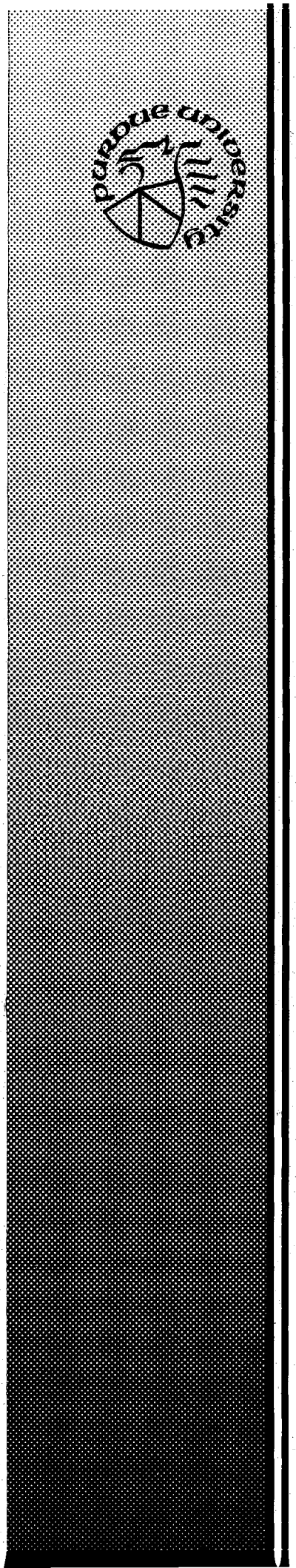
6-1-1990

Hierarchical Classification in High Dimensional, Numerous Class Cases

Byungyong Kim
Purdue University

D. A. Landgrebe
Purdue University



Hierarchical Classification in High Dimensional, Numerous Class Cases[1]

Byungyong Kim
D. A. Landgrebe

TR-EE 90-47
June 1990

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

[1]The work was sponsored in part by NASA under Grant NAGW-925.

HIERARCHICAL CLASSIFICATION IN HIGH DIMENSIONAL, NUMEROUS CLASS CASES[1]

Byungyong Kim

D. A. Landgrebe

TR-EE 90-47

June 1990

School of Electrical Engineering

Purdue University

West Lafayette, Indiana 47907

[1] The work was sponsored in part by NASA under Grant NAGW-925.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

As progress in new sensor technology continues, increasingly high resolution imaging sensors are being developed. HIRIS, the High Resolution Imaging Spectrometer, for example, will gather data simultaneously in 192 spectral bands in the 0.4 - 2.5 micrometer wavelength region at 30 m spatial resolution. AVIRIS, the Airborne Visible and Infrared Imaging Spectrometer, covers the 0.4 - 2.5 micrometer in 224 spectral bands. These sensors give more detailed and complex data for each picture element and greatly increase the dimensionality of data over past systems.

In applying pattern recognition methods to remote sensing problems, an inherent limitation is that there is almost always only a small number of training samples with which to design the classifier. Both the growth in the dimensionality and the number of classes is likely to aggravate the already significant limitation of training samples. Thus ways must be found for future data analysis which can perform effectively in the face of large numbers of classes without unduly aggravating the limitations on training.

A set of requirements for a valid list of classes for remote sensing data is that the classes must each be of informational value (i.e. useful in a pragmatic sense) and the classes be spectrally or otherwise separable (i.e., distinguishable based on the available data). Therefore, a means to simultaneously reconcile a property of the data (being separable) and a property of the application (informational value) is important in developing the new approach to classifier design. In this work we propose decision tree classifiers which have the potential to be more efficient and accurate in this situation of high dimensionality and large numbers of classes. In particular, we discuss three methods for designing a decision tree classifier, a top down approach, a bottom up approach, and a hybrid approach.

Also, remote sensing systems which perform pattern recognition tasks on high dimensional data with small training sets require efficient methods for feature extraction and prediction of the optimal number of features to achieve minimum classification error. Three feature extraction techniques are implemented. Canonical and extended canonical techniques are mainly dependent upon the mean difference between two classes. An autocorrelation technique is dependent upon the correlation differences.

The mathematical relationship between sample size, dimensionality, and risk value is derived. It is shown that the incremental error is simultaneously affected by two factors, dimensionality and separability. For predicting the optimal number of features, it is concluded that in a transformed coordinate space it is best to use the best one feature when only small numbers of samples are available. Empirical results indicate that a reasonable sample size is six to ten times the dimensionality.

CHAPTER 1. INTRODUCTION

1.1 Preliminary Remarks.

As the progress in new sensor technology continues, increasingly high resolution imaging sensors are being developed. For example, HIRIS, the High Resolution Imaging Spectrometer, will have 192 spectral bands which gather data simultaneously in the 0.4 - 2.5 micrometer wavelength region at 30 m spatial resolution. AVIRIS, the Airborne Visible and Infrared Imaging Spectrometer, covers the 0.4 - 2.5 micrometer in 224 spectral bands. These sensors give more detailed and complex data for each picture element and increase the dimensionality of data. The growth of dimensionality and the higher spectral resolution provides the opportunity to identify a larger number of classes within a scene than in the past.

For high dimensional, multi-class pattern recognition problems, a decision tree classifier (hereafter referred to as DTC) instead of a single layer classifier is the most appropriate scheme, because a DTC divides the complex global decision-making process in high dimensional spaces into a number of simpler and local decisions at various levels of the tree. As a result, proper subsets of features at each node can be chosen to improve the classification accuracy while at the same time possibly reducing the required amount of computation.

A decision tree is a means for showing the relationship of intermediate decisions in a complex decision process in order to reach a final decision. Decision trees consist of three parts, the root node, intermediate nodes, and terminal nodes. The root node has only descendent nodes while terminal nodes each have only a unique ascendent node. Intermediate nodes have both descendent and ascendent nodes. If an acyclic graph is defined to be one which contains no cycle, a tree is a connected acyclic graph. A tree thus defined has the property that a path from the root node to any given node is unique. Single layer classifiers test the degree of membership of the unknown sample against all classes and finally assign the unknown sample to one of those classes. Decision tree classifiers test the degree of membership of the unknown sample against subgroups which contain several classes at the intermediate nodes and assign the unknown sample to one of the subgroups. If the subgroup is not one of the final classes, classification procedures are continued until reaching the terminal nodes.

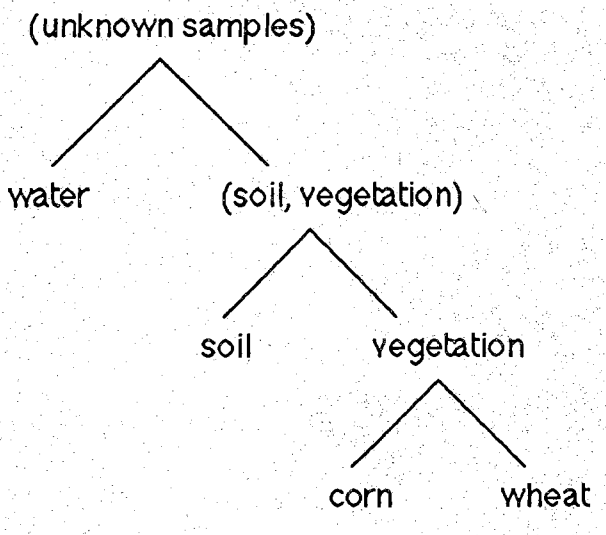


Figure 1.1 A Simple Decision Tree

The primary objective of this work is to develop a DTC design procedure which leads to a DTC that is more efficient and more accurate in case of high dimensionality, limited training set sizes, and large numbers of classes.

A second objective of this thesis is to find means to extract optimal features and predict the optimal number of features in remote sensing situations. The estimation of the optimal number of features is an essential part of the DTC design process as the dimensionality grows. Probability of correct classification is closely related to design sample size, dimensionality, and noise.

1.2 A Review of Related Work

The DTC has been studied for a number of applications. Examples are remote sensing, character recognition, blood cells classification. The presumed advantages of the DTC are reported as computational efficiency and improvement in classification accuracy with finite samples. Disadvantages of DTC are complexity, error accumulation, and design difficulty for an optimal DTC.

The study of the DTC may be categorized into three phases: the design phase, the feature selection phase, and the decision rule. However, to be truly optimal, these are not sequential but must be simultaneously accomplished.

For the purpose of ideal design, Wu et al. [1] developed an evaluation function which consists of a computation time factor and classification accuracy. The classification error is estimated by assuming that the direct descendent nodes are single layer classifiers. Kanal [21] defined two types of admissible search strategies

to obtain the optimal decision tree structure, namely S-admissible and B-admissible which use a cost of path and risk function in a state space graph model. In an S-admissible search, the risk function depends only on the features measured along the path from the initial state to final state. In a B-admissible search, the risk function depends on all features, not just those on the path to the final state. Using these functions, it is impossible to evaluate successfully every combination of tree structure to determine the overall optimal tree classifier.

For practical design purposes, minimizing the classification error is frequently pursued at each node, although that procedure does not guarantee that the overall structure will be optimal. For the same reason, computational efficiency is often considered as an independent performance index.

Wu et al [1] and Swain and Hauska [8] suggested a histogram approach which plotted means and covariances for all classes for each feature. A suitable boundary was sought to separate the subgroups which might be homogeneous with a particular feature. However, since this approach uses only one feature, the inter-relationships with other features are disregarded.

You and Fu [13] designed binary trees by splitting a set of classes into two non-overlapping subgroups at every node. The two subgroups are found by comparing the measure of separability for different pairs of subgroups over various subsets of feature space with fixed numbers of features. To reduce the possible combinations of tree structures, they suggested two restrictions. The first restriction was to limit the number of features selected at each node. For the sake of accuracy, the second restriction was the size of the tolerable error probability at each node. However, if the numbers of classes are large, the number of possible tree structures is still large. In the case of limited training samples, the measure function itself may be poorly estimated.

Sethi and Sarvarayudu [27] suggested a tree design based upon the mutual information obtained about the pattern classes from the observation event X, which can be written as

$$I(C;X) = \sum_{i=1}^{2} \sum_{j=1}^{2} P(C_i, X_j) \log_2 \left\{ \frac{P(C_i|X_j)}{P(C_j)} \right\} \qquad (1.1)$$

where C represents the set of pattern classes, C_1 and C_2 having a priori probabilities $p(C_1)$ and $p(C_2)$. $p(C_i, X_j)$ is the joint probability of occurrence of C_i and X_j and $p(C_i|X_j)$ is the probability that the observation comes from class C_i given the outcome X_j of event X. Let P_e be the probability of error allowed in recognition; then the following inequality determines the limit on the equivocation $H(C|X)$ of C with respect to X;

$$H(C|X) \leq H(P_e) + P_e \log_2 (n-1) \qquad (1.2)$$

where $H(P_e)$ is the error entropy and m is the number of pattern classes. Since the average mutual information can also be written as,

$$I(C;X) = H(C) - H(C|X) \tag{1.3}$$

we obtain the following inequality,

$$\begin{aligned} I(C;X) &\geq H(C) - H(P_e) - P_e \log_2 (n-1) \\ &= -\sum_{i=1}^{2} P(C_i)\log_2 P(C_i) + P_e \log_2 P_e + (1-P_e)\log_2(1-P_e) - P_e \log(n-1) \\ &\equiv I_{min} \end{aligned} \tag{1.4}$$

Equation (1.4) thus relates the probability of error and the corresponding minimum value of average mutual information required for a recognition process. Their method generated a partitional tree for a specified probability of error by maximizing the amount of average mutual information gain or minimizing the average error of recognition for the given size of tree. The above algorithm was implemented by a non-parametric procedure.

Casey and Nagy [30] developed a binary tree for optical character recognition using an information theoretic approach. The effectiveness of a node-by-node design scheme is highly dependent on the rule by which pixels are evaluated for assignment to a given node. The first pixel to be tested is predetermined for the root node. A measure based on entropy is used for a pixel selection criterion. The rule employed for pixel selection is to choose the pixel that minimizes entropy, i.e, the one that maximizes the information gain. A priori class probabilities and class conditional frequencies of individual pixels are estimated from labeled samples. Their approach is a special case of binary tree character recognition.

Landeweered et al. [29] suspected that binary tree classifiers improved the correct recognition rate compared with the application of single layer classifiers. A binary tree was constructed in a stepwise, bottom-up fashion, such that in each step the two classes with the smaller Mahalanobis distance were merged to form a new group. Since their binary tree classifiers might not be the optimal tree structure in some sense, they did not take a look at the improvement of correct recognition rate compared with the result of the single layer classifiers in some cases.

For the feature selection phase, in the ideal case, feature selection should be simultaneously considered with decision tree design parameters. Practically, Swain and Hauska [8] chose a feature selection criterion based on pairwise separability over all pairs of classes after designing the decision tree classifiers.

Muasher and Landgrebe [2] experimentally studied an effective feature ordering technique in cases where the the number of training samples was limited in classifying multivariate two-class normal distributions.

For the decision phase, parametric and non-parametric procedures may be used. Maximum likelihood Gaussian classifiers are usually used at each node in a parametric approach.

M. W. Kurzynski [28] dealt with the decision rules of tree classifiers for performing the classification at each non-terminal node, under the assumption of complete probabilistic information. For a given tree structure and feature subsets to be used, the optimum decision rules were derived which minimized the overall probability of misclassification.

When the classifier design is to be based upon finite sets of samples from the various classes and features, the estimation of the class-conditional densities of the measurement vector is necessarily a key step. These estimates are then used for the classification. One might initially assume that as the dimensionality of the sample vectors is increased, classification accuracy would generally increase because the information available is increased. If the added sample vector dimensionality does not contribute in any way to classification results, one might suppose that the classification error rate should at least stay the same. In practice, the performance of the classifier based on estimated parameters improves up to a certain point, then begins deteriorating as further features are added.

In a Bayesian formulation, some priori densities on the parameter of P_i are assumed, and class-conditional densities $P_i(x)$ can be calculated. On the other hand, one can arrive at maximum likelihood estimates of the density function. After obtaining the maximum likelihood estimates of the unknown parameters, $P_i(x)$ can be obtained by substituting those estimated values instead of true parameters. Therefore, the classification performance depends on the estimation procedure, and problems pertaining to the relationship between dimensionality and sample size are in the context of the method of estimation.

Hughes [22] considered the behavior of a finite sample Bayesian classifier with respect to increasing measurement complexity. Even if a Bayesian procedure is used which is optimal in the sense that it minimizes the probability of misclassification, one could get into trouble by using too many measurements when the number of training samples is small. It was shown that if the measurements are independent and binary [55], or first order nonstationary Markov dependent and binary [56], then there wil be no peaking of performance in the Bayesian context with respect to the measurement complexity for a finite sample size [57].

When the approach to classifier design is non-Bayesian e.g. parameters are estimated by maximum likelihood methods, peaking effects occur. While the original Bayes' rule is optimal, the decision rule that results from substituting the maximum likelihood estimates of the parameters is no longer optimal. The errors caused by the non-optimal use of added information overrides the advantages of extra information. Foley [11] investigated the design set error rate for a two class problem with multivariate normal distributions, and derived it as a function of the

sample size per class and dimensionality. The design set error rate was compared to both the corresponding Bayes error rate and test set error rate. It was shown that the design-set error rate is biased below the true error rate and the test-set error rate is biased above the true error rate of a classifier when the ratio of sample size to feature size is small. Jain [58] showed that when features have multinomial or univariate Gaussian distributions, the estimate of the Bhattacharyya distance is biased and consistent. The bias and the variance of the estimate are not only a function of the number of training samples but also depend on the true parameters of the densities.

1.3 Problem Statement

In applying pattern recognition methods in remote sensing problems, an inherent limitation is that there is almost always only a small number of training samples with which to design the classifier. The growth in both the dimensionality and the number of classes is likely to aggravate the already significant limitation of training samples. Thus, ways must be found for future data analysis which can perform effectively in the face of large numbers of classes without unduly aggravating the limitations on training.

Until now, decision tree classifiers for remote sensing have been designed by only considering one property of the data, that of separability. In that case, the final decisions do not necessarily coincide with classes of informational value. In addition to being adequately exhaustive, the requirements for a valid list of classes for remote sensing data are:

1. The classes must each be of informational value (i.e. useful in a pragmatic sense).

2. The classes must be spectrally or otherwise separable (i.e., distinguishable based on the available data).

Therefore, a means to reconcile a property of the data (being separable) and a property of the application (informational value) is the main objective in developing a new approach to tree design.

In designing the classifier, one would like to know how many features one should use to maximize the classification accuracy. The number of features, the number of samples, and the correct classification accuracy are related in a complex fashion. In remote sensing, the reflected and emitted electromagnetic energy of each pixel of a scene in a number of wavelength bands is measured by a multispectral remote sensor system mounted on board an aircraft or spacecraft. The output of the sensor system for a given scene pixel may be represented as a point in a multidimensional space. The number of training samples is frequently limited because it is expensive to accumulate the information by which to label many samples. In the case of limited training samples and multidimensional space, the estimates of the first and

second order statistics cannot accurately depict all the information which is contained in the data. In particular, the estimate of the covariance matrix may be poor. Therefore how to relate the inaccuracy of estimate with classification error directly is another objective of this work.

1.4 Outline of the Report

To obtain an optimal DTC, one must consider three components simultaneously, which are the tree structure, feature extraction, and the decision rule. The chosen criterion for tree structure results from the kind of decision rule selected at non-terminal nodes for the specific application. Also, a criterion for feature extraction will be chosen based upon the decision rule. Therefore, once a decision rule is determined, both a criterion for tree structure and for feature extraction can be chosen to obtain the best-performing tree structure and best classification results.

In chapter 2, three methods for tree design, the top down, bottom up, and hybrid approaches, will be discussed. In a top down approach, the entire feature space is sequentially subdivided into increasingly local decision regions. Suppose that we decide to use the maximum likelihood rule as the decision rule; we may then use clustering at each non-terminal node to divide the data into appropriate subgroups, and we might select the sum of squared error criterion as a clustering criterion for the tree structure. Since there is no information about the covariances, minimization of Euclidean distance is used.

In a bottom up approach, just the opposite procedure from the top down method is pursued. Joining of local decision regions to make increasingly global decision regions is used. Since we have the estimated mean and covariance of informational classes, the Bhattacharyya distance may be chosen as the criterion for the tree structure. In the hybrid approach, top down and bottom up approaches are sequentially used to achieve the combined effect of the two approaches. The normalized sum of squared error for top down and the Bhattacharyya distance for bottom up are used sequentially as the criteria for tree structure.

In chapter 3, two feature extraction methods, canonical analysis and extended canonical analysis, are reviewed. Also, another feature extraction method, autocorrelation analysis, is proposed.

In chapter 4, a risk function is presented for estimating the error rate due to small numbers of design samples which cause the variability of estimated parameters. Results show that if the dimensionality is increased without increasing the separability between classes, the incremental error is increased by a factor of the square of the dimensionality. On the other hand, if the dimensionality is increased with separability as usual, the incremental error is simultaneously affected by two factors, dimensionality and separability. An optimal number of features which give the smallest risk value (or error rate) is predicted. Also, a relationship between error rate and the estimated decision boundary is studied empirically.

In chapter 5, experimental results are presented which show comparisons between trees, and between the single layer and a tree classifier, DTC for multisource data, and feature selection are described. It is found that a hybrid DTC with the best single feature in transformed coordinates has better performance compared to other methods such as a single layer classifier, top down and bottom up DTC's.

In chapter 6, the final conclusions are summarized.

CHAPTER 2. TREE CLASSIFIER DESIGN

2.1 Introduction

Optimal DTC's have been studied previously [1,21]. However, the existing methods for tree design, e.g., use of an evaluation function or admissible search, are not always feasible since the complete conditional density functions are often not available and a very large amount of computation time would be needed to evaluate all combinations of the tree classifier parameters. Practically, the methods of minimizing the classification error at each node are implemented to obtain locally optimum results, and the overall performance are not globably optimal [13].

When a remote sensing data set is categorized by partitioning the measurement space (or feature space) into non-overlapping decision regions, spectral classes which are discriminable because the multispectral properties of the corresponding ground covers are different, are defined. Remote sensing is successful if these spectral classes coincide with informational classes, i.e., classes of ground covers which are meaningful, such as crop species, major land uses, soil types, etc [9].

To be a valid class, a distribution must be simultaneously of informational value and separable from other classes. Supervised procedures can guarantee the former, but not the latter. Unsupervised procedures can provide the latter, but do not guarantee the former. Thus, a practical classification scheme for the DTC must contain both procedures in such a way that the simultaneity of satisfaction is guaranteed.

As far as DTC design is concerned, there exist only alternatives which are top down or bottom up approachs. Terminal classes must be both separable and of informational value. Non-terminal nodes are not required to be classes of informational value, but they must still be separable. Thus clustering, which insures separability, may be used in a top down approach, while correspondence with training sets is only required at the bottom and thus is related to a bottom up approach.

In the DTC literature [1,8,13,21,29], the top down approach has most often been studied. Only when the informational classes are easily discriminated in multispectral data, can the unsupervised classification of top down analysis be expected to produce reliable results. The most critical problem which occurs in the unsupervised top down analysis is terminal classes which do not coincide with the informational classes.

As stated previously, tree structure, decision rules, and optimal feature sets should be simultaneously considered to obtain an optimal DTC, however, due to the complexity of the problem, some compromise with respect to absolute optimality is appropriate. For a parametric approach, the decision rule may be selected first because the criterion for designing a DTC is determined by the decision rule. For our work here, the binary tree is chosen as a DTC structure, since any tree can be reduced to a binary tree, and the most effective feature subsets can be obtained for a binary tree. In order to achieve the other requirements of terminal classes of informational value and spectral separability, a hybrid DTC will be proposed. We shall begin by investigating top down and bottom up approach approaches for comparison purposes, and because certain of their characteristics will be needed in the hybrid scheme.

2.2 Top Down Approach

A complete DTC can be designed in a natural fashion by first defining a structure for the root node. Next, for each subset associated with the root node, it is necessary to define another node which performs a further decomposition into smaller subsets. In a top down approach, a clustering algorithm may be used to obtain two subgroups at each node. The initialization and the separation criterion are two important factors for clustering.

First, we shall consider the initialization method for clustering. How to determine the initial cluster centers is an important factor because the clustering results differ depending on the initial cluster centers. MacQueen [46] chose the first *k*- points in the sample as the initial *k*- cluster mean vectors. Beale [47] started with a trial value of *k*- larger than was thought necessary, and set up cluster centers regularly spaced at intervals of one standard deviation on each variable and then reduced the number of groups until a criterion based on the residual sum of squares is satisfied.

Another common way for choosing initial cluster centers is as follows. Let x_1, x_2, ... , x_n be *n*- sample vectors which are *q*- dimensional, and assume they are to be grouped into *C*- classes. Let $\mathbf{c}_i$, $\mathbf{u}_i$, and $\mathbf{s}_i$ (all *q*-dimensional vectors) be the i^{th} cluster center, i^{th} cluster mean, and i^{th} cluster standard deviation, respectively, for the i^{th} cluster. To establish an initial set of cluster centers, compute the mean vector $\mathbf{m}$ and variance vector $\mathbf{s}^2$ for the entire set of *n*- sample measurement vectors. A rectangular parallelepiped, which usually will contain a large percentage of the measurement vectors, has edges oriented parallel to the coordinate axes and given by $m_1 \pm s_1$, ..., $m_i \pm s_i$, ... , $m_q \pm s_q$. The initial cluster centers are chosen to be uniformly spaced along a principal diagonal of this parallelepiped [54].

Next, the criterion for clustering should be considered. The sum of squared error(SSE) criterion is defined by

$$SSE = \sum_{i=1}^{c} \sum_{x \in c_i} (\mathbf{x} - \mathbf{m}_i)^T (\mathbf{x} - \mathbf{m}_i) \qquad (2.1)$$

where $\mathbf{m}_i$ is the mean of the i^{th} cluster and $\mathbf{x} \in C_i$ is a pattern assigned to the i^{th} cluster. SSE is the cumulative distance between each point in the data set and the mean of the cluster to which that data point is assigned. To minimize the sum of squared error as a clustering criterion, each point (or pattern) should be assigned to the cluster which has minimum Euclidean distance from that point. This squared error can be expressed in many ways, such as the sum of the within and between class squared errors used in discriminant analysis [6]. The minimization of within class or maximization of between class squared error is identical to minimization of the sum of squared error criterion.

For for our method of designing the binary DTC, the mean vector $\mathbf{m}$ and variance vector $\mathbf{s}^2$ for the whole training set are computed to establish the two initial cluster centers. One initial cluster center is assigned to a *q*-dimensinal vector, $m_1+0.5s_1$, ... , $m_i+0.5s_i$, ... , $m_q+0.5s_q$. Another initial cluster center is assigned to another *q*-dimensional vector, $m_1-0.5s_1$, ... , $m_i-0.5s_i$, ... , $m_q-0.5s_q$ as shown in Figure 2.1.

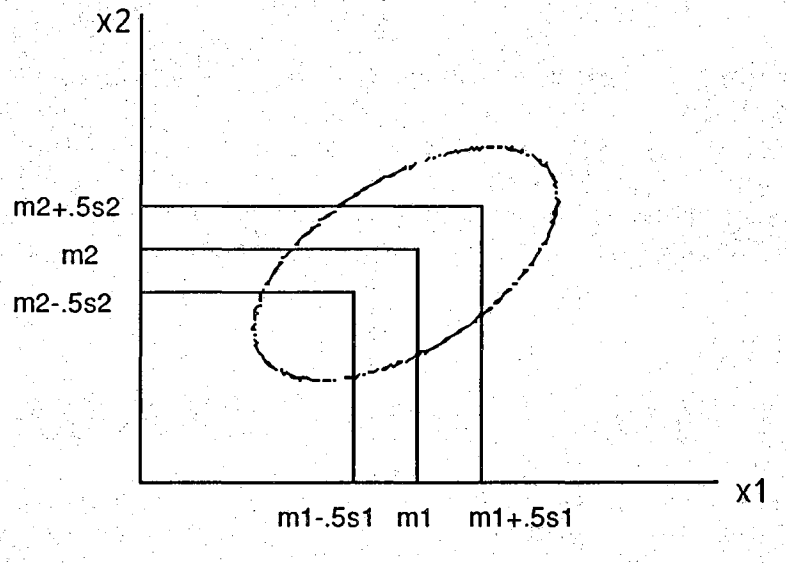


Figure 2.1 Initialization of Two Cluster Centers

Once the initial cluster centers are established, each training sample is assigned to one of two cluster centers which minimizes the Euclidean distance to the training sample. After all training samples are allocated to the nearest cluster center, the new class mean and class variance vectors for each cluster are computed, and those class mean vectors become new cluster centers. Again, all training samples are assigned to the new cluster center which minimizes the Euclidean distance.

When the number of training samples which are moved from one cluster to another cluster passes through a minimum and begins to increase, the clustering procedure is complete as shown in Figure 2.2.

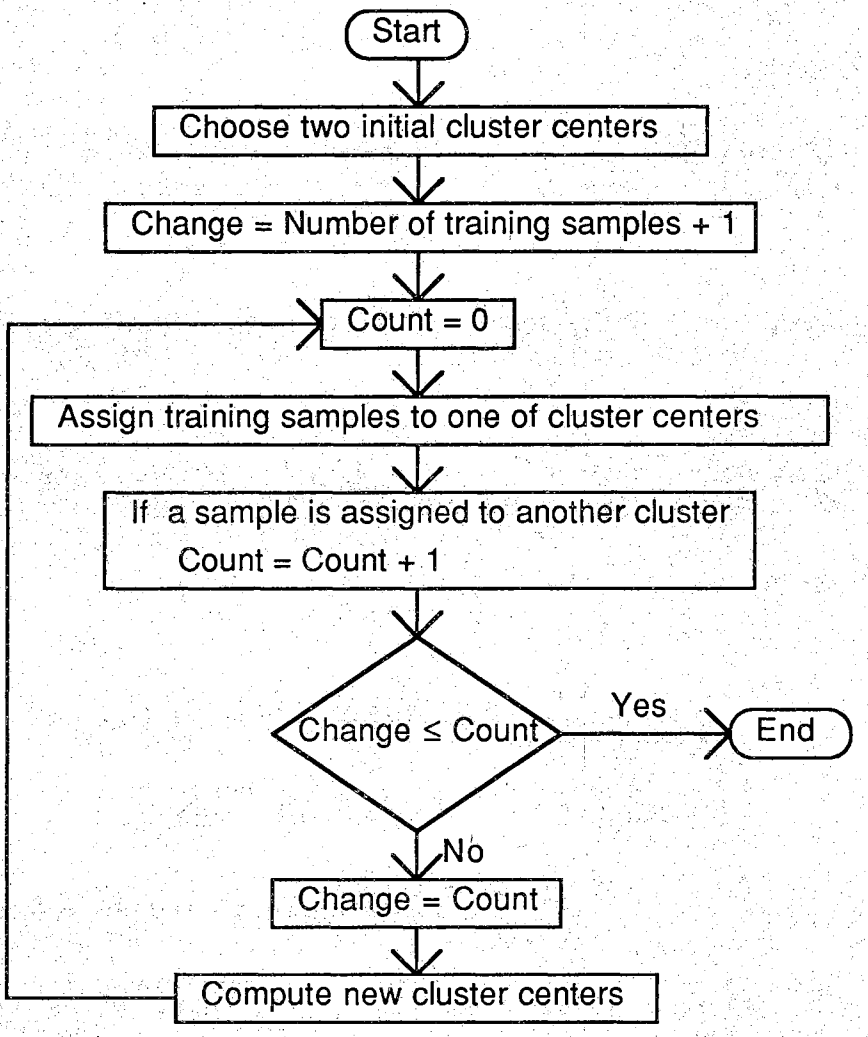


Figure 2.2 Clustering Algorithm

The clustering results produce two subclusters. After training samples for each class are compared to two subclusters, each class is assigned to one of two subclusters. Then each subcluster becomes a descendent node. Each subgroup is tested to determine if it is an informational class. If the subgroup is an informational class, the subgroup does not have a descendent node at the next level. If not, the whole clustering procedure is repeated for each subgroup as shown in Figure 2.3.

If the number of training samples from a class is split between two subgroup almost equally, that class is referred to as an overlapping class. If there is an overlapping class, that class is assigned to both subgroups. Whether a class is deemed an overlapping class or not is determined by the designer.

