1999

Clustering in multivariate data: visualization, case and variable reduction

Sunhee Kwon
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Part of the Statistics and Probability Commons

Recommended Citation
Kwon, Sunhee, "Clustering in multivariate data: visualization, case and variable reduction" (1999). Retrospective Theses and Dissertations. 12146.
https://lib.dr.iastate.edu/rtd/12146

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600
Clustering in multivariate data: Visualization, case and variable reduction

by

Sunhee Kwon

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Statistics

Major Professor: Dianne H. Cook

Iowa State University
Ames, Iowa
1999

Copyright © Sunhee Kwon, 1999. All rights reserved.
Graduate College
Iowa State University

This is to certify that the Ph.D. dissertation of

Sunhee Kwon

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Signature was redacted for privacy.

For the Graduate College
# TABLE OF CONTENTS

1 SCOPE OF THESIS .......................... 1

2 INTRODUCTION .............................. 2
   2.1 Cluster Analysis .......................... 5
      2.1.1 Non-hierarchical Cluster Analysis .......... 6
      2.1.2 Hierarchical Cluster Analysis ............ 7
      2.1.3 Model-Based Cluster Analysis ............ 7
   2.2 Visualization .......................... 10
      2.2.1 Motivation and Related Works .......... 10
      2.2.2 Grand Tour, Guided Tour, and Manual Tour .. 11
      2.2.3 Overview of Program .................. 15
   2.3 Case Reduction .......................... 15
      2.3.1 Motivation .......................... 15
      2.3.2 Related Works ........................ 17
      2.3.3 Overview .......................... 19
   2.4 Dimension Reduction ...................... 20
      2.4.1 Problem and Related Works ............ 20
      2.4.2 Projection Pursuit .................... 21
      2.4.3 Overview .......................... 21
   2.5 Descriptions about Data Sets Used in Examples ............. 22
      2.5.1 9-D Cube Data ...................... 22
2.5.2 Flea-Beetle Data ........................................... 22
2.5.3 Satellite Image Data ........................................ 22

3 EXPLORATORY VISUALIZATION SOFTWARE: ViVA ............ 24
  3.1 JAVA for Statistical Data Visualization ....................... 25
    3.1.1 What is JAVA ............................................. 25
    3.1.2 JAVA for Statistical Software Development ............... 25
    3.1.3 Implementation ........................................... 26
  3.2 Use of Software on Flea-Beetle Data .......................... 27

4 INITIAL PARTITIONING USING MINIMAL SPANNING TREE ......... 31
  4.1 Structure Detection by Minimal Spanning Tree ................. 31
    4.1.1 Properties of Minimal Spanning Tree ..................... 31
    4.1.2 The Asymptotic Distribution of Ordered MST Edge Weight ... 33
  4.2 Simulation Study about Penrose's Theorem ..................... 36
  4.3 Initial Partitioning: Peeling(Ours) .......................... 38
  4.4 Initial Partitioning: Pruning ................................ 43
    4.4.1 Posse's Algorithm ....................................... 43
    4.4.2 Our Algorithm ........................................... 44
    4.4.3 Comparison Between Two Pruning Methods .................. 46

5 DIMENSION REDUCTION USING PROJECTION PURSUIT .......... 52
  5.1 Holes Index ................................................. 52
  5.2 Theoretical Maximum and Minimum ............................. 54
  5.3 Deciding Best Projection Dimension .......................... 55
  5.4 Optimization Algorithm ..................................... 60
  5.5 Optimum Parameter Values ................................... 62

6 EXAMPLES ..................................................... 64
  6.1 Projection Pursuit and Principal Component Analysis: 9-D Cube Data . 64
6.2 Projection Pursuit Algorithm and Relative Structure Measurement: Flea Beetle Data ........................................... 66

6.3 Initial Partitioning Algorithm and Hierarchical Model-Based Cluster Analysis: Satellite Image Data ........................................... 66

6.3.1 HMCLUST without Initial Partition ........................................... 66

6.3.2 HMCLUST with Initial Partition ........................................... 68

6.3.3 Comparison of Two Results ........................................... 69

7 DISCUSSION ........................................... 73

APPENDIX A TECHNICAL DETAILS OF VISUALIZATION PROGRAM ........................................... 75

APPENDIX B ALGORITHM FOR GROUPING FROM PEELING RESULT (SECTION 4.3) ........................................... 77

APPENDIX C PRUNING ALGORITHM (SECTION 4.4.2) ........................................... 79

APPENDIX D DERIVATION OF (3.4), (3.5) and (3.6) ........................................... 81

APPENDIX E ACCOMPANYING COMPACT DISC AND RELEVANT INFORMATION ........................................... 83

BIBLIOGRAPHY ........................................... 88
LIST OF TABLES

Table 2.1 Different parameterizations of the covariance matrix $\Sigma_k$ and the corresponding criteria to be minimized ........................................... 9
Table 2.2 Clustering methods for large data .................................................. 19
Table 2.3 Original classifications ............................................................... 23
Table 4.1 Average largest edge weights of MSTs of 100 simulated standard normal data sets ................................................................. 39
Table 4.2 Average largest edge weights of MSTs of 100 simulated 3-D data sets of two normal clusters with different means ................................. 40
Table 4.3 Average second longest edge weights of minimal spanning trees of 100 simulated standard normal data sets ................................. 41
Table 4.4 TD and A arrays for our pruning algorithms: 2-D projection of flea-beetle data ................................................................. 47
Table 4.5 TD and A arrays for our pruning algorithms: 2-D projection of flea-beetle data ................................................................. 47
Table 4.6 Comparison of distributions of size of partitions between Posse's and our pruning algorithms ................................................................. 50
Table 5.1 Maximum and minimum Holes index values using (5.9) and (5.10) 55
Table 5.2 Data 1: Holes index values for best projections, relative structure amount ................................................................. 59
Table 5.3 Data2: Holes index values for best projections, relative structure amount ........................................... 59
Table 5.4 Data3: Holes index values for best projections, relative structure amount ........................................... 59
Table 5.5 Holes index for different parameter values ........................................... 63
Table 6.1 Ratio of holes index values for projections to theoretical maximum 66
Table 6.2 True and MCLUST classification for sphered data without partition 68
Table 6.3 Number of partitions for different partition sizes ............................... 69
Table 6.4 True and MCLUST classification for sphered data with initial partition ........................................... 69
Table 6.5 Comparisons between HMCLUST running and results without and with initial partition ............................... 71
Table 6.6 Correct and incorrect classification with initial partition and without initial partition for points corresponding to partition components with different sizes ........................................... 71
Table 6.7 Cluster descriptions and colors in Figure 5.6, 5.7 and 5.8 ............................... 71
# LIST OF FIGURES

| Figure 3.1 | Grand tour with minimal spanning tree with the longest 5 edges (47-51, 6-10, 41-38, 22-24, 10-4) peeled for flea-beetle data. Points isolated by the peeling are colored differently so the 5 colors represent clusters resulting from the peeling: Original colors are available in book form only. |
| Figure 3.2 | Grand tour with minimal spanning tree with the longest 8 edges peeled for flea-beetle data. Points isolated by the peeling are colored differently so the 5 colors represent clusters resulting from the peeling: Original colors are available in book form only. |
| Figure 3.3 | Grand tour without minimal spanning tree with the longest 8 edges peeled for flea-beetle data. Points isolated by the peeling are colored differently so the 5 colors represent clusters resulting from the peeling: Original colors are available in book form only. |
| Figure 4.1 | Two different first partition components from Posse's and our algorithms |
| Figure 4.2 | Result of peeling for 2-D projection of flea-beetle data |
| Figure 4.3 | Partitioning for flea-Beetle data using Posse's algorithm |
| Figure 4.4 | Partitioning for flea-beetle data using our algorithm |
| Figure 5.1 | Plot of cluster means of data1 in 3-D subspace |
| Figure 5.2 | Plot of cluster means of data2 in 3-D subspace |
| Figure 5.3 | Plot of cluster means of data3 in 3-D subspace | 57 |
| Figure 5.4 | Plots of $M_1$ and $M_2$ versus dimension for each data | 60 |
| Figure 6.1 | Histograms of 1-D projection by PP, 1st PC and 5th PC | 65 |
| Figure 6.2 | 2-D Projection by PP optimization | 65 |
| Figure 6.3 | Two dimensional projection found by algorithm | 67 |
| Figure 6.4 | Two dimensional projection found by guided tour | 67 |
| Figure 6.5 | Original clusters: Original colors are available in book form only. | 70 |
| Figure 6.6 | Clusters found by HMCLUST with initial partition: Original colors are available in book form only. | 70 |
| Figure 6.7 | Clusters found by HMCLUST without initial partition: Original colors are available in book form only. | 72 |
1 SCOPE OF THESIS

We present a new method for partitioning large data into a set of many small classifications in Section 3.3 and 3.4. By starting clustering from this partitioned data instead of the usual set of singleton clusters, we can obtain the faster cluster result without much loss of accuracy, which relieves the speed problem of complicated hierarchical agglomerative clustering methods such as hierarchical model-based clustering.

We introduce our new multivariate visualization software in JAVA(ViVA) in Chapter 2. This is the first JAVA software that performs grand tour, constructs minimal spanning tree(MST) of data and shows the cluster separations of MST edges with linked brushing between grand tour window and MST edge weight graph window.

A new projection pursuit algorithm which searches the projection maximizing Holes index value both globally and locally is suggested in Section 4.4 along with possible criteria for deciding best projection dimension. The random search is used in this algorithm in order to avoid the problem of confronting too many local optima in gradient based search methods.
2 INTRODUCTION

Exploratory data analysis involves probing data for unknown structures and searching for trends as well as anomalies. It may involve using prior knowledge or preconceived models in the exploration process, but it also facilitates finding the unexpected. It reveals the structures in data through the analysis process. Exploratory data analysis is for describing the given data, rather than estimating the properties of assumed population or predicting future observations. It is more interested in grasping and understanding the data first than confirming the significance of findings (Tukey and Tukey, 1981; Glymour et al, 1997). Thus, the findings of exploratory data analysis have meanings when they can show "what the data seems to be" (Tukey and Tukey, 1981), even though they have not been tested against any alternatives.

Cluster analysis is a common exploratory multivariate data analysis method which groups similar objects together (Hartigan, 1975). Its aim is to group the given data, not to give inferences about any objects not in the data. It is data-driven. In most cluster analysis problems, we don't have any prior knowledge or assumptions about clusters in the data.

The rapid growth of data size in cases and dimensions leads cluster analysis to receive more attention. Large data sets first ask if they can be grouped into more homogeneous groups before we generate models and test their appropriateness for each groups. Also the developments of new dynamic and interactive visualization tools are giving us more indications of inhomogeneity in data which might have been unnoticed with other graphical methods and leading us to group the data into more homogeneous
Visualization of data plays an important role in cluster analysis. It not only indicates the existence of clusters and leads to clustering, also often does the cluster analysis interactively.

In this thesis, we will introduce our multivariate visualization software which is designed for visualizing the cluster structures in data by providing grand tour, minimal spanning tree in grand tour and results of data partitioning. Grand tour is a dynamic visualization method for multivariate data which shows a continuous sequence of lower dimensional projections and enables us to detect structures of the data in the original high-dimensional space better by giving the context between projections. Minimal spanning tree (MST) is a widely used tool for structure findings in pattern recognition field. The MST connects all the data points with minimum sum of edge weights. By looking at the changing context of points connected with MST edges in grand tour, the projection views with complex structures and many data points will become easier to decipher. Also this program offers the graph of MST edge weights and linked brushing/peeling. It is written in JAVA so that it can be run broadly across platforms and uses the strong graphics classes of JAVA.

The time and memory requirements for clustering are often problems especially for large data sets. This is more serious with hierarchical agglomerative methods where the speed is more than quadratic in the number of cases. Posse (1998) suggests to initially partition the data with a minimal spanning tree and to start the hierarchical agglomeration from the partition classification, rather than from the usual singleton clusters. His partitioning consists of two steps: peeling the longest minimal spanning tree (MST) edges to delete outliers and disconnect well-separated clusters, and pruning the surviving fragments into smaller partitions. Some problems arise with his peeling and pruning processes. Peeling, which is solely based on the global properties of MST edges such as edge weight distribution, may fail to separate locally well-separated clusters. Also
we have found from simulation studies that the convergence to asymptotic distribution of MST edge weights from one standard Gaussian distribution (Penrose, 1998) where Posse compares the edge weight distribution of data, is too slow to be compared. Thirdly, his algorithm for traversing MST edges in the pruning step may produce the partitions which assign distant points together while assigning closer points to different partitions. The reason for this is because his algorithm considers only one MST edge for each node while it is possible for one node to be connected to many other nodes in MST. This is a more serious problem than that of peeling stage, because points assigned to the same partition inappropriately will not have a chance to be regrouped in real cluster analysis.

In this thesis we will present different peeling and pruning algorithms (Chapter 4). The peeling algorithm we suggest compares the weight of each edge in MST to its neighborhood by adopting Zahn's (1975) strategies for structure finding. This not only helps to separate local small clusters, but enables us to assign the points in one large cluster with low density into same partition. Our pruning algorithm successfully keeps unrelated points from being partitioned together. The arrays from data required for running this algorithm are defined, and their construction algorithms are also presented.

In many cases of high-dimensional data, clusters exist only in lower dimensional subspace of original dimension. The reduction of data dimension enables us to detect clusters better by eliminating the influence of nuisance variables. Typically principal component analysis is used for dimension reduction, but this method can seriously destroy cluster structures in the process. We use projection pursuit of cluster for reducing dimension, by looking for the linear projections that are the most revealing of the cluster structure of data in the original dimension. Projection pursuit involves optimization. The traditional search algorithms for best projections, based on gradient of projection index (Jones and Sibson, 1987), have too many local optimum projections. Posse (1995) developed the algorithm for producing one global optimum projection which combines global and local search. His algorithm was devised for only 2-dimensional projection.
In Chapter 5, we present new projection index optimization algorithm which combines global and local approaches in its search process. It adopts the algorithm of Posse and generalizes it to any dimensional projections.

