4-1-1998

CLASSIFICATION OF HIGH DIMENSIONAL DATA WITH LIMITED TRAINING SAMPLES

Saldju Tadjudin
Purdue University School of Electrical and Computer Engineering

David Landgrebe
Purdue University School of Electrical and Computer Engineering

CLASSIFICATION OF HIGH DIMENSIONAL DATA WITH LIMITED TRAINING SAMPLES

SALDJU TADJUDIN
DAVID LANDGREBE

TR-ECE 98-8
APRIL 1998

SCHOOL OF ELECTRICAL
AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

CLASSIFICATION OF HIGH DIMENSIONAL DATA WITH LIMITED TRAINING SAMPLES

Saldju Tadjudin
David Landgrebe

May 1998

SCHOOL OF ELECTRICAL
AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

TABLE OF CONTENTS

ABSTRACT

An important problem in pattern recognition is the effect of limited training samples on classification performance. When the ratio of the number of training samples to the dimensionality is small, parameter estimates become highly variable, causing the deterioration of classification performance. This problem has become more prevalent in remote sensing with the emergence of a new generation of sensors. While the new sensor technology provides higher spectral and spatial resolution, enabling a greater number of spectrally separable classes to be identified, the needed labeled samples for designing the classifier remain difficult and expensive to acquire. In this thesis, several issues concerning the classification of high dimensional data with limited training samples are addressed. First of all, better parameter estimates can be obtained using a large number of unlabeled samples in addition to training samples under the mixture model. However, the estimation method is sensitive to the presence of statistical outliers. In remote sensing data, classes with few samples are difficult to identify and may constitute statistical outliers. Therefore, a robust parameter estimation method for the mixture model is introduced. Motivated by the fact that covariance estimates become highly variable with limited training samples, a covariance estimator is developed using a Bayesian formulation. The proposed covariance estimator is advantageous when the training set size varies and reflects the prior of each class. Finally, a binary tree design is proposed to deal with the problem of varying training sample size. The proposed binary tree can function as both a classifier and a feature extraction method. The benefits and limitations of the proposed methods are discussed and demonstrated with experiments.

Work leading to the report was supported in part by NASA Grant NAG5-3975. This. support is gratefully acknowledged.

CHAPTER 1: INTRODUCTION

1.1 Background

Remote sensing technology involves the measurement and analysis of the electromagnetic radiation reflected from the earth's surface by a passive or an active source. The radiation responses in various wavelengths indicate the types or properties of the materials on the surface being measured and collectively form a multispectral image. Early on, multispectral scanners were developed which measured radiation in 3 to 12 spectral bands. Current sensors can gather data in hundreds of spectral bands and generate hyperspectral data. For example, the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) collects data in 224 spectral bands covering 0.4-2.5 μm wavelength region with 20 m spatial resolution. By representing the spectrum of a pixel in a multispectral image as a random process [1] statistical pattern recognition methods have been successfully applied to process multispectral data. Figure 1.1 illustrates the representation of a pixel as multivariate data.

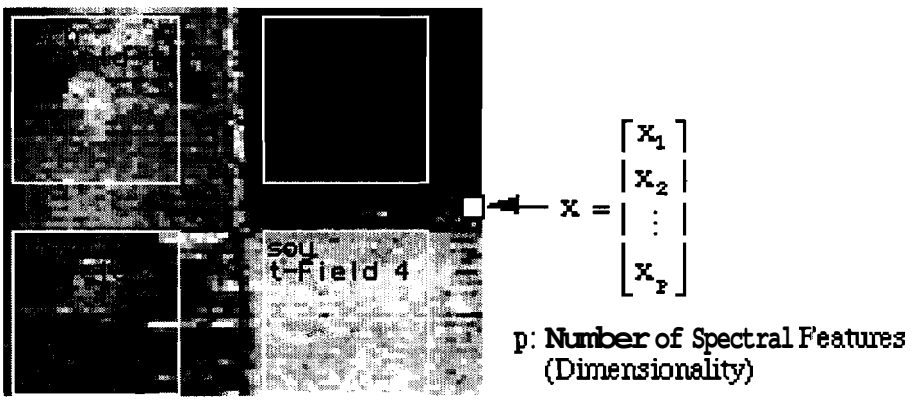


Figure 1.1 A Multispectral Image

The process of designing a classifier using training samples from the classes of interest is referred to as supervised classification. A typical supervised classification system for multispectral data consists of several stages as shown in Figure **1.2.**

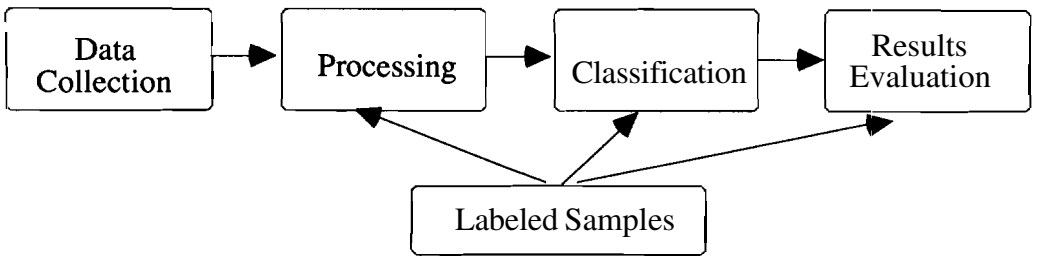


Figure **1.2** Multispectral Data Processing System

Before classifying the data, some form of processing is usually performed on the data. The purpose of the processing stage is to obtain a better representation of the data based on the available labeled samples in preparation for classification. If the probability density functions (pdfs) of the classes are assumed known, a better representation usually means a good set of parameter estimates for the pdfs. The common approach in remote sensing is to assume normally distributed classes and estimate the mean vectors and covariances matrices using the training samples. The processing stage may then involve covariance estimation, statistics enhancement using an expectation maximization (EM) algorithm and feature extraction.

The types of classifier can be broadly divided into two categories: pixel-based and spectral-spatial classifiers. A pixel-based classifier assigns each pixel to one of the classes by applying a decision rule. In other words, each pixel is classified individually based on its spectral measurements alone. Usually, the decision rule can be written in terms of the pdfs of the classes or their parameters. In spectral-spatial classifiers, it is assumed that the classes of neighboring pixels are not independent. Therefore, the decision can either be formed on a group of adjacent pixels or can take into account the classes of neighboring pixels.

After the classifier is designed, it is usually tested by measuring the error probability, which can be obtained from classifying the labeled samples. In practical situiitions, the number of these labeled samples is limited so one must decide how to

divide them to both design and test the classifier. An unbiased estimator is provided by using a set of samples for design and the other set of samples for testing the classifier. This approach, called the holdout method, is adopted for this thesis.

1.2 Objective of Research

The increase in spectral resolution brought about by the new sensor technology has offered new possibilities and challenges. It is the goal of this thesis to investigate the problems presented by the new sensors.

The availability of a large number of spectral bands should allow more detailed classes to be identified with higher accuracy than previously possible. However, for remote sensing applications, the needed number of labeled samples for designing and testing the classifier remains expensive and difficult to acquire. For example, the ground truth information may be gathered by visual inspection of the actual site or by matching the spectral responses of the samples against the responses of known samples. As a result, the class statistics have to be estimated by the limited training sample set. When the ratio of the number of training samples to the number of features is small, the parameter estimates become highly variable causing the classification performance to deteriorate. Typically, the performance of the classifier improves up to a certain point as additional features are added, and then deteriorates. This is referred to as the Hughes phenomenon [2] (See Figure 1.3). The number of training samples required for different classifiers to obtain reasonable parameter estimates has been studied in [3]. Thus, the goal of this research is essentially to circumvent the Hughes phenomenon caused by limited training set size.

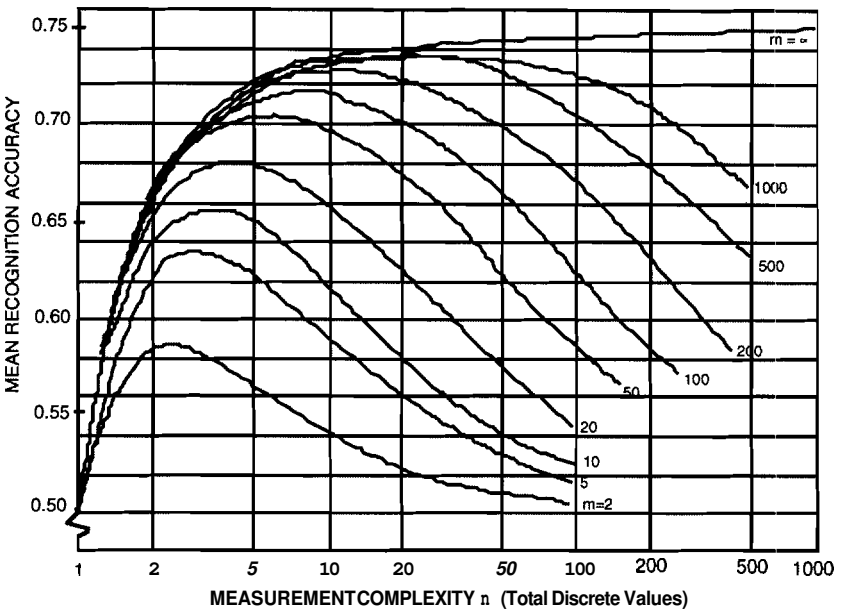


Figure 1.3 The Hughes Phenomenon [2]

1.3 Organization of this Report

In Chapter 2, the problem of limited training set size is addressed by including unlabeled samples for parameter estimation under the mixture model. It is common to view remote sensing data in terms of a mixture model fitted with normally distributed components (spectral classes). Then the parameters of the mixture moclel are estimated, for example, using the expectation maximization (EM) algorithm. For the EM algorithm to perform well, the classes must be exhaustive. In other words, the existence of statistical outliers may degrade the performance. As a direct consequence of increased spectral and spatial resolution, hyperspecral data consists of more spectral classes with varying sizes. Some of these classes can be small and tedious to identify and may constitute outlying pixels which are not consistent with the statistics of the other classes. Therefore, a robust estimation method for estimating the mean vectors and covariance matrices under the mixture model is presented in Chapter 2. The proposed method gives reduced weights to pixels which are considered as statistical outliers and thereby limiting their influence in estimating parameters.

In Chapter 3, the issue of small training sample size is addressed as a parameter estimation problem, in particular, covariance estimation. When the training sample size

is small compared to the dimensionality, the sample estimates of the parameters becomes highly variable. The problem of limited training samples is especially severe for covariance matrices since sample covariance estimates become singular as the number of training samples is less than dimensionality. In such circumstances, several studies have found that a linear classifier often performs better than a quadratic classifier. However, the choice between either a linear or quadratic classifier is quite restrictive. A covariance estimator is therefore proposed which can be viewed as an intermediate approach between linear and quadratic classifiers. The proposed estimator is derived using a Bayesian formulation, which is desirable when the classes have varying sizes and the training sample size is proportional to the class sample size.

In Chapter 4, the problem posed by limited training samples and numerous classes is addressed by introducing a binary tree algorithm for classification and feature extraction. In a single stage classifier, the same number of features have to be applied for all classes. In a complex image with classes of varying sizes, those classes with very limited training samples may impose a serious constraint on the total number of features to be used for classification. Motivated by the need for a more flexible classification procedure in which different number of features can be applied to discriminate different classes, a binary tree design is proposed. In addition to performing as a classifier, the proposed binary tree design can also function as a feature extraction method to generate features for a single-stage classifier. The detail of implementation and experimental results are presented in Chapter 4.

Finally, a general conclusion and directions for future research are presented in Chapter 5.

CHAPTER 2: ROBUST PARAMETER ESTIMATION FOR MIXTURE MODEL

2.1 Introduction

In a mixture model, data are assumed to consist of two or rnore component distributions mixed in varying proportions. For remote sensing applications, it is a common practice to consider several "spectral subclasses" within each "information class" or ground cover type. Each of such spectral subclasses is assumed to be normally distributed and classification is performed with respect to the spectral subclasses. Under this model, remote sensing data can be considered as a mixture model fitted with normally distributed components.

To estimate the model parameters in a mixture, a common approach is to apply the expectation maximization (EM) algorithm to obtain the maximum likelihood (ML) estimates. For the EM algorithm to converge to the global solution, several conditions have to be met. First of all, the initial estimates must be reasonably good. Usually the training samples provide a good starting point for the iteration. Moreover, the defined classes must be exhaustive. This means that all samples are accounted for by the component distributions in the mixture. Unfortunately, for the analysis of remote sensing data, to arrive at a set of exhaustive classes is an iterative process by trial and error, and usually depends on the expertise of the user. In addition, there might be some scattered background pixels which are difficult or tedious to identify. These pixels form the so-called "information noise" whose spectral responses may not be consistent with the majority of samples. Such statistical outliers are usually eliminated using a chi-square threshold before applying the EM algorithm. This method can be viewed as a hard decision. However, a suitable threshold value is difficult to select and is usually arbitrary. Consequently, "useful" pixels might be rejected as outliers.

In this chapter, a robust method is proposed to estimate the mean vector and covariance matrix for classifying multispectral data under the mixture model. This approach assigns full weight to the training samples, but automatically gives reduced weight to unlabeled samples. Therefore, it avoids the risk of rejecting useful pixels while still limiting the influence of outliers in obtaining the ML estimates of the parameters. The experimental results show that the proposed robust method prevents performance deterioration due to outliers in the image as compared with the EM approach.

2.2 Expectation Maximization Algorithm for Mixture Density Estimation

2.2.1 Previous work

There has been extensive research on the problem of parameter estimation for a normal mixture density over the past few decades. An excellent review can be found in [4]. Karl Pearson [5] first employed the method of moments to decompose a finite mixture of distributions in the case of a mixture of two univariate distributions with different variances. The likelihood estimation of parameters in a mixture model was first proposed by Rao [6] who used Fisher's method of scoring for a mixture of two univariate distributions with equal variances. Later, it was shown that the method of moments is inferior to likelihood estimation of a mixture model [7]. The solution for the likelihood approach was then presented and formalized in an iterative form as the expectation maximization (EM) algorithm by Dempster, Laird and Rubin [8]. They :proposed the EM algorithm as a solution to the maximum likelihood (ML) problem involving missing data, of which the mixture identification problem is an example. In the review article [9], the EM equations for obtaining the ML estimates of the parameters and their properties were studied in detail. The convergence properties were investigated in [10].

In [11], the EM algorithm has been studied and applied to remote sensing data. It was shown that by assuming a mixture model and using both training samples and unlabeled samples in obtaining the estimates, the classification performance can be improved. Also, the Hughes phenomenon can be delayed to a higher dimensionality and hence more features can be used to obtain better performance. In addition, the parameter estimates represent the true class distributions more accurately. However, the unrepresented pixel classes have been dealt with by rejection using a chi-square threshold. In the next section, the EM algorithm is reviewed and discussed.

2.2.2 Expectation Maximization Algorithm

The Expectation Maximization (EM) algorithm is an iterative method for numerically approximating the maximum likelihood (ML) estimates of the parameters in a mixture model. Alternatively, it can be viewed as an estimation problem involving incomplete data in which each unlabeled observation on the mixture is regarded as missing a label of its origin [12].

Under the mixture model, the distribution of the data $x \in \Re^p$ is given as:

$$f(x;\Phi) = \sum_{i=1}^{L} \alpha_i f_i(x;\phi_i)$$

where a,,...,α_L are the prior probabilities or the mixing proportions, f_i is the component density parametrized by ϕ_i and L is the total number of components. The mixture density f is then parametrized by $\Phi = (a,...,a_L, \phi_1,...,\phi_L)$.

Under the incomplete data formulation, each unlabeled sample x is considered as the :labeled sample y with its class origin missing. Therefore, we can denote $y = (x,i)$ where $i = 1 \cdots L$ indicates the sample origin. Let $g(x|\Phi)$ be the probability density function (pdf) of the incomplete data $x = (x,,..,x_n)$ and $f(y|\Phi)$ be the pdf of the completely labeled data $y = (y_1,...,y_n)$. The maximum likelihood estimation then involves the maximization of the log likelihood of the incomplete data $L(\Phi) = log\, g(x|\Phi)$. The estimation is complicated by the fact that the sample origin is missing. Hence, the EM algorithm uses the relationship between $f(y|\Phi)$ and $g(x|\Phi)$ to maximize the incomplete data log-likelihood $L(\Phi) = log\, g(x|\Phi)$. Using an iterative approach, the EM algorithm obtains the maximum likelihood estimates by starting with an initial estimate Φ^0 and repeating the following two steps at each iteration:

E-Step) Determine $Q(\Phi|\Phi^c) = E\{log\, f(y|\Phi)|x,\Phi^c\}$.

M-Step) Choose $\Phi^+ = arg\, max\, Q(\Phi|\Phi^c)$.

The next and current values of the parameters are denoted by the superscripts "+" and "c" respectively. The algorithm begins with an initial estimate and it: has been shown that under some relatively general conditions the iteration converges to ML estimates, at least locally. Since the convergence is only guaranteed to a local. maximum, the

algorithm usually has to be repeated from various initial points. However, the training samples, if available, can provide good initial estimates.

Assume that $y = (y_1, \ldots, y_{m_i})$ are the m_i training samples from class i. Also, there are L Gaussian classes and a total of n unlabeled samples denoted by $x = (x_1, \ldots, x_n)$. The parameter set Φ then contains all the prior probabilities, mean vectors and covariance matrices. The EM algorithm can then be expressed as the following iterative equations [9]:

E-Step:

$$\tau_{ij}^c = \tau_i(x_j|\phi_i^c) = \alpha_i^c f_i(x_j|\phi_i^c) \Big/ \sum_{t=1}^{L} \alpha_t^c f_t(x_j|\phi_t^c) \tag{2.1}$$

where τ_{ij}^c is the posterior probability that x_j belongs to class i

M-Step:

$$\alpha_i^+ = \sum_{j=1}^{n} \tau_{ij}^c / n \tag{2.2}$$

$$\mu_i^+ = \frac{\sum_{j=1}^{m_i} y_{ij} + \sum_{j=1}^{n} \tau_{ij}^c x_j}{m_i + \sum_{j=1}^{n} \tau_{ij}^c} \tag{2.3}$$

$$\Sigma_i^+ = \frac{\sum_{j=1}^{m_i} (y_{ij} - \mu_i^+)(y_{ij} - \mu_i^+)^T + \sum_{j=1}^{n} \tau_{ij}^c (x_j - \mu_i^+)(x_j - \mu_i^+)^T}{m_i + \sum_{j=1}^{n} \tau_{ij}^c} \tag{2.4}$$

There are several factors affecting the convergence of the EM algorithm to the maximum likelihood estimates. First of all, the selection of training samples as initial estimates can affect the convergence to a great extent. In this work, the training set is assumed to provide a good initial estimate. Another factor that decides the performance of the EM algorithm is the presence of statistical outliers. Assume that the number of components have been decided and given by the training set. Statistical outliers are defined as those observations which have great discrepancy from the distributions of the mixture components. As indicated by Eq. (2.1) through Eq. (2.4), the EM algorithm

assigns each observation to one of the components with the sample's posterior probability as its weight. Even though an outlying sample is inconsistent with distributions of all the defined components, it may still have a large posterior probability for one or more of the components. As a result, the iteration converges to erroneous solutions.

The problem of outliers is not uncommon for practical applications. In remote sensing, a scene usually contains pixels of unknown origin which form "information noise". For example, in an agricultural area, there could be pixels belonging to houses, trees or rural roads. The statistical distributions of these pixels may be significantly different from those of training classes and constitute statistical outliers. Unfortunately, these outlying pixels are usually scattered throughout the image and are small in number. Consequently, identifying these pixels could be a tedious task. A common approach to eliminate those pixels in the EM algorithm is to apply a chi-square threshold test [11]. In other words, pixels whose distances are greater than the threshold value are considered as outliers and are subsequently excluded from updating the estimates. The chi-square threshold T_α for a given probability α is defined as the squared distance between the sample $x \in \Re^p$ and the mean vector for class i based on the chi-square distribution as shown in the following:

$$\Pr\left\{x \middle| (x-m_i)^T \Sigma_i^{-1} (x-m_i) \le T_\alpha\right\} = \alpha .$$

The problem of outliers can be illustrated by the following simulation. The data set contains three classes and only Class 1 and Class 2 are represented. by the training samples in the mixture density. These two classes are generated with the normal densities $N(0,2)$ and $N(8,2)$ respectively. A total of 500 samples are generated for the two classes and 50 samples are selected as the training samples. A third class with a normal density $N(20,1)$ is generated to represent outliers. The number of samples for Class 3 are chosen to be 50. Figure 2.1 shows the densities for these classes. The experiments are repeated with the sample estimates, the estimates with EM algorithm after 10 iterations without thresholding and with thresholding. The chi-square threshold is chosen to be $T_\alpha = 3.84$ and $\alpha = 95\%$ for one degree of freedom. The experiment is repeated 50 times and the mean accuracy and standard deviations are recorded. The estimated densities are illustrated in Figure 2.2, which demonstrates that the presence of outliers can have an undesirable effect on the EM algorithm. The classification results are shown in Table 2.1 and Figure 2.3. The standard deviations are indicated in

parenthesis next to the mean accuracy. The results show that the classification performance deteriorates when the EM algorithm is applied in the presence of outliers.

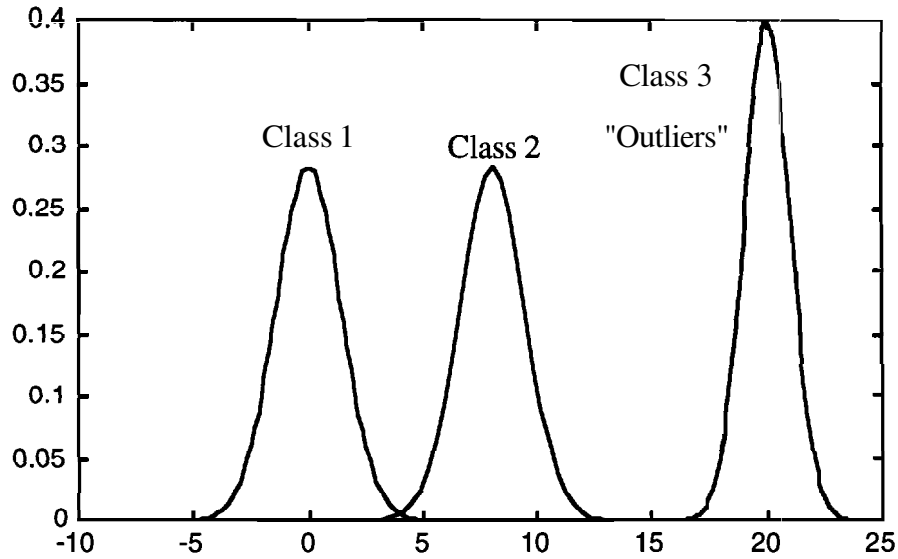


Figure 2.1 Probability Densities for Simulation Data

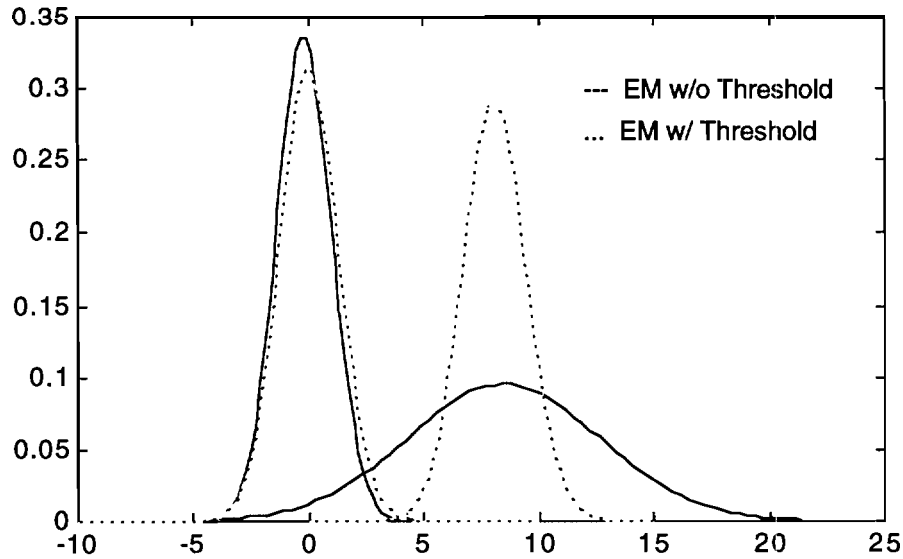


Figure 2.2 Estimated Probability Densities after Performing EM Algorithm

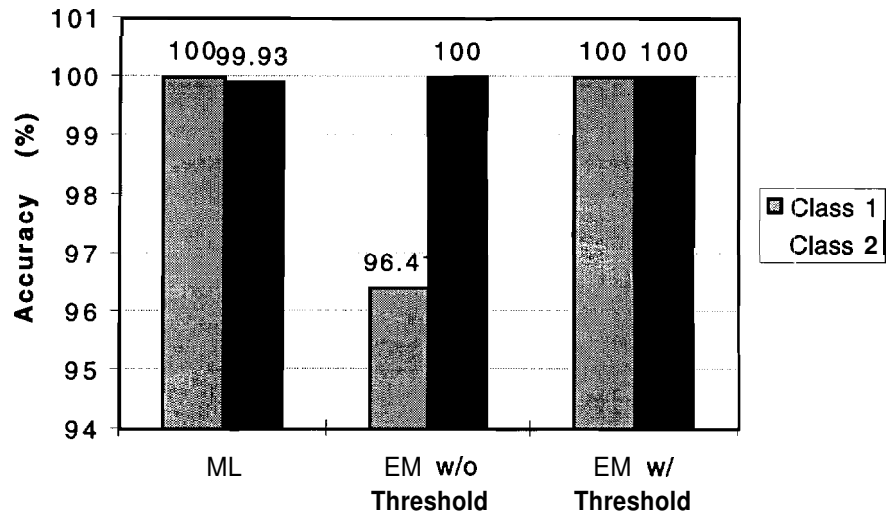


Figure 2.3 Mean Accuracy for the Simulation with Class 3 as Outliers

Table 2.1
Experiment Results for the Simulation with Class 3 as Outliers

	Class 1	Class 2
True Mean	0	8
True Variance	2	2
Sample Mean	-0.08 (.17)	8.09 (.19)
Sample Variance	1.82 (.34)	1.86 (.34)
Accuracy (%)	100 (.00)	99.93 (.15)
EM: w/o Threshold		
Estimated Mean	-0.21 (.01)	8.46 (.03)
Estimated Variance	1.48 (.03)	17.50 (.17)
Accuracy (%)	96.41 (.69)	100 (.00)
EM: with Threshold		
Estimated Mean	0.00 (.01)	8.08 (.00)
Estimated Variance	1.62 (.03)	1.89 (.00)
Accuracy (%)	100 (.00)	100 (.00)

In the above simulation, the samples from Class 3 are not represented in the mixture model. Since those samples are closer in statistical distance to Class 2., they have high posterior probability with respect to Class 2. Therefore, the estimated density for Class 2 is degraded as shown in Figure 2.2. The figure also shows that the estimated densities of Class 1 and Class 2 overlap such that some samples from Class 1 are rnisclassified as Class 2, causing the decrease in the accuracy for Class 1. By applying the threshold,

marly of the Class 3 samples are excluded from the EM algorithm. Consequently, better density estimates are obtained.

The thresholding approach can be regarded as performing a hard decision to eliminate outlying samples before initiating the EM algorithm. Unfortunately, the choice of threshold is arbitrary and useful pixels could be rejected at the outset. An alternative would be to assign a different weight to each pixel and use all available unlabeled pixels for updating the statistics. This method can be regarded as applying a soft decision. In the next section, the robust EM equations will be discussed and modified to process remote sensing data.

2.3 Robust Estimation

2.3.1 Previous work

The robust estimation of model parameters was first developed as Huber [13] proposed a theory of robust estimation of a location parameter using M-estimates in a non-mixture context. It was later extended to the multivariate case by taking an elliptically symmetric density and then associating it with a contaminated normal density [14]. Campbell [15] derived the M-estimates for the mixture density and obtained an EM-like algorithm but with a weight function assigned to each pixel as a measure of typicality. The outlier problem in remote sensing has been addressed in [16]. The author proposed a modified M-estimation of the parameters to deal with the situation when the training samples of a certain information class contain samples of other classes. This is typical for a mixture model. The modified M-estimates were shown to be robust with respect to the contamination in the training samples as compared to the least-square estimates. However, the use of unlabeled samples in updating statistics was not addressed. The next section will describe the method of robust EM algorithm following the discussion in [15], and adapting the approach for remote sensing data.

2.3.2 Robust EM Algorithm

The expectation maximization (EM) algorithm first estimates the posterior probabilities of each sample belonging to each of the component distributions, and then computes the parameter estimates using these posterior probabilities as weights. With

this approach, each sample is assumed to come from one of the component distributions, even though it may greatly differ from all components. The robust estimation attempts to circumvent this problem by including the typicality of a sample with respect to the component densities in updating the estimates in the EM algorithm.

To incorporate a measure of typicality in the parameter estimation of the mixture density, the component densities $f_i(x|\phi_i)$ are assumed to be a member of the family of p-dimensional elliptically symmetric densities with parameters μ_i and Σ_i:

$$|\Sigma_i|^{-1/2} f_S\{\delta_i(x;\mu_i,\Sigma_i)\}$$

where $\delta_i^2 = (x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)$. Typically, $f_S(\delta_i)$ is assumed to be the exponential of some symmetric function $\rho(\delta_i)$:

$$f_S(\delta_i) = exp\{-\rho(\delta_i)\}.$$

Then, the likelihood parameter estimation for these component densities can be obtained by applying the expectation and maximization steps.

Expectation Step

Assume that unlabeled samples $\{x_1, x_2, \cdots, x_n\}$ are available, $Q(\Phi|\Phi^c)$ can then be written as the summation of two terms:

$$\begin{aligned} Q(\Phi|\Phi^c) &= E\{\log f(y|\Phi)|x,\Phi^c\} \\ &= \sum_{i=1}^{L}\sum_{j=1}^{n} \tau_{ij}^c \log \alpha_i + \sum_{i=1}^{L}\sum_{j=1}^{n} \tau_{ij}^c \log f_i(x_j|\phi_i) \\ &= \sum_{i=1}^{L}\sum_{j=1}^{n} \tau_{ij}^c \log \alpha_i + \sum_{i=1}^{L}\sum_{j=1}^{n} \tau_{ij}^c \log\{|\Sigma_i|^{-1/2} f_S\{\delta_i(x_j;\mu_i,\Sigma_i)\}\} \end{aligned} \quad (2.5)$$

where $\tau_{ij}^c = \alpha_i^c |\Sigma_i^c|^{-1/2} f_S(\delta_{ij}^c) \Big/ \sum_{t=1}^{L} \alpha_t^c |\Sigma_t^c|^{-1/2} f_S(\delta_{ij}^c)$ is the posterior probability and δ_{ij}^2 is the squared distance $\delta_{ij}^2 = (x, -\mu_i)^T \Sigma_i^{-1}(x_j - \mu_i)$.

Maximization Step

The maximization of Eq. 2.5 is carried out by taking the derivatives with respect to the parameters a,, μ_i and Σ_i and setting these derivatives to zero. The optimization of α_i involves only the first term in Eq. 2.5, and is given by [9]

$$\alpha_i^+ = \sum_{j=1}^{n} \tau_{ij}^c / n$$

The iterative equations for μ_i and Σ_i are obtained by solving the following equations:

$$\frac{\partial}{\partial \mu_i} Q\left(\Phi \middle| \Phi^c\right) = 0$$

and

$$\frac{\partial}{\partial \Sigma_i} Q\left(\Phi \middle| \Phi^c\right) = 0.$$

The following equations can then be derived from Eq. 2.5:

$$\frac{\partial}{\partial \mu_i} \sum_{j=1}^{n} \tau_{ij}^c \log\left\{\left|\Sigma_i\right|^{-1/2} f_S\left\{\delta_i\left(x_j; \mu_i, \Sigma_i\right)\right\}\right\} = 0$$

$$\frac{\partial}{\partial \mu_i} \sum_{j=1}^{n} \tau_{ij}^c \left\{\left(-\frac{1}{2}\right)\log\left|\Sigma_i\right| - \rho\left(\delta_i\left(x_j; \mu_i, \Sigma_i\right)\right)\right\} = 0$$

Substituting μ_i^+ for μ_i, taking the derivative and simplifying, one obtains

$$\sum_{j=1}^{n} \tau_{ij}^c \frac{\psi\left(\delta_{ij}^c\right)}{\delta_{ij}^c}\left(x_j - \mu_i^+\right) = 0$$

where $\psi\left(\delta_{ij}^c\right) = \rho'\left(\delta_{ij}^c\right)$ is the first derivative of $\rho\left(\delta_{ij}^c\right)$. Rearranging and letting $w_{ij}^c := \psi\left(\delta_{ij}^c\right)/\delta_{ij}^c$, the maximum likelihood estimator for μ_i is expressed as follows:

$$\mu_i^+ = \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^c x_j \Big/ \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^c .$$

The term $w_{ij} = \psi(\delta_{ij})/\delta_{ij}$ reflects the contribution of sample x_j to the ith mean. Therefore, it is a weight function and provides a measure of typicality for the samples. Note that the value of the weight function is obtained using the parameter values from the previous iteration.

