

DIVISIVE HIERARCHICAL CLUSTERING FOR INTERVAL-VALUED DATA

by

JIANKUN ZHU

(Under the Direction of Lynne Billard)

ABSTRACT

In the real world, we may encounter various kinds of data types to which conventional statistical theories and methods may not be directly applied, and symbolic data is one kind. The observations of a symbolic data set are described by categorical variables, intervals, histograms, distributions and so forth, instead of single values. Therefore, symbolic data need novel methods for analysis. In this dissertation, we develop divisive hierarchical clustering methodologies for interval-valued data which are the most commonly-used symbolic data. We first propose three monothetic divisive clustering algorithms for interval-valued data and a weighted symbolic principal component analysis (PCA) method for interval-valued data. The first algorithm is based on the symbolic covariance PCA method proposed by Le-Rademacher (2008) and Le-Rademacher and Billard (2012). The second algorithm is based on our proposed weighted symbolic PCA method. The third one is based on the endpoints of intervals. Then, two mixed-strategy algorithms combining these three algorithms are also proposed. A series of simulations is conducted to compare these algorithms with an existing monothetic divisive algorithm proposed by Chavent (1998, 2000). The two mixed-strategy algorithms outperform all the other monothetic algorithms in these simulations and they are also applied to real-world data to further validate their effectiveness. Furthermore, we propose a polythetic divisive clustering algorithm for interval-valued data based on minimum

spanning trees. The effectiveness of this algorithm is also verified through several simulations and real data applications.

INDEX WORDS: Symbolic data, Interval-valued data, Divisive hierarchical clustering, Monothetic algorithm, Polythetic algorithm, Symbolic principal component analysis, Minimum spanning tree

DIVISIVE HIERARCHICAL CLUSTERING FOR INTERVAL-VALUED DATA

by

JIANKUN ZHU

B.S., The University of Science and Technology of China, 2014

M.S., The University of Georgia, 2016

M.S., The University of Georgia, 2018

A Dissertation Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2019

© 2019

Jiankun Zhu

All Rights Reserved

DIVISIVE HIERARCHICAL CLUSTERING FOR INTERVAL-VALUED DATA

by

JIANKUN ZHU

Major Professor: Lynne Billard

Committee: T.N. Sriram
Cheolwoo Park
Shuyang Bai

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
August 2019

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my major advisor Dr. Lynne Billard. I have greatly benefited from her patient guidance, insightful comments and suggestions through my research. Her selfless dedication to the students makes a deep impression on me. No matter how busy she is, her door is always open to the students who need her guidance and advice. I would like to thank her for all her tremendous help and support.

I would also like to thank my committee members, Dr. T.N. Sriram, Dr. Cheolwoo Park and Dr. Shuyang Bai for their valuable suggestions and their time spending in reviewing this dissertation. I would also like to thank the faculty, staff and friends in the Department of Statistics and the Department of Computer Science for the knowledge they taught me, the help they gave me and the time they spent with me.

Lastly but most importantly, I want to thank my parents for their unconditional and endless love. Their love makes me grateful for what I already have.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	iv
LIST OF FIGURES	vii
LIST OF TABLES	x
CHAPTER	
1 INTRODUCTION	1
2 LITERATURE REVIEW	4
2.1 SYMBOLIC DATA	4
2.2 PRELIMINARIES OF THE DIVISIVE HIERARCHICAL CLUSTERING FOR SYMBOLIC DATA	8
2.3 CURRENT DIVISIVE HIERARCHICAL CLUSTERING ALGORITHMS FOR INTERVAL-VALUED DATA	15
2.4 PRINCIPAL COMPONENT ANALYSIS METHODOLOGIES FOR INTERVAL- VALUED DATA	23
3 MONOTHETIC DIVISIVE CLUSTERING FOR INTERVAL-VALUED DATA BASED ON SYMBOLIC PRINCIPAL COMPONENT ANALYSIS AND ENDPOINTS OF INTERVALS	27
3.1 MONOTHETIC DIVISIVE CLUSTERING ALGORITHMS	28
3.2 SIMULATION STUDIES	48
3.3 APPLICATIONS	71

4	POLYTHETIC DIVISIVE CLUSTERING FOR INTERVAL-VALUED DATA BASED ON MINIMUM SPANNING TREES	96
4.1	A POLYTHETIC DIVISIVE ALGORITHM BASED ON MINIMUM SPAN- NING TREES	96
4.2	SIMULATION STUDIES	99
4.3	APPLICATIONS	101
5	CONCLUSION AND FUTURE WORK	110
	BIBLIOGRAPHY	113
	APPENDIX: R CODE	120