2.1 Cluster Analysis

Cluster analysis groups the observations into some clusters so that observations within clusters are more similar to each other than to the observations in different clusters. It reduces the overwhelming mass of data to a manageable number of discrete categories and help in making new hypotheses about the structure in data (Eckes, 1986). The similarity or dissimilarity should be defined before performing clustering algorithms. Clustering differs from classification in that classification uses the knowledge of real group assignments of any data sets to classify different data into the clusters which were found from previous data with known groups, but the user doesn't have separate data with known cluster assignments in cluster analysis. In this respect cluster analysis is also referred to as unsupervised classification. Methods of cluster analysis are divided into non-hierarchical and hierarchical ones. Non-hierarchical methods provide single partition of data as the final answer for usually pre-defined number of clusters. Hierarchical methods produce partitions for every possible number of clusters, that is, from one to as many as data size, because it proceeds by either a series of agglomerations or a series of divisions (Johnson and Wichern, 1988). The layout of clusters and the observations belonged to each clusters are decided when the user cuts the resulted dendogram at specific number of clusters. Thus the problem of deciding (or finding) the right number of clusters exists in both non-hierarchical and hierarchical methods.
2.1.1 Non-hierarchical Cluster Analysis

Non-hierarchical cluster analysis methods are more iterative than hierarchical. The number of clusters, \( K \), is usually specified in advance. Non-hierarchical methods start from either initial partitions of objects into \( K \) groups or an initial set of \( K \) seed points consisting the nuclei of clusters. K-means, K-medoid and their varieties are typical and mostly-used non-hierarchical methods. K-means method is to start with assuming \( K \) initial seeds which represent \( K \) clusters and assign each case to the cluster with minimum distance between its seed and the case. After each assignment, the cluster seed is updated as the new mean of the cluster, and this process is repeated until the cluster assignments don’t change any more. K-medoid method also relocates cases to the nearest clusters. But it chooses “seeds” from the data itself, and the updated seeds are not newly computed means but newly chosen cases which minimize the distances between them and the cases in the clusters. Non-hierarchical clustering method using neural network (NN) regards data attributes and final cluster assignments as the nodes in input layer and output layer respectively (Balakrishnan et al, 1994). But NN clustering methods with supervised learning rules, in which separate training data with known correct classification of each cases in the data are needed to “learn” the rules, should be excluded in terms of our definition of clustering analysis. Unsupervised learning NN clustering method is trained using the data attributes themselves. In these methods, the weight vectors are updated through iterations starting from random initial weights and final weight vectors are used to classify data. The iterations to get reliable weight vectors are usually mode than 1000-2000 times. The amount of time and memory even for medium-sized (by the definition of Huber, 1994) data would be tremendous. Also, the performance of unsupervised NN clustering with Kohonen learning rule, which is the unsupervised learning method used most frequently (Balakrishnan et al, 1994), is poor compared to K-means method. Because of this it requires other learning rules in
unsupervised NN clustering to be used in general clustering applications.

2.1.2 Hierarchical Cluster Analysis

Hierarchical methods are either agglomerative or divisive. In agglomerative methods, the first stage starts with the singleton clusters as many as the data size. The pair of clusters which optimizes some criterion are agglomerated at each stage corresponding to each number of clusters. Except model-based methods, most of hierarchical clustering methods use the criteria for optimization as between clusters distance (single linkage, average linkage, ...) or sum of squares (Ward method) (Fraley and Raftery, 1998). On the other hand, model based clustering method agglomerates two clusters which maximize the estimated likelihood after agglomeration. This is based on the Bayesian assumption that true classifications are parameters to be estimated and these parameters can be estimated by maximizing the posterior probability given the data over all possible classifications. Banfield and Raftery (1993) showed that the tendency of the Ward method to make “circular” clusters of roughly same size and its sensitivity to outliers caused the Ward method to perform poorly with three partially-overlapping elongated clusters. (The performance of single linkage clustering was worse with this data.) The model-based clustering method successfully reshaped the three clusters in this simulated data. Also recently this method has produced particularly good results in real examples such as detection of tissue types from MRI brain scan and classifications in geophysical science. The program for model-based clustering algorithm was implemented as a S-Plus function (mclust) and many of the limitations in its earlier versions were improved. We use this method in this research.

2.1.3 Model-Based Cluster Analysis

In this method we assume the following probability model for a p-dimensional data with size N, \( x = (x_i, i = 1, \cdots, N) \). The population is composed of G different subpop-
ulations and the density of a p-dimensional observation \( y \) from the k-th subpopulation is \( f_k(y|\theta) \) for some unknown vector of parameters \( \theta \). If we denote the cluster identifying label of each \( x_i \) as \( \gamma_i \), where \( \gamma_i = k \) if \( x_i \) comes from the subpopulation \( k \), the likelihood for data \( x = (x_i, i = 1, \ldots, N) \) is

\[
L(x; \theta, \gamma) = \prod_{i=1}^{N} f_{\gamma_i}(x_i; \theta),
\]

where \( \gamma = (\gamma_1, \ldots, \gamma_N)^T \). In the classification maximum likelihood approach, \( \theta \) and \( \gamma \) are chosen so as to maximize (2.1) (Banfield and Raftery, 1993). The multivariate normal distributions (MVN) for \( f_k \)'s were shown to work fairly well with many cases (Banfield and Raftery, 1993; Celeux and Govaert, 1993), though other distributions are possibly assumed. Currently the \texttt{mclust} program, which is a software package for hierarchical model-based cluster analysis in Fortran and interfaced to the S-PLUS (Fraley and Raftery, 1998), considers only normal distribution for \( f_k \)'s. If we assume MVN(\( \mu_k, \Sigma_k \)) for \( f_k \)'s for \( 1 \leq k \leq G \), (2.1) becomes

\[
L(x; \theta, \gamma) = \prod_{k=1}^{G} \prod_{i \in \mathcal{I}_k} (2\pi)^{-\frac{p}{2}} |\Sigma_k|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k)\right),
\]

where \( \mathcal{I}_k = \{i; \gamma_i = k\} \), i.e. \( \mathcal{I}_k \) is the set of indices of \( x_i \)'s which belong to cluster \( k \). If we replace \( \mu_k \) with the maximum likelihood estimator \( \hat{\mu}_k \), which is the mean vector of \( x_i \)'s in cluster \( k \), for each \( k \), the log-likelihood is

\[
l(x_1, \ldots, x_N, \mu_1, \mu_2, \ldots, \mu_G; \Sigma_1, \Sigma_2, \ldots, \Sigma_G, \gamma) =
-\frac{mn \log(2\pi)}{2} - \frac{1}{2} \sum_{k=1}^{G} \{tr(W_k \Sigma_k^{-1}) + n_k \log |\Sigma_k|\},
\]

where \( W_k = \sum_{i \in \mathcal{I}_k} (x_i - \bar{x}_k)(x_i - \bar{x}_k)^T \) and \( W_k/n_k \) is the maximum likelihood estimator of \( \Sigma_k \). Different assumptions on the covariance matrices (\( \Sigma_k \)) should lead to different criteria that classifications \( \gamma \) should optimize to maximize (2.3). The assumptions can be parameterized in expressing the covariance matrices in terms of their eigenvalue...
decompositions.

\[ \Sigma_k = \lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T (1 \leq k \leq G), \]  

(2.4)

, where

\[
\begin{aligned}
\lambda_k &= \text{the largest eigenvalue of } \Sigma_k \\
\mathbf{D}_k &= \text{the orthogonal matrix of eigenvectors of } \Sigma_k \\
\mathbf{A}_k &= \text{a diagonal matrix with the normalized eigenvalues of } \Sigma_k \text{ on the diagonal in a decreasing order}
\end{aligned}
\]

Table 2.1 summarizes the criteria to be minimized (in order to maximize the log-likelihood) to corresponding six different parameterizations of \( \Sigma_k \) as above (Fraley, 1996; Celeux and Govaert, 1995).

Table 2.1 Different parameterizations of the covariance matrix \( \Sigma_k \) and the corresponding criteria to be minimized

<table>
<thead>
<tr>
<th>( \Sigma_k )</th>
<th>eigenvalue decomposition</th>
<th>size</th>
<th>shape</th>
<th>orientation</th>
<th>criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda \mathbf{I} )</td>
<td>same</td>
<td>same</td>
<td>NA</td>
<td>( \text{tr}(\sum_{k=1}^G \mathbf{W}_k) )</td>
<td></td>
</tr>
<tr>
<td>( \lambda_k \mathbf{I} )</td>
<td>different</td>
<td>same</td>
<td>NA</td>
<td>( \sum_{k=1}^G n_k \log[\text{tr}(\mathbf{W}_k)] )</td>
<td></td>
</tr>
<tr>
<td>( \Sigma )</td>
<td>( \lambda \mathbf{D} \mathbf{A} \mathbf{D}^T )</td>
<td>same</td>
<td>same</td>
<td>same</td>
<td>( \sum_{k=1}^G \mathbf{W}_k )</td>
</tr>
<tr>
<td>( \Sigma_k )</td>
<td>( \lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T )</td>
<td>different</td>
<td>different</td>
<td>different</td>
<td>( \sum_{k=1}^G n_k \log[\mathbf{W}_k] )</td>
</tr>
<tr>
<td>( \Sigma_k )</td>
<td>( \lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T )</td>
<td>different</td>
<td>same</td>
<td>different</td>
<td>( \sum_{k=1}^G n_k \log(\mathbf{S}_k) )</td>
</tr>
<tr>
<td>( \Sigma_k )</td>
<td>( \lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T )</td>
<td>same</td>
<td>same</td>
<td>different</td>
<td>( \sum_{k=1}^G \mathbf{S}_k )</td>
</tr>
</tbody>
</table>

\( \mathbf{S}_k = \text{tr}(\mathbf{A}^{-1} \mathbf{\Omega}_k), \mathbf{W}_k = \mathbf{L}_k \mathbf{\Omega}_k \mathbf{L}_k^T \) in its eigenvalue decomposition form.

If we assume the first type of covariance matrix (spherical normal distribution uniform over all groups), the result is the same as the well-known Ward's method (also the same as K-means). For other types of covariance matrices, this method requires additional computation and memory for determinants of \( \mathbf{W}_k \)'s. This additional computation makes the running of hierarchical model-based clustering slow especially for large data. Starting with singleton clusters (if the user doesn't provide initial partitions) the hierarchical agglomeration process proceeds as follows until all observations are in a single
cluster.

(i) For every pair of two clusters, the cost of merging these two ($\Delta(i,j)$) is computed. For instance, in the simplest case of $\lambda I$ for all $1 \leq k \leq G$, the cost of merging two singleton clusters $i$ and $j$,

$$\Delta(i,j) = tr(W_{<i,j>}) - [tr(W_i) + tr(W_j)],$$

where $<i,j>$ is the cluster formed by merging singleton clusters $i$ and $j$.

(ii) The pair of two clusters which minimizes the above cost is combined into a new cluster.

Consequently, $\binom{N}{2}$ values of $\Delta(i,j)$ should be computed at the beginning stage of clustering. Also each $\Delta(i,j)$ requires extensive computations such as the determinants of cross-product matrix. Fraley (1996) proposed the efficient methods for these computations using the relationship between values of criteria at successive stages, for each covariance matrix types in Table 2.1. But even the improved computations are still quadratic in the number of initial singletons.

2.2 Visualization

2.2.1 Motivation and Related Works

The visualization of data helps us to detect clusters which are hardly detected by other clustering algorithms using human pattern perception ability (Zahn, 1975). Two touching uniform clusters in a two-dimensional data would be easily detected by human eyes, while most clustering algorithms fail to separate these. It also guides us in choosing which clustering algorithm we should use for the specific type of structure in data and/or deciding values for the parameters required for running the algorithms to obtain correct
results. For instance, after we watch the rotation of a three dimensional data which has two Gaussian clusters, we will be able to give more correct covariance matrix type as the parameter for model-based cluster analysis. Grand tour of a high-dimensional data may give more plausible candidates for the number of clusters in K-means clustering. This is a crucial role of visualization in clustering process, since different clustering algorithms and different parameter values very often lead to totally different clustering results for the same data. Graphical representations of clustering results can also be used as validation tool for clustering and complement the numerical algorithm (Kim, Kwon, and Cook (1998)).

Grand tour has been successfully used in cluster detection for multivariate data by providing an overview of the original data on the projected space, within time constraints. It is more successful when combined with projection pursuit. Its possibility to spend too much time presenting uninteresting projections is somewhat relieved by the guidance of the projection index ("guided tour" : Cook, Buja, Cabrera, and Hurley, 1995) and the user-interactive projection control facilities in grand tour programs ("manual tour" : Cook and Buja, 1997). However, with more complex structure and many data points, the projection views can be messy and hard to decipher (Kwon and Cook, 1998).

We suggest to overlay minimal spanning tree (MST) of data in grand tour as an approach to solve this problem. The MST is a tree which connects all data points with its edges and minimizes the sum of edge weights among all such trees. The changing context of points connected with MST edges in grand tour will give more information about cluster structures in data than what grand tour alone gives.

2.2.2 Grand Tour, Guided Tour, and Manual Tour

Statistical visualization methods for high-dimensional data should consider the following three points of view: data's geometry in the high-dimensional space, the user's
perception in 3-dimensional space, the limitation of computer screen to 2-dimension and computational limits (Young, Faldowsky and McFarlane, 1993). In addition, these requirements should be met for large data which are common these days. Considering that the computer screen seems to continue to be fixed at a 2-dimensional plane at least within near future and the scopes of speed and memory are being broadened very fast, it would be more reasonable to concentrate on the first and the second requirement. Most static graphics methods suffer from lack of the first and second property, especially with large data. It is very difficult for human to perceive structures in parallel plots or glyphs drawn for large data. Also they are ignorant of the geometry of data in its original space. The scatterplot matrix, especially with linking, brushing and subsetting, gives a much more informative glimpse of data structure to the user. It can show where the cases at certain point in two-dimensional XY-plot of any two variables are located in another XY-plot of another pair of variables. But scatterplot matrix doesn’t give any information about the structures which any subspace, generated not only by selection of two variables, but also by any linear combinations of all variables, of the data might have. In fact, a scatterplot matrix can be regarded as a matrix of all possible two dimensional projection where the projection matrices have only 0 and 1 values as their elements. From the same reasons that Fisherkeller, Friedman and Tukey (1974) pointed out why rotation should be added in static scatterplot matrix, we are not interested in all the scatterplots in scatterplot matrix but only in the scatterplots which are showing interesting structures. That is, we automatically select the interesting plots among all \( \binom{p}{2} \) scatterplots. But at the same time, we want to look at many, though not all, XY-plots together and find structures in data. This is difficult with static plots like scatterplot matrix. The most helpful way for doing this is to add some form of motion. The motion can be either smooth or alternative, but smooth motion is more effective than plain alternation of static graphics. One strong advantage of graphics with smooth motion over static graphics is that
• Smooth transitions between images help the user to recognize objects as the same across changing views.

With these smooth transitions, objects preserve their identity in user's perception, but static images presents the task of identifying objects across separate pictures. This can be frustrating when all objects in data look alike, such as dots. Grand tour is one of statistical graphic methods with smooth motion. It is showing all possible projections of data onto a lower-dimension within time constraints (Cook, Buja, Carbrera and Hurley, 1995), whereas static scatterplot matrix is showing all possible XY-plots of variables. Grand tour is data's moving through a continuous sequence of projections which are chosen to be dense in the set of all projections (Asimov, 1995). But even though grand tour gives the neighborhood context which scatterplot matrix falls short of, it may take too much time to locate, or never meet, the interesting projections because we only can watch it within a limited amount of time (Cook, Buja, Carbrera and Hurley, 1995). To circumvent this, the user can be guided by some index which can represent the amount of structures in each projection while watching the grand tour. This index is called the projection index and measures the “interestingness” of projections. The normal distribution is considered to be the least interesting and projection index represents the departure from non-normality of the projection (Friedman, 1987). Projection pursuit is to pursue the most interesting structured projections by looking for the projections with maximum projection index. Either grand tour or projection pursuit alone would not be enough for catching interesting projections, because grand tour alone may skip the interesting projections due to time constraint as described above and projection pursuit alone depends only on global optimum and may miss interesting local optima (Cook and Buja, 1997). These two should be combined together and guided by each other in structure findings (Projection pursuit guided tour: Cook, Buja, Carbrera and Hurley, 1995).
To be properly guided by the projection pursuit index value, the tools for motion alteration such as to track the path of grand tour backward and/or temporary restriction of movements to certain projections should be added so that we can avoid unnecessary touring through the sequence of uninteresting projections. The authority to use these facilities should be in the user's hand, not in the system's. That is, the tools should be performed through user interaction. The grand tour which possesses these facilities is called a "manual tour" (Cook, Buja, Carbrera and Hurley, 1995). In manual tour, interactive manipulation methods allow the user to control the data point cloud in a natural and perceptual way. The major benefit of this interactive direct manipulation by the user would be an ability to control the orientation of a surface in a natural way, as the user moves it around like a real object held in his/her hands (Cleveland, 1993). This would assist the user to guess the structures in data (Young, Faldowsky, and McFarlane, 1993). Also the combination of other multivariate visualization tools such as linked brushing and masking with grand tour, can much improve the user's structure detection ability (Buja, Cook and Swayne, 1996).