To obtain the iterative equation for the covariance matrix, the following equation is set up:

$$\frac{\partial}{\partial \Sigma_i} \sum_{j=1}^{n} \tau_{ij}^{c} \left\{ \left(-\frac{1}{2} \right) \log|\Sigma_i| - \rho\left(\delta_i(x_j; \mu_i, \Sigma_i)\right) \right\} = 0.$$

Using the matrix derivative formulas in [17], the following equations are derived:

$$\sum_{j=1}^{n} \tau_{ij}^{c} \left\{ -\Sigma_i^{+-1} + \frac{1}{2} diag\left(\Sigma_i^{+-1}\right) + \frac{\psi(\delta_{ij}^{c})}{\delta_{ij}^{c}} \Sigma_i^{+-1} (x_j - \mu_i^{+})(x_j - \mu_i^{+})^T \Sigma_i^{+-1} \cdots \right.$$

$$\left. \cdots - \frac{\psi(\delta_{ij}^{c})}{2\delta_{ij}^{c}} diag\left[\Sigma_i^{+-1} (x_j - \mu_i^{+})(x_j - \mu_i^{+})^T \Sigma_i^{+-1} \right] \right\} = 0$$

where $diag(A)$ is a diagonal matrix, keeping only the diagonal terms of the matrix A. Simplifying and multiplying the equation by Σ_i^{+} from left and right, the following equation is obtained:

$$\sum_{j=1}^{n} \tau_{ij}^{c} \left[-\Sigma_i^{+} + w_{ij}^{+} (x_j - \mu_i^{+})(x_j - \mu_i^{+})^T \right] = 0.$$

The value of the weight function is obtained by using the parameters $(\mu_i^{+}, \Sigma_i^{c})$. Hence, the iterative equation for Σ_i can be written as:

$$\Sigma_i^{+} = \sum_{j=1}^{n} \tau_{ij}^{c} w_{ij}^{+} (x_j - \mu_i^{+})(x_j - \mu_i^{+})^T \Big/ \sum_{j=1}^{n} \tau_{ij}^{c} \qquad (2.6)$$

It was noted that the estimator for Σ_i in Eq. *(2.6)* has two disadvantages [15]. First of all, the weights are not incorporated into the denominator. Secondly, using the weight function w_{ij} to estimate the covariance matrix fails to bound the influence of large atypical observations. Therefore, the estimator for Σ_i is modified and given as:

$$\Sigma_i^+ = \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^{+2} \left(x_j - \mu_i^+\right)\left(x_j - \mu_i^+\right)^T \Big/ \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^{+2}.$$

Assuming that both training and unlabeled samples are available, the iterative equations then become:

$$\alpha_i^+ = \sum_{j=1}^{n} \tau_{ij}^c / n$$

$$\mu_i^+ = \frac{\sum_{j=1}^{m_i} w_{ij}^c y_{ij} + \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^c x_j}{\sum_{j=1}^{m_i} w_{ij}^c + \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^c}$$

$$\Sigma_i^+ = \frac{\sum_{j=1}^{m_i} w_{ij}^{+2} \left(y_{ij} - \mu_i^+\right)\left(y_{ij} - \mu_i^+\right)^T + \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^{+2} \left(x_j - \mu_i^+\right)\left(x_j - \mu_i^+\right)^T}{\sum_{j=1}^{m_i} w_{ij}^{+2} + \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^{+2}}$$

The weight function has been chosen to be $\psi(s)/s$ where $s = \delta_{ij}$ and $\delta_{ij}^2 = \left(x_j - \mu_i\right)^T \Sigma_i^{-1} \left(x_j - \mu_i\right)$. A popular choice of $\psi(s)$ is the Huber's ψ-function which is defined by $\psi(s) = -\psi(-s)$ where for $s > 0$

$$\psi(s) = \begin{cases} s & 0 \le s \le k_1(p) \\ k_1(p) & s > k_1(p) \end{cases}$$

for an appropriate choice of the "tuning" constant $k_1(p)$, which is a function of the dimensionality p. This selection of $\psi(s)$ gives:

$$\rho(s)=\begin{cases}\frac{1}{2}s^2 & 0\le s\le k_1(p)\\ k_1(p)s-\frac{1}{2}k_1^2(p) & s>k_1(p)\end{cases}$$

The value of the tuning constant $k_1(p)$ is a function of dimensionality. It also depends on the amount of contamination in the data. Since the amount of contamination is usually not known, the value of $k_1(p)$ is chosen so that the estimators have reasonable performances over a range of situations. A variety of choices have been suggested in literature [15][18].

Like other parametric estimation applications, the performance of the classifier for remote sensing relies heavily on the proper choice of the training samples. Since the training samples are representative of the classes, it is desirable that they are given more emphasis in the updates of the estimates. Therefore, in the proposed approach, the training samples are assigned unit weight. To do so, the value of $k_1(p)$ is defined to be

$$k_1(p)=max\left(\hat{d}_{ij}\right)$$

where $\hat{d}_{ij}^2=\left(y_{ij}-\mu_i\right)^T\Sigma_i^{-1}\left(y_{ij}-\mu_i\right)$ and y_{ij} is the training sample j from class i. In other words, the tuning constant is selected such that the training samples are given unit weight and the weights for the unlabeled samples are inversely proportional to the square root of their squared distance to the class mean. To eliminate further the extreme outliers, another tuning constant can be applied which allocates zero weights to those samples. The chi-square threshold is recommended for the second tuning constant $k_2(p)$. In summary, the proposed weight function is defined as the following:

$$\psi(s)=\begin{cases}s & 0\le s\le k_1(p)\\ k_1(p) & k_1(p)\le s\le k_2(p).\\ 0 & s>k_2(p)\end{cases}$$

Alternatively, the weight assigned to each sample can be expressed as:

$$w_{ij}=\begin{cases}1 & d_{ij}\ I\ max\left(\hat{d}_{ij}\right)\\ max\left(\hat{d}_{ij}\right)/d_{ij} & max\left(\hat{d}_{ij}\right)<d_{ij}\le T_\alpha\\ 0 & d_{ij}>T_\alpha\end{cases}$$

where $d_{ij}^2 = (x_j - \mu_i)^T \Sigma_i^{-1} (x_j - \mu_i)$ and T_α is a user-defined chi-square threshold with a given probability a. The iterative equations for the mean and covariance estimates can then be expressed as:

$$\mu_i^+ = \frac{\sum_{j=1}^{m_i} y_{ij} + \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^c x_j}{m_i + \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^c}$$

$$\Sigma_i^+ = \frac{\sum_{j=1}^{m_i} (y_{ij} - \mu_i^+)(y_{ij} - \mu_i^+)^T + \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^{+2} (x_j - \mu_i^+)(x_j - \mu_i^+)^T}{m_i + \sum_{j=1}^{n} \tau_{ij}^c w_{ij}^{+2}}$$

In future reference, the proposed robust version of the EM algorithm is designated as REM. Also, the tuning constant $k_2(p)$ is not used in the following experiments.

2.4 Experimental Results

The following experiments are performed using a portion of an AVIRIS image taken over NW Indiana's Indian Pine test site in June 1992. The scene contains four information classes: corn-notill, soybean-notill, soybean-min and grass. By visual inspection of the image, the list of these ground cover types is assumed to be exhaustive. A total of 20 channels from the water absorption and noisy bands (104-108, 150-163, 220) are removed from the original 220 spectral channels, leaving 200 spectral features for the experiments. The test image and the ground truth map are shown in Figure 2.4. The number of labeled samples in each class is shown in Table 2.2.

Figure 2.4 Portion of AVIRIS Data and Ground Truth (Original in Color)

Table 2.2
Class Description for AVIRIS Data in Figure 2.4

Class Names	**No. of Labeled Samples**
Corn-notill	910
Soybean-notill	638
Soybean-min	1421
Grass	618

Experiment 2.1

The first experiment is intended to compare the expectation maximization (EM) and the proposed robust algorithm (REM) when no outliers are present in the data. The experiment is first conducted using simulation data. The data is obtained using the statistics computed from all the labeled samples of the four classes. A total of 2000 test samples per class is generated, 500 of which are used as the training samples. Since the training samples are selected at random, the experiment is repeated 5 tinnes and the mean and standard deviation of the classification accuracy are recorded. The numbers of spectral channels are set at 10, 20, 50, 67, 100 and 200. These channels are chosen by sampling the spectral range at fixed intervals. The algorithms are repeated for 10 iterations and the classification is performed using the Gaussian maximum likelihood classifier. The maximum likelihood (ML) method using only the training samples to estimate the parameters is denoted as ML in the following experiments. The results are shown in Table 2.3 and Figure 2.5. The standard deviation is shown in. parenthesis next to the mean accuracy.

Table 2.3
Classification Results for Experiment 2.1
with 500 Training Samples and 1000 Test Samples

Dimension	ML (%)	EM (%)	REM (%)
10	91.75 (.31)	91.25 (.08)	91.50 (.08)
20	96.29 (.31)	96.37 (.02)	96.37 (.02)
50	97.80 (.30)	98.54 (.002)	98.54 (.002)
67	98.61 (.20)	99.12 (.002)	99.12 (.002)
100	99.04 (.12)	99.66 (.001)	99.65 (.001)
200	99.93 (.12)	99.98 (.001)	99.98 (.001)

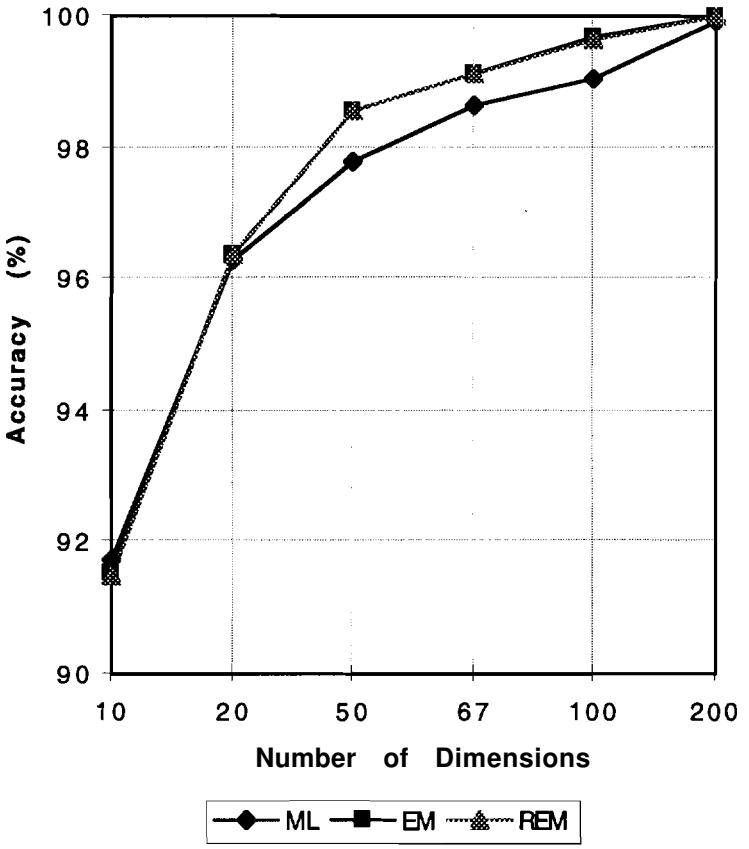


Figure 2.5 Mean Accuracy for Experiment 2.2
with 500 Training Samples and 1500 Test Samples

The results show that when no outliers are present in the data, the EM and REM algorithms have similar performance and both result in a better performance than the maximum likelihood classifier using the training samples alone. Since there are many design samples available, the best performance is obtained at 200 features.

Experiment 2.2

In this experiment, the simulation data from the Experiment **2.1** is used with the exception that only **250** training samples are selected for each class. The number of test samples is kept at **1500.** Again, no outliers are present in the data. The results are shown in Table **2.4** and Figure **2.6.**

Table **2.4**
Classification Results for Experiment **2.2**
with **250** Training Samples and **1500** Test Samples

Dimension	ML (%)	EM (%)	REM (%)
10	91.34 (.30)	91.74 (0.12)	91.74 (0.11)
20	**95.97** (.21)	**96.92 (0.11)**	**96.92 (0.10)**
50	**96.19** (.31)	**98.60 (0.09)**	**98.60 (0.09)**
67	**96.74** (.31)	**99.08 (0.08)**	**99.08 (0.08)**
100	**96.48** (.28)	**99.68 (0.04)**	**99.68 (0.03)**
200	**92.56** (.62)	**99.86 (0.04)**	**99.90 (0.03)**

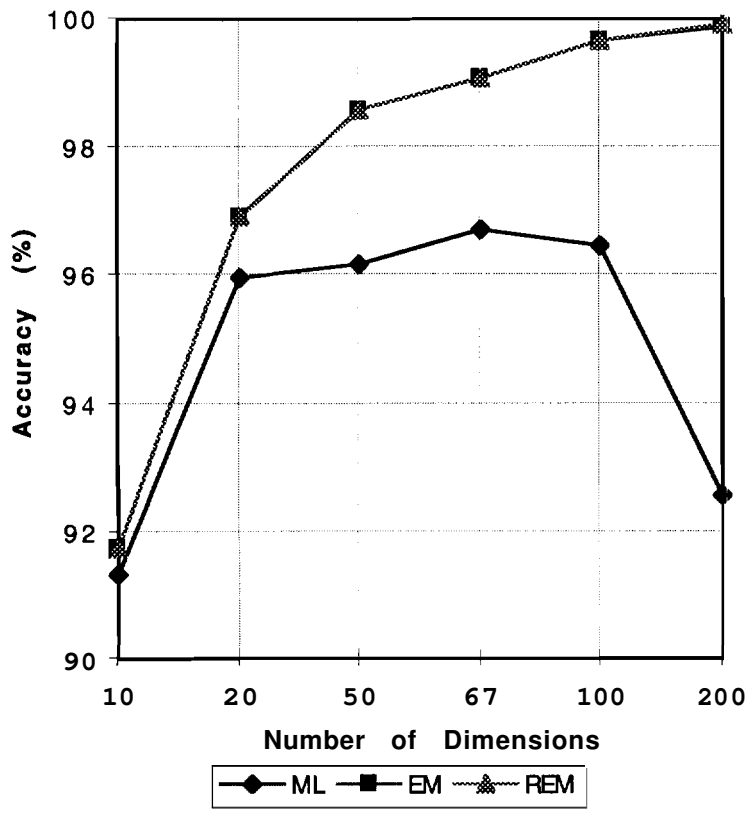


Figure **2.6** Mean Accuracy for Experiment **2.2**
with **250** Training Samples and **1500** Test Samples

Since fewer training samples are used, the performance of the maximum likelihood classifier (ML) using the training samples alone deteriorates. The decline is particularly obvious at higher dimensionality. Compared to the previous experiment, the accuracy has dropped 7% at 200 features. However, when unlabeled samples are used for the mixture model, the performance remains stable even when the nurnber of training samples declines. The results again show that when no outliers are present in the data, the EM and REM algorithms have comparable performance and both achieve better classification accuracy than the NIL classifier without using additional unlabeled samples.

Experiment 2.3

The previous experiment is repeated with only 400 test samples generated for each class. The number of training samples per class is 250. Again, no outliers are present in the data. The results are shown in Table 2.5 and Figure 2.7.

Table 2.5
Classification Results for Experiment 2.3
with 250 Training Samples and 400 Test Samples

Dimension	ML (%)	EM (%)	REM (%)
10	91.06 (.74)	91.41 (.18)	91.46 (.24)
20	95.94 (.28)	96.40 (.28)	96.40 (.28)
50	96.39 (.31)	97.61 (.26)	97.61 (.23)
67	96.14 (.76)	97.88 (.31)	97.90 (.33)
100	96.44 (.41)	97.56 (.52)	97.66 (.50)
200	92.16 (1.13)	92.31 (1.12)	94.10 (1.12)

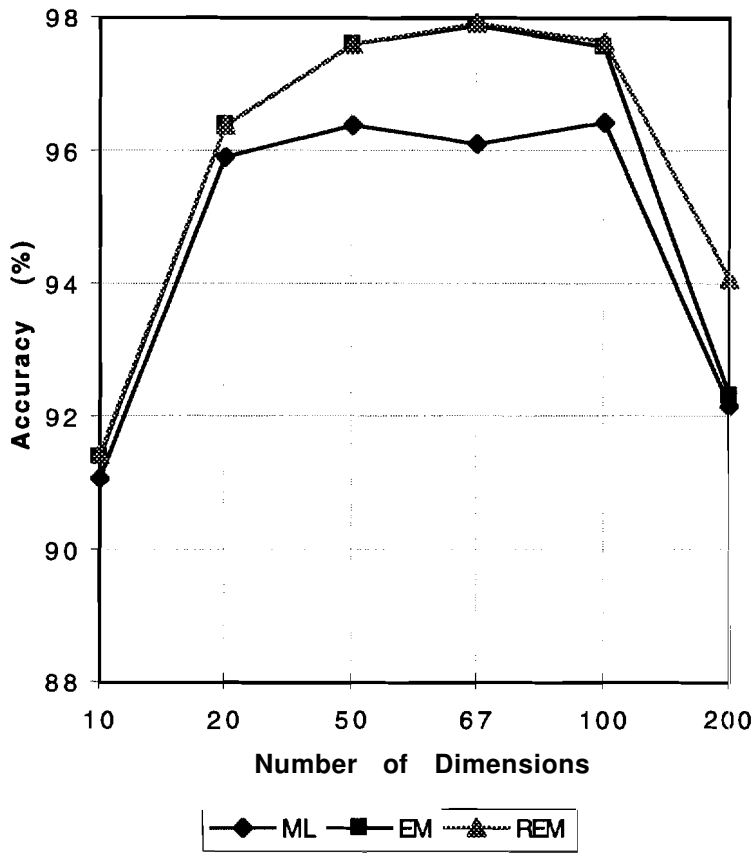


Figure 2.7 Mean Accuracy for Experiment **2.3**
with 250 Training Samples and 400 Test Samples

Compared to the results from two previous experiments in wlnich many more unlabeled samples were used, the classification results for all three methods deteriorate in this experiment. This deterioration is manifested as the Hughes phenomenon. Hence, the likelihood parameter estimation for the mixture model is shown to be affected by the number of unlabeled samples relative to dimensionality. Specifically, it implies that 650 samples are still inadequate to characterize 200-dimensional Gaussian distribution. The results again indicate that without outliers, the EM and REM algorithms have comparable performance and both have better classification accuracy than the ML classifier without using additional unlabeled samples.

Experiment 2.4

This experiment is conducted using the real samples from the data. Again, since all four classes are represented by the training samples, the classes are assumed to be exhaustive. As indicated in Table 2.2, the number of labeled samples is small. To retain enough test samples, only about 200 training samples are chosen for each class. The number of training samples are shown in Table 2.6. Due to the limited labeled sample size, to obtain reasonably good initial estimates for comparing the EM and REM algorithms, the number of spectral channels are selected at 10, 20, 50, 67 and 100. These spectral features are again chosen by sampling the spectral channels at fixed intervals. Table 2.6 and Figure 2.8 show the classification results at the selected dimensions.

Table 2.6
Training Set Size for Experiment 2.4

Class Names	**No. of Training Samples**
Corn-notill	221
Soybean-notill	221
Soybean-min	225
Grass	224

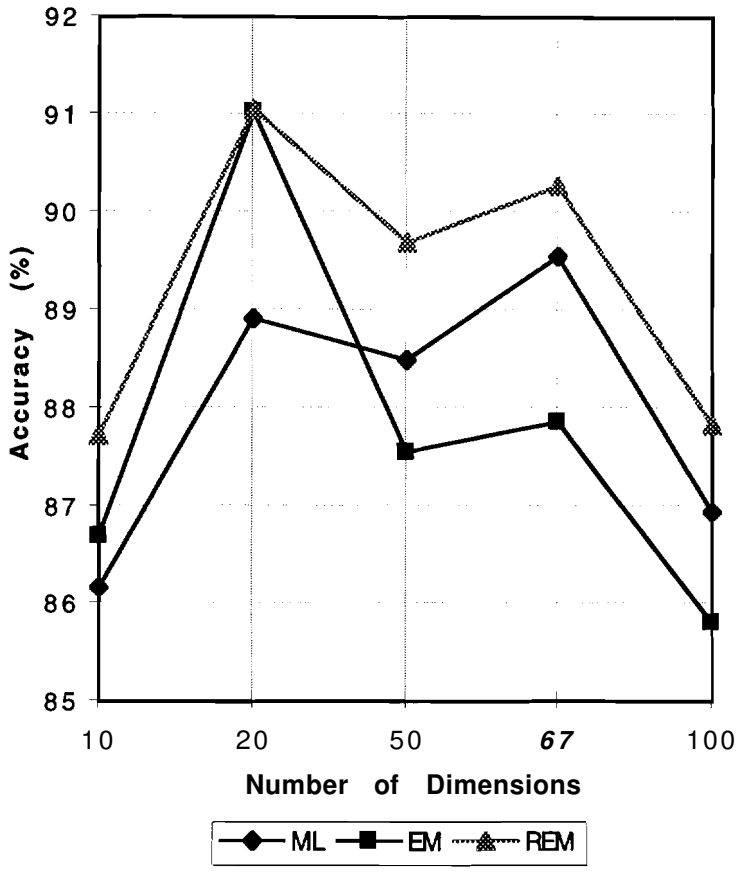


Figure 2.8 Accuracy for Experiment 2.4 using AVIRIS Data

The results show that the REM algorithm performs better than the ML and EM methods. This demonstrates that although it is assumed that the scene contains no outliers, there are some outlying pixels which were not identified. This further justifies the motivation of using a robust parameter estimation method for the mixture model. The results also show that all methods exhibit Hughes phenomenon. As discussed previously, the decline in performance at high dimensionality is caused by the limited number of unlabeled samples available in the image.

Experiment 2.5

In order to investigate the effect of outliers on the algorithms, the following experiment is conducted with the class Grass removed from the set of information classes. Therefore, the pixels other than the labeled samples from the three information classes are considered as outliers. The samples used for updating the statistics then

include the labeled samples and some outliers. The amount of outliers is varied to simulate different degrees of contamination. The numbers of outliers are chosen to be 200, 600 and 2000. Since the outliers are chosen randomly from the pool of unlabeled samples, the experiment is repeated 5 times. The mean and standard deviation of the classification accuracy are recorded. The results are presented in Table 2.7. The standard deviation is written in parenthesis next to the mean accuracy. In Figure 2.9 and 2.10, the mean accuracy is plotted against different number of outliers present in the data for 50 and 100 dimensions, respectively.

Table 2.7
Classification Results for Experiment 2.5 with Outliers

No. of Outliers	Dimension = 50			Dimension = 100		
	ML	EM	REM	ML	EM	REM
0	84.71 (0)	89.20 (0)	88.42 (0)	82.61 (0)	85.34 (0)	84.71 (0)
200	84.71 (0)	90.62 (.20)	90.29 (.11)	82.61 (0)	87.34 (.29)	86.56 (.36)
600	84.71 (0)	88.59 (.44)	88.69 (.58)	82.61 (0)	87.21 (.64)	87.08 (.45)
2000	84.71 (0)	62.57 (2.27)	76.34 (1.64)	82.61 (0)	83.33 (.73)	86.97 (.64)

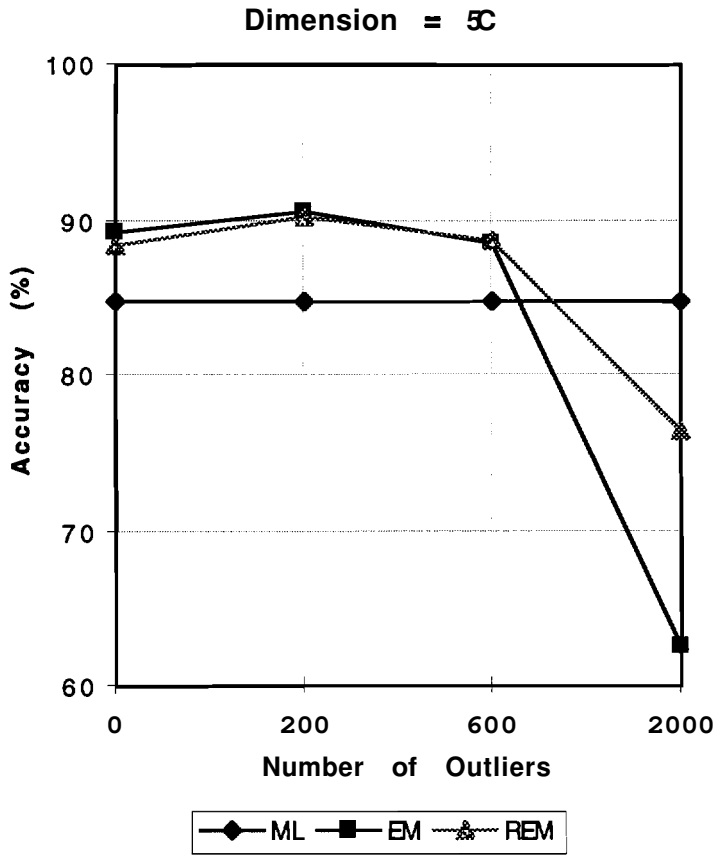


Figure 2.9 Mean Accuracy for Experiment 2.5 for 50 Dimensions

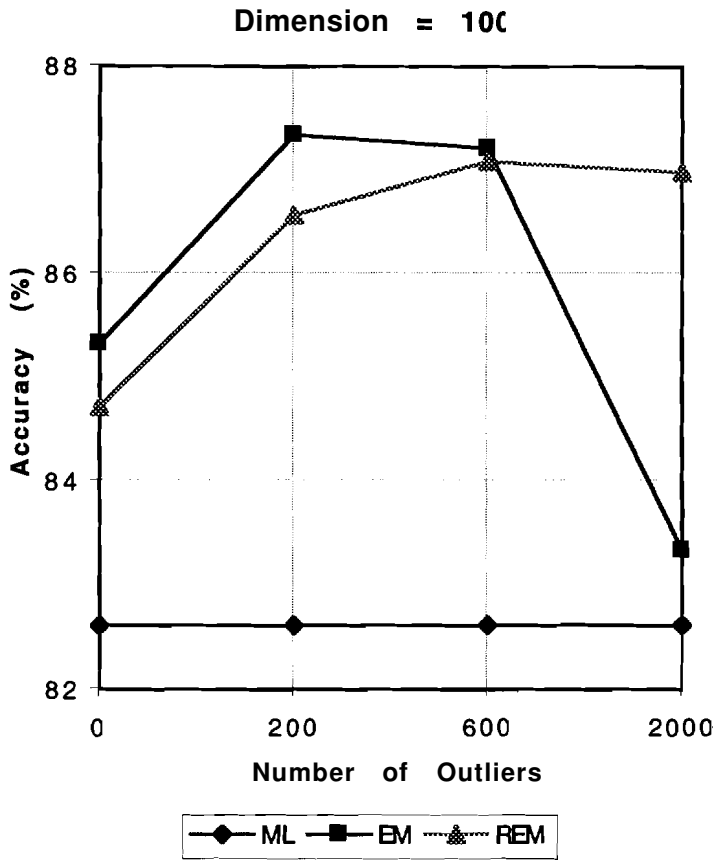


Figure 2.10 Mean Accuracy for Experiment 2.5 for 100 Dimensions

The results show that the REM algorithm reduces the effect of outliers contaminating the data as compared to the EM algorithm. The improvement is especially marked at higher dimensions. This may be attributed to the fact that at higher dimensionality, the weight assigned to each outlier is much more reduced since the weight is a function of dimensionality. Therefore, the effectiveness of the REM algorithm becomes more obvious.

Experiment 2.6

This experiment is conducted using a portion of the Flightline C1 (FLC1) data set, which is a 12 band multispectral image taken over Tippecanoe County, Indiana by the M7 scanner in June, 1966. The scene contains six information classes: Corn, Oats, Red Clover, Soybeans, Wheat and Rye. By visual inspection of the image, the list of these ground cover types is assumed to be exhaustive. The image and the ground truth map are shown in Figure 2.11. The training fields are marked in the ground truth map. The number of labeled samples and training samples in each class is shown in Table 2.8.

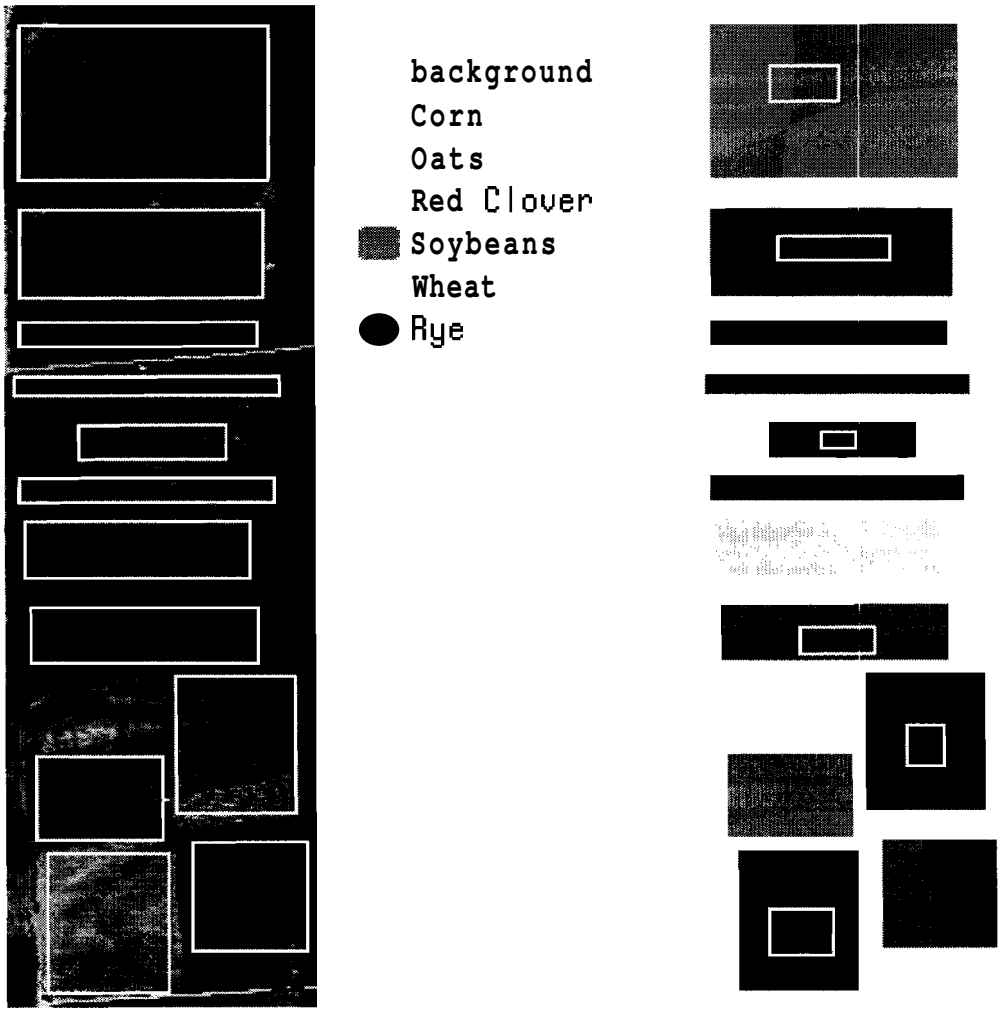


Figure 2.11 Portion of Flightline C1 Image and Ground Truth Map (Original in Color)

Table 2.8
Class Description for Flightline C1 Image in Figure 2.11

Class Names	No. of Labeled Samples	No. of Training Samples
Corn	1764	128
Oats	1516	78
Red Clover	3548	280
Soybeans	6758	338
Wheat	6846	588
Rye	2385	408

To create outliers in the data on purpose, the class Rye is excluded from the training class set and its samples are treated as outliers. Therefore, the classification is performed based on the 5 remaining classes only. The parameters are estimated using the training samples alone, the EM algorithm with various threshold settings, and the REM algorithm. For the EM algorithm, two chi-square threshold values (1% and 5%) are applied for comparison. The classification results are shown in Figure 2.12.

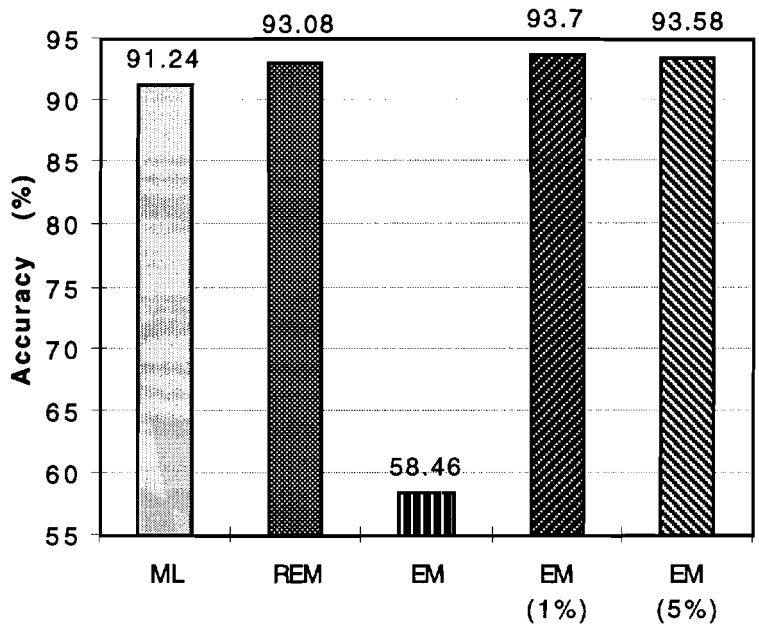


Figure 2.12 Classification Results for Portion of Flightline C1 Image with Outliers

As shown in Figure 2.12, when there are statistical outliers in the data, the performance of the EM algorithm declines drastically. However, by rejecting outliers using chi-square thresholds, the EM algorithm shows significant improvement. The result also indicates that REM and EM with thresholding have comparable performance and are better than the ML method with training samples alone.

Experiment 2.7

The above experiment is repeated with the entire Flightline C1 image. The image and the ground truth map are shown in Figure 2.13. The training fields are marked in the ground truth map. The number of labeled samples and training samples in each class is shown in Table 2.9. The classification results are plotted in Figure 2.14.