LIST OF FIGURES

2.1	An example of a minimum spanning tree.	17
2.2	An example of a possible limitation of the monothetic method.	20
3.1	Projections of the centers of hyper-rectangles onto a principal component line.	31
3.2	An example of how to stack intervals into a histogram.	34
3.3	Six scenarios when two intervals move toward each other along the axis.	41
3.4	Projections of the V_s 's and V_l 's of two hyper-rectangles onto a principal component line.	45
3.5	(a) 3D scatter plot of the classical samples from Simulation 1; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.	52
3.6	(a) 3D scatter plot of the classical samples from Simulation 2; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.	55
3.7	(a) 3D scatter plot of the classical samples from Simulation 3; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.	56
3.8	(a) 3D scatter plot of the classical samples from Simulation 4; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.	59
3.9	(a) 3D scatter plot of the classical samples from Simulation 5; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.	60
3.10	Summary of the first five simulations.	62

3.11	(a) 3D scatter plot of the classical samples from Simulation 6; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.	64
3.12	(a) Projections of the interval-valued objects aggregated by the classical samples onto the xz-plane; (b) projections onto the yz-plane.	64
3.13	(a) 3D scatter plot of the classical samples from Simulation 7; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.	67
3.14	Dendrograms for the fats and oils dataset using the DIVEWPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	81
3.15	Dendrograms for the fats and oils dataset using the DIVECPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	89
3.16	Dendrograms for the face recognition dataset using the DIVEWPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	94
3.17	Dendrograms for the face recognition dataset using the DIVECPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	94
3.18	Validity indices for the clustering result of the face recognition dataset using the DIVEWPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	95
3.19	Validity indices for the clustering result of the face recognition dataset using the DIVECPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	95

4.1	Minimum spanning trees for the interval-valued Iris dataset: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	103
4.2	Dendrograms for the interval-valued Iris dataset using the polythetic algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	106
4.3	Validity indices for the clustering result of the interval-valued Iris dataset using the polythetic algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	106
4.4	Minimum spanning trees for the face recognition dataset: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	108
4.5	Dendrograms for the face recognition dataset using the polythetic algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	108
4.6	Validity indices for the clustering result of the face recognition dataset using the polythetic algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.	109

LIST OF TABLES

2.1	Bird colors and lengths.	5
2.2	Cars dataset.	6
3.1	Types of the monothetic algorithms.	49
3.2	Different simulation cases.	50
3.3	Clustering accuracies of the monothetic algorithms (Simulation 1).	54
3.4	Clustering accuracies of the monothetic algorithms (Simulation 3).	57
3.5	Clustering accuracies of the monothetic algorithms (Simulation 5).	61
3.6	Clustering accuracies of the monothetic algorithms (Simulation 6).	65
3.7	Clustering accuracies of the monothetic algorithms (Simulation 7).	68
3.8	Numbers of possible bipartitions caused by the different ordering criteria of the monothetic algorithms.	69
3.9	Average computation times of the monothetic algorithms (in seconds). . . .	70
3.10	Table of fats and oils.	72
3.11	w_{uj} values before the unit variance scaling.	73
3.12	w_{uj} values after the unit variance scaling.	73
3.13	Principal component coefficients and contribution rate for each principal com- ponent of $\mathbf{X}^s$ based on the DIVEWPCA algorithm.	75
3.14	Principal component coefficients and contribution rate for each principal com- ponent of $\mathbf{X}^l$ based on the DIVEWPCA algorithm.	76
3.15	First two PCs of $\mathbf{X}^s$ (Y_1^s and Y_2^s) and first two PCs of $\mathbf{X}^l$ (Y_1^l and Y_2^l). . . .	76
3.16	Possible bipartitions and the corresponding Δ_v 's at the first clustering stage of DIVEWPCA based on the span normalized Euclidean Hausdorff distance. . . .	79

3.17	Possible bipartitions and the corresponding Δ_v 's at the second clustering stage of DIVEWPCA based on the span normalized Euclidean Hausdorff distance.	80
3.18	Cut criterion at each clustering stage (partition node) of the two dendrograms given in Figure 3.14.	82
3.19	Validity indices for the clustering result of the fats and oils dataset using the DIVEWPCA algorithm.	82
3.20	Principal component coefficients and contribution rate for each principal component of $\mathbf{X}^s$ and $\mathbf{X}^l$ based on the DIVECPCA algorithm.	85
3.21	First three principal components of $\mathbf{X}^s$ (Y_1^s , Y_2^s and Y_3^s) and first three principal components of $\mathbf{X}^l$ (Y_1^l , Y_2^l and Y_3^l).	85
3.22	Possible bipartitions and the corresponding Δ_v 's at the first clustering stage of DIVECPCA based on the span normalized Euclidean Hausdorff distance.	87
3.23	Possible bipartitions and the corresponding Δ_v 's at the second clustering stage of DIVECPCA based on the span normalized Euclidean Hausdorff distance.	88
3.24	Cut criterion at each clustering stage (partition node) of the two dendrograms given in Figure 3.15.	89
3.25	Face recognition dataset.	91
4.1	Clustering accuracies of the polythetic algorithm in the seven simulations. .	101
4.2	Table of the interval-valued Iris dataset.	102
4.3	Removed edges, possible bipartitions and the corresponding Δ_v 's at the first clustering stage of the polythetic algorithm based on the span normalized Euclidean Hausdorff distance.	104
4.4	Removed edges, possible bipartitions and the corresponding Δ_v 's at the second clustering stage of the polythetic algorithm based on the span normalized Euclidean Hausdorff distance.	105