Due to real-time graphics and developments in computer hardware, grand tour along with real-time projection pursuit have been available in softwares since early 1990's. Setting aside softwares providing only 3-D rotations, Lisp-Stat (Tierney, 1990) and XGobi (Swayne, Cook, and Buja, 1998) are two major softwares for grand tour. But there is the need for more platform independent (such as UNIX and PC) and more graphical programs including grand tour implementation so that graphical analysis methods which are usually restricted to certain platforms can be more available. JAVA meets these requirements in that (i) it can be run on any implementation of JAVA virtual machine and (ii) JAVA class packages already include highly interactive graphical interface, which makes dynamic graphical methods like linking, rotation, and dynamic parameter control much easier to implement. In this thesis, we introduce a new data visualization program which includes grand tour, MST construction and other structure finding tools.
such as linked MST peeling and pruning.

2.2.3 Overview of Program

Chapter 3 describes major functions and layout of our exploratory visualization software. This implements the two methods above: grand tour and minimal spanning tree in its main window. Also it displays the graph of ordered MST edges in the separate window on user command along with the linked brushing function between this window and the main grand tour window. This was inspired by the fact that longest MST edges can be the estimators of cluster separations (Zahn, 1975; Banks and Lavine, 1992). The user can disconnect the edges and make partitions consequently, either by scrolling the edge weight graph or brushing the individual points, which represents the MST edges in this graph. The resulting partitions can be colored with different colors corresponding to their groupings from disconnecting edges with our MST traverse algorithm. This will allow the user to follow which MST edges are connecting certain structures in the data. Also, since the clustering result from disconnecting longest $k$ MST edges is same as single linkage clustering for $k+1$ clusters, we can look at how single linkage method separate clusters.

2.3 Case Reduction

2.3.1 Motivation

As described in Section 2.1, a main problem of hierarchical clustering algorithm is its time and memory requirement. The computation time more than quadratic in the number of observations (i.e. the number of components in the initial partitions) makes the fast clustering of large data sets difficult (Posse, 1998). This problem is worsened with model-based methods (Section 2.1.3), because they require complicated computations such as determinants of the cross-product matrices for each agglomeration stage.
(Fraley, 1996). The efforts attempted to circumvent this problem are mostly either i) to cluster sample of data and classify total data with this result or ii) to pre-cluster (initially partition) coarsely the entire data and start cluster analysis from the partitions. The use of sampling method in i) was discouraged by Posse (1998). The possible underrepresentation of small but important clusters in subsample would lead to different clustering results of the entire data. Alternatively he suggests to use classification of the data in many classes instead of usual singleton clusters at the beginning stage of agglomeration process. To obtain the initial classification, he uses a subgraph of MST. Minimal spanning tree (MST) is a spanning tree for which the sum of edge weights is a minimum. It is defined and constructed for a given edge weighted graph, which is a graph with a real number ("weight") assigned to each edge. Graph consists of a set of nodes (usually data points) and a set of edges, which link the two nodes defining them. Euclidean distance between two data points would be a possible "weight" in an edge weighted graph. A tree is a connected graph with no cycles. A connected graph is a graph which has paths between any pair of nodes. A path between two nodes is a sequence of edges joining two nodes. A cycle is a path beginning and ending with the same node. A subgraph of a given graph is a graph with all of nodes and edges in the given graph. A spanning subgraph of a given graph is a subgraph which has a same node set as that of the given graph. A spanning tree of a graph is a spanning subgraph that is a tree (Friedman and Rafsky, 1979; Zahn, 1971).

Given data with n points, MST of the data consists of n-1 edges and the node pairs defining the edges represent points that tend to be close to each other (Friedman and Rafsky, 1979), which makes MST be used in structure findings.

The subgraph of MST which Posse uses for making initial classification is produced by i) trimming out the longest MST edges to separate outliers and observations in the surroundings of clusters and edges connecting any well-separated clusters and ii) breaking the surviving connected components of the MST into smaller ones of roughly
the same size. MST is a good tool for fast classification, because i) it connects all data points (say n as the number) with only n-1 edges and ii) it has minimum total sum of edge weights among trees such that they can connect entire data set with tree type.

But Posse’s peeling and pruning algorithms have the following possible problems.

[i] While peeling the longest edges, it may fail to peel the locally long edges which are connecting small clusters. This is because his peeling method is considering only the entire comparison of all n-1 MST edges.

[ii] The convergence rate to the asymptotic distribution of MST edge weight from one normal cluster by Penrose (1998) which is a basis of criterion for deciding the number of longest edges to be peeled to separate clusters in Posse’s peeling method is too slow to be applied even for relatively large data (with 5000 cases).

[iii] His pruning method (Section 4.4.1) is prone to produce line-type partition component without including closer points together. This is because his way to traverse MST is simply to follow the connection in A array (see Section 4.4.3 for explanation).

Considering above problems, we need a different partitioning algorithm which takes into account i) different criteria for deciding the edges to be peeled, ii) the local comparison of edge weights with its neighborhoods (related to i), and iii) including other information about MST connection in addition to A array from Prim’s (1957) MST construction algorithm. We suggest an initial partitioning algorithm which consider these three problems in Chapter 4. The overview of this chapter is described in Section 2.3.3.

2.3.2 Related Works

Between the two different methods to reduce the data size for cluster analysis: subsampling and initial partitioning as discussed in the previous section, we use the initial
partitioning method. Initial partitioning method should be fast and simple without taking the risk of mis-partitioning the points, because mis-partitioned data points would not have a chance to be relocated to the right cluster later. Thus, this method should leave any unconvincing case as a singleton partition, rather than force it to be assigned to any previously formed partitions. Initial partitioning using prior hierarchical cluster analysis was tried by Davis and Kalkstein(1990), Stooksbury and Michaels(1990) and others. Most of them used the result of average linkage cluster analysis as the initial partition for the later K-means clustering. Their aim of initial partitioning is to get the appropriate cluster numbers and use them as the inputs of K-means clustering than to reduce large data. Opposite to these methods, K-means, C-means, or related SOM(self-organized map) methods can be used for initial partitioning, but as Posse(1998) pointed out, they have higher rates of misclassification. While trying to assign every point to the nearest mean(k-means) or medoid(k-medoid), the forced misclassification of outliers would be inevitable as well.

Table 2.2 describes which methods have been developed and used for large data with one of these two methods. For methods using sampling, Banfield and Raftery(1993) classified total data with the discriminant function which was the result of model-based cluster analysis of a sample of the original data. Kauffman and Rousseeuw performed non-hierarchical K-medoid clustering(software PAM) on total data with K seeds which were computed from previous K-medoid method run on a sample of total data(software CLARA). CLARANS is the software developed by Raymond and Han(1994), which performs K-medoid method for total data, but search the medoids only in the neighborhoods of previous medoids. CLAN developed by Stahl(1986) uses the “leader” algorithm by Hartigan(1975) for pre-clustering data, which classifies data into the groups of which the “leader” has the distances from each data points in the group within predefined threshold values. It allows any computationally heavy clustering methods to run for this pre-groups and uses the results for clustering total data. Posse(1998) pointed out
that clustering performed on sub-sampling might lead to different solutions from the one on the total data. He instead broke the total data into small partitions ("prune") with the use of minimal spanning tree and ran model-based clustering for this partition set.

<table>
<thead>
<tr>
<th>Software name</th>
<th>developers</th>
<th>sampling or initial partitioning</th>
<th>clustering method</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLARA</td>
<td>Kaufman and Rousseeuw</td>
<td>sampling</td>
<td>K-Medoid</td>
</tr>
<tr>
<td>CLARANS</td>
<td>Raymond and Han</td>
<td>K-medoid</td>
<td></td>
</tr>
<tr>
<td>CLAN</td>
<td>Stahl</td>
<td>initial partitioning</td>
<td>any</td>
</tr>
<tr>
<td></td>
<td>Banfield and Raftery</td>
<td>sampling</td>
<td>model-based</td>
</tr>
<tr>
<td></td>
<td>Posse</td>
<td>initial partitioning</td>
<td>model-based</td>
</tr>
</tbody>
</table>

2.3.3 Overview

In Section 4.1, we introduce the properties of MST which have been used for structure detection. Penrose's theorem about the asymptotic distribution of the largest MST edge weights of data from one standard normal point cloud and Posse(1998)'s generalization of it to \( l \)-th largest edge weights are described. The weight defined here is the Euclidean distance between two ending nodes. We derive the asymptotic estimators of means and variances based on their results. In Section 4.2, we simulate 100 data sets from one standard normal distribution for each different dimensions (2, 3, 5, and 10) and different sizes (50, 100, \( \cdots \), 5000). The average mean of largest edge weights from 100 simulated data sets is compared to the theoretical asymptotic confidence interval for the mean. The same studies are done for second largest edge weight. From the numerical result about slow convergence of asymptotic distributions, we suggest a different peeling method in Section 4.3. Also, we introduce a different pruning method from Posse which can classify more relevant points into same components in initial partition. The pruning process of our algorithm is visualized using the flea-beetle data example and is compared to Posse's pruning result for the same data.
2.4 Dimension Reduction

2.4.1 Problem and Related Works

Principal component analysis (PCA) has been being the most prevalent method for reducing data dimension with linear projections. PCA computes the eigenvalues and eigenvectors consisting the covariance (or correlation) matrix and reduces the data to lower dimensions by projecting the data onto the space orthogonally spanned by the first few eigenvectors. PCA is successful in finding structures where the distances between clusters are larger than the distances within clusters, usually in case of few large clusters. But when the data sets have (i) many small clusters not well separated, and/or (ii) meaningless variables with high noise level, the clusters may not be picked out by PCA (Huber, 1985; Nason, 1995). The example of cases where PCA fails to find clusters were illustrated by Friedman and Tukey (1974). Also, the order of principal components (PC), which is according to their contribution to the total variability, is not always same as its structure discriminating power (Rao, 1964; Kshirsagar, 1990). The important PCs may not be in the first few. Thus, we may have to think that PCA is to remake the same number of variables from the original variables and reorder them. To select the important PC's has been an issue in terms of dimension reduction.

Multidimensional scaling (MDS) is a mapping data from original dimension to lower dimensions while preserving the ordering of original distance matrix (Johnson and Wichern, 1991). But the nonlinear minimization process of stress (a measure of the extent to which the mapping falls short of a perfect match for order in original dimension) is very computationally demanding and makes it difficult to be widely used especially for large data.
2.4.2 Projection Pursuit

Projection pursuit (PP) is to look for the “interesting” lower dimensional projections by numerically maximizing a criterion function (Projection Index (PI): Huber, 1985). The interestingness is defined as the non-normality, since

(i) The multivariate normal density is completely specified by its covariance structure.

(ii) All projections of data from a multivariate normal distribution are normal. Thus the non-normality in lower dimensional projections is evident for non-normality in original dimension.

(See Huber(1985)). PP doesn’t look for structures which can be described by location, scale and covariance structure, because they are completely represented in PCs and it is wasteful to search for these. Thus PP starts from sphered data, that is, the PCs. To optimize (maximize) the PI value, gradient-based method and random search method are used. The traditional gradient-based optimization method in maximizing PI has high possibility to get stuck to dummy local optima because most PIs are highly oscillating functions. Huber(1990) combined the global random search and the gradient based local optimization method in his optimization. His method to go over many local optima started from random starting points and find the maximum. Posse(1993) applied Huber’s method to 2-D projection pursuit.

2.4.3 Overview

We propose our optimization algorithm for projection index in Chapter 5. Section 5.1 explains about Holes index which is used in this algorithm. Tools for measuring the relative amount of structures in the best projections found by the algorithm are presented in Section 5.3. Section 5.4 describes the optimization algorithm. The optimum values for parameters in the algorithm from empirical studies are suggested in Section 5.5.
2.5 Descriptions about Data Sets Used in Examples

2.5.1 9-D Cube Data

This data is composed of points in the vertices of 9-D cube in 9-dimensional space (Buja, Cook, and Swayne, 1996) which has statistical relevance in two areas: 9 Bernoulli random variables and $2^8$ factorial experimental design. But most random projections of this data onto lower dimensions look as if they were distributed as multivariate normal (Diaconis and Freedman, 1984).

2.5.2 Flea-Beetle Data

The data consists of 74 flea beetles from three species (concinna, heptapotamica and heikertingeri). Each beetle has six variables which are measurements of different parts of its shape (Lubischew, 1962). The aim is to cluster 74 objects into three groups without species information.

2.5.3 Satellite Image Data

The data was generated from Lansat Multi-Spectral Scanner image data by the Center for Remote Sensing, Australia. It consists of four different spectral values of 4435 pixels in one satellite image with the classification of each pixel. That is, the same pixel has four different spectral values in each different spectral bands. Each value can have 0 corresponding to black to 255 corresponding to white. The data was given in random order and certain lines of data had been removed so its user cannot reconstruct the original image from this data set. The classification labels of pixels are in Table 2.3. The aim is to predict these classifications, given the multi-spectral values (Srinvasan, 1993).
Table 2.3  Original classifications

<table>
<thead>
<tr>
<th>Height</th>
<th>Class number</th>
<th>Class description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Red Soil</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Cotton Crop</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Grey Soil</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Damp Grey Soil</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Soil with Vegetation Stubble</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>Very Damp Grey Soil</td>
</tr>
</tbody>
</table>
3 EXPLORATORY VISUALIZATION SOFTWARE: ViVA

This software implements mainly 2-D grand tour, MST construction in the original space in grand tour and plotting the graph of ordered MST edge weights. The MST edge weight graph provides linked brushing with grand tour window. As the user brushes the points in MST edge weight graph, the corresponding edges in grand tour window are automatically faded so that the user can follow the cluster separation from disconnecting MST edges. Instead of brushing points directly, the user can peel the longest edges using the scroll-bar. As he/she scrolls the bar from the left(0%) edge to the right, the edges with weights larger than the quantile represented by the scroll position are faded so that the user can look at the changes in cluster separation as different quantiles of MST edges are peeled. The grouping of points by peeling MST edges and coloring the points according their groups can be done by pushing buttons in grand tour window. The group assignments and the information about peeled edges can be written into separate file. (Whether each edge has been peeled or not becomes the element of F array, input for pruning step(see Section 4.4)). Currently two manual control methods, pausing/resuming of grand tour and tour speed control are provided with button and control-bar. These allow the user to stop at interesting projections or examine the sequence of interesting projection carefully. The technical details are described in APPENDIX.
3.1 JAVA for Statistical Data Visualization

3.1.1 What is JAVA

JAVA is a network programming language such as Perl and JavaScript and also an object-oriented application development language like C++ (Niemeyer and Peck, 1996). It was originally developed for making dynamic Web pages in 1995 by Sun Microsystems, but by now(1999) it has become one of the most dominant high-level languages for general implementation purposes due to the following major advantages.

