eliminate such redundancy in multiclass problems, we define the discriminant feature matrix of P_t ($\Sigma_{DFM(Pt)}$) as follows:

Definition 6.4 Let L_t be the number of eigenvectors corresponding to largest eigenvalues needed to obtain P_t of the classification accuracy obtained with all features. Then the discriminant feature matrix of P_t ($\Sigma_{DBFM(Pt)}$) as

$$\Sigma_{DFM(Pt)} = \sum_{i=1}^{L_t} \lambda_i \varphi_i \varphi_i^t$$

where λ_i and φ_i are eigenvalues and eigenvectors of the discriminant feature matrix.

And the total discriminant feature matrix of P_t of multiclass problem can be defined as

$$\Sigma_{DFM(Pt)} = \sum_{i=1}^{M} \sum_{j=1\ j\neq i}^{M} P(\omega_i)P(\omega_j)\Sigma^{ij}_{DFM(Pt)}$$

where $\Sigma^{ij}_{DFM(Pt)}$ is the discriminant feature matrix of P_t between class ω_i and class ω_j and $P(\omega_i)$ is the prior probability of class ω_i if available. Otherwise let $P(\omega_i)=1/M$.

In the experiments to follow, P_t is set to between 0.95 and 0.97 (see section 3.5.6).

From Definition 6.4, we can calculate the discriminant feature matrix of 0.95 of Table 6.1 as follows:

The classification accuracy using full dimensionality (20) is 94.1%. The number of features needed to achieve classification accuracy of 89.4%(=94.1*0.95) is 1. Therefore, the discriminant feature matrix of 0.95 of Table 6.1 is given by

$$\Sigma_{DFM(0.95)} = \sum_{i=1}^{1} \lambda_i \varphi_i \varphi_i^t = \lambda_1 \varphi_1 \varphi_1^t$$

where λ_i's are eigenvalues of Σ_{DFM} sorted in descending order and φ_i's are the corresponding eigenvectors.

6.5 Experiments

6.5.1 Parametric Classification

6.5.1.1 Experiments with Generated Data

To evaluate closely how the proposed algorithm performs under various circumstances, tests are conducted on data generated with given statistics assuming Gaussian distributions. In all parametric examples, a Gaussian ML classifier is used.

Example 6.1 In this example, data are generated for the following statistics.

$$M_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} M_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \Sigma_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$P(\omega_1) = P(\omega_2) = 0.5$$

200 samples are generated for each class. Since the covariance matrices are the same, it can be easily seen that the decision boundary will be a straight line and just one feature is needed to achieve the same classification accuracy as in the original space. The eigenvalues λ_i and the eigenvectors ϕ_i of Σ_{DFM} are calculated as follows:

$$\lambda_1 = 0.99971 \; \lambda_2 = 0.00029 \; \phi_1 = \begin{bmatrix} 0.67 \\ -0.74 \end{bmatrix} \phi_2 = \begin{bmatrix} 0.74 \\ 0.67 \end{bmatrix}$$

Since one eigenvalue is significantly larger than the other, it can be said that the rank of Σ_{DFM} is 1. That means only one feature is needed to achieve the same classification accuracy as in the original space. Table 6.2 shows the classification accuracies. The proposed algorithm finds the right feature achieving the same classification accuracy as in the original space with one feature.

Table 6.2 Classification accuracies of Example 6.1.

No. Features	Classification Accuracy
1	95.8 (%)
2	95.8 (%)

Example 6.2 In this example, data are generated with the following statistics.

$$M_1 = \begin{bmatrix} 0.01 \\ 0 \end{bmatrix} \Sigma_1 = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} M_2 = \begin{bmatrix} -0.01 \\ 0 \end{bmatrix} \Sigma_2 = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

$$P(\omega_1) = P(\omega_2) = 0.5$$

200 samples are generated for each class. In this case, there is almost no difference in the mean vectors. The variance of feature 1 of class ω_1 is equal to that of class ω_2 while the variance of feature 2 of class ω_1 is larger than that of class ω_2. Thus the decision boundary will consist of two hyperbolas. However, the effective decision boundary could be approximated by a straight line. As a result, only one feature may be needed to achieve almost the same classification accuracy as in the original space. The eigenvalues λ_i and the eigenvectors ϕ_i of Σ_{DFM} are calculated as follows:

$$\lambda_1 = 0.99330 \; \lambda_2 = 0.00670 \; \phi_1 = \begin{bmatrix} 0.09 \\ 1.00 \end{bmatrix} \phi_2 = \begin{bmatrix} -1.00 \\ 0.09 \end{bmatrix}$$

Since one eigenvalue is significantly larger than the other, it can be said that the rank of Σ_{DFM} is 1. That means only one feature is needed to achieve the same classification accuracy as in the original space. Considering the statistics of the two classes, the rank of Σ_{DFM} gives the correct number of features to achieve the same classification accuracy as in the original space. Table 6.3 shows the classification accuracies. The proposed algorithm find the right feature achieving the same classification accuracy as in the original space with one feature.

Table 6.3 Classification accuracies of Example 6.2.

No. Features	Classification Accuracy
1	61.0 (%)
2	61.0 (%)

From the examples, it can be seen that the proposed discriminant feature extraction algorithm finds a good feature set even though there is no class mean difference (Example 6.2) or no class covariance difference (Example 6.1).

6.5.1.2 Experiments with Real Data

Along with the proposed Discriminant Feature Extraction, five other feature selection/extraction algorithms, Uniform Feature Design, Principal Component Analysis (the Karhunen-Loeve transformation) (Richards 1986), Canonical Analysis (Richards 1986), the Foley & Sammon method (Foley and Sammon 1975), and Decision Boundary Feature Extraction are tested to evaluate and compare the performance of Discriminant Feature Extraction. The Foley & Sammon method is based on the generalized Fisher criterion (Foley and Sammon 1975). For a two class problem, the Foley & Sammon method is used for comparison. If there are more than 2 classes, Canonical Analysis is used for comparison.

In the following test, two classes are chosen from the FSS data. Table 6.4 provides information on the classes. Figure 3.17 in Chapter 3 shows the mean graph of the two classes.

Table 6.4 Class description of data collected at Finney Co. KS.

SPECIES	No. of Sample	No. of Training Sample
WINTER WHEAT	691	400
UNKNOWN CROPS	619	400

Figure 6.3 show a performance comparison. First the original data set is reduced to a 17-dimensional data set using Uniform Feature Design. With 17 features, the classification accuracy is 95.5%. Discriminant Feature Extraction achieves 91.2% and 93.7% with one and two features, respectively. Though Discriminant Feature Extraction showed a better performance than Principal Component Analysis and Uniform Feature Design, Decision Boundary Feature Extraction and the Foley & Sammon method show the best performance, achieving about the maximum possible classification accuracy with one feature.

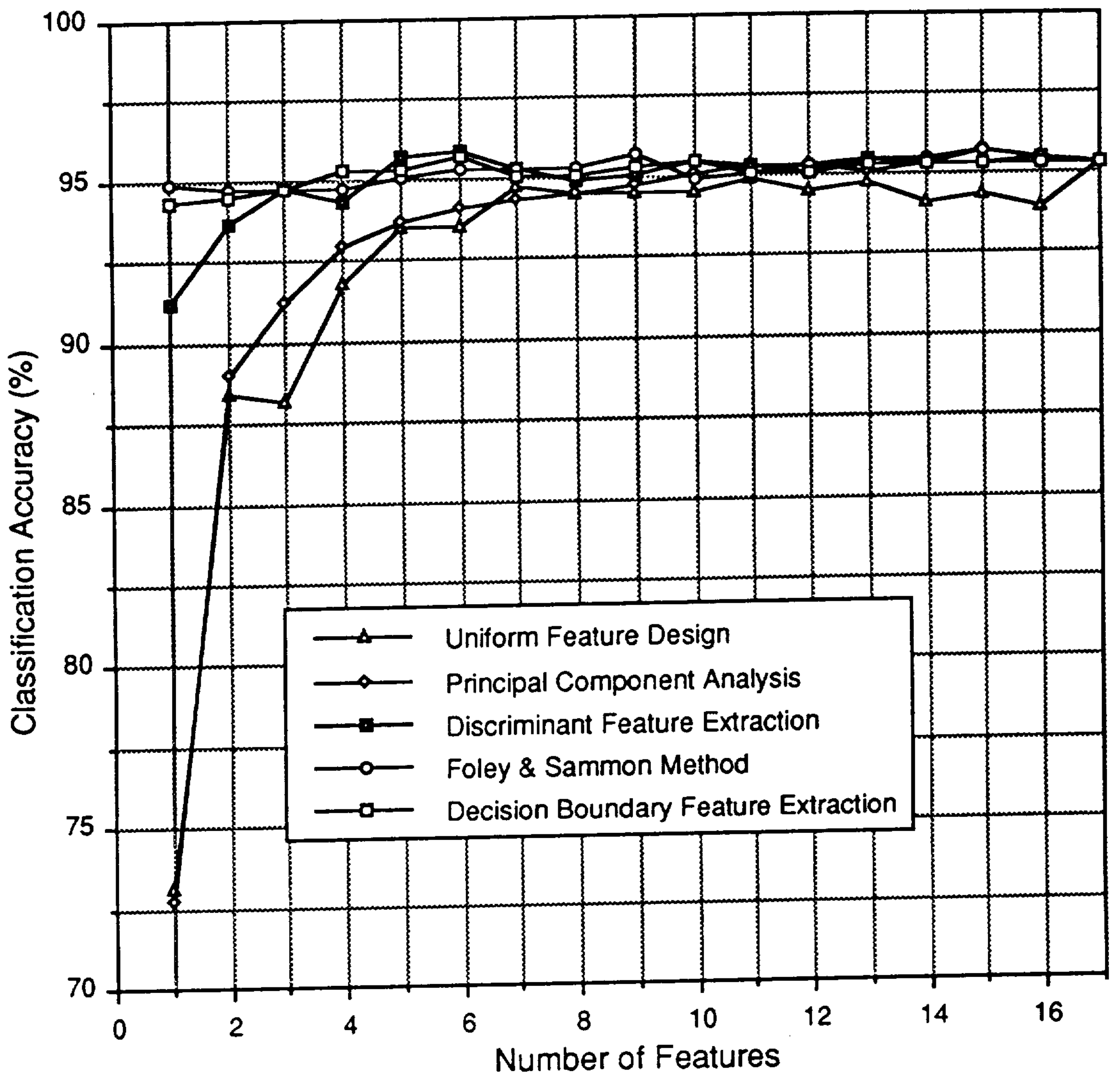

Figure 6.3 Performance comparison.

In the following test, two classes are chosen from the FSS data. Table 6.5 provides information on the classes. Figure 3.19 in Chapter 3 shows the mean graph of the two classes. There is relatively little difference in the mean vectors.

Table 6.5 Class description of data collected at Finney Co. KS.

SPECIES	No. of Sample	No. of Training Sample
WINTER WHEAT	223	223
SPRING WHEAT	474	474

Figure 6.4 show a performance comparison. With 25 features, the classification accuracy is 92.4%. Decision Boundary Feature Extraction and Discriminant Analysis show similar performance, outperforming other methods.

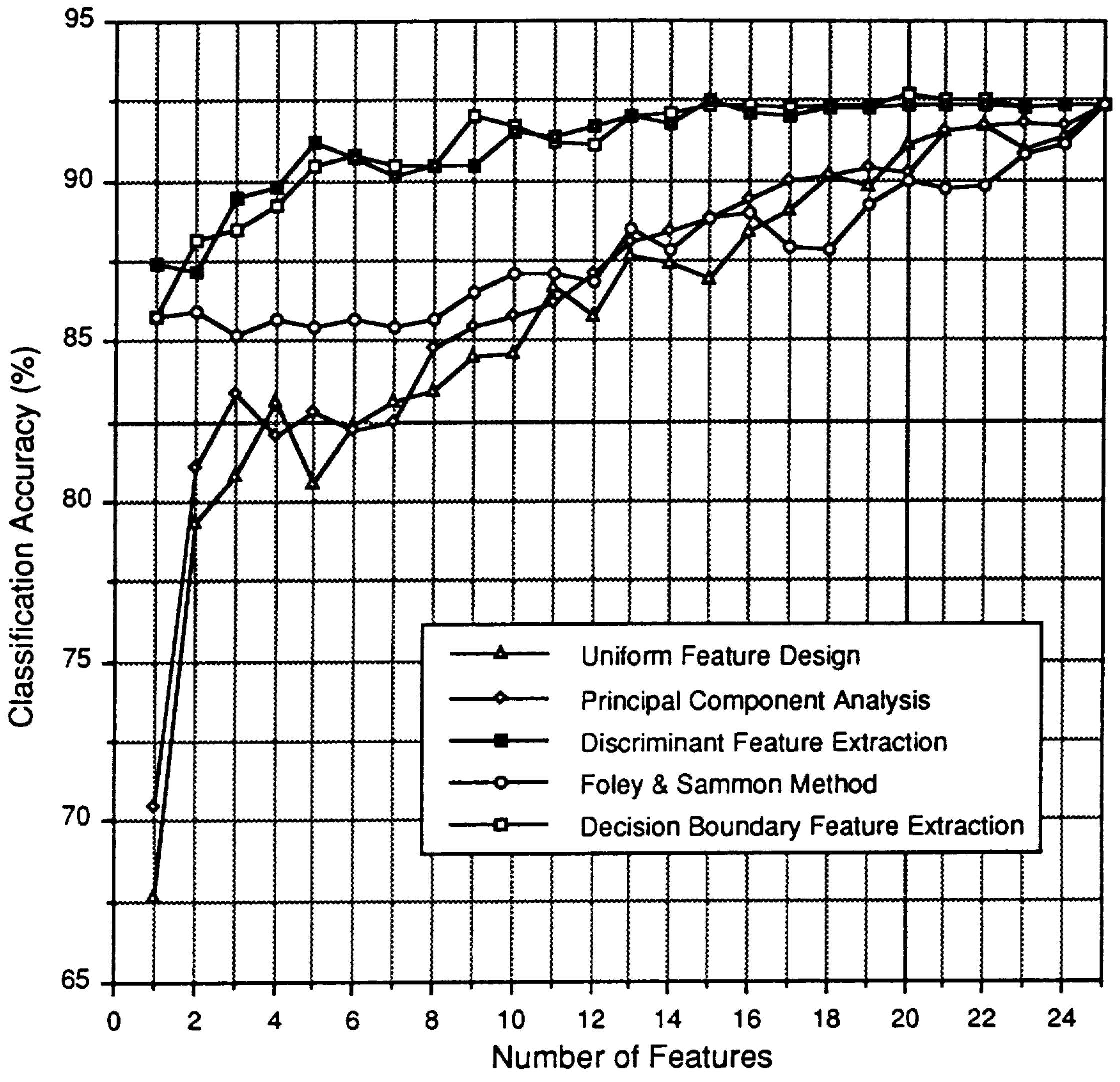

Figure 6.4 Performance comparison.

In the following test, 4 classes are chosen from the data collected at Hand Co. SD. on May 15, 1978. Table 6.6 provides class information. Figure 3.23 in Chapter 3 shows the mean graph of the 4 classes.

Table 6.6 Class description.

Species	Date	No. of Samples	No. of Training Samples
Winter Wheat	May 15, 1978	223	223
Native Grass Pas	May 15, 1978	196	196
Oats	May 15, 1978	163	163
Unknown Crops	May 15, 1978	253	253

Figure 6.5 shows a performance comparison. In this experiment, Decision Boundary Feature Extraction and Discriminant Feature Extraction outperform other methods. Though Decision Boundary Feature Extraction and Discriminant Feature Extraction show similar performance, Decision Boundary Feature Extraction shows better performance when 7-10 features are used.

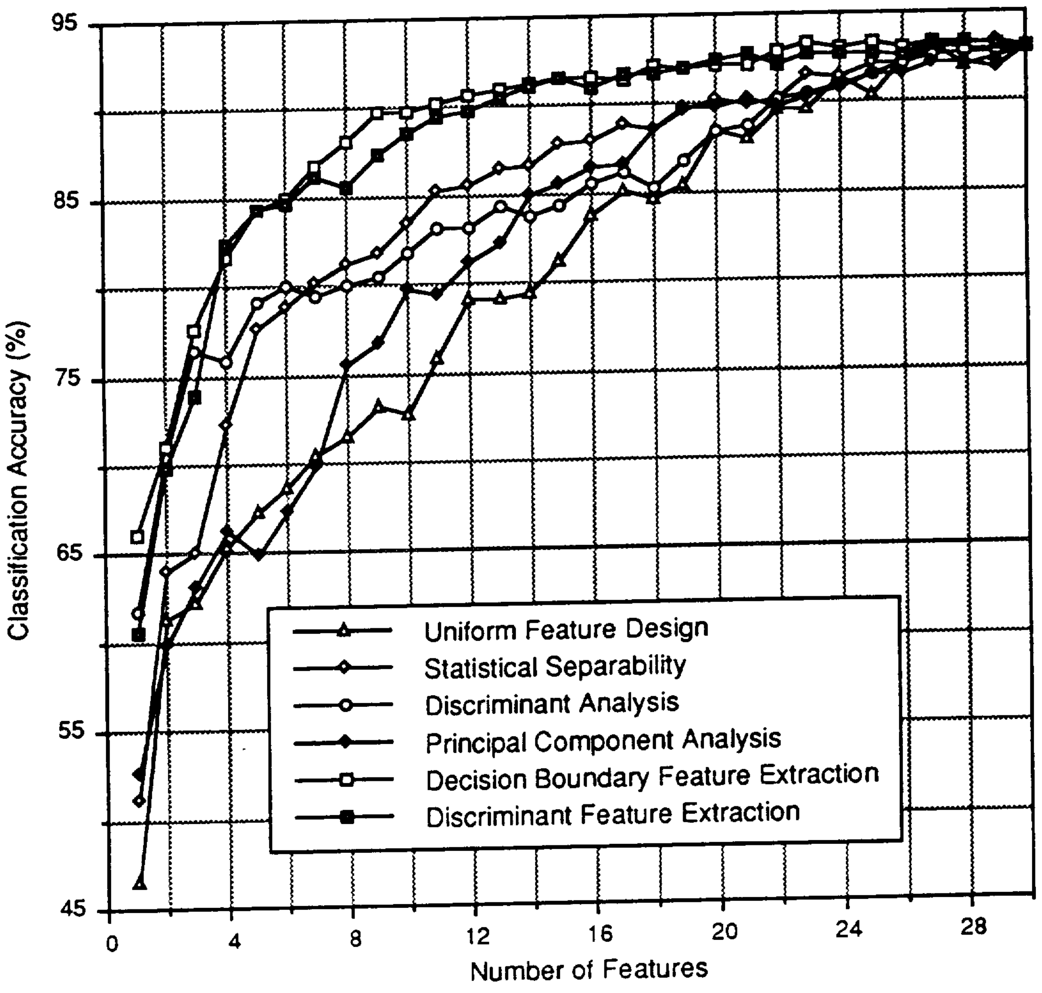

Figure 6.5 Performance comparison.