CHAPTER 1

INTRODUCTION

In traditional statistics textbooks, we learn how to calculate probabilities, how to conduct hypothesis tests, how to fit regression lines, how to design experiments and so forth. These techniques are widely used in analyzing classical data sets and their effectiveness has been proved. However, we may encounter various kinds of data in the real world. The data may contain data points indexed in time order or contain variables representing topological and geometric features. The data may be composed of observations with some missing variable values. The sample size of the data set or the dimension of the variables in the data set may be extremely high. These examples are just a small portion of the data types to which conventional statistical theories and methods may not be directly applied.

Symbolic data is another example of these data types which need novel methods for analysis. The concept of symbolic data was first introduced by Diday (1989). The observations of a symbolic data set can be described by categorical variables, intervals, histograms, distributions and so forth, instead of single numerical values. Symbolic data arise from different aspects of modern life. For example, systolic blood pressure and daily temperature are sometimes recorded by intervals using maximum values and minimum values as upper and lower limits. The exam grades of a class can be recorded by a histogram for better visualization. The medical records of groups of people with certain characteristics retained by a health insurance corporation can be recorded by a symbolic data table composed of interval-valued and categorical variables. In this dissertation, we focus on interval-valued data which are the most commonly-used symbolic data type.

Clustering is the task of grouping objects which share similar features or properties. There are numerous clustering algorithms and they can not easily be categorized. One way is to divide them into partitional clustering algorithms and hierarchical clustering algorithms (Jain et al. (1999)). Hierarchical clustering algorithms usually fall into two types: agglomerative hierarchical clustering and divisive hierarchical clustering. In this research, we focus on divisive hierarchical clustering algorithms for interval-valued data. We seek to build a hierarchy tree from one single cluster which consists of all the interval-valued observations and split the clusters recursively until the bottom of the tree is reached (only one observation in each cluster).

Several divisive hierarchical clustering algorithms have been proposed for interval-valued data. For example, a monothetic divisive hierarchical clustering algorithm for symbolic objects was developed by Chavent (1998, 2000), which gives the intervals an initial order by the midpoint values of the intervals at each clustering stage. Another example is a polythetic divisive clustering algorithm proposed by Kim (2009) and Kim and Billard (2011), which applies a kind of average dissimilarity to find the splinter cluster at each clustering stage. Although this method was only applied to histogram-valued data in Kim (2009) and Kim and Billard (2011), it can also be a useful tool for interval-valued data. They also extended two validity indices developed by Dunn (1974) and Davis and Bouldin (1979) from classical data to symbolic data to determine the optimal number of clusters.

This research aims to develop new divisive hierarchical clustering algorithms for interval-valued data and compare them with the existing algorithms to find the advantages and disadvantages of each algorithm.

This dissertation consists of five chapters. Chapter 2 reviews the concepts of symbolic data and interval-valued data, the preliminaries of the divisive hierarchical clustering for interval-valued data, current literature on the divisive hierarchical clustering algorithms for interval-valued data and current studies on the principal component analysis (PCA) methodologies for interval-valued data. In Chapter 3, we first propose three monothetic divisive clustering

algorithms for interval-valued data and a weighted symbolic PCA method for interval-valued data. The first algorithm is based on the symbolic covariance PCA method proposed by Le-Rademacher (2008) and Le-Rademacher and Billard (2012). The second algorithm is based on our proposed weighted symbolic PCA method. The third one is based on the endpoints of intervals. Then, two mixed-strategy algorithms combining these three algorithms are also proposed. Simulations are conducted to compare these algorithms with an existing monothetic divisive algorithm proposed by Chavent (1998, 2000). The two mixed-strategy algorithms are also applied to real-world data and the performances are summarized. In Chapter 4, we propose a polythetic divisive clustering algorithm for interval-valued data based on minimum spanning trees. Simulation studies and real data applications are also conducted to evaluate this algorithm. Chapter 5 is a summary of the contributions of this dissertation and a brief discussion of future work.

CHAPTER 2

LITERATURE REVIEW

This chapter is a review of current literature which is closely related to our work. Section 2.1 is an introduction to symbolic data and the empirical distribution function and descriptive statistics for interval-valued data. In Section 2.2, some preliminaries of the divisive hierarchical clustering for interval-valued data are introduced, including the dissimilarity and distance measures for interval-valued data, the clustering criterion and the validity indices. Section 2.3 is a review of current literature on divisive hierarchical clustering algorithms for interval-valued data. In Section 2.4, recent symbolic principal component analysis (PCA) methodologies for interval-valued data are briefly summarized since two of our proposed algorithms are based on them.