1) portability: JAVA classfiles can be run on any platform wherever there are JAVA Virtual Machine(VM) and the necessary classfiles of JAVA packages.

2) strong built-in features for handling diverse file types such as graphical user interface(GUI) components and multimedia(audio, images, video, ...) 

3) Internet-based client/server networking programming

(Deitel and Deitel(1998))

3.1.2 JAVA for Statistical Software Development

As the internet becomes the most common and most important route for information exchange, the explosive need for JAVA softwares in most fields is not surprising. JAVA provides classfiles for software implementation almost as much as C++ does while having unique advantages as discussed in above section. Having started from statistical education and simple descriptive graphics, many statistical JAVA softwares have been developed during the short period of last two years. JAVA can be used as an universal tool for developing and exchanging statistical techniques due to its portability and GUI are much easier to program and port, which is encouraging the implementation of dynamic/interactive visualization packages especially. Currently, WebStat(URL: http://www.stat.sc.edu/west/webstat/) and Statlets(URL:
http://www.statlets.com) are two major softwares entirely in JAVA for general statistical analysis methods. S-Plus and XploRe are building the interface between JAVA and their software engines so that JAVA applications can use their pre-designed analyses (URL: http://cm.bell-labs.com/cm/ms/who/cocteau/comsci/abs/sannella/abs.html (S-Plus and JAVA), http://www.xplore-stat.de/WWWJava/x4java.html (XploRe and JAVA)).

While all these softwares are providing JAVA applets for some statistical graphics, there is no JAVA software which can implement extensive dynamic and interactive visualization techniques for high-dimensional data. Considering that grand tour and projection pursuit are becoming important tools for exploratory structure findings in high-dimensional data, JAVA visualization software which can implement these should be developed. This software should be able to provide what XGobi has been doing to UNIX users (see Section 2.2.2) regardless of platforms (though some limitations can be applied). With these purposes, we developed and now introduce the first JAVA visualization software that can implement grand tour as well as other functions.

3.1.3 Implementation

This software is written entirely in JAVA using Java Development Kit (JDK) 1.1.4 class libraries, thus it can be run on any platform where 1.1.4 or higher version of JDK1.1 and the Java virtual machine (Java Runtime Environment (JRE)) exist. JDK and JRE can be down-loaded at no cost from Sun Microsystem's JAVA site (http://java.sun.com/cgi-bin/java-ports.cgi). This software was tested on the following three different machines and OSs: SGI Irix 6.2, Windows NT and DEC VAX V4.0B. Technical details and classfile names for functions are described in APPENDIX. The maximum number of cases of the raw data file with which this software can be run is 1300 when the minimal spanning tree is constructed on-line, and at least 6000 when the minimal spanning tree is constructed off-line and read as another input file.
3.2 Use of Software on Flea-Beetle Data

Figure 3.1 displays the software. One window shows a single projection from the grand tour sequence which illuminates the three clusters, with the points overlaid with minimal spanning tree computed in the full 6-D variable space. The window at the bottom displays the graph of ordered MST edge weights. The 5 longest edges (47-51, 6-10, 41-38, 22-24, 10-4) have been brushed, so that they fade to gray. Points isolated by the corresponding peeling are colored differently so the 6 colors here (yellow, pink, red, green, orange, and magenta) represent clusters resulting from peeling. We notice that after deleting these 5 longest edges, three clusters have not been separated yet. Since single linkage cluster analysis defines the smallest distance between each element of groups as the between groups distance, the resulting 6 groups from peeling the 5 longest edges is the same as the result of single linkage clustering with 6 clusters. Thus we can see how single linkage method fails to find the right clusters.

Figure 3.2 is the picture after peeling 3 more longest edges. Now the trimmed edges are separating three clusters and identifying some outliers.

Figure 3.3 is the scene after making all the MST edges in Figure 3.2 invisible. Since the grand tour continued during the interval between taking two pictures, the projection in Figure 3.3 has slightly changed from that Figure 3.2.
Figure 3.1 Grand tour with minimal spanning tree with the longest 5 edges (47-51, 6-10, 41-38, 22-24, 10-4) peeled for flea-beetle data. Points isolated by the peeling are colored differently so the 5 colors represent clusters resulting from the peeling: Original colors are available in book form only.
Figure 3.2 Grand tour with minimal spanning tree with the longest 8 edges peeled for flea-beetle data. Points isolated by the peeling are colored differently so the 5 colors represent clusters resulting from the peeling: Original colors are available in book form only.
Figure 3.3 Grand tour without minimal spanning tree with the longest 8 edges peeled for flea-beetle data. Points isolated by the peeling are colored differently so the 5 colors represent clusters resulting from the peeling: Original colors are available in book form only.
4 INITIAL PARTITIONING USING MINIMAL SPANNING TREE

4.1 Structure Detection by Minimal Spanning Tree

4.1.1 Properties of Minimal Spanning Tree

The properties of minimal spanning tree which can be used in cluster detection are described below, analogous to those of the Simplicial Graph by Vincent et al (1976) and Hoffman and Jain (1983). The Simplicial Graph is the graph which links every neighborhood point pairs in the Vornoi diagram. The Vornoi diagram is the total set of the Vornoi polygons (V(i)) of each point p_i (1 ≤ i ≤ n, n is the data size) in data, which is a convex polygon having the property that p_i is the closest of the all p_i's in data to any x∈V(i). Neighborhood point pair is a pair of two points of which Vornoi polygons have a common boundary. (Different definition of neighborhood is used in Section 3.3) Minimal spanning tree is analogous to Simplicial graph which has only n-1 edges while having paths between every point pairs. Vincent et al (1976) and Hoffman and Jain (1983) compare the first three of following distributions of data to those of null data which is random/normal (Hoffman and Jain), or Poisson (Vincent et al) to test the existence of clusters.

[a] edge weight distribution:

[a.1] If the data of size n is from 2-dimensional uniform distribution in area A, the expected weight of a randomly chosen MST edge is asymptotically propor-
tional to $\sqrt{\frac{2\pi}{n-1}}$(Hoffman and Jain(1983)).

[a.2] If the data is from a 2-dimensional Poisson process pattern, the distribution of the edge weight ($l$) is,

$$\psi(l) = \frac{\pi \rho l}{3} (l + \sqrt{\rho} \exp\left(-\frac{\pi \rho l^2}{4}\right)), \quad (4.1)$$

where $\rho$ is the total density (number of points/volume of region)(Vincent et al, 1976).

[b] interior angle distribution: If the data is from a 2-dimensional Poisson process pattern, the distribution of the angle ($\theta$) between two incident MST edges is,

$$\omega(\theta) = \frac{4}{3\pi} \sin \theta [\sin \theta + (\pi - \theta) \cos \theta]. \quad (4.2)$$

c] maximum, l-th largest edge weight: Section 4.1.2

d] node degree distribution(Zahn, 1975)

Zahn(1975) found out different properties of MST can be successfully used for solving different cluster problems, though they need previous human perception of data. His idea is based on the fact that our perception system automatically seeks the most economical encoding of the data, i.e. clusters. His strategies for finding structures in different problems are as follows.

[a] separate clusters: peeling inconsistent edges

[b] structure reconstruction of composite clusters: node depth histogram of fragments after peeling

[c] separate noise from line-type cluster: node degree comparison

[d] separate touching clusters: local minimum of node depth histogram
[e] separate touching Gaussian clusters: minimum edge weight between two local
maximum in diameter path

The inconsistency of any edge is defined as significant largeness of its weight compared
to the average of nearby edges on both sides. The criteria for inconsistency are the

[i] size of neighborhood: the depth from both nodes of the edge within which the
nodes are defined as neighborhood

[ii] number of standard deviation of neighborhood edge weights by which the edge
weight should exceed to be defined as inconsistent

[iii] factor: the ratio between the edge weight and the average weight of neighborhood
edges

Here we notice that he consistently uses the local comparison of MST properties
for separating clusters in all problem types. We adopt his local comparison in our
peeling stage, since the main purpose of the peeling process is to separate well-separated
clusters. As discussed in Section 2.3.1, another criterion for peeling in Posse's method is
the comparison between distribution of ordered MST edge weights from data and that of
one standard normal cluster. The comparison is based on the asymptotic distribution of
the largest MST edge weight of one standard normal point cloud by Penrose(1998). In
the next two sections, we review this and show its slow convergence rate from simulation
studies.

4.1.2 The Asymptotic Distribution of Ordered MST Edge Weight

The asymptotic distribution of maximum edge weight(Euclidean distance between
two ending nodes) was initially used to detect outliers in multivariate normal data by
Rohlf(1975). But Rohlf's examination of squared standardized maximum edge weight
to gamma quantile plot was recently criticized as being inaccurate by the simulation
study done by Caroni and Prescott(1995). The most recent findings on the minimal spanning tree edge weight were done by Penrose(1998). He proved that the maximum minimal spanning tree edge weight of one standard normal point cloud converges weakly to Gumbel distribution in the following theorem.

His proof is based on the fact that the maximum minimal spanning tree edge weight of one standard normal point cloud is the infimum of those $r$ such that the union of the Euclidean balls of radius $r/2$ centered at the points of the normal cloud is connected.

**Theorem 1** (Penrose, 1998) Let $x_i, i=1,2,3,\ldots$ be independent $\nu(\geq 1)$ dimensional random vectors with the standard multivariate normal density function. Define the point process $\mathcal{X}_n = \{x_1, x_2, \ldots, x_n\}$ and $\Lambda(\beta) = \exp(-\beta)$, the Gumbel distribution function for $\beta \in \mathbb{R}^1$, with the density $= \exp(-\exp(-\beta)) \exp(-\beta)$. Let $M_n$ denote the maximum edge weight (Euclidean distance between two ending nodes of the edge) for $\mathcal{X}_n$.

Then,

$$\lim_{n \to \infty} P[(a_n M_n - b_n) \leq \beta] = \Lambda(\beta), \quad (4.3)$$

$$a_n = \sqrt{2 \log n},$$

$$b_n = (\nu - 1) \log_2 n - \left(\frac{\nu - 1}{2}\right) \log_3 n - \log(k_\nu)$$

where

$$k_\nu = 2^{-\frac{\nu}{2}} (2\pi)^{-\frac{\nu}{2}} \frac{\Gamma(\nu/2)(\nu - 1)^{\nu+1}}{\nu^2}$$

$$\log_2 n = \log (\log n)$$

$$\log_3 n = \log (\log_2 n)$$

Any random variable $Y_n$ from Gumbel distribution has mean and variance as

$$E(Y_n) = \gamma, V(Y_n) = \frac{\pi^2}{6}, \quad (4.4)$$

from its m.g.f. $\Gamma(1 - t)$, where $\gamma = 0.5772156649 \ldots$ is the Euler's constant (Gumbel, 1957). The derivation of (4.4) is in APPENDIX.
Thus from (4.3) and (4.4),
\[ E(M_n) = \frac{\gamma + b_n}{a_n} = \frac{\gamma + (\nu - 1) \log_2 n - \left(\frac{\nu - 1}{2}\right) \log_3 n - \log(k_\nu)}{\sqrt{2 \log n}} \] (4.5)
\[ V(M_n) = \frac{\frac{1}{2} \pi^2}{\log n} \] (4.6)

Posse (1998) generalized Theorem 1 to the \( l \)-th longest MST edge weight to lead the following corollary.

**Corollary 1 (Posse, 1998)** Under the same conditions of Theorem 1, define \( M_{nl} \) as the \( l \)-th largest edge weight for \( X_n \).
Then,
\[ \lim_{n \to \infty} P[(a_n M_{nl} - b_n) \leq \beta] = \Lambda_l(\beta), \] (4.7)
where \( \Lambda_l(\beta) = \exp(-\beta)(1 + \sum_{j=1}^{l-1} \frac{\exp(-j\beta)}{\Gamma(j+1)}) \) is the \( l \)-th order Gumbel distribution function with the density \( \exp(-\exp(-\beta)) \frac{\exp(-\beta)}{\Gamma(1)} \) and \( a_n \) and \( b_n \) are same as in Theorem 1.

Using this corollary, we obtain the mean and the variance of the random variable, \( Y_{n2} \) which follows 2nd order Gumbel distribution with mean and variance
\[ E(Y_{n2}) = \gamma - 1, V(Y_{n2}) = \frac{\pi^2}{6} - 1 \] (4.8)

The derivation of (4.8) is in APPENDIX.

Thus for the second largest MST edge weight from one standard normal cluster with size \( n \), \( M_{n2} \) is,
\[ E(M_{n2}) = \frac{\gamma - 1 + b_n}{a_n} = \frac{\gamma + (\nu - 1) \log_2 n - \left(\frac{\nu - 1}{2}\right) \log_3 n - \log(k_\nu) - 1}{\sqrt{2 \log n}} \] (4.9)
\[ V(M_{n2}) = \frac{1}{2 \log n} \left( \frac{\pi^2}{6} - 1 \right) \] (4.10)

In next section, we will show these asymptotic results converge too slow for \( l = 1 \) (i.e. Theorem 1) and \( l = 2 \).

4.2 Simulation Study about Penrose's Theorem

Table 4.1 shows the comparison of averages of largest MST edge weights of 100 simulated normal data sets and the computed largest edge weight with their variances from (4.5). Different sets of 100 standard normal data were generated for each dimension (2, 3, 5, and 10) and each data size (50, 100, 200, \( \cdots \), 5000). Table 4.1 also includes the upper limit of 95% confidence interval for the mean largest edge weight of 100 standard normal data sets. That is, the fourth row in each cell in Table 4.1 was computed from 
\[ E(M_n) + \frac{\sqrt{V(M_n)}}{100} t(1 - 0.025; 100 - 1). \]

The * with average in the first row represents a significant difference from the asymptotic mean by Theorem 1, at 0.05 level. We notice that except the two dimensional case, in most cases the average maximum edge weights are greater than the upper limit of 95% confidence interval of asymptotic mean from Penrose's theorem. The difference between the average MST edge weight and the upper limit of confidence interval is becoming bigger as dimension increases.

Table 4.2 contains the averages of the largest minimal spanning tree edge weights of 100 simulated data sets each of which consists of two data sets from two 3-dimensional normal distributions, with means apart from 1, 2, and 5 respectively and identity covariance matrices. The second row in each cell is the upper limit of 95% confidence interval from the second column (3-dimension) in Table 4.1. As expected, all the average weights are bigger than those in the column of dimension 3 in Table 4.1, but not drastically so. For instance, in the case of 100 data sets with size 1000, the averages of maximum MST
edge weights of one standard normal cluster (2nd column and 6th row in Table 4.1) and two normal clusters with means 2 apart from each other (1st column and 6th row in Table 4.2) are 1.5240 and 1.5831, both of which fall beyond the 95% upper limit 1.4594. That is, they are significantly larger than asymptotic mean, 1.3909. But the difference between these two averages is 0.0591, which is less than the difference between the mean of 100 maximum edge weight from one standard normal and the upper limit of 95% confidence interval for theoretical mean, 0.0646 = 1.5240 − 1.4594. This shows that we cannot test the existence of two clusters by the longest minimal spanning tree edge weight, nor can we suspect that from the numbers themselves. All the average longest edge weights of 100 data sets from one standard normal 3-dimensional distribution (2nd column of Table 4.1) are larger than the upper 95% limit of asymptotic mean from Gumbel distribution for each size. Only the averages from the 2-dimensional standard normal are mostly within the ranges.