In the next test, 6 classes chosen from the FSS data. Table 6.7 provides description of the 6 classes. In this test, 300 samples are used for training and the rest are used for test.

Table 6.7 Class description of the multi-temporal 6 classes.

Date	Location	Species	No. Sample
770308	Finney CO. KS.	Winter Wheat	691
770626	Finney CO. KS.	Winter Wheat	677
771018	Hand CO. SD.	Winter Wheat	662
770503	Finney CO. KS.	Winter Wheat	658
770626	Finney CO. KS.	Summer Fallow	643
780726	Hand CO. SD.	Spring Wheat	518

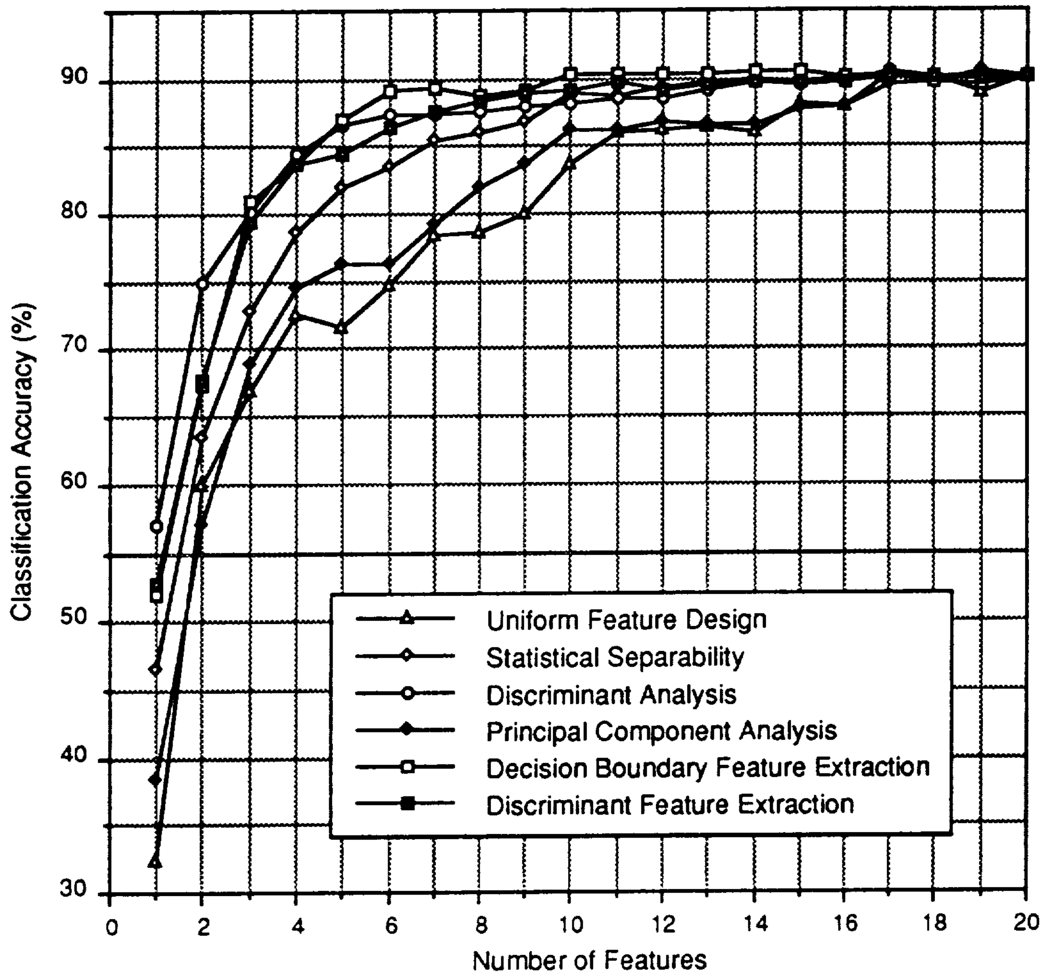

Figure 6.6 Performance comparison.

Figure 6.6 shows a performance comparison. In this example, Discriminant Analysis shows the best performance until 3 features are used.

When more than 2 features are used, Decision Boundary Feature Extraction shows the best performance.

6.5.2 Non-Parametric Classifications

6.5.2.1 Experiments with Generated Data

The non-parametric classifier was implemented by Parzen density estimation using a Gaussian kernel function.

Example 6.3 In this example, class ω_1 is normal with the following statistics:

$$\mathbf{M}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma_1 = \begin{bmatrix} 3 & 0.5 \\ 0.5 & 3 \end{bmatrix}$$

And class ω_2 is equally divided between two normal distributions with the following statistics:

$$\mathbf{M}_2^1 = \begin{bmatrix} -3 \\ 3 \end{bmatrix} \Sigma_2^1 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix} \text{ and } \mathbf{M}_2^2 = \begin{bmatrix} 3 \\ -3 \end{bmatrix} \Sigma_2^2 = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix}$$

200 samples are generated for each class. Eigenvalues λ_i and eigenvectors ϕ_i of Σ_{DFM} are calculated as follows:

$$\lambda_1 = 0.74820,\ \lambda_2 = 0.25180 \text{ and } \phi_1 = \begin{bmatrix} 0.68 \\ -0.74 \end{bmatrix},\ \phi_2 = \begin{bmatrix} 0.74 \\ 0.68 \end{bmatrix}$$

Figure 6.7 shows the distribution of the data and the eigenvectors of Σ_{DFM}. Table 6.8 shows the classification accuracies. The proposed algorithm find the right feature achieving the same classification accuracy as in the original space with one feature.

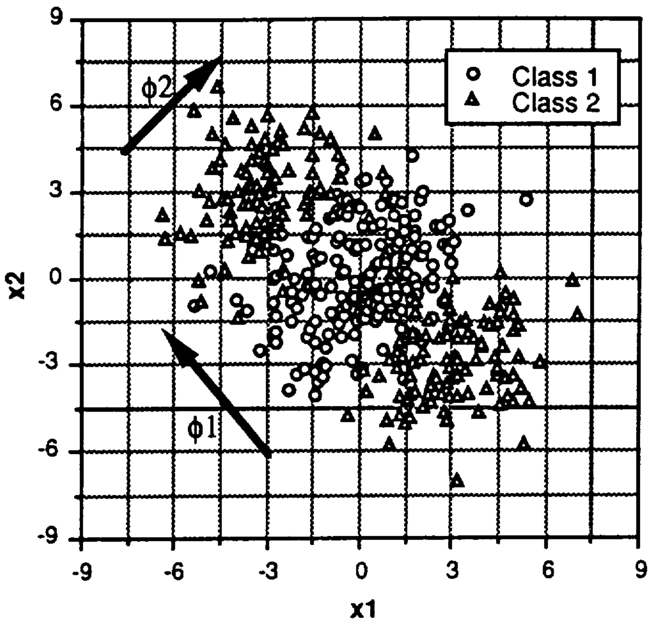

Figure 6.7 Data distribution of Example 6.3 and the feature vectors found by the discriminant feature extraction method.

Table 6.8 Classification accuracies of Example 6.3.

No. Features	Classification Accuracy
1	90.3 (%)
2	90.3 (%)

6.5.2.2 Experiments with Real Data

In the next test, there are 3 classes and each class has 2 subclasses. In other words, 2 subclasses were combined to form a new class. By purposely combining data from different classes, the data are made to be multi-modal. Table 6.9 provides information on the classes. Figure 4.14 in Chapter 4 shows a mean value graph of the 6 subclasses and Figure 4.15 in Chapter 4 shows a mean value graph of the 3 classes each of which has 2 subclasses. 500 randomly selected samples from each classes are used as training data and the rest are used for test.

Table 6.9 Class description.

Class	Subclass	No. of Samples	Total No. of Sample
Class ω_1	Winter Wheat March 8, 1977	691	1209
	Spring Wheat July 26, 1978	518	
Class ω_2	Winter Wheat June 26, 1977	677	1146
	Spring Wheat Sep. 21, 1978	469	
Class ω_3	Winter Wheat Oct. 18, 1977	662	1103
	Spring Wheat Oct. 26, 1978	441	

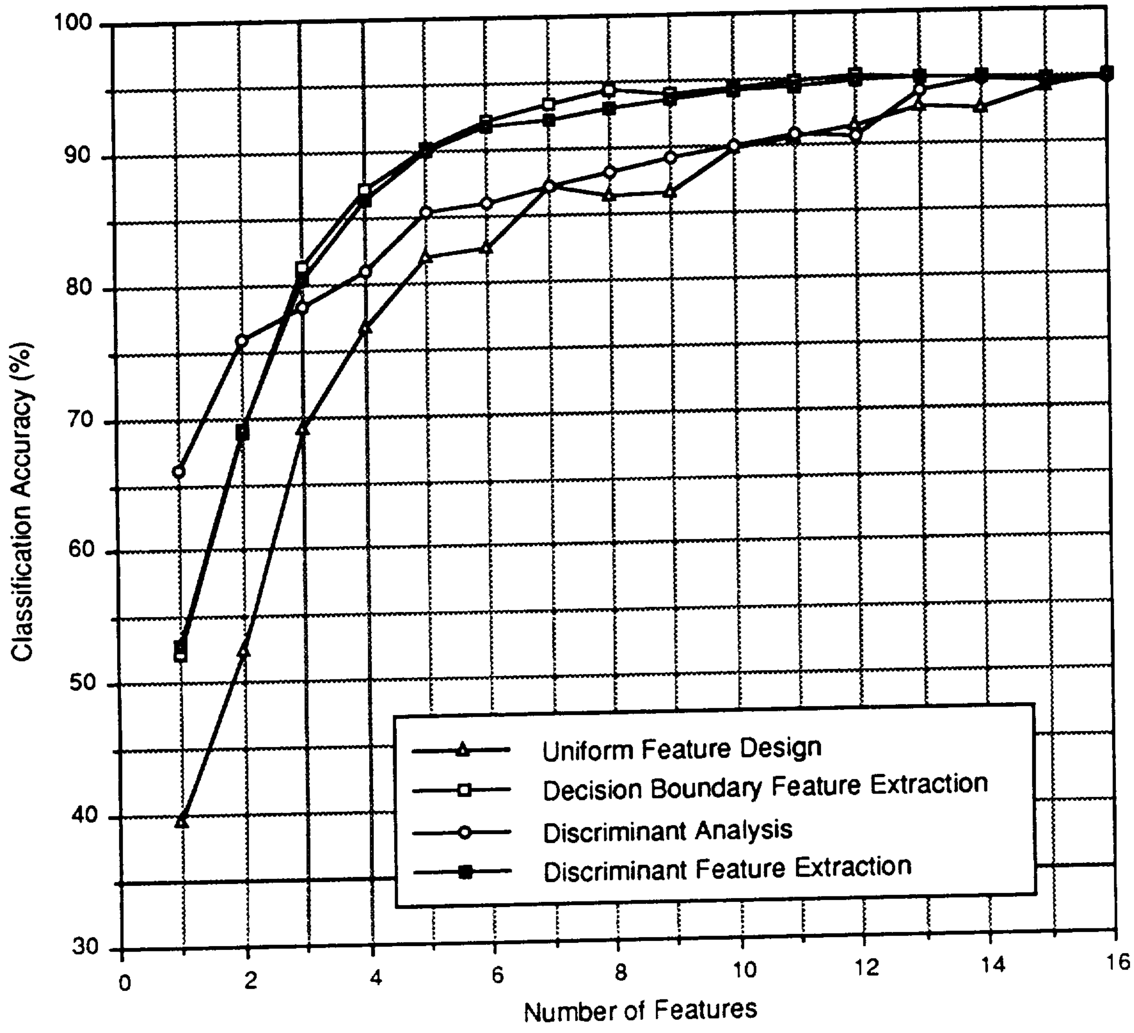

Figure 6.8 Performance comparison.

Figure 6.8 shows a performance comparison. Discriminant Feature Extraction and Decision boundary Feature Extraction show similar performance, outperforming other methods. It is noted that, in Discriminant Feature Extraction, the decision boundary need not to be found, reducing computing time.

In the next test, there are 3 classes and each class has 2 subclasses. In other words, 2 subclasses were combined to form a new class. Table 6.10 provides information on the classes. 500 randomly selected samples from each classes are used as training data and the rest are used for test.

Table 6.10 Class description.

Class	Subclass	No. of Samples	Total No. of Sample
Class ω_1	Winter Wheat March 8, 1977	691	1145
	Spring Wheat July 9, 1978	454	
Class ω_2	Winter Wheat June 26, 1977	677	1195
	Spring Wheat July 26, 1978	518	
Class ω_3	Winter Wheat Oct. 18, 1977	662	1103
	Spring Wheat Oct. 26, 1978	441	

Figure 6.9 shows a performance comparison. The classification accuracy with 17 features is 96.0%. Discriminant Analysis shows the best performance until 3 features are used. However, when more than 2 features are used, Decision Boundary Feature Extraction and Discriminant Feature Extraction outperform all other methods. Overall, Discriminant Feature Extraction and Decision boundary Feature Extraction show similar performances.

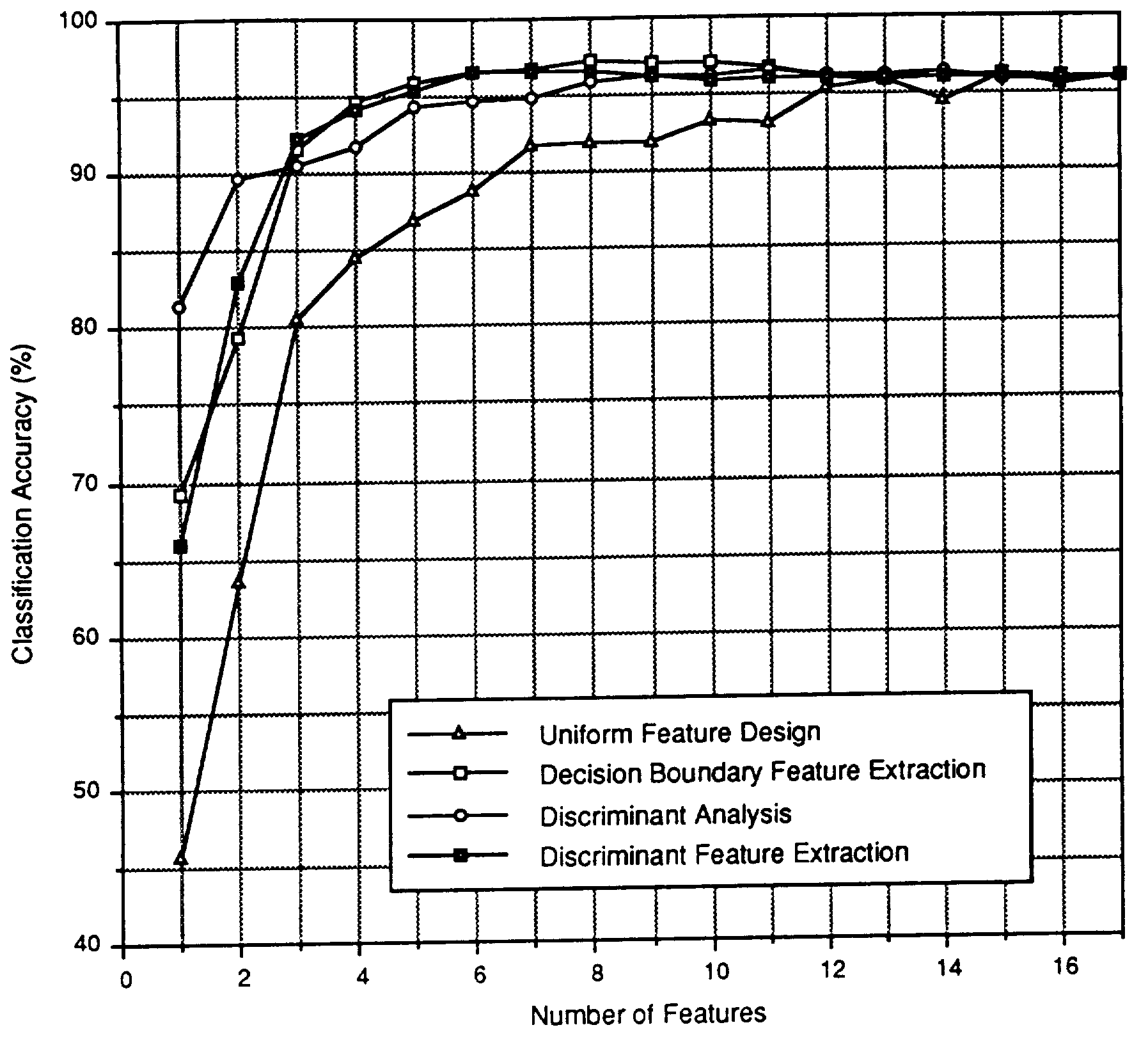

Figure 6.9 Performance comparison.