2.1 SYMBOLIC DATA

The concept of symbolic data was first introduced by Diday (1989). In Billard (2006), symbolic data are defined as hypercubes or Cartesian products of distributions in p -dimensional space R^p , in contrast to classical observations which are defined as points in R^p . In each dimension of the space R^p , a symbolic observation can be described by a categorical variable (multi-valued variable), a modal categorical variable, an interval, a histogram, a distribution and so forth, instead of a single numerical value. Table 2.1 is an example of symbolic data. This table displays the major colors and the ranges of lengths of seven different bird species. The major colors are described by a categorical variable while the ranges of lengths are described by intervals.

Table 2.1: Bird colors and lengths.

Bird Species	Major Color	Length (cm)
1	{black, white}	[10, 23]
2	{black}	[21.5, 25.5]
3	{gray}	[20, 53]
4	{black, white}	[15.5, 35]
5	{green, yellow}	[25, 30]
6	{gray, yellow}	[36.0, 50]
7	{gray}	[37, 43]

Symbolic data have two main sources. The first source is the data which are inherently symbolic such as systolic blood pressure, pulse rate and daily temperature. These kinds of data are sometimes characterized and recorded by intervals using maximum values and minimum values as upper and lower limits. Another typical example is the colors of animals. The color variable of a single subject may take more than one value such as half black and half yellow. In that case, it is straightforward to characterize the color of the subject using a modal categorical value, i.e., $\{black\ 0.5, yellow\ 0.5\}$. More examples of natural symbolic data can be found in Billard and Diday (2003, 2006).

The second source of symbolic data is the aggregated data sets which summarize some large data sets. The aggregated symbolic datasets turn the original datasets into manageable sizes while also keeping useful information. For example, some online learning management systems for universities summarize the exam grades of a whole class by a histogram graph for better visualization. In a world full of big data, analyzing data in this aggregated manner actually provides statisticians a natural way to become involved in the research relevant to big data using statistical theories and methodologies.

Table 2.2: Cars dataset.

Car	Price	Maximum Velocity	Acceleration Time
Aston Martin	[260.5, 460.0]	[298, 306]	[4.7, 5.0]
Audi A6	[68.2, 140.3]	[216, 250]	[6.7, 9.7]
Audi A8	[123.8, 171.4]	[232, 250]	[5.4, 10.1]
BMW 7	[104.9, 276.8]	[228, 240]	[7.0, 8.6]
Ferrari	[240.3, 391.7]	[295, 298]	[4.5, 5.2]
Honda NSR	[205.2, 215.2]	[260, 270]	[5.7, 6.5]
Mercedes C	[55.9, 115.2]	[210, 250]	[5.2, 11.0]
Porsche	[147.7, 246.4]	[280, 305]	[4.2, 5.2]

2.1.1 INTERVAL-VALUED DATA

Interval-valued data are the most commonly used type of symbolic data and we focus on interval-valued symbolic data in this dissertation. A p -dimensional interval-valued observation is actually an observation composed by p interval-valued variables. For an interval-valued dataset which consists of n p -dimensional interval-valued observations $\{Y_u = ([a_{u1}, b_{u1}), \dots, [a_{up}, b_{up}]), u = 1, \dots, n\}$, each observation can be visualized as a hyper-rectangle in R^p . The intervals may be open or closed at either end (i.e., $[a_{uj}, b_{uj})$, $(a_{uj}, b_{uj}]$, $[a_{uj}, b_{uj}]$ or (a_{uj}, b_{uj})). Table 2.2 is an example of an interval-valued dataset, which is part of the Cars Data extracted from Billard and Diday (2006). This dataset consists of eight 3-dimensional interval-valued observations and the three interval-valued variables in this table are Price ($\times 10^{-3}$, in euros), Maximum Velocity and Acceleration Time to reach a given speed.

Denote X_j as the j th variable of the interval-valued observations and use W to denote a point in the variable X_j of a random sample Y_u , i.e., $[a_{uj}, b_{uj})$. Symbolic data analysis

assumes that the points across each interval are uniformly distributed. Thus, for each ϵ ,

$$P(W \leq \epsilon) = \begin{cases} 0, & \epsilon < a_{uj}, \\ \frac{\epsilon - a_{uj}}{b_{uj} - a_{uj}}, & a_{uj} \leq \epsilon < b_{uj}, \\ 1, & \epsilon \geq b_{uj}. \end{cases}$$

The sample mean and sample variance for an interval-valued variable $X_j, j = 1, \dots, p$, were first defined by Bertrand and Goupil (2000). The sample mean $\bar{X}_j$ and sample variance S_j^2 are given by

$$\bar{X}_j = \frac{1}{2n} \sum_{u=1}^n (a_{uj} + b_{uj}), \quad (2.1)$$

$$S_j^2 = \frac{1}{3n} \sum_{u=1}^n (a_{uj}^2 + a_{uj}b_{uj} + b_{uj}^2) - \frac{1}{4n^2} \left[\sum_{u=1}^n (a_{uj} + b_{uj}) \right]^2. \quad (2.2)$$