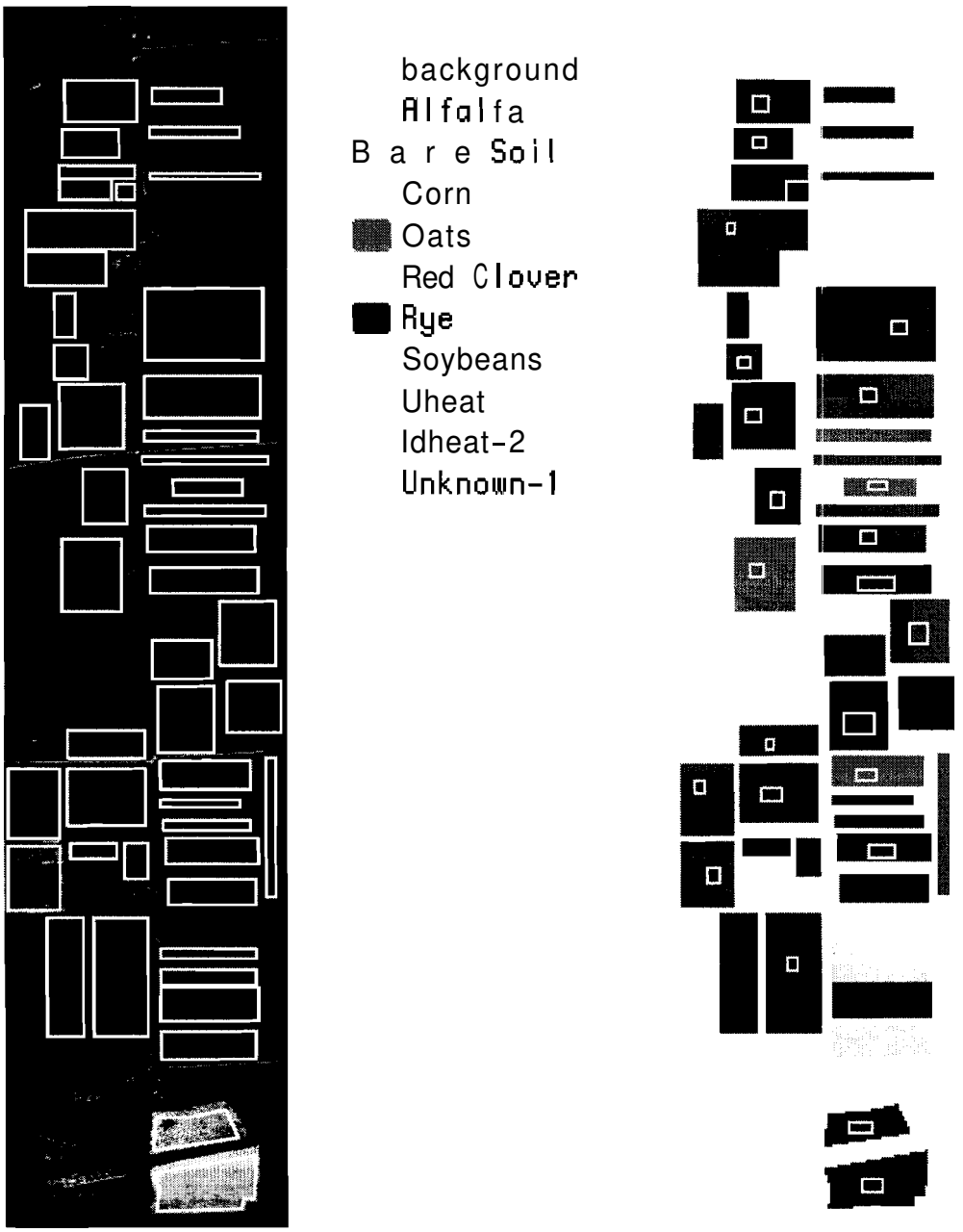


Figure 2.13 Flightline C1 Image and Gound Truth Map (Original in Color)

Table **2.9**
Class Description for Flightline C1 Image in Figure **2.13**

Class Names	No. of Labeled Samples	No. of Training Samples
Alfalfa	3375	156
Bare Soil	1230	90
Corn	10625	331
Oats	5781	306
Red Clover	12147	614
Rye	2385	408
Soybeans	25133	631
Wheat	7827	340
Wheat-2	2091	120
Unknown-1	4034	322

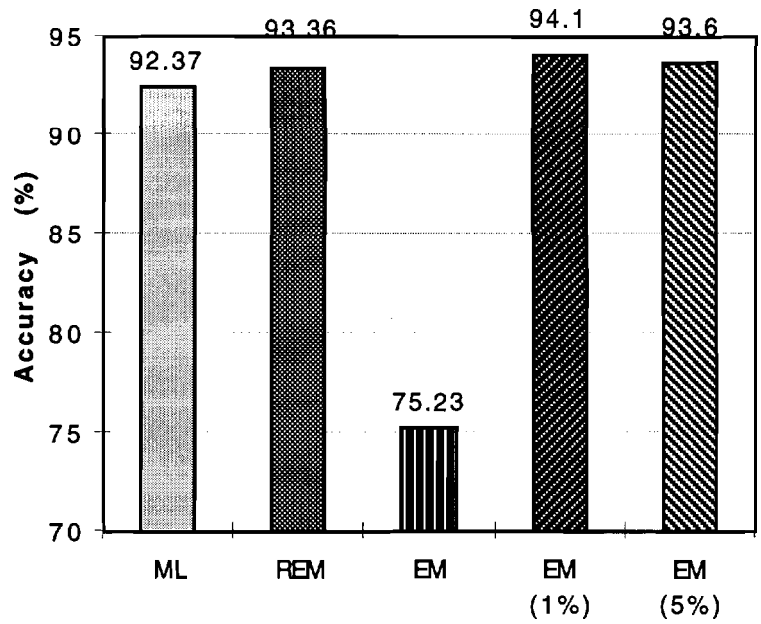


Figure **2.14** Classification Results for Flightline C1 Image

The entire Flightline C1 image contains classes with few pixels such as rural roads, fannstead and water which are not included in the training set. There may be other unknown classes which are not identified in the ground truth information. Therefore, it is highly likely that statistical outliers are present in the image. This is confirmed by experimental results. The performance of the EM algorithm is significantly lower than those of ML, REM and EM with thresholding. Again, the experiment demonstrates that REM has similar performance as EM with thresholding, but without the need of setting a threshold.

2.5 Summary

In this chapter, a robust method for parameter estimation under the mixture model (REIM) is proposed and implemented for classifying multispectral data. This work is motivated by the fact that a multispectral image usually contains pixels of unknown classes which can be time-consuming to identify. These pixels of unknown origin may have density distributions quite different from the training classes and constitute statistical outliers. Without a list of exhaustive classes for the mixture model, the expectation maximization (EM) algorithm can converge to erroneous solutions due to the presence of statistical outliers. This problem necessitates a robust version of the EM algorithm which includes a measure of typicality for each sample.

The experimental results have shown that the proposed robust method performs better than the parameter estimation methods using the training samples alone (ML) and the EM algorithm in the presence of outliers. When no outliers are present, the EM and REM have similar performance and both are better than the ML approach. Specifically, when there are many unlabeled samples available, the EM and REM algorithms can mitigate the Hughes phenomenon since they utilize unlabeled samples in addition to the training samples. When the number of unlabeled samples are limited, both EM and REM methods exhibit the Hughes phenomenon, but still achieve better classification accuracy than the ML approach at lower dimensionality. Despite the promising results, the proposed REM algorithm has several limitations. Since the weight function in the REM algorithm is based on class statistics, the initial parameter estimates are important in determining the convergence. In particular, a good covariance estimate requires sufficient number of training samples. When the number of training samples is close to or less than dimensionality, the covariance estimate becomes singular and the EM or REM algorithm cannot be applied. This issue is addressed in the next chapter where a covariance estimation method for limited training samples is proposed.

CHAPTER 3: COVARIANCE ESTIMATION FOR LIMITED TRAINING SAMPLES

3.1 Introduction

In Gaussian maximum likelihood classification, the mean vector and covariance matrix are usually estimated from the training samples. When the training sample size is small compared to the dimensionality, the sample estimates, especially the covariance estimate become highly variable and consequently, the classifier performs poorly. In particular, if the number of training samples is less than the dimensionality, the covariance estimate becomes singular and hence quadratic classifiers cannot be applied. Unfortunately, the problem of limited training samples is prevalent in remote sensing applications. While the recent progress in sensor technology has increased the number of spectral features, making possible for more classes to be identified, the training data remain expensive and difficult to acquire. In this chapter, the problem of small training set size on the classification performance is addressed by introducing a covariance estimation method for limited training samples. The proposed approach can be viewed as an intermediate method between linear and quadratic classifiers by selecting an appropriate mixture of covariance matrices of various forms using the training samples. The covariance estimator is derived under an empirical Bayesian setting which is advantageous when the training sample size reflects the prior of each class. The effect of covariance estimation on discriminant analysis feature extraction technique is also investigated. Extensive experiments are performed using simulation data and hyperspectral images. The experimental results show that the proposed covariance estimator improves classification performance when the training samples; are limited.

3.2 Preliminaries

3.2.1 Introduction

The purpose of classification is to assign unlabeled samples to one of several groups or classes. In the conventional Gaussian maximum likelihood (MIL) classifier, the classification rule can expressed in the form of a discriminant function and a sample is assigned to the class with the largest value. A multivariate Gaussian distribution is given as

$$f_i(x) = (2\pi)^{-p/2}|\Sigma_i|^{-1/2} exp\left[-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right] \qquad 1 \le i \le L$$

where μ_i and Σ_i are the ith class mean vector and covariance matrix, respectively, and L is the number of classes and $x \in \Re^p$. Assuming a [0,1] loss function, the maximum likelihood classification rule then becomes

$$d_i(x) = \min_{1 \le i \le L} d_i(x) \tag{3.1}$$

where d_i is the discriminant function given by

$$d_i(x) = (x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) + ln|\Sigma_i|.$$

This classification rule is also called a quadratic classifier. A special case occurs when all of the class covariance matrices are identical and it becomes a linear classifier:

$$\Sigma_i = \Sigma \qquad 1 \le i \le L$$

In practical situations, the true class distributions are rarely known. Therefore, the sample estimates are computed from training samples and are used as the maximum likelihood estimates of the parameters:

$$\hat{\mu}_i^{ML} = m_i = \frac{1}{N_i}\sum_{j=1}^{N_i} x_{i,j}$$

$$\hat{\Sigma}_i^{ML} = S_i = \frac{1}{N_i - 1}\sum_{j=1}^{N_i} (xi,j - m_i)\left(x_{i,j} - mi\right)^T$$

where N_i is the total number of training samples from class i and $x_{i,j}$ is the training sample j from class i.

The performance of Eq. *(3.1)* can be seriously degraded when the number of dimensions is large compared to the training set size due to the instabillity of the sample estimates. In particular, the sample covariance estimate S_i becomes highly variable. The inverse of S_i is especially problematic, as can be seen by the fact that S_i is not invertible for $p \geq N_i - 1$.

One way to deal with the instability of S_i is to employ the linear classifier which is obtained by replacing each S_i with their average:

$$S_w = \frac{(N_1 - 1)S_1 + (N_2 - 1)S_2 + \cdots + (N_L - 1)S_L}{N - L} \tag{3.2}$$

where N is the total number of training samples from all classes. Since S_w is a weighted average of S_i, the number of parameters becomes smaller, and the variances of the elernents of S_w are smaller than the variances of the corresponding elements of S_i. Even if each Σ_i differs substantially, the decrease in variance accomplished by using S_w usually leads to better performance for limited training samples. This has been verified by several studies [19], [20], [21].

In view of these results, several methods have been proposed vvhere the sample covariance estimate is replaced by partially pooled covariance matrices of various forms. In this formulation, some degree of regularization is applied to reduce the number of parameters to be estimated and thereby to improve classification performance in small training set size.

3.2.2 Regularization for covariance estimation

Although a linear classifier often performs better than a quadratic classifier for small training set size, the choice between linear and quadratic classifiers is quite restrictive. There are less limiting alternatives by applying varying degrees of regularization depending on the training samples. Thus, regularization techniques can also be viewed as choosing an intermediate classifier between the linear and quadratic classifiers.

Regularization techniques have shown much success in solving ill- and poorly-posed inverse problems [22]. Briefly stated, a problem is poorly posed if the number of parameters to be estimated is comparable to the training data size and ill-posed if that

number exceeds the training sample size. As a result of limited training set size, the parameter estimates become highly variable. Regularization methods attempt to reduce the variance of these estimates by biasing them toward values that are deemed more "physically plausible" [23]. Therefore, the variance is reduced at the expense of potentially increased bias. The extent of this bias-variance trade-off is controlled by one or more regularization parameters.

In general, regularization procedures can be divided into two tasks: 1) the choice of covariance mixture models, and 2) model selection.

To perform regularization, one must first decide upon a set of appropriate covariance mixture models that represent a "plausible" set of covariance estimates. Normally, a covariance mixture of the following form is assumed:

$$\hat{\Sigma}_i = (1 - w_i)S_i + w_i S_p \qquad 0 \le w_i \le 1$$

The regularization or mixing parameter w_i then controls the biasing of individual class covariance sample estimate S_i to a pooled covariance matrix S_p. However, this partially pooled covariance estimate may not provide enough regularization even for a linear classifier. In the case when the total number of training samples N is comparable to or is less than the dimension p, even the linear classifier becomes ill- or poorly-posed. Therefore, an alternative covariance mixture is provided by biasing the sample covariance estimate towards some non-singular diagonal matrix A:

$$\hat{\Sigma}_i = (1 - w_i)S_i + w_i \Lambda \qquad 0 \le w_i \le 1$$

For given value(s) of the mixing parameter(s), the amount of bias will depend on how closely the estimates $\hat{\Sigma}_i$ actually represent those true parameters Σ_i. Therefore, the goal of model selection is to select appropriate values for the mixing parameters which can be estimated from minimizing a loss function based on the training samples.

A popular minimization criterion is based on cross-validated estimation of classification error. In the leave-one-out cross-validation error procedure, the classification rule is obtained from $N_i - 1$ training samples excluding $x_{i,k}$, the sample k frorn class i, and then used to classify $x_{i,k}$. This criterion has the benefit of being directly related to classification accuracy even though it is computationally intensive. However, the process of estimating each class covariance matrix involves the covariance estimates

of all classes, which implies that the same mixing parameter has to be used for all classes. However, the same choice of mixing parameter might not be optimal for all classes. Furthermore, the same classification error rate might occur along a wide range of parameter values and hence the optimal value of mixing parameter is non-unique. Therefore, a tie-breaking technique is needed.

Another maximization criterion which has been applied is the sum of the average leave-one-out likelihood value of each class. In this procedure, the likelihood of each $x_{i,k}$ is obtained using the parameters estimated from $N_i - 1$ training samples exclusive of $x_{i,k}$. This criterion requires less computation than the leave-one-out classification error procedure. It also has the advantage that each class covariance matrix can be estimated independently of the others. Therefore, the mixing parameter can be different for each class. Moreover, not all classes need to be subjected to regularization, especially those with sufficient training samples. However, a major drawback of this criterion is the lack of direct relationship with classification accuracy.

3.2.3 Previous work

This section gives an overview of some regularization methods for covariance estimation based on limited training samples.

REGULARIZED DISCRIMINANT ANALYSIS (RDA)

Friedman[23] has proposed a procedure called "regularized discriminant analysis" (RDA) which is a two-dimensional optimization over covariance mixtures as shown in the following:

$$\hat{\Sigma}_i(\lambda,\gamma) = (1-\gamma)\hat{\Sigma}_i(\lambda) + \gamma\left(\frac{tr\left(\hat{\Sigma}_i(\lambda)\right)}{p}\right)I \qquad 0 \le \gamma \le 1 \tag{3.3}$$

where

$$\hat{\Sigma}_i(\lambda) = \frac{(1-\lambda)(N_i-1)S_i + \lambda(N-L)S_w}{(1-\lambda)N_i + \lambda N} \qquad 0 \le \lambda \le 1 \tag{3.4}$$

and S_w is given by Eq.(3.2).

The regularization parameters are given by the pair (λ, γ), which are obtained by minimizing the leave-one-out cross-validation errors. As mentioned before, the bias toward a diagonal matrix helps stabilize the covariance estimate even when the linear classifier is ill- or poorly-posed. Furthermore, choosing the diagonal form to be the average eigenvalue times the identity matrix has the effect of decreasing the larger eigenvalues and increasing the smaller ones, thereby counteracting the bias inherent in sample-based estimation of eigenvalues. This diagonal form is also advantageous when the true covariance matrices are some multiples of the identity matrix.

As mentioned before, although using cross-validation errors to select the mixing parameters has the benefit of being directly related to classification accuracy, it has some disadvantages as well. First of all, it is computationally intensive. In addition, the optimal values of (λ, γ) are often not unique since the same error rates can take place over a wide range of parameter values [24]. Therefore, a tie-breaking method needs to be applied. As a demonstration, an experiment was conducted on two-class simulation data. The number of training samples per class is 30 for two-dimensional Gaussian data. The following table indicates the cross-validation errors obtained for the parameter grid:

Table 3.1
Cross-validation Errors for RDA

γ \ λ	0	0.25	0.5	0.75	1
0	6	5	4	4	4
0.25	4	4	**3**	**3**	**3**
0.5	7	6	6	5	4
0.75	8	8	7	6	5
1	9	9	8	8	8

As shown in the table, the optimal value of the mixing parameters occurs when three training samples are misclassified, which occurs at λ=0.5, 0.75 and 1. Therefore, the optimal value is non-unique. No studies have indicated the best method for tie-breaking. As another consequence of using cross-validation errors, the same parameter pair has to be used for all classes since the classification procedure requires all covariance estimates simultaneously. The same value of (λ, γ) may not be optimal for all classes.

LEAVE-ONE-OUT COVARIANCE (LOOC) ESTIMATOR

In [25][43], the covariance matrix is determined from the following pair-wise mixtures: diagonal sample covariance-sample covariance, sample covariance-common covariance, and common covariance-diagonal common covariance matrices. Thus, the estimator has the following form:

$$\hat{\Sigma}_i(\alpha_i) = \begin{cases} (1-\alpha_i)diag(S_i) + \alpha_i S_i & 0 \le \alpha_i \le 1 \\ (2-\alpha_i)S_i + (a,-1)S & 1 < \alpha_i \le 2 \\ (3-\alpha_i)S + (a,-2)diag(S) & 2 < \alpha_i \le 3 \end{cases}$$

where

$$S = \frac{1}{L}\sum_{i=1}^{L} S_i .$$

The variable α_i is the mixing parameter that determines which estimate or mixture of estimates is selected so that the best fit to the training samples is achieved by maximizing the average leave-one-out log likelihood of each class:

$$LOOL_i = \frac{1}{N_i}\sum_{k=1}^{N_i} ln\left[f\left(x_{i,k}\middle|m_{i\backslash k}, \hat{\Sigma}_{i\backslash k}(\alpha_i)\right)\right]$$

where sample k from class i is removed. Once the appropriate value of α_i has been estimated, the estimated covariance matrix is computed with all the training samples and is used in the Gaussian ML classifier.

Since the leave-one-out class likelihood is used as the optimization criterion, each class covariance estimate can be computed independently and each has a different mixing parameter. One benefit of deriving the class covariance matrix separately is that the computation for classes with enough training samples can be skipped and consequently the computational load is reduced. In addition, if some classes have many more training samples than others, the classes may be allowed to have different mixing parameters. Using an approximation on the diagonal matrices, LOOC also requires less computation than RDA. However, without the approximation, LOOC is more computationally expensive than RDA. Also, the average leave-one-out likelihood has no direct relationship to classification accuracy.

OTHER COVARIANCE ESTIMATION METHODS

Some earlier works on covariance estimation methods involve the estimation of a single covariance matrix based on some loss functions. It was shown that Stein-like biased estimators which shrink the eigenvalues of the sample covariance matrix are favored over the sample covariance matrix under a variety of natural loss functions [26]. Also, when the class covariance matrices are equal, the pooled covariance matrix can be replaced by ridge-like estimates [27]. This reduces the ratio of the largest and smallest eigenvalues of the pooled estimate and thus has an effect similar to shrinking the eigenvalues of the pooled estimate towards equality.

An empirical Bayesian method [24] was suggested in which the Σ_i are modeled as outcomes of a common inverted Wishart prior distribution. The form of covariance mixtures is similar to Eq (3.3) and (3.4) as in RDA except for the pooled covariance estimate which is formulated under the Bayesian context. The optimal values for (λ, γ) are selected by maximizing the sum of average leave-one-out class likelihood. As mentioned before, this criterion has the merits of fewer computations than cross-validation errors and of avoiding the need for tie-breaking. However, the criterion is not directly linked to classification accuracy. Also, this method requires two-way optimization for the parameter pair (λ, γ). Therefore, it requires more computation than LOOC.

3.3 A New Method For Covariance Estimation

3.3.1 Derivation of the proposed estimator

A new covariance estimation method is developed in this section. The proposed estimator is essentially an extension of previous works in RDA, LOOC and the empirical Bayesian approach[24].

Case 1: $N \geq (p+1)$

The: first form of covariance mixtures is derived by assuming that the total number of training samples is greater than dimensionality. In this case, the common covariance matrix is non-singular. Following Anderson [28], the assumption of normally distributed samples implies that the sample covariance matrices S_i are mutually independent with

$$S_i \sim W\left(\frac{1}{f_i}\Sigma_i, f_i\right)$$

where $f_i = N_i - 1$ and W denotes the central Wishart distribution with f_i degrees of freedom and parameter matrix Σ_i. Then the family of inverted Wishart distributions provides a convenient family of prior distributions for the Σ_i.

Assume that each Σ_i has an inverted Wishart prior distribution so that the Σ_i are mutually independent with

$$\Sigma_i \sim W^{-1}\left((t-p-1)\Psi, t\right) \qquad t > p+1$$

where W^{-1} is an inverted Wishart distribution with parameters Ψ and t. Then the prior mean Ψ represents the central location of the prior distribution of the Σ_i, and t controls the concentration of the Σ_i around Ψ.

Under squared error loss, the Bayes estimator of Σ_i is given by [24]

$$\hat{\Sigma}_i(\Psi, t) = \frac{f_i}{f_i + t - p - 1} S_i + \frac{t-p-1}{f_i + t - p - 1}\Psi.$$

By letting $w_i = \dfrac{t-p-1}{f_i + t - p - 1}$, and Ψ be a pooled covariance estimate S_p, the Σ_i can then be replaced by partially pooled estimates of the form :

$$\hat{\Sigma}_i = (1 - w_i)S_i + w_i S_p \qquad 0 \leq w_i \leq 1.$$

This is the form similar to the sample covariance-common covariance mixtures in RDA and LOOC. The value of t can then be expressed in terms of w_i:

$$t = \frac{w_i(f_i - p - 1) + p + 1}{1 - w_i} \qquad 0 \le w_i < 1. \tag{3.5}$$

The sample covariance-common covariance mixture in LOOC is obtained by defining the pooled covariance matrix to be the unweighted common covariance, that is, $S_p = S$. In the proposed method, S_p is defined by the generalized least squared estimator of Ψ, designated as $S_p^*(t)$, for a given t:

$$S_p^*(t) = \left(\sum_{i=1}^{L} \frac{f_i}{f_i + t - p - 1} \right)^{-1} \sum_{i=1}^{L} \frac{f_i}{f_i + t - p - 1} S_i$$

Therefore, by letting $S_p = S_p^*$ which is the weighted common covariance matrix, another fonn of covariance mixture is obtained. Observe that when the number of training samples in each class is equal, that is, $f_1 = f_2 = \cdots = \mathrm{f},$, $S_p^*(t)$ is equivalent to S.

Case 2: $N < (p+1)$

When the total number of training samples is close to or less than the number of features, even the pooled covariance matrix becomes unstable. In this case, biasing the sample and common covariance estimates towards some form of diagonal matrix can avoid the problem of singularity. In LOOC, the sample and common covariance estimates are biased towards their own diagonal elements. This mixture: is advantageous when the class covariance matrix is highly ellipsoidal. However, the set of covariance mixtures should represent a wide variety of covariance matrices including the spherical structure. This can be achieved using the ridge estimator. The ridge estimator has the fonn of the sample covariance plus a constant times the identity matrix. With a proper choice of the constant value, it has the benefit of compensating for the upward bias of large eigenvalues and downward bias of small eigenvalues. In addition, this covariance mixture is apparently advantageous when the class covariance is some multiple of the identity.

Hence, when the ridge estimator is adopted, the proposed estimator of the following fonn:

$$\hat{\Sigma}_i(\alpha_i) = \begin{cases} (1-\alpha_i)\dfrac{tr(S_i)}{p} I + \alpha_i S_i & 0 \le \alpha_i < 1 \\ (2-\alpha_i)S_i + (\alpha_i - 1)S_p^*(t) & 1 \le \alpha_i < 2 \\ (3-\alpha_i)S + (\alpha_i - 2)\dfrac{tr(S)}{p} I & 2 \le \alpha_i \le 3 \end{cases}$$

Observe from Eq. (3.5) that when $\alpha_i = 2$, $t \to \infty$. Therefore, the unweighted common covariance is adopted for $\alpha_i = 2$. In the following sections, this estimator is designated as bLOOC1 (Bayesian Leave-One-Out Covariance estimation).

When the mixture of covariance-diagonal covariance matrices is used, the proposed estimator is defined as the following instead:

$$\hat{\Sigma}_i(\alpha_i) = \begin{cases} (1-\alpha_i)diag(S_i) + \alpha_i S_i & 0 \le \alpha_i < 1 \\ (2-\alpha_i)S_i + (\mathrm{a},-1)S_p^*(t) & 1 \le \alpha_i < 2 \\ (3-\alpha_i)S + (\mathrm{a},-2)diag(S) & 2 \le \alpha_i \le 3 \end{cases}$$

This estimator is denoted as bLOOC2 in future reference. In the experiments, the relative merits of these two estimators are demonstrated and discussed.

3.3.2 Model selection

For the proposed estimators, the leave-one-out average likelihood is used as the criterion to select the appropriate mixture model. This criterion is equivalent to minimizing the Kullback-Leibler distance measure [29] defined as the following:

$$KL_i(\alpha_i) = \int f_i(x) log\left(\frac{f_i(x)}{\hat{f}_i^{\alpha_i}(x)}\right) dx$$

where f_i is the true density function of the ith class and $\hat{f}_i^{\alpha_i}$ is the normal density with sample mean estimates m_i and covariance matrix estimate $\hat{\Sigma}_i(\mathrm{a},)$.

Let $x_{i,k}$ denote the kth training sample from class i. The average leave-one-out log likelihood for class i is then given as:

$$LOOL_i(\alpha_i) = \frac{1}{N_i}\sum_{k=1}^{N_i} log\left(\hat{f}_{i\backslash k}^{\alpha_i}(x_{i,k})\right).$$

Then following [24] and [29],

$$E\{LOOL_i(\alpha_i)\} = E\left\{log\left(\hat{f}_{i\backslash k}^{\alpha_i}(x_{i,k})\right)\right\}$$

$$= E\left\{\int f_i(x) log\left(\hat{f}_{i\backslash k}^{\alpha_i}(x)\right)dx\right\}$$

$$\approx E\left\{\int f_i(x) log\left(\hat{f}_i^{\alpha_i}(x)\right)dx\right\}$$

$$= -E\left\{\int f_i(x) log\left(\frac{f_i(x)}{\hat{f}_i^{\alpha_i}(x)}\right)dx\right\} + \int f_i(x) log(f_i(x))dx$$

$$= -E\{KL_i(\alpha_i)\} + C.$$

Therefore, maximizing the cross-validated likelihood is equivalent to minimizing the Kullback-Leibler distance of the true and estimated densities. The mixing parameter is then selected so that the average leave-one-out likelihood is maximized.

3.3.3 Computational considerations

The direct implementation of the leave-one-out likelihood function for each class with N_i training samples would require the computation of N_i matrix inverses and determinants at each value of α_i. Fortunately, a more efficient implementation can be achieved by using the rank-one down-date of the covariance matrix. This section gives the efficient implementation of the proposed estimator bLOOC1. The efficient implementation of bLOOC2 can be derived from bLOOC1 and LOOC [25].

Efficient Implementation of the estimator for $0 \leq \alpha_i < 1$:

When the sample k is removed from class i, the sample covariance estimate can be written as follows [30]:

$$S_{i\backslash k} = \frac{1}{N_i - 2} \sum_{\substack{j=1 \\ j \neq k}}^{N_i} \left(x_{i,j} - m_{i\backslash k}\right)\left(x_{i,j} - m_{i\backslash k}\right)^T$$

$$= \frac{N_i - 1}{N_i - 2}\left[S_i - \frac{N_i}{(N_i - 1)^2} rr^T \right] \qquad \text{where } r = x_{i,k} - m_i.$$

The proposed estimator for $0 \leq \alpha_i < 1$ then becomes

$$\hat{\Sigma}_{i\backslash k}(\alpha_i) = (1 - \alpha_i)\frac{tr(S_{i\backslash k})}{P} I + \alpha_i S_{i\backslash k}$$

$$= \frac{(1 - \alpha_i)}{P} tr\left(\frac{N_i - 1}{N_i - 2} S_i - \frac{N_i}{(N_i - 1)(N_i - 2)} rr^T \right) I \cdots$$

$$\cdots + \alpha_i \left(\frac{N_i - 1}{N_i - 2} S_i - \frac{N_i}{(N_i - 1)(N_i - 2)} rr^T \right)$$

$$= \alpha_i \frac{N_i - 1}{N_i - 2} S_i + (1 - \alpha_i) \frac{N_i - 1}{p(N_i - 2)} tr(S_i) I - (1 - \alpha_i) \frac{N_i}{p(N_i - 1)(N_i - 2)} \|r\|^2 I \cdots$$

$$\cdots - \alpha_i \frac{N_i}{(N_i - 1)(N_i - 2)} rr$$

$$= S_i^+ - aI - k, rr^T$$

where $S_i^+ = \alpha_i \frac{N_i - 1}{N_i - 2} S_i$

$$a = \frac{(1 - \alpha_i)}{p}\left[\frac{N_i}{(N_i - 1)(N_i - 2)} \|r\|^2 - \frac{N_i - 1}{N_i - 2} tr(S_i) \right]$$

$$k_1 = \alpha_i \frac{N_i}{(N_i - 1)(N_i - 2)}.$$

Denote the eigenvalues of S_i^+ and their corresponding eigenvectors as e_i^+ and v_i^+, respectively. Then the following matrix inverse is obtained:

$$\left(S_i^+ - aI\right)^{-1} = \sum_{i=1}^{p} \frac{v_i^+ v_i^{+T}}{e_i^+ - a}.$$

Therefore, $\hat{\Sigma}_{i\backslash k}(\alpha_i)^{-1}$ can be computed efficiently using the Sherman-Morrison-Woodbury formula [31]:

$$\begin{aligned}
\hat{\Sigma}_{i\backslash k}(a,)' &= \left(S_i^+ - aI - k_1 r r^T\right)^{-1} \\
&= \left(A - zz^T\right)^{-1} \qquad \text{where } A = S_i^+ - aI \text{ and } z = \sqrt{k_1}\, r \\
&= A^{-1} + \frac{A^{-1} z z^T A^{-1}}{1 - z^T A^{-1} z} \\
&= A^{-1} + \frac{k_1 A^{-1} r r^T A^{-1}}{1 - k_1 r^T A^{-1} r}.
\end{aligned}$$

Using the matrix inverse obtained above, the squared generalized distance can then be expressed as:

$$\begin{aligned}
d_{i\backslash k} &= \left(x_{i,k} - m_{i\backslash k}\right)^T \hat{\Sigma}_{i\backslash k}(\alpha_i)^{-1} \left(x_{i,k} - m_{i\backslash k}\right) \\
&= \left(\frac{N_i}{N_i - 1}\right)^2 r^T \hat{\Sigma}_{i\backslash k}(\alpha_i)^{-1} r \qquad \text{where } x_{i,k} - m_{i\backslash k} = \frac{N_i}{N; - 1} r \\
&= \left(\frac{N_i}{N_i - 1}\right)^2 r^T \left[A^{-1} + \frac{k_1 A^{-1} r r^T A^{-1}}{1 - k_1 r^T A^{-1} r}\right] r \\
&= \left(\frac{N_i}{N; - 1}\right)^2 \left[\frac{d(1 - k_1 d) + k_1 d^2}{1 - k_1 d}\right] \qquad \text{where } d = r^T A^{-1} r \\
&= \left(\frac{N_i}{N_i - 1}\right)^2 \left[\frac{d}{1 - k_1 d}\right].
\end{aligned}$$

The: determinant also has a convenient form:

$$\left|\hat{\Sigma}_{i\backslash k}(\alpha_i)\right| = \left|A - rr^T\right|$$

$$= |A|(1 - k_1 d)$$

Therefore, the log likelihood of class i without sample k can be computed as [30]:

$$ln\left[f\left(x_{i,k}\middle|m_{i\backslash k}, \hat{\Sigma}_{i\backslash k}(a,)\right)\right] = -\frac{p}{2}ln(2\pi) - \frac{1}{2}ln(|A|) - \frac{1}{2}ln(1 - k_1 d)\cdots$$

$$\cdots - \frac{1}{2}\left(\frac{N_i}{N_i - 1}\right)^2\left[\frac{d}{1 - k_1 d}\right] \qquad 0 \le \alpha_i < 1.$$

The above computation can be further simplified by assuming the trace of the sample covariance changes little when a single sample is removed, that is, $\left(tr(S_{i\backslash k})/p\right)I \approx \left(tr(S_i)/p\right)I$. Experiments will confirm the validity of this approximation. In this case, the mixture can be re-expressed as:

$$\hat{\Sigma}_{i\backslash k}(\alpha_i) = (1 - \alpha_i)\frac{tr(S_{i\backslash k})}{p}I + \alpha_i S_{i\backslash k}$$

$$\approx (1 - \alpha_i)\frac{tr(S_i)}{p}I + \alpha_i S_{i\backslash k}$$

$$= (1 - \alpha_i)\frac{tr(S_i)}{p}I + \alpha_i\left(\frac{N_i - 1}{N_i - 2}S_i - \frac{N_i}{(N_i - 1)(N_i - 2)}rr^T\right)$$

$$= A_2 - z_2 z_2^T$$

where $A_2 = (1 - \alpha_i)\frac{tr(S_i)}{p}I + \alpha_i\frac{N_i - 1}{N_i - 2}S$

$$k_2 = \frac{\alpha_i N_i}{(N_i - 1)(N_i - 2)}$$

$$z_2 = \sqrt{k_2}\, r$$

$$d_2 = r^T A_2^{-1} r\,.$$

Finally, the log likelihood function is obtained as:

$$ln\left[f\left(x_{i,k}\middle|m_{i\backslash k},\hat{\Sigma}_{i\backslash k}(\alpha_i)\right)\right]=-\frac{p}{2}ln(2\pi)-\frac{1}{2}ln\left(|A_2|\right)-\frac{1}{2}ln(1-k_2d_2)\cdots$$

$$\cdots-\frac{1}{2}\left(\frac{N_i}{N_i-1}\right)^2\left[\frac{d_2}{1-k_2d_2}\right] \qquad 0\le\alpha_i<1.$$

Efficient Implementation of the estimator for $1\le\alpha_i<2$:

Consider the sample k is removed from class i. The sample covariance-common covariance mixture is then given by the following:

$$\hat{\Sigma}_{i\backslash k}(\alpha_i)=(2-\alpha_i)S_{i\backslash k}+(\alpha_i-1)S^*_{p\backslash k}(t)$$

$$=\frac{(f_i-1)}{(f_i-1)+t-p-1}S_{i\backslash k}+\frac{}{(f_i-1)+t-p-1}S^*_{p\backslash k}(t)$$

The weighted common covariance estimate without sample k from class i can then be derived as:

$$S^*_{p\backslash k}=\left(\sum_{j\neq i}^{L}\frac{f_j}{f_j+t-p-1}+\frac{f_i-1}{(f_i-1)+t-p-1}\right)^{-1}\cdots$$

$$\cdots\left[\sum_{\substack{j=1\\j\neq i}}^{L}\frac{f_j}{f_j+t-p-1}S_j+\frac{f_i-1}{(f_i-1)+t-p-1}S_{i\backslash k}\right]$$

$$=\left(\sum_{j=1}^{L}\frac{f_j}{f_j+t-p-1}-\frac{f_i}{f_i+t-p-1}+\frac{f_i-1}{(f_i-1)+t-p-1}\right)^{-1}\cdots$$

$$\cdots\left[\sum_{j=1}^{L}\frac{f_j}{f_j+t-p-1}S_j-\frac{f_i}{f_i+t-p-1}S_i+\frac{f_i-1}{(f_i-1)+t-p-1}S_{i\backslash k}\right]$$