6.6 Conclusion

In this chapter, we considered feature extraction as eliminating features which have no impact on the value of the discriminant function. In order to find the discriminantly irrelevant features which have no impact on the value of the discriminant function, we defined the discriminant feature matrix and showed that eigenvectors of the discriminant feature matrix corresponding to zero eigenvalues are discriminantly irrelevant features and can be eliminated without increasing classification error. Then we proposed a procedure for the discriminant feature extraction algorithm.

We compared the discriminant feature extraction method with the decision boundary feature extraction method in the previous chapters and showed that the discriminant feature extraction method is a generalized form of the decision boundary feature extraction method. When the decision boundary is well defined and can be easily found as in parametric classifiers, the decision boundary feature extraction feature extraction method gives better performances. However, if the decision boundary is not well defined or difficult to find as in some non-parametric classifiers, the discriminant feature extraction method gives comparable performance without the need to find the decision boundary which is very time-consuming in some non-parametric classifiers. Furthermore, by generalizing the concept, the technique can be used for constructing a matrix from vectors which are useful for a given problem, such as non-parametric regression, density estimation, and feature extraction for other applications.

Experiments show that the discriminant feature extraction method can be used for parametric and non-parametric classifiers, and does not deteriorate even if there is no difference in mean vectors or in covariance matrices. Although the discriminant feature extraction method was developed for the discriminant function which uses a posteriori probabilities, it can be used for any discriminant function.

CHAPTER 7 ANALYZING HIGH DIMENSIONAL DATA

7.1 Introduction

Developments with regard to sensors for Earth observation are moving in the direction of providing much higher dimensional multispectral imagery than is now possible. The HIRIS instrument now under development for the Earth Observing System (EOS), for example, will generate image data in 192 spectral bands simultaneously (Goetz 1989). MODIS (Ardanuy et al. 1991), AVIRIS (Porter et al. 1990) and the proposed HYDICE are additional examples. Although conventional analysis techniques primarily developed for relatively low dimensional data can be used to analyze high dimensional data, there are some problems in analyzing high dimensional data which have not been encountered in low dimensional data. In this chapter, we address some of these problems. In particular, we investigate (1) the relative potential of first and second order statistics in discriminating between classes in high dimensional data, (2) the effects of inaccurate estimation of first and second order statistics on discriminating between classes, and (3) a visualization method for second order statistics of high dimensional data.

7.2 First and Second Order Statistics in High Dimensional Data

The importance of the second order statistics in discriminating between classes in multispectral data was recognized by Landgrebe (1971). In that study, it was found that small uncorrelated noise added to each band caused a greater decrease in classification accuracy than larger correlated noise. We begin with a test to investigate the role of first and the second order statistics in high dimensional data. The test was done using FSS (Field Spectrometer

System) data obtained from a helicopter platform (Biehl et al. 1982). Table 7.1 shows major parameters of FSS.

Table 7.1 Parameters of Field Spectrometer System (FSS).

Number of Bands	60
Spectral Coverage	0.4 - 2.4 μm
Altitude	60 m
IFOV(ground)	25 m

In order to evaluate the roles of first and second order statistics in high dimensional data, three classifiers were tested. The first classifier is the Gaussian Maximum Likelihood (ML) classifier which utilizes both class mean and class covariance information. For the second case, the mean vectors of all classes were made zero. Thus, the second classifier, which is a Gaussian ML classifier, is constrained to use only covariance differences among classes. The third classifier is a conventional minimum distance classifier (Richards 1986) which utilizes only first order statistics (Euclidean distance). Note that the first and third classifiers were applied to the original data set; the second classifier was applied to the modified data set where the mean vectors of all classes were made to zero so that there were no mean differences among classes.

To provide data with different numbers of spectral features, a simple band combination procedure, referred to as Uniform Feature Design, was used. In this procedure, adjacent bands were combined to form the desired number of features. For example, if the number of features is to be reduced from 60 to 30, each two consecutive bands are combined to form a new feature. Where the number of features desired is not evenly divisible into 60, the nearest integer number of bands is used. For example, for 9 features, the first 6 original bands were combined to create the first feature, then the next 7 bands were combined to create the next feature, and so on.

In the following test, 12 classes were selected from FSS data. The selected data were multi-temporal. Table 7.2 provides information on the 12 classes. 100 randomly selected samples were used as training data and the rest were used as test data. Figure 7.1 shows the graph of the class mean values of the 12 classes.

Table 7.2 Description of the multi-temporal 12 classes.

Date	Location	Species	No. of Samples
770308	Finney CO. KS.	Winter Wheat	691
770626	Finney CO. KS.	Winter Wheat	677
771018	Hand CO. SD.	Winter Wheat	662
770503	Finney CO. KS.	Winter Wheat	658
770626	Finney CO. KS.	Summer Fallow	643
780726	Hand CO. SD.	Spring Wheat	518
780602	Hand CO. SD.	Spring Wheat	517
780515	Hand CO. SD.	Spring Wheat	474
780921	Hand CO. SD.	Spring Wheat	469
780816	Hand CO. SD.	Spring Wheat	464
780709	Hand CO. SD.	Spring Wheat	454
781026	Hand CO. SD.	Spring Wheat	441

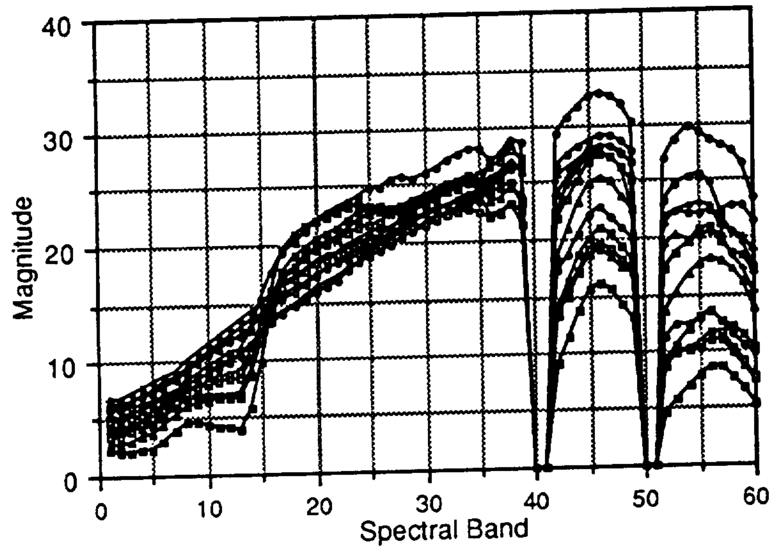

Figure 7.1 Class means of the 12 multi-temporal classes.

The original 60 band data were reduced using Uniform Feature Design to 1 through 20 feature data and the three classifiers were tested on the reduced feature sets (1 through 20). Figure 7.2 shows a performance comparison of the three classifiers. As expected, the Gaussian ML classifier performs better that the other two classifiers, achieving 94.8% with 20 features. On the other hand, the minimum distance classifier achieved about 40 % classification accuracy with 20 features. Actually the performance of the minimum distance classifier was saturated after four features. Meanwhile, the classification accuracies of the Gaussian ML classifier with zero mean data continuously increased as more features were used achieving 73.2% with 20 features. In low dimensionality (no. of features < 4), the minimum distance classifier shows better performance than the Gaussian ML classifier with zero mean data. When more than 3 features are used, the Gaussian ML classifier with zero mean data shows better performance than the minimum distance classifier.

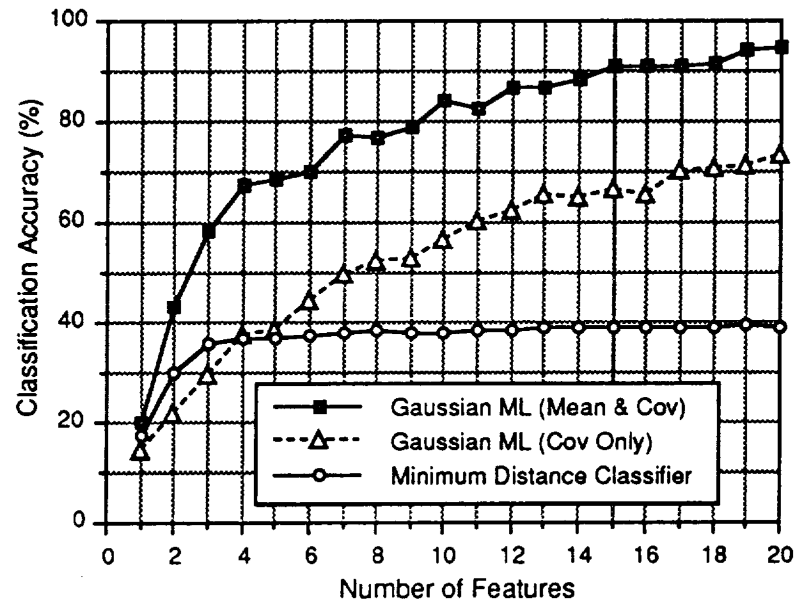

Figure 7.2 Performance comparison of the Gaussian ML classifier, the Gaussian ML classifier with zero mean data, and the minimum distance classifier.

Figures 7.3-4 show the performances of the minimum distance classifier and the Gaussian ML classifier with zero mean data for various number of classes. It is interesting that the performances of the minimum distance classifier reached saturation with 4-5 features and after that adding more features did not make any significant change in classification accuracy. On the other hand, the performances of the Gaussian ML classifier with zero mean data shows improvements as more features are used as can be seen in Figure 7.4.

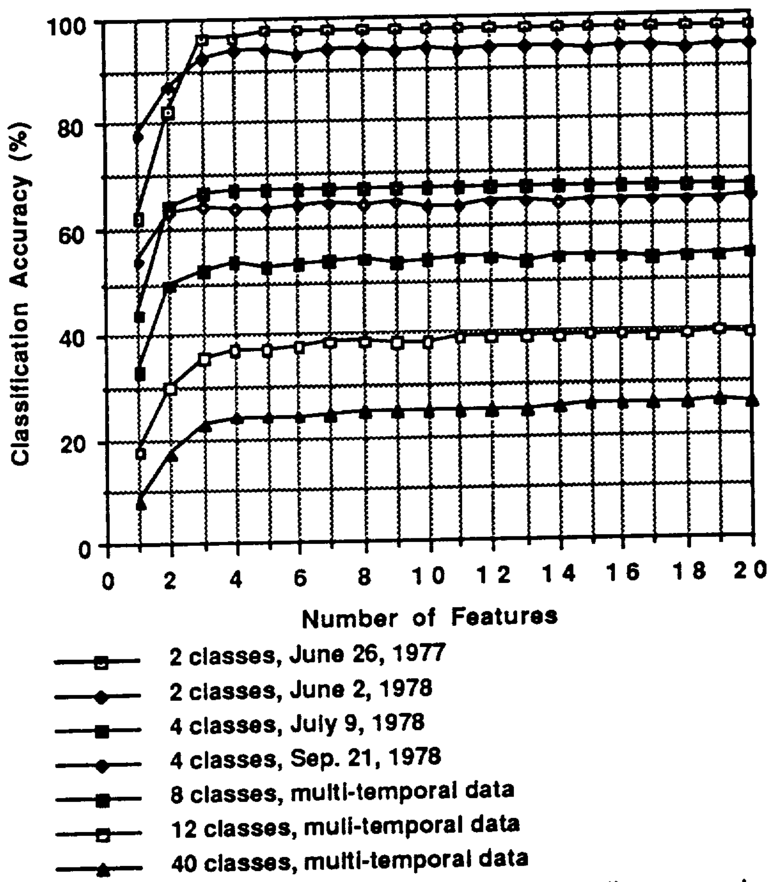

Figure 7.3 Performance of the minimum distance classifier.

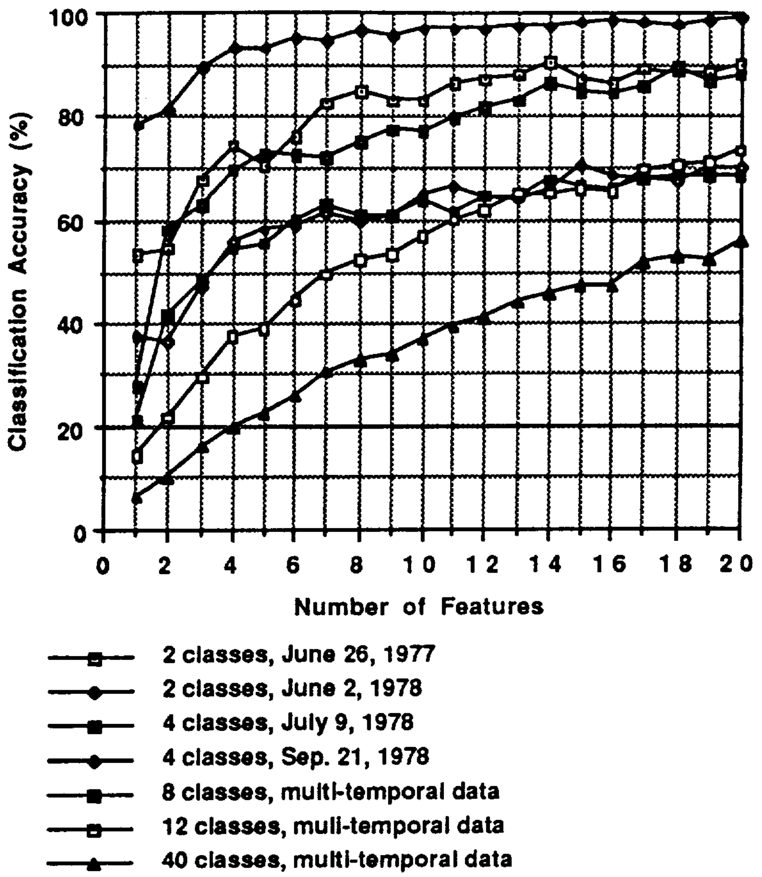

Figure 7.4 Performance of the Gaussian ML classifier with zero mean data.

It can be observed from the experiments that the second order statistics play an important role in high dimensional data. The ineffectiveness of the minimum distance classifier, which does not use second order statistics, is particularly noteworthy. Though the Euclidean distance is not as effective a measure as other distance measures which utilize the second order statistics, the minimum distance classifier is still widely used in relative low dimensional data due to computation cost. In particular, in computationally intensive tasks such as clustering, the Euclidean distance is widely used.

It is noteworthy that, in the low dimension case, class mean differences play a more important role in discriminating between classes than the class covariance differences. However, as the dimensionality increases, the class covariance differences become more important, especially when adjacent bands are highly correlated and there are sizable variations in each band of each class. This suggests that care must be exercised in applying classification algorithms such as the minimum distance classifier to high dimensional data.

7.3 Minimum Distance Classifier in High Dimensional Space

7.3.1 Average Class Mean Difference and Average Distance From Mean

We will next further investigate the performance of the minimum distance classifier in high dimensional remotely sensed data. In order to analyze qualitatively the performance of the minimum distance classifier, the Average Class Mean Difference (ACMD) is defined as follows:

$$\text{Average Class Mean Difference (ACMD)} = \frac{2}{L(L-1)}\sum_{i=2}^{L}\sum_{j=1}^{i-1} |\mathbf{M}_i - \mathbf{M}_j|$$

where L is the number of classes and $\mathbf{M}_i$ is the mean of class ω_i.

Generally, increasing the ACMD should improve the performance of the minimum distance classifier. Similarly, the Average Distance From Mean (ADFM) is defined as follows:

$$\text{Average Distance From Mean (ADFM)} = \frac{1}{N}\sum_{i=1}^{L}\sum_{j=1}^{N_i} |\mathbf{X}_j^i - \mathbf{M}_i|$$

where N is the total number of samples;
L is the number of classes;
N_i is the number of samples of class ω_i;
$\mathbf{X}_j^i$ is the j-th sample of class ω_i;
$\mathbf{M}_i$ is the mean of class ω_i.

The ADFM is thus the average distance that samples are located from the mean. Generally, decreasing ADFM will improve the performance of the minimum distance classifier. Figure 7.5 shows the ACMD and the ADFM of the 12 classes of Table 7.2. As can be seen, the ACMD increases as more features are added. However, the ADFM also increases.