Billard (2007) and Billard (2008) prove that the sample variance in Eq.(2.2) can be rewritten as

$$S_j^2 = \frac{1}{3n} \sum_{u=1}^n [(a_{uj} - \bar{X}_j)^2 + (a_{uj} - \bar{X}_j)(b_{uj} - \bar{X}_j) + (b_{uj} - \bar{X}_j)^2]. \quad (2.3)$$

Furthermore, the total sum of squares (TSS) for the variable X_j , i.e., nS_j^2 , can be written as

$$nS_j^2 = \sum_{u=1}^n (\bar{X}_{uj} - \bar{X}_j)^2 + \sum_{u=1}^n [(a_{uj} - \bar{X}_{uj})^2 + (a_{uj} - \bar{X}_{uj})(b_{uj} - \bar{X}_{uj}) + (b_{uj} - \bar{X}_{uj})^2]/3 \quad (2.4)$$

where

$$\bar{X}_{uj} = \frac{a_{uj} + b_{uj}}{2}.$$

The first summation in Eq.(2.4) is to measure the variation of the interval midpoints, i.e., the between sum of squares (BSS). The second summation in Eq.(2.4) is a measure of the internal variation of the n observations; Billard (2007, 2008) calls this term the within sum of squares (WSS). Thus, the total sum of squares nS_j^2 can be decomposed into

$$TSS = BSS + WSS \quad (2.5)$$

where

$$BSS = \sum_{u=1}^n (\bar{X}_{uj} - \bar{X}_j)^2$$

and

$$WSS = \sum_{u=1}^n [(a_{uj} - \bar{X}_{uj})^2 + (a_{uj} - \bar{X}_{uj})(b_{uj} - \bar{X}_{uj}) + (b_{uj} - \bar{X}_{uj})^2]/3.$$

Billard (2007, 2008) also derives that

$$WSS = \sum_{u=1}^n (b_{uj} - a_{uj})^2/12,$$

which conforms with the assumption that the points within each interval are uniformly distributed.

Similarly, the sample covariance between two interval-valued variables X_i and X_j can be calculated by the formula from Billard (2008):

$$\begin{aligned} Cov(X_i, X_j) = \frac{1}{6n} \sum_{u=1}^n [2(a_{ui} - \bar{X}_i)(a_{uj} - \bar{X}_j) + (a_{ui} - \bar{X}_i)(b_{uj} - \bar{X}_j) \\ + (b_{ui} - \bar{X}_i)(a_{uj} - \bar{X}_j) + 2(b_{ui} - \bar{X}_i)(b_{uj} - \bar{X}_j)]. \end{aligned} \quad (2.6)$$

Then, the Pearson correlation coefficient between X_i and X_j can be obtained by

$$r(X_i, X_j) = \frac{Cov(X_i, X_j)}{S_i S_j} \quad (2.7)$$

where S_i and S_j are the square roots of the sample variances from Eq.(2.2).

2.2 PRELIMINARIES OF THE DIVISIVE HIERARCHICAL CLUSTERING FOR SYMBOLIC DATA

Clustering plays an important role in data mining and statistical data analysis. There are hundreds of clustering methodologies and they can not easily be categorized. One way is to divide them into partitional clustering methodologies and hierarchical clustering methodologies (Jain et al. (1999)). The most important function of hierarchical clustering is to show the hierarchical relationships between the clusters by dendrograms. Hierarchical clustering

methodologies can be further divided into two types: agglomerative methodologies and divisive methodologies, depending on whether the hierarchy is merged in a “bottom-up” way or split in a “top-down” way. Hierarchical clustering can also be divided into polythetic methodologies and monothetic methodologies depending on whether one variable at a time is used or all variables are used simultaneously for the bipartition at each clustering stage. Hierarchical clustering methodologies for symbolic data can also be divided into agglomerative or divisive type and polythetic or monothetic type. Furthermore, we can divide hierarchical clustering methodologies for symbolic data by the data types to which they are applied. In this dissertation, our main topic is divisive hierarchical clustering methodologies for interval-valued data, including both polythetic methodologies and monothetic methodologies.

The skeleton or structure of the divisive hierarchical clustering algorithms for interval-valued data can be briefly summarized as:

Step 1:

Find suitable dissimilarity and distance measures for interval-valued data to obtain the distance matrix.

Step 2:

Start from a single cluster which contains all the objects. Apply some rules specified by different algorithms and some commonly-used clustering criteria to perform a bipartition at each clustering stage.

Step 3:

Continue the bipartition until a prespecified clustering stage or until the bottom of the hierarchical clustering tree is reached (only one observation in each cluster).

Some divisive clustering algorithms such as the monothetic algorithm developed by Chavent (1998, 2000) have already been developed for symbolic data. Though important in the development of clustering for symbolic data, Chavent’s algorithm was limited to interval data and the Hausdorff distance. To better understand those algorithms, some preliminaries

of the divisive hierarchical clustering for symbolic data are introduced in the following subsections. The preliminaries include the dissimilarity and distance measures for interval-valued data, the clustering criterion and the validity indices for symbolic hierarchical clustering.