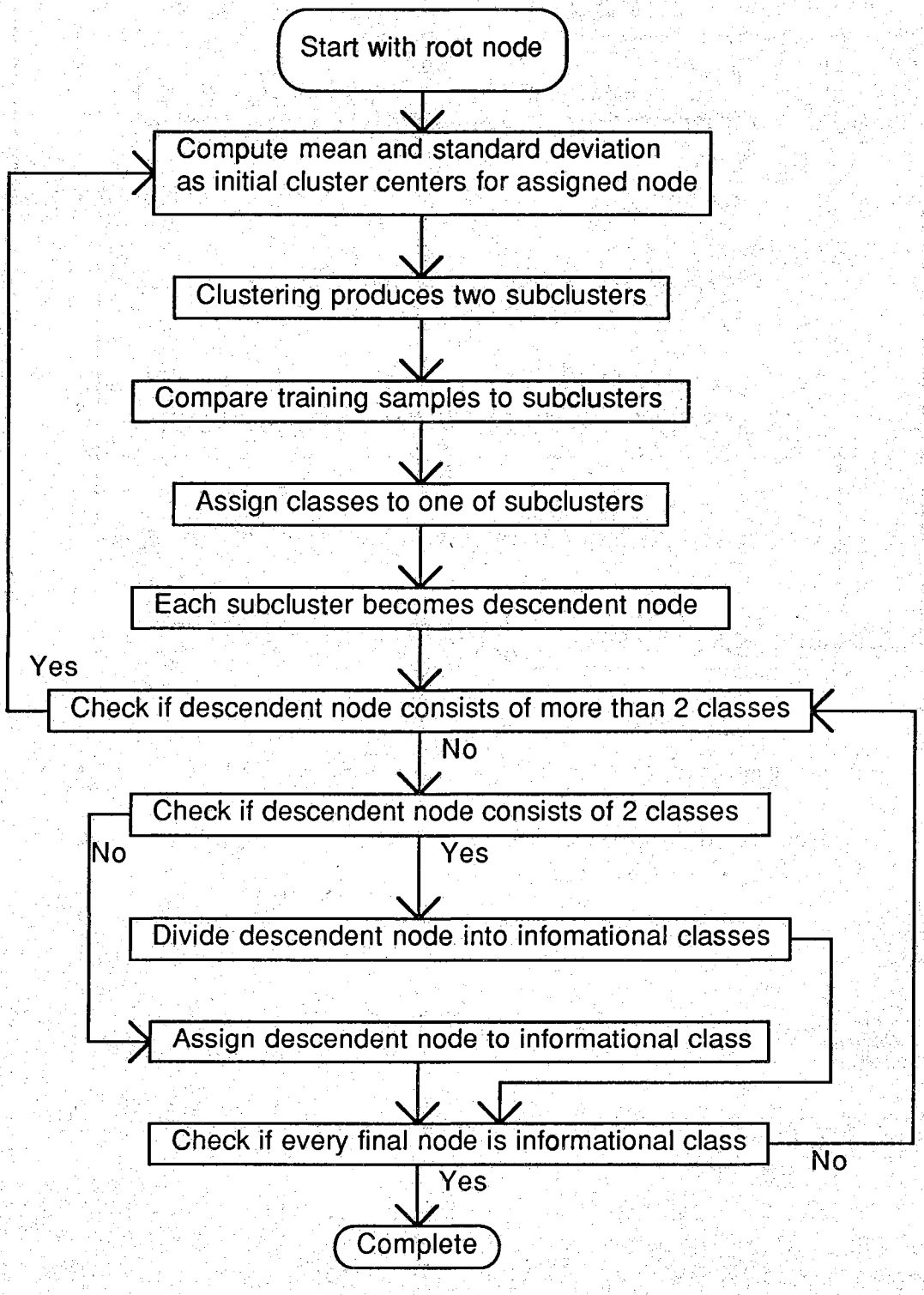


Figure 2.3 Top Down Design Algorithm

2.3 Bottom Up Approach

The bottom up approach begins with distinct informational classes as compared to overlapping informational classes which the top down approach sometimes produces. In the top down approach, the decision boundary which has the largest distance measure between groups is chosen since the effect of the error rate of the root node is minimized. In the bottom up approach, once the criterion values such as separabilities between every class pairs are computed, the two classes which have the smallest criterion value, join together and become a new class, i.e., the new subgroup at the next higher level.

In obtaining the new subgroup, the updated criterion values between the new subgroup and the remaining classes are needed. When the updated criterion values are determined from the previous criterion values, this will be referred to as a static minimum criterion spanning tree procedure. On the other hand, if the updated criterion values are newly computed, this will be called a dynamic minimum criterion spanning tree procedure.

2.3.1 The static minimum spanning tree

A graph G is an ordered pair $(V(G), E(G))$ consisting of a non-empty set $V(G)$ which represents the vertices and a set $E(G)$ which represents the edges.

Definition 1. Two vertices u and v are said to be connected if there is a path in G.

Definition 2. An undirected tree is an undirected graph which is connected and acyclic. A rooted undirected tree is an undirected tree in which one vertex is distinguished as the root. A spanning tree is an undirected tree that connects all vertices in V.

Lemma 1. Let $G = (V, E)$ be a connected, undirected graph and $S = (V, T)$ a spanning tree for G. Then,
a) for all u and v in V, the path between u and v in S is unique, and,
b) if any edge in $E - T$ is added to S, a unique cycle results.

Proof) See [39].

Lemma 2. Let $G = (V, E)$ be a connected, undirected graph and c a cost function on its edges. Let $\{(V_1, T_1), (V_2, T_2), \ldots\ldots, (V_k, T_k)\}$ be any spanning forest for G with $k > 1$. Let $T = \cup_{i=1}^{k} T_i$. Suppose $e = (v, w)$ is an edge of lowest cost in $E-T$ such that $v \in V_1$ and $w \notin V_1$. Then there is a spanning tree for G which includes $T \cup \{e\}$ and is of as low a cost as any spanning tree for G that includes T.

Proof) See [39].

By Lemma 1 and 2, If $G = (V, E)$ is a connected, undirected graph with a criterion value mapping edges to real numbers, a spanning tree is an undirected tree that connects all vertices in V. The cost of a spanning tree is just the sum of the cost of its edges. A spanning tree of minimum criterion value for G is the static minimum spanning tree.

The set of classes to be classified may be considered as a set of classes in a multidimensional space. The distance measure for every pair of classes is calculated. The set of classes and the distance measures are represented by a complete valued graph in Figure 2.4 which is a connected, undirected graph. A spanning tree can be extracted from this complete valued graph. Among the spanning trees, the minimum spanning tree is of particular interest. The static minimum spanning tree can be realized by the following single linkage method. The cost of its edge is represented with the similarity measure.

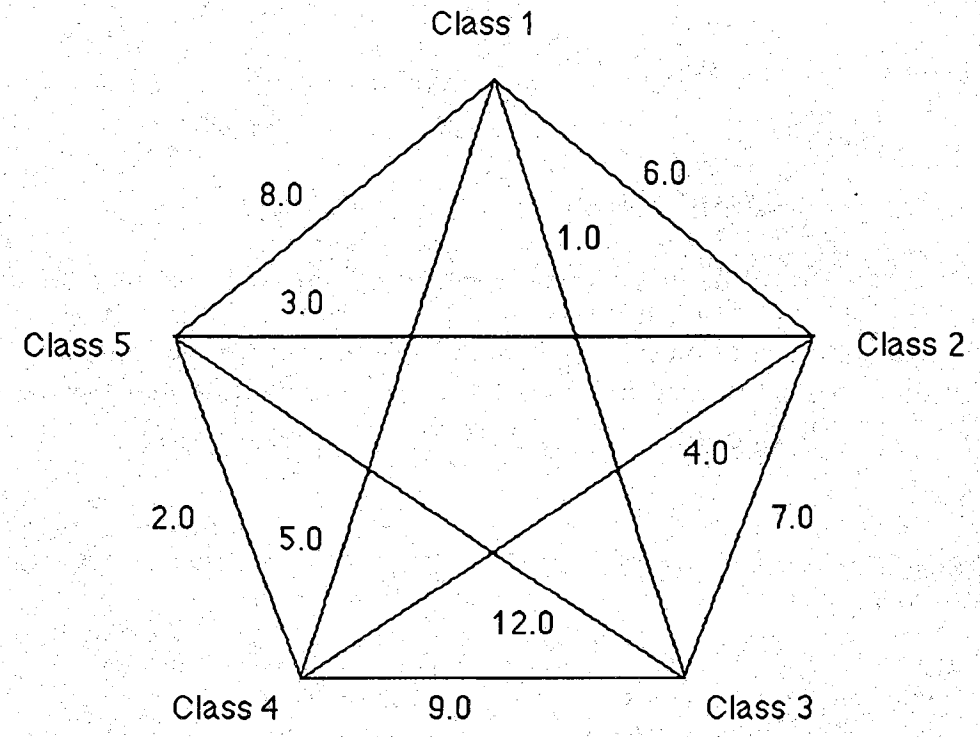


Figure 2.4 Complete Valued Graph

Classes are combined according to the similarity measure between their nearest members. Each matrix decreases by one after joining two classes. For the single linkage (nearest neighbor) method, then, new similarity measures between subgroups can be obtained by the similarity measure between their closest classes.

Suppose five classes are to be classified, and the matrix of distances between the classes is as follows:

$$\mathbf{D} = \begin{bmatrix} 0 & 6.0 & 1.0 & 5.0 & 8.0 \\ 6.0 & 0 & 7.0 & 4.0 & 3.0 \\ 1.0 & 7.0 & 0 & 9.0 & 12.0 \\ 5.0 & 4.0 & 9.0 & 0 & 2.0 \\ 8.0 & 3.0 & 12.0 & 2.0 & 0 \end{bmatrix}$$

In this matrix the element in the i^{th} row and j^{th} column gives the distance, d_{ij}, between classes i and j. The minimum d_{ij} is $d_{13} = 1.0$ so that classes 1 and 3 are joined to minimize the error rate at the upper levels and the new classes are (1, 3), (2), (4), and (5). New distances between these subgroups are obtained from D as follows:

$$d(2)(1, 3) = \min \{d_{21}, d_{23}\} = d_{21} = 6.0,$$

$$d(4)(1, 3) = \min \{d_{41}, d_{43}\} = d41 = 5.0,$$

$$d(5)(1, 3) = \min \{d_{51}, d_{53}\} = d51 = 8.0,$$

and we may form a new distance matrix D_1 giving inter-individual distances, and group-individual distances.

$$\mathbf{D}_1 = \begin{bmatrix} 0 & 6.0 & 5.0 & 8.0 \\ 6.0 & 0 & 4.0 & 3.0 \\ 5.0 & 4.0 & 0 & 2.0 \\ 8.0 & 3.0 & 2.0 & 0 \end{bmatrix}$$

The smallest entry in D_1 is now $d_{45} = 2.0$ and so classes 4 and 5 are combined and new subgroups become (1, 3), (2), and (4, 5), and distances now become

$$d(1, 3)(4, 5) = \min \{d(4)(1, 3), d(5)(1, 3)\} = d(4)(1, 3) = 5.0,$$

$$d(2)(4, 5) = \min\{d_{42}, d_{52}\} = d_{52} = 3.0.$$

These may be arranged in a matrix D_2,

$$\mathbf{D}_2 = \begin{bmatrix} 0 & 6.0 & 5.0 \\ 6.0 & 0 & 3.0 \\ 5.0 & 3.0 & 0 \end{bmatrix}.$$

The smallest entry now is d(2)(4, 5) = 3.0, so that class 2 is joined to subgroup (4, 5) and the subgroups are now (1, 3) and (2, 4, 5). Finally, the combination of the two subgroups at this stage takes place to form a single group containing all five classes. The tree showing these processes is shown in Figure 2.5

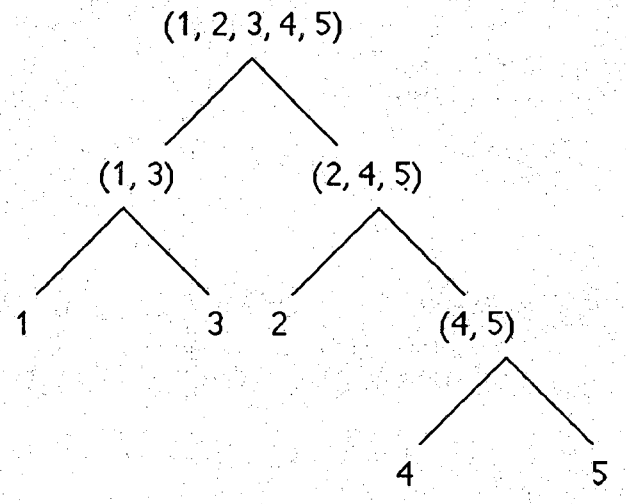


Figure 2.5 Decision Tree Corresponding to Single Linkage

The complete linkage (farthest neighbor) method is exactly the opposite of the single linkage method in that the distance between two subgroups is defined in terms of the largest dissimilarity between a member of c_1 and a member of c_2, namely

$$d_{(c_1)(c_2)} = \max \{d_{ij} ; i \in c_1, j \in c_2\} \tag{2.2}$$

Using this technique for the distance matrix D, we begin as with the single linkage method by combining classes 1 and 3. The distances between this subgroup and the three remaining individuals 2, 4, and 5 are obtained from D as follows:

$$d(2)(1, 3) = \max \{d_{21}, d_{23}\} = d_{23} = 7.0,$$

$$d(4)(1, 3) = \max \{d_{41}, d_{43}\} = d_{43} = 9.0,$$

$$d(5)(1, 3) = \max \{d_{51}, d_{53}\} = d_{53} = 12.0,$$

$$\mathbf{D} = \begin{bmatrix} 0 & 6.0 & 1.0 & 5.0 & 8.0 \\ 6.0 & 0 & 7.0 & 4.0 & 3.0 \\ 1.0 & 7.0 & 0 & 9.0 & 12.0 \\ 5.0 & 4.0 & 9.0 & 0 & 2.0 \\ 8.0 & 3.0 & 12.0 & 2.0 & 0 \end{bmatrix}$$

The new distance matrix is

$$\mathbf{D}_1 = \begin{bmatrix} 0 & 7.0 & 9.0 & 12.0 \\ 7.0 & 0 & 4.0 & 3.0 \\ 9.0 & 4.0 & 0 & 2.0 \\ 12.0 & 3.0 & 2.0 & 0 \end{bmatrix}.$$

The smallest entry is d_{45}, so that classes 4 and 5 are joined and new subgroups become (1, 3), (4, 5), and (2), with

$$d(1, 3)(4, 5) = \max \{d(4)(1, 3), d(5)(1, 3)\} = d(5)(1, 3) = 12.0,$$

$$d(2)(4, 5) = \max \{d_{42}, d_{52}\} = d_{42} = 4.0.$$

and

$$\mathbf{D}_2 = \begin{bmatrix} 0 & 7.0 & 12.0 \\ 7.0 & 0 & 4.0 \\ 12.0 & 4.0 & 0 \end{bmatrix}.$$

The smallest entry is d(2)(4, 5) = 4.0, so that class 2 is joined to subgroup (4, 5) and the subgroups are now (1, 3) and (2, 4, 5). The final result is shown in Figure 2.6. which is seen to be identical in shape to that resulting from the single linkage method incidentally .

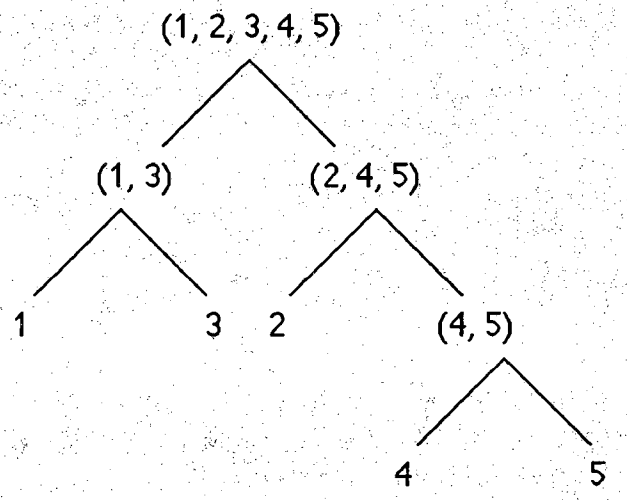


Figure 2.6 Decision Tree Corresponding to Complete Linkage

2.3.2 The dynamic minimum spanning tree

The Bhattacharyya distance measure is selected as the criterion for the bottom up DTC. The Bhattacharyya distance measure for two Gaussian classes is as follows.

$$B = \frac{1}{8}(\mathbf{m}_1 - \mathbf{m}_2)^T \left[\frac{\Sigma_1 + \Sigma_2}{2}\right]^{-1} (\mathbf{m}_1 - \mathbf{m}_2) + \frac{1}{2} \ln \frac{\left|\frac{\Sigma_1 + \Sigma_2}{2}\right|}{\sqrt{|\Sigma_1||\Sigma_2|}} \qquad (2.3)$$

where $\mathbf{m}_i$ is the mean vector of class i and Σ_i is the covariance matrix of class i. The first term of the Bhattacharyya distance reflects the separation due to mean differences between two classes and the second term reflects the covariance difference. The Bhattacharyya distance measure is more closely related with classification accuracy than other measure functions such as divergence[50].

In the static minimum spanning tree, the mean and covariance vectors for each class are computed just one time throughout the design of the decision tree classifier. But new subgroups based upon combining two classes or subgroups have new mean and covariance matrices. Two classes are combined according to the similarity measure between their nearest members. Groups or classes are decreased by one. After combining the smallest Bhattacharyya distance pair, the new mean and covariance matrices are obtained. Then the new Bhattacharyya distances are computed between the new subgroup and the remaining classes or subgroups. The above procedures are continued until all classes become two subgroups.

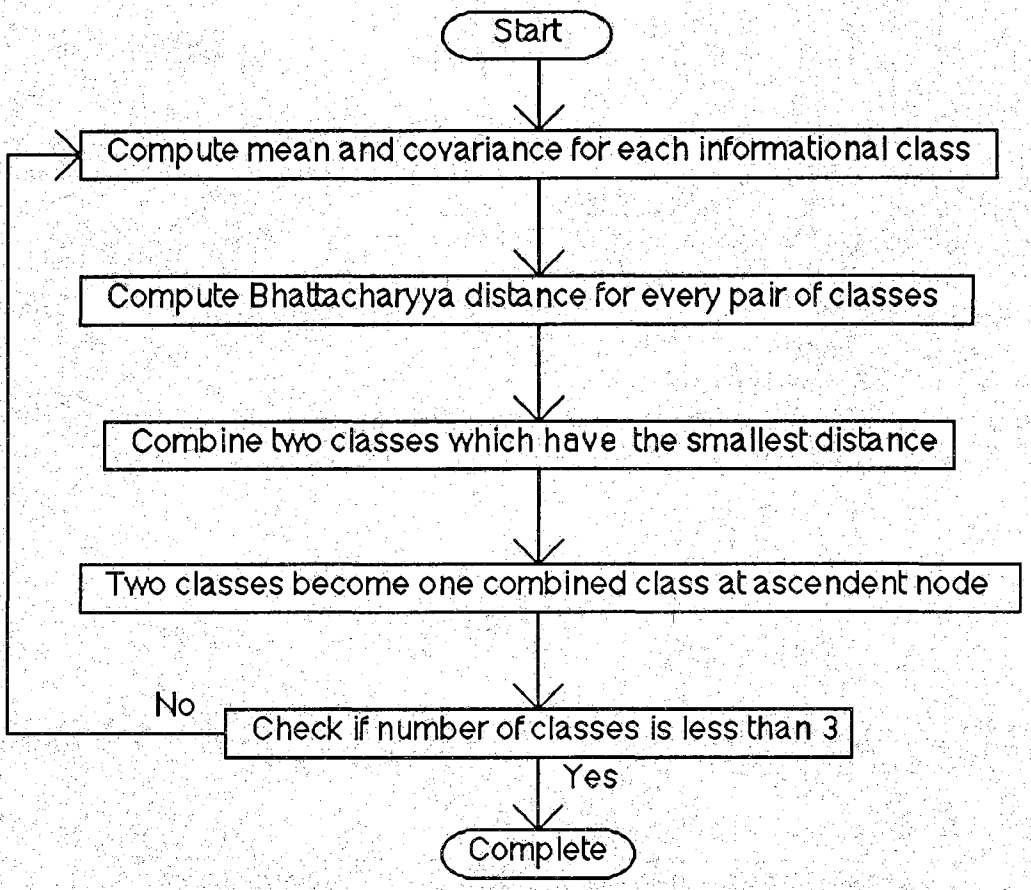


Figure 2.7 Bottom Up Design Algorithm

2.4 Hybrid Approach

To be a valid class, a class must be simultaneously of informational value and separable from all other classes. The top down approaches make use of the characteristic of class separability while the bottom up approach starts with informational classes. The hybrid approach uses bottom up and top down methods sequentially.

The bottom up approach may produce subgroups. The number of subgroups is determined by the designer. Those subgroups may give the subgroup information which is used for the top down approach. In the top down approach, how to use the subgroup information which is generated by bottom up approach is dependent upon the algorithm which is applied for the top down approach.

In nonsupervised classification algorithms, initialization information has a role in determining the final results. Correct initial information facilitates obtaining the correct results. The bottom up approach is used as the method to obtain this initialization information (mean vectors and covariance matrices). For the hybrid

approach, the normalized sum of squared error (NSSE) criterion, defined as follows, is used.

$$NSSE = \sum_{i=1}^{c} \sum_{x \in c_1} \left[(\mathbf{x} - \mathbf{m}_i)^T \Sigma_i^{-1} (\mathbf{x} - \mathbf{m}_i) + \ln|\Sigma_i| \right] \quad (2.4)$$

The sum of squared error criterion produces the spherical clusters [37]. However, if the clusters are not spherical in shape, variance effects must be accounted for. In the normalized sum of squared error criterion, variance effects are considered. In the hybrid approach, subgroup means and covariances are obtained from bottom up results.

The hybrid design, thus proceeds as follows:

1. Divide the entire data set into two subgroups for descendent nodes by the bottom up approach.

2. Compute the mean and covariance vectors of the two subgroups and re-divide the classes into two subgroups using the Normalized Sum of Squared Error Clustering (Appendix B).

3. If the separated subgroups are informational classes, design is complete. Otherwise, return to step 1 for each subgroup which is not an informational class.

There are several advantages to the hybrid approach. It is more likely to converge to classes of informational value because the initialization provides early guidance in that direction while the top down approach does not guarantee such convergence. It can use overlapping classes while there are no overlapping classes in the bottom up approach. Covariance information can be applied in the hybrid approach to separate non-spherical subgroups.

2.5 Tree Classifier for Multisource Data

Modern data sets include not only spectral data but may also include other types of data, such as forest type maps, ground class cover maps, radar data, and topographic information e.g., elevation, slope, and aspect data. These are called multisource data. Such data are not necessarily in common units, and therefore scaling problems may arise. Further, the data may not even be numerical. As a result, multisource data cannot be modeled conveniently by multivariate distributions, thus conventional multivariate classification methods cannot be used satisfactorily in processing multisource data. Several methods have been proposed to classify the multisource data.

Hutchinson [51] proposed ambiguity reduction techniques. If the data are classified based on one or more data sources, the remaining ambiguities from the results of classification are resolved by other sources.

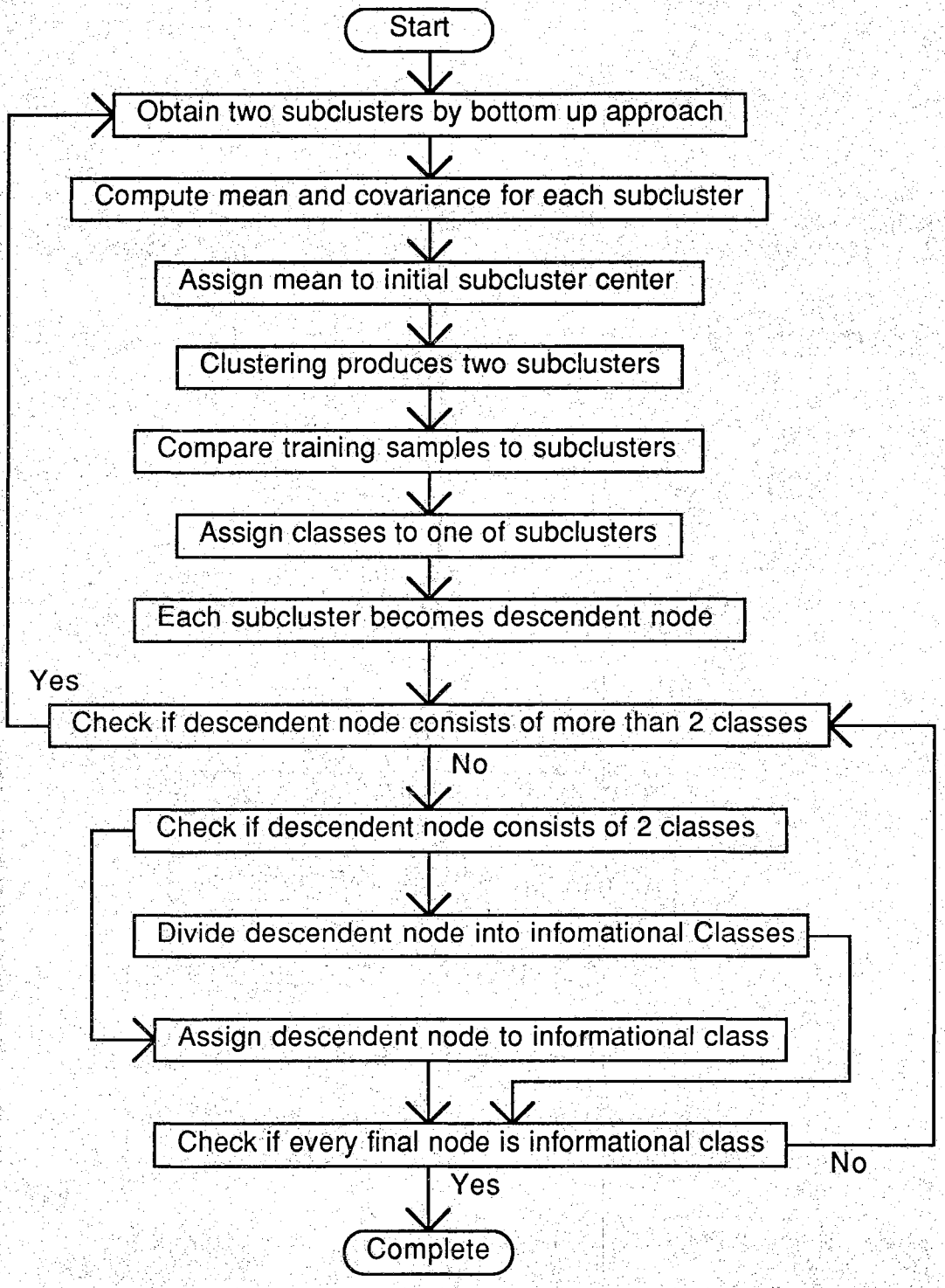


Figure 2.8 Hybrid Design Algorithm

The stacked vector approach, which consists of a concatenation of all components of data sources, has also been used [52]. This method is straightforward and simple, however, the method is not applicable when the various sources cannot be modeled by the multivariate distributions.

Swain, Richards and Lee [53] proposed a statistically based analysis. In general there may not be a simple relation between the user-desired information classes and the set of data classes available. One of the requirements of a multisource analytical procedure is to devise a method by which inferences about information classes can be drawn from the collection of data classes. They defined a set of global membership functions that collect together the inferences concerning a single information class from all of the data sources. They use the global membership function in the nature of a discriminant function, so that a pixel is then classified according to the usual maximum selection rule. In that case, the inter-source independence assumption is often made, however, that assumption is not usually fully satisfied in the case of real data.

In the DTC approach, each source may be considered separately, something not possible in a single layered scheme. The basic idea is that the optimal source and classification rules are determined to minimize the classification error at each node. To separate the subgroups evaluation functions are defined as a function of minimum error and minimum overlap. The overlap is defined as follows;

$$d = \sum_{i=1}^{c} n_i - n \tag{2.5}$$

where n_i is the number of samples of class i. The evaluation function is given by

$$E_i = \sum_{j=1}^{c} P_e(j) + \alpha d \tag{2.6}$$

where α is weighting factor.

For a hybrid DTC with a Gaussian maximum likelihood rule, two initial subgroups can be obtained by the bottom up approach with respect to each source. The subgroup consists of more than one informational class. To obtain new subgroups, the Normalized Sum of Squared Error is applied for two clusters with respect to each source. To determine the best source, the evaluation function for each source is computed by evaluating the results of two clusters. Every node has the appropriate source to minimize the evaluation function. If a subgroup is not an informational class, the hybrid design procedure is applied again to obtain two descendent nodes.

2.6 Computational Efficiency

Computational efficiency is potentially one of DTC's advantages over a single layer classifier. As far as the number of decisions is concerned, a DTC needs less numbers of decisions than a single layer classifier. When a DTC is well balanced, only ($\log_2$ n) comparisons are required if n classes are given. Even though a DTC is completely unbalanced, $\frac{(n+2)(n-1)}{2n}$ comparisons are required. This is less than the (n-1) comparisons are required in a single layer classifier, and therefore a DTC is expected to have better computational efficiency.

To improve the classification accuracy in the limited training sample situation, a reduced number of features may be used in transformed coordinates. In a DTC, (n-1) transformations of a test sample are needed while in a single layer classifier, only one transformation of a test sample is required. In other words, a DTC needs less comparisons but more mappings while a single layer classifier requires more comparisons but less mapping. Therefore, neither of the two classifiers always has superior computation efficiency.

CHAPTER 3. FEATURE EXTRACTION

3.1 Hughes Phenomenon

Hughes[22] showed that recognition accuracy can first increase as the number of measurements (or features) increases. However, in the presence of a limited training sample size, as the dimension of the data increases beyond the optimum value the classification results decline.

If there were no such dimensionality phenomenon, the single layer maximum likelihood classifier (assuming proper prior class probabilities) would provide better performance than the any other DTC because the conventional Bayes classifier gives the minimum classification error.

However, when the number of training samples are limited, the Hughes phenomenon, i.e. the dimensionality problem, must be considered. In such cases, the conditional density functions are incorrectly estimated because of the lack of adequate training samples. The poor estimates cause complex decision boundary to be biased. A properly designed DTC may have better performance than a single layer classifier because the decision boundaries in a DTC are much simpler than in a single layer classifier.

In most DTC's, untransformed subsets of features are used since a reduction of dimensionality is required to increase the classification accuracy. A typical procedure might be to calculate the pairwise Bhattacharyya distance or divergence at each node, then the subsets of features having the largest distance are selected for dimensionality reduction. Since the estimated means and covariances themselves are randomly biased in the limited sample situation, a better way to pick the best subsets of features is required.

3.2 Canonical Analysis

Fisher's suggestion[37] was to look for the linear function which maximizes the ratio of the between class scatter to the within class scatter. Canonical analysis finds a set of linear combinations of the variables whose values are as close as possible within classes and as far apart as possible between classes. In canonical analysis, within-class and between-class scatter matrices are used to formulate a criterion of class separability.