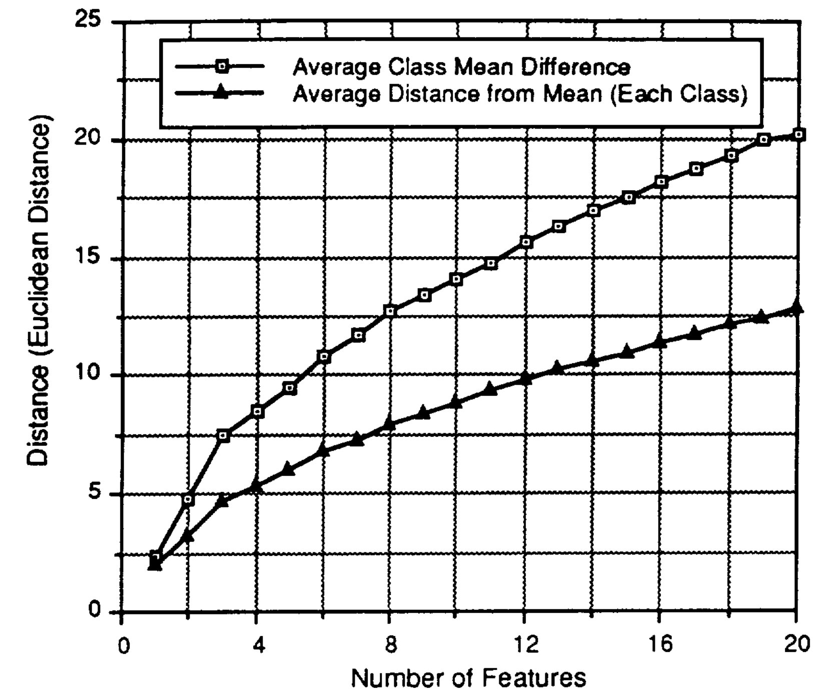

Figure 7.5 Graph of the Average Class Mean Difference and the Average Distance From Mean of the 12 classes of Table 7.2.

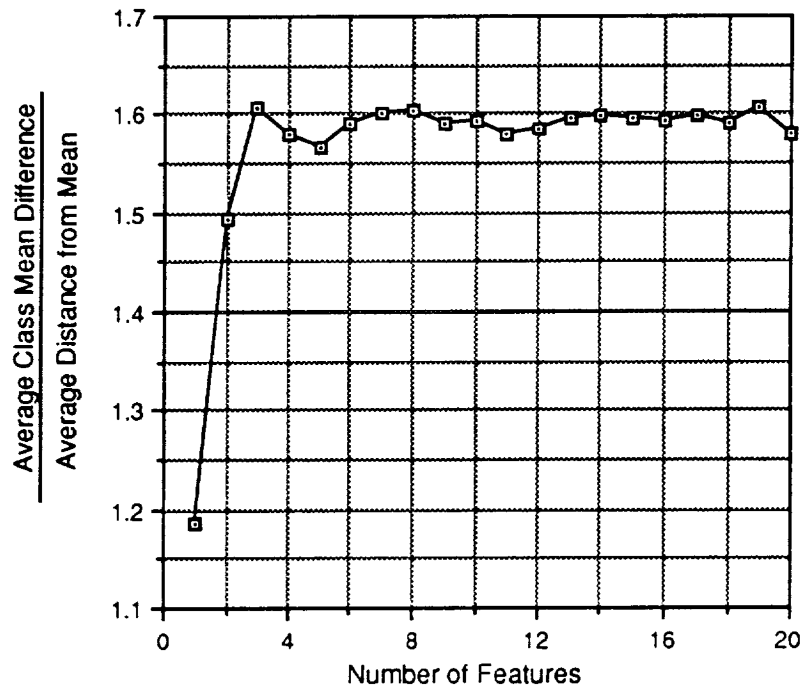

Figure 7.6 Ratio of the Average Class Mean Difference and the Average Distance From Mean of the 12 classes of Table 7.2.

Figure 7.6 shows the ratio of the ACMD and the ADFM. Note that the ratio increases up to 3 features and then is saturated thereafter. Though one should expect variations in this effect from problem to problem, the implication is that the performance of classifiers which mainly utilize class mean differences may not improve much in high dimensional data, especially when correlation between adjacent bands is high.

7.3.2 Eigenvalues of Covariance Matrix of High Dimensional Data

In high dimensional remotely sensed data, there is frequently very high correlation between adjacent bands, and most data are distributed along a few major components. Table 7.3 shows the eigenvalues (ordered by size) of the covariance matrix estimated from 643 samples of Summer Fallow collected at Finney County, Kansas in July 26, 1977, as well as proportions and accumulations of the eigenvalues. Figure 7.7 shows the magnitude of eigenvalues on a log scale. As can be seen, there are very large differences among eigenvalues. The ratio between the largest eigenvalue and the smallest is on the order of 10^6. A few eigenvalues are dominant and the rest are very small in value.

It can be seen from Table 7.3 that the largest 3 eigenvalues account for more than 95% of the total mean square value. The largest 8 eigenvalues account for more than 99%. Most variation of data occurs along a few eigenvectors corresponding to the largest eigenvalues and there is very little variation in the other eigenvectors. This indicates that, assuming a Gaussian distribution, the data will be distributed in the shape of an elongated hyperellipsoid with its origin at the mean of the data and whose semi-axes are in the directions of the eigenvectors of the covariance matrix of the data with lengths proportional to the corresponding eigenvalues. Since the lengths of the semi-axes are proportional to the eigenvalues, there are very large differences among the lengths of the semi-axes.

Table 7.3 Eigenvalues of covariance of high dimensional remotely sensed data.

	Eigenvalue	Proportion	Accumulation		Eigenvalue	Proportion	Accumulation
1	559.597	87.559	87.559	29	0.035	0.005	99.945
2	44.204	6.917	94.476	30	0.034	0.005	99.950
3	9.687	1.516	95.992	31	0.032	0.005	99.955
4	6.241	0.976	96.968	32	0.029	0.005	99.960
5	5.307	0.830	97.799	33	0.025	0.004	99.964
6	3.438	0.538	98.336	34	0.025	0.004	99.968
7	3.066	0.480	98.816	35	0.024	0.004	99.972
8	1.463	0.229	99.045	36	0.023	0.004	99.975
9	1.139	0.178	99.223	37	0.020	0.003	99.978
10	0.975	0.153	99.376	38	0.018	0.003	99.981
11	0.651	0.102	99.478	39	0.016	0.003	99.984
12	0.599	0.094	99.572	40	0.015	0.002	99.986
13	0.493	0.077	99.649	41	0.012	0.002	99.988
14	0.307	0.048	99.697	42	0.012	0.002	99.990
15	0.251	0.039	99.736	43	0.010	0.002	99.991
16	0.196	0.031	99.767	44	0.010	0.002	99.993
17	0.173	0.027	99.794	45	0.008	0.001	99.994
18	0.152	0.024	99.818	46	0.008	0.001	99.995
19	0.122	0.019	99.837	47	0.007	0.001	99.997
20	0.121	0.019	99.856	48	0.006	0.001	99.998
21	0.101	0.016	99.871	49	0.005	0.001	99.998
22	0.085	0.013	99.885	50	0.003	0.001	99.999
23	0.077	0.012	99.897	51	0.002	0.000	99.999
24	0.065	0.010	99.907	52	0.002	0.000	99.999
25	0.060	0.009	99.916	53	0.002	0.000	100.000
26	0.054	0.009	99.925	54	0.001	0.000	100.000
27	0.049	0.008	99.932	55	0.001	0.000	100.000
28	0.046	0.007	99.940	56	0.000	0.000	100.000

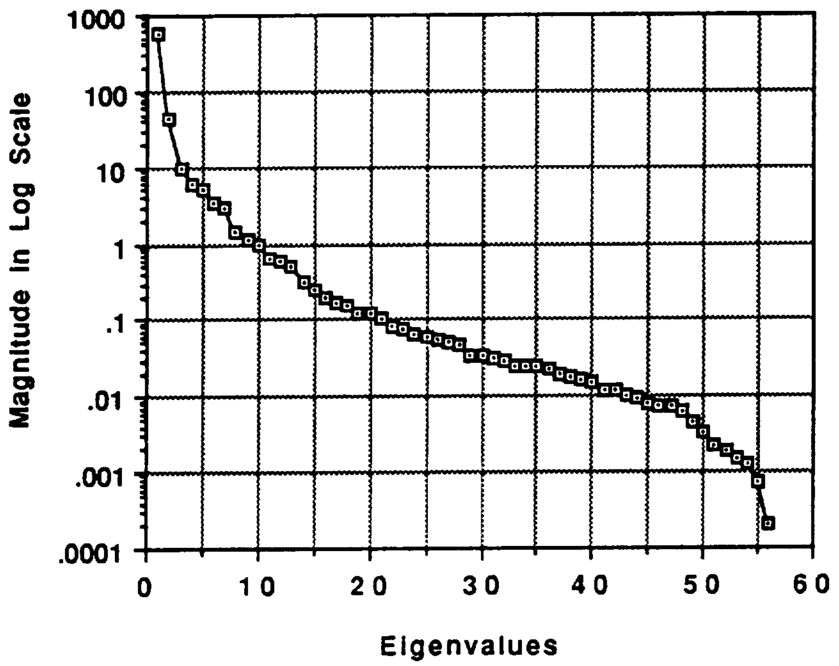

Figure 7.7 Magnitude of eigenvalues (log scale).

Without utilizing the second order statistics, a classifier such as the minimum distance classifier assumes that data are distributed in the shape of a hypersphere instead of hyperellipsoid. As a result, the minimum distance classifier defines a very ineffective decision boundary, particularly in high dimensional data. Figure 7.8 shows an example in two dimensional space. The two classes in Figure 7.8 are, in fact, quite separable by using second order statistics which give the information about the shape of the distribution, and in particular, the major component along which most data are distributed. However, the minimum distance classifier, using only the first order statistics, defines a very unsatisfactory decision boundary, causing avoidable errors. This phenomenon becomes more severe if data are distributed along a few major components. On the other hand, if classes are distributed in the shape of hypersphere, the minimum distance classifier will give a better performance.

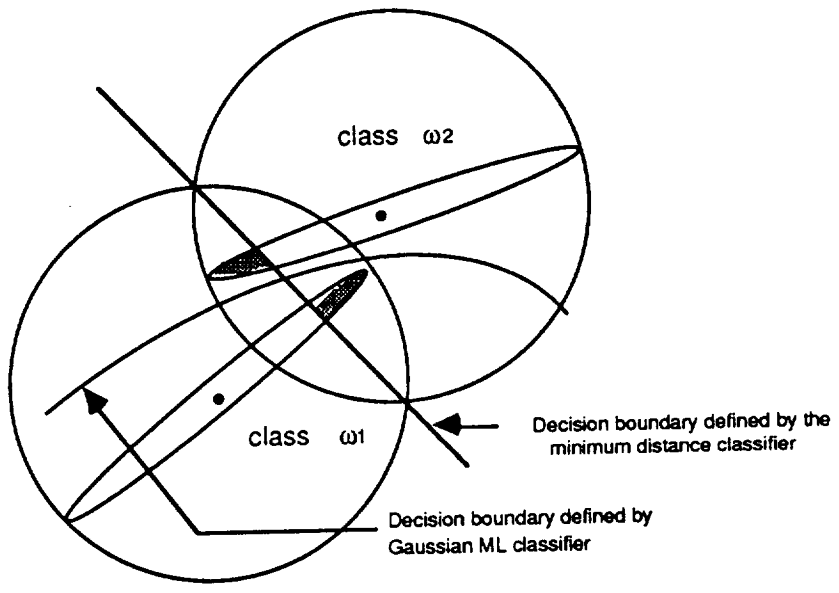
class ω2
class ω1
Decision boundary defined by the minimum distance classifier
Decision boundary defined by Gaussian ML classifier

Figure 7.8 Classification error of the minimum distance classifier.

7.3.3 Determinant of Covariance Matrix of High Dimensional Data

The determinant is equal to the product of the eigenvalues, i.e.,

$$\mathrm{DET} = \prod_{i=1}^{N} \lambda_i$$

a-3

As can be seen in Table 7.3, most of the eigenvalues of the covariance matrix of high dimensional data are very small in value. Therefore, determinants of high dimensional remotely sensed data will have very small values. Figure 7.9 shows the magnitudes of the determinants of the 12 classes for various number of features. In low dimensionality, the differences of determinants among classes are relatively small. As the dimensionality increases, the determinants decrease exponentially, indicating that the data are distributed in the highly elongated shape. In addition, there are significant differences between classes, indicating that there are significant differences in the actual volumes in which the classes are distributed.

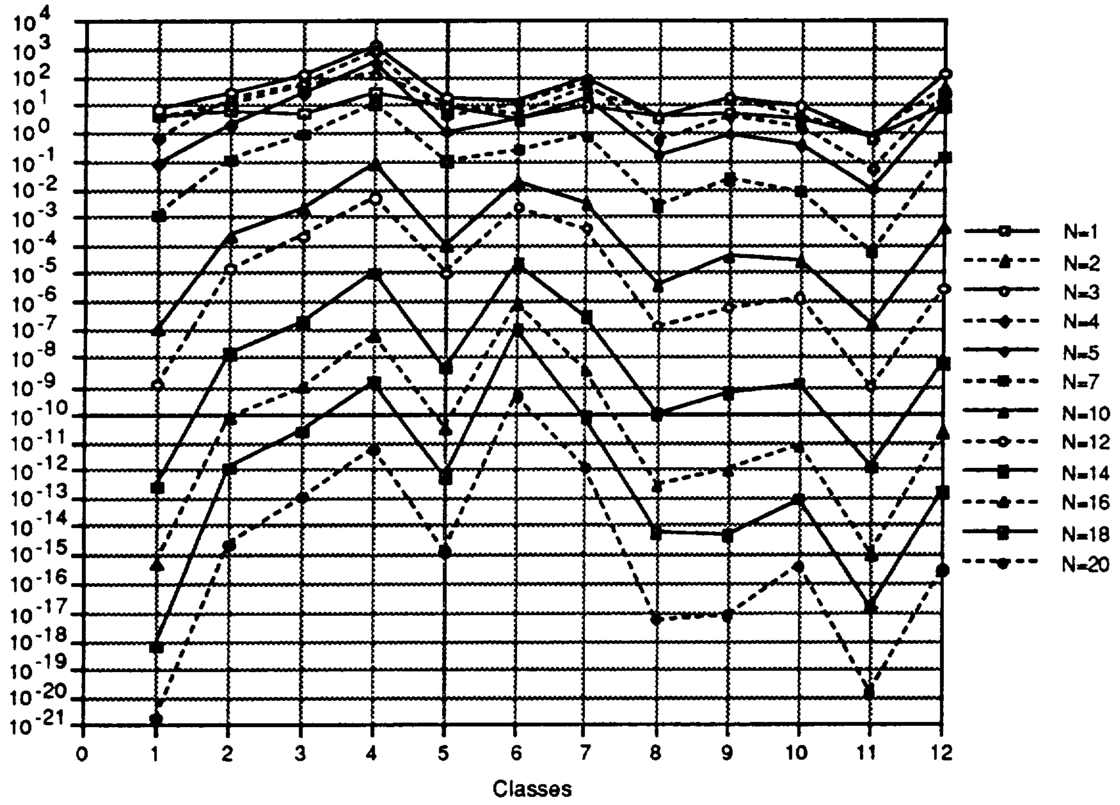

Figure 7.9 Determinant of the 12 classes.

7.4 Diagonalizing and The Number of Training Samples

7.4.1 Diagonalizing the Data

The limited performance of the minimum distance classifier in the previous section is mainly due to the fact that there is very high correlation between adjacent bands in high dimensional remotely sensed data. As a result, it is difficult to evaluate the roles of class mean differences and class covariance differences in discriminating between classes in high dimensional data. To better compare the roles of class mean differences (first order statistics) and class covariance differences (second order statistics), the entire data set is diagonalized (Fukunaga 1990), i.e., a linear transformation is applied to the data such that the transformed data will have a unit covariance matrix. Let,

$$\mathbf{Y} = \Lambda^{-\frac{1}{2}}\Phi^t\mathbf{X}$$

where Φ is a matrix whose column vectors are the eigenvectors of Σ_X, the covariance matrix of the original data

Λ is a diagonal matrix whose diagonal elements are eigenvalues of Σ_X, the covariance matrix of the original data

Then the covariance matrix of the transformed data $\mathbf{Y}$, Σ_Y, will be an identity matrix, i.e.,

$$\Sigma_Y = \mathbf{I}$$

It will be seen that this linear transformation affects only the performance of the minimum distance classifier. The performance of the Gaussian ML classifier is invariant under any linear transformation[2] since

$$(\mathbf{X} - \mathbf{M}_X)^t\Sigma_X^{-1}(\mathbf{X} - \mathbf{M}_X)$$

where $\mathbf{M}_X$ is the mean vector of $\mathbf{X}$ and Σ_X is the covariance matrix of $\mathbf{X}$

2 Note that this implies that any preprocessing procedure, e.g. calibration, which is merely a linear transformation of the data will not affect classification accuracy for a Gaussian ML classifier.

is invariant under any linear transformation if the transformation matrix is non-singular (Fukunaga 1990). After diagonalizing, it is expected that the performance of the minimum distance classifier will be improved since the diagonalization process makes the data distribution closer to the shape of hypersphere (Figure 7.8).