$\hat{\Sigma}_{i\backslash k}(\alpha_i)$ can then be expressed in terms of S_p^*, S_i and r as follows:

$$\hat{\Sigma}_{i\backslash k}(\alpha_i) = k_1 S_p^*(t) + k_2 S_i - k_3 r r^T.$$

where the constants are defined in the following:

$$k_1 = (\alpha_i - 1)\left(C_1 - C_2 + (2 - \alpha_i)\right)^{-1} C_1$$

$$k_2 = (2 - a \quad \frac{\text{N.} - 1}{N_i - 2}) + \left(C_1 - C_2 + (2 - \alpha_i)\right)^{-1}(\alpha_i - 1)\left[(2 - \alpha_i)\frac{N_i - 1}{N_i - 2} - C_2\right] \text{ and}$$

$$k_3 = (2 - \alpha_i)\frac{N_i}{(N_i - 1)(N_i - 2)} + \left(C_1 - C_2 + (2 - \alpha_i)\right)^{-1}(\alpha_i - 1)(2 - \alpha_i)\frac{N_i}{(N_i - 1)(N_i - 2)}$$

and

$$(2 - a,) = \frac{f_i - 1}{(f_i - 1) + t - p - 1}$$

$$C_1 = \sum_{j=1}^{L} \frac{f_j}{f_j + t - p - 1}$$

$$C_2 = \frac{f_i}{f_i + t - p - 1}$$

The mixture can then be re-written as the following using previous derivations:

$$\hat{\Sigma}_{i\backslash k}(\alpha_i) = k_1 S_p^*(t) + k_2 S_i - k_3 r r^T$$

$$= A_3 - z_3 z_3^T$$

where $A_3 = k_1 S_p^*(t) + k_2 S_i$

$$z = \sqrt{k_3}\, r$$

$$d_3 = r^T A_3^{-1} r.$$

Then, the log likelihood function is given as:

$$ln\left[f\left(x_{i,k}\middle|m_{i\backslash k},\hat{\Sigma}_{i\backslash k}(\alpha_i)\right)\right]=-\frac{p}{2}ln(2\pi)-\frac{1}{2}ln\left(\left|A_3\right|\right)-\frac{1}{2}ln(1-k_3d_3)\cdots$$

$$\cdots-\frac{1}{2}\left(\frac{N_i}{N_i-1}\right)^2\left[\frac{d_3}{1-k_3d_3}\right] \qquad 1\le\alpha_i<2.$$

Effiicient Implementation of the estimator for $2 \le \alpha_i \le 3$:

For $2 \le \alpha_i \le 3$, the unweighted common covariance S is used: The common covariance estimate without sample k from class i can be written as follows:

$$S_{\backslash k}=\frac{1}{L}\sum_{\substack{j=1\\ j\neq i}}^{L}S_j+\frac{1}{L}S_{i\backslash k}$$

$$=\frac{1}{L}\sum_{j=1}^{L}S_j-\frac{1}{L}S_i+\frac{N_i-1}{L(N_i-2)}\left[S_i-\frac{N_i}{(N_i-1)^2}rr^T\right]$$

$$=S+\frac{1}{L(N_i-2)}S_i-\frac{N_i}{L(N_i-1)(N_i-2)}rr^T$$

The proposed estimator for $2 \le \alpha_i \le 3$ can then be written as:

$$\hat{\Sigma}_{i\backslash k}(\alpha_i)=(3-\alpha_i)S_{\backslash k}+\frac{tr(S_{\backslash k})}{p}I$$

$$=(3-\alpha_i)\left[S+\frac{1}{L(N_i-2)}S_i-\frac{N_i}{L(N_i-1)(N_i-2)}rr^T\right]\cdots$$

$$\cdots+\frac{(\alpha_i-2)}{p}tr\left[S+\frac{1}{L(N_i-2)}S_i-\frac{N_i}{L(N_i-1)(N_i-2)}rr^T\right]I$$

$$=S_i^{\#}-bI-k_4rr^T$$

where

$$S_i^{\#} = (3-\alpha_i)S + \frac{(3-\alpha_i)}{L(N_i-2)}S_i$$

$$b = \frac{(\alpha_i-2)N_i}{pL(N_i-1)(N_i-2)}\|r\|^2 - \frac{(\alpha_i-2)}{P}tr(S) - \frac{(\alpha_i-2)}{pL(N_i-2)}tr(S_i)$$

$$k_4 = \frac{(3-\alpha_i)N_i}{L(N_i-1)(N_i-2)}$$

Denote the eigenvalues of $S_i^{\#}$ and their corresponding eigenvectors as $e_i^{\#}$ and $v_i^{\#}$, respectively. Then the following matrix inverse is obtained:

$$\left(S_i^{\#} - bI\right)^{-1} = \sum_{i=1}^{p} \frac{v_i^{\#} v_i^{\#T}}{e_i^{\#} - b}.$$

Therefore, $\hat{\Sigma}_{i\backslash k}(\alpha_i)^{-1}$ can be computed efficiently following previous derivation:

$$\hat{\Sigma}_{i\backslash k}(\alpha_i)^{-1} = \left(S_i^{\#} - bI - k_4 rr^T\right)^{-1}$$

$$= \left(A_4 - z_4 z_4^T\right)^{-1}$$

where $A_4 = S_i^{\#} - bI$

$$z = \sqrt{k_4}\, r$$

$$d_4 = r^T A_4^{-1} r.$$

The log likelihood function is then given as:

$$ln[f(x_{i,k}|m_{i\backslash k}, \hat{\Sigma}_{i\backslash k}(\alpha,))] - \frac{p}{2}ln(2\pi) - \frac{1}{2}ln(|A_4|) - \frac{1}{2}ln(1-k_4 d_4) \cdots$$

$$\cdots - \frac{1}{2}\left(\frac{N_i}{N_i-1}\right)^2\left[\frac{d_4}{1-k_4 d_4}\right] \qquad 2 \le \alpha_i \le 3$$

The above computation can be further simplified by assuming the trace of the common covariance estimate changes little when a single sample is removed, that is, $\left(tr(S_{\backslash k})/p\right)I \approx \left(tr(S)/p\right)I$. In this case, the mixture can be derived as:

$$\begin{aligned}\hat{\Sigma}_{i\backslash k}(\alpha_i) &= (3-\alpha_i)S_{\backslash k} + (\alpha_i - 2)\frac{tr(S_{\backslash k})}{p} I \\ &\approx (3-\alpha_i)S_{\backslash k} + (\alpha_i - 2)\frac{tr(S)}{p} I \\ &= (3-\alpha_i)\left[S + \frac{1}{L(N_i-2)}S_i - \frac{N_i}{L(N_i-1)(N_i-2)}rr^T\right] + (\alpha_i - 2)\frac{tr(S)}{p} I \\ &= A_5 - z_5 z_5^T\end{aligned}$$

where $A_5 = (3-\alpha_i)S + \frac{(3-\alpha_i)}{L(N_i-2)}S_i + (\alpha_i - 2)\frac{tr(S)}{P} I$

$$z_5 = \sqrt{k_5}\, r$$

$$k_5 = \frac{(3-\alpha_i)N_i}{L(N_i-1)(N_i-2)}$$

$$d_5 = r^T A_5^{-1} r.$$

Finally, the log likelihood function is obtained as:

$$\begin{aligned}ln\left[f\left(x_{i,k}\middle|m_{i\backslash k}, \hat{\Sigma}_{i\backslash k}(\alpha_i)\right)\right] = &-\frac{p}{2}ln(2\pi) - \frac{1}{2}ln\left(|A_5|\right) - \frac{1}{2}ln(1-k_5 d_5)\cdots \\ &\cdots - \frac{1}{2}\left(\frac{N_i}{N_i-1}\right)^2\left[\frac{d_5}{1-k_5 d_5}\right] \qquad 2 \le \alpha_i \le 3.\end{aligned}$$

For notational purposes, in the following sections and experiments, the estimator with approximation on the diagonal term is designated as bLOOC1 (Bayesian Leave-One-Out Covariance estimation), whereas the implementation without approximation is denoted as bLOOC1-Exact.

3.4 Use of Covariance Estimation with Feature Extraction

When the number of training samples is few, the use of covariance estimation can help increase the stability of the covariance estimate and hence improve classification performance. Another usual way to deal with small training set size is to reduce the number of features using feature extraction algorithms. The goal of feature extraction is to project the original data to its subspace of lower dimensionality where the class sepiirability is preserved as much as possible. There are two feature extraction algorithms commonly used for remote sensing data, namely, Decision Boundary Feature Extraction (DBFE) [32] and Discriminant Analysis Feature Extraction (DAFE) [30]. The effects of covariance estimation on these two feature extraction algorithms are discussed in this section.

The procedure of DBFE involves finding the effective decision boundary between classes. For normally distributed classes, the description of decision boundary requires both the first and second order statistics. Therefore, a good covariance estimate is vital to the performance of DBFE. For a two-class case, the first step in DBFE uses the estimated mean vector and covariance matrix to find training samples which lie within the main body of the distribution using the chi-square threshold test. Then the nearest training samples from each of two classes are connected and a vector normal to the decision boundary is found at the point where the straight line connects the two training samples. It is desirable that the number of these unit normal vectors is proportional to the coniplexity of the decision boundary. For example, a linear boundary requires only one normal vector. Few training samples generates few normal vectors and usually result in inadequate description of the decision boundary. Hence, the effectiveness of DBFE depends not only on the covariance estimate, but also on the number of training samples. In other words, depending on the distribution, even though a reasonable covariance estimate can be found with no less than three training samples using the leave-one-out likelihood procedure, much more than three training samples may be required for DBFE to perform well. When these covariance estimates are extremely biased, a chi-square test may fail to find enough training samples for obtaining the effective decision boundary. Therefore, DBFE may not perform well when the number of training samples is limited, even with the covariance estimate stabilized.

On the other hand, the criterion used in DAFE procedure is the maximization of the ratio of between-class scatter matrix to within-class scatter matrix. For DAFE to perform

well, the mean difference cannot be zero and the common covariance has to be non-singular. These requirements can be met with considerably fewer trairdng samples than DBFE. When the mean estimates are fixed, it is helpful to reduce the between class scatter matrix, which is given by the pooled covariance matrix. Therefore, the use of covariance estimation should help improve the estimation of the within-class scatter matrix based on available training samples. In addition, when the inverse of the common covariance estimate approaches singularity, the covariance estimation techniques can be used for stabilization.

In summary, DBFE requires more training samples and computations than DAFE to perform well for small training set size even though it works better when classes have similar mean values. Due to these reasons, only the DAFE is used along with covariance estimation for mitigating small sample size problem.

3.5 Simulation Studies

In this section, the experimental results from computer generated data are presented. Seven covariance estimates, namely, the identity matrix, sample covariance, common covariance, and those obtained from RDA, LOOC, LOOC-Exact, bLOOC1, bLOOC1-Exact, bLOOC2, bLOOC2-Exact are compared. The mixing values are chosen to be 0, 0.25, 0.5, 0.75, 2, 2.25, 2.5, 2.75, and 3. Using the identity matrix as the covariance estimates for all classes is equivalent to the Euclidean distance classifier. The sample covariance and common covariance estimates lead to the quadratic and linear classifiers, respectively. The data distributions are generated from four different covariance structures as adapted from [23]. These simulated data represent the two extremes where one covariance matrix is spherical and the other is highly elliptical. The purpose of using these different types of covariance matrices is to demonstrate that the performance of covariance estimation techniques are affected by the underlying class covariance structure.

Two sets of experiments are conducted by having different proportions of training samples. In the first set, 10 training samples are randomly generated from each normally distributed class. The classification accuracy was estimated using 200 test samples. Each experiment is repeated 20 times from which the mean and variance of the classification accuracy are computed. The values of the mixing parameter are also

recorded. Since only 10 training samples are used for dimensions ranging from $p = 6$ to $p = 40$, the training set size is small compared to dimensionality.

In the second set of experiments, the number of samples differs for each class. The three classes are designed to have 100,400 and 2000 samples. Then, ten percent of the these samples are selected to be the training samples so that the number of training samples are 10, 40 and 200 for class one, class two and class three respectively. For $p = 6$ to $p = 40$, the training set size for the first and second classes is considered small. These experiments serve to represent the setting in which the number of training samples is unequal and is proportional to the size of test data.

In the tables below, the standard deviation of each result is listed in parentheses next to the corresponding mean value.

3.5.1 Equal spherical covariance matrices

In this experiment, all three classes have the identity covariance matrix. The mean of the first class is the origin. The mean of the second class is taken to be 3.0 in the first variable and zeros in the others, and the mean of the third class is 3.0 in the second variable and zeros in the rest. The results are shown in Tables 3.2, 3.3 and Figures 3.1, 3.2.

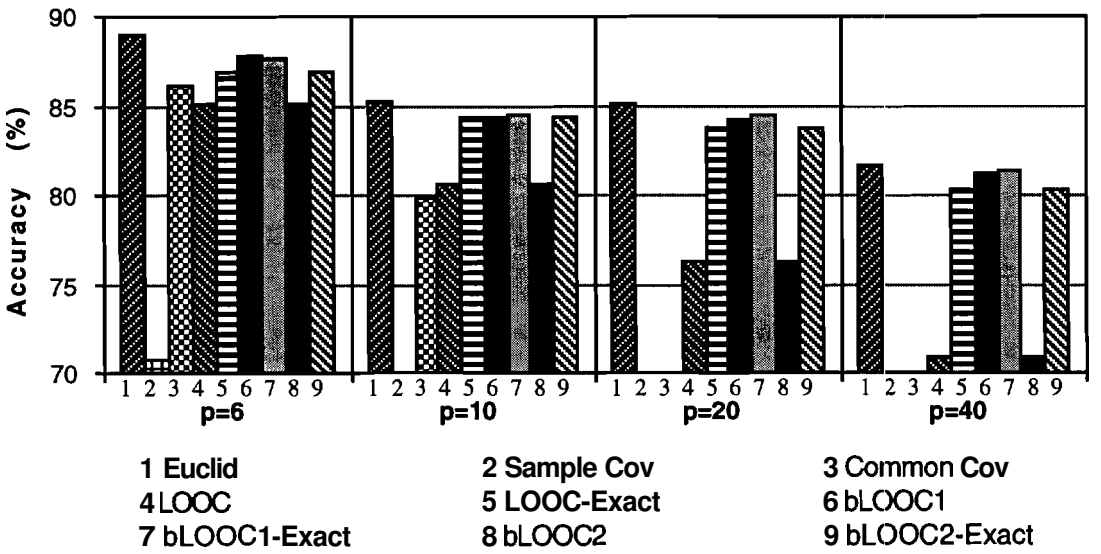


Figure 3.1 Mean Classification Accuracy for Equal Spherical Covariance Matrices (Equal Training Set Size)

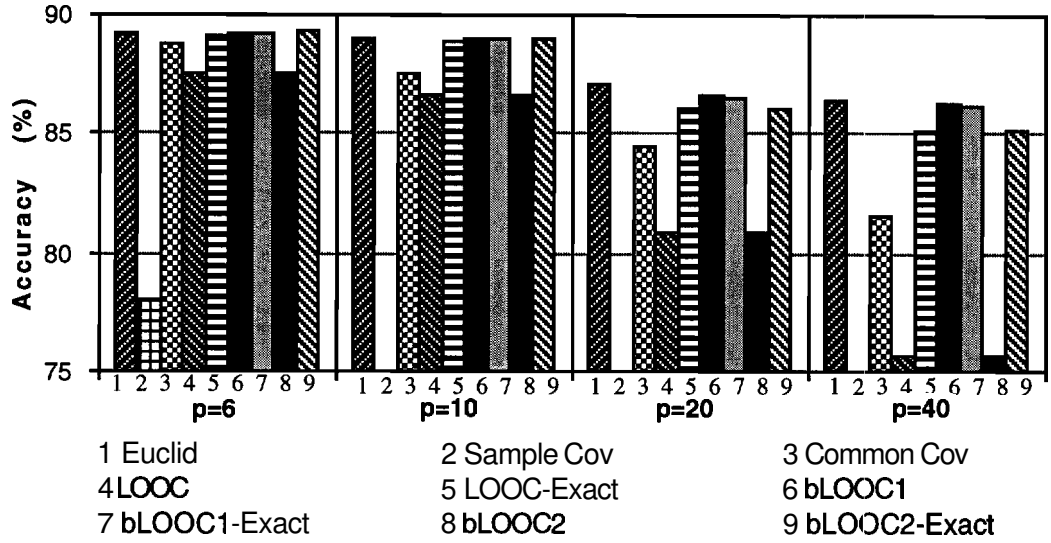
Table 3.2
Classification Results for Equal Spherical Covariance Matrices
(Equal Training Set Size)

Accuracy (%)					
		p=6	p=10	p=20	p=40
	Euclid	88.95 (4.29)	85.18 (4.79)	85.01 (4.94)	81.65 (6.24)
	Sample Cov	70.68 (17.1)	N/A	N/A	N/A
	Common Cov	86.10 (5.78)	79.87 (6.54)	66.89 (11.25)	N/A
	LOOC	85.08 (8.39)	80.59 (9.98)	76.34 (9.79)	70.89 (14.46)
	LOOC-Exact	86.90 (7.65)	84.27 (7.11)	83.70 (6.05)	80.34 (7.89)
	bLOOC1	87.69 (6.83)	84.28 (5.40)	84.24 (5.83)	81.22 (6.80)
	bLOOC1-Exact	87.68 (6.91)	84.54 (4.82)	84.43 (5.47)	81.27 (6.78)
	bLOOC2	85.08 (8.39)	80.59 (9.98)	76.34 (9.79)	70.89 (14.46)
	bLOOC2-Exact	86.90 (7.65)	84.27 (7.11)	83.70 (6.05)	80.34 (7.89)
Mixing Values					
LOOC	class1	0.01 (0.06)	0.03 (0.08)	0.00 (0.00)	0.00 (0.00)
	class2	0.00 (0.00)	0.00 (0.14)	0.00 (0.00)	0.01 (0.06)
	class3	0.08 (0.33)	0.03 (0.08)	0.01 (0.06)	0.00 (0.00)
LOOC-Exact	class1	2.53 (0.96)	2.64 (0.80)	2.98 (0.08)	2.99 (0.06)
	class2	2.66 (0.67)	2.93 (0.14)	2.96 (0.10)	3.00 (0.00)
	class3	2.89 (0.29)	2.80 (0.62)	2.93 (0.11)	3.00 (0.00)
bLOOC1	class1	1.23 (1.40)	1.35 (1.44)	1.78 (1.49)	1.35 (1.53)
	class2	0.09 (1.41)	1.92 (1.40)	1.49 (1.53)	1.35 (1.51)
	class3	0.66 (1.28)	1.24 (1.45)	1.49 (1.48)	1.50 (1.54)
bLOOC1-Exact	class1	1.96 (1.34)	1.93 (1.38)	2.08 (1.40)	2.53 (1.47)
	class2	2.40 (1.05)	1.75 (1.45)	2.36 (1.22)	1.50 (1.44)
	class3	2.46 (1.02)	2.24 (1.27)	1.64 (1.47)	1.66 (1.33)
bLOOC2	class1	0.01 (0.06)	0.03 (0.08)	0.00 (0.00)	0.00 (0.00)
	class2	0.00 (0.00)	0.00 (0.14)	0.00 (0.00)	0.01 (0.06)
	class3	0.08 (0.33)	0.03 (0.08)	0.01 (0.06)	0.00 (0.00)
bLOOC2-Exact	class1	2.53 (0.96)	2.64 (0.80)	2.98 (0.08)	2.99 (0.06)
	class2	2.66 (0.67)	2.93 (0.14)	2.96 (0.10)	3.00 (0.00)
	class3	2.89 (0.29)	2.80 (0.62)	2.93 (0.11)	3.00 (0.00)

Table 3.3
Classification Results for Equal Spherical Covariance Matrices
(Unequal Training Set Size)

Accuracy (%)

		p=6	p=10	p=20	p=40
	Euclid	89.19 (2.95)	89.01 (3.07)	87.05 (4.05)	86.35 (3.48)
	Sample Cov	78.02 (5.71)	N/A	N/A	N/A
	Common Cov	88.79 (3.32)	87.55 (3.65)	84.52 (4.26)	81.56 (3.41)
	LOOC	87.52 (4.27)	86.60 (4.48)	80.90 (4.63)	75.67 (3.59)
	LOOC-Exact	89.12 (3.17)	88.87 (3.04)	86.03 (4.21)	85.14 (3.31)
	bLOOC1	89.19 (3.21)	88.96 (3.00)	86.58 (4.08)	86.27 (3.40)
	bLOOC1-Exact	89.20 (3.14)	88.95 (3.08)	86.45 (4.30)	86.20 (3.33)
	bLOOC2	87.50 (4.27)	86.58 (4.00)	80.90 (4.63)	75.67 (3.59)
	bLOOC2-Exact	89.30 (3.12)	88.98 (3.10)	86.03 (4.21)	85.14 (3.31)
Mixing Values					
LOOC	class1	0.02 (0.08)	0.01 (0.06)	0.03 (0.08)	0.00 (0.00)
	class2	0.00 (0.00)	0.03 (0.08)	0.00 (0.00)	0.00 (0.00)
	class3	0.02 (0.08)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
LOOC-Exact	class1	2.74 (0.36)	2.73 (0.27)	2.87 (0.22)	2.96 (0.09)
	class2	2.54 (1.06)	2.24 (1.27)	2.98 (0.08)	3.00 (0.00)
	class3	0.28 (0.67)	0.33 (0.92)	0.04 (0.09)	0.00 (0.00)
bLOOC1	class1	1.71 (1.40)	1.41 (1.36)	1.30 (1.42)	1.64 (1.52)
	class2	0.63 (1.19)	1.15 (1.40)	1.65 (1.51)	1.20 (1.50)
	class3	0.38 (0.91)	0.45 (1.07)	0.01 (0.06)	0.15 (0.67)
bLOOC1-Exact	class1	2.51 (0.88)	2.22 (1.10)	2.11 (1.26)	1.46 (1.32)
	class2	1.60 (1.45)	1.74 (1.40)	2.39 (1.23)	2.10 (1.47)
	class3	1.12 (1.42)	1.65 (1.51)	0.91 (1.40)	0.60 (1.53)
bLOOC2	class1	0.02 (0.08)	0.01 (0.06)	0.03 (0.08)	0.00 (0.00)
	class2	0.03 (0.02)	0.03 (0.08)	0.00 (0.00)	0.00 (0.00)
	class3	0.02 (0.08)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
bLOOC2-Exact	class1	2.55 (0.20)	2.54 (0.50)	2.87 (0.22)	2.96 (0.09)
	class2	2.31 (0.72)	2.40 (0.67)	2.98 (0.08)	3.00 (0.00)
	class3	0.28 (0.67)	0.23 (0.72)	0.04 (0.09)	0.00 (0.00)



:Figure 3.2 Mean Classification Accuracy for Equal Spherical Covariance Matrices (Unequal Training Set Size)

For both equal and unequal number of training samples, the Euclidean distance classifier led to higher classification accuracy than any of the other covariance estimates, followed by bLOOC1 and bLOOC1-Exact. This result is expected since the Euclidean distance is equivalent to assuming the covariance matrices are the identity. Similarly, it is not surprising that the common covariance estimate led to higher accuracy than the sample covariance since the classes all have the same true covariance matrix. Since there are only 10 training samples for each class, the sample covariance could not be inverted for the higher dimensional data (p=10, 20, and 40), and so the classification accuracy could not be computed. The estimators LOOC, LOOC-Exact have the same performance as bLOOC2 and bLOOC2-Exact because the mixing values fall within the range of [0,1] and [2,3], under which these estimators have the same form of mixture:. The estimators bLOOC1 and bLOOC1-Exact perform better than LOOC, LOOC-Exact, bLOOC2 and bLOOC2-Exact in all four trials as the result of using the ridge estimator. Notice that bLOOC1 has similar performance as bLOOC1-Exact, which shows that the approximation of the trace of the sample and common covariance estimates is valid. On the other hand, LOOC and LOOC-Exact as well as bLOOC2 and bLOOC2-Exact produce rather different results when the training set size is moderate or small.

3.5.2 Unequal spherical covariance matrices

In this experiment, the three classes have unequal mean vectors and spherical covariance matrices. The mean vectors are the same as those in Experiment 3.5.1. The covariance matrices of class one, two and three are I, 2I, and 3I respectively. The results are presented in Tables 3.4, 3.5 and the mean accuracy are plotted in Figures 3.3, 3.4.

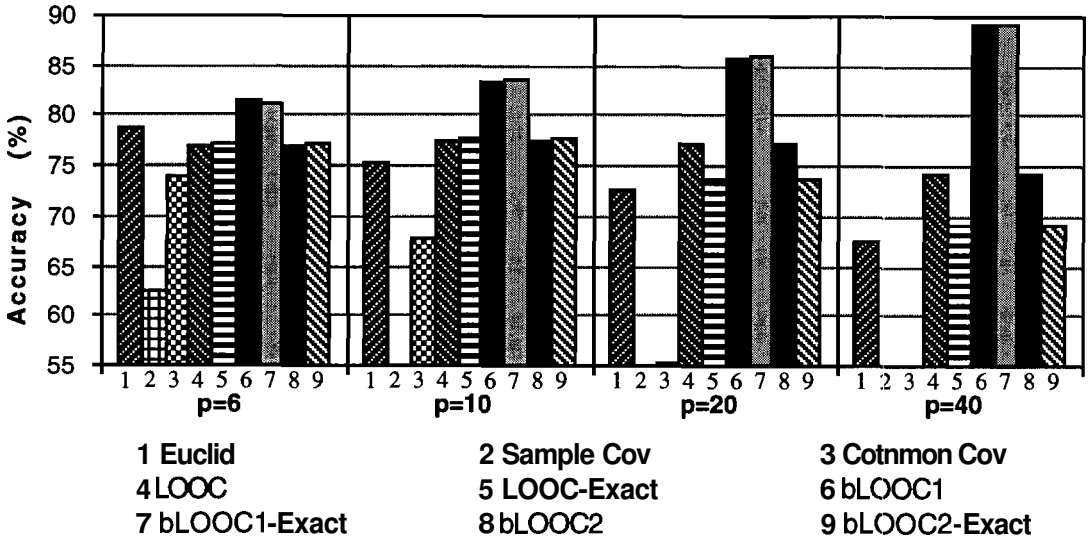


Figure 3.3 Mean Classification Accuracy for Unequal Spherical Covariance Matrices (Equal Training Set Size)

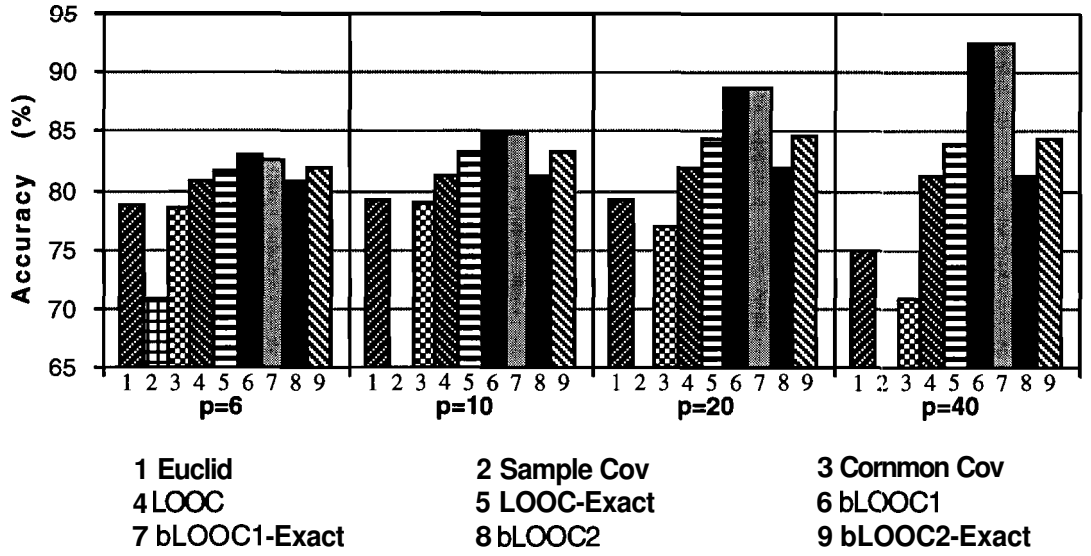


Figure 3.4 Mean Classification Accuracy for Unequal Spherical Covariance Matrices (Unequal Training Set Size)

Table 3.4
Classification Results for Unequal Spherical Covariance Matrices
(Equal Training Set Size)

Accuracy (%)

		p=6	p=10	p=20	p=40
	Euclid	78.82 (5.69)	75.40 (7.71)	72.59 (6.99)	67.64 (6.62)
	Sample Cov	62.54 (15.67)	N/A	N/A	N/A
	Common Cov	73.95 (7.79)	67.77 (9.35)	55.27 (11.72)	N/A
	LOOC	76.96 (8.31)	77.56 (7.65)	77.10 (10.61)	74.25 (9.48)
	LOOC-Exact	77.17 (7.54)	77.84 (9.87)	73.61 (10.12)	69.04 (12.40)
	bLOOC1	81.40 (5.78)	83.33 (5.68)	85.74 (5.64)	89.17 (4.03)
	bLOOC1-Exact	81.21 (6.22)	83.50 (5.59)	85.92 (5.30)	89.19 (4.11)
	bLOOC2	76.96 (8.31)	77.56 (7.65)	77.10 (10.61)	74.25 (9.48)
	bLOOC2-Exact	77.17 (7.54)	77.84 (9.87)	73.61 (10.12)	69.04 (12.40)
Mixing Values					
LOOC	class1	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	class2	0.01 (0.06)	0.14 (0.56))	0.01 (0.06)	0.00 (0.00)
	class3	0.00 (0.00)	0.01 (0.06)	0.01 (0.00)	0.00 (0.00)
LOOC-Exact	class1	1.94 (0.96)	1.59 (1.41)	2.63 (0.88)	2.62 (0.91)
	class2	2.66 (0.84)	2.81 (0.67)	2.83 (0.61)	3.00 (0.00)
	class3	1.06 (1.41)	0.88 (1.32)	1.36 (1.50)	0.75 (1.51)
bLOOC1	class1	0.05 (0.10)	0.05 (0.10)	0.03 (0.08)	0.01 (0.06)
	class2	1.08 (1.40)	0.91 (1.35)	1.21 (1.45)	1.35 (1.53)
	class3	0.06 (0.14)	0.03 (0.08)	0.04 (0.09)	0.00 (0.00)
bLOOC1-Exact	class1	0.34 (0.84)	0.06 (0.11)	0.03 (0.08)	0.01 (0.06)
	class2	1.96 (1.38)	1.96 (1.40)	1.95 (1.42)	1.93 (1.47)
	class3	0.06 (0.14)	0.05 (0.10)	0.04 (0.09)	0.00 (0.00)
bLOOC2	class1	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	class2	0.01 (0.06)	0.14 (0.56))	0.01 (0.06)	0.00 (0.00)
	class3	0.00 (0.00)	0.01 (0.06)	0.01 (0.00)	0.00 (0.00)
bLOOC2-Exact	class1	1.94 (0.96)	1.59 (1.41)	2.63 (0.88)	2.62 (0.91)
	class2	2.66 (0.84)	2.81 (0.67)	2.83 (0.61)	3.00 (0.00)
	class3	1.06 (1.41)	0.88 (1.32)	1.36 (1.50)	0.75 (1.51)

Table 3.5
Classification Results for Unequal Spherical Covariance Matrices
(Unequal Training Set Size)

Accuracy (%)		p=6	p=10	p=20	p=40
	Euclid	78.86 (3.71)	79.26 (3.95)	79.11 (3.09)	75.02 (3.97)
	Sample Cov	70.87 (6.77)	N/A	N/A	N/A
	Common Cov	78.53 (3.64)	79.00 (3.85)	77.04 (3.32)	70.89 (4.50)
	LOOC	80.69 (5.57)	81.25 (4.75)	81.88 (5.47)	81.24 (3.81)
	LOOC-Exact	81.71 (5.61)	83.36 (4.38)	84.31 (5.84)	83.97 (5.56)
	bLOOC1	82.98 (4.52)	85.01 (4.05)	88.63 (3.16)	92.41 (2.24)
	bLOOC1-Exact	82.64 (4.66)	84.96 (4.26)	88.59 (3.13)	92.41 (2.25)
	bLOOC2	80.69 (5.57)	81.25 (4.75)	81.88 (5.47)	81.24 (3.81)
	bLOOC2-Exact	81.81 (5.37)	83.29 (4.99)	84.56 (5.60)	84.38 (5.83)
Mixing Values					
LOOC	class1	0.05 (0.13)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	class2	0.00 (0.00)	0.01 (0.06)	0.00 (0.00)	0.00 (0.00)
	class3	0.01 (0.06)	0.01 (0.06)	0.00 (0.00)	0.00 (0.00)
LOOC-Exact	class1	1.24 (0.86)	1.83 (0.51)	1.96 (0.79)	2.58 (0.98)
	class2	2.94 (0.16)	2.88 (0.17)	3.00 (0.00)	2.69 (0.09)
	class3	0.05 (0.13)	0.06 (0.11)	0.04 (0.09)	0.00 (0.00)
bLOOC1	class1	0.20 (0.29)	0.03 (0.08)	0.01 (0.08)	0.00 (0.00)
	class2	0.74 (1.26)	1.01 (1.34)	0.74 (1.31)	1.35 (1.53)
	class3	0.04 (0.09)	0.04 (0.09)	0.04 (0.09)	0.00 (0.00)
bLOOC1-Exact	class1	0.78 (0.95)	0.05 (0.10)	0.03 (0.08)	0.00 (0.00)
	class2	2.38 (1.16)	2.61 (0.83)	2.39 (1.23)	2.10 (1.41)
	class3	0.05 (0.10)	0.04 (0.09)	0.04 (0.09)	0.00 (0.00)
bLOOC2	class1	0.05 (0.13)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	class2	0.00 (0.00)	0.01 (0.06)	0.00 (0.00)	0.00 (0.00)
	class3	0.01 (0.06)	0.01 (0.06)	0.00 (0.00)	0.00 (0.00)
bLOOC2-Exact	class1	1.34 (0.66)	1.83 (0.51)	1.90 (0.60)	2.64 (0.69)
	class2	2.84 (0.26)	2.90 (0.42)	3.00 (0.00)	2.96 (0.09)
	class3	0.05 (0.13)	0.06 (0.11)	0.04 (0.09)	0.00 (0.00)

In this experiment, bLOOC1 and bLOOC1-Exact have the best performance, followed by LOOC, LOOC-Exact and bLOOC2, bLOOC2-Exact. This is again not surprising because the ridge estimator produces a bias towards a constant value times the identity matrix. This is verified by the mixing values chosen by both bLOOC1 and bLOOC1-Exact, which are closer to either the average eigenvalue times the identity or the sample covariance matrix. Since the true covariance matrices are some multiple of the identity matrix, the Euclidean distance which assumes equal identity matrix is no longer in favor.