A within-class scatter matrix shows the scatter of samples around their class mean vector $\mathbf{m}_i$, and is expressed by

$$\mathbf{S}_w = \sum_{i=1}^{m} P(W_i)\, E\{(\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^T | w_i\}$$

$$= \sum_{i=1}^{m} P(W_i)\, \Sigma_i \qquad (3.1)$$

The matrix $\mathbf{S}_w$ is proportional to the sample covariance matrix. A between-class scatter matrix is given by

$$\mathbf{S}_b = \sum_{i=1}^{m} P(W_i)\, (\mathbf{m}_i - \mathbf{m}_o)(\mathbf{m}_i - \mathbf{m}_o)^T \qquad (3.2)$$

$$\mathbf{m}_o = E\{\mathbf{x}\} = \sum_{i=1}^{m} P(W_i)\, \mathbf{m}_i \qquad (3.3)$$

$$\text{rank}(\mathbf{S}_b) = m-1$$

$$\text{rank}(\mathbf{S}_w^{-1}\, \mathbf{S}_b) = m-1$$

where $\mathbf{m}_i$ is the mean of i^{th} class and $\mathbf{m}_o$ is the global mean. All these scatter matrices are invariant under coordinate shifts. We define the ratio of the between class scatter to the within class scatter as follows:

$$\frac{\mathbf{d}^T\, \mathbf{S}_b\, \mathbf{d}}{\mathbf{d}^T\, \mathbf{S}_w\, \mathbf{d}} \qquad (3.4)$$

The vector $\mathbf{d}$, which maximizes the ratio $\mathbf{d}^T\mathbf{x}$, is called the Fisher's linear discriminant function or the first canonical variate.

The spectral decomposition allows us to express the inverse of a square matrix in terms of its eigenvalues and eigenvectors, and this leads to a square root matrix. Let $\mathbf{S}_w$ be an n by n positive definite matrix with the spectral decomposition

$$\mathbf{S}_w = \sum_{i=1}^{n} \lambda_i\, \mathbf{e}_i\, \mathbf{e}_i^T.$$

Let the normalized eigenvectors be the columns of matrix $\mathbf{P} = [\, \mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n \,]$. Then

$$\mathbf{S}_w = \sum_{i=1}^{n} \lambda_i \mathbf{e}_i \mathbf{e}_i^T = \mathbf{P}^T \Lambda \mathbf{P} \tag{3.5}$$

where $\mathbf{S}_w = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \lambda_n \end{bmatrix}$ with $\lambda_i \geq 0$.

$$\mathbf{S}_w = \mathbf{P}^T \Lambda^{-1} \mathbf{P} = \sum_{i=1}^{n} \frac{1}{\lambda_i} \mathbf{e}_i \mathbf{e}_i^T \tag{3.6}$$

$$\mathbf{S}_w^{\frac{1}{2}} = \sum_{i=1}^{n} \sqrt{\lambda_i}\ \mathbf{e}_i \mathbf{e}_i^T = \mathbf{P}^T \Lambda^{\frac{1}{2}} \mathbf{P} \tag{3.7}$$

Let $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_s \geq 0$ denote the nonzero eigenvalues of $\mathbf{S}_w^{-1} \mathbf{S}_b$ and $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_s$ the corresponding normalized eigenvectors where *s* is less than or equal to *m* and *n - 1* (*m* is the number of classes and *n* is dimension of the matrix or the total number of features). The symmetric square root matrix

$$\mathbf{S}_w^{\frac{1}{2}} = \sum_{i=1}^{n} \sqrt{\lambda_i}\ \mathbf{e}_i \mathbf{e}_i^T = \mathbf{P}^T \Lambda^{\frac{1}{2}} \mathbf{P}$$

and its inverse

$$\mathbf{S}_w^{-\frac{1}{2}} = \mathbf{P}^T \Lambda^{-\frac{1}{2}} \mathbf{P}$$

$$\mathbf{d}^T \mathbf{S}_w \mathbf{d} = \mathbf{d}^T \mathbf{S}_w^{\frac{1}{2}} \mathbf{S}_w^{\frac{1}{2}} \mathbf{d} \tag{3.8}$$

$$\mathbf{d}^T \mathbf{S}_b \mathbf{d} = \mathbf{d}^T \mathbf{S}_w^{\frac{1}{2}} \mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_b \mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_w^{\frac{1}{2}} \mathbf{d} \tag{3.9}$$

Let be $\mathbf{a} = \mathbf{S}_w^{\frac{1}{2}} \mathbf{d}$ then

$$\frac{\mathbf{d}^T \mathbf{S}_b \mathbf{d}}{\mathbf{d}^T \mathbf{S}_w \mathbf{d}} = \frac{\mathbf{a}^T \mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_b \mathbf{S}_w^{-\frac{1}{2}} \mathbf{a}}{\mathbf{a}^T \mathbf{a}} \tag{3.10}$$

Finally, the problem reduces to maximizing equation (3.10) with respect to **a**. This is the so called Rayleigh's quotient. From Rayleigh's principle[40], the maximum of the ratio is λ_1 which is the largest eigenvalue of $\mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_b \mathbf{S}_w^{-\frac{1}{2}}$ when **a** is equal to $\mathbf{e}_1$. When **a** is equal to eigenvector $\mathbf{e}_2$, which is orthonormal to $\mathbf{e}_1$, and corresponding to λ_2, **a** maximizes the ratio secondarily. Therefore, λ_k is the k^{th} largest value which corresponds to eigenvector $\mathbf{e}_k$.

$$\mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_b \mathbf{S}_w^{-\frac{1}{2}} \mathbf{e} = \lambda \mathbf{e} \tag{3.11}$$

$$\mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_b \mathbf{S}_w^{-\frac{1}{2}} \mathbf{e} = \mathbf{S}_w^{-1} \mathbf{S}_b (\mathbf{S}_w^{-\frac{1}{2}} \mathbf{e}) = \mathbf{S}_w^{-1} \mathbf{S}_b \mathbf{d} = \lambda \mathbf{d} \tag{3.12}$$

The eigenvector $\mathbf{d}_i$ which corresponds to eigenvalue λ_i is directly obtained from $\frac{\mathbf{S}_b}{\mathbf{S}_w}$. The eigenvector $\mathbf{d}_i$ is called the i^{th} canonical variate. If we have only two classes the ratio has only one nonzero eigenvalue. The other *n-1* features do not contribute to the ratio. The final solution for two classes is

$$\mathbf{d} = \mathbf{S}_w^{-1} (\mathbf{m}_1 - \mathbf{m}_2) \tag{3.13}$$

This is also called Fisher's linear discriminant function which has the maximum ratio of between-class scatter to within-class scatter.

3.3 Extended Canonical Analysis

The following method was developed by Foley and Sammon[12]. In the two class problem, Fisher's vector is given by $\mathbf{d}_1 = \frac{\alpha_1 (\mathbf{m}_1 - \mathbf{m}_2)}{\mathbf{S}_w}$ where α_1 is chosen such that $\mathbf{d}_1^T \mathbf{d}_1 = 1$. The next best direction can be found for maximixing Fisher criterion subject to the constraint that $\mathbf{d}_1$ and $\mathbf{d}_2$ are orthogonal. Using the method of Lagrange multipliers, we wish to maximize the Fisher criterion subject to the constraints that $\mathbf{d}_i^T \mathbf{d}_n = 0$ for i = 1,2,...,n-1. Let C be

$$C = \frac{\left\{\mathbf{d}_n^T(\mathbf{m}_1 - \mathbf{m}_2)\right\}^2}{\mathbf{d}_n^T\mathbf{S}_w\mathbf{d}_n} - \lambda_1\mathbf{d}_n^T\mathbf{d}_1 - \ldots - \lambda_{n-1}\mathbf{d}_n^T\mathbf{d}_{n-1} \tag{3.14}$$

Setting the partial of C with respect to d_n equal to zero,

$$2K_n(\mathbf{m}_1 - \mathbf{m}_2) - K_n^2 2\mathbf{S}_w\mathbf{d}_n - \lambda_1\mathbf{d}_1 - \ldots - \lambda_{n-1}\mathbf{d}_{n-1} = 0 \tag{3.15}$$

where $K_n = \dfrac{\mathbf{d}_n^T(\mathbf{m}_1 - \mathbf{m}_2)}{\mathbf{d}_n^T\mathbf{S}_w\mathbf{d}_n}$. Therefore,

$$\mathbf{d}_n = \frac{1}{K_n}\mathbf{S}_w^{-1}\left[(\mathbf{m}_1 - \mathbf{m}_2) - \frac{\lambda_1}{2K_n}\mathbf{d}_1 - \ldots - \frac{\lambda_{n-1}}{2K_n}\mathbf{d}_{n-1}\right] \tag{3.16}$$

Applying the constraints, and letting $\beta_i = \dfrac{\lambda_i}{2K_n}$ and $s_{ij} = \mathbf{d}_i\mathbf{S}_w\mathbf{d}_j$, then

$$\begin{aligned} &s_{11}\beta_1 + s_{12}\beta_2 + \ldots + s_{1(n-1)}\beta_{n-1} = 1/\alpha_1 \\ &s_{i1}\beta_1 + \ldots + si_{(n-1)}\beta_{n-1} = 0 \\ &\qquad\vdots \\ &s_{(n-1)1}\beta_1 + \ldots + s_{(n-1)(n-1)}\beta(n-1) = 0 \end{aligned} \tag{3.17}$$

Let $\beta^T = [\beta_1 \ldots \beta_{n-1}]$, then $\beta = \mathbf{S}_{n-1}^{-1}\left[\dfrac{1}{a_1}0 \ldots 0\right]^T$. A recursive definition for the n^{th} discriminant vector is

$$\mathbf{d}_n = \alpha_n\mathbf{S}_w^{-1}\left\{(\mathbf{m}_1 - \mathbf{m}_2) - [\mathbf{d}_1 \ldots \mathbf{d}_{n-1}]\mathbf{S}_{n-1}^{-1}\begin{bmatrix}\frac{1}{a_1}\\0\\\vdots\\0\end{bmatrix}\right\} \tag{3.18}$$

3.4 Autocorrelation Analysis

The Fisher concept can be applied to an autocorrelation matrix. In a two class problem, the criterion which maximizes $\mathbf{S}_1$ and minimizes $\mathbf{S}_2$ simultaneously or vice versa can be defined. An autocorrelation matrix is defined by

$$\mathbf{S}_i = \Sigma_i + \mathbf{m}_i\mathbf{m}_i^T \tag{3.19}$$

The criterion function is defined by $r = \frac{\mathbf{d}_i^T\mathbf{S}_1\mathbf{d}_i}{\mathbf{d}_i^T\mathbf{S}_2\mathbf{d}_i}$ or $\frac{\mathbf{d}_i^T\mathbf{S}_2\mathbf{d}_i}{\mathbf{d}_i^T\mathbf{S}_1\mathbf{d}_i}$. The optimally separable feature set is a feature set such that $\mathbf{S}_1$ is minimized and $\mathbf{S}_2$ is maximized or vice versa after the transformation. The ratio r is maximized by the selection of feature d if $\frac{\partial r}{\partial \mathbf{d}} = 0$. That equation can be reduced to $(\mathbf{S}_1 - r\mathbf{S}_2)\mathbf{d} = 0$ which is called a generalized eigenvalue equation.

$$(\mathbf{S}_2^{-1}\mathbf{S}_1 - \mathbf{R})\mathbf{d} = 0 \tag{3.20}$$

$$\mathbf{S}_2^{-1}\mathbf{S}_1[\mathbf{d}_1 \ldots \mathbf{d}_q] = \mathbf{R}[\mathbf{d}_1 \ldots \mathbf{d}_q] \tag{3.21}$$

We can diagonalize two symmetric matrices as

$$\mathbf{D}^T\mathbf{S}_1\mathbf{D} = \mathbf{I} \tag{3.22}$$

$$\mathbf{D}^T\mathbf{S}_2\mathbf{D} = \mathbf{R} \tag{3.23}$$

where $\mathbf{D}$ and $\mathbf{R}$ are the eigenvector and eigenvalue matrices of $\mathbf{S}_1^{-1}\mathbf{S}_2$. To find the orthonormal eigenvectors of $\mathbf{S}_1^{-1}\mathbf{S}_2$ as

$$\mathbf{S}_1^{-1}\mathbf{S}_2\mathbf{d}_i = r_i\mathbf{d}_i \text{ and } \mathbf{d}_i^T\mathbf{d}_j = \delta_{ij} \tag{3.24}$$

We should change the scale of $\mathbf{d}_i$ to satisfy

$$\alpha^2\mathbf{d}_i^T\mathbf{S}_1\mathbf{d}_i = 1 \tag{3.25}$$

Therefore, the i^{th} orthonormal eigenvector is

$$\mathbf{d}_i' = \frac{\mathbf{d}_i}{(\mathbf{d}_i^T \mathbf{S}_1 \mathbf{d}_i)^{\frac{1}{2}}} \tag{3.26}$$

For each discriminant vector $\mathbf{d}_i'$, there corresponds an r_i, given by

$$r_i = \frac{\mathbf{d}_i'^T \mathbf{S}_1 \mathbf{d}_i'}{\mathbf{d}_i'^T \mathbf{S}_2 \mathbf{d}_i'} \tag{3.27}$$

Each r_i represents the value of the discriminatory criterion for the corresponding discriminant vector $\mathbf{d}_i'$. The discriminant vectors can be ordered according to their respective ratio values such that

$$r_1 \geq r_2 \geq \ldots \geq r_q \geq 0 \tag{3.28}$$

However, we want to maximize the relative ratio between $\mathbf{S}_1$ and $\mathbf{S}_2$ which is greater than one. If an r_i is less than one, we should use the inverse value of r_i to compare the relative ratio. We may define the relative ratio as follows:

$$r_1 \geq r_2 \geq \ldots \geq r_i \geq 1$$

$$\frac{1}{r_q} \geq \frac{1}{r_{q-1}} \geq \ldots \geq \frac{1}{r_{i+1}} \geq 1 \tag{3.29}$$

The best feature or effective basis function for both classes is the eigenvector corresponding to the largest relative ratio. The autocorrelation analysis can be used in place of canonical analysis when the mean difference between two classes is almost zero.

When the mean difference is zero canonical analysis and extended canonical analysis can not be used since the feature vector is defined by the mean difference. Autocorrelation analysis is useful when the mean difference is small and the covariance difference is dominant while canonical analysis and extended canonical analysis are more effective than the autocorrelation analysis when the mean difference is dominant. After extracting features, the mean difference and the covariance difference in a subspace may be checked to use one of the two methods, extended canonical analysis or autocorrelation analysis, for the next feature extraction.

CHAPTER 4. ESTIMATION OF OPTIMAL NUMBER OF FEATURES

It is well known that classifier accuracy expressed as a function of the number of features used shows a maximum at some finite dimensionality [22]. For a given class-conditional density function set, the occurrence of this peak is dependent upon the training sample size [3], as a result of the accuracy dependence upon the quality with which the density parameters are estimated.

A long-known fundamental barrier to the optimal design of classifiers is the inability to be able to directly calculate the expected accuracy of a trial classifier design. As a result, a common practice is to use a statistical measure, e.g. Bhattacharyya distance, to estimate the expected accuracy. However the relationship of such distance measures to classification accuracy, though monotonic, is not precisely one-to-one, and thus, if such a distance is to be used in the design process, it is important to clearly understand just what the relationship is between expected accuracy and a specific distance measure used to estimate it. It is this relationship which is studied next, paying specific attention to the effects of sample size and parameter estimation variability.

4.1 Optimal Number of Features

The quality of a density parameter is specified by following theorems.

Theorem 1 (Rao-Blackwell)

> Suppose $\mathbf{T}(\mathbf{x})$ is sufficient for θ and that $E_\theta[|S(\mathbf{x})|] < \infty$ for all $\theta \in \Theta$. Let $T^*(\mathbf{x}) = E[S(\mathbf{x})|T(\mathbf{x})]$. Then for all $\theta \in \Theta$, $E_\theta[T^*(\mathbf{x}) - q(\theta)]^2 \leq E_\theta[S(\mathbf{x}) - q(\theta)]^2$. If $Var_\theta[S] < \infty$, strict inequality holds unless $T^*(\mathbf{x}) = S(x)$.

Proof) See [59] pp.121.

Theorem 2 (Lehmann-Scheffe)

> If $T(\mathbf{x})$ is a complete sufficient statistic and $S(\mathbf{x})$ is an unbiased estimate of $q(\theta)$, then $T^*(\mathbf{x}) = E[S(\mathbf{x})|T(\mathbf{x})]$ is an U. M. V. U.

(uniformly minimum variance unbiased) estimate of $q(\theta)$. If $Var_\theta[T^*(\mathbf{x})] < \infty$ for all θ, $T^*(\mathbf{x})$ is the unique U. M. V. U. estimate of $q(\theta)$.

Proof) See pp.[59] 122.

The idea of sufficiency is to reduce the data with statistics whose use involves no loss of information. A statistic $T(\mathbf{x})$ is called sufficient for a parameter θ, if and only if, the conditional distribution of x given $T(\mathbf{x}) = t$ does not involve θ. Thus, once the value of a sufficient statistic T is known, the sample $\mathbf{x} = (x_1, \ldots, x_n)$ does not contain any further information about θ.

A statistic T is said to be complete, if the only real valued function g defined on the range of T which satisfies

$$E_\theta[g(T)] = 0 \quad \text{for all } \theta$$

is the function $g(T) = 0$. Completeness is evidently a property of the family of distributions of T generated as θ varies.

Theorem 3

Let $\{P_\theta : \theta \in \Theta\}$ be a k parameter exponential family as given by

$$P(\mathbf{x},\theta) = \{\exp\left[\sum_{i=1}^{k} c_i(\theta)T_i(\mathbf{x}) + d(\theta) + s(\mathbf{x}) \right] \} I_A(\mathbf{x})$$

Suppose that the range of $c = (c_1(\theta), \ldots, c_k(\theta))$ has a non-empty interior. Then, $T(\mathbf{x}) = (T_1(\mathbf{x}), \ldots, T_k(\mathbf{x}))$ is complete as well as sufficient.

Proof) See [59] pp.123.

Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a sample from a $N(\mu,\sigma^2)$ population where both μ and σ^2 are unknown. The distribution of $\mathbf{x}$ forms a two parameter exponential family in $q = (\mu,\sigma^2)$.

$$T(\mathbf{x}) = \left(\sum_{i=1}^{n} x_i, \sum_{i=1}^{n} x_i^2 \right) \tag{4.1}$$

is complete and sufficient. Since $\bar{x}$ is a function of T, it must be the U. M. V. U. estimate of μ if σ^2 is unknown.

$$\hat{\sigma}^2 = \frac{1}{(n-1)} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{4.2}$$

is also a function of the complete sufficient statistic T and the U.M.V.U. of σ^2 if μ is unknown.

The risk function of an estimate T(x) is defined by

$$R(\theta, T) = E_{\theta}^{x}[L(\theta, T(x))] = \int L(\theta, T(x))\, dF(x|\theta) \tag{4.3}$$

where $L(\theta, T(x))$ is loss function. One may choose the mean squared error as a loss function such that

$$L(\theta, T) = (\theta - T)^2 \tag{4.4}$$

Then

$$R(\theta, T) = E\left[(T(x) - q(\theta))^2\right]$$

$$= \text{Var}(T(x)) + \left[E(T(x) - q(\theta))\right]^2 \tag{4.5}$$

Let $T(x) = \hat{m}$ and $q(\theta) = m$. T(x) is the unbiased estimate of m. Then

$$R(m, \hat{m}) = \frac{\sigma^2}{n} \tag{4.6}$$

Let $T(x) = \hat{\sigma}^2$ and $q(\theta) = \sigma^2$. T(x) is the unbiased estimate of σ^2. Then

$$R(\theta, \hat{\sigma}^2) = \text{Var}(\hat{\sigma}^2)$$

$$= \frac{\sigma^4}{(n-1)^2} \text{Var}\left[\frac{(n-1)\hat{\sigma}^2}{\sigma^2}\right]$$

$$= \frac{2\sigma^4}{n-1} \tag{4.7}$$

Therefore the quality of an estimate is dependent upon the size of the training set and the variance. However, the above risk value cannot show what the relationships are between the amount of error and the quality of the estimates.

Usually, it is very difficult to obtain the risk value of a functional directly. Therefore, Taylor series expansion techniques may be applied to approximate the risk value of the functional*. The Taylor series expansion can be written as follows:

$$f(a+h, b+k) = f(a,b) + \left(h\frac{\delta}{\delta x} + k\frac{\delta}{\delta y}\right) f(x,y)\Big|_{x=a,\, y=b} + \ldots. +$$

$$\frac{1}{n!}\left(h\frac{\delta}{\delta x} + k\frac{\delta}{\delta y}\right)^n f(x,y)\Big|_{x=a,\, y=b} + \ldots\ldots\ldots \tag{4.8}$$

where a+h, b+k are estimates and a, b are real parameters. If a+h and b+h are unbiased estimates, E[h] = 0 and E[k] = 0. Then, E [f(a+h,b+k)] is as follows.

$$E\left[f(a+h, b+k)\right] = f(a, b) + \frac{1}{2}E\left[\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right)^2 f(x, y)\Big|_{x=a,\, y=b}\right] + \ldots \tag{4.9}$$

Then, Var [f(a+h,b+k)] is as follows.

$$\mathrm{Var}\,[f(a+h, b+k)] = E\left[f(a+h, b+k) - E[f(a+h, b+h)]\right]^2$$

$$\cong E\left\{\left[\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right) f(x,y)\Big|_{x=a,\, y=b} + \frac{1}{2}\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right)^2 f(x,y)\Big|_{x=a,\, y=b}\right.\right.$$

$$\left.\left. - \frac{1}{2}E\left[\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right)^2 f(x,y)\Big|_{x=a,\, y=b}\right]\right]^2\right\}$$

---

* A Similar derivation of the functional relationship has appeared [60] since the derivation which follows was first obtained.

$$\cong E\left[\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right) f(x, y)\Big|_{x=a,\, y=b}\right]^2 \tag{4.10}$$

Bayes error ε^* can be expressed by

$$\varepsilon^* = \int_{-\infty}^{\infty} \min[P_1 p_1(x),\, P_2 p_2(x)]\, dx \tag{4.11}$$

and bounded by

$$\varepsilon^* \subseteq \sqrt{P_1 P_2} \int \sqrt{p_1(x) p_2(x)}\, dx \tag{4.12}$$

If the class-conditional density functions are Gaussian, then

$$\varepsilon^* \subseteq \sqrt{P_1 P_2}\, \exp[-B] \tag{4.13}$$

where

$$B = \frac{1}{8}(m_1 - m_2)^T \left[\frac{\Sigma_1 + \Sigma_2}{2}\right]^{-1} (m_1 - m_2) + \frac{1}{2} \ln \frac{\left|\frac{\Sigma_1 + \Sigma_2}{2}\right|}{\sqrt{|\Sigma_1||\Sigma_2|}}$$

If the Bayes error is assumed to be directly related to the Bhattacharyya distance, the estimated Bhattacharyya distance behavior can show that the increment of Bayes error has as its origin the inaccurately estimated Bhattacharyya distance. However, the Bayes error is not bounded by the Bhattacharyya distance but by a function of Bhattacharyya distance.

The transformed Bhattacharyya distance is defined as follows:

$$X_B = 1 - \sqrt{P_1 P_2}\, \exp[-B] \tag{4.14}$$

The transformed Bhattacharyya distance is assumed to be directly related to classification accuracy. Assume that the Bayes error is approximately equal to the upper bound that is characterized by Bhattacharyya distance, then

$$\varepsilon^* \cong \sqrt{P_1 P_2}\, e^{-B} \tag{4.15}$$

The transformed Bhattacharyya distance is a lower bound of the correct classification accuracy. If P_1 and P_2 are equal, the estimated error of the classifier designed by training samples can be expressed as

$$\hat{\varepsilon} \cong \frac{1}{2} e^{-\hat{B}} \tag{4.16}$$

In multivariate statistical analysis, a most powerful property is that the Bhattacharyya distance is invariant under any one-to-one mapping. By the simultaneous diagonalization,

$$\mathbf{m}^{(1)} = \mathbf{0},\ \mathbf{m}^{(2)} = \mathbf{m},\ \Sigma^{(1)} = \mathbf{I},\ \Sigma^{(2)} = \Lambda \tag{4.17}$$

The number of parameters for the estimated Bhattacharyya distance is $2(q + q^2)$.

$$\hat{\mathbf{m}}^{(1)} = \begin{bmatrix} \hat{m}_1^{(1)} \\ \hat{m}_2^{(1)} \\ \vdots \\ \hat{m}_q^{(1)} \end{bmatrix} \qquad \hat{\Sigma}^{(1)} = \begin{bmatrix} \hat{\lambda}_{11}^{(1)} & \hat{\lambda}_{12}^{(1)} & \cdots & \hat{\lambda}_{1q}^{(1)} \\ \hat{\lambda}_{21}^{(1)} & \cdots & & \hat{\lambda}_{2q}^{(1)} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \hat{\lambda}_{q1}^{(1)} & \cdots & & \hat{\lambda}_{qq}^{(1)} \end{bmatrix} \tag{4.18}$$

The estimated transformed Bhattacharyya distance can be expressed as

$$\hat{X}_B = f(\hat{m}_1^{(1)} \cdots \hat{m}_q^{(1)}\ \hat{m}_1^{(2)} \cdots \hat{m}_q^{(2)}\ \hat{\lambda}_{11}^{(1)} \cdots \hat{\lambda}_{qq}^{(1)}\ \hat{\lambda}_{11}^{(2)} \cdots \hat{\lambda}_{qq}^{(2)}) \tag{4.19}$$

The estimated Bhattacharyya distance is a biased estimate. The bias of estimated Bhattacharyya distance is well derived in [58,60].

$$E[(\hat{m}_i - m_i)^{2k+1}] = 0,\ k = 1,2,3, \ldots \tag{4.20}$$

$$E[(\hat{m}_i - m_i)^{2k}] = 1 \cdot 3 \cdots (2k-1)\frac{\lambda_{ii}^{2k}}{n^k} \tag{4.21}$$

$$E[(\hat{\lambda}_{ij} - \lambda_{ij})(\hat{\lambda}_{kl} - \lambda)] = \frac{\lambda_{ii}\lambda_{jj}}{n},\ i \neq j,\ i=k,\ j=l \text{ or } i=l,\ j=k$$

$$= \frac{2\lambda_{ii}^2}{n-1} \cong \frac{2\lambda_{ii}^2}{n},\ i=j=k=l \tag{4.22}$$

For the computation of the derivatives of the Bhattacharyya distance containing matrix, three basic matrix differential equations are needed.

$$\frac{\partial \Sigma^{-1}}{\partial \lambda_{ij}} = -\Sigma^{-1}\mathbf{U}_{ij}\Sigma^{-1}, \quad \frac{\partial |\Sigma|}{\partial \Sigma} = |\Sigma|\,(\Sigma^{-1})^T \tag{4.23}$$

$$\frac{\partial \Sigma^T \mathbf{A} \Sigma}{\partial \lambda_{ij}} = \mathbf{U}_{ij}\,\Sigma^T + \Sigma\,\mathbf{U}_{ij}^T \tag{4.24}$$

where $\mathbf{U}_{ij}$ has all zero valued components except that i^{th} column and j^{th} row component is one.

$$\frac{\partial^2 f(B)}{\partial x_i \partial x_j} = \frac{\partial f(B)}{\partial B}\frac{\partial^2 B}{\partial x_i \partial x_j} + \frac{\partial^2 f(B)}{\partial^2 B}\frac{\partial B}{\partial x_i}\frac{\partial B}{\partial x_j} \tag{4.25}$$

$$\frac{\partial B}{\partial m_i^{(1)}} = -\frac{\partial B}{\partial m_i^{(2)}} = -\frac{m_i^{(2)} - m_i^{(1)}}{2(1+\lambda_{ii})} \tag{4.26}$$

$$\frac{\partial^2 B}{\partial m_i^{(1)2}} = \frac{\partial^2 B}{\partial m_i^{(2)2}} = \frac{1}{2(1+\lambda_{ii})} \tag{4.27}$$

$$\frac{\partial B}{\partial \lambda_{ij}^{(1)}} = -\frac{m_i m_j}{4(1+\lambda_{ii})(1+\lambda_{jj})} + \frac{\delta_{ij}}{2(1+\lambda_{ii})} - \frac{\delta_{ij}}{4} \tag{4.28}$$

$$\frac{\partial B}{\partial \lambda_{ij}^{(2)}} = -\frac{m_i m_j}{4(1+\lambda_{ii})(1+\lambda_{jj})} + \frac{\delta_{ij}}{2(1+\lambda_{ii})} - \frac{\delta_{ij}}{4\lambda_{ii}} \tag{4.29}$$

$$\frac{\partial^2 B}{\partial \lambda_{ij}^{(1)} \partial \lambda_{ij}^{(1)}} = \frac{\delta_{ij}}{4(1+\lambda_{ii})(1+\lambda_{jj})}\left[\frac{m_i^2}{1+\lambda_{ii}} + \frac{m_j^2}{1+\lambda_{jj}}\right] + \frac{1}{4} - \frac{1}{2(1+\lambda_{ii})(1+\lambda_{jj})} \tag{4.30}$$

$$\frac{\partial^2 B}{\partial \lambda_{ij}^{(2)} \partial \lambda_{ij}^{(2)}} = \frac{\delta_{ij}}{4(1+\lambda_{ii})(1+\lambda_{jj})}\left[\frac{m_i^2}{1+\lambda_{ii}} + \frac{m_j^2}{1+\lambda_{jj}}\right] + \frac{1}{4\lambda_{ii}\lambda_{jj}} - \frac{1}{2(1+\lambda_{ii})(1+\lambda_{jj})} \tag{4.31}$$

$$\frac{\partial^2 B}{\partial \lambda_{ij}^{(1)} \lambda_{ji}^{(1)}} = \frac{1}{4(1+\lambda_{ii})(1+\lambda_{jj})}\left[\frac{m_i^2}{1+\lambda_{ii}} + \frac{m_j^2}{1+\lambda_{jj}}\right] + \frac{\delta_{ij}}{4} - \frac{\delta_{ij}}{2(1+\lambda_{ii})(1+\lambda_{jj})} \tag{4.32}$$

$$\frac{\partial^2 B}{\partial \lambda_{ij}^{(2)} \lambda_{ji}^{(2)}} = \frac{1}{4(1+\lambda_{ii})(1+\lambda_{jj})}\left[\frac{m_i^2}{1+\lambda_{ii}} + \frac{m_j^2}{1+\lambda_{jj}}\right] + \frac{\delta_{ij}}{4\lambda_{ii}\lambda_{jj}} - \frac{\delta_{ij}}{2(1+\lambda_{ii})(1+\lambda_{jj})} \tag{4.33}$$

$$\frac{\partial^2 B}{\partial \lambda_{ij}^{(2)} \partial \lambda_{ij}^{(2)}} = \frac{\delta_{ij}}{4(1+\lambda_{ii})(1+\lambda_{jj})}\left[\frac{m_i^2}{1+\lambda_{ii}} + \frac{m_j^2}{1+\lambda_{jj}}\right] + \frac{1}{4\lambda_{ii}\lambda_{jj}} - \frac{1}{2(1+\lambda_{ii})(1+\lambda_{jj})} \tag{4.34}$$