Table 4.3 shows the result of similar simulation study, but about 2nd largest MST edge weights. We notice that the convergence rate of 2nd largest MST edge weight to 2nd-order Gumbel distribution is slower than maximum (Table 4.1). All averages of 100 2nd largest MST edge weights from one standard normal cluster for each data and each dimension are larger than their theoretical upper limit of 95% confidence intervals from (4.9).

From these simulation results above, we find that the slow convergence of asymptotic Gumbel distribution of MST edge weights from one normal cluster makes the comparison between two (MST edge weights from data and from one normal cluster) difficult to be used to test the existence of more than one cluster with different means in data set of the size up to 5000. Rather than comparing the two distributions, we suggest to choose the edges to be peeled in exploratory method. In the next two sections, we introduce the partitioning strategy which uses local comparison of edge weights. The partitioning is performed in two stages like Posse's. First, the "long" edges are peeled.
(Here the length of an edge means the Euclidean distance between two ending nodes.)

To decide whether any edge is long or not, the weight of that edge is compared to the average weight of neighborhood edges, not to all the edges in minimal spanning tree. Second, the information about peeled edges and the connections of nodes in minimal spanning tree are used as input for pruning. The nodes are grouped into partitions of approximately same size.

4.3 Initial Partitioning: Peeling (Ours)

As discussed in Section 4.1.1, Zahn (1971) defined the inconsistency of any minimal spanning tree edge, denoted by XY where X and Y are two nodes at each end, as whether the weight (usually Euclidean distance between two end nodes) of the edge is significantly larger than the average of nearby edge weights on both sides of XY or not. As a way to measure the inconsistency of an edge, he suggested seeing how many standard deviations separate the weight of edge XY from the average edge weights on each side, and how big the ratio between the edge weight and the average is. For doing this, the following three were defined and possible values for them were proposed by Zahn.

1) the size of neighborhood explored for each node: 2
2) the multiples of standard deviation, \( \sigma_T \): 2
3) the "factor", \( f_T = \frac{\text{edge weight}}{\text{the average of other nearby edge weights}} \): 2

If the weight of edge XY exceeds the average local edge weight on each end (node X and Y) by \( \sigma_T \) of the respective standard deviation and/or the factor exceeds \( f_T \), the edge is defined as inconsistent. Zahn proposed 2 for the values of \( \sigma_T \), \( f_T \), and the size of neighborhood visited ("weight" of the path). (The weight of a path is the number of edges it contains. This is different from the weight of edge. A path between two prescribed nodes is an alternating sequence of nodes and edges with the prescribed nodes
Table 4.1 Average largest edge weights of MSTs of 100 simulated standard normal data sets

1st row: Average weight
2nd row: Computed $E(M_n)$ using (4.5)
3rd row: Computed $V(M_n)$ using (4.5)
4th row: $E(M_n) + \sqrt{\frac{V(M_n)}{100}} t(1-0.05/2; 100 - 1)$
* : Average weight is within the upper limit of 95% C.I.

<table>
<thead>
<tr>
<th>n (sample size)</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1.2759*</td>
<td>1.8287</td>
<td>2.5171</td>
<td>3.8635</td>
</tr>
<tr>
<td></td>
<td>1.2149</td>
<td>1.5663</td>
<td>1.7901</td>
<td>0.9923</td>
</tr>
<tr>
<td></td>
<td>0.2102</td>
<td>0.2102</td>
<td>0.2102</td>
<td>0.2102</td>
</tr>
<tr>
<td></td>
<td>1.3059</td>
<td>1.6573</td>
<td>1.8811</td>
<td>1.0833</td>
</tr>
<tr>
<td>100</td>
<td>1.2079*</td>
<td>1.7154</td>
<td>2.4244</td>
<td>3.7882</td>
</tr>
<tr>
<td></td>
<td>1.1548</td>
<td>1.5139</td>
<td>1.7904</td>
<td>1.2309</td>
</tr>
<tr>
<td></td>
<td>0.1786</td>
<td>0.1786</td>
<td>0.1786</td>
<td>0.1786</td>
</tr>
<tr>
<td></td>
<td>1.2387</td>
<td>1.5978</td>
<td>1.8743</td>
<td>1.3148</td>
</tr>
<tr>
<td>200</td>
<td>1.1744*</td>
<td>1.6001</td>
<td>2.3609</td>
<td>3.6776</td>
</tr>
<tr>
<td></td>
<td>1.1062</td>
<td>1.4706</td>
<td>1.7875</td>
<td>1.4138</td>
</tr>
<tr>
<td></td>
<td>0.1552</td>
<td>0.1552</td>
<td>0.1552</td>
<td>0.1552</td>
</tr>
<tr>
<td></td>
<td>1.1844</td>
<td>1.5488</td>
<td>1.8657</td>
<td>1.4920</td>
</tr>
<tr>
<td>300</td>
<td>1.1502*</td>
<td>1.6359</td>
<td>2.2968</td>
<td>3.6790</td>
</tr>
<tr>
<td></td>
<td>1.0816</td>
<td>1.4482</td>
<td>1.7845</td>
<td>1.5014</td>
</tr>
<tr>
<td></td>
<td>0.1442</td>
<td>0.1442</td>
<td>0.1442</td>
<td>0.1442</td>
</tr>
<tr>
<td></td>
<td>1.1569</td>
<td>1.5235</td>
<td>1.8598</td>
<td>1.5767</td>
</tr>
<tr>
<td>500</td>
<td>1.1439</td>
<td>1.5827</td>
<td>2.3000</td>
<td>3.6123</td>
</tr>
<tr>
<td></td>
<td>1.0537</td>
<td>1.4224</td>
<td>1.7796</td>
<td>1.5960</td>
</tr>
<tr>
<td></td>
<td>0.1323</td>
<td>0.1323</td>
<td>0.1323</td>
<td>0.1323</td>
</tr>
<tr>
<td></td>
<td>1.1259</td>
<td>1.4946</td>
<td>1.8518</td>
<td>1.6682</td>
</tr>
<tr>
<td>1000</td>
<td>1.0338*</td>
<td>1.5240</td>
<td>2.2667</td>
<td>3.5544</td>
</tr>
<tr>
<td></td>
<td>1.0203</td>
<td>1.3909</td>
<td>1.7715</td>
<td>1.7017</td>
</tr>
<tr>
<td></td>
<td>0.1191</td>
<td>0.1191</td>
<td>0.1191</td>
<td>0.1191</td>
</tr>
<tr>
<td></td>
<td>1.0888</td>
<td>1.4594</td>
<td>1.8400</td>
<td>1.7702</td>
</tr>
<tr>
<td>5000</td>
<td>0.9944*</td>
<td>1.4139</td>
<td>2.0785</td>
<td>3.4724</td>
</tr>
<tr>
<td></td>
<td>0.9572</td>
<td>1.3292</td>
<td>1.7485</td>
<td>1.8770</td>
</tr>
<tr>
<td></td>
<td>0.0966</td>
<td>0.0966</td>
<td>0.0966</td>
<td>0.0966</td>
</tr>
<tr>
<td></td>
<td>1.0189</td>
<td>1.3909</td>
<td>1.8102</td>
<td>1.9387</td>
</tr>
</tbody>
</table>
Table 4.2  Average largest edge weights of MSTs of 100 simulated 3-D data sets of two normal clusters with different means

1st row: Average weight
2nd row: $E(M_n) + \sqrt{\frac{V(M_n)}{100}} t(1 - 0.05/2; 100 - 1)$

<table>
<thead>
<tr>
<th>Size</th>
<th>Distance between two means</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>50</td>
<td>1.9525</td>
</tr>
<tr>
<td></td>
<td>1.6573</td>
</tr>
<tr>
<td>100</td>
<td>1.8085</td>
</tr>
<tr>
<td></td>
<td>1.5978</td>
</tr>
<tr>
<td>200</td>
<td>1.7918</td>
</tr>
<tr>
<td></td>
<td>1.5488</td>
</tr>
<tr>
<td>300</td>
<td>1.7325</td>
</tr>
<tr>
<td></td>
<td>1.5235</td>
</tr>
<tr>
<td>500</td>
<td>1.6691</td>
</tr>
<tr>
<td></td>
<td>1.4946</td>
</tr>
<tr>
<td>1000</td>
<td>1.5831</td>
</tr>
<tr>
<td></td>
<td>1.4594</td>
</tr>
<tr>
<td>5000</td>
<td>1.4999</td>
</tr>
<tr>
<td></td>
<td>1.3909</td>
</tr>
</tbody>
</table>
Table 4.3  Average second longest edge weights of minimal spanning trees of 100 simulated standard normal data sets

1st row : Average weight
2nd row : Computed $E(M_{n2})$ using (4.9)
3rd row : Computed $V(M_{n2})$ using (4.9)
4th row : $E(M_{n2}) + \sqrt{V(M_{n2})/100} t(1 - 0.05/2; 100 - 1)

<table>
<thead>
<tr>
<th>n</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1.0155</td>
<td>1.4761</td>
<td>2.2230</td>
<td>3.6324</td>
</tr>
<tr>
<td></td>
<td>0.8573</td>
<td>1.2088</td>
<td>1.4325</td>
<td>0.6348</td>
</tr>
<tr>
<td></td>
<td>0.0824</td>
<td>0.0824</td>
<td>0.0824</td>
<td>0.0824</td>
</tr>
<tr>
<td></td>
<td>0.9143</td>
<td>1.2658</td>
<td>1.4895</td>
<td>0.6918</td>
</tr>
<tr>
<td>100</td>
<td>0.9623</td>
<td>1.4565</td>
<td>2.1703</td>
<td>3.5262</td>
</tr>
<tr>
<td></td>
<td>0.8253</td>
<td>1.1844</td>
<td>1.4609</td>
<td>0.9014</td>
</tr>
<tr>
<td></td>
<td>0.0700</td>
<td>0.0700</td>
<td>0.0700</td>
<td>0.0700</td>
</tr>
<tr>
<td></td>
<td>0.8778</td>
<td>1.2369</td>
<td>1.5134</td>
<td>0.9539</td>
</tr>
<tr>
<td>200</td>
<td>0.9092</td>
<td>1.3928</td>
<td>2.1095</td>
<td>3.4553</td>
</tr>
<tr>
<td></td>
<td>0.7990</td>
<td>1.1634</td>
<td>1.4803</td>
<td>1.1066</td>
</tr>
<tr>
<td></td>
<td>0.0609</td>
<td>0.0609</td>
<td>0.0609</td>
<td>0.0609</td>
</tr>
<tr>
<td></td>
<td>0.8480</td>
<td>1.2123</td>
<td>1.5293</td>
<td>1.1555</td>
</tr>
<tr>
<td>300</td>
<td>0.9083</td>
<td>1.3577</td>
<td>2.1110</td>
<td>3.4515</td>
</tr>
<tr>
<td></td>
<td>0.7855</td>
<td>1.1521</td>
<td>1.4884</td>
<td>1.2054</td>
</tr>
<tr>
<td></td>
<td>0.0565</td>
<td>0.0565</td>
<td>0.0565</td>
<td>0.0565</td>
</tr>
<tr>
<td></td>
<td>0.8327</td>
<td>1.1993</td>
<td>1.5356</td>
<td>1.2525</td>
</tr>
<tr>
<td>500</td>
<td>0.8647</td>
<td>1.3418</td>
<td>2.1021</td>
<td>3.4036</td>
</tr>
<tr>
<td></td>
<td>0.7701</td>
<td>1.1388</td>
<td>1.4960</td>
<td>1.3123</td>
</tr>
<tr>
<td></td>
<td>0.0519</td>
<td>0.0519</td>
<td>0.0519</td>
<td>0.0519</td>
</tr>
<tr>
<td></td>
<td>0.8153</td>
<td>1.1840</td>
<td>1.5412</td>
<td>1.3575</td>
</tr>
<tr>
<td>1000</td>
<td>0.8564</td>
<td>1.2765</td>
<td>2.0549</td>
<td>3.3363</td>
</tr>
<tr>
<td></td>
<td>0.7513</td>
<td>1.1219</td>
<td>1.5025</td>
<td>1.4327</td>
</tr>
<tr>
<td></td>
<td>0.0467</td>
<td>0.0467</td>
<td>0.0467</td>
<td>0.0467</td>
</tr>
<tr>
<td></td>
<td>0.7942</td>
<td>1.1647</td>
<td>1.5453</td>
<td>1.4755</td>
</tr>
<tr>
<td>5000</td>
<td>0.7953</td>
<td>1.2236</td>
<td>1.9299</td>
<td>3.2595</td>
</tr>
<tr>
<td></td>
<td>0.7149</td>
<td>1.0869</td>
<td>1.5062</td>
<td>1.6348</td>
</tr>
<tr>
<td></td>
<td>0.0379</td>
<td>0.0379</td>
<td>0.0379</td>
<td>0.0379</td>
</tr>
<tr>
<td></td>
<td>0.7535</td>
<td>1.1255</td>
<td>1.5448</td>
<td>1.6734</td>
</tr>
</tbody>
</table>
as first and last elements, all other nodes distinct, and each edge linking the two nodes adjacent to it in the sequence (Friedman and Rafsky, 1979). We use a increased value for the neighborhood size (3 instead of 2) and reduce $\sigma_T$ to 1.5 ($f_T$ is not used in our peeling algorithm). As the size of neighborhood used above gets smaller, the decision gets more local since we examine the edges within a narrower neighborhood. By increasing the neighborhood size slightly, we try to avoid the case where an edge is decided as inconsistent from local data fluctuation and not from separating clusters. Also, we find that it is difficult for the $\sigma_T$ value of any MST edge of real data to exceed 2 even when the edge is truly separating clusters. By increasing the neighborhood size and decreasing $\sigma_T$ slightly, we should be able to peel the MST edges, which are actually connecting clusters, though their weights are not extremely larger than nearby edges.

The separated data from peeling MST edges using this algorithm shows different results from the single linkage clusters which are same as the resulting clusters from Posse's peeling algorithm. This is because our algorithm doesn't peel the MST edges in the descending order of edge weights as Posse's. Thus to show the result of clustering from peeling only before pruning, we need a different algorithm for traversing nodes following MST and labeling them according to whether each MST edge has been peeled or not. The algorithm for this is explained in APPENDIX. Other MST traversing method was proposed by Friedman and Rafsky (1979). This is for ranking multivariate data points using MST. In this algorithm, the node with the largest eccentricity is designated as the root. The eccentricity of a node in a tree is the number of edges in a path with the largest length beginning with that node (length of a path is the number of edges in it). Each node is associated with its depth, which is defined as the length of the path between it and the root. The height of the MST is defined as the maximum depth of any node in it. The nodes are then visited and ranked in a height directed pre-order (HDP) traversal of the tree. This traversal is defined recursively as follows.

(i) visit the root
(ii) traverse in ascending order of height of the subtrees rooted at the daughters of the root. (The daughters of a node are those nodes that are not its parent but are linked to it. The parent of a given node is the penultimate node encountered on the path from the root to the given node.) (Friedman and Rafsky, 1979)

4.4 Initial Partitioning: Pruning

4.4.1 Posse’s Algorithm

Posse(1998) developed the algorithm for grouping the nodes into many small partitions that are roughly same sized. His algorithm for traversing the points and partitioning them is simply following the A array from the output of Prim’s minimal spanning tree construction (Prim, 1957). If we construct the minimal spanning tree of data with size n, this algorithm produces an array of size n-1, such that its i-th element is the number of the node(observations) which is connected to the i-th observation in the minimal spanning tree. Thus if we start at i=1 and repeat the following steps until all the nodes are traversed.