3.5.3 Equal highly elliptical covariance matrices

In this experiment, all three classes have the same highly elliptical covariance matrix given by the diagonal matrix whose diagonal elements are: $\sigma_i = \left[9(i-1)/(p-1)+1\right]^2 \quad 1 \le i \le p$. The mean vector of the first class is the origin. The elements of the mean vector of class two are defined by $\mu_{2,i} = 2.5\sqrt{\sigma_i/p}\left[(p-i)/\left((p/2)-1\right)\right]$, and the mean vector of class three is given by $\mu_{3,i} = (-1)^i \mu_{2,i}$. The results are shown in Tables 3.6, 3.7 and Figures 3.5, 3.6.

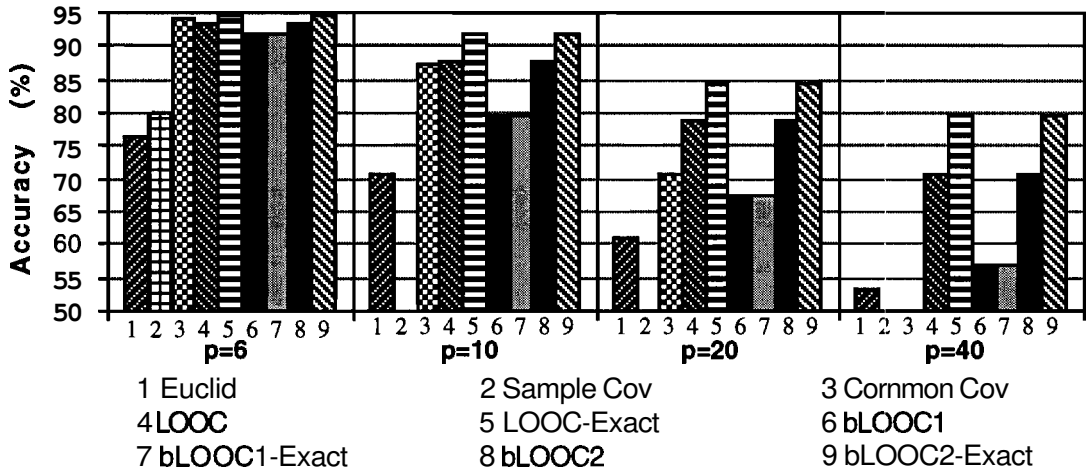


Figure *3.5* Mean Classification Accuracy for Equal Highly Elliptical Covariance Matrices (Equal Training Set Size)

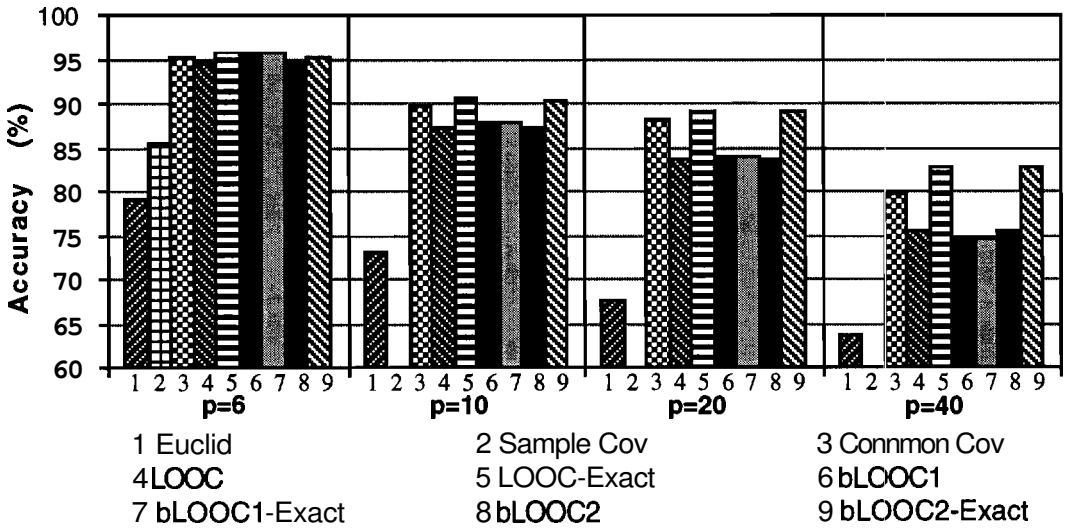


Figure *3.6* Mean Classification Accuracy for Equal Highly Elliptical Covariance Matrices (Unequal Training Set Size)

Table 3.6
Classification Results for Equal Highly Elliptical Covariance Matrices
(Equal Training Set Size)

Accuracy (%)

		p=6	p=10	p=20	p=40
	Euclid	76.16 (7.14)	70.63 (8.28)	61.07 (7.44)	53.40 (8.22)
	Sample Cov	79.88 (17.93)	N/A	N/A	N/A
	Common Cov	94.20 (2.68)	87.12 (6.30)	70.54 (10.74)	N/A
	LOOC	93.25 (6.35)	87.61 (6.77)	78.61 (9.40)	70.68 (7.82)
	LOOC-Exact	94.60 (3.88)	91.67 (3.70)	84.65 (5.50)	79.74 (5.85)
	bLOOC1	91.91 (4.98)	79.68 (9.57)	67.40 (8.54)	56.88 (8.76)
	bLOOC1-Exact	91.91 (4.98)	79.76 (9.58)	67.40 (8.54)	56.88 (8.76)
	bLOOC2	93.25 (6.35)	87.61 (6.77)	78.61 (9.40)	70.68 (7.82)
	bLOOC2-Exact	94.60 (3.88)	91.67 (3.70)	84.65 (5.50)	79.74 (5.85)

Mixing Values

bLOOC1-Exact	class1	1.96 (0.33)	2.25 (0.08)	2.41 (0.15)	2.61 (0.13)
	class2	2.02 (0.19)	2.21 (0.15)	2.44 (0.14)	2.66 (0.12)
	class3	1.92 (0.36)	2.12 (0.35)	2.48 (0.14)	2.68 (0.12)
bLOOC2	class1	0.04 (0.09)	0.03 (0.08)	0.01 (0.06)	0.00 (0.00)
	class2	0.06 (0.14)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	class3	0.16 (0.51)	0.03 (0.08)	0.00 (0.00)	0.00 (0.00)
bLOOC2-Exact	class1	2.39 (1.06)	2.88 (0.15)	2.96 (0.09)	3.00 (0.00)
	class2	2.43 (1.03)	2.89 (0.17)	2.99 (0.06)	2.99 (0.06)
	class3	2.54 (0.77)	2.63 (0.87)	2.94 (0.11)	2.99 (0.06)

Table 3.7
Classification Results for Equal Highly Elliptical Covariance Matrices
(Unequal Training Set Size)

Accuracy (%)					
		p=6	p=10	p=20	p=40
	Euclid	79.22 (4.55)	72.96 (5.92)	67.46 (4.32)	63.74 (4.67)
	Sample Cov	85.52 (6.37)	N/A	N/A	N/A
	Common Cov	95.30 (1.34)	89.82 (2.87)	88.05 (2.91)	79.64 (3.82)
	LOOC	94.40 (2.64)	87.38 (3.87)	83.52 (5.05)	75.39 (4.50)
	LOOC-Exact	95.64 (1.12)	90.68 (2.96)	89.20 (3.58)	82.67 (4.29)
	bLOOC1	95.63 (1.29)	87.88 (3.27)	83.79 (2.87)	74.50 (5.06)
	bLOOC1-Exact	95.63 (1.29)	87.88 (3.27)	83.86 (2.87)	74.50 (5.06)
	bLOOC2	94.40 (2.64)	87.38 (3.87)	83.52 (5.05)	75.39 (4.50)
	bLOOC2-Exact	95.30 (1.54)	90.32 (2.16)	89.19 (3.58)	82.67 (4.29)
Mixing Values					
LOOC	class1	0.03 (0.11)	0.00 (0.00)	0.01 (0.06)	0.00 (0.00)
	class2	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	class3	0.01 (0.06)	0.05 (0.10)	0.00 (0.00)	0.00 (0.00)
LOOC-Exact	class1	2.81 (0.33)	2.79 (0.32)	2.88 (0.17)	2.96 (0.09)
	class2	2.16 (1.29)	2.78 (0.67)	2.40 (1.20)	3.00 (0.00)
	class3	0.38 (0.91)	0.13 (0.15)	0.03 (0.08)	0.00 (0.00)
bLOOC1	class1	1.90 (0.11)	1.96 (0.15)	1.95 (0.22)	2.25 (0.00)
	class2	1.75 (0.00)	1.75 (0.00)	1.75 (0.00)	1.75 (0.00)
	class3	1.70 (0.11)	1.74 (0.01)	1.75 (0.00)	1.75 (0.00)
bLOOC1-Exact	class1	1.90 (0.11)	1.96 (0.05)	1.95 (0.30)	2.25 (0.00)
	class2	1.93 (0.06)	1.75 (0.01)	1.75 (0.00)	1.75 (0.00)
	class3	1.83 (0.11)	1.74 (0.01)	1.75 (0.09)	1.75 (0.00)
bLOOC2	class1	0.03 (0.11)	0.00 (0.00)	0.01 (0.06)	0.00 (0.00)
	class2	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
	class3	0.01 (0.06)	0.05 (0.10)	0.00 (0.00)	0.00 (0.00)
bLOOC2-Exact	class1	2.58 (0.31)	2.67 (0.22)	2.82 (0.27)	2.96 (0.09)
	class2	2.25 (1.11)	2.58 (0.47)	2.32 (0.61)	3.00 (0.00)
	class3	0.38 (0.91)	0.13 (0.15)	0.03 (0.08)	0.00 (0.00)

Since the true covariance matrices are highly elliptical, the estimators LOOC, LOOC-Exact, bLOOC2 and bLOOC2-Exact out-perform the others. However, for the unequal number of training samples per class, the performance of bLOOC1 and bLOOC1-Exact has increased substantially. The mixing values indicate that the weighted pooled covariance estimate is favored. This shows the benefit of using the Bayesian formulation when the training set size reflects the true priors. Again, bLOOC1 and bLOOC1-Exact produce similar results showing the validity of the approximation.

3.5.4 Unequal highly elliptical covariance matrices

In this experiment, the mean vectors of all classes are at the origin but the class covariance matrices are highly elliptical and vary for all classes. The diagonal elements of the covariance matrices for each class are as follows: $\sigma_{1,i} = \left[p(i-1)/(p-1)+1\right]^2 \quad 1 \le i \le p; \quad \sigma_{2,i} = \left[9(p-i)/(p-1)+1\right]^2 \quad 1 \le i \le p$ and $\sigma_{3,i} = \left\{9\left[i-(p-1)/2\right]/(p-1)\right\}^2 \quad 1 \le i \le p$. The results are summarized in Tables 3.8, 3.9 and Figures 3.7, 3.8.

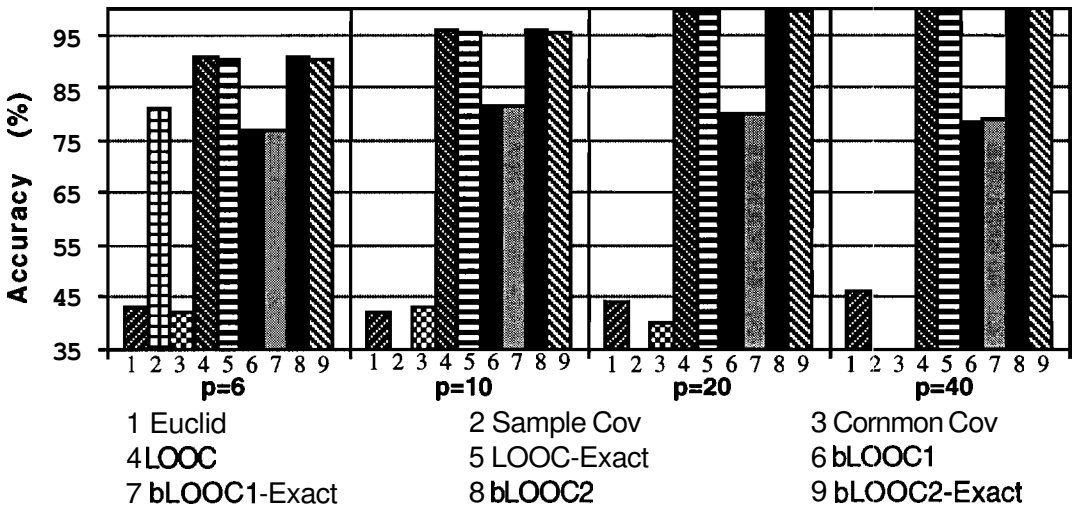


Figure 3.7 Mean Classification Accuracy for Unequal Highly Elliptical Covariance Matrices (Equal Training Set Size)

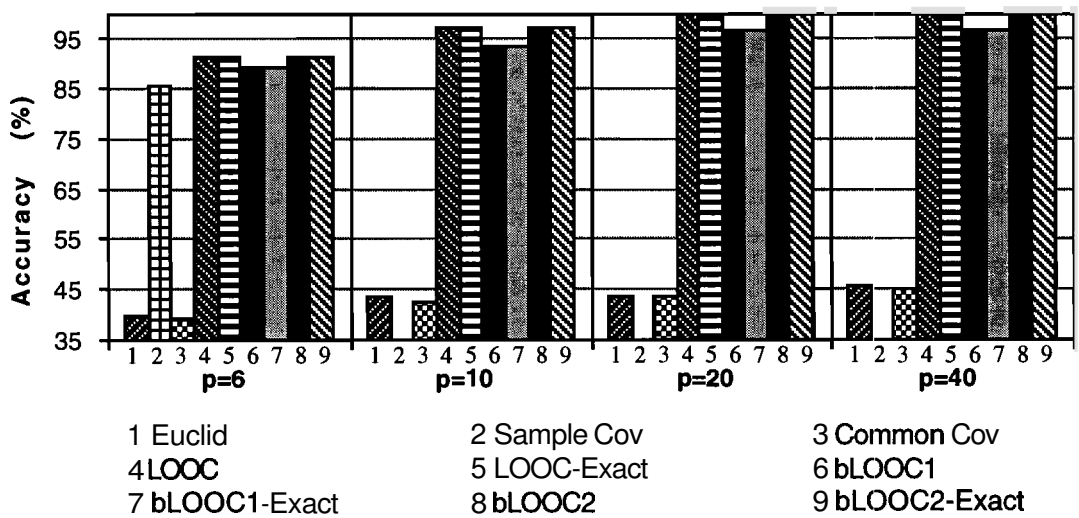


Figure 3.8 Mean Classification Accuracy for Unequal Highly Elliptical Covariance Matrices (Unequal Training Set Size)

Table 3.8
Classification Results for Unequal Highly Elliptical Covariance Matrices
(Equal Training Set Size)

Accuracy (%)

		p=6	p=10	p=20	p=40
	Euclid	43.03 (7.68)	42.08 (6.85)	44.24 (9.12)	46.11 (5.37)
	Sample Cov	80.89 (9.25)	N/A	N/A	N/A
	Common Cov	42.00 (7.96)	43.03 (8.39)	39.89 (10.55)	N/A
	LOOC	90.77 (4.03)	95.69 (2.69)	99.42 (0.77)	99.97 (0.09)
	LOOC-Exact	90.41 (4.22)	95.57 (2.64)	99.39 (0.79)	99.97 (0.09)
	bLOOC1	76.89 (9.01)	81.50 (8.48)	80.05 (11.24)	78.68 (11.57)
	bLOOC1-Exact	77.11 (8.95)	81.39 (8.53)	79.97 (11.14)	79.11 (11.44)
	bLOOC2	90.77 (4.03)	95.69 (2.69)	99.42 (0.77)	99.97 (0.09)
	bLOOC2-Exact	90.41 (4.22)	95.57 (2.64)	99.39 (0.79)	99.97 (0.09)
Mixing Values					
LOOC	class1	0.00 (0.00)	0.01 (0.06)	0.00 (0.00)	0.00 (0.00)
	class2	0.06 (0.14)	0.04 (0.12)	0.00 (0.00)	0.00 (0.00)
	class3	0.01 (0.06)	0.01 (0.06)	0.00 (0.00)	0.00 (0.00)
LOOC-Exact	class1	0.05 (0.10)	0.04 (0.09)	0.00 (0.00)	0.00 (0.00)
	class2	0.15 (0.22)	0.09 (0.15)	0.01 (0.06)	0.00 (0.00)
	class3	0.13 (0.30)	0.06 (0.11)	0.03 (0.08)	0.00 (0.00)
bLOOC1	class1	0.58 (0.18)	0.46 (0.19)	0.23 (0.14)	0.15 (0.13)
	class2	0.59 (0.20)	0.44 (0.18)	0.20 (0.15)	0.14 (0.13)
	class3	0.76 (0.35)	0.64 (0.25)	0.26 (0.13)	0.10 (0.13)
bLOOC1-Exact	class1	0.60 (0.17)	0.48 (0.20)	0.23 (0.14)	0.16 (0.12)
	class2	0.89 (0.73)	0.43 (0.18)	0.19 (0.14)	0.14 (0.13)
	class3	0.86 (0.38)	0.64 (0.25)	0.26 (0.13)	0.10 (0.13)
bLOOC2	class1	0.00 (0.00)	0.01 (0.06)	0.00 (0.00)	0.00 (0.00)
	class2	0.06 (0.14)	0.04 (0.12)	0.00 (0.00)	0.00 (0.00)
	class3	0.01 (0.06)	0.01 (0.06)	0.00 (0.00)	0.00 (0.00)
bLOOC2-Exact	class1	0.05 (0.10)	0.04 (0.09)	0.00 (0.00)	0.00 (0.00)
	class2	0.15 (0.22)	0.09 (0.15)	0.01 (0.06)	0.00 (0.00)
	class3	0.13 (0.30)	0.06 (0.11)	0.03 (0.08)	0.00 (0.00)

Table **3.9**
Classification Results for Unequal Highly Elliptical Covariance Matrices
(Unequal Training Set Size)

Accuracy (%)					
		p=6	p=10	p=20	p=40
	Euclid	39.97 (6.57)	43.53 (6.04)	43.38 (4.98)	45.91 (3.78)
	Sample Cov	85.38 (5.25)	N/A	N/A	N/A
	Common Cov	39.26 (6.61)	42.46 (6.99)	43.26 (4.94)	45.07 (4.68)
	LOOC	91.53 (1.76)	97.42 (1.11)	99.88 (0.16)	100 (0.00)
	LOOC-Exact	91.44 (2.05)	97.38 (1.18)	99.88 (0.16)	100 (0.00)
	bLOOC1	89.08 (2.60)	93.59 (3.25)	96.99 (2.13)	96.78 (2.51)
	bLOOC1-Exact	89.26 (2.52)	93.53 (3.21)	96.99 (2.13)	96.78 (2.51)
	bLOOC2	91.53 (1.76)	97.42 (1.11)	99.88 (0.16)	100 (0.00)
	bLOOC2-Exact	91.44 (2.05)	97.38 (1.18)	99.88 (0.16)	100 (0.00)
Mixing Values					
LOOC	class1	0.01 (0.06)	0.01 (0.06)	0.01 (0.06)	0.00 (0.00)
	class2	0.01 (0.06)	0.03 (0.08)	0.00 (0.00)	0.00 (0.00)
	class3	0.01 (0.06)	0.01 (0.06)	0.00 (0.00)	0.00 (0.00)
LOOC-Exact	class1	0.24 (0.67)	0.08 (0.12)	0.04 (0.09)	0.00 (0.00)
	class2	0.10 (0.15)	0.08 (0.12)	0.00 (0.00)	0.00 (0.00)
	class3	0.05 (0.10)	0.05 (0.10)	0.04 (0.09)	0.00 (0.00)
bLOOC1	class1	0.68 (0.59)	0.36 (0.22)	0.18 (0.14)	0.11 (0.13)
	class2	1.00 (0.00)	0.98 (0.14)	0.75 (0.00)	0.05 (0.00)
	class3	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
bLOOC1-Exact	class1	0.84 (0.77)	0.95 (1.06)	0.18 (0.14)	0.11 (0.13)
	class2	1.00 (0.00)	0.98 (0.14)	0.75 (0.00)	0.05 (0.00)
	class3	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
bLOOC2	class1	0.01 (0.06)	0.01 (0.06)	0.01 (0.06)	0.00 (0.00)
	class2	0.01 (0.06)	0.03 (0.08)	0.00 (0.00)	0.00 (0.00)
	class3	0.01 (0.06)	0.01 (0.06)	0.00 (0.00)	0.00 (0.00)
bLOOC2-Exact	class1	0.24 (0.67)	0.08 (0.12)	0.04 (0.09)	0.00 (0.00)
	class2	0.10 (0.15)	0.08 (0.12)	0.00 (0.00)	0.00 (0.00)
	class3	0.05 (0.10)	0.05 (0.10)	0.04 (0.09)	0.00 (0.00)

As expected, LOOC, LOOC-Exact, bLOOC and bLOOC-Exact have the best results because the class covariance matrices are diagonal and vary differently from class to class. The mixing values selected are close to zero which is appropriate since the mixture is essentially the diagonal sample covariance matrix. Again, bLOOC1 and bLOOC1-Exact have similar performance, which indicates that the approximation holds.

3.6 Experiment using a Small Segment of AVIRIS Data

In this section, the estimators are tested using hyperspectral data which consists of a small segment of the AVIRIS data of NW Indiana's Indian Pine test site obtained in June 1992. Out of the original 220 spectral channels, 20 channels (104-108, 150-163, 220) frorn the water absorption bands are discarded. Therefore, the test data consists of 200 spectral features and four classes, namely, corn-notill, soybean-notill, soybean-min and grass. The test image and the ground truth map are shown in Figure 3.9. The training samples are chosen to be 1%, 4%, 10% and 20% of the labeled samples. The number of labeled and training samples in each class is shown in Table 3.10 and Table 3.11, respectively. Since these training samples are randomly selected, each experiment is repeated 10 times and the mean and standard deviation of the classification accuracy is obtained.

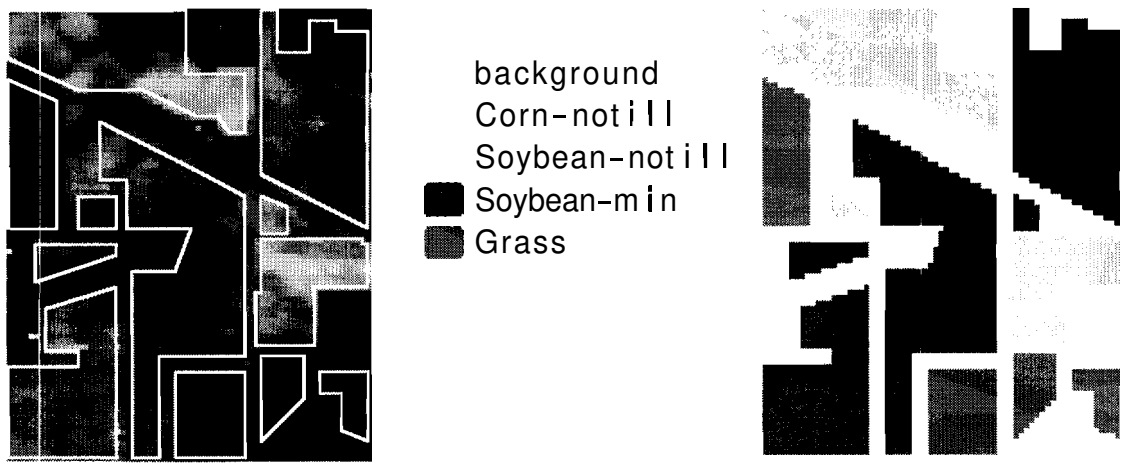


Figure 3.9 Portion of AVIRIS data and Ground Truth Map (Original in Color)

Table 3.10
Class Description for AVIRIS Data in Figure 3.9

Class Names	No. of Labeled Samples
Corn-notill	910
Soybean-notill	638
Soybean-min	1421
Grass	618

Table 3.11
Number of Training Samples for Experiment 3.6

	1%	5%	8%	10%	20%	40%
Corn-notill	9	45	72	91	182	364
Soybean-notill	6	31	51	63	127	255
Soybean-min	14	71	113	142	284	586
Grass	6	30	49	61	123	247
Total Samples	35	177	285	357	716	1452

The previous results from simulation data indicate that the estimators bLOOC1 and bLOOC1-Exact would produce similar results. The simulation also shows that LOOC-Exact outperforms LOOC in cases when the training set size is small. However, LOOC-Exact and bLOOC2-Exact require considerably more computation for 200 spectral channels. In view of these results, the estimators LOOC-Exact, bLOOC1-Exact and bLOOC2-Exact are not considered in the following experiments. In the analysis of hyperspectral data, feature extraction is often employed to reduce dimensionality. Hence, discriminant analysis feature extraction (DAFE) is incorporated in this experiment to demonstrate the effect of covariance estimators on the classification process. Since there are four classes, the number of features are reduced to three using DAFE. The results of the experiments are shown in Tables 3.12, 3.13 and Figures 3.10, 3.11.

Table 3.12
Classification Results for Small AVIRIS Image (Part 1)

Accuracy (%)		**No. of Training Samples**		
		1%	**5%**	**8%**
	Euclid	66.00 (8.70)	67.75 (3.29)	67.23 (2.04)
	Sample Cov	N/A	N/A	N/A
	Sample Cov+DAFE	N/A	N/A	71.59 (4.33)
	Common Cov	N/A	N/A	78.40 (3.33)
	Common-Cov+DAFE	N/A	N/A	78.40 (3.33)
	LOOC	72.55 (13.60)	54.54 (6.43)	51.50 (2.08)
	LOOC+DAFE	82.28 (8.98)	91.62 (2.45)	92.02 (2.12)
	bLOOC1	61.68 (18.16)	51.38 (8.64)	73.61 (3.97)
	bLOOC1+DAFE	84.45 (7.82)	92.39 (1.68)	78.56 (3.34)
	bLOOC2	72.55 (13.60)	54.54 (6.43)	51.50 (2.08)
	bLOOC2+DAFE	82.28 (8.98)	91.62 (2.45)	92.02 (2.12)
Mixing Values				
LOOC	Corn-notill	2.10 (0.47)	2.25 (0.00)	2.25 (0.00)
	Soybean-notill	1.80 (0.72)	0.75 (0.00)	0.75 (0.00)
	Soybean-min	0.90 (0.47)	0.75 (0.00)	0.75 (0.00)
	Grass	1.58 (0.87)	0.75 (0.00)	0.75 (0.00)
bLOOC1	Corn-notill	2.10 (0.47)	2.25 (0.00)	2.00 (0.00)
	Soybean-notill	1.20 (0.72)	0.75 (0.00)	2.00 (0.00)
	Soybean-min	0.75 (0.00)	0.75 (0.00)	1.75 (0.00)
	Grass	1.05 (0.63)	0.75 (0.00)	2.00 (0.00)
bLOOC2	Corn-notill	2.10 (0.47)	2.25 (0.00)	2.25 (0.00)
	Soybean-notill	1.80 (0.72)	0.75 (0.00)	0.75 (0.00)
	Soybean-min	0.90 (0.47)	0.75 (0.00)	0.75 (0.00)
	Grass	1.58 (0.87)	0.75 (0.00)	0.75 (0.00)

Table 3.13
ClassificationResults for Small AVIRIS Image (Part 2)

Accuracy (%)		**No. of Training Samples**		
		10%	20%	40%
	Euclid	66.72 (2.50)	67.43 (2.16)	67.16 (1.94)
	Sample Cov	NIA	N/A	53.85 (1.79)
	Sample Cov+DAFE	82.31 (2.51)	92.37 (0.95)	95.82 (0.80)
	Common Cov	86.16 (2.25)	93.06 (0.85)	95.72 (0.70)
	Common-Cov+DAFE	86.16 (2.25)	93.06 (0.85)	95.72 (0.70)
	LOOC	78.41 (3.06)	90.75 (1.30)	95.79 (0.78)
	LOOC+DAFE	86.35 (2.18)	93.20 (0.92)	96.00 (0.67)
	bLOOC1	80.13 (3.38)	90.97 (1.41)	96.05 (0.98)
	bLOOC1+DAFE	86.44 (2.15)	93.50 (0.88)	96.07 (0.69)
	bLOOC2	80.13 (3.38)	90.97 (1.41)	96.05 (0.98)
	bLOOC2+DAFE	86.44 (2.15)	93.50 (0.88)	96.07 (0.69)
Mixing Values				
LOOC	Corn-notill	2.00 (0.00)	2.00 (0.00)	1.75 (0.00)
	Soybean-notill	2.00 (0.00)	2.00 (0.00)	2.00 (0.00)
	Soybean-min	1.75 (0.00)	1.75 (0.00)	1.75 (0.00)
	Grass	2.00 (0.00)	2.00 (0.00)	2.00 (0.00)
bLOOC1	Corn-notill	1.98 (0.08)	1.75 (0.00)	1.75 (0.00)
	Soybean-notill	2.00 (0.00)	2.00 (0.00)	1.95 (0.11)
	Soybean-min	1.75 (0.00)	1.75 (0.00)	1.75 (0.00)
	Grass	2.00 (0.00)	2.00 (0.00)	1.75 (0.00)
bLOOC2	Corn-notill	1.98 (0.08)	1.75 (0.00)	1.75 (0.00)
	Soybean-notill	2.00 (0.00)	2.00 (0.00)	1.95 (0.11)
	Soybean-min	1.75 (0.00)	1.75 (0.00)	1.75 (0.00)
	Grass	2.00 (0.00)	2.00 (0.00)	1.75 (0.00)

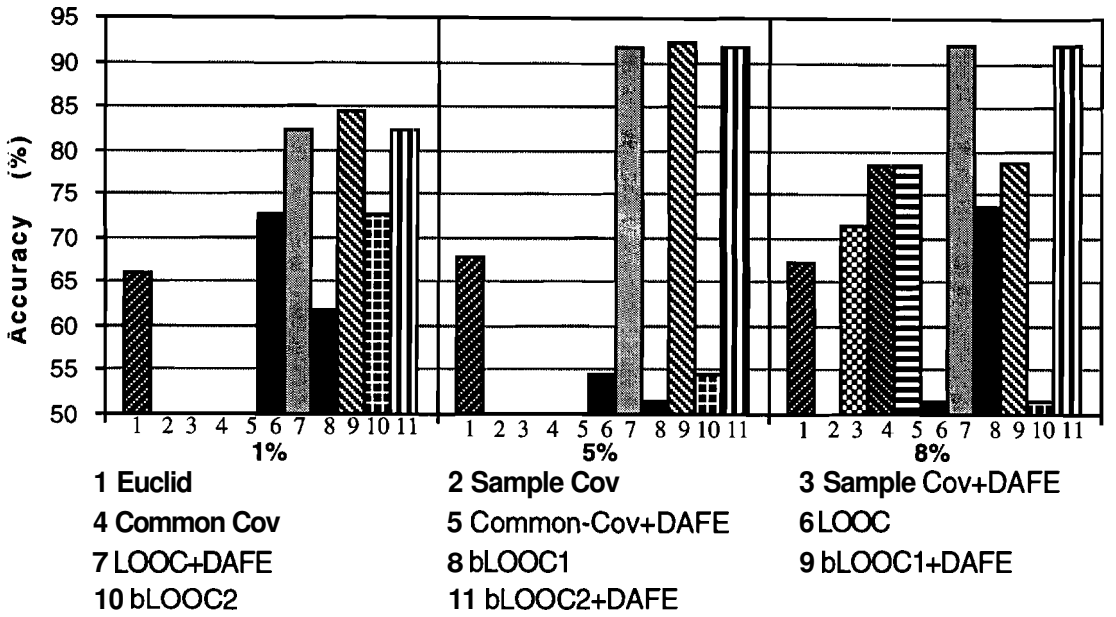


Figure 3.10 Mean Classification Accuracy using Small AVIRIS Image (Part 1)

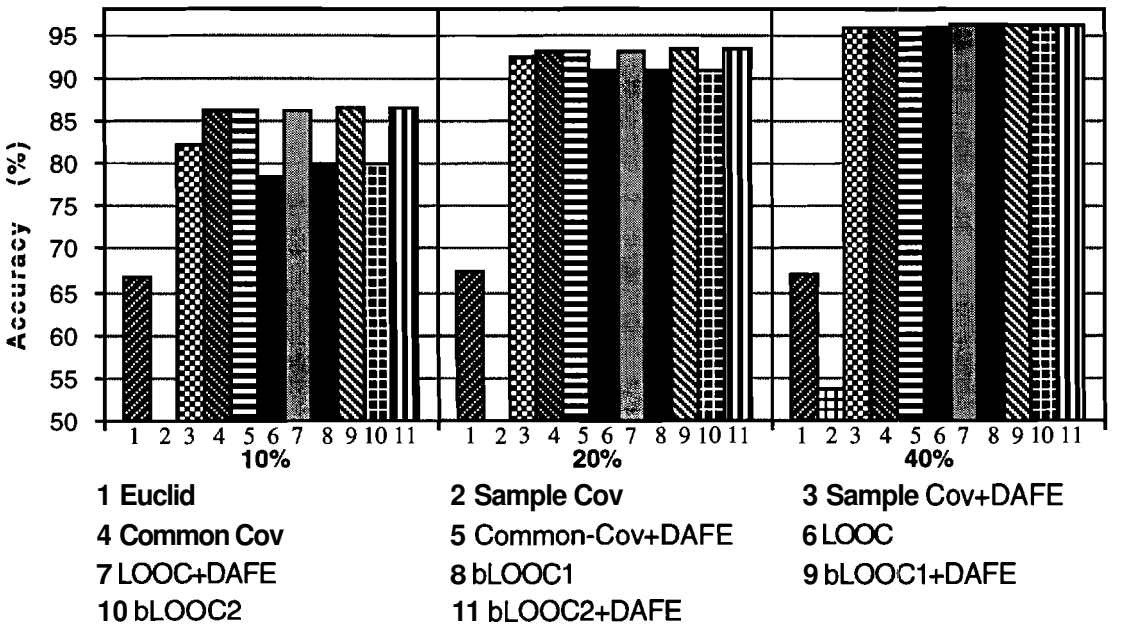


Figure 3.11 Mean Classification Accuracy using Small AVIRIS Image (Part 2)

The results show that the covariance estimation combined with DAFE increases the classification performance substantially even when the training samples are limited. When the number of training samples are selected to be 1% and 5% of the labeled samples, the total number of training samples is less than dimensionality. In this case, the best performance is achieved by bLOOCl together with **DAFE.** This shows that the ridge estimator gives rise to a better pooled covariance estimate by counteracting the upward bias of large eigenvalues and downward bias of smallest eigenvalues when the training set size is less than dimensionality. On the other hand, when the total number of training samples is more than dimensionality, bLOOC2 combined with **DAFE** gives the best performance. The result suggests that the true covariance matrices are elliptical. This can be verified by the poor performance of bLOOCl at 8% of the labeled samples. In this case, the total number of training samples is 285 which suggests that the pooled coviiriance estimate is highly variable. When bLOOCl is used, the chosen mixing values indicate that the mixture of partially pooled covariance matrix is favored over the ridge estimator, which has been shown to perform poorly for elliptical covariance matrices. On the other hand, LOOC and bLOOC2 which use the diagonal covariance matrices perform significantly better. In conclusion, it is suggested that when $N < (p+1)$, bLOOCl and DAFE can lead to better performance and when $N \geq (p+1)$, bLOOC2 and **DAFE** should be used instead.