The computation of the derivatives of the transformed Bhattacharyya distance can be derived as follows.

$$\frac{\partial e^{-B}}{\partial B} = -e^{-B}, \frac{\partial^2 e^{-B}}{\partial B^2} = e^{-B} \tag{4.35}$$

$$\frac{\partial^2 e^{-B}}{\partial m_i^{(1)2}} = \frac{\partial e^{-B}}{\partial B}\frac{\partial^2 B}{\partial m_i^{(1)2}} + \frac{\partial^2 e^{-B}}{\partial B^2}\left(\frac{\partial B}{\partial m_i^{(1)}}\right)^2 = e^{-B}\left[\left(\frac{m_i}{2(1+\lambda_{ii})}\right)^2 - \frac{1}{2(1+\lambda_{ii})}\right] \tag{4.36}$$

$$\frac{\partial^2 e^{-B}}{\partial m_i^{(2)2}} = \frac{\partial e^{-B}}{\partial B}\frac{\partial^2 B}{\partial m_i^{(2)2}} + \frac{\partial^2 e^{-B}}{\partial B^2}\left(\frac{\partial B}{\partial m_i^{(2)}}\right)^2 = e^{-B}\left[\left(\frac{m_i}{2(1+\lambda_{ii})}\right)^2 - \frac{1}{2(1+\lambda_{ii})}\right] \quad (4.37)$$

$$\frac{\partial^2 e^{-B}}{\partial \lambda_{ij}^{(1)}\partial \lambda_{ij}^{(1)}} + \frac{\partial^2 e^{-B}}{\partial \lambda_{ij}^{(1)}\partial \lambda_{ji}^{(1)}} = e^{-B}\left[2\left(\frac{m_i m_j}{4(1+\lambda_{ii})(1+\lambda_{jj})}\right)^2 + \frac{1}{2(1+\lambda_{ii})(1+\lambda_{jj})} - \frac{1}{4}\right.$$

$$\left. - \frac{1}{4(1+\lambda_{ii})(1+\lambda_{jj})}\left(\frac{m_i^2}{1+\lambda_{ii}} + \frac{m_j^2}{1+\lambda_{jj}}\right)\right] \quad (4.38)$$

$$\frac{\partial^2 e^{-B}}{\partial \lambda_{ij}^{(2)}\partial \lambda_{ij}^{(2)}} + \frac{\partial^2 e^{-B}}{\partial \lambda_{ij}^{(2)}\partial \lambda_{ji}^{(2)}} = e^{-B}\left[2\left(\frac{m_i m_j}{4(1+\lambda_{ii})(1+\lambda_{jj})}\right)^2 + \frac{1}{2(1+\lambda_{ii})(1+\lambda_{jj})} - \frac{1}{4\lambda_{ii}\lambda_{jj}}\right.$$

$$\left. - \frac{1}{4(1+\lambda_{ii})(1+\lambda_{jj})}\left(\frac{m_i^2}{1+\lambda_{ii}} + \frac{m_j^2}{1+\lambda_{jj}}\right)\right] \quad (4.39)$$

$$\frac{\partial^2 e^{-B}}{\partial \lambda_{ii}^{(1)2}} = e^{-B}\left[\left(\frac{1}{2(1+\lambda_{ii})} - \frac{1}{4} - \left(\frac{m_i}{2(1+\lambda_{ii})}\right)^2\right)^2\right.$$

$$\left. - \left(\frac{1}{2(1+\lambda_{ii})}\right)^2\left(\frac{2m_i^2}{(1+\lambda_{ii})}\right) + \frac{1}{2(1+\lambda_{ii})^2} - \frac{1}{4}\right] \quad (4.40)$$

$$\frac{\partial^2 e^{-B}}{\partial \lambda_{ii}^{(2)2}} = e^{-B}\left[\left(\frac{1}{2(1+\lambda_{ii})} - \frac{1}{4\lambda_{ii}} - \left(\frac{m_i}{2(1+\lambda_{ii})}\right)^2\right)^2\right.$$

$$- \left(\frac{1}{2(1+\lambda_{ii})}\right)^2 \left(\frac{2m_i^2}{(1+\lambda_{ii})}\right) + \frac{1}{2(1+\lambda_{ii})^2} - \frac{1}{4\lambda_{ii}^2}\Bigg] \tag{4.41}$$

$$\left(\frac{\partial e^{-B}}{\partial m_i^{(1)}}\right)^2 = \left(\frac{\partial e^{-B}}{\partial m_i^{(2)}}\right)^2 = e^{-2B}\left[\frac{m_i}{2(1+\lambda_{ii})}\right]^2 \tag{4.42}$$

$$\left(\frac{\partial e^{-B}}{\partial \lambda_{ij}^{(1)}}\right)^2 + \frac{\partial e^{-B}}{\partial \lambda_{ij}^{(1)}}\frac{\partial e^{-B}}{\partial \lambda_{ji}^{(1)}} = \left(\frac{\partial e^{-B}}{\partial \lambda_{ij}^{(2)}}\right)^2 + \frac{\partial e^{-B}}{\partial \lambda_{ij}^{(2)}}\frac{\partial e^{-B}}{\partial \lambda_{ji}^{(2)}} = 2e^{-2B}\left[\frac{m_i\, m_j}{4(1+\lambda_{ii})(1+\lambda_{jj})}\right]^2 \tag{4.43}$$

$$\left(\frac{\partial e^{-B}}{\partial \lambda_{ii}^{(1)}}\right)^2 = e^{-2B}\left[\frac{1}{2(1+\lambda_{ii})} - \frac{1}{4} - \left(\frac{m_i}{2(1+\lambda_{ii})}\right)^2\right]^2 \tag{4.44}$$

$$\left(\frac{\partial e^{-B}}{\partial \lambda_{ii}^{(2)}}\right)^2 = e^{-2B}\left[\frac{1}{2(1+\lambda_{ii})} - \frac{1}{4\lambda_{ii}} - \left(\frac{m_i}{2(1+\lambda_{ii})}\right)^2\right]^2 \tag{4.45}$$

The bias of the transformed Bhattacharyya distance can be obtained as follows.

$$E\left[\frac{1}{2}e^{-B} - \frac{1}{2}e^{-\hat{B}}\right] \approx \frac{e^{-B}}{8n}\Bigg[q - \sum_{i=1}^{q}\frac{m_i^2}{(1+\lambda_{ii})}$$

$$+ \sum_{i=1}^{q}\sum_{j=1}^{q}\left(\frac{m_j^2(1+\lambda_{ii}\lambda_{jj})}{(1+\lambda_{jj})^2(1+\lambda_{ii})} - \frac{(1+\lambda_{ii}\lambda_{jj})}{4}\left(\frac{m_i\, m_j}{(1+\lambda_{ii})(1+\lambda_{jj})}\right)^2\right)$$

$$+ \sum_{i=1}^{q}\left(\frac{m_i^2(1+\lambda_{ii}^2)}{(1+\lambda_{ii})^3} - \left(\frac{1}{(1+\lambda_{ii})} - \frac{1}{2} - \frac{m_i^2}{2(1+\lambda_{ii})^2}\right)^2\right)$$

$$+\sum_{i=1}^{q}\lambda_{ii}^2\left(\frac{1}{(1+\lambda_{ii})}-\frac{1}{2\lambda_{ii}}-\frac{m_i^2}{2(1+\lambda_{ii})^2}\right)^2\Bigg]$$

$$+\frac{e^{-B}}{8n}\left[q(q+1)-\sum_{i=1}^{q}\sum_{j=1}^{q}\frac{(1+\lambda_{ii}\lambda_{jj})}{(1+\lambda_{ii})(1+\lambda_{jj})}-\sum_{i=1}^{q}\frac{(1+\lambda_{ii}^2)}{(1+\lambda_{ii})^2}\right] \quad (4.46)$$

where q is the dimensionality, n is the number of samples, and B is the Bhattacharyya distance.

The variance of the transformed Bhattacharyya distance can be obtained as follows.

$$\mathrm{Var}\left[1-\frac{1}{2}e^{-\hat{B}}\right]\approx\frac{e^{-2B}}{16n}\left[\sum_{i=1}^{q}\frac{m_i^2}{(1+\lambda_{ii})}+\sum_{i=1}^{q}\sum_{\substack{j=1\\ i\neq j}}^{q}\frac{m_i^2m_j^2(1+\lambda_{ii}\lambda_{jj})}{2(1+\lambda_{ii})^2(1+\lambda_{jj})^2}\right.$$

$$+\sum_{i=1}^{q}2\left(\frac{1}{(1+\lambda_{ii})}-\frac{1}{2}-\frac{m_i^2}{2(1+\lambda_{ii})^2}\right)^2$$

$$\left.+\sum_{i=1}^{q}2\lambda_{ii}^2\left(\frac{1}{(1+\lambda_{ii})}-\frac{1}{2\lambda_{ii}}-\frac{m_i^2}{2(1+\lambda_{ii})^2}\right)^2\right] \quad (4.47)$$

Although the Bayes error from the estimated classifier is not the actual error, we want to find the risk value of the transformed Bhattacharyya distance because increasing the risk value makes the classification error increase. The risk function of the transformed Bhattacharyya distance is

$$R(\hat{X}_B,X_B)=\frac{1}{4}\left(E\left[e^{-B}-e^{-\hat{B}}\right]\right)^2+\mathrm{Var}\left[1-\frac{1}{2}e^{-\hat{B}}\right]$$

$$\approx\frac{1}{4}\left(E\left[e^{-B}-e^{-\hat{B}}\right]\right)^2 \quad (4.48)$$

In equation (4.48), one may note that the exponential term, $\frac{e^{-2B}}{16n}$, reduces the value of the risk function while the rest of the terms cause the value of the risk function increase. To minimize the risk value with a constraint to maximize the Bhattacharyya distance, the dimensionality must be reduced when the number of training samples is small.

If the features are ordered in decending order, the first feature reduces the risk function and expands the Bhattacharyya distance the most. As the number of features is increased, the summation terms increase more rapidly than the exponential term. The strategy for the prediction problem can be established as follows. If one wants to use as small a number of features as possible and achieve as a large Bhattacharyya distance as possible, one should take advantage of the transformed coordinates. The best one feature having the smallest risk value and largest Bhattacharyya distance between any two classes can be extracted in the transformed coordinates in the case of small training sample situation.

It may be noted from equation (4.46), that if only the mean difference term is considered as in the case of a linear classifier with a fixed Bhattacharyya distance, the bias increases linearly with the dimensionality, while if both the mean and covariance terms are considered with a fixed separability for the case of a quadratic classifier, the bias increases quadratically with the dimensionality.

4.2 Empirical Approach

Figure 4.1 shows in hypothetical fashion classification error for several cases. Case (a) shows the true class conditional density and (b) and (c) are estimated class-conditional densities. The area (1) is the true Bayes error. The area (2) or (3) is the estimated Bayes error which is obtained by the estimated parameters. The area (4) or (5) is the summation of Bayes error and increment error when the classifier is designed by training samples and the error rate estimated by test samples. In Section 4.1, the difference between area (1) and area (2) or area (3) is minimized. In this section, the incremental error due to the inaccurate estimates which is shown in the area (4) or (5) is studied empirically. Fukunaga and Krile[5] developed an algorithm for calculating recognition error when applying pattern vectors to an optimum Bayes' classifier. When the q random variables of the vector $\mathbf{x}$ are independent the minus-log-likelihood ratio $h(\mathbf{x})$ is as follows.

$$h(\mathbf{x}) = \sum_{l=1}^{q} h(x_l) \tag{4.49}$$

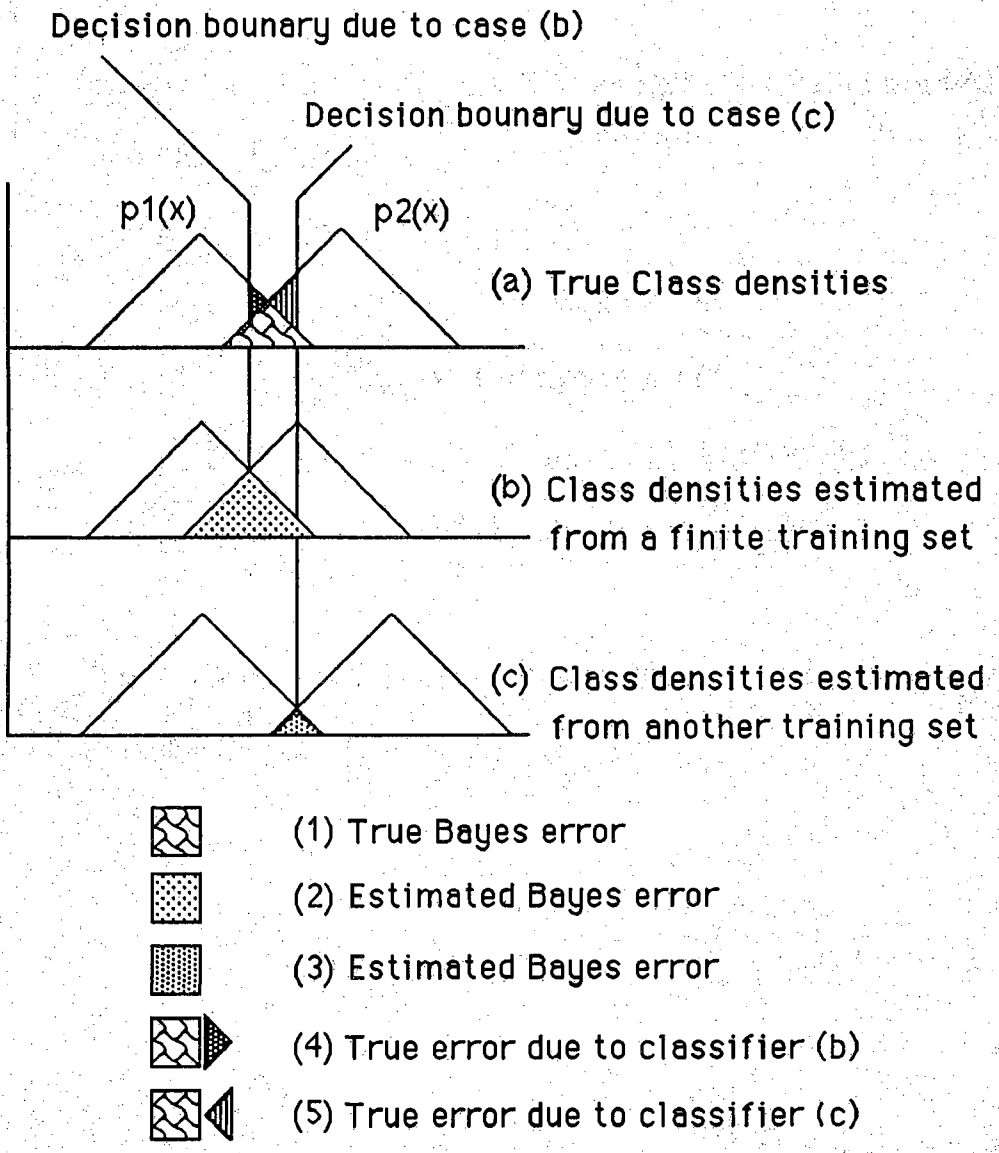


Figure 4.1 Class-conditional Densities and Decision Boundaries for a Hypothetical 2-class Case (a) True Class Densities (b), (c) Class Densities Estimated from a Finite Training Set

The characteristic function of h(**x**) for class *i* is

$$\phi_i(w) = E\left\{e^{jwh(\mathbf{x})} \mid \text{class } i\right\} = \int_{-\infty}^{\infty} e^{jwh(\mathbf{x})}\, p_i(\mathbf{x})\, d\mathbf{x}$$

$$= \sum_{l=1}^{q} \int_{-\infty}^{\infty} e^{jwh(x_l)}\, p_i(x_l)\, dx_l \qquad (4.50)$$

By definition, once the characteristic function of h(**x**) is obtained the density function of h(**x**) is its inverse Fourier transform.

$$p(h|\text{class i}) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \phi_i(w)\, e^{-jwh}\, dw \qquad (4.51)$$

When the distributions are normal, two covariance matrices can be diagonalized simultaneously by linear transform. In the transformed coordinate system, all features are independent. The errors are invariant under any transformation because the likelihood ratio is independent of any coordinate system. Characteristic functions of the minus log likelihood ratio for class 1 and class 2 can be easily computed because the q random variables of vector **x** are independent. This approach reduce the q-dimensional integral to a one-dimensional integral for the error from each class.

$$\varepsilon^* = P_1 \int_{0}^{\infty} p_1(h)\, dh + P_2 \int_{-\infty}^{0} p_2(h)\, dh \qquad (4.52)$$

The increment error due to the inaccurately trained classifier may be expressed as

$$\Delta\varepsilon = \hat{\varepsilon} - \varepsilon^*$$

$$= \left(P_1 \int_{0}^{\infty} p_1(\hat{h})dh + P_2 \int_{-\infty}^{0} p_2(\hat{h})dh \right) - \left(P_1 \int_{0}^{\infty} p_1(h)dh + P_2 \int_{-\infty}^{0} p_2(h)dh \right) \qquad (4.53)$$

To investigate the global relationships between the dimensionality, the sample size, and the correct classification accuracy, a Monte Carlo simulation is used here. The true Bayes' error can be computed numerically by Fukunaga's algorithm if one has perfect information of the mean and covariance for Gaussian classes. Although only 1-dimensional numerical integration is needed for Fukunaga's algorithm, it is difficult to obtain accurate Bayes' error easily in high dimensions. Therefore, a more simple means to estimate the Bayes' error is needed to study relationships

between sample size, dimensionality, and added error empirically. Whitsitt and Landgrebe [50] suggested that if we let f = erf, then we are assured that the locus of (p_e, f) contains p_e = f, and in this sense, f approximates the error. The Chernoff bound for a multivariate normal distribution is given by

$$C(s) = \frac{1}{2}s(1-s)(m_1 - m_2)^T\{(1-s)\Sigma_1 + s\Sigma_2\}^{-1}(m_1 - m_2) + \frac{1}{2}\ln\frac{|(1-s)\Sigma_1 + s\Sigma_2|}{|\Sigma_1|^{1-s}|\Sigma_2|^s} \tag{4.54}$$

The Bhattacharyya distance B = C(0.5). The error function Bhattacharyya distance is defined by

$$E = 0.5 - 0.5\mathrm{erf}\left(\sqrt{B}\right) \tag{4.55}$$

The error function transformed Bhattacharyya distance is defined by

$$E_B = 1 - E = 0.5 + 0.5\mathrm{erf}\left(\sqrt{B}\right) \tag{4.56}$$

where the error function is given by

$$\mathrm{erf}(\sqrt{B}) = \int_{\sqrt{B}}^{\infty}\frac{\exp[-\frac{x^2}{2}]}{\sqrt{2\pi}}\,dx$$

Ambiguity and linearity are two significant characteristics of separability measures. It is empirically illustrated that the probability of correct classification and the error function transformed Bhattacharyya distance have a linear relationship. Figures 4.2, 4.3, and 4.4 show why the error function Bhattacharyya distance has linear relationship with Bayes' error, as explained in the following. The q is the dimensionality and n is the number of samples.

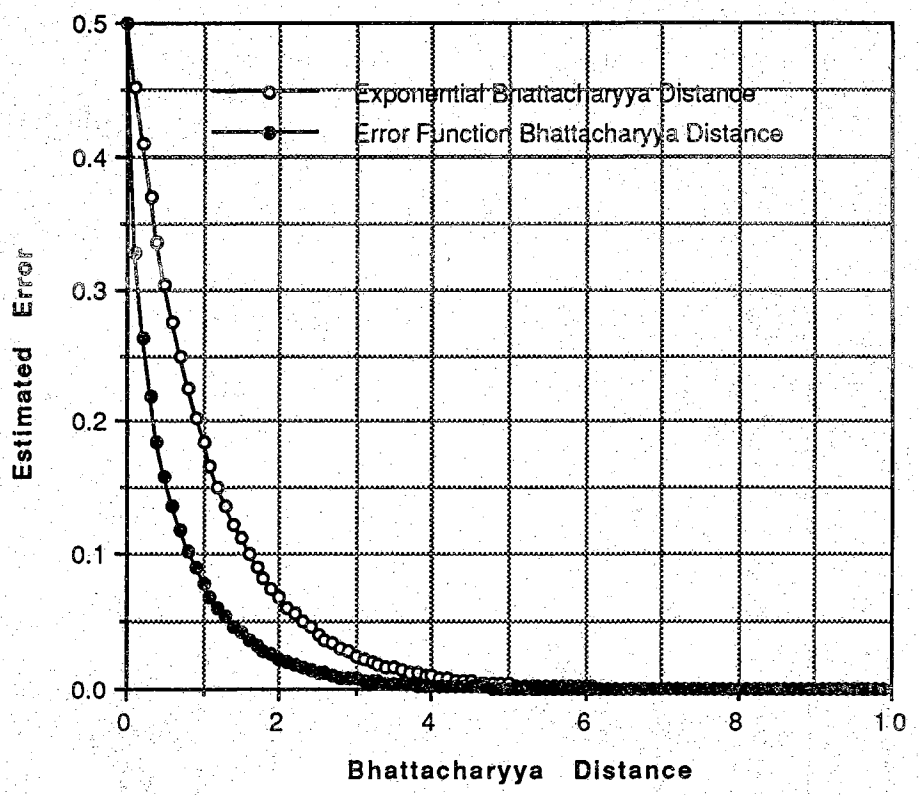


Figure 4.2 Simulation Result for Exponential Bhattacharyya Distance vs Error Function Bhattacharyya Distance (q=10, n=∞)

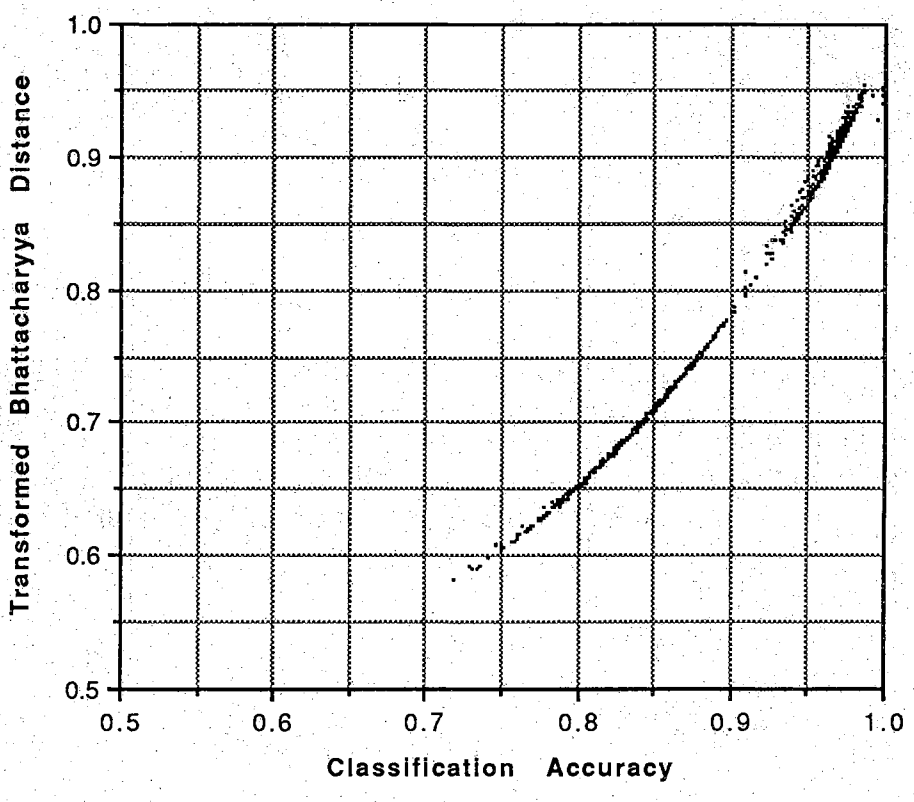


Figure 4.3 Simulation Result for P_c vs X_B (q = 10, n = ∞)

An empirical simulation was performed for ten and thirty dimensions. One thousand test samples are used to estimate the classification accuracy. The number of simulations in Figures 4.3 and 4.6 is one thousand at a given number of training samples. The error function Bhattacharyya distance is a tighter bound than the exponential Bhattacharyya distance as shown in Figure 4.2.

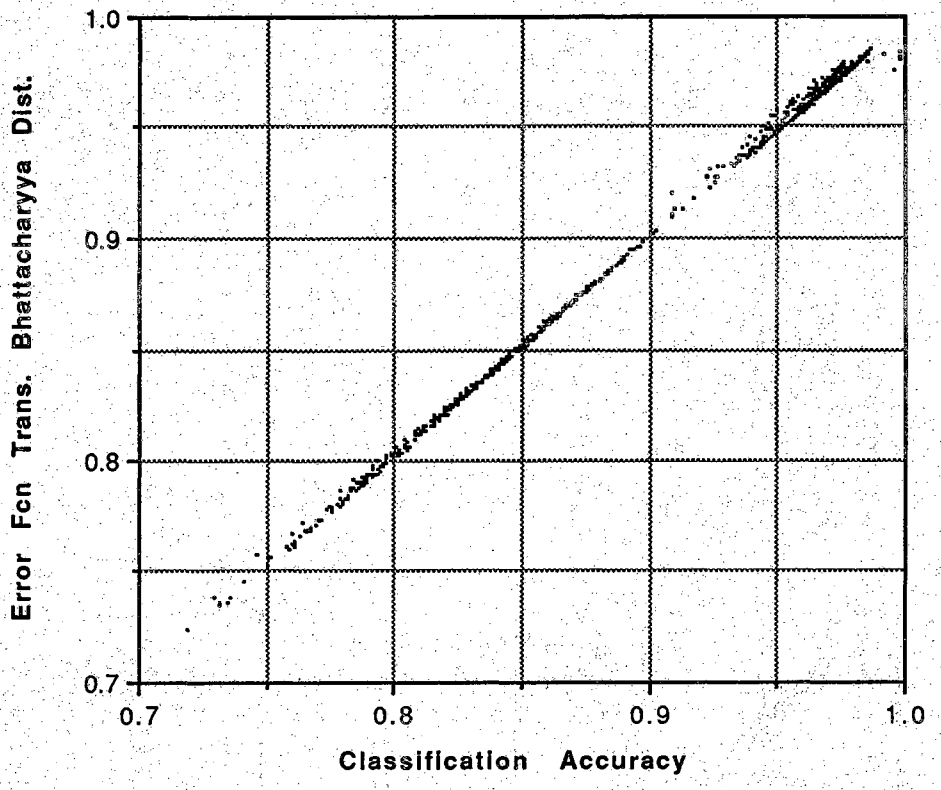


Figure 4.4 Simulation Result for P_c vs E_B ($q = 10$, $n = \infty$)

In Figure 4.3, classification accuracy versus transformed Bhattacharyya distance is plotted where the dimensionality is ten. In Figure 4.5, classification accuracy versus transformed Bhattacharyya distance is plotted where the dimensionality is thirty. The classification accuracy is obtained by Fukunaga's algorithm. By using the error function Bhattacharyya distance, classification accuracy versus error function transformed Bhattacharyya distance are illustrated in Figures 4.4 and 4.6. The probability of correct classification and the error function transformed Bhattacharyya distance are seen to have a linear relationship, and the error function Bhattacharyya distance is a tighter bound than the exponential Bhattacharyya distance.

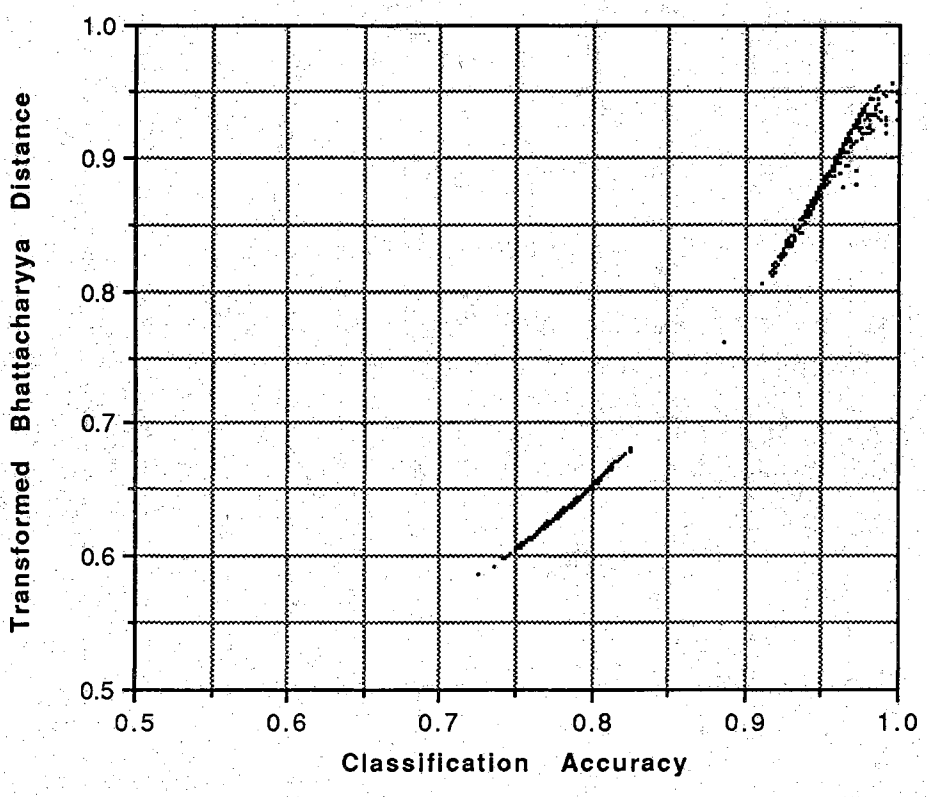


Figure 4.5 Simulation Result for P_c vs X_B (q = 30, n = ∞)

Therefore, E_B is selected to study relationships between sample size, dimensionality, and classification results, and to observe the Hughes phenomenon to determine the optimal number of features in two class cases. In Figure 4.7, the dimensionality problem such as the Hughes phenomenon is observed. The classification accuracy shown in Figure 4.7 is much less than the ideal classification accuracy in Figure 4.6.