2.2.1 DISSIMILARITY AND DISTANCE MEASURES FOR INTERVAL-VALUED DATA

To build a divisive hierarchical clustering tree (dendrogram), we need first to know how far apart different objects are, which dissimilarity and distance measures can tell us. Thus, we give the definitions of dissimilarity and distance measures.

Definition 1. *Let A_1 and A_2 be any two objects in the space Ω . Then, a **dissimilarity measure**, $d(A_1, A_2)$, between A_1 and A_2 , is a measure that satisfies*

1. $d(A_1, A_2) = d(A_2, A_1)$;
2. $d(A_1, A_2) > 0$ for all $A_1 \neq A_2$;
3. $d(A, A) = 0$ for all $A \in \Omega$.

Definition 2. *Let A_1 and A_2 be any two objects in the space Ω . Then, a **distance measure**, $d(A_1, A_2)$, between A_1 and A_2 , is a dissimilarity measure which satisfies reflexivity (property 1) and triangle inequality (property 2)*

1. $d(A_1, A_2) = 0$ implies $A_1 = A_2$;
2. $d(A_1, A_2) \leq d(A_1, A_3) + d(A_3, A_2)$ for all $A_1, A_2, A_3 \in \Omega$.

The Hausdorff distance (Hausdorff (1957)) is frequently used for interval observations. The distance measure used in this dissertation is another frequently occurring form of this distance, i.e., the span normalized Euclidean Hausdorff distance, which is extended by Chavent (2000). Besides, there are several other extensions of the Hausdorff distance, such as the Euclidean Hausdorff distance and the normalized Euclidean Hausdorff distance. Chen (2014) made a comprehensive comparison among these Hausdorff distances. The Gowda-Diday dissimilarity for interval-valued data is proposed by Gowda and Diday (1991a). Gowda and

Ravi (1995a) discussed the disadvantages of the Gowda-Diday dissimilarity and proposed new measures based on the position, span and content of each symbolic object. Ichino and Yaguchi (1994) proposed another distance for interval-valued data which is called the Ichino-Yaguchi distance. They also extended the Minkowski distance to the generalized (weighted) Minkowski distance of order $p \geq 1$. The generalized Minkowski distance is also a generalized form of the city block distance (when $p = 1$) and the normalized Euclidean distance (when $p = 2$). De Carvalho (1994, 1998) proposed two extensions of the Ichino-Yaguchi distance. Kim (2009) extended the Gowda-Diday dissimilarity and the Ichino-Yaguchi distance from interval-valued data to histogram-valued data.

The definitions of three of these distance or dissimilarity measures are given below, specifically the span normalized Euclidean Hausdorff distance, the Gowda-Diday dissimilarity measure and the Ichino-Yaguchi distance.

Definition 3. Let $A_u = ([a_{uj}, b_{uj}], j = 1, \dots, p), u = 1, \dots, n$, be interval-valued objects in $\Omega = \Omega_1 \times \dots \times \Omega_p$. Then, the **span normalized Euclidean Hausdorff distance** between A_{u_1} and A_{u_2} , $d(A_{u_1}, A_{u_2}), u_1, u_2 = 1, \dots, n$, is

$$d(A_{u_1}, A_{u_2}) = \left\{ \sum_{j=1}^p [|\Upsilon_j|^{-1} d_j(A_{u_1}, A_{u_2})]^2 \right\}^{1/2} \quad (2.8)$$

where $|\Upsilon_j|$ is the span of the observations, i.e.,

$$|\Upsilon_j| = \max_u \{b_{uj}\} - \min_u \{a_{uj}\} \quad (2.9)$$

and

$$d_j(A_{u_1}, A_{u_2}) = \max\{|a_{u_1j} - a_{u_2j}|, |b_{u_1j} - b_{u_2j}|\},$$

which is the Hausdorff distance between A_{u_1j} and A_{u_2j} .

Definition 4. Let $A_u = ([a_{uj}, b_{uj}], j = 1, \dots, p), u = 1, \dots, n$, be interval-valued objects in $\Omega = \Omega_1 \times \dots \times \Omega_p$. Then, the **Gowda-Diday dissimilarity** between A_{u_1} and A_{u_2} , $d(A_{u_1}, A_{u_2}), u_1, u_2 = 1, \dots, n$, is

$$d(A_{u_1}, A_{u_2}) = \sum_{j=1}^p [d_{1j}(A_{u_1}, A_{u_2}) + d_{2j}(A_{u_1}, A_{u_2}) + d_{3j}(A_{u_1}, A_{u_2})] \quad (2.10)$$

where

$$\begin{aligned} d_{1j}(A_{u_1}, A_{u_2}) &= (|(b_{u_1j} - a_{u_1j}) - (b_{u_2j} - a_{u_2j})|)/k_j, \\ d_{2j}(A_{u_1}, A_{u_2}) &= (b_{u_1j} - a_{u_1j} + b_{u_2j} - a_{u_2j} - 2I_j)/k_j, \\ d_{3j}(A_{u_1}, A_{u_2}) &= |a_{u_1j} - a_{u_2j}|/|\Upsilon_j| \end{aligned}$$