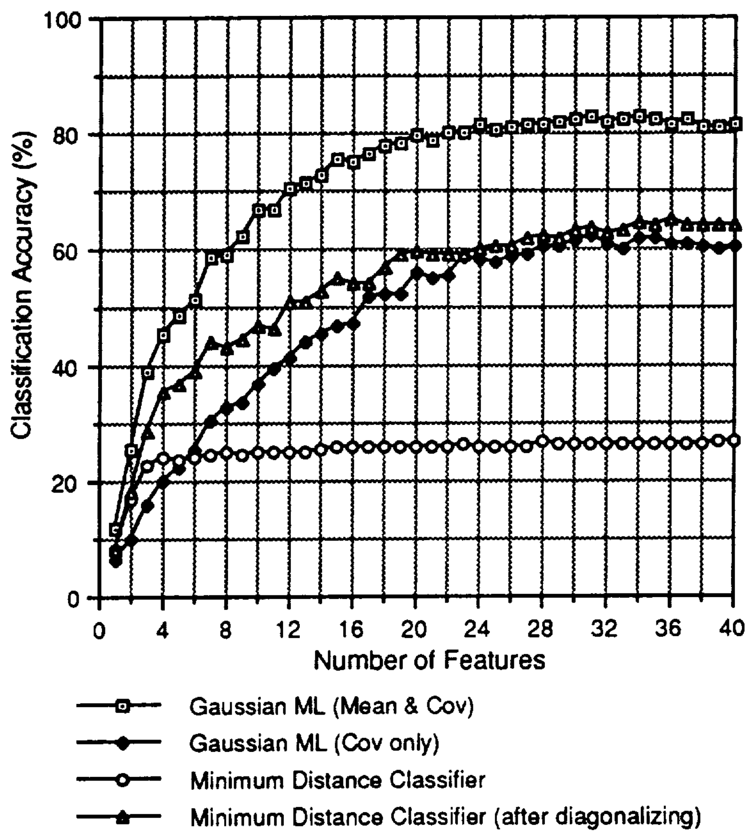

Figure 7.10 Performance comparison (100 training samples).

Figure 7.10 shows the classification accuracy vs. numbers of features after diagonalization. There are 40 multi-temporal classes. 100 randomly selected samples are used for training data and the rest are used for test. As expected, the Gaussian ML classifier shows the best performance and the peak accuracy of the Gaussian ML classifier occurs when the number of features is 31, achieving 82.8%. When more than 31 features are used, the performance of the Gaussian ML classifier begins to decrease slightly, indicating the Hughes phenomenon is occurring (Hughes 1968). The Gaussian ML classifier applied to the zero-mean data also shows peak performance with 31 features, achieving 62.4% classification accuracy. When more than 31 features are used, the Gaussian ML classifier applied to the zero-mean data also shows the Hughes phenomenon.

The minimum distance classifier applied to the original data shows very limited performance, achieving just 26.6% classification accuracy with 40 features. In fact, the performance of the minimum distance classifier is saturated after 4 features. After diagonalization, the performance of the minimum distance classifier is greatly improved, achieving 64.8% classification accuracy with 36 features. It appears that, when the data are diagonalized, class mean differences are more important than class covariance differences in discriminating between classes in this example. However, the difference in classification accuracy decreases as dimensionality increases. For example, when 4 features are used, the classification accuracy of the minimum distance classifier applied to the diagonalized data is 35.3% while the classification accuracy of the Gaussian ML classifier applied to the zero-mean data is 19.8%, a difference of 15.5%. When 31 features are used, the classification difference is just 1.3% It is interesting that the Hughes phenomenon of the minimum distance classifier occurs later compared with the Gaussian ML classifier. A possible reason is that the number of parameters the minimum distance classifier uses is much smaller than the number of parameters the Gaussian ML classifier uses.

7.4.2 Estimation of Parameters and Number of Training Samples

In supervised classification, parameters are estimated from training data. When the parameter estimation is not accurate, the performance of the classifier is affected. In particularly, when the number of training data is limited, adding more features does not necessarily improve the classification accuracy. In this section, we will illustrate how inaccurate estimation of parameters affect the performance of the minimum distance classifier and the Gaussian ML classifier applied to the zero-mean data.

Generally, the classification error is a function of two sets of data, training and test data and can be expressed by (Fukunaga 1990)

$$\varepsilon(\Theta_{train}, \Theta_{test})$$

where Θ_{train} and Θ_{test} are a set of parameters of training and test data.

In Fukunaga (1990), it is shown that the Bayes error, $\varepsilon(\Theta,\Theta)$, is bounded by two sample-based estimates, *i.e.*,

$$E\{\varepsilon(\hat{\Theta},\hat{\Theta})\} \leq \varepsilon(\Theta,\Theta) \leq E_{\hat{\Theta}_{test}}\{\varepsilon(\hat{\Theta},\hat{\Theta}_{test})\} \tag{7.1}$$

The term $\varepsilon(\hat{\Theta},\hat{\Theta}_{test})$ is obtained by generating two independent sample sets, $\hat{\Theta}$ and $\hat{\Theta}_{test}$, and using $\hat{\Theta}$ for training and $\hat{\Theta}_{test}$ for testing. $\varepsilon(\hat{\Theta},\hat{\Theta})$ is obtained by using the same data for training and test.

In the following test, the 3 classifiers are tested on the 40-class problem (Table 3.14). The average number of samples of the 40 classes is about 300. To obtain a lower bound of the Bayes error, all data are used for training and test (resubstitution method) (Fukunaga 1990). The *leave-one-out* method (Fukunaga 1990) is also used to obtain an upper bound of the Bayes error.

Figure 7.11 shows the performance comparison of the resubstitution method and the leave-one-out method. Let's compare Figure 7.11 and Figure 7.10 where 100 randomly chosen samples are used for training. When 40 features are used, the classification accuracy of the Gaussian ML classifier improved from 81.3%(100 training samples) to 93.8%(all data are used for training). However, the improvement of the Gaussian ML classifier applied to the zero-mean data is particularly noteworthy. The classification accuracy increased from 60.5%(100 training samples) to 86.1%(all data are used for training) with 40 features. When 100 training samples are used, the difference of the classification accuracies of the Gaussian ML classifier applied to the original data and the Gaussian ML classifier applied to the zero-mean data was 20.8% with 40 features. When all samples are used for training, the difference is reduced to 7.7%. On the other hand, the performance of the minimum distance classifier improves only slightly. The classification accuracy of the minimum distance classifier applied to the original data increased from 26.6%(100 training samples) to 27.5%(all data are used for training) with 40 features, and the classification accuracy of the minimum distance classifier applied to the diagonalized data increased from 64.2%(100 training samples) to 67.3%(all data are used for training).

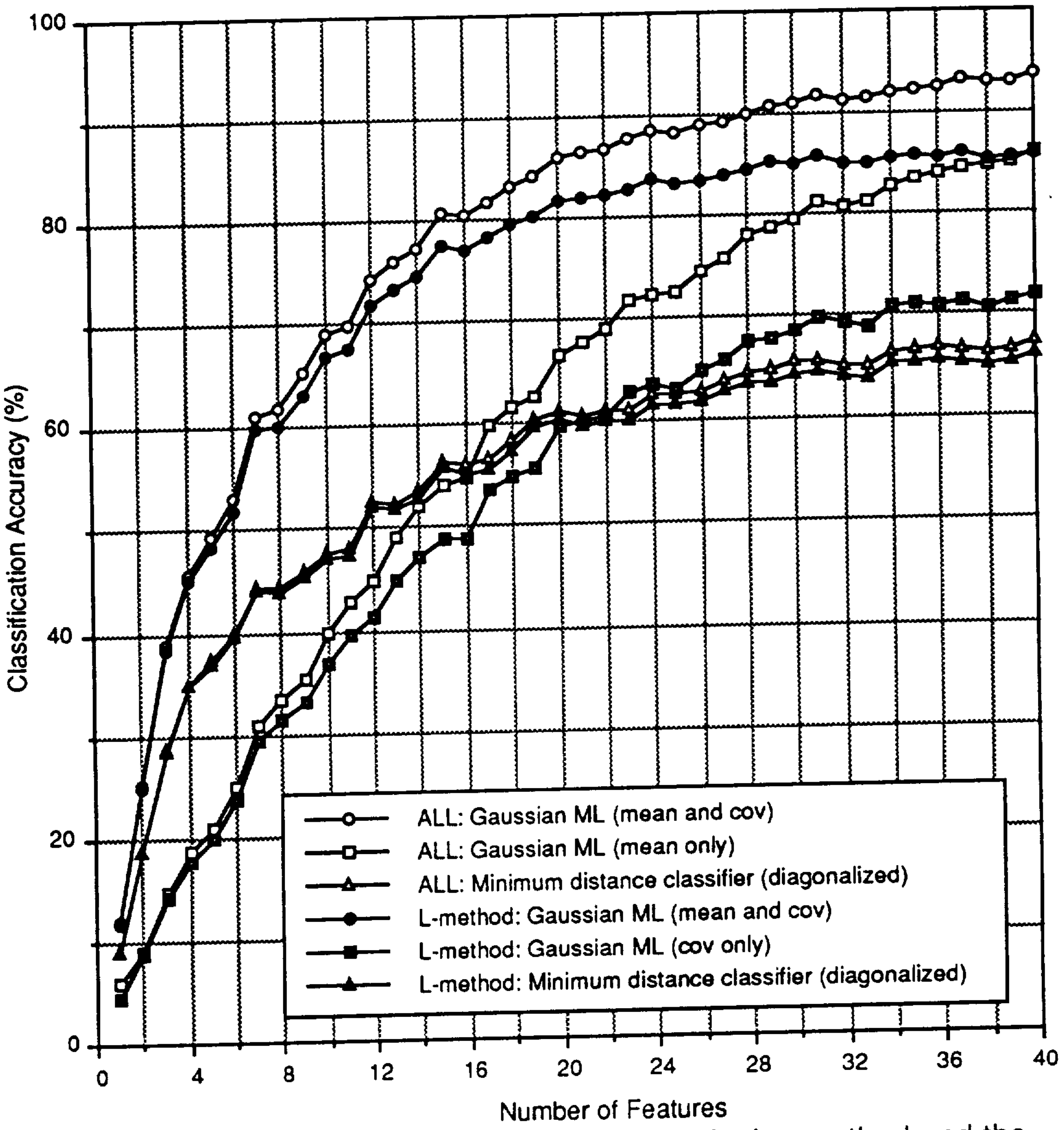

Figure 7.11 Performance comparison of the resubstitution method and the leave-one-out method.

In the leave-one-out method, the accuracy improvements are smaller. The classification accuracy of the Gaussian ML classifier is about 85.9% with 40 features and 71.9% for the Gaussian ML classifier with zero mean data. The classification accuracy of the minimum distance classifier is 66.1% with 40 features.

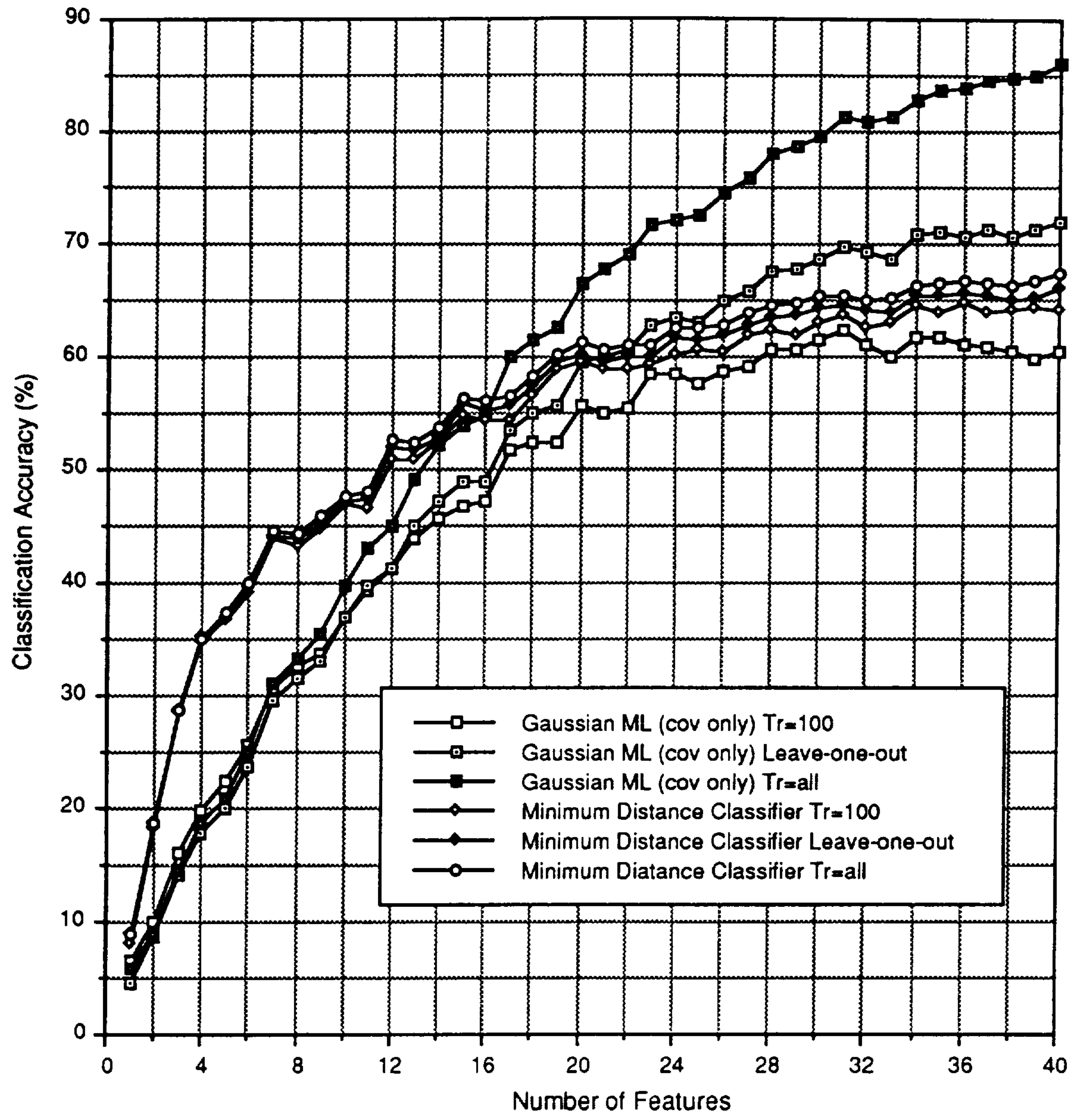

Figure 7.12 Performance comparison of the minimum distance classifier applied to the diagonalized data and the Gaussian ML classifier with the zero mean data for various numbers of training samples.

Figure 7.12 shows the classification accuracy vs. number of features when various numbers of training samples are used. Note that the performance of the Gaussian ML classifier with the zero mean data greatly improved when all data are used for training or the leave-one-out method is used, while the performance of the minimum distance classifier improved slightly. It is noted that, when all data are used for training or the leave-one-out method is used, the Gaussian ML classifier applied to the zero-mean data outperforms the minimum distance classifier in high dimensionality. Since the Bayes error is bounded by the two sample-based estimates (equation 7.1), it appears that the second order statistics play an increased role in discriminating between classes in high dimensionality.

In Figure 7.12, the difference between the resubstitution method and the leave-one-out method is large, resulting in a loose bound on the Bayes error. A reason for the large difference is that some of the classes have a relatively small number of samples. To overcome that problem, we generated data from the statistics estimated from the classes. 1000 samples were generated for each class. The resubstitution method and the leave-one-out method were applied to obtain a lower and a upper bound. Figure 7.13 shows the result. The classification accuracies of the Gaussian ML classifier, the Gaussian ML classifier applied to zero mean data, and the minimum distance classifier are 99.5%, 97.5%, and 56.9%, respectively, when all data are used for training and test, and 99.2%, 96.0%, and 54.3%, respectively, when the leave-one-out method is used. It is interesting that the Gaussian ML classifier applied to zero mean data shows almost the same performance as the Gaussian ML classifier applied to the original data in high dimensionality, significantly outperforming the minimum distance classifier.

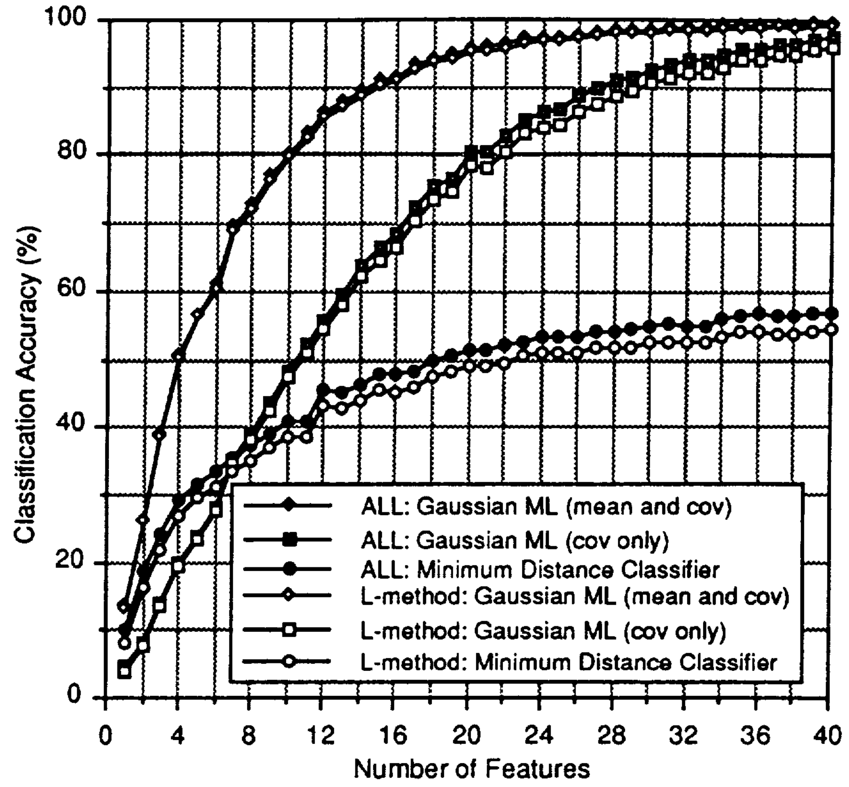

Figure 7.13 Performance comparison of generated data when all data are used for training and test, and the leave-one-out (L-method) is used.