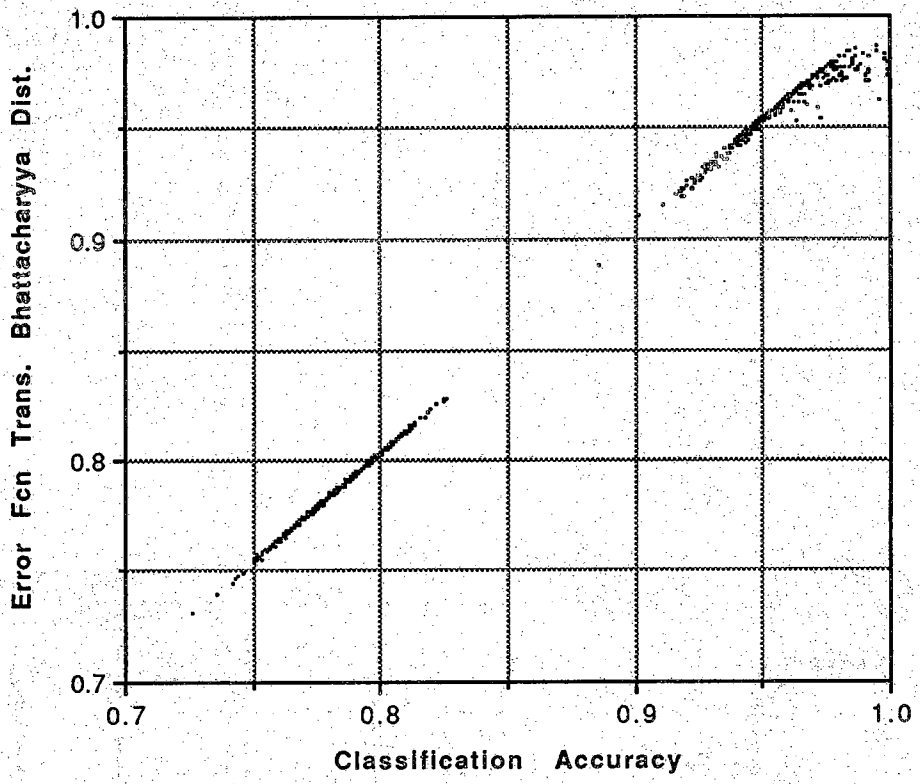


Figure 4.6 Simulation Result for P_c vs E_B (q = 30, n = ∞)

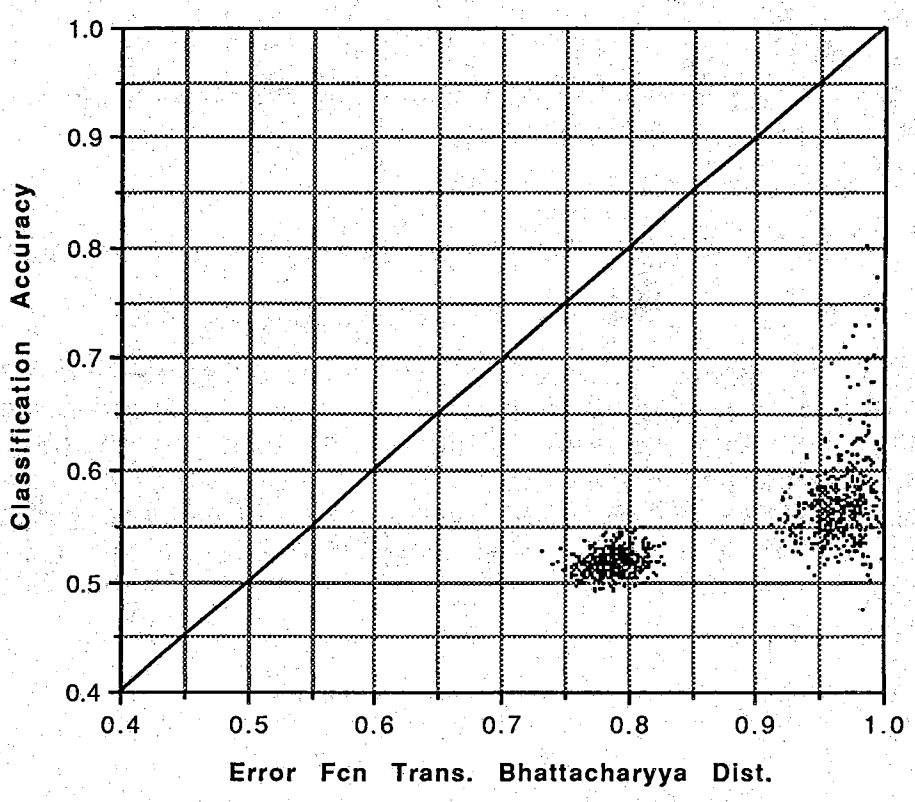


Figure 4.7 Simulation Result for P_c vs E_B (q = 30, n = 60)

Figure 4.7 shows that the estimated classification accuracy is well below the real classification accuracy when only 60 training samples are used to estimate the class-conditional densities in 30 dimensions. An empirical simulation was performed for from thirty to one hundred dimensions. One thousand test samples were used to estimate the classification accuracy. The number of simulations in Figures 4.8 and 4.9 is fifty for each given number of training samples. The corresponding number in Figures 4.7, 4.10, and 4.11 is one thousand.

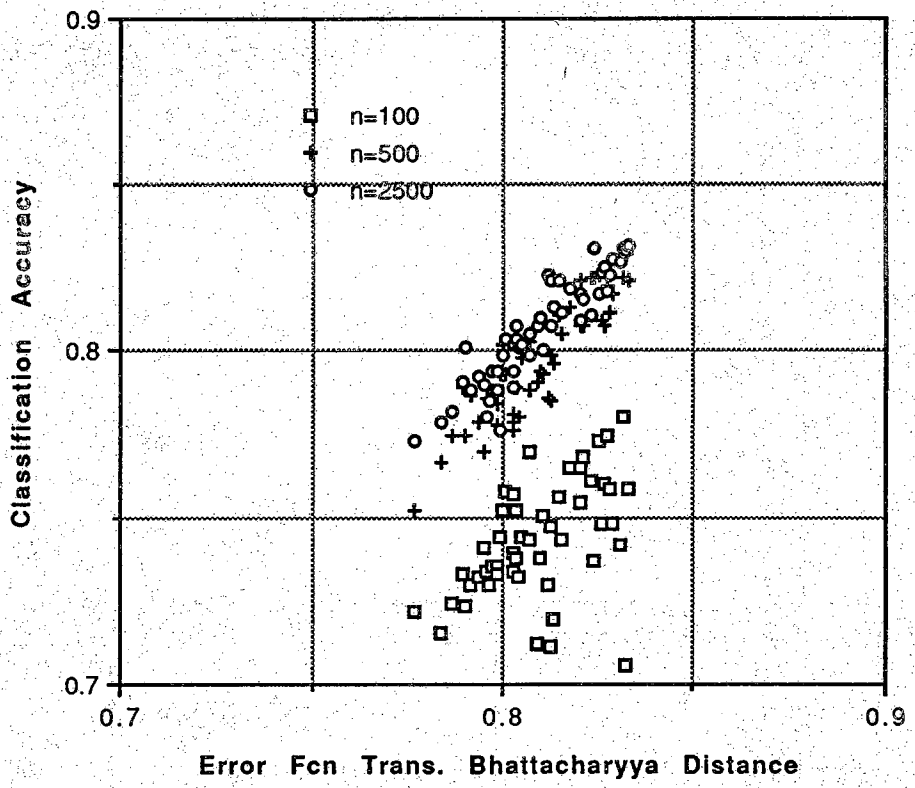


Figure 4.8 Classification Accuracy vs E_B (q = 50)

As the number of training samples is increased, P_c gradually approaches the ideal P_c. In Figures 4.8 and 4.9, two times, ten times the dimensionality and power of two of the dimensionality of the data are used to estimate the class-conditional densities in the simulations. When two times the dimensionality of the data are used, the estimated classification accuracy is well below the real classification accuracy. When ten times the dimensionality of the data are used, the estimated classification accuracy is almost the same as the real classification accuracy. When the power of two of the data are used, the estimated classification accuracy is similar to the result in case of ten times the dimensionality of the data.

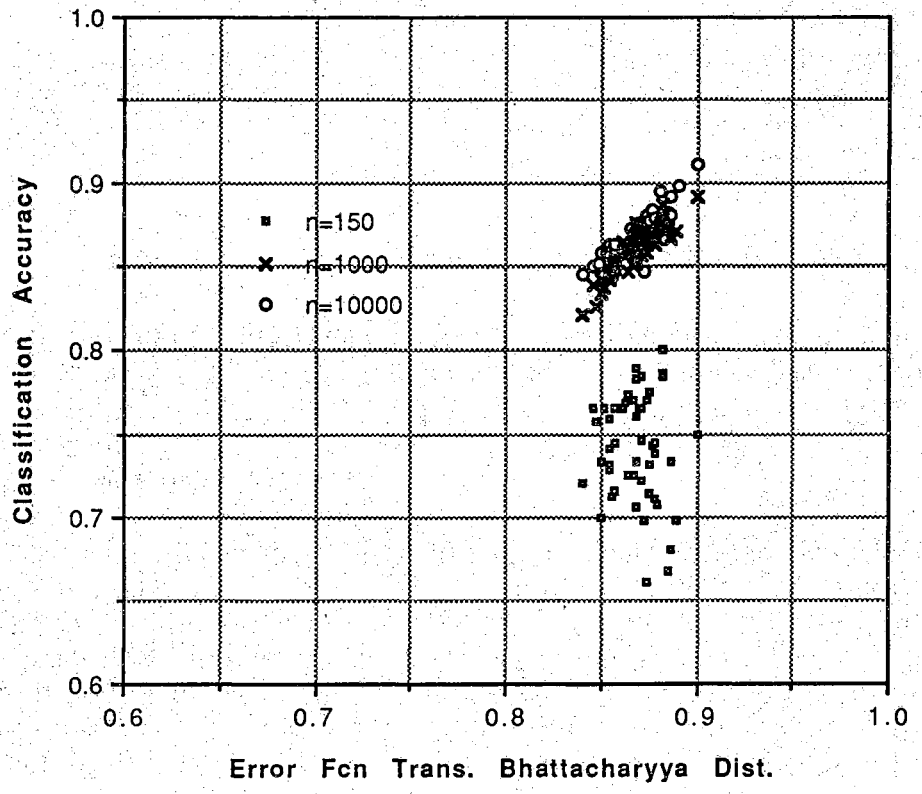


Figure 4.9 Classification Accuracy vs E_B (q = 100)

To illustrate the one-to-one relationship between estimated classification accuracy and E_B, the average classification accuracy are plotted in Figures 4.10 and 4.11. As a result, it appears that approximately six to ten times the number of training samples with respect to the dimensionality are needed to achieve a satisfactory design at this dimensionality.

As the dimensionality is increased, the separability is also increased since the added features give more information. When the dimensionality is increased with a fixed separability, the added error increases quadratically [60]. However, when the dimensionality and the separability increase together, it is difficult to find a simple relationship because the increased separability reduces the added error and the increased dimensionality cause the added error to increase, as in equation (4.48). In this section, the cases of increasing both dimensionality and separability are tested.

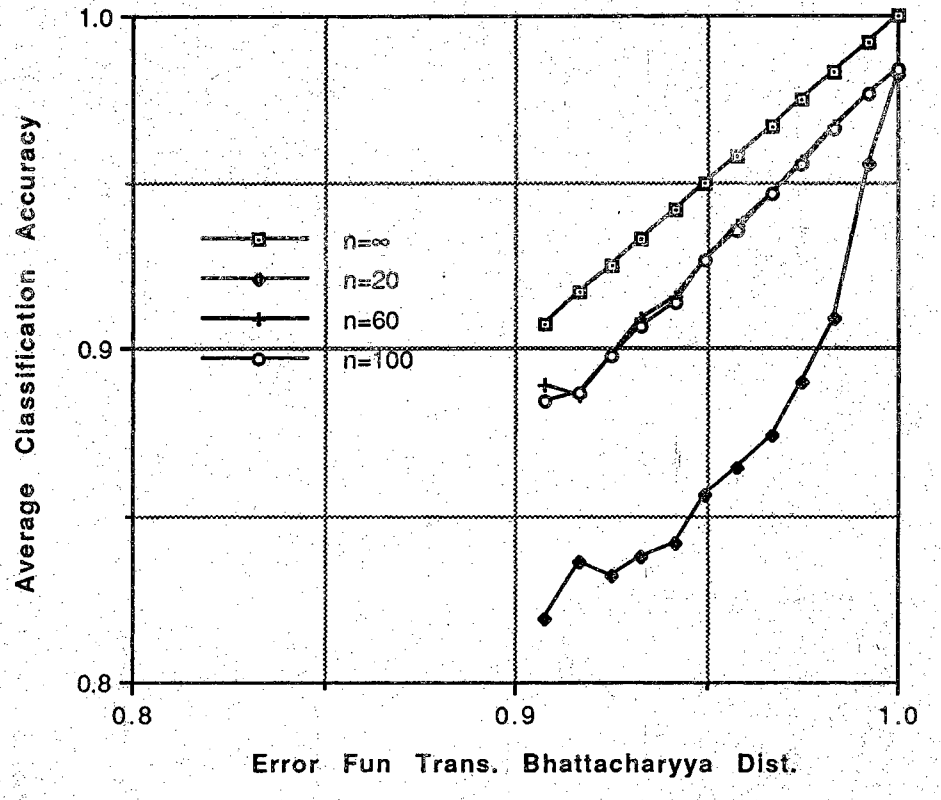


Figure 4.10 Mean Value of P_c vs E_B (q = 10)

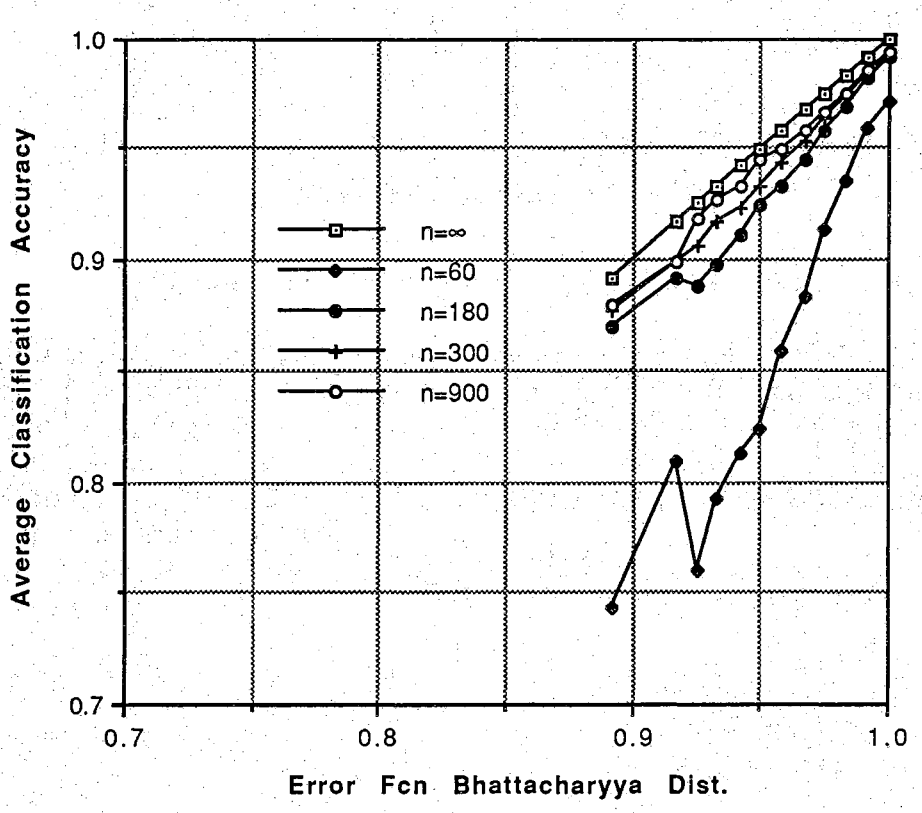


Figure 4.11 Mean Value of P_c vs E_B (q = 30)

The increment error is simultaneously affected by two factors, dimensionality and separability. For predicting the optimal number of features, we conclude that the optimal number of features in transformed coordinates is just one when only small numbers of samples are available. Empirically, it is shown that a reasonable sample size is six to ten times the dimensionality if the dimensionality and separability simultaneously increase.

CHAPTER 5. DATA PROCESSING AND EXPERIMENTS

5.1 Introduction

In this chapter, Decision Tree Classifiers (DTC's) designed by various procedures will be compared to verify which design procedure provides the better performance. Further matters presented are,

- a performance comparison of a DTC and a single layer classifier;
- a DTC approach for multitype data;
- the effects of the feature extraction in DTC design; and,
- a strategy for feature selection.

The Bayesian decision rule with the assumption of a 0-1 loss function and multivariate normal distributions is used as decision rule in all experiments when classification is involved. The 0-1 loss function assigns no loss to a correct decision, and unit loss to any error. Thus, all errors are assumed equally costly. Three kinds of data sets are used and will be referred to as follows: Flight Line C-1(hereafter referred to as FLC-1), Anderson River, Field Spectrometer System (FSS).

FLC-1 data were measured and recorded from an aircraft flight on June 28, 1966, at approximately 12:30 PM local time, at an altitude of 2600 feet above terrain in Tippecanoe County, Indiana. A spatially scanning radiometer with a 3 milliradian spatial resolution was used to obtain relative measurements of the energy reflected from the ground in twelve different wavelength bands. As shown in Table 5.1, the last two wavelength bands are in the reflective infrared portion of the spectrum. The other bands encompass the visible wavelengths. Part of the selected area is used for training and a much larger portion is used for testing.

Table 5.1 Multispectral Scanner Data of FLC-1

Feature No.	Spectral Band (microns)
1	0.40 - 0.44
2	0.44 - 0.46
3	0.46 - 0.48
4	0.48 - 0.50
5	0.50 - 0.52
6	0.52 - 0.55
7	0.55 - 0.58
8	0.58 - 0.62
9	0.62 - 0.66
10	0.66 - 0.72
11	0.72 - 0.80
12	0.80 - 1.00

Table 5.2 FSS Data

Location	Date
Kansas	9-28-76
Kansas	5-03-77
Kansas	6-26-77
North Dakota	5-08-77
North Dakota	6-29-77
North Dakota	8-04-77

Six sets of high spectral resolution field measurement data were taken over Williams County, North Dakota and Finney County, Kansas. These data were taken by the Field Spectrometer System (FSS) mounted in a helicopter. The spectral resolution was 0.02 μm for the interval from 0.4 μm to 2.4 μm. Location and date information is given in Table 5.2.

The Anderson River data set consist of 11 bands of airborne multispectral scanner (A/B MSS) data, 4 bands (X and L) of synthetic aperture radar imagery (horizontal polarization transmit and horizontal/vertical polarization receive) and digital terrain model information including digital elevation, slope and aspect (DEM, DSM and DAM respectively). The A/B MSS band intervals are given in Table 5.3.

Table 5.3 A/B MSS for Anderson River

Feature No.	Spectral Band (microns)
1	0.38 - 0.42
2	0.42 - 0.45
3	0.45 - 0.50
4	0.50 - 0.55
5	0.55 - 0.60
6	0.60 - 0.65
7	0.65 - 0.69
8	0.70 - 0.79
9	0.80 - 0.89
10	0.92 - 1.10
11	8.00 - 14.0

The A/B MSS Anderson River data was obtained over a Canadian forest site (2.8 km by 2.8 km) on July 29, 1978 at an altitude 3100 meters above sea level. The spatial resolution was 7 meters. Weather conditions were clear. Steep Mode SAR data was measured on July 25, 1978 over the site at an altitude 6700 meters above sea level. The raw data resolution was 3 meters. The X band wavelength is 3 cm and L band wavelength is 23 cm. Shallow Mode SAR data was obtained on July 31,1978 at an altitude 6400 meters above sea level.

5.2 Comparisons for Bottom Up DTC

In this section, three tree design methods, single linkage, complete linkage, and dynamic linkage are used to design a bottom up DTC as described in section 2.3. To design the bottom up DTC, mean vectors and covariance matrices are estimated from the training samples. The Bhattacharyya distance is used to estimate the separability between groups because of its smaller ambiguity [50].

Experiment 5.2.1

Eight classes of FLC-1 data were selected as follows: Alfalfa, Corn, Oats, Red Clover, Soybeans, Wheat, Bare Soil, and Rye. As shown in Table 5.4, the number of training samples for each class was chosen such that it is only slightly larger than the number of spectral features since it is commonly the case in remote sensing situations that training set sizes are small. At least one more sample than the number of features is needed to avoid singular covariance matrices. A large number of samples were used to evaluate the classification accuracy as shown in Table 5.4.

Table 5.4 FLC-1 Data

Class	#Train	#Test	Dim
Alfalfa(b)	15	760	12
Corn(e)	15	1360	12
Oats(j)	15	1380	12
Red Clover(l)	15	1357	12
Soy Bean(p)	15	1058	12
Wheat(u)	15	492	12
Bare Soil(x)	15	1012	12
Rye(y)	15	2322	12

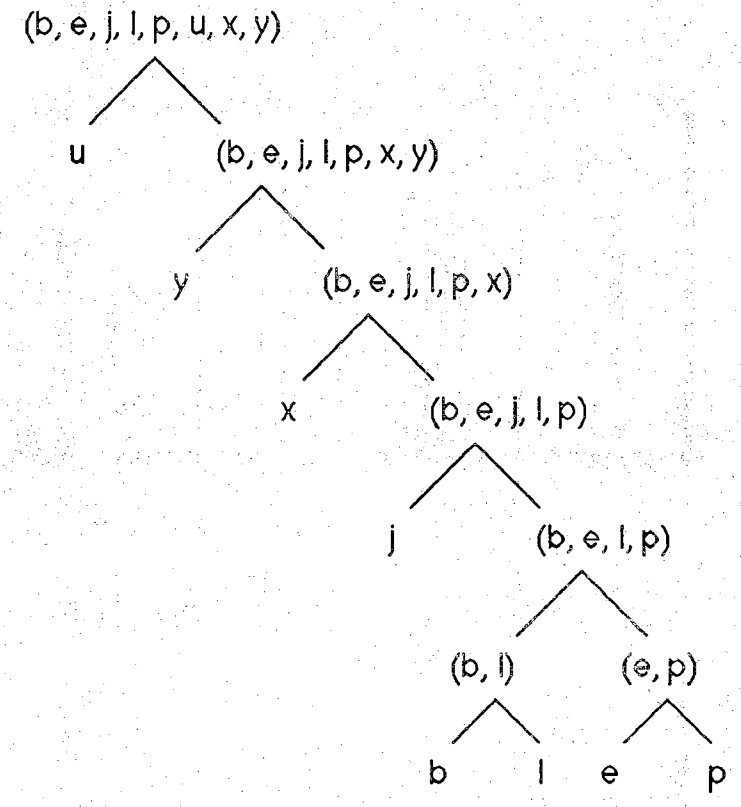


Figure 5.1 Single Linkage DTC (FLC-1, 8 Class)

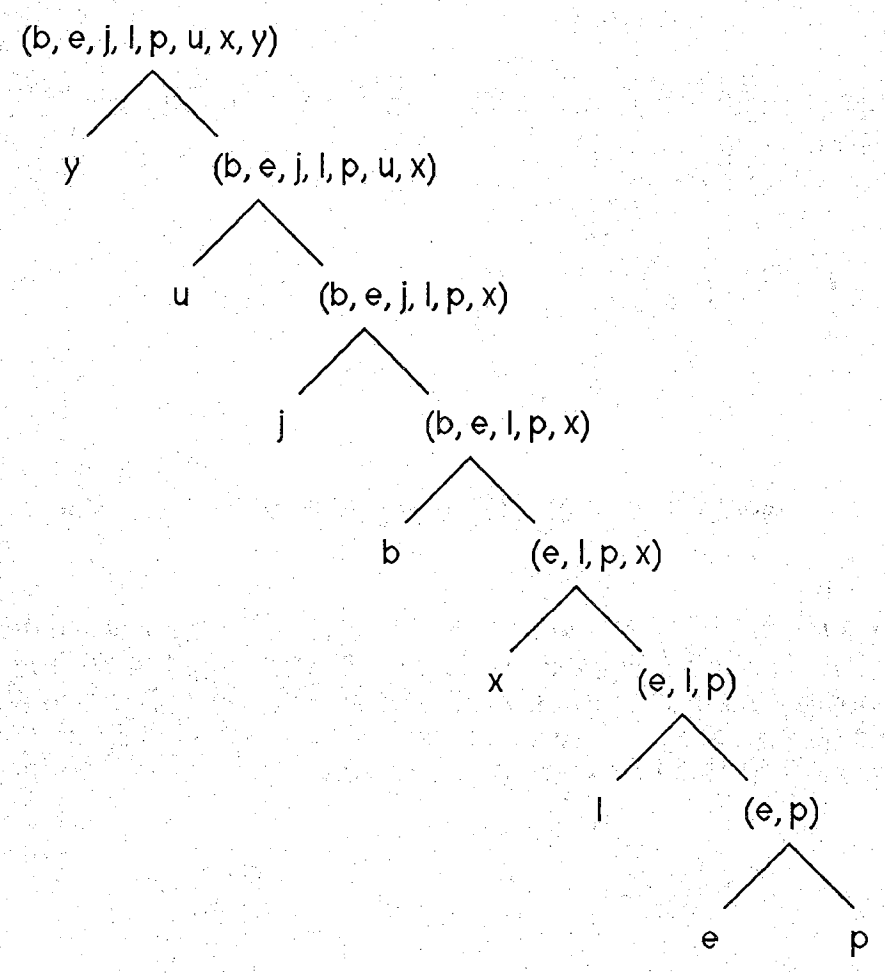


Figure 5.2 Dynamic Linkage DTC (FLC-1, 8 Class)

The resulting single linkage DTC for eight classes is shown in Figure 5.1 and the dynamic linkage DTC in Figure 5.2. The complete linkage DTC is shown in Figure 5.3. The results of classification for various numbers of features are given in Table 5.5, 5.6, and 5.7. Because the mean differences between classes are dominant, the extended canonical analysis for the feature subsets was used to seek maximum separability.

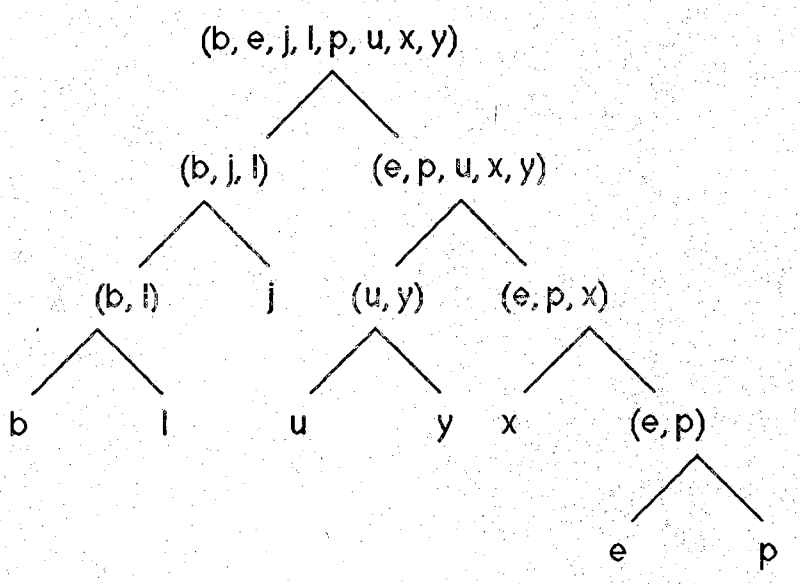


Figure 5.3 Complete Linkage DTC (FLC-1, 8 Class)

Table 5.5 Classification Accuracy(%) for the Single Linkage Design

Class	1*	2	3	4	5	6	7	12
Alfalfa	71.2	69.7	72.8	79.6	81.8	85.8	85.7	57.8
Corn	97.4	97.4	97.9	97.8	96.5	96.0	94.6	84.7
Oats	97.8	97.3	97.1	97.0	97.0	96.5	95.9	61.3
Clover	84.5	85.6	84.3	80.3	77.4	69.6	71.2	62.1
Bean	85.9	83.4	72.1	75.0	74.0	75.1	71.9	71.0
Wheat	99.2	94.7	93.3	91.5	90.9	83.7	77.4	24.4
Soil	91.6	91.4	90.7	92.0	90.9	85.2	80.7	20.7
Rye	94.5	93.2	86.6	86.6	86.6	85.8	84.2	22.4
Avg	90.3	89.1	86.9	87.5	86.9	84.7	82.7	50.6

* the number of features which is used at each node.

Table 5.6 Classification Accuracy(%) for the Complete Linkage Design

Class	1	2	3	4	5	6	7	12
Alfalfa	82.0	81.8	75.5	83.8	83.3	85.1	85.0	52.8
Corn	24.9	22.7	20.0	19.9	21.5	26.5	24.0	87.9
Oats	90.4	91.5	88.3	86.6	82.5	82.3	79.8	43.0
Clover	83.7	85.4	84.8	80.3	75.7	67.2	69.7	54.8
Bean	86.2	83.5	72.1	75.0	74.1	75.2	72.2	71.5
Wheat	99.6	99.6	99.4	99.8	99.4	99.6	98.0	98.8
Soil	100	100	100	100	100	99.7	99.7	95.6
Rye	96.9	96.5	96.5	96.1	96.0	96.9	98.7	97.9
Avg	83.0	82.6	79.6	80.2	79.1	79.1	78.4	75.3

Table 5.7 Classification Accuracy(%) for the Dynamic Linkage Design

Class	1	2	3	4	5	6	7	12
Alfalfa	91.3	84.2	82.8	69.5	63.4	58.8	51.6	6.3
Corn	96.0	96.5	96.9	97.1	97.4	97.3	97.1	88.9
Oats	97.2	97.0	96.7	96.7	96.1	95.5	94.3	60.3
Clover	88.4	91.2	90.1	87.6	86.4	78.0	74.7	59.9
Bean	85.4	83.0	71.5	74.6	73.4	74.6	71.8	71.5
Wheat	99.2	99.2	98.6	97.8	97.0	95.3	87.4	39.6
Soil	91.4	83.3	86.4	86.1	80.1	70.5	57.9	20.0
Rye	83.6	81.8	70.9	63.9	63.0	57.6	55.2	9.3
Avg	91.6	89.5	86.7	84.2	82.1	78.5	73.8	44.5

Figure 5.4 shows the average classification accuracy vs the number of features used for all three design methods. The dynamic linkage tree gave the best performance, and at the lowest feature dimensionality in this experiment. All three methods showed a decrease in classification accuracy as the number of features was increased. This characteristic is to be expected, given the small number of training samples since the quality of class statistics estimation become poorer with added features.