Step1: j ← i

Step2: If A(j) is not a dead end, then j ← A(j); repeat Step2

else if i ← i+1; goto Step1

Following these steps, the nodes are visited and partitioned until the size of current component gets bigger than pre-determined limit. If the size becomes larger than this, a new component is created and the following nodes are partitioned into this new component. Also in the case that the edge (j, A(j)) has been peeled, a new component is created. Since MST consists of n-1 edges and thereby A array has n-1 elements, we must have a dead end. For instance if A array for a data of size 100 has 99 elements from A(1) to A(99), we would meet a dead end at k such that A(i) equals to 100. Whenever this happens, his algorithm would go back to the next observation from the original i; while
doing so, it should pass through many already visited nodes. Also Posse’s algorithm
tends strongly to assign the nodes which are connected with MST edges, in the sequen­
tial order. For any node j with a degree (the number of nodes incident on it) bigger than
one, the edge in MST connecting j and A(j) is just one of edges which are incident to
this node j. Thus, if we group j and the nodes in the above traverse order sequentially
we may assign the other node, say k, incident to j into different partition from j, even
when the length of the path between nodes j and k is just 1 while the nodes with lengths
larger than 1 are partitioned to same partition component as j. This can cause a bigger
problem when the data has a small core in the middle and sparse density in other areas.

Let’s suppose any node j in the core of data like this has large degrees, that is, there are
many nodes with small lengths incident to this node j and most of the weights of edges
between the node j and these nodes would be short. But, all these nodes except one have
the risk of being assigned to a different partition component from the node j. This can
be more dangerous when we partition the nodes into large components. On the contrary
since his peeling algorithm disconnects the edges with globally large weights, the nodes
in a same large and sparse cluster may be assigned to different partition components or
the component of only the node itself. This may cause a smaller problem than the above
one, since the partition components with each point themselves are to be clustered more
accurately in the real clustering process. But in this case we lose the opportunity to
save computational requirements.

4.4.2 Our Algorithm

We suggest a different pruning algorithm from Posse’s. This algorithm follows MST
dges similarly to Posse’s algorithm. But while his algorithm visits the nodes simply
following A array (i→A(i) starting from i=1), in our algorithm all the nodes incident
to the current node are visited and partitioned first in ascending order of the weight
of the edge connecting the two nodes. Unless the edge between the current node and

the incident node has been peeled previously in the peeling stage, the incident node is
assigned to same partition as the current node as long as the size of partition component
is smaller than pre-determined maximum size. If the size of partition component becomes
larger than the limit, a new component is created and the next incident nodes are
assigned to this partition. The nodes visited and partitioned are stored in a separate
array (S array in next page) in the order of visits. After all the incident nodes have been
visited and partitioned, the algorithm looks for the most recently stored node in this
array among those which have any other unvisited incident nodes. This node becomes
new "current node" and the partitioning steps above are repeated with this. For running
this algorithm, we need the following arrays in addition to A array:

[a] $F(i) = 0$ if the edge $(i,A(i))$ has been peeled
    $= 1$ otherwise
    $(1 \leq i \leq n - 1)$
    This is produced from peeling stage.

[b] $D(i) =$ the degree of node $i$, i.e. the number of nodes incident to $i$
    $(1 \leq i \leq n)$

[c] $TD$ array : $TD(i,j) =$ the node which is incident to $i$ and $j$-th closest among such
    nodes. For example, if node 3 has 2 incident nodes, 5 and 7 in MST and node 5 is
    closer to node 3 than node 7 is, $TD(3,1) = 5$ and $TD(3,2) = 7$ ($D(3) = 2$). ($1 \leq i \leq n,$
    $1 \leq j \leq D(i)$ for each $i$)

[e] $S$ array : $S(k) =$ the $k$-th visited and labeled node $(1 \leq k \leq n)$.

Starting from $i=1$ and $k=1$, our pruning algorithm proceeds as follows;

Step 1: The node $i$ with $D(i) = J$, say, is visited and labeled as $k$ (that is, partitioned into
component $k$) (The $J$ nodes incident to node $i$ ($TD_{ij}, 1 \leq j \leq J$) are in ascending
order of their weights from the node $i$).
Step 2: Visit the nodes from $TD_{i1}$ to $TD_{iJ}$ and partition them into the same component to which the node $i$ belongs, unless the edge is peeled or the component size becomes larger than maximum, similarly as in Posse’s algorithm. The nodes visited are saved in $S$ array.

Step 3: The latest node in $S$ array which has unvisited incident nodes becomes new current $i$.

Step 4: Go to Step 2.

The flow chart for these steps is found in APPENDIX.

### 4.4.3 Comparison Between Two Pruning Methods

The comparison of above two pruning algorithms is explained more clearly in Table 4.4, 4.5 and Figure 4.1. These are extracted from the MST of 2-D projection of sphered flea-beetle data (see Section 6.2) and corresponding arrays in Section 4.4.2. Table 4.4 includes the parts of $TD$, $A$ and $S$ array describing the MST in Figure 4.1. The first partition component is created using each Posse’s algorithm and our algorithm with maximum size as 4. Posse’s pruning traverses the nodes of MST in the order of 1, 8, 15 and 21, and these 4 nodes make one component of size 4. Our pruning algorithm visits the nodes in the order of 1, 3, 8 and 13, and one component is created with these 4 nodes. After that, the next $i$ should be 13 from $S$ array, since it has an unvisited incident node (19) and has been stored more recently than node 8, which also has an unvisited incident node (15). From Figure 4.1, we notice that the shape of Posse’s first component is long while the shape of ours is rounder.

Figure 4.2 is showing the result of peeling step for 2-D projection of sphered flea beetle data found by the projection pursuit algorithm in Section 5.4. Three different glyphs (filled circle, open rectangle and open circle) represent three different clusters where each data points actually belong. Since this data set is small, the suggested
Table 4.4 TD and A arrays for our pruning algorithms: 2-D projection of flea-beetle data

<table>
<thead>
<tr>
<th>i</th>
<th>TD(i,j)</th>
<th>A(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>21</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>74</td>
</tr>
<tr>
<td>17</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 4.5 TD and A arrays for our pruning algorithms: 2-D projection of flea-beetle data

<table>
<thead>
<tr>
<th>k</th>
<th>S(k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>S(k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>S(k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
</tr>
</tbody>
</table>
values for the neighborhood size and factor of 3 and 2 are too large. We used the size of neighborhood and the value of factor as 2 and 1.5 respectively. As the result of the peeling step, two edges which are connecting (24,6) and (46,54), were disconnected. Peeling step separated the clusters of filled circle and open rectangle, but failed to do so with the clusters of open rectangle and open circle. The value of factor for the edge (16,74) which is connecting these two clusters is 1.45, close to 1.5. If these two nodes, 16 and 74, are grouped into same partition in pruning step, one of them should be clustered incorrectly in real cluster analysis. If they are assigned into different partitions, they will have another chance to be clustered correctly. The peeled edge (46,54) which is not separating real clusters won't cause any harm in real cluster analysis, though it won't help it either. It groups the two nodes 46 and 54 into different partitions, but both of the partitions will be reconsidered in real cluster analysis.

Figure 4.2 and Figure 4.3 are showing the results of two partitioning algorithms, Posse's and ours. Table (4.6) is about the distribution of sizes of partition components in two results. We notice that Posse's algorithm produces more singleton partitions than
Figure 4.2 Result of peeling for 2-D projection of flea-beetle data

ours. Also, it tends to produce linear shape partitions while ours produces rounder ones. Considering that the pruning has been performed after the peeling step and the aim of pruning is to group close points together after unconvincing points have been isolated, the production of too many singleton partitions is undesirable. Also, the points in a linearly shaped partition component tend to be farther apart from each other than the points in a rounder shaped one, if they are of the same size. For the two partitions which contains node 56 in both methods, the maximum distances between any point pairs in the partition are 0.551734 (node 44 and node 48) for Posse's and 0.29746 (node 71 and node 48).
Table 4.6  Comparison of distributions of size of partitions between Posse's and our pruning algorithms

<table>
<thead>
<tr>
<th>Size of Partition</th>
<th>Total Number of Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Ours</td>
<td>10</td>
</tr>
<tr>
<td>Posse's</td>
<td>25</td>
</tr>
</tbody>
</table>

Figure 4.3  Partitioning for flea-Beetle data using Posse's algorithm
Figure 4.4  Partitioning for flea-beetle data using our algorithm
5 DIMENSION REDUCTION USING PROJECTION PURSUIT

5.1 Holes Index

Projection pursuit indices are measures of the departure from normality of the projected data by a result by Diaconis and Freedman(1984). Generally, this involves estimating the density of the projected data, which is often done using kernel density estimation. This produces indices which are slow to compute, hence they are ill-suited to interactive situations (Klinke and Cook(1995) explain how to reduce the computational time by using binning in the kernel density estimation). An alternative approach is to use a function expansion to approximate the density, and then estimate the integral of the result using a sample mean. Examples of this approach are the Legendre index(Friedman, 1987), Hermite index(Hall, 1989), and the Natural Hermite index(Cook, Buja, and Cabrera, 1993). While proposing the Natural Hermite Index Cook et al(1993) found that separating off the low order terms in the function expansion provided indices that were honed to find particular types of structured projections. One such index is called the Holes index, which targets projections with an absence of data points in the center. It is computed from the negative square root of a first order Hermite index, that is, from the Hermite index where the series was truncated at the first term:

\[ I_{Holes} = -(a_0 - b_0), \]  

(5.1)
where

\[ a_0 = a_0(F) = \int_{\mathbb{R}^d} \phi_d(X) dF(X) = \frac{1}{(\sqrt{2\pi})^d} \int_{\mathbb{R}^d} \exp\left(-\frac{1}{2}(X_1^2 + \cdots + X_d^2)\right) dF(X), \]  

(5.2)

where \( X = (X_1, \cdots, X_d)^T \), \( \phi_d \) is the d-dimensional standard normal density, and \( F(X) \in \mathcal{F} = \{ F(X) : E_F(X) = 0_d, V_F(X) = I_d \} \), \( 0_d = (0, \cdots, 0)^T \), \( I_d \) is the d-dimensional identity matrix.

Also

\[ b_0 = \int_{\mathbb{R}^d} \phi_d(X) d\phi_d(X) \]

\[ = \frac{1}{(\sqrt{2\pi})^d} \prod_{j=1}^{d} \phi_j(-\frac{1}{2}X_j^2) \]

\[ = \frac{1}{(2\pi)^{d/2}}(\sqrt{\pi})^d = (2\sqrt{\pi})^{-d}. \]

(5.4)

The coefficient \( a_0 \) is estimated using the method of moments as

\[ \hat{a}_0 = (2\pi)^{-d/2} \frac{1}{n} \sum_{i=1}^{n} \exp\left(-\frac{1}{2} \sum_{j=1}^{d} x_{ij}^2\right), \]

(5.5)

where sample \( x = \begin{pmatrix} x_{11} & \cdots & x_{1d} \\ x_{21} & \cdots & x_{2d} \\ \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nd} \end{pmatrix}_{nxd} \)

Thus the estimate for d-dimensional Holes index in (5.1) is

\[ \hat{J}_{\text{holes}} = -\frac{1}{n} \sum_{i=1}^{n} \exp\left(-\frac{1}{2} \sum_{j=1}^{d} x_{ij}^2\right) + (2\sqrt{\pi})^{-d}. \]  

(5.6)
5.2 Theoretical Maximum and Minimum

For each dimension, the Holes index has its theoretical maximum and minimum values. Cook, Buja and Cabrera (1993) showed that for a 1-dimensional case, $a_0$ has its minimum and maximum as

$$\min_{F \in \mathcal{F}} a_0 = (2\pi)^{-\frac{1}{2}}, \text{ where the density } f(x) = \begin{cases} 0.5 & : x = 1 \\ 0.5 & : x = -1 \end{cases}$$

and

$$\max_{F \in \mathcal{F}} a_0 = (2\pi)^{-\frac{1}{2}}, \text{ where } f(x) = \begin{cases} 1 & : x = 0 \\ 0 & : \text{o.w.} \end{cases}$$

Here $\mathcal{F}$ is the weak closure of $\mathcal{F}$ in (4.2) (Cook, Buja and Cabrera (1993)). Thus, the 1-dimensional $\max I_{\text{Holes}}$ and $\min I_{\text{Holes}}$ are

$$\max I_{\text{Holes}}^1 = -\min_{\mathcal{F}} a_0 + b_0 = -(2\pi)^{-1/2} \exp(-1/2) + (2\sqrt{\pi})^{-1},$$

$$\min I_{\text{Holes}}^1 = -\max_{\mathcal{F}} a_0 + b_0 = -(2\pi)^{-1/2} + (2\sqrt{\pi})^{-1}.$$ 

We generalize this to any d-dimensional projection by conjecturing that the maximum value is achieved by the distribution which place equal mass at the vertices of a d-dimensional cube with corners at (-1,1) in each dimension:

$$\max I_{\text{Holes}}^d = -(2\pi)^{-\frac{d}{2}} \exp(-\frac{d}{2}) + (4\pi)^{-\frac{d}{2}},$$

since $\sum_{i=1}^{d} X_i^2 = d$. This maximum is also achieved by the uniform distribution on a d-dimensional sphere with radius $\sqrt{d}$. The minimum value for d-dimension is achieved by the distribution of point mass at $0_d = (0,0,\ldots,0)^T$:

$$\min I_{\text{Holes}}^d = -(2\pi)^{-\frac{1}{2}} + (4\pi)^{-\frac{d}{2}}.$$ 

Table 5.1 shows the above maximum and minimum values of Holes index for each dimension from 1 to up to 10.
Table 5.1  Maximum and minimum Holes index values using (5.9) and (5.10)

<table>
<thead>
<tr>
<th>d</th>
<th>max $I^d_{Holes}$</th>
<th>min $I^d_{Holes}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.04012407</td>
<td>-0.1168475</td>
</tr>
<tr>
<td>2</td>
<td>0.02102764</td>
<td>-0.0795775</td>
</tr>
<tr>
<td>3</td>
<td>0.00828104</td>
<td>-0.0410453</td>
</tr>
<tr>
<td>4</td>
<td>0.00290449</td>
<td>-0.0189977</td>
</tr>
<tr>
<td>5</td>
<td>0.00095689</td>
<td>-0.0083189</td>
</tr>
<tr>
<td>6</td>
<td>0.00030322</td>
<td>-0.0035275</td>
</tr>
<tr>
<td>7</td>
<td>0.00009359</td>
<td>-0.0014662</td>
</tr>
<tr>
<td>8</td>
<td>0.00002835</td>
<td>-0.0006016</td>
</tr>
<tr>
<td>9</td>
<td>0.00000847</td>
<td>-0.0002447</td>
</tr>
<tr>
<td>10</td>
<td>0.00000259</td>
<td>-0.0000989</td>
</tr>
</tbody>
</table>

5.3 Deciding Best Projection Dimension

We propose two possible measurements, $M_1$ and $M_2$, of structures in projection using the results in (5.9) and (5.10), where

$$M_1 = \frac{I^d_{Holes}}{\max I^d_{Holes}}$$

$$M_2 = \frac{I^d_{Holes} - \min I^d_{Holes}}{\max I^d_{Holes} - \min I^d_{Holes}}$$

These two measures are motivated by the scree plot of PCA and $R^2$ in multiple regression. To assess and compare the appropriateness of above two measurements, we generate three different 6-dimensional data sets.