In practice, estimation of the second order statistics of high dimensional data is a difficult problem, particularly when the number of training samples are limited. However, these results suggest that second order statistics provide a great potential for discriminating between classes in high dimensionality if the second order statistics can be accurately estimated. In many feature extraction algorithms, the lumped covariance is used [(Fukunaga 1990) and (Foley and Sammon 1975)]. However, the above results indicate that covariance differences among classes also provides important information in discriminating between classes in high dimensional data. Recently the possibility of obtaining a better estimation of parameters using a large number of unlabeled samples in addition to training samples has been shown, and this should be particularly relevant in the case of high dimensional data (Shahshahani and Landgrebe 1992).

7.5 Visualization of High Dimensional Data

As the dimensionality of data increases, it becomes more difficult to compare class statistics, and in particular, the second order statistics. For instance, it would not be feasible to print out mean vectors and covariance matrices of 200 dimensional data and compare them manually. Table 7.4 shows an example of a 20 dimensional correlation matrix. It is very difficult to manually perceive much from the numerical values; some type of visualization aid seems called for.

Table 7.4 Correlation Matrix of 20 dimensional data.

1.00																			
0.95	1.00																		
0.94	0.97	1.00																	
0.94	0.96	0.99	1.00																
0.93	0.95	0.98	0.99	1.00															
0.91	0.94	0.97	0.99	0.99	1.00														
0.90	0.93	0.96	0.98	0.99	1.00	1.00													
0.89	0.92	0.95	0.97	0.98	0.99	1.00	1.00												
0.88	0.91	0.95	0.97	0.98	0.99	0.99	1.00	1.00											
0.86	0.89	0.93	0.96	0.97	0.98	0.99	0.99	1.00	1.00										
0.85	0.88	0.92	0.95	0.96	0.98	0.98	0.99	0.99	1.00	1.00									
0.83	0.86	0.91	0.93	0.95	0.97	0.98	0.98	0.99	0.99	1.00	1.00								
0.82	0.85	0.90	0.93	0.94	0.96	0.97	0.98	0.98	0.99	1.00	1.00	1.00							
0.81	0.84	0.89	0.92	0.94	0.96	0.97	0.98	0.98	0.99	0.99	1.00	1.00	1.00						
0.79	0.82	0.87	0.90	0.92	0.94	0.96	0.97	0.97	0.98	0.98	0.98	0.98	0.99	1.00					
0.77	0.80	0.85	0.87	0.90	0.92	0.94	0.95	0.95	0.95	0.95	0.95	0.95	0.97	0.99	1.00				
0.76	0.78	0.83	0.85	0.88	0.90	0.92	0.94	0.94	0.94	0.94	0.93	0.93	0.95	0.98	0.99	1.00			
0.76	0.78	0.82	0.85	0.87	0.89	0.92	0.93	0.93	0.93	0.93	0.93	0.92	0.94	0.98	0.99	0.99	1.00		
0.75	0.77	0.81	0.84	0.86	0.89	0.91	0.92	0.93	0.93	0.93	0.92	0.92	0.94	0.98	0.99	1.00	0.99	1.00	
0.74	0.75	0.80	0.83	0.85	0.87	0.90	0.91	0.92	0.92	0.92	0.92	0.91	0.94	0.97	0.98	0.99	0.99	1.00	1.00

Kim and Swain proposed a method to visualize the magnitude of correlation using gray levels (Kim and Swain 1990). We further elaborate on this method and propose a visualization method of mean vectors and covariance matrices along with standard variations using a color coding scheme and a graph. We will call this visualization method of statistics the *statistics image*. Figure 7.14 shows the format of the statistics image. Statistics images consists of a color-coded correlation matrix, a mean graph with standard deviation and a color code. Figure 7.15 shows the palette design for the color code. Figure 7.16 shows the actual look of the color code for correlation matrix in gray scales. The color changes continuously from blue to red with blue indicating a correlation coefficient of –1 and red indicates that the correlation coefficient is 1. In the mean graph part, the mean vector is displayed plus or minus one standard deviation. At the bottom of the statistics image, the color code is added for easy comparison.

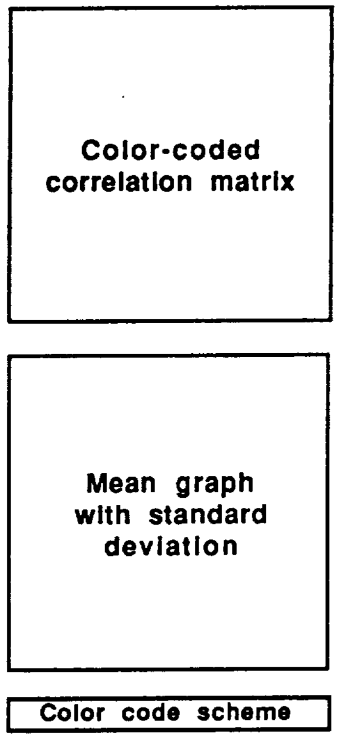
Color-coded correlation matrix
Mean graph with standard deviation
Color code scheme

Figure 7.14 Format of the statistics image.

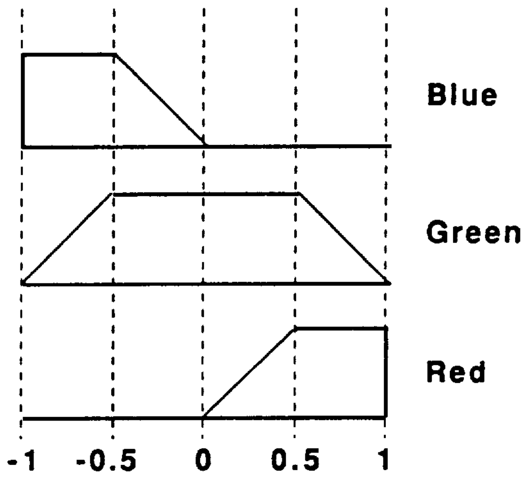

Figure 7.15 Palette design.

Figure 7.17 shows the statistics images of Spring Wheat, Oats, Summer Fallow, and Native Grass Pasture which were collected on July 26, 1978 in gray scale. The green lines in the images indicate water absorption bands. At a glance, one can subjectively perceive how each band is correlated and easily compare the statistics of the different classes. It is easy to see that there are significant differences in the class correlation, suggesting probable separability via a classifier. Figure 7.18 shows the statistics images of Spring Wheat collected on May 15 1978, June 2 1978, July 26 1978, and August 16 1978 in gray scale. The statistics images clearly show how the statistics of the Spring Wheat have changed over the period. The statistics image will provide a valuable means in visualizing statistics of high dimensional data.

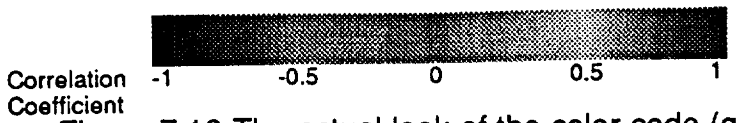

Figure 7.16 The actual look of the color code (gray scale).

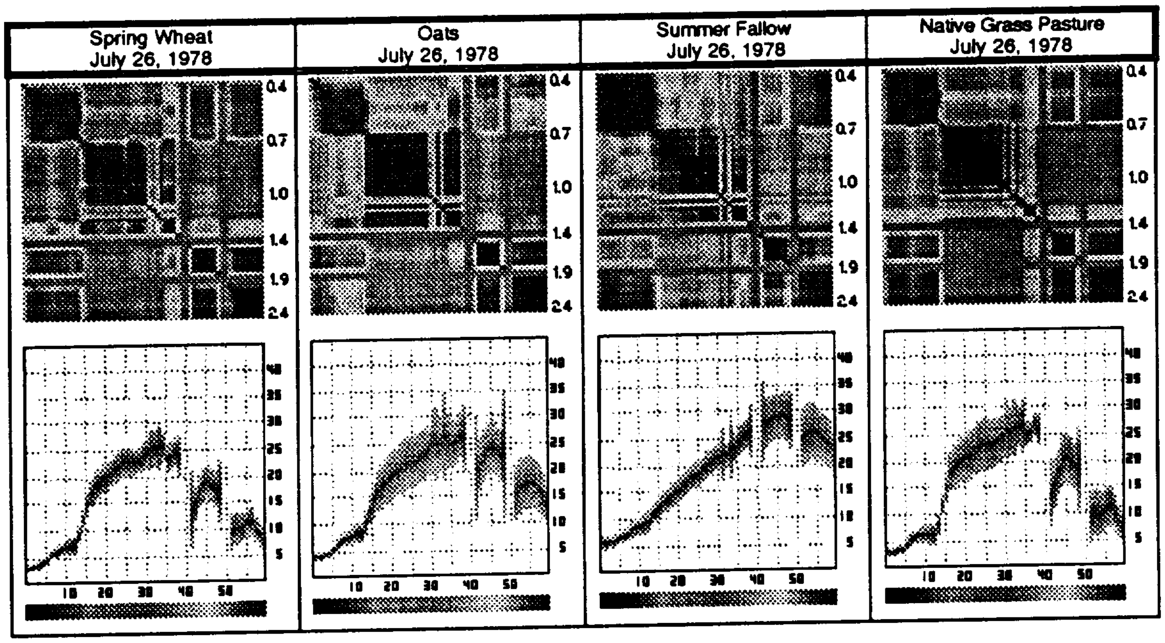

Figure 7.17 Statistics images of spring wheat, oats, summer fallow, and native grass pasture on July 26, 1978 (gray scale).

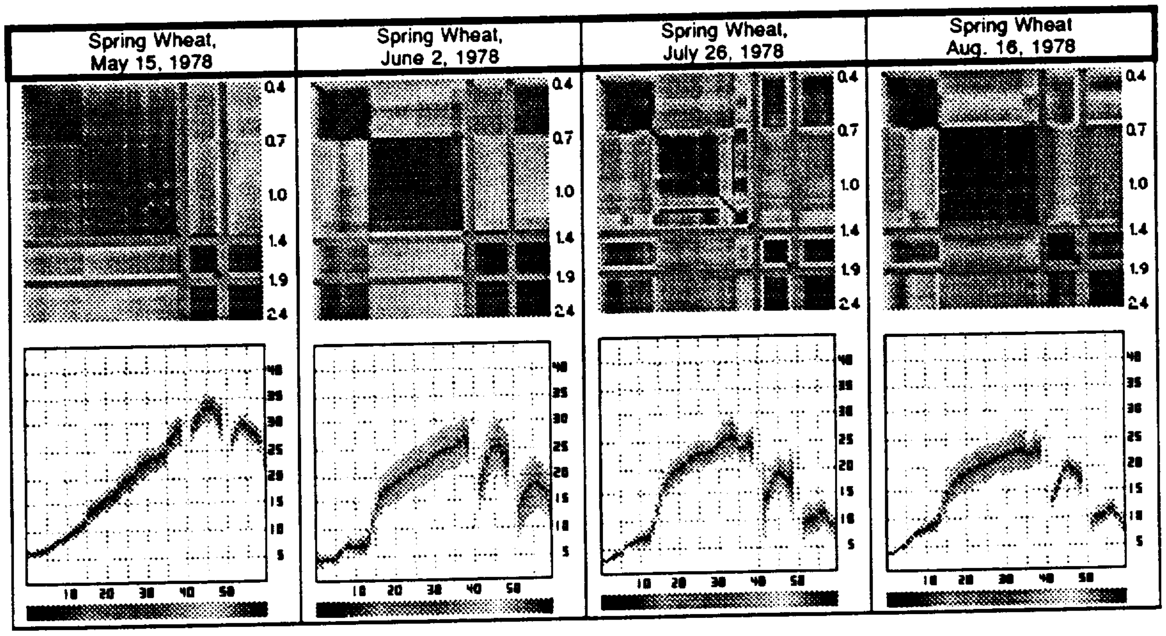

Figure 7.18 Statistics images of spring wheat over 4 months period (gray scale).

7.6 Conclusion

Advancement in sensor technology will provide data in much higher dimensions than previous sensors. Although such high dimensional data will present a substantial potential for deriving greater amounts of information, some new problems arise that have not been encountered in relatively low dimensional data. In this chapter, we examined the possible roles of first and second order statistics in discriminating between classes in high dimensional space. It is observed that a conventional minimum distance classifier which utilizes only the first order statistics failed to fully exploit the discriminating power of high dimensional data. By investigating the characteristics of high dimensional remotely sensed data, we demonstrated the reason for this limited performance. We also investigated how the degree of accuracy in estimating parameters affects the performance of classifiers and especially the potential of second order statistics in discriminating among classes in high dimensional data.

Recognizing the importance of second order statistics in high dimension data, it is clear that there is a greater need to better represent the second order statistics. For that purpose, we proposed a visualization method of the first and the second order statistics using a color coding scheme. By displaying the first and the second order statistics using this scheme, one can more easily compare spectral classes and visualize information about the statistics of the classes.

CHAPTER 8 SUMMARY AND SUGGESTIONS FOR FURTHER WORK

8.1 Summary

In this research, three main subjects are studied: (1) fast likelihood classification; (2) a new feature extraction algorithm; (3) characteristics of high dimensional data and problems in analyzing high dimensional data.

In Chapter 2, a fast likelihood classification was proposed to reduce the processing time of high dimensional data. As the dimensionality and the number of classes grow, the computation time becomes an important factor. Based upon the recognition that only a small number of classes are close to each other even when there are a large number of classes, a multistage classification was proposed. In the early stages where a fraction of the total features are used, classes whose likelihood values are smaller than a threshold are truncated, i.e., eliminated from further consideration so that the number of classes for which likelihood values are to be calculated at the following stages is reduced. It was shown that the computing time can be reduced by a factor of 3 to 7 using the proposed multistage classification while maintaining essentially same accuracies when the Gaussian ML classifier is used. This method will make it possible to extract detailed information from high dimensional data without increasing the processing time significantly.

In Chapter 3, a new feature extraction algorithm was proposed which better utilizes the potential of high dimensional data. The method is directly based on the decision boundary. It was shown that all the necessary features for classification can be extracted from the decision boundary. The proposed decision boundary feature extraction algorithm has desirable properties: (1) it does not deteriorate when there is little or no difference in mean vectors or when there is little or no differences in covariance matrices; (2) it predicts the minimum number of features necessary to achieve the same classification accuracy as in the original space; (3) it can be used both for parametric classifiers and non-parametric classifiers. In Chapter 3, the decision boundary feature extraction algorithm was applied to parametric classifiers. It was shown the performance of the decision boundary feature extraction method compares favorably with those of the conventional methods.

In Chapter 4, the decision boundary feature extraction algorithm was adapted to non-parametric classifiers. Since non-parametric classifiers do not define decision boundaries in analytic form, decision boundaries must be found numerically. In Chapter 5, the decision boundary feature extraction algorithm was applied to neural networks. First, a feature extraction method for neural networks using the Parzen density estimator was proposed. To apply the decision boundary feature extraction method directly to neural networks, we defined the decision boundary in neural networks. From the decision boundary, a new feature set is calculated. Experiments showed that the decision boundary feature extraction method works well with neural networks.

In Chapter 6, the discriminant feature extraction method, which is a generalization of the decision boundary feature extraction method, was proposed. Comparisons between the decision boundary feature extraction method and the discriminant feature extraction method were made.

In Chapter 7, some problems in analyzing high dimensional data are investigated. In particular, the increased importance of the second order statistics were studied. We also investigated how inaccurate estimation of first order and second order statistics affect the performance of classifiers in discriminating between classes in high dimensionality. To help human interpretation and perception of the second order statistics of high dimensional data, a visualization method of the second order statistics using a color code and a graph was proposed.

8.2 Suggestions for Further Work

The high dimensional multispectral imagery that future sensors are projected to generate will provide a great potential for analyzing the Earth resources. For example, the HIRIS instrument will generate image data in 192 spectral bands. In processing such high dimensional data, there will be many challenges to be overcome. It will be almost infeasible to use all 192 bands in analysis. First of all, estimation of statistics of such high dimensional data will be a very difficult problem, particularly when the number of training samples is limited. As a result, using all 192 bands for analysis will not necessarily produce an improved result. Figure 8.1 shows an example. There are 6 classes and Table 8.1 provides information about the classes.

Table 8.1 Class description of the multi-temporal 6 classes.

Date	Location	Species	No. Sample
May 3, 1977	Finney CO. KS.	Winter Wheat	658
May 3, 1977	Finney CO. KS.	Unknown Crops	682
March 8, 1977	Finney CO. KS.	Winter Wheat	691
March 8, 1977	Finney CO. KS.	Unknown Crops	619
June 26, 1977	Finney CO. KS.	Winter Wheat	677
June 26, 1977	Finney CO. KS.	Summer Fallow	643

Let 100 randomly selected samples be used for training and the rest for test. As can be seen from Figure 8.1, the peak of the classification accuracy occurs when 29 features are used. When more than 29 features are used, the classification accuracy actually begins to decrease.

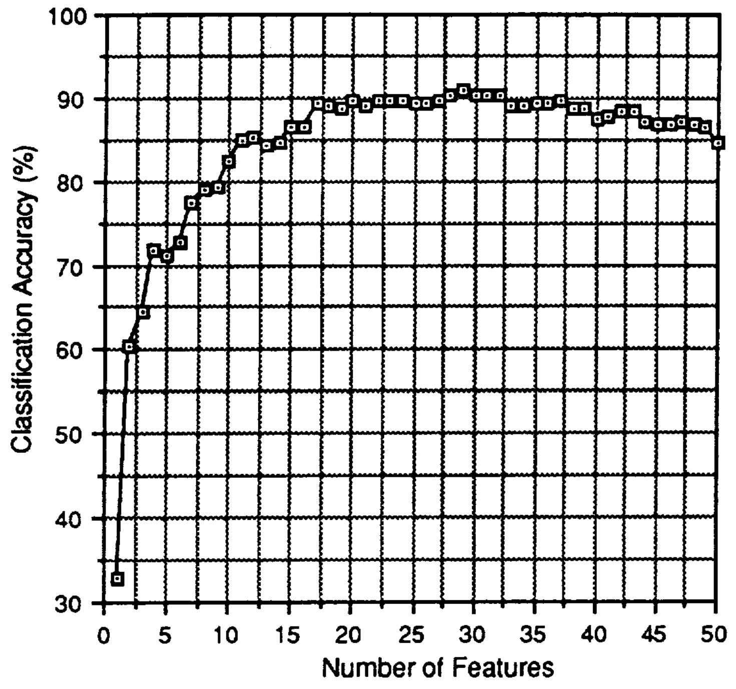

Figure 8.1 Classification accuracy vs. number of features.