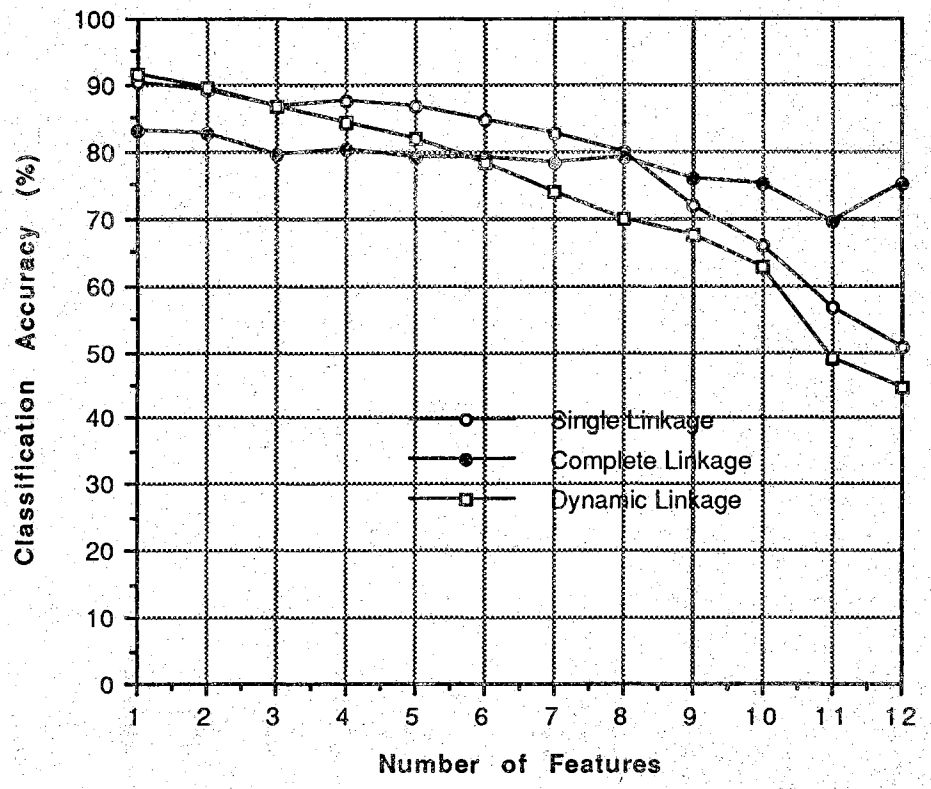


Figure 5.4 Average Classification Accuracy vs Number of Features Used for Experiment 5.2.1.

Experiment 5.2.2

Twenty three classes of FLC-1 data are chosen as shown in Table 5.8, to test performance of a bottom up DTC in case of a larger number of less separable classes. The twenty three classes consist of two alfalfa fields, four corn fields, four oats fields, three red clover fields, five soybeans fields, three wheat fields, bare soil, and rye field. Fifteen training samples for each class are chosen and at least 492 samples are used to evaluate the bottom up DTC. To test the more complex data, the same species located on different areas are considered as different classes. Since the mean difference of the Bhattacharyya distance is dominant between classes, canonical analysis for feature extraction was applied.

Table 5.8 FLC-1 Data (23 Class)

Class	#Train	#Test	Dim
Alfalfa1(a)	15	675	12
Alfalfa2(b)	15	760	12
Corn1(c)	15	651	12
Corn2(d)	15	1656	12
Corn3(e)	15	1360	12
Corn4(f)	15	1998	12
Oats1(g)	15	1034	12
Oats2(h)	15	737	12
Oats3(i)	15	1872	12
Oats4(j)	15	1380	12
Red Clover1(k)	15	1360	12
Red Clover2(l)	15	1357	12
Red Clover3(m)	15	836	12
Soy Bean1(n)	15	1189	12
Soy Bean2(o)	15	2491	12
Soy Bean3(p)	15	1053	12
Soy Bean4(q)	15	1349	12
Soy Bean5(r)	15	1890	12
Wheat1(s)	15	576	12
Wheat2(t)	15	671	12
Wheat3(u)	15	492	12
Bare Soil(x)	15	1012	12
Rye(y)	15	2322	12

Figures 5.5, 5.6, and 5.7 show the single linkage, the complete linkage, and the dynamic linkage DTC designs. All three classifiers are constructed by the bottom up approach.

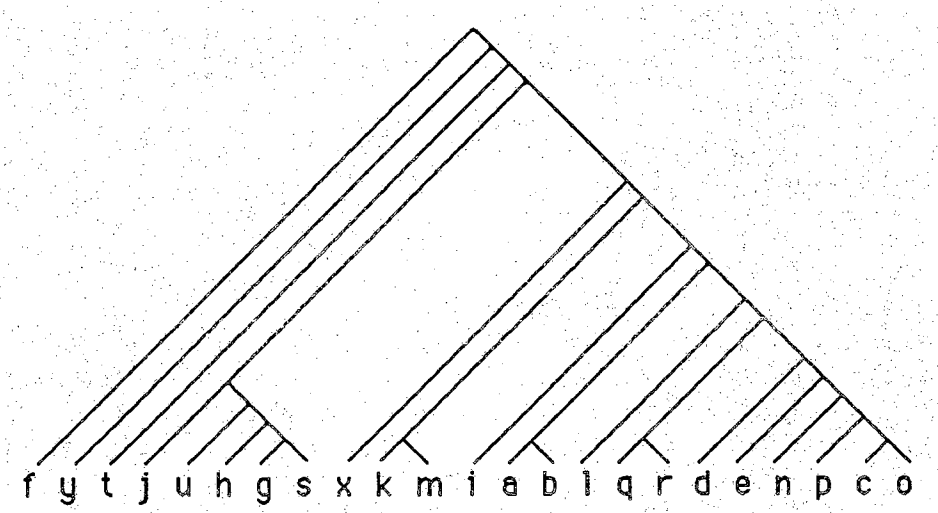


Figure 5.5 Single Linkage DTC (FLC-1, 23 Class)

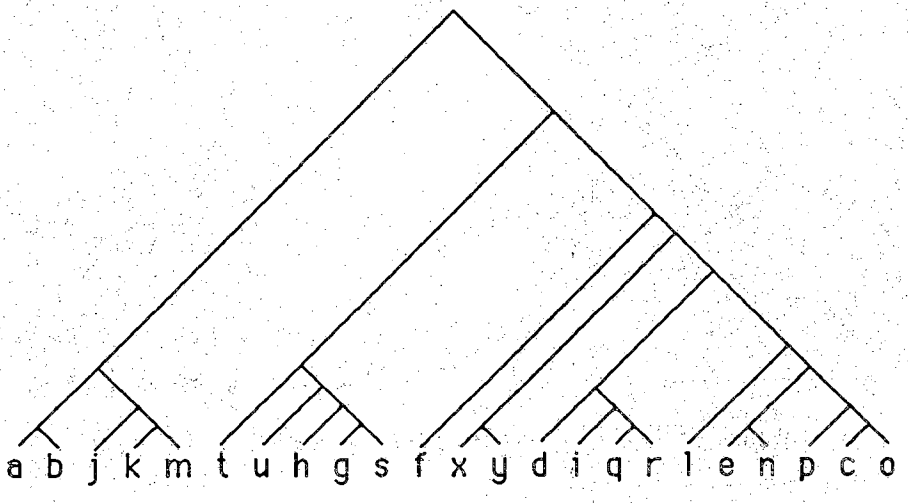


Figure 5.6 Complete Linkage DTC (FLC-1, 23 Class)

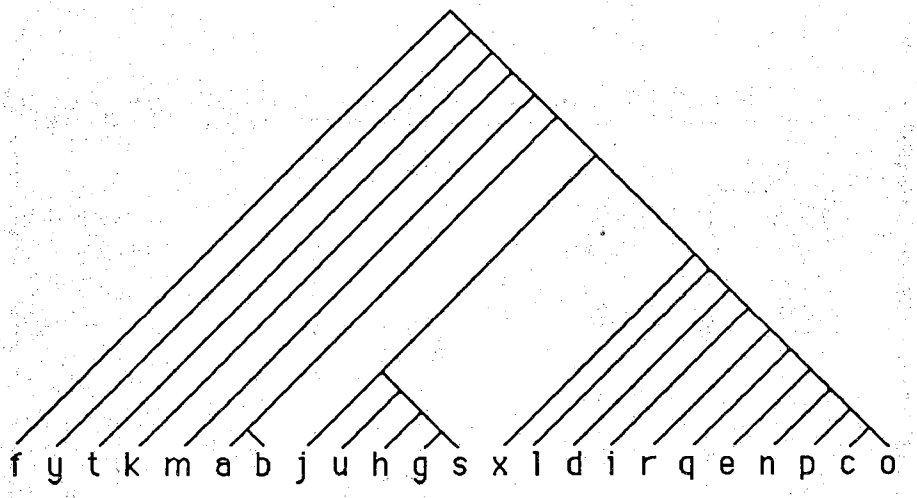


Figure 5.7 Dynamic Linkage DTC (FLC-1, 23 Class)

Table 5.9. Twenty Three Class Test Sample Accuracies in Per Cent.

Class	Single	Complete	Dynamic
Alfalfa1(a)	58.1	60.4	44.0
Alfalfa2(b)	40.5	45.5	29.5
Corn1(c)	57.1	20.3	36.3
Corn2(d)	76.9	66.9	72.8
Corn3(e)	87.2	59.6	74.0
Corn4(f)	0	0	0
Oats1(g)	57.8	47.1	53.6
Oats2(h)	67.4	71.6	67.4
Oats3(i)	51.6	4.7	78.0
Oats4(j)	71.0	70.4	85.3
Red Clover1(k)	69.7	93.8	69.3
Red Clover2(l)	81.5	51.9	85.0
Red Clover3(m)	36.5	37.8	42.6
Soy Bean1(n)	30.1	12.1	26.4
Soy Bean2(o)	33.6	31.4	33.3
Soy Bean3(p)	51.1	42.1	52.9
Soy Bean4(q)	6.9	9.8	11.9
Soy Bean5(r)	91.3	96.9	89.6
Wheat1(s)	42.9	42.9	42.9
Wheat2(t)	52.8	88.2	52.8
Wheat3(u)	97.6	99.4	97.6
Bare Soil(x)	94.7	98.8	97.7
Rye(y)	84.1	85.6	84.1
Average	58.3	53.8	57.7

Table 5.9 shows that the single linkage DTC gave slightly better performance than the dynamic linkage approach, with the complete linkage method somewhat lower than these.

Experiment 5.2.3

For a high dimensional data test of the bottom up design approach, ten classes of FSS which are spatially and temporally varying data were chosen as shown in Table 5.10. The ten classes consisted of three summer fallows fields, two unknowns fields, and five wheat fields. The FSS contained sixty-one spectral bands, however, since there are water absorption regions in the higher range, the first thirty features of the sixty-one were selected for this experiment.

Table 5.10 FSS Data

Class	#Train	#Test	Dim
Fallow1(a)	40	603	30
Fallow2(b)	40	374	30
Fallow3(c)	40	397	30
Unknown1(d)	40	642	30
Unknown2(e)	40	611	30
Wheat1(f)	40	618	30
Wheat2(g)	40	637	30
Wheat3(h)	40	891	30
Wheat4(i)	40	624	30
Wheat5(j)	40	747	30

Several hundred test samples for each class were used to insure a reliable estimate of performance. To simulate the limited sample situation, forty training samples for each class were selected on the basis of evenly spaced separability. The location information is given in Table 5.11.

Table 5.11 FSS Class Assignment

Location	Class
Kansas(9-28-76)	b, e
Kansas(5-3-77)	d, f
Kansas (6-6-77)	a, g
North Dakota(5-8-77)	c, i
North Dakota(6-29-77)	j
North Dakota(8-4-77)	h

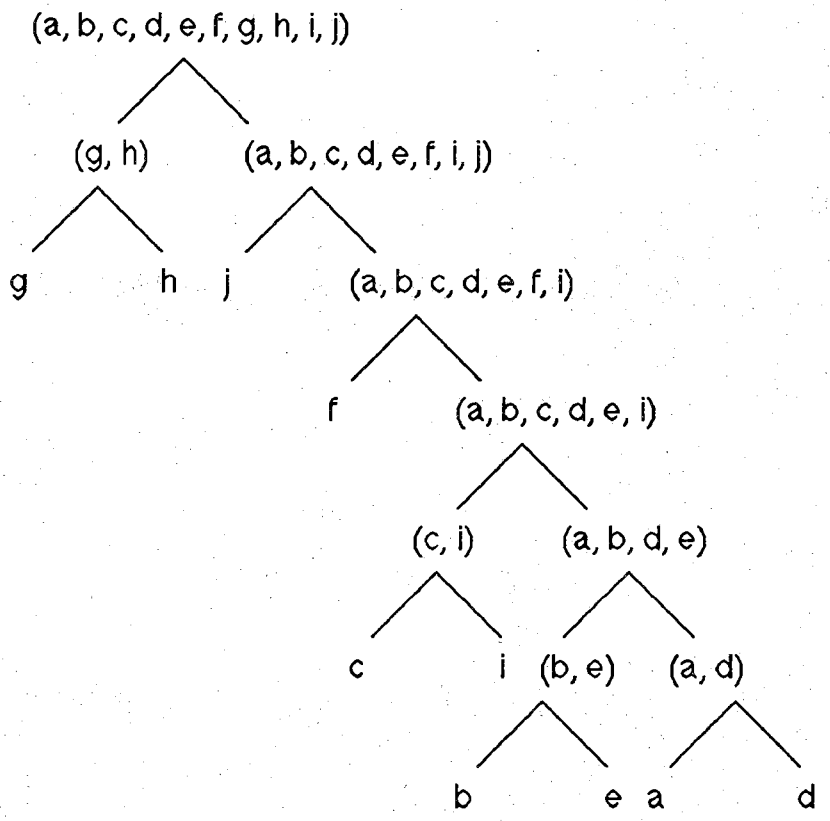


Figure 5.8 Single Linkage DTC (FSS)

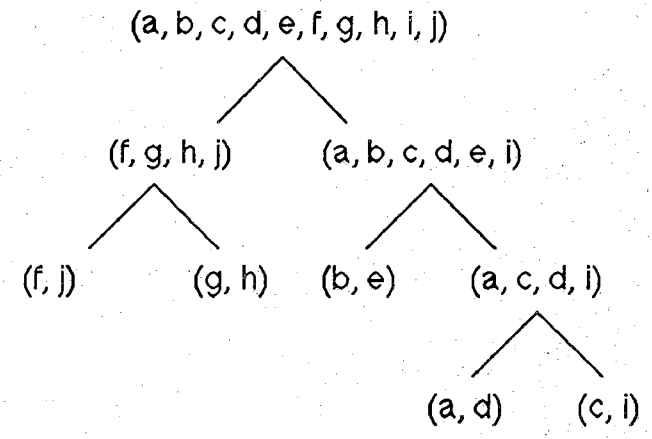


Figure 5.9 Complete Linkage DTC (FSS)

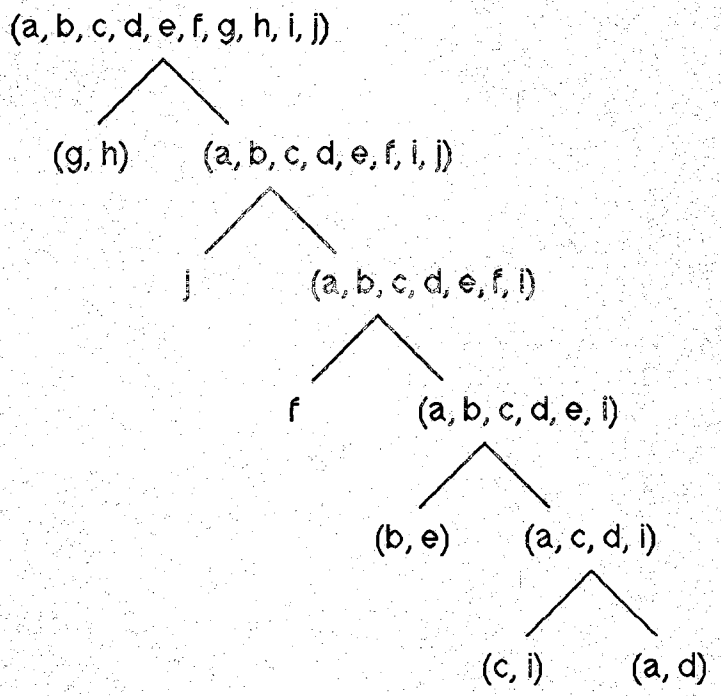


Figure 5.10 Dynamic Linkage DTC (FSS)

Figures 5.8, 5.9, and 5.10 show the single linkage, the complete linkage, and the dynamic linkage DTC. Table 5.12 shows the classification performance in each case. As is seen, there was very little difference in the performance of the three design approaches in this experiment.

Table 5.12 FSS Data Result

Class	Single	Complete	Dynamic
Fallow1(a)	55.9	63.5	63.7
Fallow2(b)	82.4	82.6	82.6
Fallow3(c)	66.0	67.5	61.7
Unknown1(d)	44.1	41.3	43.9
Unknown2(e)	69.7	71.0	71.7
Wheat1(f)	59.1	56.5	59.2
Wheat2(g)	84.0	83.5	84.0
Wheat3(h)	85.8	85.3	85.8
Wheat4(i)	48.7	49.4	50.6
Wheat5(j)	81.9	83.8	81.9
Average	67.8	68.4	68.5

In assessing the three experiments so far, although there was some difference in the performance of the three bottom up design procedures, no one method appears to clearly dominate. In cases where a bottom up approach is called for, any of the three might be useful, with a perhaps slight preference for the dynamic approach.

5.3 Top Down and Hybrid DTC

In this section, the top down DTC and the hybrid DTC will be compared. In the top down approach, a clustering algorithm is applied to separate the subgroups. the criterion function used for the top down approach is Euclidean distance while the criterion function for the hybrid approach is the normalized Euclidean distance. In the hybrid approach, the tree structure is dependent upon the initial cluster points. To obtain the initial points, a bottom up grouping method is used.

Experiment 5.3.1

Eight classes which are the same in experiment 5.2.1 were selected again and the same training and test sets were used. To construct the top down tree, the mean vector and covariance matrix of the combined training data were computed to obtain initial cluster centers uniformly spaced along the principal diagonal of the rectangular parallelepiped enclosing that data. After obtaining two subgroups, clustering is applied to each subgroup again. Figure 5.11 shows the top down DTC which results. Note here that are three overlapping classes , e, l, and u.

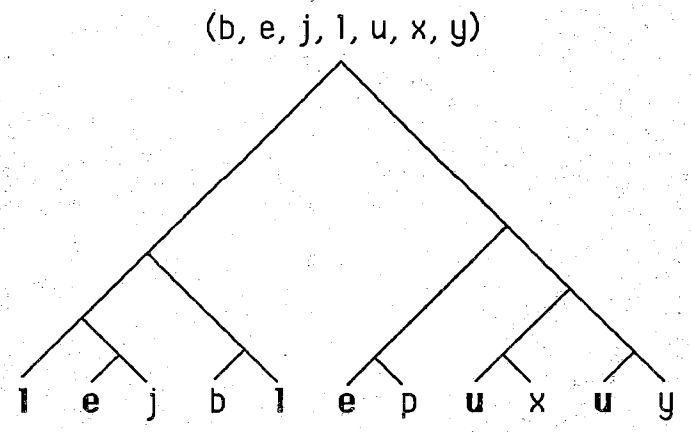


Figure 5.11 Top Down DTC (FLC-1, 8 Class)

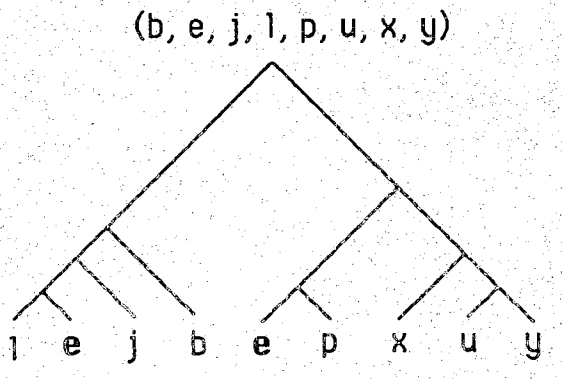


Figure 5.12 Hybrid DTC (FLC-1, 8 Class)

For the hybrid DTC, the complete linkage method is used to obtain the initial cluster centers and initial subgroups. Figure 5.12 shows the hybrid DTC. There is one overlapping class, e. For feature extraction, extended canonical analysis is applied in each subspace to ascertain the largest Bhattacharyya distance. The hybrid DTC is seen to improve the classification accuracy as shown in Table 5.13 and 5.14.

Table 5.13 Top Down DTC Result (FLC-1, 8 Class)

Class	1*	2	3	4	5	6	7	12
Alfalfa	70.9+	58.8	61.2	65.5	59.3	65.8	55.7	49.7
Corn	97.1	96.5	96.3	96.8	98.8	98.2	98.5	81.8
Oats	97.2	96.5	96.2	95.4	92.6	88.6	87.5	76.7
Clover	87.0	88.9	91.6	85.2	83.3	81.6	70.4	37.9
Bean	85.9	83.1	72.1	74.7	73.8	74.7	71.6	70.9
Wheat	99.4	99.2	99.2	99.4	99.2	99.0	98.2	98.8
Soil	99.7	99.7	99.9	96.3	95.3	95.5	94.4	100
Rye	94.0	93.2	89.6	89.1	89.0	90.9	91.5	98.6
Avg	91.4	89.5	88.3	88.3	86.5	86.8	83.6	76.8

* the number of features which is used at each node.
+ correct classification accuracy(%)

Table 5.14 Hybrid DTC Result (FLC-1, 8 Class)

Class	1	2	3	4	5	6	7	12
Alfalfa	88.6	76.2	73.0	75.4	68.4	63.4	54.2	9.0
Corn	97.0	96.5	96.3	96.8	98.8	98.2	98.5	81.8
Oats	92.0	96.0	96.2	95.4	93.2	89.6	88.0	81.7
Clover	88.5	91.6	92.8	90.0	89.9	89.8	74.2	37.1
Bean	85.9	83.1	72.1	74.7	73.8	74.7	71.6	70.9
Wheat	99.4	99.2	99.2	99.4	99.2	99.0	98.2	98.8
Soil	99.7	99.7	99.9	99.9	99.9	100	100	94.8
Rye	95.5	93.5	93.7	93.4	93.3	94.0	94.4	93.2
Avg	93.3	92.0	90.4	90.6	90.0	88.6	84.9	71.0

Experiment 5.3.2

In order to test the top down and hybrid DTC design schemes against the case of large numbers of less separable classes, the twenty three classes, training and test samples of experiment 5.2.2 are again used. Figure 5.13 shows the top downDTC resulting. Note that there are two overlapping classes c and k.

For the hybrid DTC, the complete linkage method was used to obtain the initial cluster centers and initial subgroups. In the first stage, two subgroups were initially determined by complete linkage. After the mean vectors and covariance matrices of the subgroups were obtained, mean vectors became initial center points and covariance matrices were used to normalize the distance from sample to cluster point. After merging and migrating, new subgroups were obtained.

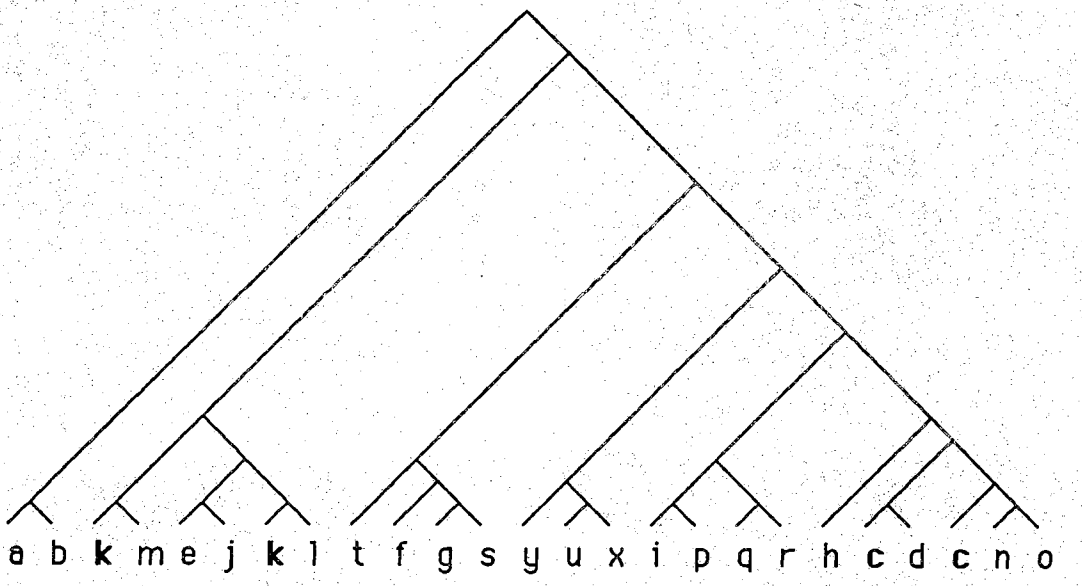


Figure 5.13 Top Down DTC (FLC-1, 23 Class)

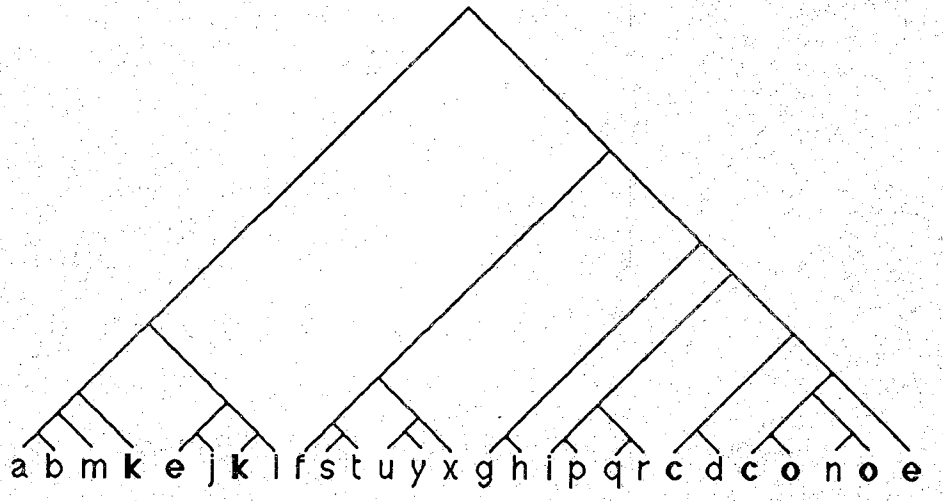


Figure 5.14 Hybrid DTC (FLC-1, 23 Class)

Figure 5.14 shows the hybrid DTC. There are four overlapping classes c, e, k and o. For feature extraction, extended canonical analysis was applied in each subspace to maximize the Bhattacharyya distance. The hybrid DTC reduced the

error rate by 5.2 percent with respect to the top down classifier in this experiment, as shown in Tables 5.15 and 5.16.

Table 5.15 Top Down DTC Result (FLC-1, 23 Class)

Class	1*	2	12
Alfalfa1(a)	31.7+	25.2	19.1
Alfalfa2(b)	18.3	22.1	7.1
Corn1(c)	39.3	39.8	22.4
Corn2(d)	66.6	66.6	59.2
Corn3(e)	80.4	79.6	62.2
Corn4(f)	0	0	0
Oats1(g)	38.0	38.9	1.1
Oats2(h)	56.5	55.4	35.8
Oats3(i)	37.1	39.5	19.7
Oats4(j)	70.9	70.8	72.9
Red Clover1(k)	82.7	81.7	92.8
Red Clover2(l)	84.0	86.2	81.9
Red Clover3(m)	17.3	16.8	17.2
Soy Bean1(n)	0.4	0.8	0
Soy Bean2(o)	39.2	40.2	72.8
Soy Bean3(p)	53.3	53.1	55.4
Soy Bean4(q)	8.3	37.8	61.4
Soy Bean5(r)	95.7	89.3	56.0
Wheat1(s)	49.3	49.5	5.9
Wheat2(t)	89.3	87.6	24.3
Wheat3(u)	81.7	82.1	98.0
Bare Soil(x)	100	100	99.9
Rye(y)	82.5	81.4	56.3
Average	53.2	54.1	44.4

* the number of features which is used at each node.
+ correct classification accuracy(%)

Table 5.16 Hybrid DTC Result (FLC-1, 23 Class)

Class	1	2	12
Alfalfa1(a)	41.2	35.0	67.9
Alfalfa2(b)	35.5	47.1	11.7
Corn1(c)	85.3	85.7	45.8
Corn2(d)	61.8	61.1	66.2
Corn3(e)	81.9	80.6	60.8
Corn4(f)	0	0	0
Oats1(g)	44.6	44.4	20.5
Oats2(h)	74.5	73.8	72.1
Oats3(i)	41.1	43.0	17.9
Oats4(j)	69.6	67.6	65.1
Red Clover1(k)	82.4	81.7	60.9
Red Clover2(l)	77.6	78.5	79.6
Red Clover3(m)	29.5	29.0	32.4
Soy Bean1(n)	20.0	20.3	21.8
Soy Bean2(o)	22.7	14.4	38.9
Soy Bean3(p)	53.0	52.8	52.5
Soy Bean4(q)	8.0	40.2	58.0
Soy Bean5(r)	95.6	89.1	56.0
Wheat1(s)	45.3	45.3	21.5
Wheat2(t)	94.9	97.6	97.4
Wheat3(u)	97.6	99.0	24.8
Bare Soil(x)	100	100	94.3
Rye(y)	82.4	85.5	95.4
Average	58.5	59.4	52.3

5.4 Bottom Up and Hybrid DTC

In the previous section, the hybrid DTC was shown to provide a greater classification accuracy than the top down DTC. The hybrid DTC will next be compared to the bottom up DTC. Table 5.17 shows that the hybrid DTC reduces the error rate by 4.7 % over the complete linkage DTC. Table 5.18 shows that the hybrid DTC improves 10.3 % classification accuracy over the complete DTC and has the highest performance among the methods tested.