Data1: 50 observations from $\mathcal{N}(\mathbf{0}_6, \mathbf{I}_6)$

50 observations from $\mathcal{N}((5,0,0,0,0,0)^T, \mathbf{I}_6)$

50 observations from $\mathcal{N}(\mathbf{0}_6, \mathbf{I}_6)$

Data2: 50 observations from $\mathcal{N}((5,0,0,0,0,0)^T, \mathbf{I}_6)$

50 observations from $\mathcal{N}((\frac{5}{2}, \frac{5\sqrt{3}}{2}, 0, 0, 0, 0)^T, \mathbf{I}_6)$
50 observations from $\mathcal{N}(0_6, I_6)$
50 observations from $\mathcal{N}((5, 0, 0, 0, 0, 0)^T, I_6)$

Data3:
50 observations from $\mathcal{N}((\frac{5}{2}, \frac{5\sqrt{3}}{2}, 0, 0, 0, 0)^T, I_6)$
50 observations from $\mathcal{N}((\frac{5}{2}, \frac{5\sqrt{3}}{3}, 5\sqrt{\frac{3}{3}}, 0, 0, 0)^T, I_6)$

Data1 has two 6-D normal clusters of which mean vectors are apart from each other by 5 in 1-D subspace. Data2 has three 6-D normal clusters of which mean vectors are apart from each other by 5 in 2-D subspace (i.e. three mean points make an equilateral triangle). Data3 has four 6-D clusters of which mean vectors are apart from each other by 5 in 3-D subspace (i.e. four mean points make a regular tetrahedron). Above coordinate values of means have been calculated so that all the means are apart from each other by 5 starting the first mean at $0_6 = (0, 0, \cdots, 0)^T$. Figure 5, Figure 5 and Figure 5 show the mean points embedded in 3-D space.

For Data1, the best projection dimension should be 1-D, because most structure is only in 1-D. For Data2 and Data3, the best projection dimensions should be 2-D and
Figure 5.2 Plot of cluster means of data2 in 3-D subspace

Figure 5.3 Plot of cluster means of data3 in 3-D subspace
3-D, respectively. Thus, good structure measurement tools should be able to pick 1-D for Data1, 2-D for Data2, and 3-D for Data3. Now, we compare how differently $M_1$ and $M_2$ in (5.11) select the best projection dimension. The value of $\max I_H^d$ for each $d$ was computed.

Table 5.2, Table 5.3, and Table 5.4 show the Holes index values of the best projections found by our optimization algorithm (see Section 4.4) and the values of two measures $M_1$ and $M_2$, and the theoretical maxima.

Figure 5.4 shows the plot of $M_1$ and $M_2$ versus dimension for each data sets. We notice that $M_1$ reaches maximum at 1-D for Data1 and 2-D for Data2, which match the ideals. For Data3, where the real best projection dimension is 3-D, the values of $M_1$ at 1 to 3-D are similar. As for $M_2$, the value generally increases as dimension increases. But for Data1, it has larger value at 1-D than 2-D. For Data2, the slope from the $M_2$ value at 1-D to 2-D is bigger than that from 2-D to 3-D. For Data3, two slopes are close to each other. From this we notice that these two measurements, especially $M_1$ can be used in selecting the best projection dimension, though the interpretation of plot should be examined more.
Table 5.2  Data1 : Holes index values for best projections, relative structure amount

<table>
<thead>
<tr>
<th>d</th>
<th>$\max I_H^d$</th>
<th>$I_H^d$</th>
<th>$M_1$</th>
<th>$M_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.040124</td>
<td>0.026780</td>
<td>0.6674381</td>
<td>0.9149926</td>
</tr>
<tr>
<td>2</td>
<td>0.021028</td>
<td>0.010877</td>
<td>0.5172992</td>
<td>0.8991099</td>
</tr>
<tr>
<td>3</td>
<td>0.008281</td>
<td>0.004489</td>
<td>0.5420801</td>
<td>0.9231230</td>
</tr>
<tr>
<td>4</td>
<td>0.002905</td>
<td>0.001397</td>
<td>0.4809742</td>
<td>0.9311711</td>
</tr>
<tr>
<td>5</td>
<td>0.000957</td>
<td>0.000393</td>
<td>0.4108134</td>
<td>0.9392198</td>
</tr>
</tbody>
</table>

Table 5.3  Data2 : Holes index values for best projections, relative structure amount

<table>
<thead>
<tr>
<th>d</th>
<th>$\max I_H^d$</th>
<th>$I_H^d$</th>
<th>$M_1$</th>
<th>$M_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.040124</td>
<td>0.016108</td>
<td>0.4014621</td>
<td>0.8470056</td>
</tr>
<tr>
<td>2</td>
<td>0.021028</td>
<td>0.009595</td>
<td>0.4562859</td>
<td>0.8863574</td>
</tr>
<tr>
<td>3</td>
<td>0.008281</td>
<td>0.003482</td>
<td>0.4204282</td>
<td>0.9026997</td>
</tr>
<tr>
<td>4</td>
<td>0.002905</td>
<td>0.001187</td>
<td>0.4085054</td>
<td>0.9215609</td>
</tr>
<tr>
<td>5</td>
<td>0.000957</td>
<td>0.000297</td>
<td>0.3107588</td>
<td>0.9288982</td>
</tr>
</tbody>
</table>

Table 5.4  Data3 : Holes index values for best projections, relative structure amount

<table>
<thead>
<tr>
<th>d</th>
<th>$\max I_H^d$</th>
<th>$I_H^d$</th>
<th>$M_1$</th>
<th>$M_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.040124</td>
<td>0.018806</td>
<td>0.4687077</td>
<td>0.8641945</td>
</tr>
<tr>
<td>2</td>
<td>0.021028</td>
<td>0.009480</td>
<td>0.4508463</td>
<td>0.8852205</td>
</tr>
<tr>
<td>3</td>
<td>0.008281</td>
<td>0.003732</td>
<td>0.4506325</td>
<td>0.9077705</td>
</tr>
<tr>
<td>4</td>
<td>0.002905</td>
<td>0.001241</td>
<td>0.4271771</td>
<td>0.9240369</td>
</tr>
<tr>
<td>5</td>
<td>0.000957</td>
<td>0.000362</td>
<td>0.3783314</td>
<td>0.9358689</td>
</tr>
</tbody>
</table>
5.4 Optimization Algorithm

The algorithm for maximizing projection index consists of global random search and local optimization. First, the algorithm generates a set of solutions randomly. Next, at each solution, the local maximization step improves the index as much as possible. This step searches the neighborhood of current solution. The tangent of "squint angle" (Tukey and Tukey, 1981; Huber, 1990) is recommended as the initial size of this neighborhood. Squint angle is such that an interesting structure is recognizable only if the projection direction is within this angle from the ideal direction (Huber, 1990). The improved solutions are compared to produce a global optimum projection matrix. The algorithm for 2-D projection is presented as follows by Posse (1995).

(i) Global Search: generate two random candidates \((a_1, a_2), (b_1, b_2)\), within a neighborhood of current solutions \((a, \theta)\):
where \( c \) is the size of the neighborhood visited around the current solution and \( \omega \) is a randomly generated unit vector uniformly distributed on the p-sphere.

That is, only \( a_1 \) and \( b_1 \) are randomly generated within the neighborhood of \((\alpha, \beta)\) and \( a_2 \) and \( b_2 \) are simply Gram-Schmidt vectors from each.

(ii) Local Optimization : optimize the index.

Step 1: Use a constant value for \( c \) and generate two candidates as in (i). The current solution moves to the one among the two candidates which improves the index value more.

Step 2: If both candidates continue to fail to increase the index within a specified number of trials, the value of \( c \) is halved.

- If \( c \) falls behind a pre-defined value, the optimization is stopped.

Posse's application of Huber's method is restricted to a two dimensional projection. We extend it to arbitrary projection dimension (p). The global search step above is extended to p-dimension as follows.

\((\alpha_1, \alpha_2, \ldots, \alpha_p)\) : current best projection

(i)-1 Generate one candidate \((\alpha_1, \alpha_2, \ldots, \alpha_p)\) from \((\alpha_1, \alpha_2, \ldots, \alpha_p)\)

\[ a_1 = \frac{\alpha_1 + c\omega}{||a_1 + c\omega||}, a_2 = \frac{\alpha_2 + c\omega}{||a_2 + c\omega||}, \ldots, a_p = \frac{\alpha_p + c\omega}{||a_p + c\omega||} \]  

(5.12)

(i)-2 Gram-Schmidt above.

For each \( 1 \leq j \leq p \),

\[ a_i = \frac{a_i - (a_j^T a_i) a_j}{||a_i - (a_j^T a_i) a_j||}, \text{ for all } j < i \leq p \]  

(5.13)
(i)-3 Generate second candidate \((b_1, b_2, \ldots, b_p)\) from \((\alpha_1, \alpha_2, \ldots, \alpha_p)\)

\[
b_1 = \frac{\alpha_1 - \omega}{\|\alpha_1 - \omega\|}, \quad b_2 = \frac{\alpha_2 - \omega}{\|\alpha_2 - \omega\|}, \quad \cdots, \quad b_p = \frac{\alpha_p - \omega}{\|\alpha_p - \omega\|}.
\]

(5.14)

(i)-4 Gram-schmidt above.

For each \(1 \leq j \leq p\),

\[
b_i = \frac{b_i - (b_j^T b_i) b_j}{\|b_i - (b_j^T b_i) b_j\|}, \text{ for all } j < i \leq p
\]

(5.15)

(ii) Local Optimization: same as Posse's 2-D case.

5.5 Optimum Parameter Values

The optimization algorithm needs three parameters: \(m\), the number of initial solutions; \(half\), the speed of the neighborhood shrinking, that is, the number of random generations at each candidate; \(c\), the size of the neighborhood visited surrounding current candidate. As described in the previous section, the tangent of squint angle 80 degree \(\approx 5.67\) is recommended as the initial value of \(c\). Table 5.5 is showing the Holes index values for 3-D projections found by optimization algorithm for sphered 9-D cube data at different combinations of \(m\) and \(half\) with \(c=5.67\) fixed. 9-D cube data consists of 512 vertexes of a 9 dimensional cube (Section 2.5.1). We notice that the Holes index value does not always increase as \(m\) and \(half\) increase, unlike the expectations. The index for 200 initial solutions and 30 random generation at each solution is the largest. Thus we notice that to increase values for these parameters does not always lead to better projections. Instead of increasing the number of candidates or iterations, the gradient-based method could be added to search for a more accurate projection at each candidate locally.
Table 5.5  Holes index for different parameter values

<table>
<thead>
<tr>
<th>$m$</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.0159454</td>
<td>0.0152434</td>
<td>0.0159803</td>
<td>0.0157230</td>
</tr>
<tr>
<td>200</td>
<td>0.0168854</td>
<td>0.0170266</td>
<td>0.0161503</td>
<td>0.0166874</td>
</tr>
<tr>
<td>300</td>
<td>0.0173011</td>
<td>0.0159317</td>
<td>0.0165020</td>
<td>0.0165782</td>
</tr>
<tr>
<td>400</td>
<td>0.0166309</td>
<td>0.0162940</td>
<td>0.0164927</td>
<td>0.0164725</td>
</tr>
<tr>
<td>Average</td>
<td>0.0166907</td>
<td>0.0161239</td>
<td>0.0162813</td>
<td>0.0163653</td>
</tr>
</tbody>
</table>
6 EXAMPLES

6.1 Projection Pursuit and Principal Component Analysis: 9-D Cube Data

As briefly described in Section 2.5.1, if we look at the random projections of this data, they are approximately normal mostly. But since every marginal distribution of this data has two clear clusters (0 and 1), the global optimum projections for 1-D and 2-D should be the projection on two distinct points on a line, and the projection on the 4 vertices of a rectangle respectively. Principal component analysis (PCA) should not be able to find these since the covariance matrix is already diagonal and its eigenvalues are all identical. Thus, PCA would pick random projections for all 9 principal components (PCs). We run our PP algorithm on these 9 PCs, that is sphered data. Figure 6.1 shows the histograms of 1-D projection of sphered data found by our projection pursuit (PP) algorithm, 1st PC and 5th PC. The order of PCs doesn't have any meaning since they are picked randomly. 5th PC happened to be generated closest to optimum 1-D projection (two points on a line) among the 9 PCs. The 1-D projection optimized by our PP algorithm found the structure most clearly among these three 1-D reduction.

2-D projection found by PP optimization in Figure 6.2 captures the structure of 4 vertexes of the rectangle though the projected points are scattered around the vertexes.
Figure 6.1  Histograms of 1-D projection by PP, 1st PC and 5th PC

Figure 6.2  2-D Projection by PP optimization
6.2 Projection Pursuit Algorithm and Relative Structure Measurement: Flea Beetle Data

We reduced the dimension of data from 6 to all five (1-D, · · · , 5-D) lower dimensions using our optimization algorithm. The raw data was sphered before projection pursuit. Table 6.1 shows the Holes index for each of the projections found and the ratio of them to theoretical maximum in Section 5.2. The computed value of \( M_1 \) in (5.11) decreases as the dimension of projection found increases. To show the clusters which are preserved in the projection found by algorithm more clearly, we select 2-D projection which is the second largest of \( \frac{I_H^d}{\max I_H^d} \)'s in Table 6.1. Figure 6.3 and Figure 6.4 are showing two-dimensional projections found by optimization algorithm and 2-D guided tour each with colors representing real species. Three clusters are clear in both figures.

Table 6.1 Ratio of holes index values for projections to theoretical maximum

<table>
<thead>
<tr>
<th>heightd</th>
<th>( \frac{I_H^d}{\max I_H^d} )</th>
<th>( \frac{\max I_H^d}{\max I_H^d} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.027329</td>
<td>0.0401241</td>
</tr>
<tr>
<td>2</td>
<td>0.012681</td>
<td>0.0210176</td>
</tr>
<tr>
<td>3</td>
<td>0.004858</td>
<td>0.0082810</td>
</tr>
<tr>
<td>4</td>
<td>0.001642</td>
<td>0.0029045</td>
</tr>
<tr>
<td>5</td>
<td>0.000464</td>
<td>0.0009569</td>
</tr>
</tbody>
</table>

6.3 Initial Partitioning Algorithm and Hierarchical Model-Based Cluster Analysis: Satellite Image Data

6.3.1 HMCLUST without Initial Partition

Hierarchical model-based cluster analysis (HMCLUST) was performed for the sphered data set without the initial partition given. Table 6.2 shows the cross-tabulation of true classification and the clusters found by HMCLUST without initial partition.
Figure 6.3  Two dimensional projection found by algorithm

Figure 6.4  Two dimensional projection found by guided tour
Table 6.2  True and MCLUST classification for sphered data without partition

<table>
<thead>
<tr>
<th></th>
<th>MCLUST</th>
<th>True</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>914</td>
<td>1</td>
<td>27</td>
<td>120</td>
<td>0</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>433</td>
<td>16</td>
<td>6</td>
<td>9</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>55</td>
<td>0</td>
<td>884</td>
<td>17</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0</td>
<td>309</td>
<td>33</td>
<td>34</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>44</td>
<td>40</td>
<td>56</td>
<td>157</td>
<td>138</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>286</td>
<td>145</td>
<td>131</td>
<td>476</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The percentage of correct clustering by HMCLUST without initial partition is 65.32%\(^{2897}\). Total CPU time was 720 seconds.