In addition, if feature selection/extraction is done based on the estimated statistics of such high dimensional data, the resulting feature set may not be reliable. Figure 8.2 shows an example. There are 6 classes and Table 8.1 provides information on the classes. Again let 100 randomly selected samples be used for training and the rest for test. The decision boundary feature extraction method was applied to 29 dimensional data and 50 dimensional data. As can be seen, the classification accuracy with 29 dimensional data is better than that with 50 dimensional data. The result indicates that when the number of training samples is limited, using more features results in a poorer estimation of statistics which, in turn, decreases the performance of the feature extraction method which uses the estimated statistics.

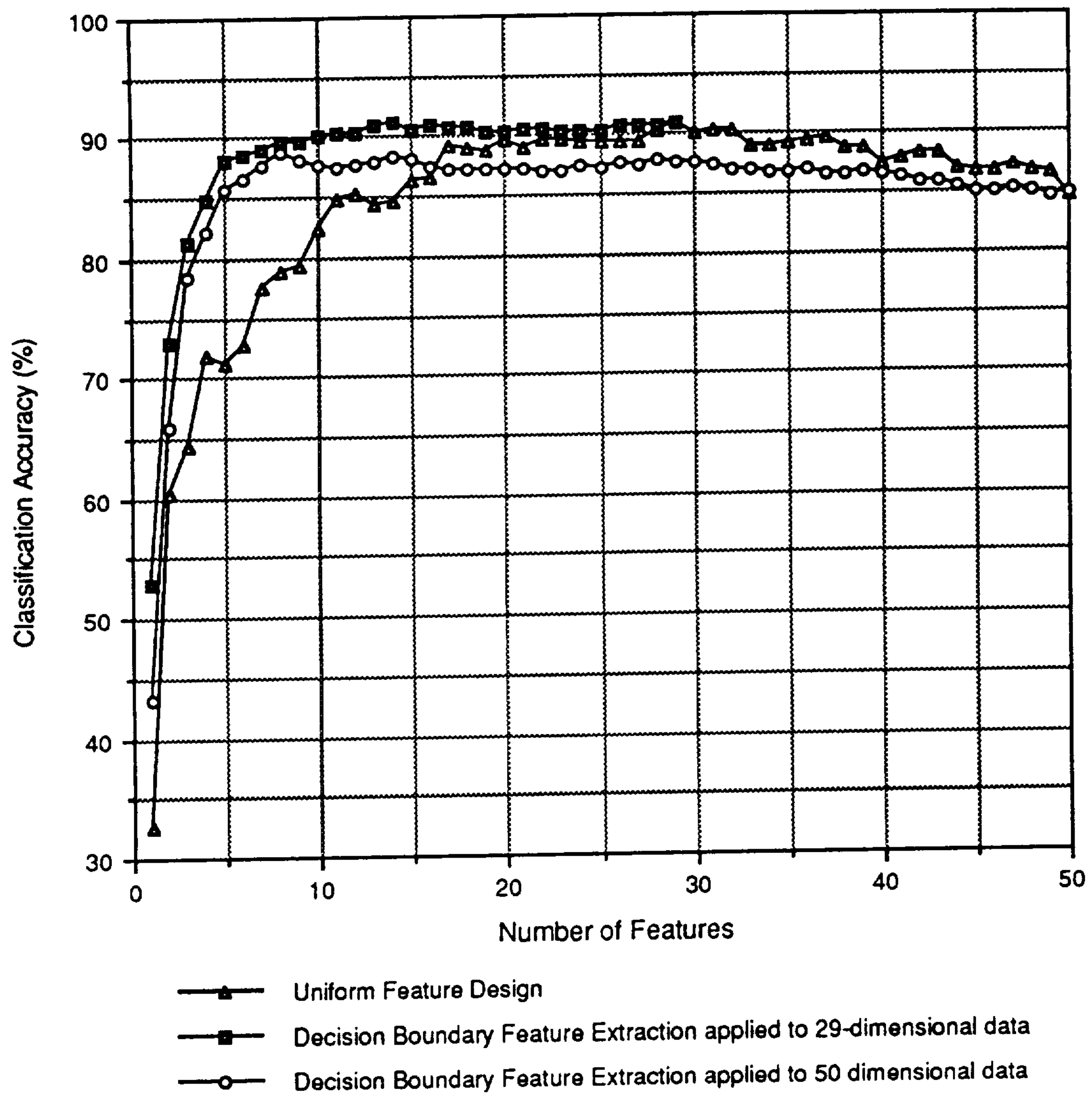

Figure 8.2 Feature extraction and number of features.

Thus, it is desirable to reduce the original dimensionality using some kinds of preprocessing techniques such that an estimation of statistics in the reduced dimensionality can be reliable. Then feature selection/extraction methods can be applied at the reduced dimensionality, further reducing the dimensionality. Finally, some classification/analysis techniques can be applied to the new data set selected by the feature selection/extraction method. Figure 8.3 illustrates such a processing scheme for high dimensional data. With the FSS data which has 60 spectral bands, it was observed that a dimensionality of about 20-30 gave the peak performance for preprocessing. In most cases, 10-20 features gave about the maximum classification accuracy. However, these number can be different depending on the original dimensionality, the complexity of problem, the number of available training samples, the quality of estimation of statistics, etc. Analytically determining the dimensionality for preprocessing and for classification/analysis is one of the important topics in analyzing high dimensional data.

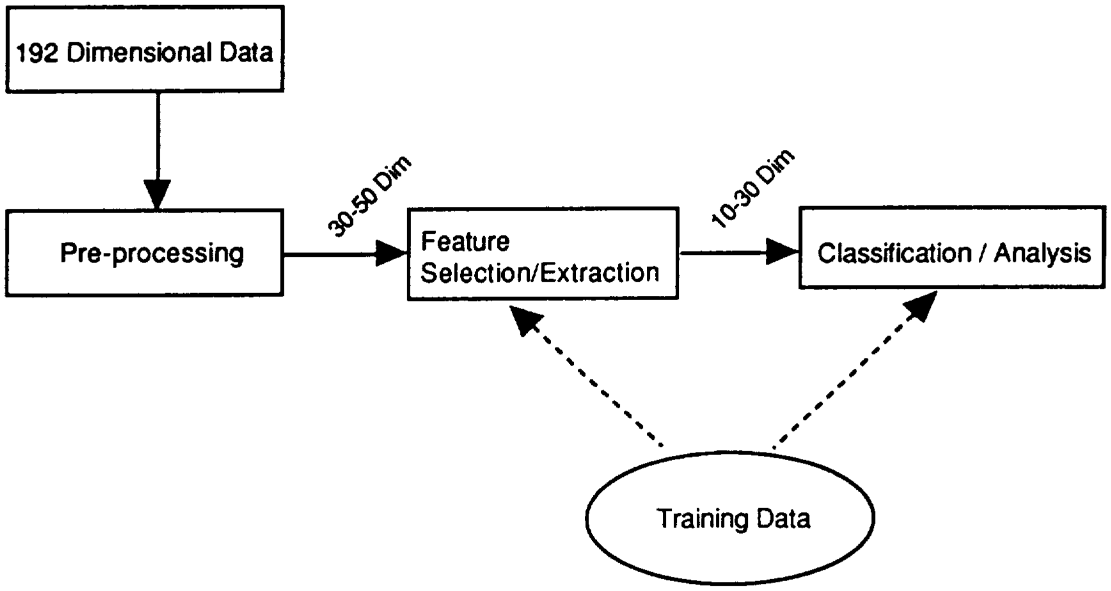

Figure 8.3 Pre-processing of high dimensional data.

The preprocessing techniques must not be too complex nor depend too much on the estimated statistics. Otherwise, the Hughes phenomenon may still occur. In this research, a band combination procedure (Uniform Feature Design) has been used as the preprocessing technique. Although the band combination procedure (Uniform Feature Design) has given acceptable and reliable results, the method is not optimum. Another possible way is to base the preprocessing on the estimated statistics of the whole data set [(Wiersma and Landgrebe 1980) and (Chen & Landgrebe 1989)]. Using the whole data set, it is expected that the estimation of parameters may be more accurate.

More research in the preprocessing techniques will definitely enrich the benefits of the high dimensional data, and improve the performance of classifiers and analyzer.

LIST OF REFERENCES

Ardanuy, P. E., D. Han, and V. V. Salomonson, "The Moderate Resolution Imaging Spectrometer (MODIS) Science and Data System Requirements," IEEE Trans. Geoscience and Remote Sensing, vol. 29, no. 1, pp. 75-88, 1991.

Argentiero, P., R. Chin, and P. Beaudet, "An Automated Approach to the Design of Decision Tree Classifiers," IEEE Trans. Pattern Analyses and Machine Intelligence, Vol. PAMI-4, No. 1, pp. 51-57, January, 1982.

Benediktsson, J. A., P. H. Swain and O.K. Ersoy, "Neural Network Approaches Versus Statistical Methods in Classification of Multisource Remote Sensing Data," IEEE Transactions on Geosciences and Remote Sensing, vol. GE-28, no. 4., pp. 540-552, July 1990.

Biehl, L. L., M. E. Bauer, B. F. Robinson, C. S. T. Daughtry, L. F. Silva and D. E. Pitts, "A Crops and Soils Data Base For Scene Radiation Research," Proceedings of the Machine Processing of Remotely Sensed Data Symposium, West Lafayette, IN 1982, pp. 169-177.

Chang, L. P. and T. Pavlidis, "Fuzzy Decision Tree Algorithms," IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-7, No. 1, pp28-35, 1977.

Chen, Chin-Chien Thomas and D. A. Landgrebe, "A Spectral Feature Design System for the HIRIS/MODIS Era," IEEE Trans. Geosci. Remote Sensing, vol. GE-17, pp. 681-686, November 1989.

Chien, Y. T. and K. S. Fu, "A Modified Sequential Recognition Machine Using Time-Varying Stopping Boundaries," IEEE Trans. Information Theory, Vol. IT-12, No. 2, pp. 206-214, 1966.

Cover, T. M. and P. E. Hart, "Nearest Neighbor Pattern Classification," IEEE Trans. Information Theory, Vol. IT-13, pp. 21-27, 1967.

Cullen, C. G., "Matrices and Linear Transformation, Addison Wesley," 1972.

Decell, H. P. and L. F. Guseman, "Linear Feature Selection with Applications," Pattern Recognition, Vol. 11, pp. 55-63, 1979.

Decell, H. P., P. Odell, and W. A. Coberly, "Linear Dimension Reduction and Bayes Classification," Pattern Recognition, Vol. 13, No. 3, pp. 241-243, 1981.

Devijver, P. & J. Kittler, "Pattern Recognition: A Statistical Approach, Prentice/Hall International," 1982.

Duchene, J. and S. Leclercq, "An Optimal Transformation for Discriminant and Principal Component Analysis," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-10, No. 6, pp. 978-983, Nov. 1988.

Duda, R. O. and P. E. Hart, "Pattern Classification and Scene Analysis," John Wiley & Sons, 1973.

Eppler, E., "Canonical Analysis for Increased Classification Speed and Channel Selection," IEEE Trans. Geosci. Remote Sensing, vol. GE-14, No. 1, pp. 26-33, 1976.

Ersoy, O. and D. Hong, "Parallel, Self-Organizing, Hierarchical Neural Networks," IEEE Trans. on Neural Networks, vol. 1, No. 2, June 1990.

Faux, I. D. and M. J. Pratt, "Computational Geometry for Design and Manufacture," Ellis Horwook, 1981.

Feiveson, A. H., "Classification by thresholding," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No.1, pp. 48-54, 1983.

Fisher, W. A., R. J. Fugimoto, and R. C. Smithson, "A Programmable Analog Neural Network Processor," IEEE Trans. on Neural Networks, Vol. 2, No. 2, pp. 222-229, March 1991.

Foley, D. H. and J. W. Sammon, "An Optimal Set of Discriminant Vectors," IEEE Trans. Computer, vol. C-24, No. 3, pp.281-289, Mar. 1975.

Fu, K. S., "A sequential decision model for optimum recognition," in Biological Prototypes and Synthetic Systems, Vol. I. Plenum Press, New York, 1962.

Fukunaga, K. and W. L. G. Koontz, "Application of the Karhunen-Loeve Expansion to Feature Selection and Ordering," IEEE Trans. Computer, Vol. C-19, No. 4, pp. 311-318, April 1970.

Fukunaga, K. and R. D. Short, "Generalized Clustering for Problem Localization," IEEE Transactions on Computers, Vol. C-27, No. 2, pp. 176-181, Feb. 1978.

Fukunaga, K. and J. M. Mantock, "Nonparametric Discriminant Analysis," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 6, pp. 671-678, November 1983.

Fukunaga, K. and D. M. Hummels, "Bayes Error Estimation Using Parzen and k-NN Procedures," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No. 5, pp. 634-643, September 1987.

Fukunaga, K., "Introduction to Statistical Pattern Recognition," Academic Press, 2dn edition, New York, 1990.

Fukushima, F. and N. Wake, "Handwritten Alphanumeric Character Recognition by the Neocognitron," IEEE Trans. on Neural Networks, vol. 2, No. 3, pp. 355-365, May 1991.

Goetz, A. F. H. and M. Herring, "The High Resolution Imaging Spectrometer (HIRIS) for Eos," IEEE Transactions on Geoscience and Remote Sensing, Vol. 27, No. 2, pp. 136-144, 1989.

Hertz, J., A. Krogh, and R. G. Palmer, "Introduction to the Theory of Neural Computation," Addison-Wesley Publishing Company, 1991.

Heydorn, R. P., "Redundancy in Feature Extraction," IEEE Trans. Computer, pp.1051-1054, Sep. 1971.

Hughes, G. F., "On the Mean Accuracy of Statistical Pattern Recognizers," IEEE Trans. Information Theory, vol. IT-14, No. 1, pp. 55-63, 1968.

Kailath, T., "The Divergence and Bhattacharyya Distance Measures in Signal Selection," IEEE Trans. Communication Technology, vol. COM-15, No. 1, pp. 52-60, 1967

Kazakos, D., "On the Optimal Linear Feature," IEEE Trans. Information Theory, vol. IT-24, No. 5, pp.651-652, Sept. 1978.

Kim, H. and P. H. Swain, "A Method of Classification for Multisource Data in Remote Sensing Based on Interval-Valued Probabilities," Ph. D dissertation, Purdue University, West Lafayette, Indiana, 1990.

Landgrebe, D. A., "Description and Results of the LARS/GE Data Compression Study," LARS Information Note 021171, Purdue University, Indiana, 1971.

Lee, C. and D. A. Landgrebe, "Fast Multistage Gaussian Maximum Likelihood Classifier," in Proc. IEEE International Geoscience & Remote Sensing Symposium, pp. 349-352, 1990.

Lee, C. and D. A. Landgrebe, "Decision Boundary Feature Selection for Non-Parametric Classifiers," in Proc. SPSE's 44th Annual Conference, pp. 475-478, May 1991-1.

Lee, C. and D. A. Landgrebe, "Feature Selection Based on Decision Boundaries," in Proc. IEEE International Geoscience & Remote Sensing Symposium, pp. 1471-1474, June 1991-2.

Lee, C. and D. A. Landgrebe, "Fast Likelihood Classification," IEEE Transactions on Geoscience and Remote Sensing, Vol. GE-29, pp. 509-517, July 1991-3.

Lee, C. and D. A. Landgrebe, "Analyzing High Dimensional Data," in Proc. IEEE International Geoscience & Remote Sensing Symposium (IGARSS), pp. 561-563, May 1992-1.

Lee, C. and D. A. Landgrebe, "Feature Selection for Neural Network Using Parzen Density Estimator," in Proc. IEEE International Geoscience & Remote Sensing Symposium (IGARSS), pp. 839-841, May 1992-2.

Lee, C. and D. A. Landgrebe, "Decision Boundary Feature Extraction for Neural Networks," in Proc. IEEE International Conf. on Systems, Man, and Cybernetics (SMC '92), pp. 1053-1058, 1992-3.

Lee, C. and D. A. Landgrebe, "Discriminant Feature Extraction for Parametric and Non-Parametric Classification," in Proc. IEEE International Conf. on Systems, Man, and Cybernetics (SMC '92), pp. 1345-1350, 1992-4.

Lippmann, R., "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, April 1987.

Longstaff, I. D., "On Extensions to Fisher's Linear Discriminant Function", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No. 2, March 1987.

Malina, W., "On an Extended Fisher Criterion for Feature selection", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-3, No. 5, September 1981.

Malina, W., "Some Multiclass Fisher Feature Selection Algorithms and their Comparison with Karhunen-Loeve Algorithms," Pattern Recognition Letters 6, pp. 279-285, 1987.

McEliece, R. J. et al., "The Capacity of the Hopfield Associate Memory," IEEE Trans. Information Theory, vol. IT-33, pp. 461-82, 1987.