Table 5.17 Hybrid and Bottom Up DTC (FLC-1, 23 Class)

Class	Hybrid	Single	Complete	Dynamic
Alfalfa1(a)	41.2	58.1	60.4	44.0
Alfalfa2(b)	35.5	40.5	45.5	29.5
Corn1(c)	85.3	57.1	20.3	36.3
Corn2(d)	61.8	76.9	66.9	72.8
Corn3(e)	81.9	87.2	59.6	74.0
Corn4(f)	0	0	0	0
Oats1(g)	44.6	57.8	47.1	53.6
Oats2(h)	74.5	67.4	71.6	67.4
Oats3(i)	41.1	51.6	4.7	78.0
Oats4(j)	69.6	71.0	70.4	85.3
Red Clover1(k)	82.4	69.7	93.8	69.3
Red Clover2(l)	77.6	81.5	51.9	85.0
Red Clover3(m)	29.5	36.5	37.8	42.6
Soy Bean1(n)	20.0	30.1	12.1	26.4
Soy Bean2(o)	22.7	33.6	31.4	33.3
Soy Bean3(p)	53.0	51.1	42.1	52.9
Soy Bean4(q)	8.0	6.9	9.8	11.9
Soy Bean5(r)	95.6	91.3	96.9	89.6
Wheat1(s)	45.3	42.9	42.9	42.9
Wheat2(t)	94.9	52.8	88.2	52.8
Wheat3(u)	97.6	97.6	99.4	97.6
Bare Soil(x)	100	94.7	98.8	97.7
Rye(y)	82.4	84.1	85.6	84.1
Average	58.5	58.3	53.8	57.7

Table 5.18 Hybrid and Bottom Up DTC (FLC-1, 8 Class)

Class	Hybrid	Complete	Single	Dynamic
Alfalfa2(b)	88.6	82.0	71.2	91.3
Corn3(e)	97.0	24.9	97.4	96.0
Oats4(j)	92.0	90.4	97.8	97.2
Red Clover2(l)	88.5	83.7	84.5	88.4
Soy Bean3(p)	85.9	86.2	85.9	85.4
Wheat3(u)	99.4	99.6	99.2	99.2
Bare Soil(x)	99.7	100	91.6	91.4
Rye(y)	95.5	96.9	94.5	83.6
Average	93.3	83.0	90.3	91.6

Experiment 5.4.1

In this experiment, the untransformed feature extraction method is applied at each node to compare the top down, the bottom up, and the hybrid DTC performance. The DTC's as shown in Figures 5.6, 5.13, and 5.14 are used for this experiment.

The number of features used for classification was chosen arbitrarily as three since classification results for this data commonly peaks at about three features. The feature subsets are selected based on the averaged pairwise Bhattacharyya distance. When the best three features are used over all nodes, the hybrid DTC has the best performance. Improvements of classification accuracy of a hybrid DTC is clearly observed, as shown in Table 5.19 and 5.20. It is noted that the classification accuracy of the hybrid DTC is about 5 % higher in eight classes and 9 % higher in twenty three classes than the bottom up DTC.

Table 5.19 Untransformed Best Three Feature Result (8 Class)

Class	a	c	o	r	s	w	x	y	Avg
Top down	58.0	98.0	66.0	78.4	85.6	99.6	99.9	98.6	85.5
Bottom up	58.2	97.4	98.7	78.7	85.7	99.6	82.6	52.1	81.6
Hybrid	66.4	98.2	66.0	81.0	85.6	99.6	99.9	98.5	86.9

Table 5.20 Untransformed Best Three Feature Result (23 Class)

Class	a	b	c	d	e	f	g	h
Top down	26.0	22.5	5.2	64.0	89.3	0	3.2	73.4
Bottom up	35.1	32.8	72.7	44.0	85.7	0	55.1	81.8
Hybrid	39.4	40.8	77.6	76.6	89.7	0	18.9	46.1

Class	i	j	k	l	m	n	o	p
Top down	37.8	57.7	97.0	77.7	23.9	8.7	7.4	55.9
Bottom up	20.2	59.7	51.5	64.9	24.0	7.0	23.7	49.6
Hybrid	27.0	90.4	76.3	75.0	37.7	8.0	17.4	55.9

Class	q	r	s	t	u	x	y	**Avg**
top down	40.3	75.4	0.5	5.1	98.6	100	71.9	**45.3**
bottom up	11.6	42.6	40.6	34.4	96.1	92.3	48.5	**46.7**
hybrid	34.7	74.0	20.0	77.5	98.4	100	94.0	**55.5**

5.5 DTC and Single Layer Classifier

As demonstrated in section 5.4, the hybrid DTC has the best performance among the methods tested. The following experiment is conducted mainly for the purpose of observing the dimensionality problem in multispectral recognition and comparing the DTC to the single layer classifier.

Experiment 5.5.1

The same eight classes, training and test as in experiment 5.2.1 and the twenty three classes, training and test as in experiment 5.2.2 were selected again. Two feature selection methods were used. The first feature selection method, called untransformed feature selection, used the pairwise comparison of Bhattacharyya distance. The second feature selection technique, called transformed feature selection used the canonical transform method. In the untransformed feature selection case, the error rate of the complete feature set was always higher than the best result which was obtained by using subsets of twelve features as shown in Figures 5.15 and 5.16. The best performance was achieved at between three and six features.

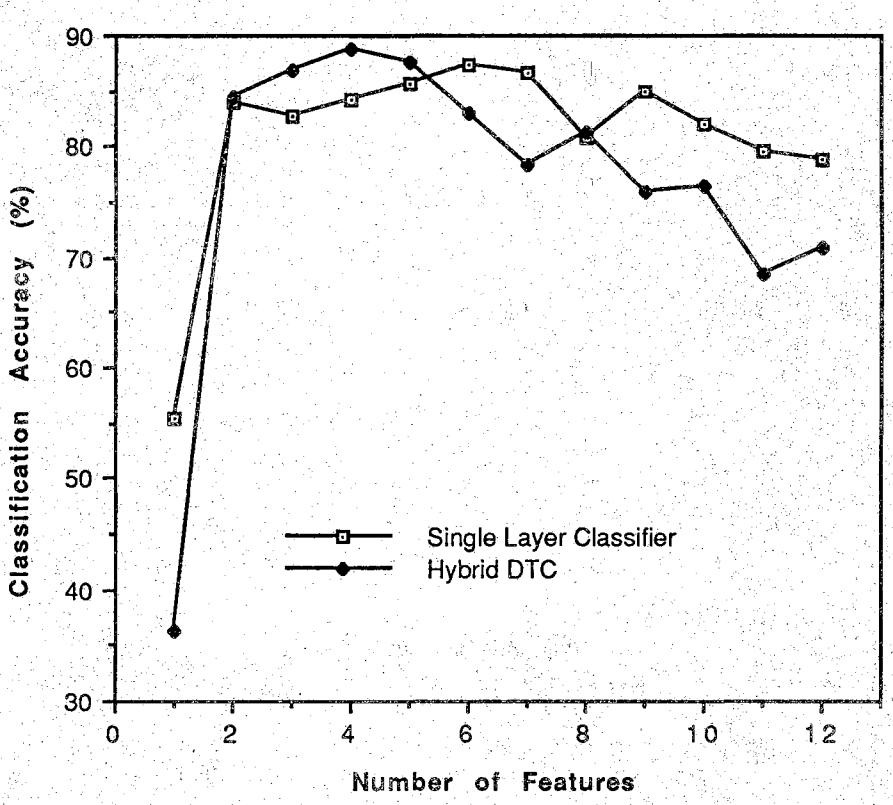


Figure 5.15 Untransformed Feature Selection Result (8 Class)

The hybrid DTC was compared to the single layer classifier in Figures 5.15 and 5.18. In Figure 5.15, the untransformed feature selection technique was applied and 8 classes of FLC-1 data were used. The result shows that the DTC has better performance at small feature subsets. There are fluctuations above 6 features due to variations of the sample mean and sample covariance which are random variables. In Figure 5.16, the untransformed feature selection technique was applied and 23 classes of FLC-1 data were used. Figure 5.16 shows that the DTC had best classification results at eight features.

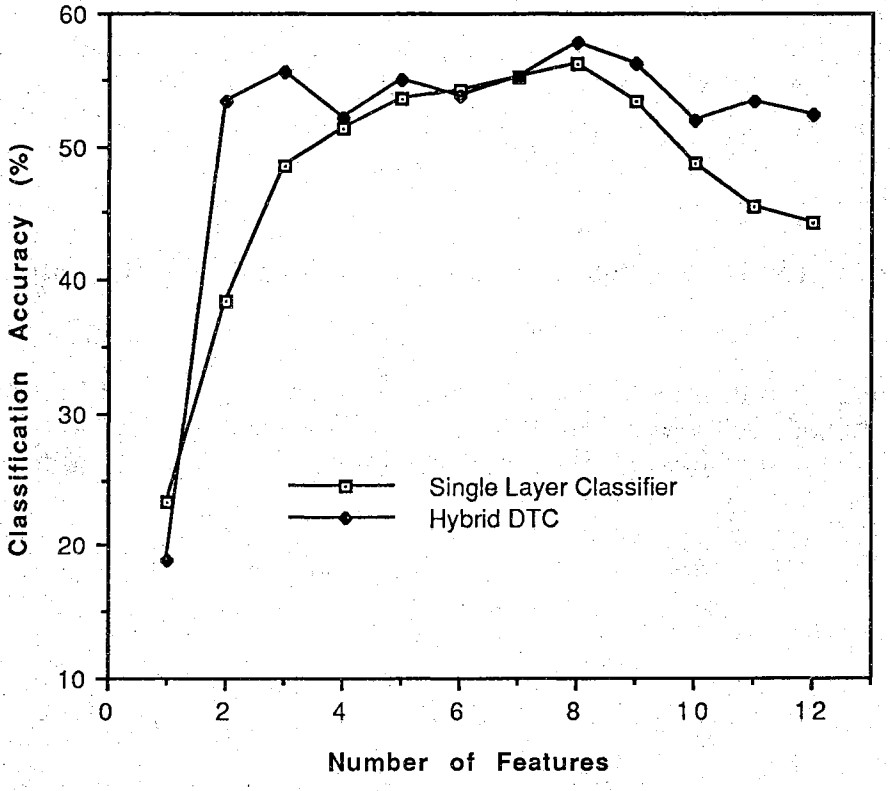


Figure 5.16 Untransformed Feature Selection Result (23 Class)

The transformed feature selection technique was applied to the 8 classes and 23 classes cases of FLC-1 data. Figures 5.17 and 5.18 show the results. These two figures show that the DTC had better performance than the single layer classifier.

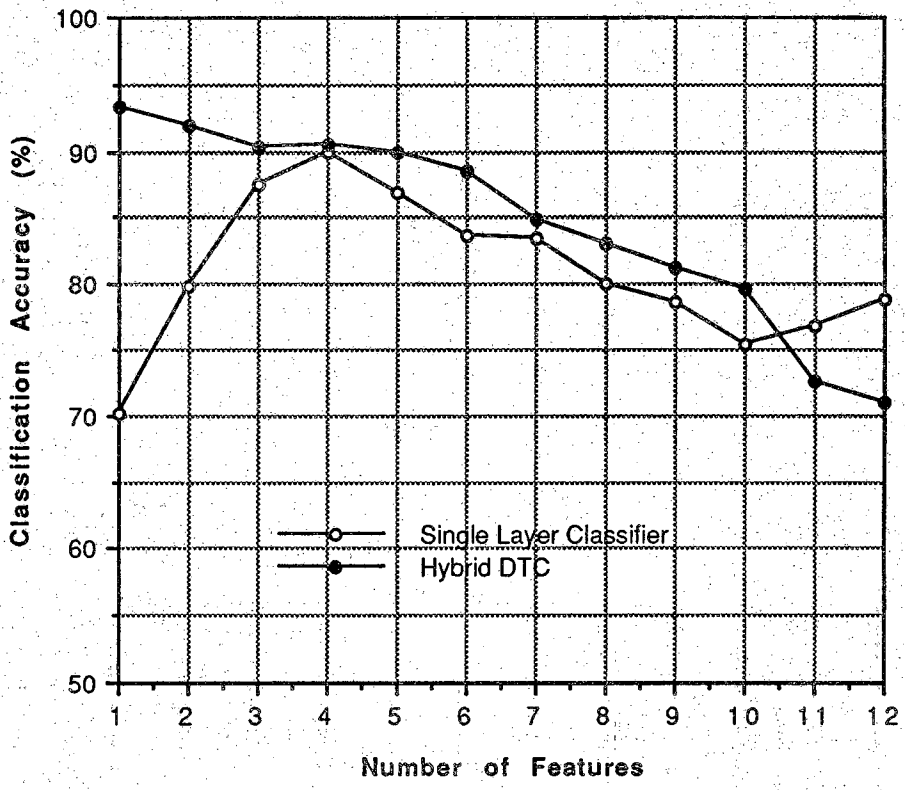


Figure 5.17 Transformed Feature Selection Result (8 Class)

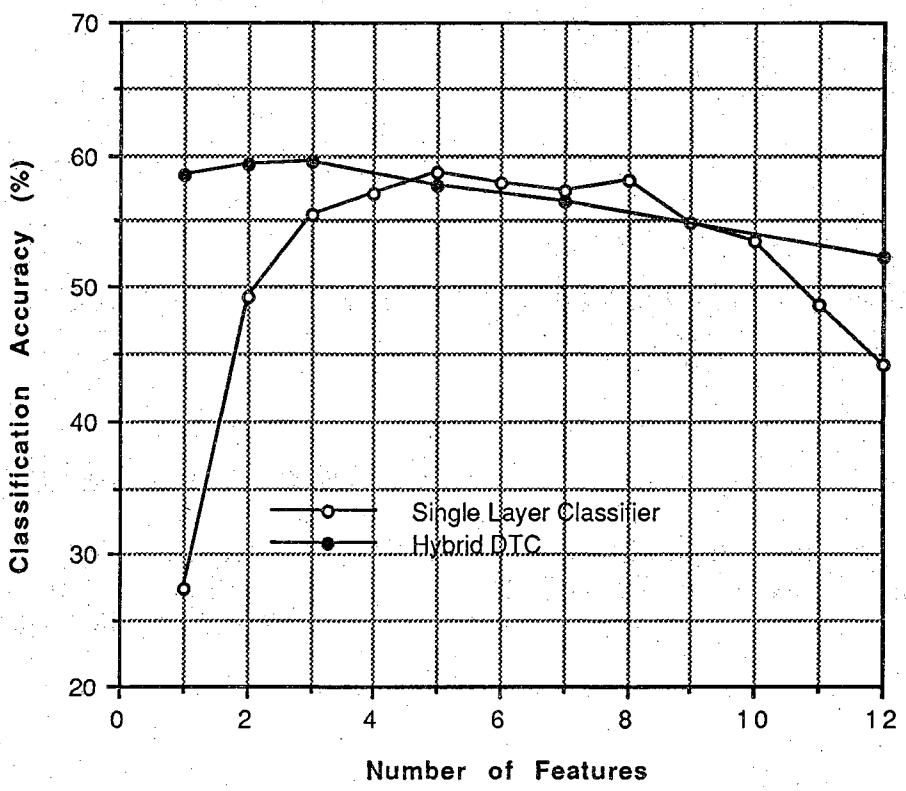


Figure 5.18 Transformed Feature Selection Result (23 Class)

5.6 DTC for Multisource Data

The DTC can be also applied to multisource, multitype data. In this section, a DTC application to such data will be described. The Anderson River data was the multisource, multitype data as described in section 5.1.

Table 5.21 Multisource Anderson River Data

Classes	#Train	#Test	Dim
Douglas-Fir(d)	30	1902	22
D-F + Lodgepole Pine(f)	30	5393	22
D-F + Cedar(i)	30	2865	22
Hemlock + Cedar(l)	30	3143	22
D-F + Other Species(q)	30	1279	22
Forest Clearings(t)	30	12594	22

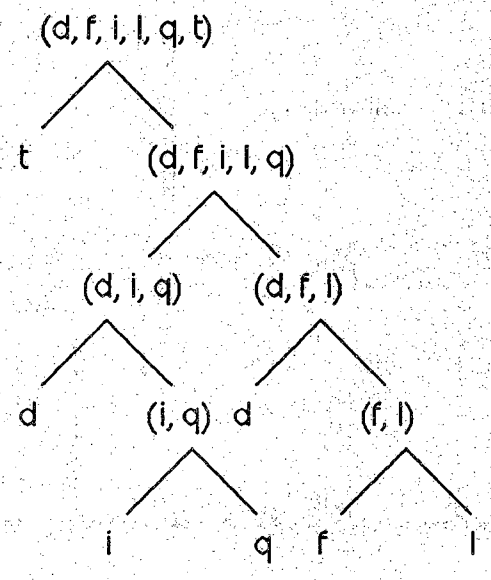


Figure 5.19 Hybrid DTC (Multisource)

Experiment 5.6.1

Six classes of Anderson River data were selected as shown in Table 5.21. In the first stage, two subgroups for each source, A/B MSS, Steep SAR, Shallow SAR, DEM, DSM, and DAM, were obtained by complete linkage. Two subgroups provided the initial information which determined two initial cluster centers. After applying the normalized clustering algorithm for the top down approach, the better source was selected by comparing the evaluation function which was defined in equation (2.5). Figure 5.19 shows the hybrid DTC resulting for multisource data. MSS was selected as the best source in the first stage. There is one node which has two subgroups. One subgroup consists of b, f, i, l,q. Another subgroup is t. In the second stage, DEM was the best source. In the third stage, MSS was the best

source for the left node and DEM was best for the right node. In the fourth stage, MSS was selected for left node and DSM for right node. The DTC reduces the error rate by 7.5 % over the single layer classifier when using A/B MSS, as tabulated in Table 5.22

Table 5.22 Multisource Data Result

Class	A/BMSS*	Steep SAR*	Shallow SAR*
Douglas-Fir	59.6	75.9	44.3
D-F + Logepole Pine	37.9	26.9	11.6
D-F + Cedar	43.1	18.9	20.1
Hemlock + Cedar	67.9	51.6	34.5
D-F + Other Species	0.1	74.6	15.3
Forest Clearings	5.1	67.7	24.1
Average	57.0	52.6	25.0

Class	DAM*	DEM*	DSM*	DTC
Douglas-Fir	12.3	44.4	0	54.9
D-F + Logepole Pine	12.0	43.7	25.8	45.7
D-F + Cedar	82.1	0	0	43.4
Hemlock + Cedar	0	85.4	91.7	93.6
D-F + Other Species	29.0	95.9	43.2	86.9
Forest Clearings	0.5	22.2	17.4	62.3
Average	22.6	48.6	29.7	64.5

* The single layer classifier is applied.

5.7 Strategy for Feature Selection

Previously in Chapter 3, we introduced extended canonical analysis and autocorrelation analysis as feature extraction methods and derived the risk function of the classification accuracy in Chapter 4. When the number of training samples are small, to minimize the risk function with a constraint to maximize the Bhattacharyya distance, the dimensionality must be reduced while maximizing the Bhattacharyya distance. Equation (4.48) showed that the best one feature in the transformed coordinate should give the best results in a situation which has only small numbers of training samples. We note that the above statements are analytically probable only for two classes. In the following experiments, we will demonstrate that the best one feature which has the maximum separability produces the best performance.

Experiment 5.7.1

FLC-1 data which was used in Experiment 5.2.1 was used again in this experiment. In transformed coordinates, the best single feature was extracted by canonical analysis, since the mean difference between classes was dominant. Next features in a transformed subspace were extracted by extended canonical analysis or autocorrelation analysis because the mean difference became smaller in a subspace. Therefore, added features are obtained from the extended canonical analysis and autocorrelation analysis. In this experiment, the best single feature produces the best result here as shown in Table 5.23 and 5.24.

Table 5.23 Extended Canonical Result (FLC-1)

Class	1*	2	3	4	5	6	7	12
Alfalfa	88.6+	76.2	73.0	75.4	68.4	63.4	54.2	9.0
Corn	97.0	96.5	96.3	96.8	98.8	98.2	98.5	81.8
Oats	92.0	96.0	96.2	95.4	93.2	89.6	88.0	81.7
Clover	88.5	91.6	92.8	90.0	89.9	89.8	74.2	37.1
Bean	85.9	83.1	72.1	74.7	73.8	74.7	71.6	70.9
Wheat	99.4	99.2	99.2	99.4	99.2	99.0	98.2	98.8
Soil	99.7	99.7	99.9	99.9	99.9	100	100	94.8
Rye	95.5	93.5	93.7	93.4	93.3	94.0	94.4	93.2
Avg	93.3	92.0	90.4	90.6	90.0	88.6	84.9	71.0

* the number of features which is used at each node.
+ correct classification accuracy(%)

Table 5.24 Canonical-Autocorrelation Result (FLC-1)

Class	1	2	3	4	5	6	7	12
Alfalfa	88.6	80.0	60.7	38.3	35.3	25.5	28.3	9.0
Corn	97.0	95.6	96.2	95.5	97.0	97.8	96.5	81.8
Oats	92.0	85.9	89.1	77.8	74.1	71.7	70.6	81.7
Clover	88.5	86.6	88.5	86.8	85.7	83.4	82.4	37.1
Bean	85.9	73.6	61.6	70.1	73.2	65.4	62.2	70.9
Wheat	99.4	99.8	99.4	99.6	99.6	99.6	99.6	98.8
Soil	99.7	100	100	99.9	99.9	99.9	99.9	94.8
Rye	95.5	96.9	99.4	98.6	98.7	96.9	98.4	93.2
Avg	93.3	89.8	86.8	83.3	82.9	80.0	79.7	71.0

Experiment 5.7.2

For a high dimensional data test, ten classes of FSS data, the same as in experiment 5.2.3, were selected again, and the same training and test sets were used. The best single feature was extracted by canonical analysis since the mean difference between classes was also dominant. Next features in a subspace were extracted by extended canonical analysis. The smallest error rate was obtained using the best single feature, as shown in Table 5.25.

Table 5.25 Extended Canonical Result (FSS)

Class	1*	2	3	4	5	10	20	30
a	63.5+	59.2	57.4	57.1	56.4	57.9	47.3	24.9
b	82.6	83.4	83.4	82.1	81.8	82.4	67.9	36.1
c	67.5	68.5	68.5	68.3	71.3	58.7	57.7	69.0
d	41.3	44.1	44.6	48.1	45.2	44.2	49.8	59.2
e	71.0	69.2	68.9	66.6	66.5	58.9	55.8	36.0
f	56.5	56.8	56.8	54.2	57.0	61.2	59.9	76.1
g	83.5	83.4	82.6	83.5	81.8	81.5	79.1	88.2
h	85.3	84.7	84.7	84.0	82.8	76.5	66.6	59.5
i	49.4	50.8	50.0	48.7	50.8	50.5	47.1	35.4
j	83.8	83.7	83.9	83.3	81.8	74.4	67.5	81.1
Avg	68.4	68.4	68.1	67.6	67.5	64.6	59.9	56.6

* the number of features which is used at each node.
+ correct classification accuracy(%)

As a result, the hybrid DTC has better classification accuracy than the maximum likelihood Gaussian classifier and the other DTC's when the best single feature is used at each node in the limited training sample situation.

CHAPTER 6. CONCLUSIONS

The fundamental objective of this research was to develop a design procedure for the DTC in a high dimensional data, large number of classes, limited training set size environment. We have defined the following three methods: top down design, bottom up design, and hybrid design. These methods are more simple and effective than previous design methods. Three kinds of bottom up design methods were described: single linkage, complete linkage, and dynamic linkage. In cases where a bottom up approach is called for, any of the three might be useful, with a perhaps slight preference for the dynamic approach. Although all three approaches were studied, the hybrid classifier was shown by empirical test to have the best performance; this was expected because it reconciles a property of data (classes being separable) and a property of the application (classes informational value).

The mathematical relationship between sample size, dimensionality, and risk value was derived. The incremental error was shown to be simultaneously affected by two factors, dimensionality and separability. For predicting the optimal number of features, we conclude that the optimal number of features in transformed coordinates is just one when only small numbers of samples are available. We have demonstrated that the best result is obtained when we use just one feature experimentally. Empirically, it was shown that a reasonable sample size is six to ten times the dimensionality if the dimensionality and separability simultaneously increase.

In the case of more than two classes, the ambiguity with respect to the optimal number of features is wider and unpredictable. A binary hierarchical classifier may solve the above drawbacks. Because only two groups are classified at each node in a binary hierarchical classifier, a minimum error rate can be obtained when the best single feature is used in transformed coordinates. Therefore, an overall minimum error rate of a hierarchical classifier is obtained by minimizing the error rate at each node. We are able to predict the optimum number of features and obtain the minimum error rate when we use a binary hierarchical classifier.

LIST OF REFERENCES

[1] C. Wu, D. Landgrebe, and P. H. Swain, "The decision tree approach to classification," Tech. Rep. TR-EE 75-17, Purdue Univ., 1975.

[2] M. J. Muasher and D. A. Landgrebe, "The K-L expansion as an effective feature ordering technique for limited training sample size," IEEE Trans. Geosci. Remote Sen., vol. GE-21, pp. 438-441, 1983.

[3] H. M. Kalayeh, M. J. Muasher, and D. A. Landgrebe, "Feature selection with limited training samples," IEEE Trans. Geosci. Remote Sen., vol. GE-21, pp. 434-438, 1983.

[4] K. Fukunaga and W. L. Koontz, "Application of the Karhunen-Loeve transform to feature selection and ordering," IEEE Trans. Comput., vol. C-19, pp. 311- 318, 1970.

[5] K. Fukunaga and T. F. Krile, "Calculation of Bayes' recognition error for two multivariate gaussian distributions," IEEE Trans. Comput., vol. C-18, pp. 220-229, 1969.

[6] K. Fukunaga, *Introduction to statistical pattern recognition.* New York: Academic, 1972.

[7] K. Fukunaga and T. E. Flick, "Classification error for a very large number of classes," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-6, pp. 779-788, 1984.

[8] P. H. Swain and H. Hauska, "The decision tree classifier : Design and potential," IEEE Trans. Geoscience Electron., vol. GE-15 pp. 142-147, 1977.

[9] P. H. Swain and S. M. Davis, *Remote Sensing : The Quantitative Approach.* McGraw-Hill, 1978.

[10] J. W. Sammon, "An optimal discriminant plane," IEEE Trans. Comput., vol. C-19, pp. 826-829, 1970.

[11] D. H. Foley, "Consideration of sample and feature size," IEEE Trans. Inform. Theory, vol. IT-18, pp. 618-626, 1972.

[12] D. H. Foley and J. W. Sammon, "An optimal set of discriminant vectors,' IEEE Trans. Comput., vol. C-24, pp. 281-289, 1975.

[13] K. C. You and K. S. Fu, "An approach to the design of a linear binary tree classifier," Proc. Symp. Machine Processing of Remotely Sensed Data, Purdue Univ., 1976, pp. 3a-1 to 3a-10.

[14] J. K. Mui and K. S. Fu, "Automated classification of nucleated blood cells using a binary tree classifier," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-2, pp. 429-443, 1980.

[15] Q. Shi and K. S. Fu, "A method for the design of binary tree classifiers," Pattern Recognition vol. 16, pp. 593-603, 1983.

[16] C. Y. Suen and Q. R. Wang, "ISOETRP-An interactive clustering algorithm with new objectives," Pattern Recognition, vol. 17, pp. 211-219, 1984.

[17] Q. R. Wang and C. Y. Suen, "Analysis and design of decision tree based on entropy reduction and its application to large character set recognition," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-6, pp. 406-417, 1984.

[18] Q. R. Wang and C. Y. Suen, "Large tree classifier with heuristic search and global training," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-9, pp. 91-102, 1987.

[19] A. V. Kulkarni and L. N. Kanal, "An optimization approach to hierarchical classifier design, "Proc. Third Int. Joint Conf. Pattern Recognition, San Diego, Calif., pp. 459-466, 1976.

[20] A. V. Kulkarni, "On the mean accuracy of hierarchical classifiers," IEEE Trans. Comput., vol. C-27, pp. 771-776, 1978.

[21] L. N. Kanal, "Problem-solving models and search strategies for pattern recognition," IEEE Trans. Pattern Anal. and Machine Intell., vol. PAMI-1, pp. 193-201, 1979.

[22] G. F. Hughes, "On the mean accuracy of statistical pattern recognizers," IEEE Trans. Inform. Theory, vol. IT-14, pp. 55-63, 1968.

[23] W. S. Meisel and D. A. Michalopoulos, "A partitioning algorithm with applications in pattern recognition and optimization of decision trees," IEEE Trans. Comput., vol. C-22, pp. 93-103, Jan. 1973.

[24] R. Dubes and A. K. Jain, "Clustering techniques : The user's dilemma," Pattern Recognition, vol. 8, pp. 247-260, 1976.

[25] R. L. P. Chang and T. Pavlidis, "Fuzzy decision tree algorithms,"IEEE Trans. Systems, Mans, Cybern., vol. SMC-7, pp. 28-35, 1977.

[26] P. Argentiero, R. Chin, and P. Beaudet, "An automated approach to the design of decision tree classifiers," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-4, pp. 51-57, 1982.

[27] I. K. Sethi and G. Savarayudu, "Hierarchical classifier design using mutual information," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-4, pp. 441-445, 1982.

[28] M. W. Kurzynski, "The optimal strategy of a tree classifier,"Pattern Recognition, vol. 16, pp. 81-87, 1983.

[29] G. H. Landeweerd, T. Timmers, and E. S. Gelsema, "Binary tree versus single level tree classification of white blood cells,"Pattern Recognition, vol. 16, pp. 571-577, 1983.

[30] R. G. Casey and G. Nagy, "Decison tree design using a probabilistic model," IEEE Trans. Inform. Theory, vol. IT-30, pp. 93-99, 1984.

[31] J. Schuermann and W. Doster, "A decision theoretic approach to hierarchical classifier design," Pattern Recognition, vol. 17, pp. 350-369, 1984.

[32] I. D. Longstaff, "On extensions to Fisher's linear discriminant function," IEEE Trans. Pattern Anal. Machine Intell., vol PAMI-9, pp. 321- 325, 1987.

[33] A. Henaut and M. Delorme, "Distance matrix comparison and tree construction," Pattern Recognition Letters, vol. 7, pp. 207-213, 1988.

[34] S. Wilks, *Mathematical statistics*. John Wiley, 1962.

[35] N. J. Nilsson, *Problem-solving methods in artifitial intelligence*. McGraw Hill, 1971.

[36] E. A. Patrick, *Fundamentals of pattern recognition*. Prentice-Hall, 1972.

[37] R.O. Duda and P. E. Hart, *Pattern classification and scene analysis*. John Wiley, 1973.

[38] J. A. Bondy and U. S. R. Murty, *Graph theory with applications*. North Holland, 1976.

[39] Aho, Hopcropt, and Ullman, *The design and analysis of computational algorithms*. Addison-Weseley, 1974.