### 6.3.2 HMCLUST with Initial Partition

Table 6.4 is the result of correct classification and clusters found by HMCLUST for the data with the initial partition given. The initial partition was obtained by the peeling and pruning steps with the parameter values in Section 4.4.2. The value for “factor” in peeling stage was 1.5. The size of neighborhood used was 3. That is, the edges which are longer than the average edge weight within 3 step neighborhood by 1.5 times standard deviation of those edges have been peeled. After peeling process, 68 edges (1.53% of total 4434 edges) were disconnected. The A and F arrays (Section 4.4.2) from this peeling stage were taken as input in following pruning process. The total CPU time for initial partitioning was 228 seconds including 15 seconds for MST construction and 213 for peeling and pruning. The size limit of partition component used in the algorithm was 4. The number of partitions from pruning was 1904. The distribution of size of partition components is as follows.

The average size of partitions is 2.33 \(\frac{4435}{1904}\). Total running time was 145 seconds and the rate of correct classification is 68.72%.
Table 6.3 Number of partitions for different partition sizes

<table>
<thead>
<tr>
<th>Partition Size</th>
<th>Number of Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>913</td>
</tr>
<tr>
<td>2</td>
<td>170</td>
</tr>
<tr>
<td>3</td>
<td>102</td>
</tr>
<tr>
<td>4</td>
<td>719</td>
</tr>
</tbody>
</table>

Table 6.4 True and MCLUST classification for sphered data with initial partition

<table>
<thead>
<tr>
<th></th>
<th>MCLUST</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>980</td>
<td>0</td>
<td>16</td>
<td>9</td>
<td>9</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>377</td>
<td>1</td>
<td>3</td>
<td>84</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>679</td>
<td>212</td>
<td>6</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>86</td>
<td>46</td>
<td>44</td>
<td>236</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>345</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>28</td>
<td>332</td>
<td>654</td>
<td></td>
</tr>
</tbody>
</table>

6.3.3 Comparison of Two Results

Surprisingly, the initial partition was shown to improve the clustering. The following Table 6.5 combines the comparisons between the program running and results of HMCLUST without initial partition and with initial partition. Table 6.6 shows the comparison of the classification correctness between HMCLUST with initial partition and HMCLUST without initial partition for each size of partitions where each data point belongs to.

Figure 6.5, 6.6 and 6.7 show the real clusters, clusters found by HMCLUST with partition and clusters found by HMCLUST without partition, respectively. The colors represent the following real clusters which each data point belong to.
Figure 6.5  Original clusters: Original colors are available in book form only.

Figure 6.6  Clusters found by HMCLUST with initial partition: Original colors are available in book form only.
Table 6.5  Comparisons between HMCLUST running and results without and with initial partition

<table>
<thead>
<tr>
<th>Number of cases</th>
<th>without Initial Partition</th>
<th>with Initial Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(at the beginning stage)</td>
<td>4435 observations</td>
<td>1904 partitions</td>
</tr>
<tr>
<td>CPU Time (sec.)</td>
<td>720</td>
<td>373</td>
</tr>
<tr>
<td></td>
<td>145 (HMCLUST)</td>
<td>15 (MST)</td>
</tr>
<tr>
<td></td>
<td>213 (peel+prune)</td>
<td></td>
</tr>
<tr>
<td>Correct classification (%)</td>
<td>65.32</td>
<td>68.72</td>
</tr>
</tbody>
</table>

Table 6.6 Correct and incorrect classification with initial partition and without initial partition for points corresponding to partition components with different sizes

<table>
<thead>
<tr>
<th></th>
<th>with partition</th>
<th>w/o partition</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correct</td>
<td>Incorrect</td>
<td>Correct</td>
</tr>
<tr>
<td>1</td>
<td>641</td>
<td>272</td>
<td>649</td>
</tr>
<tr>
<td>2</td>
<td>218</td>
<td>122</td>
<td>207</td>
</tr>
<tr>
<td>3</td>
<td>199</td>
<td>107</td>
<td>168</td>
</tr>
<tr>
<td>4</td>
<td>2023</td>
<td>853</td>
<td>1873</td>
</tr>
</tbody>
</table>

Table 6.7 Cluster descriptions and colors in Figure 5.6, 5.7 and 5.8

<table>
<thead>
<tr>
<th>Cluster number</th>
<th>cluster description</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Red Soil</td>
<td>Red</td>
</tr>
<tr>
<td>2</td>
<td>Cotton Crop</td>
<td>Green</td>
</tr>
<tr>
<td>3</td>
<td>Grey Soil</td>
<td>Yellow</td>
</tr>
<tr>
<td>4</td>
<td>Damp Grey Soil</td>
<td>Sky Blue</td>
</tr>
<tr>
<td>5</td>
<td>Soil with Vegetation Stubble</td>
<td>Orchid</td>
</tr>
<tr>
<td>7</td>
<td>Very Damp Grey Soil</td>
<td>Peru</td>
</tr>
</tbody>
</table>
Figure 6.7 Clusters found by HMCLUST without initial partition: Original colors are available in book form only.
7 DISCUSSION

Grand tour of the JAVA visualization software introduced here would give the indications of cluster structures before trying cluster analysis for small sized data sets. (Currently the software cannot be run for large data.) It would help us to understand initial partitioning algorithms addressed in this thesis. Linked brushing of specific MST edges between the edge weight graph window and the grand tour window of the software can show how the edges which have been designated as those to be peeled are connecting certain clusters and thereby help to validate the peeling step. Also grand tour would help us to determine which types of variance matrices we should give as the option for HMCLUST by giving the indications of the shape of each cluster of the (sphered and projected) data.

The major objectives of peeling in the initial partitioning process are to separate well-separated clusters and to disconnect outliers from the clusters (Posse, 1998). The criterion of “inconsistency” for choosing the edges to be peeled has been suggested mostly in terms of clusters separation in this thesis. However, it may be able to be used for detecting local outliers. Further research about detecting local outliers with the exploratory method would be worthwhile. How to discriminate the data points which are from a large sparse cluster from the outliers correctly should be the main point to be considered in this research.

The projection pursuit algorithm proposed in Section 5.4 tries to find the global maximum of the projection index in a complete random manner while avoiding many
local maxima, but it is failing to produce accurate solutions. As discussed in Section 2.4.2, complete gradient-based projection pursuit has problems such as low speed and too many local optima, but it would give the precise best projection if the search is performed near the true optimum. The combination between random global search and gradient-based method would make a better optimization algorithm. For instance, to find the rough global optimum by the random global algorithm and to improve projection index by gradient-based starting from the global optimum can be an example of this combination. Other ways to combine both for locating global and precise best projection need to be devised.

We have proposed ways of solving several problems in clustering large data. As the methods suggested in this research, the possible steps for clustering large data with many cases and variables would be to eliminate the influence of nuisance variables, the best lower dimensional projection which preserves the cluster structure in original data is found by the projection pursuit algorithm in Section 5.4. The original data should be sphered before projection pursuit. The best lower dimensional projection of sphered data is partitioned into a set of many small classifications using the peeling and pruning algorithm using the minimal spanning tree constructed from the projection.

This projection and the identifying labels of partitions for each observation become the input for hierarchical model-based clustering (HMCLUS) instead of singleton clusters so that HMCLUS can be run more efficiently. Though HMCLUS is known to have lower speed than distance-based hierarchical agglomerative clustering methods such as single linkage or complete linkage, the speeds of these methods are also quadratic in the number of cases. Starting these methods from the initial set of partitions of large data would also make clustering faster than starting from singleton clusters.
APPENDIX A TECHNICAL DETAILS OF VISUALIZATION PROGRAM

1. Major Functions

- 2-D Grand Tour: Program starts with 2-D grand tour of data. The user should answer whether the data should be standardized or not before starting the program. If the user's answer is yes, the program standardizes the data and uses for grand tour. (Grand Tour: Section 2.2.2)

- Minimal Spanning Tree: On pushing button("Draw MST") in the bottom 1, MST is constructed in entire space and its edges are added in grand tour window. User can choose either to construct MST with data points or to read MST edges connection from input file. Program recognizes the MST file by filename extension(datafilename.mst).

- MST Edge Weight Graph: When MST is constructed, the graph of its ordered edge weights is drawn on the second popped-up window. Y-axis of the graph represents the edge weight. Points in this graph are linked with the corresponding edges in grand tour window.

- Peeling MST Edges: Peeling can be done in two ways, either (i) brush points in MST edge weight graph and peel the corresponding edges in the grand tour window automatically or (ii) scroll the bar at the bottom of edge weight graph window and peel the corresponding percentage of longest edges in the grand tour window.
• Partitioning: The cluster assignment of each data points, from peeling edges can be done by pushing the button(“Make Partition”). The program automatically produce A and F array and groups the data points from these with algorithm in Section 4.4. Also, the program can read the peeling information from the input file and group the points from this.

3. Tools

• Linking: With two windows linking, the MST edges in the grand tour window which correspond to brushed points in the edge weight graph window are automatically faded.

• Coloring Partitions: User can color the points differently according to their group assignments by pushing the button(“Color Partition”).

• Suspend: User can pause the grand tour running or resume it with the “Suspend/Resume” button.

3. Input/Output Files

• Input: datafile(“datafile”), A array in MST(Section 4.3)(“datafile.mst”)

• Output: F array from peeling(“datafile.af”), group assignments(“datafile.class”)
APPENDIX B  ALGORITHM FOR GROUPING FROM PEELING RESULT (SECTION 4.3)

INPUT: 1-D arrays A, F, D; 2-D array TD
OUTPUT: 1-D array Label
WORKSPACE: 1-D arrays C, S

Set Label(i) ← 0, i=1,···,n
Set C(i)=D(i), i=1,···,n
i←1, k←1, Label(i)←k
Repeat
    While j ≤ D(i)
        If Label(TD(i,j)) ≠ 0
            j←j+1
            C(i)←C(i)-1
        Else
            If (edge(i,TD(i,j) is peeled) k←k+1
            Label(TD(i,j))←k
            Stack TD(i,j) into S
            C(i)←C(i)-1
            If D(TD(i,j))=1 C(TD(i,j))=C(TD(i,j))-1; j←j+1
            Else i←TD(i,j); j←1
End If
End While

For m=current size of S to m=1 by -1
    If C(S(m)) ≥ 1
        i ← S(m); Break;
    End If
End For

If current size of S = n Break;
j ← D(i) - C(i) + 1

End Repeat
APPENDIX C  PRUNING ALGORITHM(SECTION 4.4.2)

INPUT: 1-D arrays A, F, D ; 2-D array TD; constant SIZE
OUTPUT: 1-D arrays Label, Size ; nc (number of components)
WORKSPACE: 1-D array S

Set Label(i) ← 0, i=1, ..., n; nc← 0
i←1, k←0
Create a new component:
nc←nc+1
5 Label(i)←nc; Size(nc) ←1
For j=1 to D(i)
    If Label(TD(i,j))=0 then
        if (edge(i,TD(i,j) is peeled or Size(nc) ≥ SIZE)
            nc←nc+1
        endif
        Label(TD(i,j))←nc ; Size(nc)←Size(nc)+1
    End if
End for

Move to the next node:
If A(i) is only node among TD(i,j) s.t. node(A(i)) ≥ 1 among TD(i,j) (1 ≤ j ≤ node(i))
i← A(i); go to 5
Else if
\[ k \leftarrow k + 1; \ S(k) \leftarrow i \]
\[ i \leftarrow TD(i, j^*), \ j^* = \min_j \{ \text{node}(TD(i, j)) > 1 \} \]
\[ \text{goto 5} \]

Else (that is, node(TD(i, j))=1 for all 1 \leq j \leq \text{node}(i))

Go to the most recently saved node which have unvisited incident nodes:

If k=0 then Stop;
\[ i \leftarrow S(k); \ k \leftarrow k - 1 \]
\[ i \leftarrow TD(\text{node}(i, j^{**})) \text{ s.t.} \]
\[ j^{**} = \min_j \{ \text{Label}(\text{node}(TD(i, jj))) = 0 \text{ for any } 1 \leq j \leq \text{node}(TD(i, j)) \} \]

End if
APPENDIX D  DERIVATION OF (3.4), (3.5) and (3.6)

(4.4): If $Y$ follows the Gumbel distribution with density $\exp(-y - \exp(-y))$, its m.g.f.

$$M_y(t) = \int_{-\infty}^{\infty} \exp(yt) \exp(-y - \exp(-y)) \, dy$$

$$= \Gamma(1 - t), \text{ by the transformation } e^{-y} = z$$

$$= \exp(\gamma t + \sum_{n=2}^{\infty} S_n t^n/n), \text{ where } S_n = \sum_{i=1}^{\infty} \lambda^{-n}, \text{ for } n \geq 2$$

(Gumbel(1957))

$$\frac{\partial}{\partial t} \Gamma(1 - t) = \Gamma(1 - t)(\gamma + \sum_{n=2}^{\infty} S_n t^{n-1})$$

$$E(Y) = \frac{\partial}{\partial t} \Gamma(1 - t) \bigg|_{t=0}$$

$$= \gamma$$

$$\frac{\partial^2}{\partial t^2} \Gamma(1 - t) \bigg| = \frac{\partial}{\partial t} \Gamma(1 - t)(\gamma + \sum_{n=2}^{\infty} S_n t^{n-1}) + \Gamma(1 - t)(\sum_{n=2}^{\infty} (n-1)S_n t^{n-2})$$

$$\frac{\partial^2}{\partial t^2} \Gamma(1 - t) \bigg|_{t=0} = \gamma^2 + \frac{\pi^2}{6}, \text{ by } S_2 = \frac{\pi^2}{6} \text{ (Gumbel(1957))}$$

$$V(Y) = \frac{\pi^2}{6}$$

(D.4)
(4.8): If $Y$ follows 2nd order Gumbel distribution with density $\exp(-2y - \exp(-y))$, its m.g.f.

$$M_y(t) = \int_{-\infty}^{\infty} \exp(yt) \exp(-2y - \exp(-y))dy$$

$$= - \int_0^{\infty} z^{-t} e^{-z} \frac{z^2}{z} dz, \text{ by the transformation } e^{-y} = z$$

$$= \int_0^{\infty} z^{1-t} e^{-z} dz$$

$$= \Gamma(2 - t)$$

$$E(Y) = \frac{\partial}{\partial t} \Gamma(2 - t) |_{t=0}$$

$$= \gamma - 1, \text{ by } \Gamma(2 - t) = (1 - t)\Gamma(1 - t)$$

$$\frac{\partial^2}{\partial t^2} \Gamma(2 - t) |_{t=0} = -2\gamma + \gamma^2 + \frac{\pi^2}{6}$$

$$V(Y) = \frac{\pi^2}{6} - 1$$
APPENDIX E  ACCOMPANYING COMPACT DISC AND RELEVANT INFORMATION

The compact disk contains: JAVA programs for multivariate visualization (Chapter 3); two test datasets (flea-beetle data, flea.dat, and particle physics data, prim7.dat); flea.dat.mst for off-line MST construction (APPENDIX A); JDK1.1.8 class packages for Windows including Java Virtual Machine, JRE1.1.8; README for instruction.

See README for installations and running of programs.
BIBLIOGRAPHY