3.7 Experiment using a Large Segment of AVIRIS data

In this experiment, a large segment of AVIRIS data is used. Again, the water absorption bands have been discarded, leaving a total of 200 channels. This data contains many classes of varying sizes. The purpose of this experiment is to demonstrate the effect of covariance estimation on classes with varying covariance structures and different training sample size. The training samples are selected in proportion to the number of labeled samples for each class. The labeled samples, excluding the training samples are then used as test samples. The classes, the numbers of labeled samples and training samples are listed in Table 3.14. The image and ground truth cnap are shown in Figures 3.12.

Table 3.14
Class Description for AVIRIS Data in Figure 3.12

Class Names	No. of Labeled Samples	No. of Training Samples
1. Corn-notill	1423	286
2. Corn-min	834	166
3. Corn	234	46
4. Soybeans-notill	797	159
5. Soybeans-notill2	171	34
6. Soybeans-min	2468	493
7. Soybeans-clean	614	122
8. Alfalfa	54	10
9. Grass/Pasture	497	99
10. Grass/Trees	747	149
11. Grass/pasture-mowed	26	5
12. Hay-windrowed	489	97
13. Oats	20	4
14. Wheat	212	42
15. Woods	1294	258
16. Bldg-Grass-Tree-Drives	380	76
17. Stone-steel towers	95	19
Total samples	10355	2065

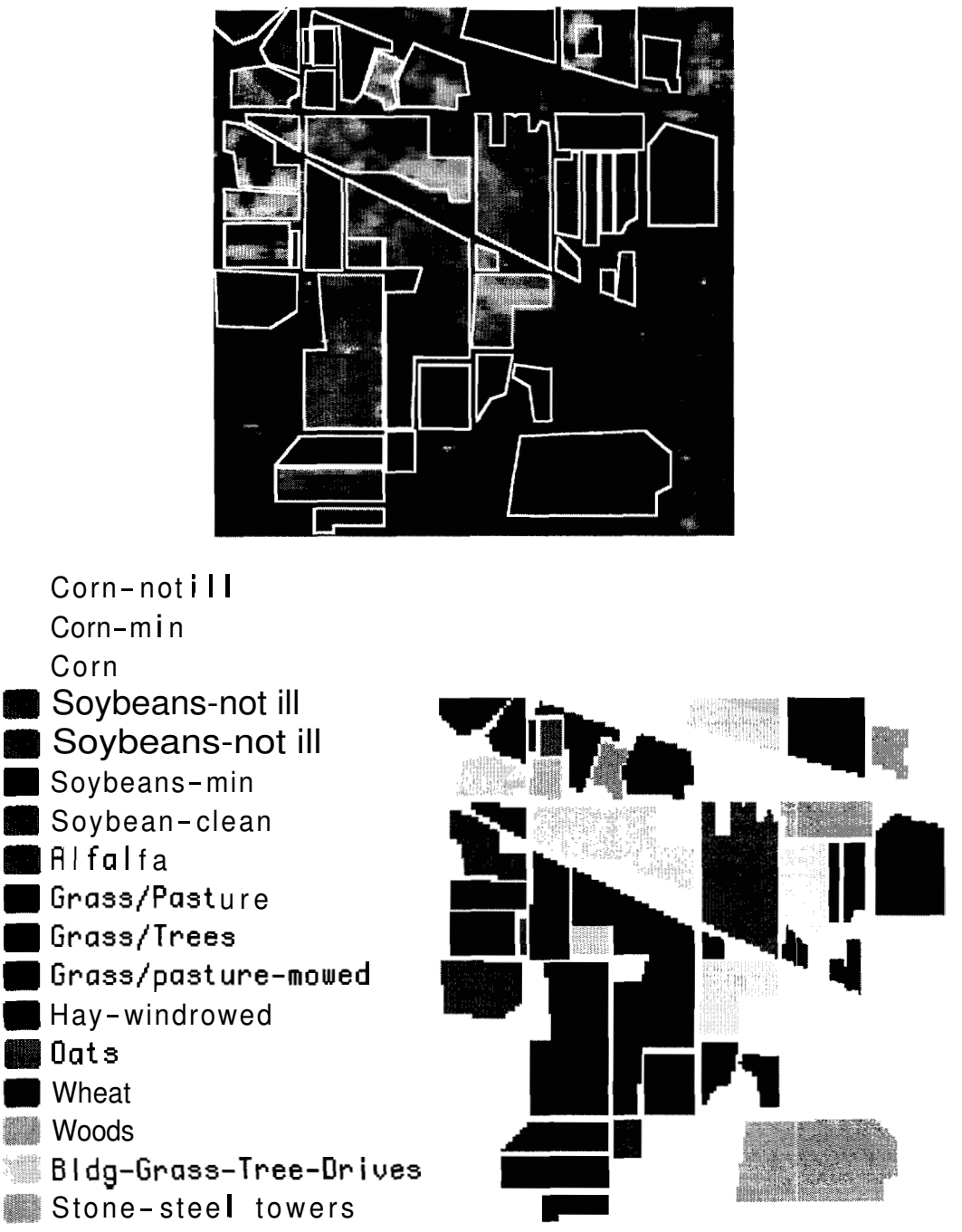


Figure 3.12 Large AVIRIS Data and Ground Truth Map (Original in Color)

The classification procedures for testing the data are shown in Table 3.15. Since the Euclidean distance classifier does not utilize the covariance information, its performance would indicate whether the second order statistics is useful for the classification of high dimensional data with limited training samples. The use of common covariance estimate for all classes is equivalent to a linear classifier. The

sample covariance estimate is not tested in this experiment since the numbers of training samples for some classes are extremely small. Even with feature extraction, only a handful of extracted features can be used to obtain non-singular covariance estimates. After performing covariance estimation, two types of classifiers, namely, the quadratic classifier (QC) and the contextual classifier ECHO (Extraction and Classification of Homogeneous Objects) [33] are then applied and compared. While the quadratic classifier assign individual pixels to one of the classes, the ECHO classifier first divides the image into groups of contiguous pixels and classifies each group to one of the classes. In other words, ECHO uses both the spatial and spectral information. The results of classification are shown in Table 3.16 and Figure 3.13, and the mixing values for each covariance estimator are listed in Table 3.17. This data was obtained in June 1992 so most of the row crops in the agricultural portion of the test site had not reached their maximum ground cover. Therefore, the classification of these crops becomes challenging since the spectral information comes from a mixture of the crops, the variations in the soil type, soil moisture, and previous crop residues. These crops are listed as the first seven classes and their mean classification accuracy is computed separately and shown in the bottom row of Table 3.16.

Table 3.15
Classification Procedures for Experiment 3.7

Notation	Procedures
C1	Euclidean Distance Classifier
C2	Common Cov+DAFE+QC
C3	Common Cov+DAFE+ECHO
C4	LOOC+DAFE+QC
C5	LOOC+DAFE+ECHO
C6	bLOOC1+DAFE+QC
C7	bLOOC1+DAFE+ECHO
C8	bLOOC2+DAFE+QC
C9	bLOOC2+DAFE+ECHO

Table 3.16
Classification Results for Experiment 3.7

Class Names	C1	C2	C3	C4	C5	C6	C7	C8	C9
1. Corn-notill	55.40	70.64	75.26	71.43	77.00	74.22	82.93	74.22	82.93
2. Corn-min	16.02	61.68	61.98	65.27	79.94	67.96	90.12	67.96	90.12
3. Corn	13.30	66.49	69.68	65.43	76.06	67.55	86.70	67.55	86.70
4. Soybeans-notill	59.40	76.02	89.18	77.12	93.89	80.09	94.83	80.09	94.83
5. Soybeans-notill2	56.20	78.83	83.94	67.88	78.83	70.07	87.59	70.07	87.59
6. Soybeans-min	20.15	54.28	58.73	67.54	86.73	62.84	86.73	62.84	86.73
7. Soybeans-clean	2.03	83.33	85.77	80.28	86.59	85.57	94.51	85.57	94.51
8. Alfalfa	81.82	61.36	61.36	61.36	61.36	54.55	54.55	54.55	54.55
9. Grass/Pasture	2.51	81.16	81.16	90.70	91.96	91.21	92.96	91.21	92.96
10. Grass/Trees	24.25	95.99	96.15	96.99	97.16	96.66	96.82	96.66	96.82
11. Grass/pasture-mowed	95.24	47.62	47.62	47.62	47.62	33.33	33.33	33.33	33.33
12. Hay-windrowed	51.79	98.72	98.72	99.23	99.23	99.49	99.49	99.49	99.49
13. Oats	43.75	31.25	31.25	31.25	31.25	31.25	31.25	31.25	31.25
14. Wheat	92.35	100	100	100	100	100	100	100	100
15. Woods	85.04	87.16	87.16	92.66	93.24	89.86	92.08	89.86	92.08
16. Bldg-Grass-Tree-Drives	27.30	82.57	82.57	70.39	70.39	84.54	90.79	84.54	90.79
17. Stone-steel towers	93.42	94.74	94.74	94.74	94.74	94.74	94.74	94.74	94.74
Average Accuracy 1-17	48.23	74.81	76.78	75.29	80.35	75.53	82.91	75.53	82.91
Average Accuracy 1-7	31.79	70.18	74.94	70.71	82.72	72.61	89.06	72.61	89.06

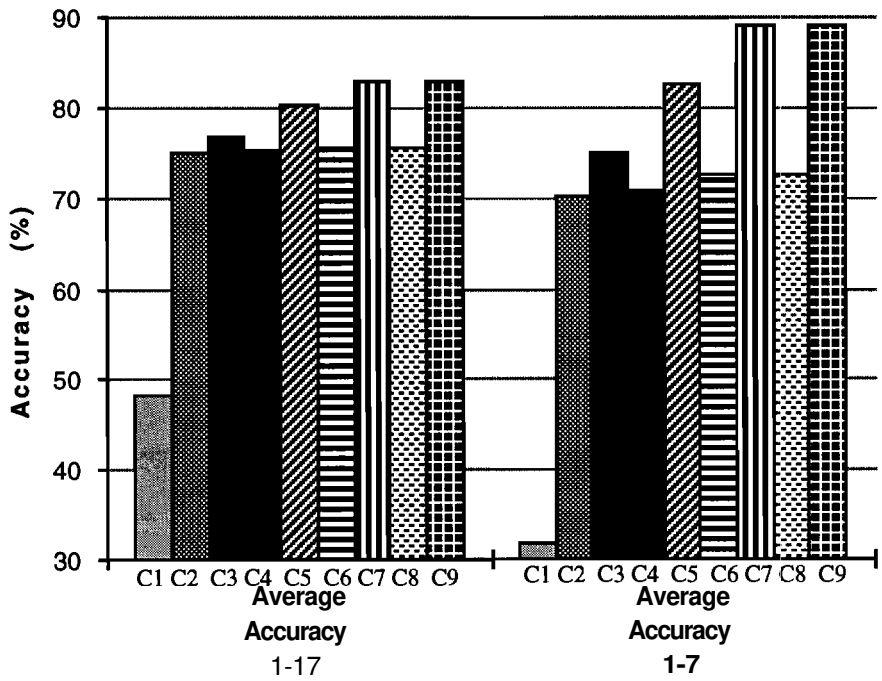


Figure 3.13 Mean Classification Accuracy for Experiment 3.7

Table 3.17
Mixing Values for Experiment 3.7

Class Names	LOOC	bLOOC1	bLOOC2
Corn-notill	1.75	1.75	1.75
Corn-min	1.75	1.75	1.75
Corn	2.00	1.75	1.75
Soybeans-notill	1.75	1.75	1.75
Soybeans-notill2	2.00	1.75	1.75
Soybeans-min	1.50	1.75	1.75
Soybeans-clean	2.00	1.75	1.75
Alfalfa	2.00	2.00	2.00
Grass/Pasture	1.75	1.75	1.75
Grass/Trees	1.75	1.75	1.75
Grass/pasture-mowed	2.00	2.00	2.00
Hay-windrowed	2.00	1.75	1.75
Oats	2.00	2.00	2.00
Wheat	2.00	1.75	1.75
Woods	1.75	1.75	1.75
Bldg-Grass-Tree-Drives	2.00	1.75	1.75
Stone-steel towers	2.00	1.75	1.75

The performance of the Euclidean distance classifier is significantly lower than the other classifiers. This shows that the second order statistics are useful for classifying high dimensional data even though the training samples are limited. Although the class covariance matrices differ substantially, the use of common covariance matrix and hence the linear classifier improves the performance substantially compared to the Euclidean distance classifier. Since the mixing values for bLOOC1 and bLOOC2 fall within the range of $1 \leq \alpha_i \leq 2$, these two estimators use the same covariance mixture and hence the classification results are the same as expected. The figure shows that the best performance is achieved by C7 and C9 where bLOOC1 and bLOOC2, followed by DAFE and the ECHO classifier are used. Many classes have mixing values of 1.75 which implies that the weighted pooled covariance mixture is favored. The classification accuracy increases substantially for the row crops 1-7. Compared with the second best result obtained from the classifier LOOC+DAFE+ECHO (C5), the accuracy increases frorn 82.72% to 89.06%. The mean accuracy for all classes improves from 80.35% to 82.90% as well. Therefore, the use of Bayesian estimators is beneficial when the sample sizes are unequal and the training set size reflects the true priors. The classification maps for C5 and C7 are shown in Figures 3.15 and 3.15, respectively.

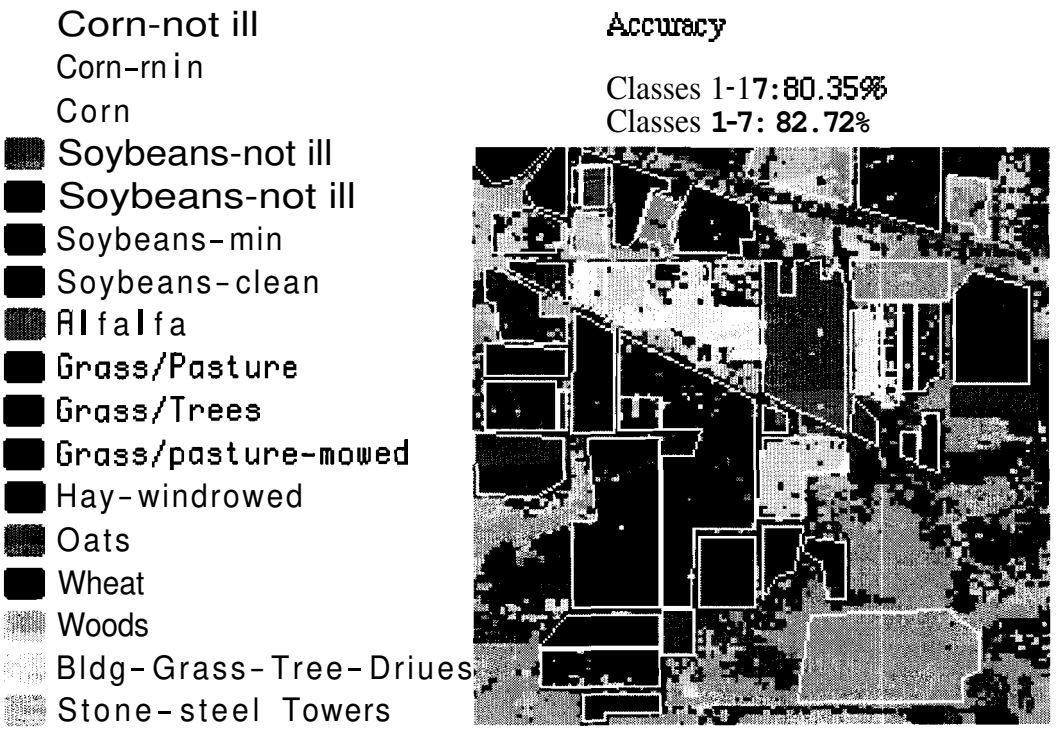


Figure 3.14 Classification Map for LOOC+DAFE+ECHO (Original in Color)

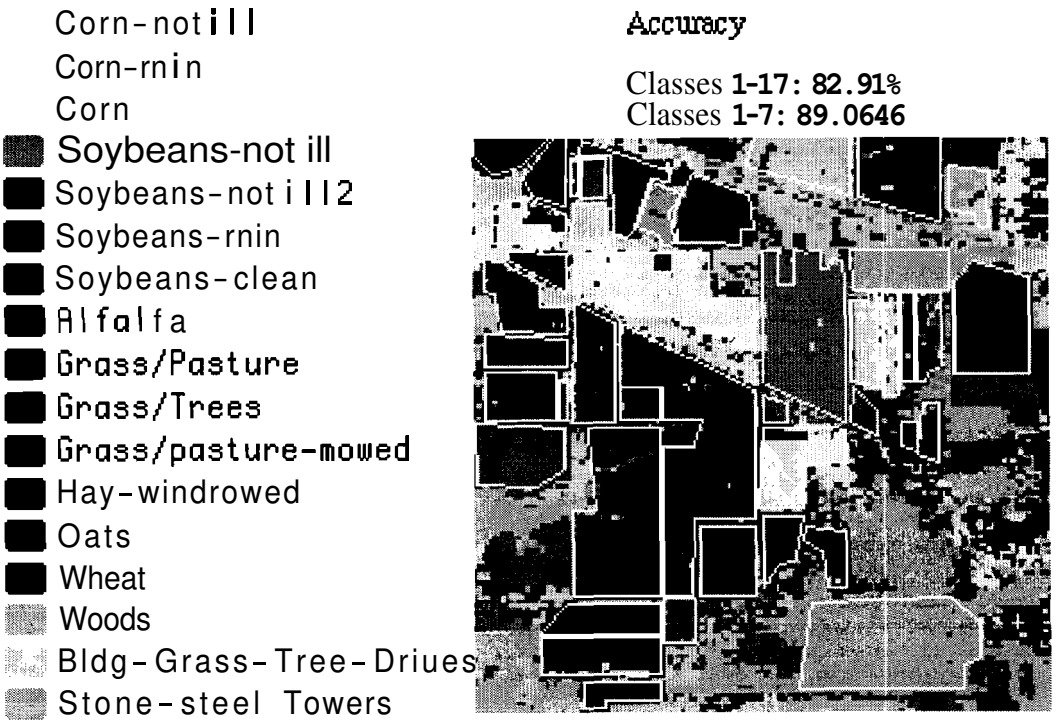


Figure 3.15 Classification Map for bLOOC1+DAFE+ECHO (Original in Color)

3.8 Summary

Two covariance estimators for limited training samples have been proposed in this work. These estimators can be viewed as an intermediate approach between the linear and quadratic classifiers. The estimators were derived under a Bayesian setting, which is advantageous when the classes have different sizes and the training set size is proportional to the sample size of each class. It was shown that the first estimator bLOOC1 combined with discriminant analysis feature extraction (DAFE) can achieve better performance when the total number of training samples is less than the dimensionality. On the other hand, when the pooled covariance matrix is non-singular, the other estimator bLOOC2 should be used. Under these conditions, the proposed estimators perform better than the leave-one-out covariance (LOOC) estimator, the linear and quadratic classifiers.

Since the leave-one-out likelihood is used as the criterion for these estimators, it has the drawback of not being directly related to class separability, and :subsequently the classification accuracy. Therefore, some smooth loss function derived from the class separability is recommended for future work. Also, since decision boundary feature extraction (DBFE) is not suitable for small training sample size and DAFE does not work well when the classes have similar mean values, an alternative feature extraction or classification methods need to be explored. The issue of feature extraction will be further studied in the next chapter.

CHAPTER 4: A BINARY TREE DESIGN FOR CLASSIFICATION AND FEATURE EXTRACTION

4.1 Introduction

Decision tree classifiers belong to a type of hierarchical classifiers in which subsets of classes are processed at multiple stages. Hierarchical classifiers have been known to overcome some of the limitations of single-stage classifiers. For example, in a single-stage classification system, the decision rule and feature reduction method are obtained by optimizing a criterion based on all classes available. Therefore, the decision boundary and the features extracted may not be optimal to discriminate among all classes. This limitation becomes more severe when classes are numerous and training set size is small due to the Hughes phenomenon [34]. A large number of classes generates more complex decision boundaries and hence requires more features to distinguish arnong them. The Hughes phenomenon indicates that in the case of limited training set size, the classification performance deteriorates when more features are added. Consequently, the advent of new hyperspectral sensors such as AVIRIS which generates 224 dimensional data presents new challenges. While the increased dimensionality enables more classes to be identified, training samples still remain relatively scarce and hard to find. As a result, the Hughes phenomenon becomes an immediate concern for a single-stage classifier. Decision tree classifiers offer a solution to circumvent these problems by focusing on fewer classes and obtaining different features and decision rules at each stage.

The decision tree classifiers have been extensively studied and applied in recent years [35]. In a binary decision tree, each node considers two subgroups of classes at a time. However, decision tree classifiers are not without their own limitations. Notably, the design of a tree classifier is complex. An optimal decision tree has to consider many factors such as the tree structure, feature reduction method and computational complexity

at the same time. Many tree design approaches have been proposed, targeting different design aspects or applications.

In this chapter, a hybrid design of a binary decision tree is proposed which considers two classes at each node, instead of two subgroups of classes. In this manner, the problem of merging classes into two nodes can be avoided. However, by processing two classes at a time, there is a tendency to generate a large tree. The bottom-up approach in the proposed hybrid design helps reduce the size by discriminating classes with the largest statistical distance near the root. The tree design also incorporates two types of feature extraction methods one of which is based on the decision boundary and the other based on optimizing the Bhattacharyya distance between two classes. The proposed method is implemented for the supervised classification of multispectral data. In addition to functioning as a classifier, the binary tree design can be used to extract the best features between pairs of classes. The features obtained from pair-wise discrimination can then be combined and used as the feature subset for a single-stage classifier. Experiments are conducted using multispectral and hyperspectral data and the results show the advantages and limitations of the proposed binary tree method as compared to single-stage classification.

4.2 Hughes Phenomenon

A landmark paper was written by Hughes [2] who first observed that there is an optimal dimensionality associated with the set of classes and their training set size. In other words, when the number of training samples is limited, the performance of the classifier first improves up to a point and then deteriorates as the number of dimensions increases. This has subsequently been referred to as the Hughes phenomenon. The deterioration in performance is essentially due to the fact that the estimation of class conditional densities which determine the decision boundary for classification is based on a limited set of samples. As more features are added, more samples are required to obtain an adequate density estimation. It has been shown that the required number of training samples to achieve a certain classification performance is proportional to the dimensionality for a linear classifier and to the square of dimensionality for a quadratic classifier [3].

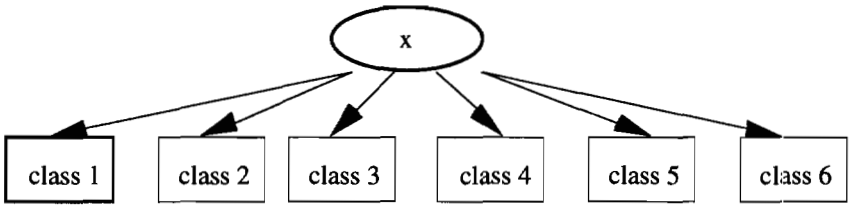
An obvious solution to circumvent the Hughes phenomenon is to reduce the number of features by applying feature selection or extraction methods [36]. Alternatively, classification rules which require fewer training samples for good performance such as the linear classifier can be adopted. It was shown that when the set of design samples is extremely limited, the linear classifier can often perform better than the quadratic one [21] even though the true covariance matrices differ substantially. In Chapter 3, a compromise between the linear and quadratic classifiers was proposed. In this chapter, the problem of limited training set size is addressed using a divide-and-conquer approach.

As a consequence of the progress in high resolution sensors for remote sensing, a multispectral image becomes more complex in the sense that more classes of varying sample size are separable with increased dimensionality. Some classes of interest can have very few design samples due to the difficulty in labeling training samples. In view of this increased complexity, it may be desirable to process different classes from the same image using different classification rules and feature reduction methods. This can only be accomplished using a multistage approach such as the decision tree classifier. In the next section, the binary tree classifier is briefly reviewed and a hybrid design is described, which may employ different classification rules and feature extraction methods based on the training samples.

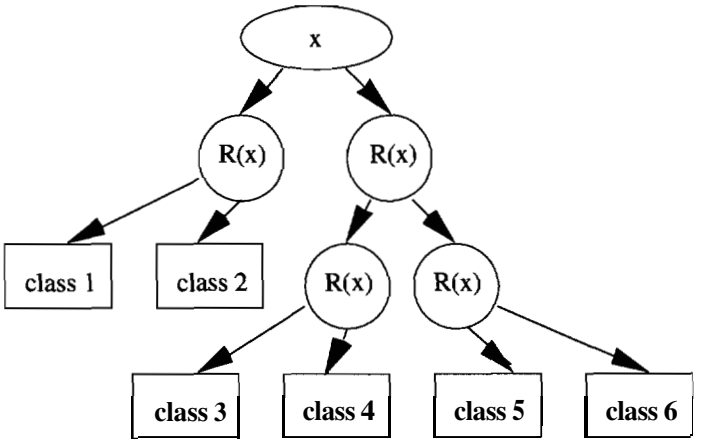
4.3 Binary Tree Design for Classification

4.3.1 Introduction

The decision tree classifier (DTC) has been widely used for classification and other purposes for the past few decades. A general review can be found in [35]. In principle, it divides a complex decision into several simpler ones in a hierarchical fashion. Figure 4.1 shows a single-stage classifier in contrast to a binary tree classifier. The circular nodes represent decision nodes and the square nodes are terminal nodes. Each decision node has a decision rule $R(x)$ with x as its input value. The terminal nodes then assign x to one of the class labels.



Single-stage classifier



Binary decision tree classifier

Figure 4.1 Comparison of Single-stage and Binary Decision Tree Classifiers

The hierarchical structure of decision tree classifiers has several desirable properties:

1) A tree classifier is more computationally efficient than a conventional single-stage classifier. In a single-stage classifier, a data sample is tested against all classes in contrast to a subset of classes as in a tree classifier.

2) A tree classifier is more flexible than a single-stage one in that the nodes can have different decision rules and subsets of features. In single-stage classifiers, a subset of features is selected by optimizing a global criterion and is used to discriminate among all classes. In contrast, a tree classifier offers the flexibility to select a dlifferent subset of features for each node such that the feature subset is focused on optimizing the classification at that particular node. Similarly, single-stage classifiers use a decision rule for all classes, while tree classifiers may use a unique decision rule for each node.

3) A tree classifier may circumvent the Hughes effect due to small training sample size by focusing on fewer classes and hence using fewer features at each node. In a single-stage classifier, the discrimination among all classes is based on a complex decision boundary which requires more training samples to obtain a good approximation. By focusing on few classes on each node, the tree classifiers essentially divide a complex decision boundary into several simpler ones. Therefore, fewer features are needed at each node and the Hughes effect can be avoided.

Although tree classifiers offer many benefits, they come with several limitations as well. First of all, the design of an optimal tree classifier still remains intangible. There are many factors to be considered, such as the tree structure, the node decision rule and the feature selection method. Since one cannot simultaneously optimize the accuracy and efficiency [37], the tree design is subject to the trade-off between design complexity and perlormance. Furthermore, without considering the optimization of all levels in the tree, errcas may accumulate at each level. In spite of these limitations, the benefits of a good tree design still outweigh the drawbacks. In the next section, several existing tree classifiers are briefly reviewed and serve to demonstrate some design issues.

4.3.2 Previous Work

Various types of the DTC have emerged during the past three decades, most notably, Quinlan's ID3[38] and Breiman *et al.*'s work on classification and regression trees (CART) [39] in 1980's. In ID3, each node of the tree performs a single test on one feature to form the so-called "axis-parallel" test. One drawback of this approach is that for data with numerous features, ID3 might result in a large, unruly tree with many repetitive tests. The CART approach involves testing a linear combination of features to form an oblique decision surface in the feature space. For linearly separable data, the latter will clearly produce a more accurate and compact tree whereas the ID3 will result in a staircase-like decision boundary (See Fig. 4.2). Both approaches are non-parametric, i.e. no assumption is made on the underlying data distribution. Therefore, they have become popular and widely studied among the machine learning researchers [35].

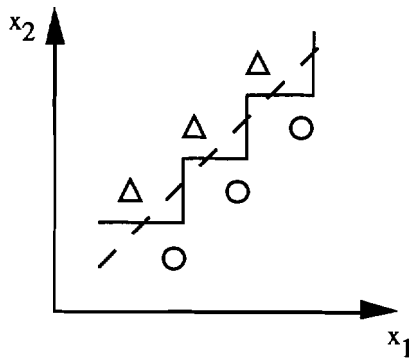


Figure 4.2 Axis-parallel and Linear Decision Boundaries

Even though a non-parametric classifier offers the flexibility to classify data of unknown distribution, it requires a generous amount of training samples to achieve the desirable accuracy. A study [3] has estimated that the required number of training samples grows exponentially with respect to the number of features for a non-parametric classifier. As previously mentioned, the number of training samples remains relatively few for hyperspectral data. Although feature selection may alleviate the problem of few training samples, the solution obtained may not be optimal. Therefore, these design approaches are not suitable for classifying hyperspectral data.

An alternative tree design approach that has been introduced in the pattern recognition research area assumes that the data distribution is known. Most commonly, the data is assumed to be normally distributed. When the classes are indeed normal as in the case of remote sensing data, the tree design problem is then reduced to parameter estimation. In addition, the parametric approach requires much fewer training samples than the non-parametric one.

You and Fu [40] suggested a linear binary tree design which combined classes into two non-overlapping subgroups at each node using class statistics. The two subgroups were found by comparing a measure of pairwise separability over all classes. A varying subset of features was selected from the feature space for each node bat the number of features remained fixed. Then using an iterative process with an initial guess, a classifier is found that provides minimum error probability. If this error exceeds the pre-defined error bound, the class that commits the maximum error is included in both of the subgroups and is removed from consideration in computing the error,. By including a class in both nodes, this method allows overlapping of classes which means that two nodes can contain at least one common class.

Another parametric tree classifier design was proposed by Kim and Landgrebe [41] using both bottom-up and top-down methods (hybrid approach) sequentially for classifying hyperspectral data. The bottom-up method computes the Bhattacharyya distance between each pair of classes and the two classes with the smallest distance are merged to form a new group. The mean vector and covariance matrix in the newly formed group are computed, and the process is repeated until two groups are left to form two cluster centers. These two subgroups are then assumed to be normally distributed and form a maximum likelihood decision rule. Several feature extraction methods are incorporated and compared for their effectiveness. This method does not allow overlapping of classes.

The above mentioned parametric decision tree classifiers require the merging of classes using the statistics obtained from training data. While merging of classes can simplify the decision boundary and hence the decision process, misclassification could easily occur if the combination of these simple decision boundaries does not yield a good approximation of the global decision boundary. This problem can be solved to some extent by allowing overlapping of classes. When a common class is included in several nodes, the overall decision boundaries generated by the tree is more complex and offers a way to improve the classification rate. Figure **4.3** illustrates an example of the decision boundaries generated by overlapping classes. It is shown that when class overlapping is allowed, the decision boundary generated is more complex and precise, thus reducing misclassification risk.

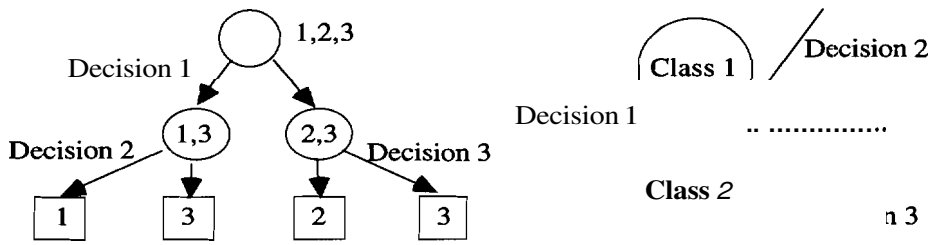


Figure **4.3** Illustration of **A** Tree Classifier with Overlapping Classes

In spite of the obvious advantage, excessive overlapping of classes will result in a large tree, and thus reducing the efficiency of the tree classifier. The following section describes a binary tree design which compares two classes at each node, instead of two

subgroups of classes. Consequently, the problem of merging can be avoided. In addition, overlapping of classes is allowed. To reduce the size of the tree due to common classes, the classes with the largest separation are processed near the top of the tree. In other words, the classes are "ordered". This forms the so-called hybrid design.

4.3.3 Proposed binary tree structure design

The proposed binary tree constructed uses the hybrid approach, which is a combination of the top-down and bottom-up methods. The bottom-up approach [42] typically uses the training samples to construct the tree and bears resemblance to agglomerative hierarchical clustering. Using some distance measure, such as Euclidean distance or Bhattacharyya distance, pairwise distances between a priori defined classes are computed. Classes with smaller distances are merged first until the root contains only one group. In the proposed method, the pairwise distance between pre-defined classes is first computed. At each node, two classes with largest separation are selected and form the two cluster centers of two nodes and the rest of classes are then classified into the nodes. If a defined amount of training samples of any class are assigned into both nodes, the class is included in both nodes for further comparison. This approach has the benefit of computing the pairwise distances only once and hence reducing computational time. Also, no merging of classes is required.