[40] B. Noble and J. W. Daniel, *Applied linear algebra*. Prentice-Hall, 1977.

[41] B. Everitt, *Cluster Anaysis*. Halsted Press, 1980.

[42] N. J. Nilsson, *Principles of artificial intelligence*. Springer-Verlag, 1982.

[43] Breiman, Friedman, and Olshen, *Classification and RegressionTrees*. Wardsworth international group, 1984.

[44] G. A. F. Seber, *Multivariate Observations*. John Wiley & Sons, 1984.

[45] L. Lebart, A. Morinau, and K. M. Warwick, *Multivariate Descriptive Statistical Analysis*. John Wiley & Sons, 1984.

[46] J. A. Richards, *Remote sensing digital image analysis*. Springer-Verlag, 1986.

[47] J. MacQueen, "Some methods for classification and analysis of multivariate observations," Proc. 5th Berkeley Sympo., 1, pp. 281-297, 1967.

[48] E. M. L. Beale, *Cluster analysis*. Scientific Control Systems, 1969.

[49] G. H. Ball and D. J. Hall, "A clustering technique for summarizing multivariate data," Behavior Sci., vol.12, pp. 153-155, 1967.

[50] S. J. Whitsitt and D. A. Landgrebe, Error estimation and separability measure in feature selection for multiclass pattern recognition, School of EE, Purdue University, TR-EE 77-34, 1977.

[51] C. F. Hutchinson, "Techniques for combining Landsat and ancillary data for digital classification improvement," Photogrammetric Engineering and Remote Sensing, vol.48, no.1, pp. 123-130, 1982.

[52] R. M. Hoffer and staff, "Computer-aided analysis of Skylab multispectral scanner data in mountaineous terrain for land use, forestry, water resources and geological applications," LARS Information Note 121275, Laboratory for Applications of Remote Sensing, Purdue University, W. Lafayette, IN 47907, 1975.

[53] P. H. Swain, J. A. Richards and T. Lee, "Multisource data analysis in remote sensing and geographic information processing," Proceedings of the 11th International Symposium on Machine Processing of Remotely Sensed Data 1985, W. Lafayette, IN., pp. 211-217, June 1985.

[54] T. L. Phillips (Ed.), "LARSYS Version 3 Users Manual," Laboratory for Applications of Remote Sensing, Purdue University, W. Lafayette, 1973.

[55] B. Chandrasekaran, "Independence of measurements and the mean recognition accuracy," IEEE Trans. Inform. Theory, vol. IT-17, pp. 452-456,1971.

[56] W. Waller and A. K. Jain, "Mean recognition accuracy of dependent binary measurements," Proc. Seventh Internat. Conf. Cybernet. Soc., pp. 586-590, 1977.

[57] A. K. Jain and B. Chandrasekaran, "Dimensionality and sample size considerations in pattern recognition practice," Handbook of Statistics, vol. 2, pp. 835-855, 1982.

[58] A. K. Jain, "On an estimate of the Bhattacharyya distance," IEEE Trans. Systems, Man, and Cyber. vol. SMC-6, pp. 763-766, 1976.

[59] Bickel and Doksum, *Mathematical Statistics*. Holden-Day, 1977.

[60] K. Fukunaga and R. A. Hayes, "Effect of Sample Size in Classifier Design ," IEEE Trans. Pattern Anal. Machine Intell., vol PAMI-11, pp. 873-885, 1989.

Appendix A Sum of Squared Error Clustering Program

```
#include <stdio.h>
#include <math.h>
/*
 * Cluster -- Multispectral Image Clustering Program
 *   Euclidean distance measure is used.
 *   Specify NP,MXCH,num_chan (MXCH = num_chan).
 *   Use for BIL format.
 */
#define NL                      180     /* # of lines */
#define NP                      1       /* # of pixels per line */
#define MXCH            100     /* maximum number of channels */
#define MAXCENTERS      2       /* maximum number of cluster centers */
#define ELF     else if
#define  c2f(n)   (float)((n)&0377)

struct center {
        float origin[MXCH];             /* current cluster origin */
        float mean[MXCH];               /* accumulated mean */
        float sumsq[MXCH];      /* sum of squares  */
        float count;                    /* number points currently in this cluster */
};
struct center centers[MAXCENTERS];

int first_pass;                 /* flag to indicate first pass */
int tf1    = 0;                 /* trace cluster main loop flag */
int tf2    = 0;                 /* trace clasify */
int tf3    = 0;                 /* trace I/O */
int vflag = 0;                  /* verbose flag */
int num_centers;                /* number of cluster centers this run */
int num_chan=100;       /* number of channels this run */
long int num_diff;              /* number of changes this pass */
int min_changes;                /* minimum number of changes for completion */
int num_passes;         /* number of passes until clusters converge */
int clneed = 0;         /* number of clusters to be created */
int     input = 0;              /* input file descriptor */
int     output = 0;             /* output file descriptor */
int     sfd = 0;        /* stat file fd */
FILE *fopen(),*fp;              /* initialize fp=0, cluster center file */

long int t0;                    /* starting time */
long int t1;                    /* time of start of current iteration */
long int t2;                    /* time of end of current iteration */
/*
 * main -- supervisory control program
 *
 */

main(argc, argp)
char **argp;
{
    min_changes = 0 ;
        parse(argc, argp);
        if(!fp)
                clinit(clneed);
        time(&t0);
```

```
        t1 = t0;
        clump();
        print_stats();
        exit(0);
}

/*
 * clump -- main clustering loop
 */

clump()
{
        register i;
        register float *fp1, *fp2;
        float *getpt(), n;
        int j;

        first_pass = 1;
        num_passes = 0;
        /*
    min_changes *= NL*NP/100;
        */
    num_diff = NL*NP;
        do {
                if(tf1)
                        print_stats();
        min_changes = num_diff;
                num_diff = 0;
                rewind_files();
                for(i=0; i<num_centers; i++){
                        centers[i].count = 0.0;
                        fp1 = centers[i].mean;
                        fp2 = centers[i].sumsq;
                        for(j=0; j<num_chan; j++)
                                *fp1++ = *fp2++ = 0.0;
                }
                while((fp1 = getpt(input)) != 0)
                        clasify(fp1);
                /* compute new cluster centers:  */
                for(i=0; i<num_centers; i++){
                        fp1 = centers[i].mean;
                        for(j=0; j<num_chan; j++)
                                if(centers[i].count != 0)
                                        *fp1++ /= centers[i].count;
                                else
                                        *fp1++ = *fp2++  = 0.0;
                        fp1 = centers[i].origin;
                        fp2 = centers[i].mean;
                        for(j=0; j<num_chan; j++)
                                *fp1++ = *fp2++;
                        fp1 = centers[i].sumsq;
                        fp2 = centers[i].mean;
                        n = centers[i].count;
                        for(j=0; j<num_chan; j++){
                                if(n > 1)
                                        *fp1 = (*fp1/(n-1)) - (n/(n-1)) * (*fp2 * *fp2);
                                else
                                        *fp1 = 0.01;
                                fp1++;
                                fp2++;
                        }
                }
                first_pass = 0;
```

```
                num_passes++;
                oflush();
        } while(num_diff < min_changes);
        /*
        } while(num_diff > min_changes);
    */
        if(vflag)
                fprintf(stderr, "clustering complete\n");
}
/*
 * clasify -- clasify a point and update cluster
 *              center information
 */

clasify(pt)
float *pt;
{
        register float *fp1, *fp2;
        register int i;
        float dist1, dist2, dist();
        int index;

        index = 0;
        fp2 = pt;
        dist2 = dist(centers[0].origin, fp2);
        for(i=1; i<num_centers; i++){
                dist1 = dist(centers[i].origin, fp2);
                if(dist1 < dist2) {
                        dist2 = dist1;
                        index = i;
                }
        }
        centers[index].count = centers[index].count + 1.0;
        /*
         * add this point to current mean
         */
        fp1 = centers[index].mean;
        for(i=0; i<num_chan; i++)
                *fp1++ += *fp2++;
        /*
         *  accumulate sum of squares:
         */
        fp1 = centers[index].sumsq;
        fp2 = pt;
        for(i=0; i<num_chan; i++){
                *fp1++ += *fp2 * *fp2;
                fp2++;
        }
        if(tf2){
                printf("\nCenter %5d (%6.1f):\t\t", index+1, centers[index].count);
                printl("%6.1f\t", num_chan, pt);
                printf("\nnew mean:\t\t\t");
                printl("%8.2f\t", num_chan, centers[index].mean);
                printf("\n");
        }
        putpt(index);
}
/*
 * getpt -- get a multispectral input point, and convert
 *              it to a floating point vector
 */

char    ibuf[MXCH][NP];
```

```
char    *ibp[MXCH] = ibuf[MXCH];
int     nib = 0;

float *getpt(fd)
{
        char buf[MXCH];
        static float fbuf[MXCH];
        register char *p;
        register float *fp;
        register int i;

        fp = fbuf;
        if(nib <= 0){
        for(i=0; i<num_chan; i++){
                        if((nib = read(fd, ibuf[i], NP)) <= 0){
                                if(tf3)
                                        printf("EOF on input\n");
                                return(0);
                        }
                        ibp[i] = ibuf[i];
                }
        }
        for(i=0; i<num_chan; i++){
                *fp++ = c2f(*ibp[i]++);
        }
        nib--;
        if(tf3){
                printf("\nINPUT:\t\t");
                printl("%8.0f", num_chan, fbuf);
                printf("\n");
        }
        return(fbuf);
}

/*
 * putpt -- output a clasified point
 */

char    obuf[NP];
char    *obp = obuf;
int     nob = 0;

putpt(n)
{
        if(first_pass || *obp != n){
                num_diff++;
        }
        if(tf3)
                printf("OUTPUT:\t%5d (%d)\n", n, *obp);
        *obp++ = n;
        if(++nob >= NP){
                write(output, obuf, nob);
                if(!first_pass){
                        nob = read(output, obuf, NP);
                        lseek(output, (long)(-nob), 1);
                }
                nob = 0;
                obp = obuf;
        }
}

oflush()
{
```

```
        if(nob)
                write(output, obuf, nob);
}

/*
 * dist -- compute distance between 2 points
 */

float dist(fp1, fp2)
char *fp1, *fp2;
{
        register float *p1, *p2;
        register int i;
        float sum, term;

        p1 = (float*)fp1;
        p2 = (float*)fp2;
        sum = 0;
        for(i=0; i<num_chan; i++){
                term = fabs(*p1++ - *p2++);
                term *= term;
                sum += term;
        }
        return(sum);
}

clinit(nc)
{
        register i;
        register float *fp1, *fp2;
        float  *pt, n;
        int j;

        /*
         * clear everything:
         */
        rewind_files();
        n = 0.0;
        fp1 = centers[0].mean;
        fp2 = centers[0].sumsq;
        for(j=0; j<num_chan; j++)
                *fp1++ = *fp2++ = 0.0;
        /*
         * now, accumulate mean & st. dev. for entire pix:
         */
        while((pt = getpt(input)) != 0){
                n += 1.0;
                /*
                 * add this point to current mean
                 */
                fp1 = centers[0].mean;
                fp2 = pt;
                for(i=0; i<num_chan; i++)
                        *fp1++ += *fp2++;
                /*
                 *  accumulate sum of squares:
                 */
                fp1 = centers[0].sumsq;
                fp2 = pt;
                for(i=0; i<num_chan; i++){
                        *fp1++ += *fp2 * *fp2;
                        fp2++;
                }
```

```
        }
        if(n < 2.0){
                printf("input file is empty\n");
                exit(4);
        }
        /*
         * now convert sum and sumsq to mean and st. dev.:
         */
        fp1 = centers[0].mean;
        for(i=0; i<num_chan; i++)
                *fp1++ /= n;
        fp1 = centers[0].sumsq;
        fp2 = centers[0].mean;
        for(j=0; j<num_chan; j++){
                *fp1 = sqrt((*fp1/(n-1)) - (n/(n-1)) * (*fp2 * *fp2));
                fp1++;
                fp2++;
        }
        /*
         * now compute each of the new cluster centers:
         */
        fp1 = centers[0].mean;
        fp2 = centers[0].sumsq;
        for(j=0; j<nc; j++){
                for(i=0; i<num_chan; i++){
                        centers[j].origin[i] = fp1[i] + fp2[i] *
                                ((j-(nc/2.0))/(2*(nc-1)));
                }
        }
        if(vflag)
                fprintf(stderr,"cluster initialization complete\n");
}

/*
 * parse -- parse input arguments
 */

int errors;

parse(argc, argp)
char **argp;
{

        errors = 0;
        if(argc == 1){
                help();
                exit();
        }
        while(--argc > 0){
                argp++;
                if(!strncmp(*argp, "if=",3))
                        input = open(*argp+3, 0);
                ELF (!strncmp(*argp, "of=",3)){
                        output = creat(*argp+3, 0644);
                        close(output);
                        output = open(*argp+3, 2);
                } ELF (!strncmp(*argp, "cf=",3)){
                        fp = fopen(*argp+3, "r");
                } ELF (!strncmp(*argp, "sf=",3)){
                        sfd = open(*argp+3, 0);
                } ELF (!strncmp(*argp, "nc=",3)){
                        num_centers = atoi(*argp + 3);
                        clneed = num_centers;
```

```
        } ELF (!strncmp(*argp, "pc=",3)){
            min_changes = atoi(*argp + 3);
                } ELF (!strncmp(*argp, "-v",2))
                        vflag++;
                ELF (!strncmp(*argp, "-t1",3))
                        tf1++;
                ELF (!strncmp(*argp, "-t2",3))
                        tf2++;
                ELF (!strncmp(*argp, "-t3",3))
                        tf3++;
                else {
                        errors++;
                        printf("bad option: %s\n", *argp);
                }
        }
        need(input, "input", "if");
        need(output, "output", "of");
        if(!clneed){
                printf("\"nc=...\"   (number of centers to create)\n");
                errors++;
        }
        if(errors)
                exit(1);
        setup_files();
}

help()
{
        printf("Summary of cluster parameters:\n\n");
        printf("if=...        input data file\n");
        printf("of=...        output results file\n");
        printf("cf=...        initial cluster center file\n");
        printf("sf=...        output statistics file\n");
        printf("pc=..  percent change desired(0-100) \n") ;
        printf("nc=nnn     number of centers to create\n");
        printf("option -t1 -t2 -t3 -v\n");
}
/*
 * need -- see if a needed file is present
 */
need(fd, fn, fi)
char *fn, *fi;
{
        if(fd == 0){
                printf("required %s file (%s=...) missing\n", fn , fi);
                errors++;
        }
}

/*
 *  setup_files -- read in once-only files & get ready to cluster
 */

setup_files()
{
        int i,j;

        /* output file */
    if(fp){
          for(i=0; i<num_centers; i++)
       for(j=0; j<num_chan; j++)
            fscanf(fp, "%f", &centers[i].origin[j]);
    }
```

```
}

/*
 * rewind_files -- rewind I/O files, and prepare for another pass
 */

rewind_files()
{
        lseek(input, 0L, 0);
        lseek(output, 0L, 0);
        if(!first_pass){
                nob = read(output, obuf, NP);
                lseek(output, 0L, 0);
                nob = 0;
                obp = obuf;
        }
}
/*
 * print_stats -- output summary of clustering run
 */

print_stats()
{
        register i;
        register float *fp1, *fp2;
        int j;

        time(&t2);
        printf("\tNumber of passes = %d", num_passes);
        printf("\n\tNumber of changes = %d\n\telapsed time = %ld / ",
                        num_diff, t2-t1);
        printf("%ld\n", t2-t0);
        for(i=0; i<num_centers; i++){
                printf("\n\n\tCenter %d, (%5.1f points):\n", i+1, centers[i].count);
                printf("\tOrigin:\t\t");
                printl("%6.2f\t", num_chan, centers[i].origin);
                printf("\n\tVariance:       ");
                printl("%6.2f\t", num_chan, centers[i].sumsq);
                printf("\n");
        }
        t1 = t2;
}

printl(f, n, afp)
char *f;
int n;
float *afp;
{
        register int i;
        register float *fp;

        fp = afp;
        for(i=0; i<n; i++)
                printf(f, *fp++);
}
```

Appendix B Normalized SSE Clustering Program

```
#include <stdio.h>
#include <math.h>
/*
 * Cluster -- Multispectral Image Clustering Program
 *   Normalized Euclidean Distance Measure is used.
 *   Specify NP,MXCH,num_chan.(NL is for min_chan)
 *   Use for BIL format.
 */

#define NP                      1       /* # of pixels per line */
#define MXCH            11      /* maximum number of channels */
#define MAXCENTERS      10      /* maximum number of cluster centers */

#define ELF     else if
#define  c2f(n)   (float)((n)&0377)

struct center {
        float origin[MXCH];             /* current cluster origin */
        float mean[MXCH];               /* accumulated mean */
        float sumsq[MXCH];      /* sum of squares  */
        float det;
        float cov[MXCH][MXCH];
        float inv[MXCH][MXCH];
        float count;                    /* number points currently in this cluster */
};
struct center centers[MAXCENTERS];


int first_pass;             /* flag to indicate first pass */
int tf1    = 0;             /* trace cluster main loop flag */
int tf2    = 0;             /* trace clasify */
int tf3    = 0;             /* trace I/O */
int vflag = 0;              /* verbose flag */

int num_centers;                    /* number of cluster centers this run */
int num_chan=11;                    /* number of channels this run */
long int num_diff;                  /* number of changes this pass */
int min_changes;                    /* minimum number of changes for completion */
int num_passes;             /* number of passes until clusters converge */
int clneed = 0;             /* number of clusters to be created */

int     input = 0;                  /* input file descriptor */
int     output = 0;                 /* output file descriptor */
int     sfd = 0;            /* stat file fd */
FILE *fopen(),*mea,*co;   /* initialize fp=0, cluster center file */
float d1,d2,fac[MXCH][MXCH];
long int t0;                        /* starting time */
long int t1;                        /* time of start of current iteration */
long int t2;                        /* time of end of current iteration */
/*
 * main -- supervisory control program
 *
 */

main(argc, argp)
```

```
char **argp;
{
      min_changes = 0 ;
            parse(argc, argp);
            if(!mea)
                  clinit(clneed);
            time(&t0);
            t1 = t0;
            clump();
            print_stats();
            exit(0);
}

/*
 * clump -- main clustering loop
 */

clump()
{
 register i;
 register float *fp1, *fp2,*fp3;
 float *getpt(), n;
 int j,k;

 first_pass = 1;
 num_passes = 0;
 num_diff = 1000*1000;
 do {
  if(tf1)
   print_stats();
  min_changes=num_diff;
  num_diff = 0;
  rewind_files();
  for(i=0; i<num_centers; i++){
   centers[i].count = 0.0;
   fp1 = centers[i].mean;
   fp2 = centers[i].sumsq;
   for(j=0; j<num_chan; j++){
    *fp1++ = *fp2++ = 0.0;
    for(k=0; k<num_chan; k++)
      centers[i].cov[j][k] = 0.0;
   }}
   while((fp1 = getpt(input)) != 0)
    clasify(fp1);

   /* compute new cluster centers:  */
   for(i=0; i<num_centers; i++){
    fp1 = centers[i].mean;
    for(j=0; j<num_chan; j++){
     if(centers[i].count != 0)
      *fp1++ /= centers[i].count;
     else
      *fp1++ = 0.0;
    }
    fp1 = centers[i].origin;
    fp2 = centers[i].mean;
    for(j=0; j<num_chan; j++)
      *fp1++ = *fp2++;
    n = centers[i].count;
    for(j=0; j<num_chan; j++){
     if(n > 1)
      centers[i].sumsq[j] = centers[i].sumsq[j]/(n-1) -
            (n/(n-1))*centers[i].mean[j]*centers[i].mean[j];
```

```
    else
     centers[i].sumsq[j] = 0.01;
    for(k=0; k<num_chan; k++){
    if(n > 1)
       centers[i].cov[j][k]=centers[i].cov[j][k]/(n-1) - (n/(n-1)) *
              centers[i].mean[j]*centers[i].mean[k];
    else
      centers[i].cov[j][k]= 0.01;
   }}
    lftds_(&num_chan,centers[i].cov,&num_chan,fac,&num_chan);
   lfdds_(&num_chan,fac,&num_chan,&d1,&d2);
   centers[i].det=d1*pow(10.0,d2);;
    linds_(&num_chan,centers[i].cov ,&num_chan,centers[i].inv,
                        &num_chan);
  }
  first_pass = 0;
  num_passes++;
  oflush();
/*
        } while(num_diff > min_changes);
*/
        } while(num_diff < min_changes);

        if(vflag)
                fprintf(stderr, "clustering complete\n");
}
/*
 * clasify -- clasify a point and update cluster
 *                center information
 *
 */

clasify(pt)
float *pt;
{
        register float *fp1, *fp2,*fp3;
        register int i,j,k;
        float dist1, dist2, dist();
        int index;

        index = 0;
        fp2 = pt;
        dist2 = dist(centers[0].origin,centers[0].inv,
                              centers[0].det,fp2);
        for(i=1; i<num_centers; i++){
                dist1 = dist(centers[i].origin,centers[i].inv,
                                        centers[i].det,fp2);
                if(dist1 < dist2) {
                        dist2 = dist1;
                        index = i;
                }
        }
        centers[index].count = centers[index].count + 1.0;
        /*
         * add this point to current mean
         */
        fp1 = centers[index].mean;
        for(i=0; i<num_chan; i++)
                *fp1++ += *fp2++;
        /*
         *  accumulate sum of squares:
         */
        fp1 = centers[index].sumsq;
```

```
        fp2 = pt;
        for(i=0; i<num_chan; i++){
          *fp1++ += *(fp2+i) * *(fp2+i);
          for(j=0; j<num_chan; j++){
            centers[index].cov[i][j] += *(fp2+i) * *(fp2+j);
        }}
        if(tf2){
                printf("\nCenter %5d (%6.1f):\t\t", index+1, centers[index].count);
                printl("%6.1f\t", num_chan, pt);
                printf("\nnew mean:\t\t\t");
                printl("%8.2f\t", num_chan, centers[index].mean);
                printf("\n");
        }
        putpt(index);
}
/*
 * getpt -- get a multispectral input point, and convert
 *                  it to a floating point vector
 *
 */

char    ibuf[MXCH][NP];
char    *ibp[MXCH] = ibuf[MXCH];
int     nib = 0;

float *getpt(fd)
{
        char buf[MXCH];
        static float fbuf[MXCH];
        register char *p;
        register float *fp;
        register int i;

        fp = fbuf;
        if(nib <= 0){
        for(i=0; i<num_chan; i++){
                        if((nib = read(fd, ibuf[i], NP)) <= 0){
                                if(tf3)
                                        printf("EOF on input\n");
                                return(0);
                        }
                        ibp[i] = ibuf[i];
                }
        }
        for(i=0; i<num_chan; i++){
                *fp++ = c2f(*ibp[i]++);
        }
        nib--;
        if(tf3){
                printf("\nINPUT:\t\t");
                printl("%8.0f", num_chan, fbuf);
                printf("\n");
        }
        return(fbuf);
}

/*
 * putpt -- output a clasified point
 *
 */

char    obuf[NP];
char    *obp = obuf;
```

```
int         nob = 0;

putpt(n)
{
        if(first_pass || *obp != n){
                num_diff++;
        }
        if(tf3)
                printf("OUTPUT:\t%5d (%d)\n", n, *obp);
        *obp++ = n;
        if(++nob >= NP){
                write(output, obuf, nob);
                if(!first_pass){
                        nob = read(output, obuf, NP);
                        lseek(output, (long)(-nob), 1);
                }
                nob = 0;
                obp = obuf;
        }
}

oflush()
{
        if(nob)
                write(output, obuf, nob);
}

/*
 * dist -- compute distance between 2 points
 *
 *
 */

float dist(fp1,fp2,fp3,fp4)
float *fp1,fp2[MXCH][MXCH],fp3,*fp4;
{
        register float *p1,*p4;
        register int i,j,k;
        float sum,term1[MXCH],term2[MXCH];

        p1 = fp1;
        p4 = fp4;
        sum = 0;
        for(i=0; i<num_chan; i++){
                term1[i] = fabs(*(p1+i) - *(p4+i));
        }
        for(i=0; i<num_chan; i++){
          term2[i] = 0.0;
          for(j=0; j<num_chan; j++){
      term2[i] += term1[j] * fp2[i][j];
    }
     sum += term1[i]*term2[i];
        }
        sum += log(fp3);
        return(sum);
}


clinit(nc)
{
        register i;
        register float *fp1, *fp2;
        float  *pt, n;
```

```
int j;

/*
 * clear everything:
 */
rewind_files();
n = 0.0;
fp1 = centers[0].mean;
fp2 = centers[0].sumsq;
for(j=0; j<num_chan; j++)
        *fp1++ = *fp2++ = 0.0;
/*
 * now, accumulate mean & st. dev. for entire pix:
 */
while((pt = getpt(input)) != 0){
        n += 1.0;
        /*
         * add this point to current mean
         */
        fp1 = centers[0].mean;
        fp2 = pt;
        for(i=0; i<num_chan; i++)
                *fp1++ += *fp2++;
        /*
         *  accumulate sum of squares:
         */
        fp1 = centers[0].sumsq;
        fp2 = pt;
        for(i=0; i<num_chan; i++){
                *fp1++ += *fp2 * *fp2;
                fp2++;
        }
}
if(n < 2.0){
        printf("input file is empty\n");
        exit(4);
}
/*
 * now convert sum and sumsq to mean and st. dev.:
 */
fp1 = centers[0].mean;
for(i=0; i<num_chan; i++)
        *fp1++ /= n;
fp1 = centers[0].sumsq;
fp2 = centers[0].mean;
for(j=0; j<num_chan; j++){
        *fp1 = sqrt((*fp1/(n-1)) - (n/(n-1)) * (*fp2 * *fp2));
        fp1++;
        fp2++;
}
/*
 * now compute each of the new cluster centers:
 */
fp1 = centers[0].mean;
fp2 = centers[0].sumsq;
for(j=0; j<nc; j++){
        for(i=0; i<num_chan; i++){
                centers[j].origin[i] = fp1[i] + fp2[i] *
                        ((j-(nc/2.0))/(2*(nc-1)));
        }
}
if(vflag)
        fprintf(stderr,"cluster initialization complete\n");
```

```
}

/*
 * parse -- parse input arguments
 *
 */

int errors;

parse(argc, argp)
char **argp;
{

        errors = 0;
        if(argc == 1){
                help();
                exit();
        }
        while(--argc > 0){
                argp++;
                if(!strncmp(*argp, "if=",3))
                        input = open(*argp+3, 0);
                ELF (!strncmp(*argp, "of=",3)){
                        output = creat(*argp+3, 0644);
                        close(output);
                        output = open(*argp+3, 2);
                } ELF (!strncmp(*argp, "cm=",3)){
                        mea = fopen(*argp+3, "r");
                } ELF (!strncmp(*argp, "cc=",3)){
                        co = fopen(*argp+3, "r");
                } ELF (!strncmp(*argp, "sf=",3)){
                        sfd = open(*argp+3, 0);
                } ELF (!strncmp(*argp, "nc=",3)){
                        num_centers = atoi(*argp + 3);
                        clneed = num_centers;
        } ELF (!strncmp(*argp, "pc=",3)){
             min_changes = atoi(*argp + 3);
                } ELF (!strncmp(*argp, "-v",2))
                        vflag++;
                ELF (!strncmp(*argp, "-t1",3))
                        tf1++;
                ELF (!strncmp(*argp, "-t2",3))
                        tf2++;
                ELF (!strncmp(*argp, "-t3",3))
                        tf3++;
                else {
                        errors++;
                        printf("bad option: %s\n", *argp);
                }
        }
        need(input, "input", "if");
        need(output, "output", "of");
        if(!clneed){
                printf("\"nc=...\"   (number of centers to create)\n");
                errors++;
        }
        if(errors)
                exit(1);
        setup_files();
}

help()
{
```

```
        printf("Summary of cluster parameters:\n\n");
        printf("if=...      input data file\n");
        printf("of=...      output results file\n");
        printf("cm=...      initial cluster mean file\n");
        printf("cc=...      initial cluster covar file\n");
        printf("sf=...      output statistics file\n");
    printf("pc=..  percent change desired(0-100) \n") ;
        printf("nc=nnn      number of centers to create\n");
        printf("option -t1 -t2 -t3 -v\n");
}
/*
 * need -- see if a needed file is present
 */
need(fd, fn, fi)
char *fn, *fi;
{
        if(fd == 0){
                printf("required %s file (%s=...) missing\n", fn , fi);
                errors++;
        }
}

/*
 *  setup_files -- read in once-only files & get ready to cluster
 */

setup_files()
{
        int i,j,k;

        /* output file */
    if(mea){
          for(i=0; i<num_centers; i++){
       for(j=0; j<num_chan; j++){
            fscanf(mea, "%f", &centers[i].origin[j]);
         for(k=0; k<num_chan; k++){
              fscanf(co, "%f", &centers[i].cov[j][k]);
     }}
       lftds_(&num_chan,centers[i].cov,&num_chan,fac,&num_chan);
      lfdds_(&num_chan,fac,&num_chan,&d1,&d2);
      centers[i].det=d1*pow(10.0,d2);;
       linds_(&num_chan,centers[i].cov ,&num_chan,centers[i].inv,
                    &num_chan);
    }}
}

/*
 * rewind_files -- rewind I/O files, and prepare for another pass
 *
 */

rewind_files()
{
        lseek(input, 0L, 0);
        lseek(output, 0L, 0);
        if(!first_pass){
                nob = read(output, obuf, NP);
                lseek(output, 0L, 0);
                nob = 0;
                obp = obuf;
        }
}
/*
```

```
 * print_stats -- output summary of clustering run
 *
 */

print_stats()
{
        register i;
        register float *fp1, *fp2;
        int j;

        time(&t2);
        printf("\tNumber of passes = %d", num_passes);
        printf("\n\tNumber of changes = %d\n\telapsed time = %ld / ",
                        num_diff, t2-t1);
        printf("%ld\n", t2-t0);
        for(i=0; i<num_centers; i++){
                printf("\n\n\tCenter %d, (%5.1f points):\n", i+1,
                                                centers[i].count);
                printf("\tOrigin:\t\t");
                printl("%6.2f\t", num_chan, centers[i].origin);
                printf("\n\tVariance:       ");
                printl("%6.2f\t", num_chan, centers[i].sumsq);
                printf("\n");
         for(j=0; j<num_chan; j++){
                printl("%6.2f ", num_chan, centers[i].cov[j]);
                printf("\n");
    }
        }
        t1 = t2;
}

printl(f, n, afp)
char *f;
int n;
float *afp;
{
        register int i;
        register float *fp;

        fp = afp;
        for(i=0; i<n; i++)
                printf(f, *fp++);
}
```