with

$$\begin{aligned} k_j &= |\max(b_{u_1j}, b_{u_2j}) - \min(a_{u_1j}, a_{u_2j})|, \\ I_j &= \left\{ \begin{array}{ll} |\max(a_{u_1j}, a_{u_2j}) - \min(b_{u_1j}, b_{u_2j})|, & \text{if } A_{u_1j} \cap A_{u_2j} \neq \phi \\ 0, & \text{if } A_{u_1j} \cap A_{u_2j} = \phi \end{array} \right\}, \\ |\Upsilon_j| &= \max_u \{b_{uj}\} - \min_u \{a_{uj}\}. \end{aligned}$$

Definition 5. Let $A_u = ([a_{uj}, b_{uj}], j = 1, \dots, p), u = 1, \dots, n$, be interval-valued objects in $\Omega = \Omega_1 \times \dots \times \Omega_p$. Then, the **Ichino-Yaguchi distance** between A_{u_1} and A_{u_2} , $d(A_{u_1}, A_{u_2})$, $u_1, u_2 = 1, \dots, n$, is

$$d(A_{u_1}, A_{u_2}) = \sum_{j=1}^p [|A_{u_1j} \oplus A_{u_2j}| - |A_{u_1j} \otimes A_{u_2j}| + \gamma(2|A_{u_1j} \otimes A_{u_2j}| - |A_{u_1j}| - |A_{u_2j}|)] \quad (2.11)$$

where

$$\begin{aligned} A_{u_1j} \oplus A_{u_2j} &= [\min\{a_{u_1j}, a_{u_2j}\}, \max\{b_{u_1j}, b_{u_2j}\}], \\ A_{u_1j} \otimes A_{u_2j} &= \begin{cases} [\max\{a_{u_1j}, a_{u_2j}\}, \min\{b_{u_1j}, b_{u_2j}\}], & A_{u_1j} \cap A_{u_2j} \neq \phi, \\ 0, & A_{u_1j} \cap A_{u_2j} = \phi, \end{cases} \end{aligned}$$

$|A|$ is the length of the interval A , and $0 \leq \gamma \leq 0.5$ is a specified constant.

2.2.2 CLUSTERING CRITERION FOR SYMBOLIC DIVISIVE HIERARCHICAL CLUSTERING

In agglomerative clustering, some criteria such as the complete linkage, the single linkage and the average linkage are often used to obtain the distances between clusters which can help to determine the clusters to be merged at each clustering stage. However, these criteria are not that useful in divisive clustering since divisive clustering starts from a single cluster.

The aim of divisive hierarchical clustering for symbolic data is to partition the dataset into a number of clusters. At each stage, only one cluster is partitioned into two sub-clusters. Suppose at the r^{th} stage, the hierarchical clustering gives r disjoint clusters $P_r = (C_1, \dots, C_r)$. Then, at the $(r + 1)^{th}$ stage, we choose C_v , one of the clusters in P_r , and divide it into two further sub-clusters $C_v = (C_v^{(1)}, C_v^{(2)})$. Thus, we can write the new partition as

$$P_{r+1} = (P_r \cup \{C_v^{(1)}, C_v^{(2)}\}) - \{C_v\}. \quad (2.12)$$

Details about this partition can be found in Chavent (1998, 2000). Then, the problem is how to determine whether this partition is the optimal partition at the r^{th} stage or not. To make the clusters after the bipartitions to be internally as homogeneous as possible and as heterogeneous as possible across clusters, Chavent (1998, 2000) extended the definitions of the within-cluster variation and the between-cluster variation from classical quantitative data to symbolic data, which are used to find the “best” bipartitions at each clustering stage.

Definition 6. For a cluster $C_v = \{y_1, \dots, y_{n_v}\}$, the within-cluster variation $I(C_v)$ is given by

$$I(C_v) = \frac{1}{\lambda} \sum_{i_1, i_2=1, i_1 < i_2}^{n_v} w_{i_1} w_{i_2} d^2(y_{i_1}, y_{i_2}) \quad (2.13)$$

where $d(y_{i_1}, y_{i_2})$ is a distance or dissimilarity measure between the observations y_{i_1} and y_{i_2} in C_v , $i_1, i_2 = 1, \dots, n_v$, and where w_i is the weight associated with the observation y_i , and where $\lambda = \sum_{i=1}^{n_v} w_i$.

Definition 7. For a partition $P_r = (C_1, \dots, C_r)$ at the r^{th} stage, the total within-cluster variation $W(P_r)$ is the sum of the within-cluster variations over all clusters in P_r , i.e.,

$$W(P_r) = \sum_{v=1}^r I(C_v) \quad (2.14)$$

where $I(C_v)$ is given in Definition 6.