Merembeck, B. F. and B. J. Turner, "Directed canonical analysis and the performance of classifiers under its associated linear transformation," IEEE Trans. on Geoscience and Remote Sensing, Vol. GE-18, pp. 190-196, 1980.

Moonpenn, A., J. Lambe, and A. P. Thakoor, "Electronic Implementation of Associative Memory Based on Neural Network Models," IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-17, No. 2, pp. 325-331, March/April, 1987.

Muasher, M. J. and D. A. Landgrebe, "The K-L expansion as an effective feature ordering technique for limited training sample size," IEEE Trans. Geoscience and Remote Sensing, Vol GE-21, pp. 438-441, 1983.

Parzen, E. "On the Estimation of a Probability Density Function and the Mode," Ann. Math. Stat., vol. 33, pp. 1065-1076, 1962.

Patrick, E. A. and F. P. Fischer II, "Nonparametric Feature Selection," IEEE Trans. on Information Theory, Vol. IT-15, No. 5, pp.577-584, 1969.

Porter, W. M., T. G. Chrien, E. G. Hansen, and C. M. Sarture, "Evolution of the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) Flight and Ground Data Processing System," Proc. SPIE, vol. 1298, pp. 11-17, Orlando FL, April 1990.

Riccia, G. and A. Shapiro, "Fisher Discriminant Analysis and Factor Analysis," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 1., pp. 99-104, January 1983.

Richards, J. A., "Remote Sensing Digital Image Analysis," Springer-Verlag, 1986.

Shahshahani, B. and D. A. Landgrebe, "Using Partially Labeled Data for Normal Mixture Identification with Application to Class Definition"," in Proc. IEEE International Geoscience & Remote Sensing Symposium (IGARSS), pp. 1603-1605, May 1992.

Short, R. D. and K. Fukunaga, "Feature Extraction Using Problem Localization," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, No. 3, pp. 323-326, May 1982.

Silverman, B. W., "Density Estimation for Statistics and Data Analysis," Chapman and Hall, 1986.

Swain, P. H. & R. C. King, "Two Effective Feature Selection Criteria for Multispectral Remote Sensing", Proc. First Int. Joint Conf. on Pattern Recognition, 536-540, Nov. 1973.

Swain, P. H. and H. Hauska, "The decision tree classifier: Design and potential," IEEE Trans. Geoscience Electronics, vol. GE-15, pp. 142-147, 1977.

Swain, P. H. & S. M. Davis, "Remote Sensing: The Quantitative Approach," McGraw–Hill, 1978.

Tubbs, J. D., W. A. Coberly, and D. M. Young, "Linear Dimension Reduction and Bayes Classification with Unknown Population Parameters," Pattern Recognition, Vol. 15, No. 3, pp.167-172, 1982.

Wald, A., "Sequential analysis," Wiley, New York, 1947.

Wang, Q. R. and C. Suen, "Large Tree Classifier with Heuristic Search and Global Training," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No. 1, pp. 91-102, January 1987.

Wasserman, P., "Neural Computing: Theory and Practice," Van Nostrand Reinhold, New York, 1989.

Wiersma, D. J. and D. A. Landgrebe, "Analytical Design of Multispectral Sensors," IEEE Trans. Geoscience and Remote Sensing, vol. 18, no. 2, pp. 180-189, 1980.

Yasunaga, M. et al., "A Self-Learning Neural Network Composed of 1152 Digital Neurons in Wafer-Scale LSIs," International Joint Conference on Neural Networks, Vol. 3, pp. 1844-1849, Singapore, 1991.

Young, D. M., P. L. Odell, and V. R. Marco, "Optimal Linear Feature Selection for a General Class of Statistical Pattern Recognition Models," Pattern Recognition Letters 3, 161-165, 1985.

Appendix A
Normal Vector to Decision Boundary

In order to find the decision boundary feature matrix of a pattern classification problem, one must be able to find a vector normal to the decision boundary at a point on the decision boundary. Under some conditions which are met in most pattern classification problems, one can find a vector normal to the decision boundary at a point on decision boundary using the following theorem.

Theorem A.1 If $\nabla h \neq 0$ at $\mathbf{X}_0$ and it is continuous in the neighborhood of $\mathbf{X}_0$, then the vector normal to the decision boundary at $\mathbf{X}_0$ is given by (Faux and Pratt 1981)

$$\nabla h(\mathbf{X})\ (\mathbf{X}=\mathbf{X}_0)$$

If the Gaussian ML classifier is used assuming a Gaussian distribution for each class, h(**X**) is given by

$$\begin{aligned} h(\mathbf{X}) &= -\ln\frac{P(\mathbf{X}|\omega_1)}{P(\mathbf{X}|\omega_2)} = -\ln P(\mathbf{X}|\omega_1) + \ln P(\mathbf{X}|\omega_2) \\ &= \frac{1}{2}(\mathbf{X}-\mathbf{M}_1)^t\Sigma_1^{-1}(\mathbf{X}-\mathbf{M}_1) + \frac{1}{2}\ln|\Sigma_1| - \frac{1}{2}(\mathbf{X}-\mathbf{M}_2)^t\Sigma_2^{-1}(\mathbf{X}-\mathbf{M}_2) - \frac{1}{2}\ln|\Sigma_2| \end{aligned}$$

And ∇h will be given by

$$\nabla h = \Sigma_1^{-1}(\mathbf{X}-\mathbf{M}_1) - \Sigma_2^{-1}(\mathbf{X}-\mathbf{M}_2) = (\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{X} + (\Sigma_2^{-1}\mathbf{M}_2 - \Sigma_1^{-1}\mathbf{M}_1)$$

Then the vector normal to the decision boundary at $\mathbf{X}_0$ is given by

$$\mathbf{N} = \nabla h(\mathbf{X})|_{\mathbf{X}=\mathbf{X}_0} = (\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{X}_0 + (\Sigma_2^{-1}\mathbf{M}_1 - \Sigma_1^{-1}\mathbf{M}_2) \qquad \text{(A.1)}$$

The following theorem gives the point where the straight line connecting two points $\mathbf{P}_1$ and $\mathbf{P}_2$ meets the decision boundary. Theorems A.1-2 can be employed to implement the proposed procedure to calculate a decision boundary feature matrix for parametric classifiers.

Theorem A.2 If $\mathbf{P}_1$ and $\mathbf{P}_2$ are on different sides of a decision boundary $h(\mathbf{X}) = t$ assuming that a Gaussian ML classifier is used, the point $\mathbf{X}_0$ where the line connecting $\mathbf{P}_1$ and $\mathbf{P}_2$ passes through the decision boundary is given by

$$\mathbf{X}_0 = u\mathbf{V} + \mathbf{V}_0 \tag{A.2}$$

where
$$\mathbf{V}_0 = \mathbf{P}_1$$
$$\mathbf{V} = \mathbf{P}_2 - \mathbf{P}_1$$
$$u = \frac{t - c'}{b} \text{ if } a = 0,$$

$$u = \frac{-b \pm \sqrt{b^2 - 4a(c' - t)}}{2a} \text{ and } 0 \le u \le 1 \text{ if } a \ne 0,$$
$$a = \frac{1}{2}\mathbf{V}^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V},$$
$$b = \mathbf{V}_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V} - (\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})\mathbf{V},$$
$$c' = \frac{1}{2}\mathbf{V}_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V}_0 - (\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})\mathbf{V}_0 + c,$$
$$c = \frac{1}{2}(\mathbf{M}^t\Sigma_1^{-1}\mathbf{M}_1 - \mathbf{M}_2^t\Sigma_2^{-1}\mathbf{M}_2) + \frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_2|}$$

Proof: $h(\mathbf{X})$ is given by

$$\begin{aligned}
h(\mathbf{X}) &= -\ln\frac{P(\mathbf{X}|\omega_1)}{P(\mathbf{X}|\omega_2)} = -\ln P(\mathbf{X}|\omega_1) + \ln P(\mathbf{X}|\omega_2) \\
&= \frac{1}{2}(\mathbf{X} - \mathbf{M}_1)^t\Sigma_1^{-1}(\mathbf{X} - \mathbf{M}_1) + \frac{1}{2}\ln|\Sigma_1| - \frac{1}{2}(\mathbf{X} - \mathbf{M}_2)^t\Sigma_2^{-1}(\mathbf{X} - \mathbf{M}_2) - \frac{1}{2}\ln|\Sigma_2| \\
&= \frac{1}{2}\mathbf{X}^t\Sigma_1^{-1}\mathbf{X} - \mathbf{M}_1^t\Sigma_1^{-1}\mathbf{X} + \frac{1}{2}\mathbf{M}^t\Sigma_1^{-1}\mathbf{M}_1 + \frac{1}{2}\ln|\Sigma_1| \\
&\quad -\frac{1}{2}\mathbf{X}^t\Sigma_2^{-1}\mathbf{X} + \mathbf{M}_2^t\Sigma_2^{-1}\mathbf{X} - \frac{1}{2}\mathbf{M}_2^t\Sigma_2^{-1}\mathbf{M}_2 - \frac{1}{2}\ln|\Sigma_2| \\
&= \frac{1}{2}\mathbf{X}^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{X} - (\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})\mathbf{X} \\
&\quad + \frac{1}{2}(\mathbf{M}^t\Sigma_1^{-1}\mathbf{M}_1 - \mathbf{M}_2^t\Sigma_2^{-1}\mathbf{M}_2) + \frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_2|}
\end{aligned}$$

$$= \frac{1}{2} \mathbf{X}^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{X} - (\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})\mathbf{X} + c$$

$$\text{where } c = \frac{1}{2} (\mathbf{M}^t\Sigma_1^{-1}\mathbf{M}_1 - \mathbf{M}_2^t\Sigma_2^{-1}\mathbf{M}_2) + \frac{1}{2} \ln\frac{|\Sigma_1|}{|\Sigma_2|}$$

Let $\mathbf{X}_0$=u$\mathbf{V}$+ $\mathbf{V}_0$ where u is a scalar. Then h($\mathbf{X}_0$)=h(u$\mathbf{V}$+ $\mathbf{V}_0$) is given by

$$\begin{aligned}
&h(u\mathbf{V}+ \mathbf{V}_0) \\
&= \frac{1}{2} (u\mathbf{V}+ \mathbf{V}_0)^t(\Sigma_1^{-1} - \Sigma_2^{-1})(u\mathbf{V}+ \mathbf{V}_0) - (\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})(u\mathbf{V}+ \mathbf{V}_0) + c \\
&= \{\frac{1}{2} \mathbf{V}^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V}\}\bullet u^2 + \{\mathbf{V}_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V}\}\bullet u + \frac{1}{2} \mathbf{V}_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V}_0 \\
&\quad - \{(\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})\mathbf{V}\}\bullet u - (\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})\mathbf{V}_0 + c \\
&= \{\frac{1}{2} \mathbf{V}^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V}\}\bullet u^2 + \{\mathbf{V}_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V} - (\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})\mathbf{V}\}\bullet u + c' \\
&\text{where } c' = \frac{1}{2} \mathbf{V}_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V}_0 - (\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})\mathbf{V}_0 + c
\end{aligned}$$

Let $a = \frac{1}{2} \mathbf{V}^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V}$, $b = \mathbf{V}_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{V} - (\mathbf{M}_1^t\Sigma_1^{-1} - \mathbf{M}_2^t\Sigma_2^{-1})\mathbf{V}$. Then

$$h(u\mathbf{V}+ \mathbf{V}_0) = a\bullet u^2 + b\bullet u + c' = t$$

Let $f(u)=h(u\mathbf{V}+ \mathbf{V}_0) - t = a\bullet u^2 + b\bullet u + c' - t$

Then the solutions to f(u)=0 are given by

$$\text{If } a = 0,\ u = \frac{t - c'}{b}$$

$$\text{If } a \neq 0,\ u = \frac{-b \pm \sqrt{b^2 - 4a(c' - t)}}{2a}$$

Therefore the point which is on the straight line u$\mathbf{V}$+ $\mathbf{V}_0$ and on the decision boundary h(X)=t can be given by

$$\mathbf{X}_0 = u\mathbf{V}+ \mathbf{V}_0$$

where $u = \frac{t - c'}{b}$ if a = 0,

$$u = \frac{-b \pm \sqrt{b^2 - 4a(c' - t)}}{2a} \text{ and } 0 \le u \le 1 \text{ if } a \neq 0,$$

$$a = \frac{1}{2} V^t(\Sigma_1^{-1} - \Sigma_2^{-1})V,$$

$$b = V_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})V - (M_1^t\Sigma_1^{-1} - M_2^t\Sigma_2^{-1})V,$$

$$c' = \frac{1}{2} V_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})V_0 - (M_1^t\Sigma_1^{-1} - M_2^t\Sigma_2^{-1})V_0 + c,$$

$$c = \frac{1}{2} (M^t\Sigma_1^{-1}M_1 - M_2^t\Sigma_2^{-1}M_2) + \frac{1}{2} \ln\frac{|\Sigma_1|}{|\Sigma_2|}$$

Equation (A.2) can be used to calculate the point on the decision boundary from two samples classified differently and equation (A.1) can be used to calculate a normal vector to the decision boundary.

Example A.1 Assuming that a Gaussian ML classifier is used, the mean vectors and covariance matrices of two classes are given as follows:

$$M_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$P(\omega_1) = P(\omega_1) = 0.5$$

The inverses and determinants of Σ_1 and Σ_2 are given by

$$\Sigma_1^{-1} = \frac{4}{3}\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}, \det(\Sigma_1) = 0.75$$

$$\Sigma_2^{-1} = \frac{4}{3}\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}, \det(\Sigma_2) = 0.75$$

Let $\mathbf{P}_1$=(1,0) and $\mathbf{P}_2$=(1,2) be points on the different sides of the decision boundary. Then the equation of a straight line connecting the two points is given as follows:

$$uV + V_0 \text{ where } V = P_2 - P_1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, V_0 = P_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Then the point(s) where the decision boundary and the straight line meets are given by

$$\mathbf{X} = uV + V_0$$

where $u = \frac{t - c'}{b} = -\frac{c'}{b}$

$b = V_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})V - (M_1^t\Sigma_1^{-1} - M_2^t\Sigma_2^{-1})V,$

$c' = \frac{1}{2} V_0^t(\Sigma_1^{-1} - \Sigma_2^{-1})V_0 - (M_1^t\Sigma_1^{-1} - M_2^t\Sigma_2^{-1})V_0 + c,$

$c = \frac{1}{2}(M_1^t\Sigma_1^{-1}M_1 - M_2^t\Sigma_2^{-1}M_2) + \frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_2|}$

Therefore,

$$c = \frac{1}{2}(M_1^t\Sigma_1^{-1}M_1 - M_2^t\Sigma_2^{-1}M_2)$$
$$= \frac{2}{3}\left(\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}1 & -0.5\\ -0.5 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix} - \begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}1 & -0.5\\ -0.5 & 1\end{bmatrix}\begin{bmatrix}-1\\ 1\end{bmatrix}\right)$$
$$= 0$$

$$c' = -(M_1^t\Sigma_1^{-1} - M_2^t\Sigma_2^{-1})V_0$$
$$= -\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}1 & -0.5\\ -0.5 & 1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} + \begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}1 & -0.5\\ -0.5 & 1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix}$$
$$= -1.5 - 1.5 = -3$$

$$b = -(M_1^t\Sigma_1^{-1} - M_2^t\Sigma_2^{-1})V$$
$$= -\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}1 & -0.5\\ -0.5 & 1\end{bmatrix}\begin{bmatrix}0\\ 2\end{bmatrix} + \begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}1 & -0.5\\ -0.5 & 1\end{bmatrix}\begin{bmatrix}0\\ 2\end{bmatrix}$$
$$= 3 + 3 = 6$$

$u = 0.5$

$$\mathbf{X} = 0.5V + V_0 = 0.5\begin{bmatrix}0\\ 2\end{bmatrix} + \begin{bmatrix}1\\ 0\end{bmatrix} = \begin{bmatrix}1\\ 1\end{bmatrix}$$

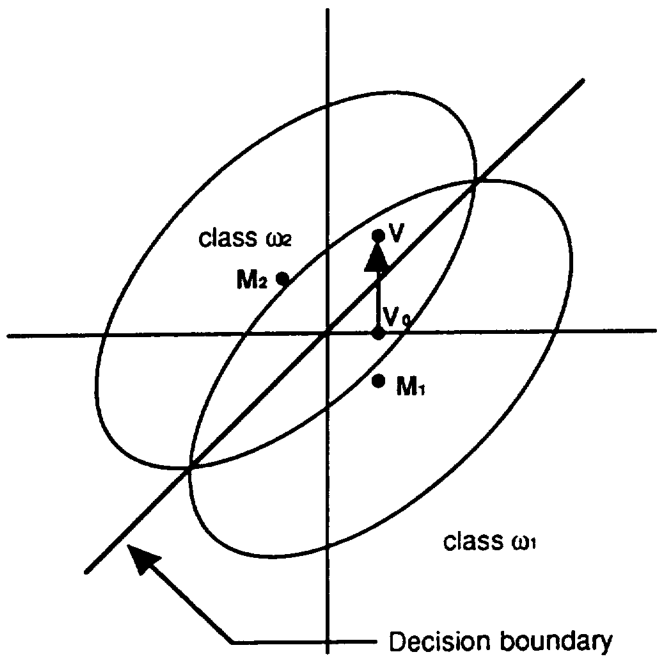

Figure A.1 Solution of h(uV+ V_0)=t where u needs to be found.

Appendix B

This appendix contains source code listings for the algorithms involved. due to its length it has not been included in all copies of this report. It is available upon request to the authors.