In the top-down approach, the design consists of the following tasks [35]:

1) selection of a node decision rule
2) termination rule
3) decision tree structure
4) feature reduction

These different aspects of a tree classifier should be considered simultaneously for an optimal design. Unfortunately, this problem of optimization is non-trivial. To simplify the design problem, a binary structure is adopted. The termination rule is simply the :majority rule, that is, the class label in the terminal node is assigned to the class with the most training samples at that node. Furthermore, the classes art: assumed to be nonnally distributed. Therefore, the decision rule for node splitting is a maximum-

likelihood classification of Gaussian classes. Feature reduction is discussed in the next section. Therefore, the proposed binary tree design algorithm is summarized as follows:

The Binary Tree Design Algorithm:

Step 1. Compute the separability (Bhattacharyya distance or Euclidean distance) between each class pair.

Step 2. Select two classes with the largest separation as two cluster centers. Compute the mean vectors and covariance matrices of the two classes and use them as the node statistics.

Step 3. Classify the remaining classes into one of the two nodes using the following decision rule:

$$J_i = \left(x - \hat{M}_i\right)^T \hat{\Sigma}_i^{-1} \left(x - \hat{M}_i\right) + ln\left|\hat{\Sigma}_i\right| \quad \text{where } i = n_L \text{ or } n_R$$

and $$x \in n_L \text{ if } J_{n_L} < J_{n_R}$$

where x is the data sample of p features

$\hat{M}_i$ is the sample mean estimate of node i

$\hat{\Sigma}_i$ is the sample covariance matrix estimate of node i

n_L and n_R represent left and right node, respectively.

Step 4. If all the training samples from one class are classified into a node, the class is no longer considered at the other node. If not, the class is retained in both nodes for further pair-wise comparison.

The proposed binary tree design has the following desirable characteristics:

1) By using the bottom-up approach in which the terminal nodes of the tree consist of the set sf pre-defined classes, the tree classifier ensures that the classes have informational values.

2) By separating classes with the largest distance first, the occurrence of overlapped classes can be reduced, thus decreasing the size of the tree.

3) Using two classes instead of two subgroups of classes avoids the problem of merging. In addition, since the classes are assumed to be normally distributed, the Gaussian maximum likelihood classification rule can be readily applied as the node splitting rule.

4) Since the statistics used at each node are defined by the training set of two pre-defined classes, this information can be stored and used subsequently for other repetitious nodes by generating a simple look-up table.

5) In a two-class hierarchical structure, only two classes are considered at each node. This greatly simplifies the analysis process in the sense that the optimization criterion for two classes often exists in closed form, such as the Bhattacharrya distance. Furthermore, the feature extraction methods for two classes are well understood, whereas the optimization for multiple classes is more complex and may not even exist. The next section will discuss this issue in detail.

4.3.4 Feature extraction

The benefits of performing feature reduction for remote sensing applications are twofold: 1) to circumvent the Hughes phenomenon and 2) to reduce the amount of computation required for classification. Feature reduction methods can be roughly divided into two categories: feature selection and feature extraction. In feature selection, features that do not contribute to the discrimination of classes can be eliminated by assessing some criteria before and after the removal. A criterion commonly used is the separability of classes or the n-fold cross-validation method. If the removal does not lower the criteria substantially, the features are redundant. Unfortunately, optimal feature selection involves exhaustive search among all features, which is computationally infeasible for hyperspectral data. Suboptimal search involving subsets of features such as

forward or backward selection may have undesirable effects for multispectral data [30]. Therefore, feature selection is not considered in this work.

Feature extraction is the other form of feature reduction and involves the transformation of data into a smaller subset of features while retaining the class sepiirability as much as possible. The transformation is usually linear and based on the optimization of some criteria. This section reviews several feature extraction algorithms and discusses their relative strengths and weaknesses when applied to a binary tree classifier.

A. Principal Component Analysis

This method involves representing $x \in \Re^p$ by the summation of p orthonormal vectors using Karhunen-Loeve transformation. The columns of transformation matrix consist of the eigenvectors corresponding to p eigenvalues of Σ_x, the covariance matrix of x, as follows:

$$\Sigma_y = \Phi^T \Sigma_x \Phi = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_N \end{bmatrix} = \Lambda$$

where $y = \Phi^T x$ is the linear transformation of x and $\boldsymbol{O} = \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_p \end{bmatrix}$ is the non-singular transformation matrix satisfying the condition

$$\phi_i \phi_j = \begin{cases} 1 & \text{for i = j} \\ 0 & \text{for } i \neq \mathbf{j} \end{cases}$$

To extract $q < p$ features, q out of p eigenvectors are selected corresponding to the q largest eigenvalues. Although this transformation is optimal with respect to fitting the data, it is not necessarily optimal with respect to discriminating the data [30].

B. Discriminant Analysis Feature Extraction (DAFE)

Discrimimant analysis or canonical analysis [30] uses the ratio of a between-class scatter matrix Σ_b to within-class scatter matrix Σ_w as a criterion function, and computes a vector d to maximize

$$\frac{d^T \Sigma_b d}{d^T \Sigma_w d}$$

where

$$\Sigma_b = \sum_{i=1}^{L} \alpha_i (M_i - M_o)(M_i - M_o)^T \qquad \text{(between-class scatter matrix)}$$

$$\Sigma_w = \sum_{i=1}^{L} \alpha_i \Sigma_i \qquad \text{(within-class scatter matrix)}$$

$$M_o = \sum_{i=1}^{L} \alpha_i M_i$$

Here M_i, Σ_i, and α_i are the mean vector, the covariance matrix, and the prior probability of class i respectively. And L is the total number of classes.

Although the discriminant analysis performs well for most cases, there are several drawbacks for this method. First of all, the approach delivers features only up to the number of classes L minus one. For a binary tree classifier, this means that there is only one feature extracted at each node. One feature may not be optimal to discriminate between classes with complex decision boundary. Second, if the mean values are similar or the same, the extracted feature vectors are not reliable. Furthermore, for multiple classes, if a class has a mean vector very different from the other classes, the between-class scatter matrix is more biased towards that class, resulting in ineffective features.

C. :Decision Boundary Feature Extraction (DBFE)

The decision boundary feature extraction technique involves extracting features based on the effective decision boundaries between classes [32]. It was shown that all the features needed for classification are normal to the effective decision boundary, which is part of the decision boundary separating 90% of the training sample!;. In addition to finding the feature vectors, this method also predicts the minimum number of features necessary to achieve the same classification accuracy as conducted in the original space.

In order to determine the effective decision boundary, the majority of training samples are first selected. Using a Gaussian maximum likelihood classifier, the procedure begins with classifying the training samples at full dimensionality and

thresholding the outliers. Therefore, for a p dimensional multispectral space the number of training samples must be greater than (p+1) to avoid singularity. Since the method depends on how well the training samples approximate the decision boundaries, the number of training samples required could be much more for high dimensional data. For hyperspectral images, the number of training samples is usually not enough to prevent singularity or to yield a good covariance estimate. The DBFE method is also computationally more intensive than the previous methods. In addition, DBFE for more than two classes is suboptimal. However, it generates more than (L−1) features. Typically, as more features are added, the class separations improve as well. It functions well even when the means or the covariances are equal, and also simultaneously provides information on the number of features required for good accuracy.

D. Bhattacharyya Distance Feature Extraction (BDFE)

The Bhattacharyya distance is a convenient measure of class separability for two classes. Furthermore, it gives an upper bound of Bayes error for norrnal distributions. The Bhattacharyya distance is given as [30]

$$\mu\left(\frac{1}{2}\right)=\frac{1}{8}(M_2-M_1)^T\left[\frac{\Sigma_1+\Sigma_2}{2}\right]^{-1}(M_2-M_1)+\frac{1}{2}\ln\frac{\left|\frac{\Sigma_1+\Sigma_2}{2}\right|}{\sqrt{|\Sigma_1||\Sigma_2|}} \tag{4.1}$$

The optimization of the Bhattacharyya distance is non-trivial. One must either consider special cases or suboptimal solutions for the general case. To consider special cases, Eq. (4.1) can be decomposed into two terms:

$$\mu_1=\frac{1}{8}(M_2-M_1)^T\left[\frac{\Sigma_1+\Sigma_2}{2}\right]^{-1}(M_2-M_1)$$

and

$$\mu_2=\frac{1}{2}ln\frac{\left|\frac{\Sigma_1+\Sigma_2}{2}\right|}{\sqrt{|\Sigma_1||\Sigma_2|}}$$

When $\Sigma_1=\Sigma_2$, $\mu(\frac{1}{2})$ is reduced to μ_1 and hence the optimization involves only μ_1. We can rewrite μ_1 as follows:

$$\mu_1 = \frac{1}{8}(M_2 - M_1)^T \left[\frac{\Sigma_1 + \Sigma_2}{2}\right]^{-1} (M_2 - M_1)$$
$$= \frac{1}{8} tr\left\{\bar{\Sigma}^{-1}(M_2 - M_1)(M_2 - M_1)^T\right\}$$
$$= \frac{1}{8} tr\left\{\Sigma_w^{-1}\Sigma_b\right\}$$

where $\bar{\Sigma} = \frac{\Sigma_1 + \Sigma_2}{2}$ and $a, = a, = \frac{1}{2}$ is assumed for Σ_w and Σ_b. Therefore, ignoring the multiplicative constant, the optimization of μ_1 alone is the same as the discriminant analysis. From previous discussion, only one linear feature is needed to maximize this criterion and the transformation is given as:

$$y = (M_2 - M_1)^T \bar{\Sigma} x$$

When $M_1 = M_2$, $\mu\left(\frac{1}{2}\right) = \mu_2$. Based on the optimization of μ_2 , q eigenvectors of $\Sigma_2^{-1}\Sigma_1$ are selected corresponding to the q largest $\left(\lambda_i + 1/\lambda_i + 2\right)$terms where λ_i is the eigenvalue. Each eigenvalue of $\Sigma_2^{-1}\Sigma_1$ gives the ratio of the ω_1- and ω_2-variances along the respective eigenvectors. By selecting the largest $\left(\lambda_i + 1/\lambda_i + 2\right)$ terms, this method extracts the features where the variances of the two classes are different.

Typically for one-stage classifiers, when the Bhattacharrya distance is used for feature reduction for more than two classes, the minimum or average pairwise distance between all class combinations is used as the sub-optimal criterion for optimization. For a binary tree classifier, this suboptimal choice of criterion can be avoided. When the node splitting rule involves two normal classes, the Bhattacharrya distance is an optimal criterion for feature extraction.

Since the optimization of the Bhattacharyya distance is infeasible, a suboptimal procedure is adopted to find the effective features. The method proceeds as follows:

Bhattacharyya Distance Feature Extraction (BDFE) Algorithm:

Step 1. Compute the eigenvalues λ_i and orthonormal eigenvectors ϕ_i of $\bar{\Sigma}^{-1}(M_2 - M_1)(M_2 - M_1)^T$ where $\bar{\Sigma} = \frac{\Sigma_1 + \Sigma_2}{2}$. Since the rank of the matrix is one, only the first eigenvalue A, is non-zero. The class separability due to the mean difference is then preserved by the transformation $\phi_1^T x$.

Step 2. Perform the transformation $y = \left[\phi_2 \cdots \phi_p\right]^T x$ by which x is mapped to the $(p-1)$-dimensional subspace where there is no information due to mean difference. Let $\Phi_{p-1} = \left[\phi_2 \cdots \phi_p\right]$.

Step 3. Extract $(q-1)$ features, $\Psi = \left[\psi_1 \cdots \psi_{q-1}\right]$, by optimizing μ_2 on y. These features then preserve the information due to covariance: difference. The overall transformation is then given by

$$z = \left[\phi_1 \ \vdots \ \Phi_{p-1}\Psi\right]^T x.$$

The above procedure has the advantage of adding more features based on covariance information to the one feature extracted by discriminant analysis for two classes. The benefit of having additional features is demonstrated in the following experiment. Using simulated data of 8 features and 2 classes, the experiment compares the effectiveness of discriminant analysis feature extraction (DAFE) and the above numerical approach based on Bhattacharyya distance (BDFE). Three sets of computer generated data are generated, in which the mean difference (M), the covariance difference (C), and both the mean and covariance differences (M-C) are dominant, respectively. Since there are only two classes, only one feature is used using DAFE. For BDFE, a total of four features are obtained. The classification results are summarized in Figure 4.4. These results show that when the class covariance matrices are: different, it is advantageous to use additional features obtained with BDFE based on covariance information.

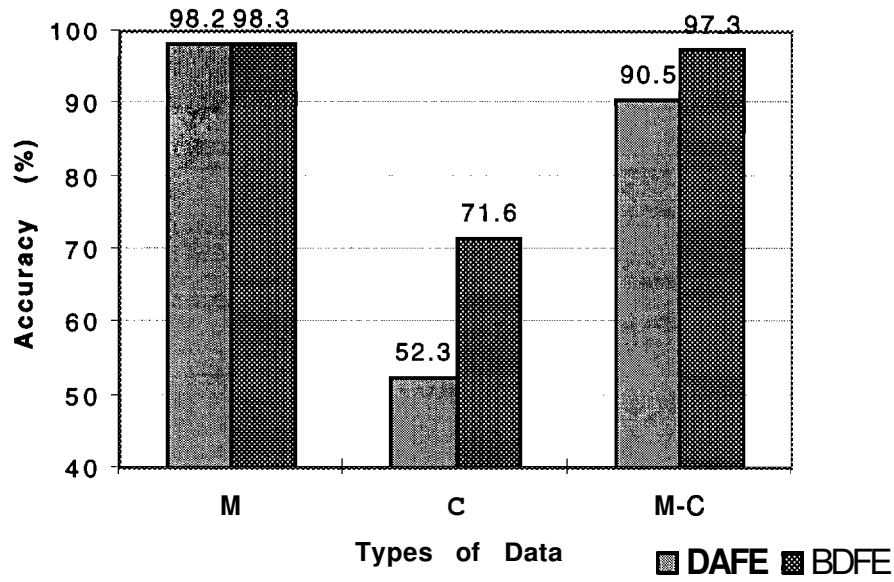


Figure 4.4 Comparison of DAFE and BDFE Methods for Data With Dominant Mean Difference (M), Dominant Covariance Difference (C), and Both Mean and Covariance Difference (M-C)

4.4 Binary Tree Design for Feature Extraction

The above binary tree design can also be used as a feature extraction method. Since two classes are processed at each node, the features generated from the feature extraction algorithm are optimal with respect to the two classes. Therefore, features from each tree node can be combined to form collectively a set of features for the single-stage classifier. The advantage of this approach is that the feature extraction algorithms such as DBFE are not necessarily optimal for more than two classes whereas the BDFE algorithm is only applicable for two classes.

For multiclass problems, DBFE generates features by averaging pairwise decision boundary feature matrices for all classes. The average matrix may not be optimal for all classes since some class pairwise decision boundaries are not necessarily effective for the multiclass situation. Therefore, the features obtained from the average rnatrix may not be optimal. The binary tree design can circumvent this problem by extracting optimal features for each pair of classes based on their decision boundary feature matrix and use

these features for the single-stage classifier. Likewise, BDFE can be extended to the multiclass problem using the binary tree to extract features for pairwise classes. These features can then be combined for the single-stage classifier. The effectiveness of these methods will be demonstrated by experiments.

4.5 Experimental Results

Experiment 4.1

In the following experiments, the proposed binary tree design is used both as a classifier and a feature extraction method. The data set consists of some agricultural classes from an AVIRIS image with 220 spectral bands taken over of NW Indiana's Indian Pine test site in June 1992. The water absorption bands and noisy bands (104-108, 150-163, 220) are removed, resulting in a total of 200 bands. Since the data were collected in the early part of the growing season, soybean and corn canopies gave only about 5% ground cover. Four classes which present a challenging classification task are selected. The mean vectors of these classes are plotted in Figure 4.6. The figure shows that these classes have very similar mean values, thus presenting a challenging classification task. The covariance information should play an irnportant part in classification. The number of labeled samples of these classes are given.in Table 4.1 and their ground truth map is shown in Figure 4.5. Since the labeled samples are few, to retain enough samples as training and testing samples, the spectral channels are sampled at a fixed interval of 4, which leaves 50 channels for the experiment.

Table 4.1
Class Description for AVIRIS Data in Figure 4.5

Class Names	No. of Labeled Samples
Corn-notill	1066
Corn-min	834
Soybean-notill	501
Soybean-min	662

Figure 4.5 AVIRIS Data and Ground Truth Used in Experiment 4.1 (Original in Color)

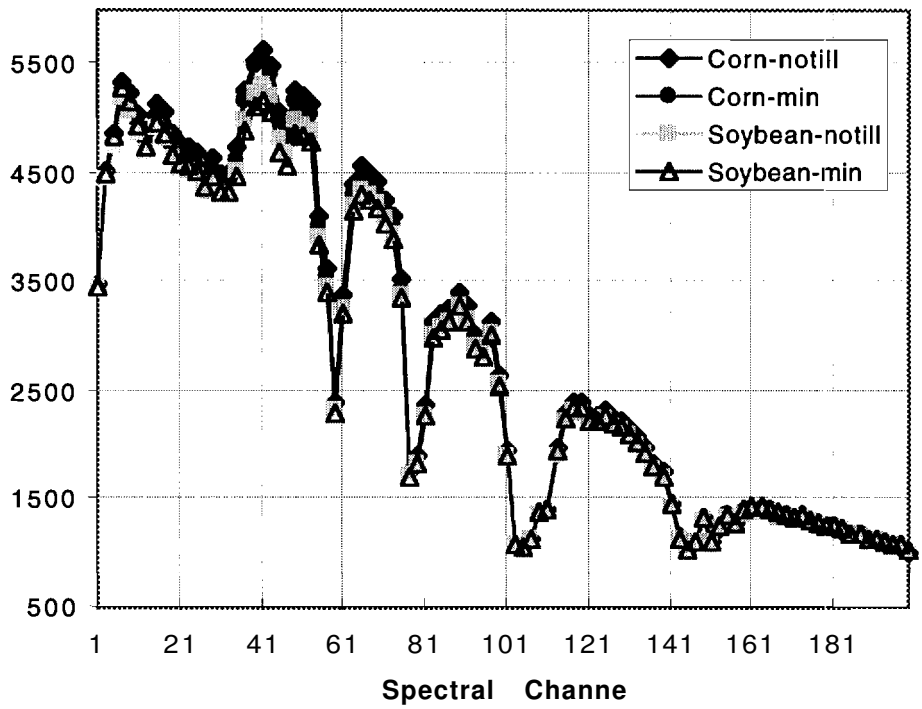


Figure 4.6 Mean Graph of AVIRIS Data in Figure 4.5

There are six methods being tested and compared. The methods and their abbreviations are listed in Table 4.2. The method "Resubstitution" essentially uses all the labeled samples for training and testing as well. Therefore, its classification accuracy provides an upper bound. In other words, it is the best performance attainable by the limited design set. The method "ML-DAFE" is the single-stage maximum likelihood classifier with discriminant analysis as the feature extraction method. "ML-DBFE" denotes the single-stage maximum likelihood classifier using decision boundary feature extraction (DBFE) method. The two-class binary tree classifier with DBFE method at each node is denoted as "DTC-DBFE" whereas the single-stage maximum likelihood classifier using features generated by the binary tree with DBFE is indicated as "ML/DTC-DBFE". Likewise, "DTC-BDFE" and "ML/DTC-BDFE" represent the binary tree classifier with Bhattacharyya distance feature extraction (BDFE) at each node and the single-stage classifier with tree generated features using BDFE.

Table 4.2
Description of Methods Tested in Experiment 4.1

Abbreviation	Methods
Resubstitution	• Single-stage Gaussian Maximum Likelihood using all labeled samples for training and testing
ML-DAFE	• Single-stage Gaussian Maximum Likelihood with DAFE
ML-DBFE	• Single-stage Gaussian Maximum Likelihood with DBFE
DTC-DBFE	• Decision Tree with DBFE
ML/DTC-DBFE	• Single-stage Gaussian Maximum Likelihood with tree generated features using DBFE
DTC-BDFE	• Decision Tree with BDFE
ML/DTC-BDFE	• Single-stage Gaussian Maximum Likelihood with tree generated features using BDFE

The experiment is repeated with different number of spectral features up to 50 features and the training set size of 55, 100 and 500 per class. The training samples are randomly selected and thus the experiment is repeated 10 times. Except for the first experiment using 5000 labeled samples, the data samples are obtained directly from the image. The simulation data with 5000 samples is generated using the statistics from the labeled samples of the four classes. The purpose of this simulation is to compare the methods for large training and test sets. The results are shown in Figures 4.8-4.10. It should be noted that for the binary tree generated features using ML/DTC-DBFE and ML/DTC-BDFE methods, the total number of features from the collection of one feature generated at each node is equivalent to the number of internal nodes with non-repeating decisions. The binary tree generated for this data is shown in Figure 4.7. Since the number of non-repeating decisions is six, in the results below, the number of features are in the multiple of six for the single-stage classifier with tree generated features using ML/DTC-DBFE and ML/DTC-BDFE methods.

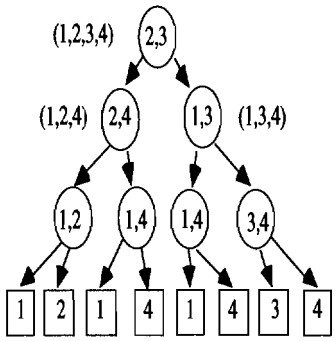


Figure 4.7 The Binary Tree Generated for AVIRIS Data

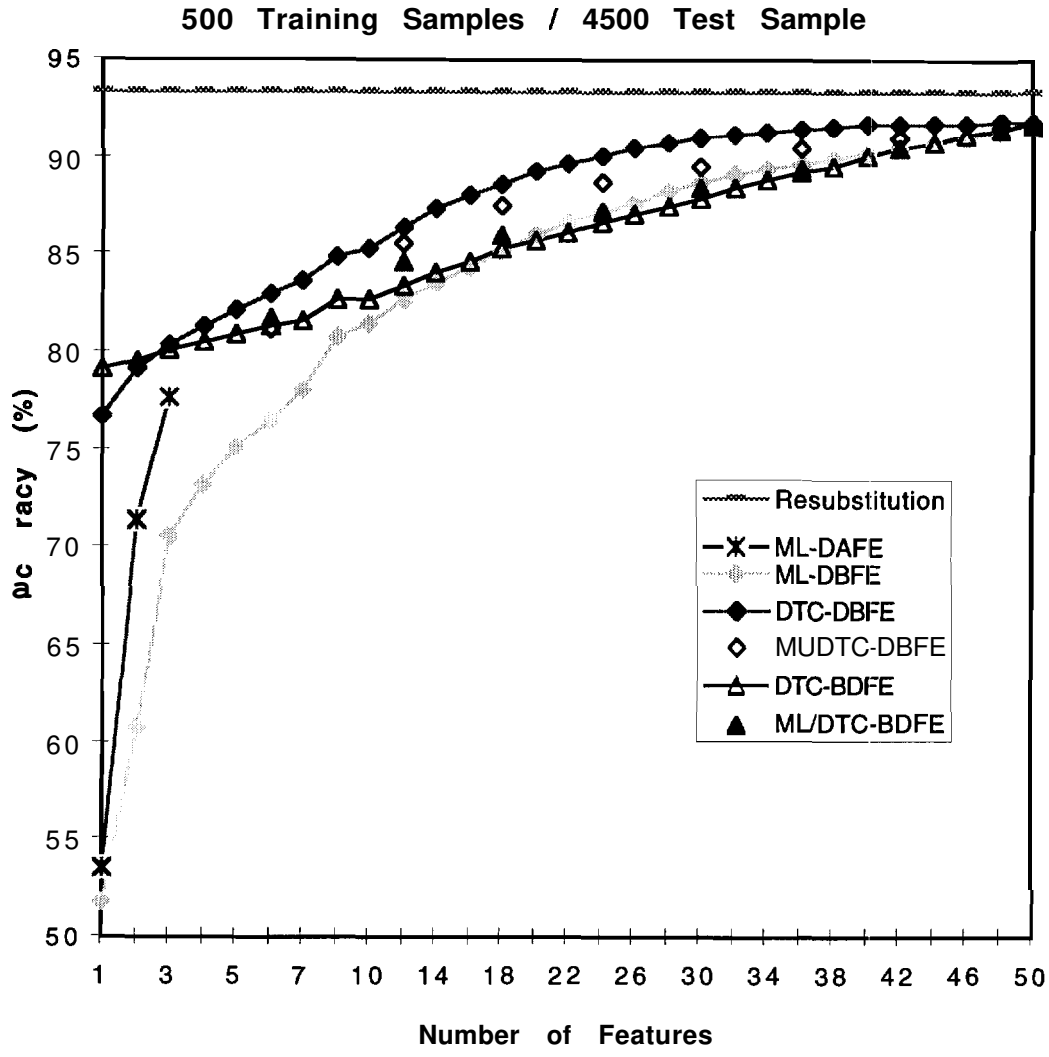


Figure 4.8 Classification Result for 500 Training Samples (AVIRIS Data)

The above figure shows the result obtained for various methods when there are ample training and test samples. Since this is the case of large sample size, no Hughes phenomenon takes place and the highest accuracy occurs at full dimensionality for all methods. The best result is achieved by the resubstitution method which provides the upper bound for the data set. The next best result is obtained by DTC-DBFE. It demonstrates that for large training set size, the optimal features are generated based on the decision boundary which is well-defined for large sample size. The two-class binary tree also generates the optimal features for DBFE as compared to the algorithm for multiclass DBFE as proposed in [32]. Since there are four classes, only three features are generated using DAFE. These features also do not utilize the covariance information.

Therefore, its performance is worse than other methods. The result also shows that BDFE is an suboptimal approach.

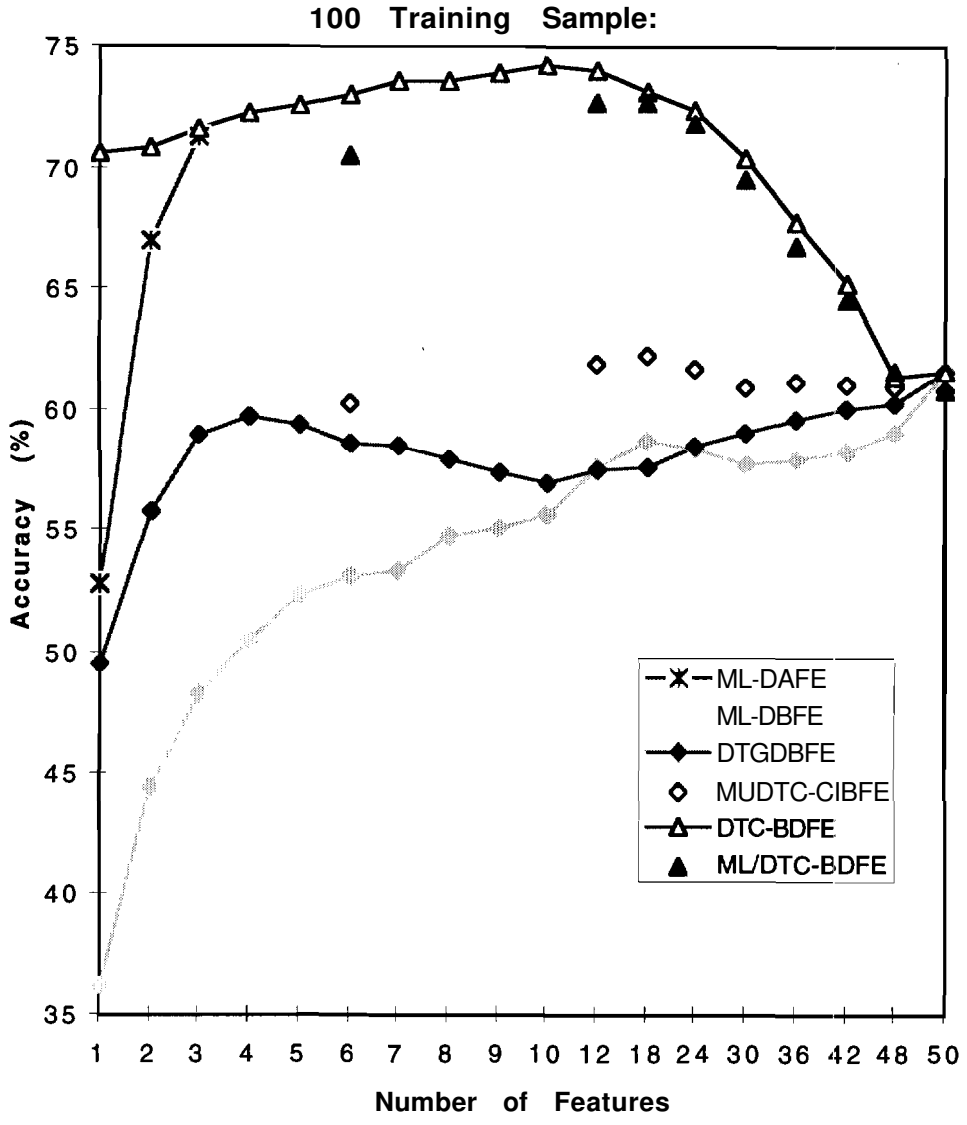


Figure 4.9 Classification Result for 100 Training Samples (AVIRIS Data)

Using 100 training samples for 50 channels represent the case of moderate training set size. Figure 4.9 shows that the best result is obtained by DTC-EIDFE. Although BDFE is suboptimal, it requires fewer training samples than DBFE for better performance. When the training set size decreases, the parameters are not well estimated which affect the decision boundary estimate as well. Therefore, the performance of DBFE suffers. Since BDFE also uses covariance matrix estimate, the performance

declines as well, but not as much as DBFE. The DAFE method uses only the mean information, so its classification accuracy remains comparatively stable for moderate training set size.

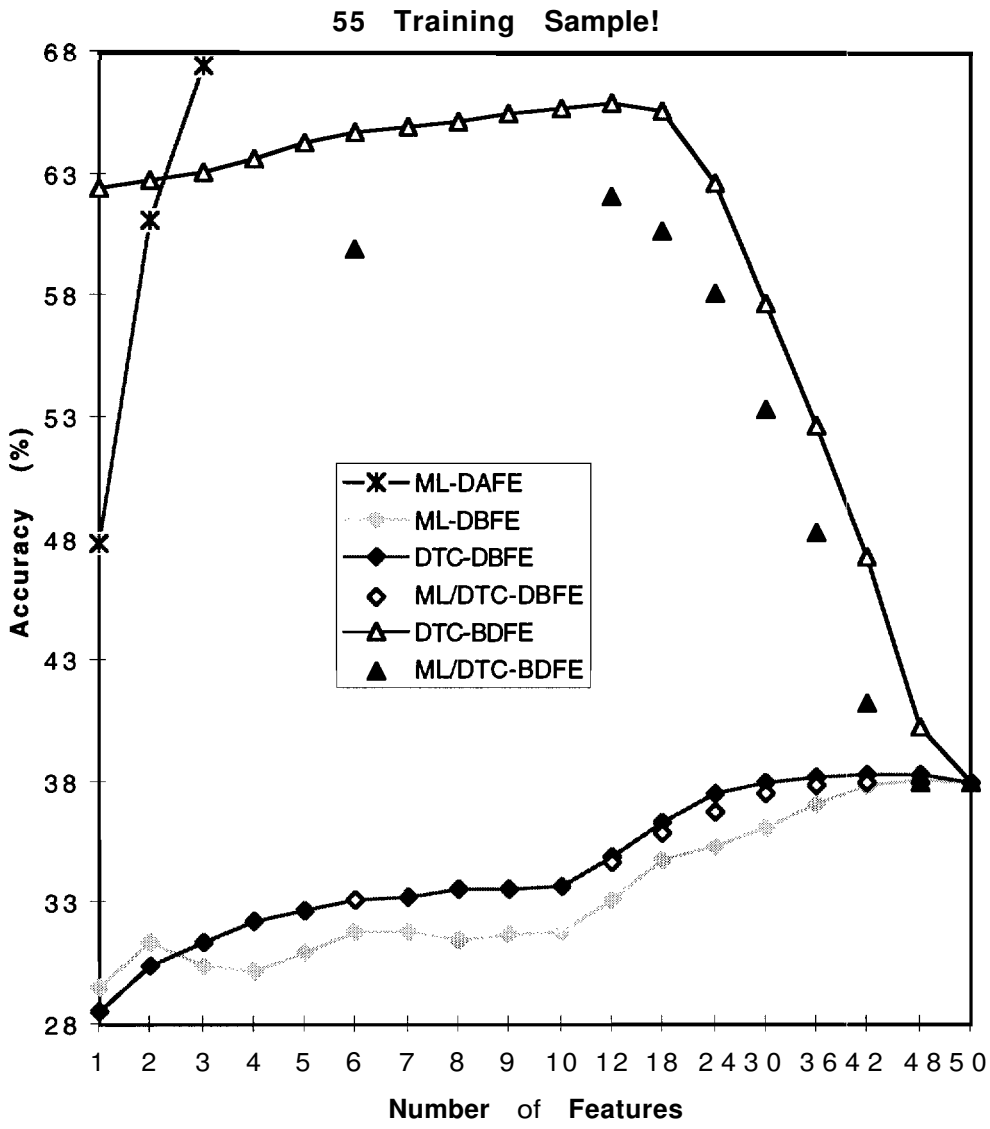


Figure 4.10 Classification Results for 55 Training Samples (AVIRIS Data)

Figure 4.10 shows the classification result for small training set size since there are only 55 training samples per class for 50 channels. In this setting, the covariance matrix estimate becomes highly variable. Since both DBFE and BDFE methods make use of covariance information, their performance deteriorates. However, by processing two classes at a time and using fewer features, the proposed tree classifier using BDFE (DTC-

BDIFE) maintains a relatively good performance up to a point. As expected, the best result is obtained by using DAFE whose optimization is based on the mean vectors only.

Experiment 4.2

The previous experiment is repeated for another set of data taken by a different sensor. The test data was taken from the multispectral data collected using Field Spectrometer System (FSS) and the major parameters are shown in Table 4.3. Four multi-temporal classes of the type "Spring Wheat" are chosen from the FSS data collected in June, July and August, 1978. The number of labeled and training samples are given in Table 4.4. A total of 20 spectral bands are selected from the original 60 spectral bands. As shown in Figure 4.11, these multi-temporal classes have some difference in mean values. The generated binary tree is illustrated in Figure 4.12. Again, there are six non-repeating internal node decisions in the tree, so the tree generated features are in the multiples of six. Figure 4.13-4.15 show the classification results using different training set size. Since the training samples are randomly selected, the experiment is repeated 10 times and the figures show the mean values of classification accuracy.

Table 4.3
Parameters of Field Spectrometer System

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

Table 4.4
Class Description of FSS Data in Figure 4.11

Multi-temporal Classes	Abbreviations	No. of Labeled Samples
Spring Wheat 8/16	SP8_16	464
Spring Wheat 7/26	SP7_26	515
Spring Wheat 7/09	SP7_09	454
Spring Wheat 6/02	SP6_02	515

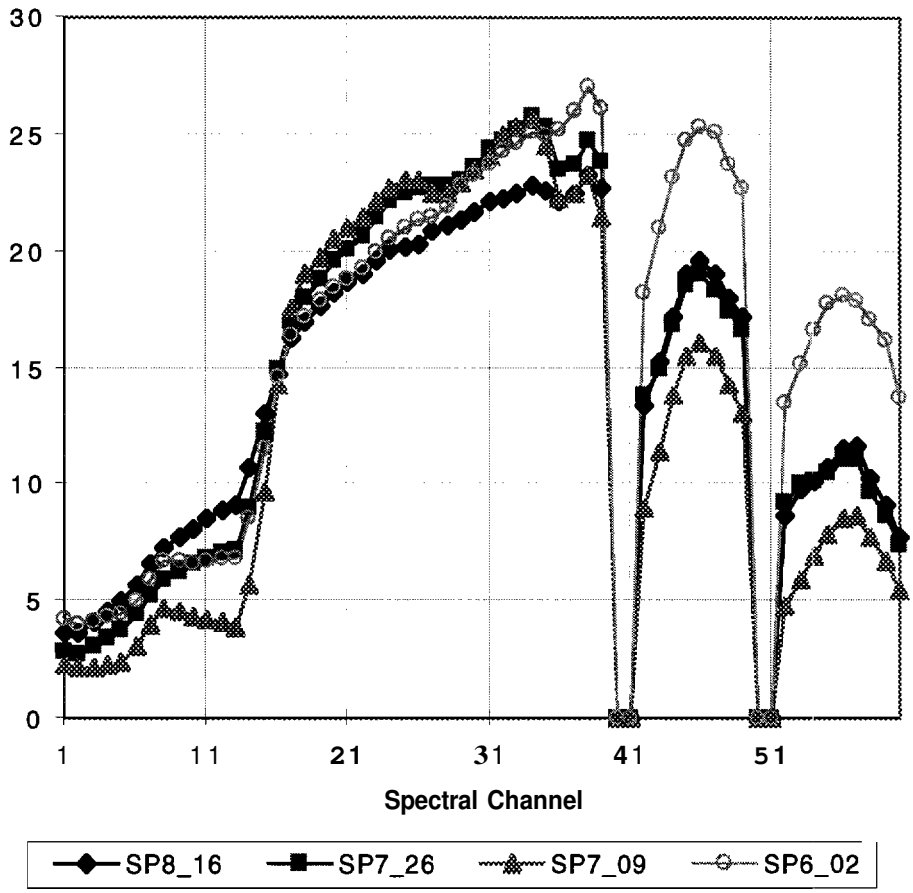


Figure 4.11 Mean Graph of Multi-temporal FSS Data

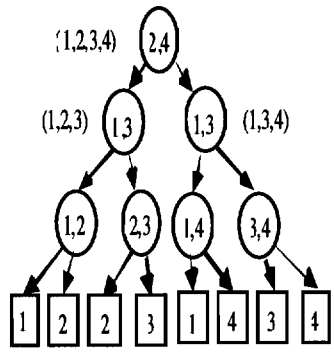


Figure 4.12 The Binary Tree Generated for FSS Data

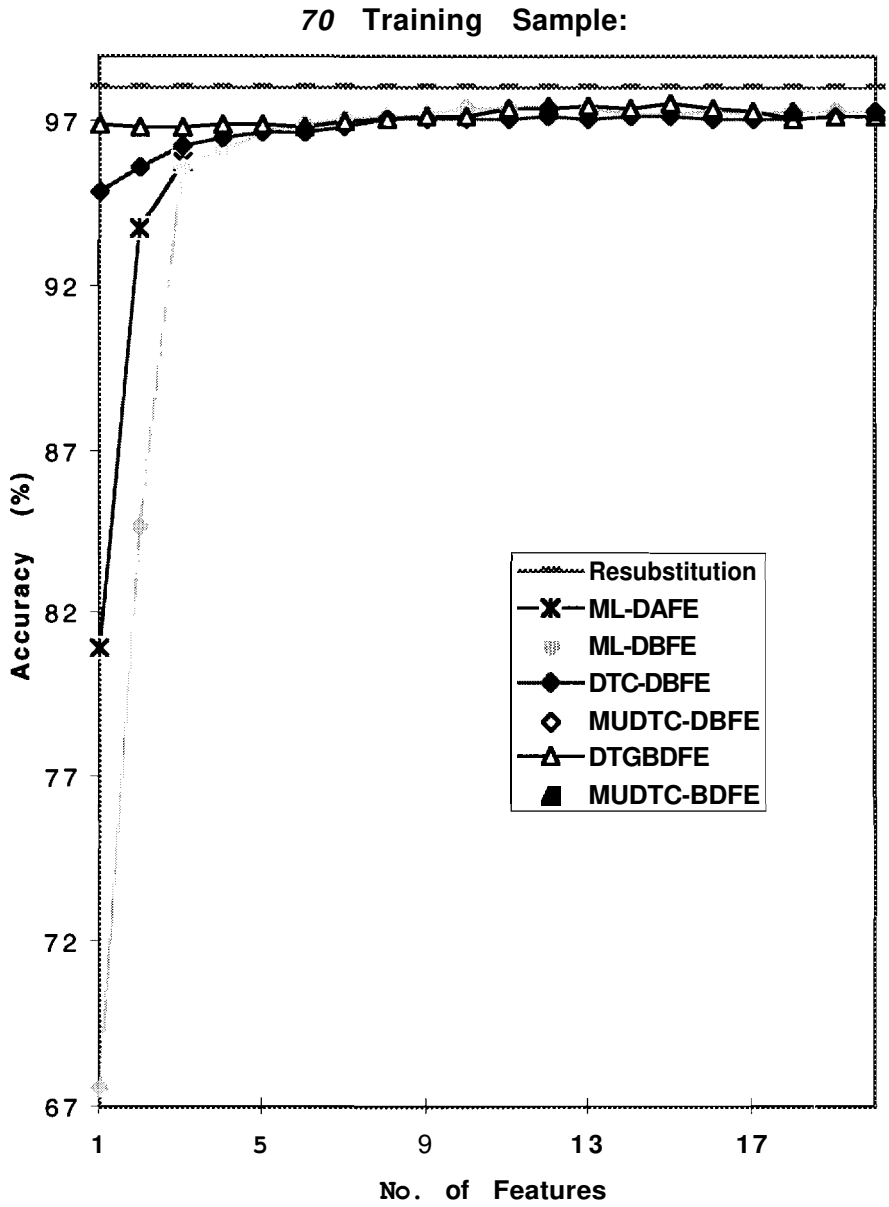


Figure 4.13 Classification Results for 70 Training Samples (FSS Data)

Using 70 training samples, all methods achieve the best result at higher dimensionality. This demonstrates that when there are many training samples, the Hughes effect does not exist. Among all methods, the tree classifier with Bhattacharyya distance feature extraction (DTC-BDFE) method achieves the best result using smaller number of features.

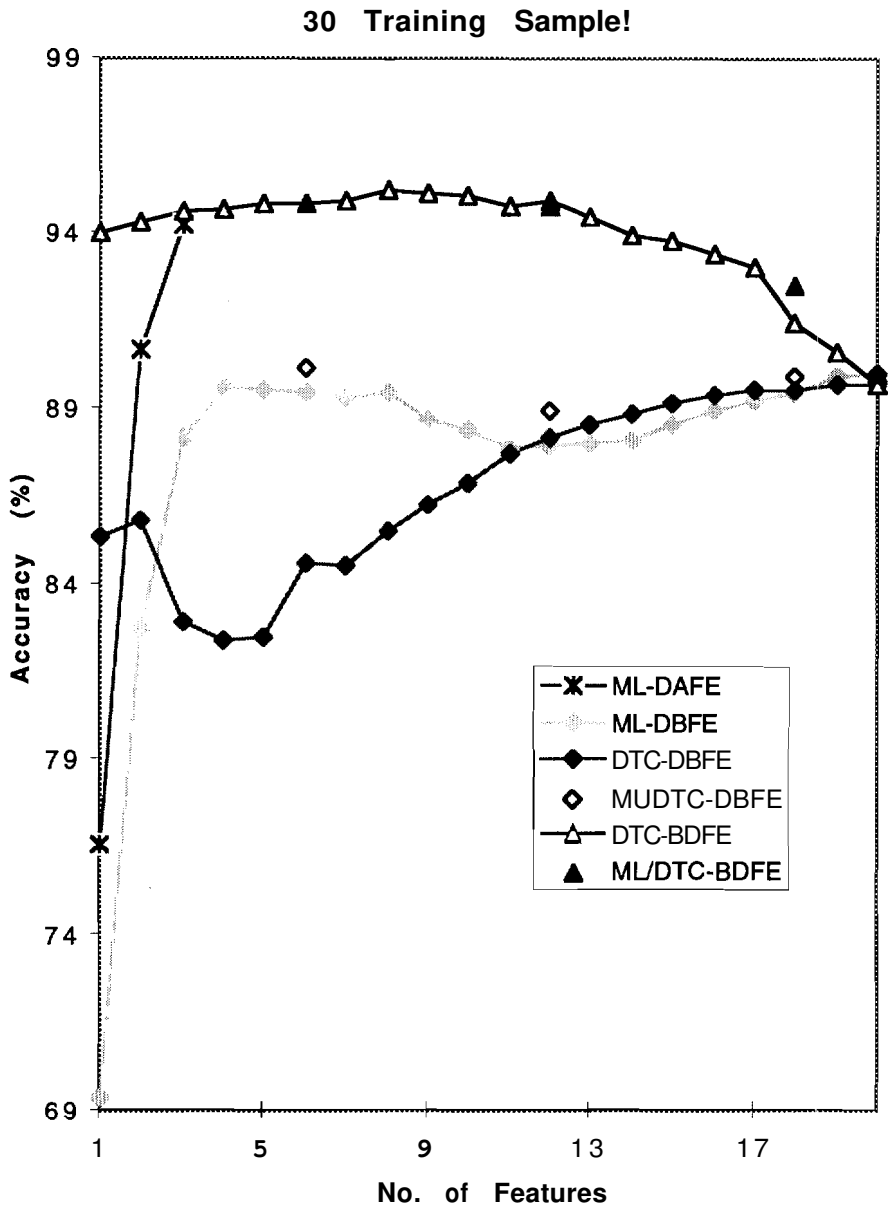


Figure 4.14 Classification Results for **30** Training Samples (FSS Data)

Using only **30** training samples for **20** dimensional data, the results start to display the Hughes effect. The best results are obtained using the binary tree classifier with Bhattacharyya distance feature extraction (DTC-BDFE) method and the single-stage classifier using features generated from the binary tree using BDFE.

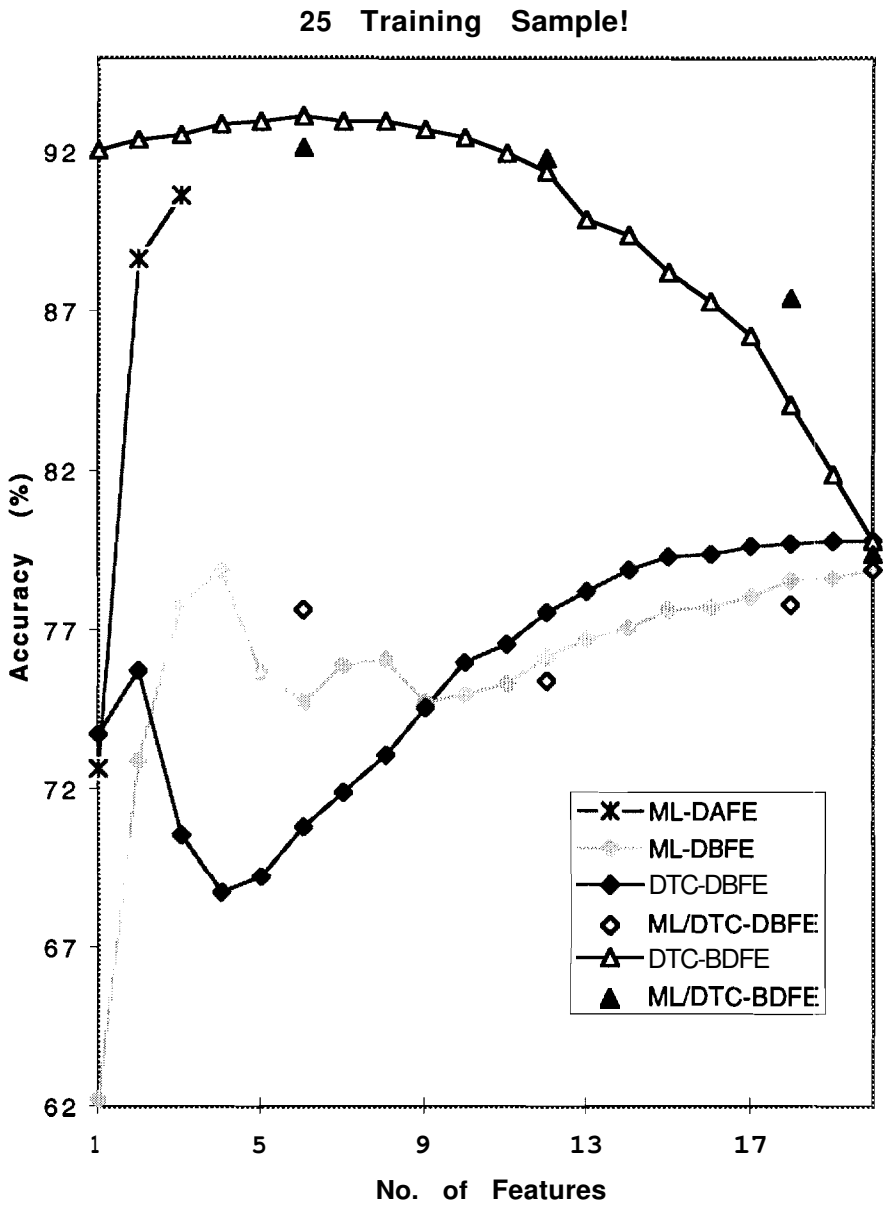


Figure 4.15 Classification Results for 25 Training Samples (FSS Data)

With only 25 training samples, the Hughes effect becomes more severe. As in the cast: of **30** training samples, the best results are still obtained by DTC-BDFE. It again shows the relative robustness of Bhattacharyya distance feature extraction (BDFE) as compared to decision boundary feature extraction (DBFE) method in the case of small training set size.

Experiment 4.3

In previous experiments, the methods are tested for the case in which the number of training samples is greater than the dimensionality. In this experiment, the proposed binary tree classifier is tested for numerous class some of which have fewer training samples than the number of spectral channels. Six classes are chosen from the FSS data collected on August 16, 1978. The number of labeled and training samples are given in Table 4.5. A total of 20 spectral bands are selected from the original 60 spectral bands. As shown in the table, the training samples of class "Alfalfa" and "Barley" are as few as 11 and 20 respectively, while some other classes have many more training samples. In this case, DBFE is no longer applicable. Therefore, DAFE is used instead for the single-stage quadratic classifier and only BDFE is used for the binary tree classifier. Since the number of training samples for each class varies quite significantly, the number of features at each node should also differ depending on the available training samples. The optimal number of features for each node is difficult to determine. Therefore, as a rule of thumb, when one or both of the classes contain training set size which is less than the dimensionality, only a single feature is generated. And when the average covariance estimates at each node is singular, BDFE cannot be applied and thus the Euclidean distance classifier is adopted instead. On the other hand, when there are many training samples, more features can be generated at each node as defined by the user. In this experiment, the number of features selected for BDFE is 10. Table 4.6 shows the performance comparison for the classifiers.

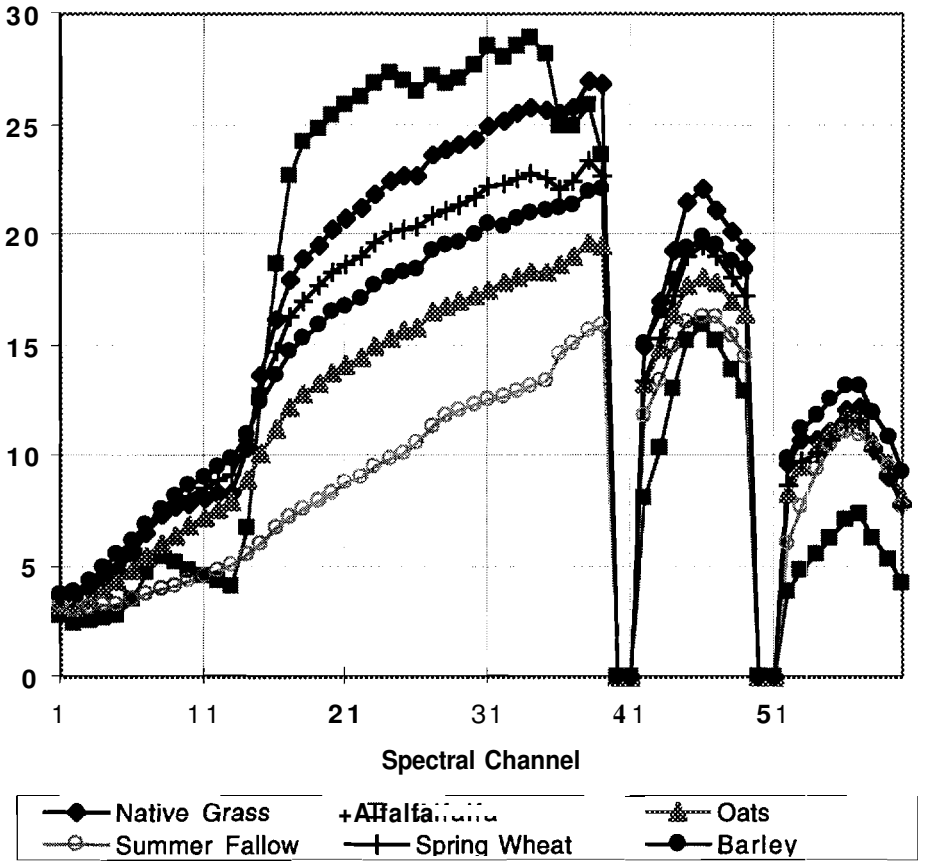


Figure **4.16** Mean Graph of FSS Data with Six Classes of Varying Size

Table **4.5**
Class Description for FSS Data in Figure **4.16**

Class Names	**No. of Labeled Samples**	**No. of Training Samples**
Native Grass	212	42
Alfalfa	59	11
Oats	165	**33**
Summer Fallow	216	**43**
Spring Wheat	464	92
Barley	103	20
Total samples	1278	252

Table 4.6
Classification Results for FSS Data with Varying Size

Class Names	Classification Accuracy (%) Resubstitution	ML-DAFE	DTC-BDFE
Native Grass	95.88	92.94	91.76
Alfalfa	91.67	43.75	79.17
Oats	78.79	77.27	65.91
Summer Fallow	90.17	89.02	89.02
Spring Wheat	77.42	72.58	82.53
Barley	87.95	67.47	68.67
Ave Accuracy (%)	**86.98**	**73.84**	**79.51**

The resubstitution accuracy presents the most optimistic performance by using all labeled samples for training and testing as well. It was shown that by using the tree classifier, the performance improves by 6% from the single-stage classifier. The performance of the class Alfalfa which has only 11 training samples has increased from 43.75% to 79.17% using the binary tree classifier.

4.6 Summary

A binary tree design for classification and feature extraction has been proposed in this work. As a classifier, the divide-and-conquer approach of the proposed binary classifier has been shown to mitigate the Hughes phenomenon when used with a proper feature extraction method. The experimental results show that when the design set size is large, the two-class binary tree classifier using decision boundary feature extraction (DBFE) gives better performance at small number of features. Also, DBFE for more than two classes is not optimal. However, since in this case the Hughes phenomenon does not exist, all methods give the same performance at full dimensionality. On the other hand, when the training set size is moderate or small compared to the number of features, the binary tree classifier with Bhattacharyya distance feature extraction (BDFE) has better results by using fewer features at each node. Also, BDFE does not suffer as much as DBFE due to limited training set size even though both methods utilize covariance information. However, when the covariance matrices are poorly estimated, the single-stage classifier using discriminant analysis feature extraction (DAFE) is more reliable since DAFE utilizes only the mean information.

A heuristic rule for employing different classification rules and :Feature extraction methods at each node has also been proposed to process data with varying sample size for each class. When the total number of training samples is close to or less than dimensionality, the Euclidean distance classifier is used instead. Experimental result shows that the multi-stage classifier has higher classification accuracy than the single-stage approach. However, the types of classification rules and feature reduction methods are by no means limited to the ones mentioned in this work. The feature extraction methods mentioned in this chapter depend on the quality of the covariance estimate. Methods which can deal with fewer training samples than dimensionality are needed. Projection pursuit [44] may be used to reduce dimensionality prior to performing feature extraction. Future research can also be directed towards finding the best approach to deal with different types of data, thus making the tree classifier more automated. In addition, a method to determine the optimal number of features at each node should be a challenging research problem. A major disadvantage of using binary tree generated features is that the number of features is generated at a multiple of the number of non-repeating binary decisions. Therefore, a method which can select the most significant features from the collection of tree extracted features should be investigated.

CHAPTER 5: CONCLUSIONS

This thesis has presented several solutions to circumvent the problems of classification associated with high dimensionality. These problems have become more prevalent in remote sensing due to the increase in spectral and spatial resolution of the new sensors with higher dynamic range. Although more classes become spectrally separable, unfortunately, when the number of spectral features increases, the classification performance deteriorates if the number of training samples remains fixed. This has been widely known as the Hughes phenomenon. The problem of the Hughes phenomenon is attributed to the fact that more training samples are required to specify the decision boundary for classification at higher dimensionality. In the case of Gaussian maximum likelihood classification, the decision boundary is defined by the mean vector and covariance matrix. The variances of these estimates increase as the ratio of training sample size to the dimensionality decreases. Therefore, the estimation of these parameters becomes crucial for classification performance. In this thesis, methods have been proposed to deal with these problems and shown to improve classification accuracy.

In Chapter 2, the problem of limited training set size is addressed by including unlabeled samples for parameter estimation. The use of unlabeled samples in addition to training samples can also be viewed as estimating parameters under the mixture model. The maximum likelihood estimates for the mixture model are obtained via the expectation maximization (EM) algorithm. Unfortunately, the EM algorithm is sensitive to the presence of statistical outliers. As a result of increased spectral and spatial resolution, more classes are spectrally separable with varying sample sizes. Some classes with few samples may be difficult to identify and may form statistical outliers. Thus, a robust version of the EM algorithm was proposed. This robust EM (REM) algorithm reduces the influence of statistical outliers by assigning less weight to samples further away from the main body of distributions. Experimental results have shown that without

statistical outliers, both the EM and REM algorithms perform better than the maximum likelihood (ML) parameter estimation using training samples alone. They can even mitigate the Hughes phenomenon if there are enough unlabeled samples available. In the presence of outliers, the REM algorithm achieves better classification accuracy than the EM and ML methods. Despite the promising results, the mixture model has to be used with caution. In addition to the presence of statistical outliers, the performance of the mixture model is also affected by the number of unlabeled samples available and the initial conditions. It has been shown experimentally that without a sufficient number of unlabeled samples, the performance of the EM and REM algorithms is as poor as using training samples alone at high dimensionality. It was assumed in this work that the training samples provide reasonable initial parameter estimates for the iterations. Without a good initial estimate, the convergence to the optimal solution is not guaranteed. Also, if the number of training samples is less than the dimensionality, the covariance matrix becomes singular and hence the iterative equations cannot be applied. In this case, either a feature reduction method must be used or a non-singular covariance estimate must be obtained from the training samples by imposing some constraint on its form. The latter approach is addressed in Chapter 3.

The inverse of a covariance matrix becomes ill- or poorly-posed if the training set size is small compared to dimensionality. Conventionally, the stabilization of the covariance estimate has been accomplished by regularization which tends to reduce the variance of the estimate at the expense of increased bias. This method can also be viewed as a compromise between the linear and quadratic classifiers. In Chapter 3, a regularization method under the Bayesian setting has been proposed.. The proposed Bayesian leave-one-out covariance (bLOOC) estimation method was shown to have better performance than other methods when the training set size reflects the true priors of the classes. This is particularly true for remote sensing applications since more training samples are usually selected for larger classes. When used in conjunction with discriminant analysis feature extraction (DAFE), the proposed covariance estimation was demonstrated to circumvent the limited training set size problem.

Since the leave-one-out likelihood is used as the criterion for these estimators, it has the drawback of not being directly related to class separability, and subsequently the classification accuracy. Therefore, some smooth loss function derived from the class separability is recommended for future work. Also, since decision boundary feature extraction (DBFE) is not suitable for small training sample size and DAFE method does

not work well when the classes have similar mean values, an alternative feature extraction or classification methods need to be explored. A solution is proposed in Chapter 4 using a two-class binary tree with a feature extraction method based on maximizing Bhattacharyya distance.

In Chapter 4, a two-class binary tree design has been proposed to function as a classifier and a feature extraction method. One advantage of using a divide-and-conquer method is that fewer features can be used at each node. Also, different decision rules can be applied depending on the training samples available at the local node. By using two classes instead of two subgroups of classes for node decision, the problem of merging can be avoided. Since the classes defined for remote sensing applications are assumed to be normally distributed, the two-class binary decision is basically a Gaussian maximum likelihood classification. The binary structure is also desirable for obtaining optimal features based on two normal classes using either decision boundary feature extraction (DBFE) or Bhattacharyya distance feature extraction (BDFE) methods. These features can then be collectively used in a single-stage classifier. Experimental results have shown that BDFE is more robust than DBFE for limited training set size. This is due to the fact that BDFE is a suboptimal approach with the main emphasis on the mean difference between two classes and with additional features based on covariance information. In contrast, DBFE relies on the decision boundary which is sensitive to the accuracy of mean and covariance estimates. DBFE for multi-class problems has also been shown to be suboptimal. The discriminant analysis feature extraction (DAFE) method is mainly based on the class mean information. Therefore, BDFE can be considered as a compromise between DAFE and DBFE. Unfortunately, BDFE also uses covariance estimate and thus cannot be applied when the training set size is smaller than the dimensionality.

To deal with the case in which some classes have fewer training samples than the dimensionality, a heuristic rule for employing different classification rules and feature extraction methods at each node was proposed. When one (or both) of the classes in each node has training samples less than dimensionality, but the combined number of training samples is greater than dimensionality, the linear classifier with their average covariance estimate may be applied. When the total number of training samples are close to or less than the dimensionality, the Euclidean distance classifier is used instead. Experimental results have confirmed the benefit of the binary tree classifier using different classification rules for each node based on the locally available training samples. Despite

the promising results, more work remains to be done. For future work, a thorough study on the types of classifiers and feature extraction methods suitable for various types of data is recommended. In particular, feature extraction methods for two classes with fewer training samples than dimensionality are needed. In this case, projection pursuit may be used to reduce dimensionality prior to performing feature extraction. In addition, methods to decide the optimal number of features for each node and for selecting most significant features from a collection of tree generated features should be explored.

BIBLIOGRAPHY

[1] A. Papoulis, Probability, Random Variables, and Stochastic Processes. McGraw-Hill, 1984.

[2] G.F. Hughes, "On the mean accuracy of statistical pattern recognizers," IEEE Trans Info. Theory, vol. IT-14, No. 1, pp. 55-63, 1968.

[3] K. Fukunaga, "Effects of sample size in classifier design," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 11, No. 8, pp. 873-885, 1989.

[4] G.J. McLachlan, K.E. Basford, Mixture Models, Inference and Applications to Clustering. Marcel Dekker, Inc., 1988.

[5] K. Pearson, "Contributions to the mathematical theory of evolution," Phil. Trans., vol. A 185, pp. 71-110, 1894.

[6] C.R. Rao, "The utilization of multiple measurements in problems of biological classification," J. R. Statist. *Soc.* A, vol. 145, pp. 285-312, 1948.

[7] W.Y. Tan and W.C. Chang, "Some comparisons of the method of moments and the method of maximum likelihood in estimating parameters of a mixture of two normal densities," J. Amer. Statist. Assoc., vol. 67, pp. 702-708, 1972.

[8] A.P. Dempster, N.M. Laird, D.B. Rubin, "Maximum likelihood estimation from imcomplete data via EM algorithm," J. R. Statist. *Soc.*, vol. B 39, pp. 1-38, 1977.

[9] R.A. Redner, H.F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM* Review, vol. 26, No. 2, pp. 195-239, 1984.

[10] C.F.J. Wu, "On the convergence properties of the EM algorithm," The Annals of Statistics, vol. 11, No. 1, pp. 95-103, 1983.

[11] B. Shahshahani, Classification of Multi-spectral Data by Joint Supervised-Unsupervised Learning, Purdue University, West Lafayette, IN, TR-EE 94-1, Jan. 1994.

[12] A.P. Dempster, N.M. Laird, D.B. Rubin, "Maximum likelihood estimation from imcomplete data via EM algorithm," J. R. Statist. *Soc.*, vol. B39, pp. 1-38, 1977.

[13] P.J. Huber, "Robust estimation of a location parameter," Ann. Math. Statist., vol. 35, pp. 73-101, 1964.

[14] R.A. Maronna, "Robust M-estimators of multivariate location and scatter," Ann. Statist., vol. 4, pp. 51-67, 1976.

[15] N.A. Campbell, "Mixture models and atypical values," Math *Geol.*, vol. 16, pp. 465-477, 1984.

[16] Y. Jhung, Bayesian Contextual Classification of *Noise-Contaminated* Multi-variate Images, Ph.D. Dissertation, School of Electrical Engineering, Purdue University, 1994.

[17] F.A. Graybill, Matrices With Applications In Statistics. Belmont: Wadsworth Inc., 1983.

[18] S.J. Delvin, R. Gnanadesikan and J.R. Kettenring, "Robust estimation of dispersion matrices and principal components," J. Amer. Statist. Assoc., vol. 76, pp. 354-362, 1981.

[19] S.P. Lin and M.D. Perlman, "A Monte Carlo comparison of four estimators of a covariance matrix," Multivariate analysis--VI : Proceedings of the Sixth International Symposium on Multivariate Analysis, P.R. Krishnaiah, ed., Amsterdam: Elsevier Science Pub. Co., 1985, pp. 411-429.

[20] P.W. Wahl and R.A. Kronmall, "Discriminant functions when covariances are equal and sample sizes are moderate," *Biometrics*, vol. 33, pp. 479-484, 1977.

[21] S. Marks and O.J. Dunn, "Discriminant functions when the covariance matrices are unequal," Journal of the American Statistical Association, vol. 69, pp. 555-559, 1974.

[22] F. O'Sullivan, "A statistical perspective on ill-posed inverse problems," Statistical Science, vol. 1, pp. 502-527, 1986.

[23] J.H. Friedman, "Regularized Discriminant Analysis," Journal of the American Statistical Association, vol. 84, pp. 165-175, March 1989.

[24] W. Rayens and T. Greene, "Covariance pooling and stabilization for classification," Computational Statistics and Data Analysis, vol. 11, pp. 17-42, 1991.

[25] J.P. Hoffbeck and D.A. Landgrebe, Classification of High Demensional Multispectral Data, Purdue University, West Lafayette, IN, TR-EE 95-14, May, 1995, pp. 43-71.

[26] K.K. Dey and C. Srinivasan, "Estimation of a covariance matrix under Stein's loss," Annals of Statistics, vol. 13, No. 4, pp. 1581-1591, 1985.

[27] N.A. Campbell, "Shrunken estimator in discriminant and canonical variate analysis," Annals of Statistics, vol. 29, pp. 5-14, 1980.

[28] T.W. Anderson, An Introduction to Multivariate Statistical Analysis. 2nd Ed., New York: John Wiley & Sons, 1984.

[29] B.W. Silverman, Density Extimationfor Statistics and Data *Analysis*. New York: Chapman and Hall, 1986.

[30] K. Fukunaga, Introduction to Statistical Pattern Recognition. 2nd Ed., Boston: Academic Press, 1990.

[31] G.H. Golub, and D.F. Van Loan, Matrix Computations. 2nd. Ed., Baltimore: Johns Hopkins University Press, 1989, p. 51.

[32] C. Lee and D.A. Landgrebe, "Feature extraction based on decision boundaries," IEEE Transaction of Pattern Analysis and Machine Intelligence, vol. 15, no. 3, pp. 388-400, April 1993.

[33] R. L. Kettig and D. A. Landgrebe, "Classification of multispectral image data by extraction and classification of homogeneous objects," *IEEE* Trans. Geosci. Electro., vol. GE-14, No. 1, pp. 19-26, Jan., 1976.

[34] G.F. Hughes, "On the mean accuracy of statistical pattern recognizers," IEEE Trans. Info. Theory, vol. IT-14, No. 1, pp. 55-63, 1968.

[35] R. Safavian and D.A. Landgrebe, "A survey of decision tree classifier methodology," IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, No. 3, pp. 660-674, 1991.

[36] J.A. Richards, Remote Sensing Digital Image Analysis. Springer-Verlag, 1993.

[37] C. Wu, D. Landgrebe, and P. Swain, The Decision Tree Approach To Classfication, Purdue University, West Lafayette, IN, TR-EE 75-17, 1975.

[38] J.R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1, pp. 81-106, 1986.

[39] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, Classification and Regression Trees. Belmont, CA: Wadsworth Int., 1984.

[40] K.C. You and K.S. Fu, "An approach to the design of a linear binary tree classifier," in Proc. 3rd Symp. Machine Processing of Remotely Sensed Data, Purdue University, W. Lafayette, IN, 1976.

[41] B. Kim and D.A. Landgrebe, "Hierarchical classification in high dimensional, numerous class cases," IEEE Transactions on Geoscience and *Remote* Sensing, vol. 29, No. 4, pp. 518-528, July, 1991.

[42] G. Landeweerd, T. Timmers, E. Gersema, M. Bins and M. Halic, "Binary tree versus single level tree classification of white blood cells," *Pattern Recog.*, vol. 16, pp. 571-577, 1983.

[43] J.P. Hoffbeck and D.A. Landgrebe, "Covariance matrix estimation and classification with limited training data" IEEE Transactions on *Pattern* Analysis & Machine Intelligence, vol 18, No. 7, pp. 763-767, July1996.

[44] L.O. Jimenez and D.A. Landgrebe, High Dimensional Feature Reduction Via Projection Pursuit, Purdue University, West Lafayette, IN, TR-ECE 96-5, 1995.