Definition 8. For a partition $P_r = (C_1, \dots, C_r)$ at the r^{th} stage, $r = 1, \dots, n - 1$, where the complete set of observations $\Omega \equiv P_1$, the between-cluster variation is given by

$$B(P_r) = W(P_1) - W(P_r) = W(\Omega) - W(P_r). \quad (2.15)$$

From Eq.(2.14), we have the total within-cluster variation for P_{r+1} ,

$$W(P_{r+1}) = W(P_r) - (I(C_v) - I(C_v^{(1)}) - I(C_v^{(2)})). \quad (2.16)$$

Thus, to obtain the clusters with the smallest internal variation at the $(r+1)^{th}$ stage (to minimize $W(P_{r+1})$), we need to find the bipartition $C_v = (C_v^{(1)}, C_v^{(2)})$ which maximizes

$$\Delta_v = I(C_v) - I(C_v^{(1)}) - I(C_v^{(2)}). \quad (2.17)$$

All of our proposed divisive clustering methods in this research are based on this clustering criterion and we assign equal weights $w_i = 1, i = 1, \dots, n_v$, when calculating $I(C_v)$.

2.2.3 VALIDITY INDICES FOR SYMBOLIC HIERARCHICAL CLUSTERING

After the hierarchical clustering, we need to estimate the real number of clusters in the dataset. Kim (2009) and Kim and Billard (2011) extended two indices developed by Dunn (1974) and Davis and Bouldin (1979) from classical data to symbolic data. These two indices can be used as a reference when we want to find the optimal number of clusters in the dataset.

Definition 9. *The **Dunn index for symbolic observations**, DI_r^s , at the r^{th} stage $P_r = (C_1, \dots, C_r)$ is given by*

$$DI_r^s = \min_{u=1, \dots, r} \left[\min_{t=1, \dots, r, t \neq u} \left\{ \frac{I(C_t \cup C_u) - I(C_t) - I(C_u)}{\max_{l=1, \dots, r} I(C_l)} \right\} \right], \quad r = 2, \dots, n-1, \quad (2.18)$$

where $I(\cdot)$ is the within-cluster variation of Definition 6.

Definition 10. *The **Davis-Bouldin index for symbolic observations**, DB_r^s , at the r^{th} stage $P_r = (C_1, \dots, C_r)$ is given by*

$$DB_r^s = \frac{1}{r} \sum_{u=1}^r \left[\frac{\max_{t=1, \dots, r, t \neq u} \{I(C_t) + I(C_u)\}}{\min_{l=1, \dots, r, l \neq u} \{I(C_l \cup C_u) - I(C_l) - I(C_u)\}} \right], \quad r = 2, \dots, n-1, \quad (2.19)$$

where $I(\cdot)$ is the within-cluster variation of Definition 6.

2.3 CURRENT DIVISIVE HIERARCHICAL CLUSTERING ALGORITHMS FOR INTERVAL-VALUED DATA

For interval-valued data, there are several agglomerative hierarchical clustering algorithms such as Gowda and Ravi (1995b), Ravi and Gowda (1999), Guru et al. (2004) and Guru and Kiranagi (2005). For divisive hierarchical clustering, Gowda and Ravi (1995a) developed a divisive hierarchical clustering algorithm using a similarity measure and a dissimilarity measure. The algorithm involves a step which aims to find the minimum spanning tree (MST) of the symbolic data set. Chavent (1998, 2000) proposed a monothetic divisive hierarchical clustering algorithm for symbolic data, which give the intervals an initial order by the midpoints of the intervals to reduce the number of possible partitions at each clustering stage. Pirçon (2004), see also Rasson et al. (2008), proposed another monothetic divisive hierarchical clustering algorithm, which is based on the generalized hypervolumes clustering criterion instead of the minimum total within-cluster variation criterion described in Section 2.2.2. A polythetic divisive clustering algorithm was developed by Kim (2009) and Kim and Billard (2011), which applied a kind of average dissimilarity to find the cluster to be split at each clustering stage. Although this algorithm was only applied to histogram-valued data in Kim (2009) and Kim and Billard (2011), it can also be a useful tool for interval-valued data. Brito and Chavent (2012) proposed a monothetic divisive clustering algorithm for data described by both interval-valued and histogram-valued variables. In this section, a brief review of these algorithms is given.

2.3.1 THE DIVISIVE HIERARCHICAL CLUSTERING ALGORITHM USING THE CONCEPTS OF BOTH SIMILARITY AND DISSIMILARITY

This algorithm is proposed by Gowda and Ravi (1995a) and it is a polythetic clustering algorithm. In their paper, they first discussed the disadvantages of the Gowda-Diday dissimilarity and similarity proposed by Gowda and Diday (1991a, 1991b) and proposed new dissimilarity and similarity measures based on the position, span and content of symbolic

objects. Then, they proposed their clustering algorithm based on these dissimilarity and similarity measures.

Another important concept mentioned in their algorithm is the **minimum spanning tree (MST)**. The MST aims to select a subset of the weighted edges from a connected and undirected graph with the minimum total edge weight. The MST is an important tool in computer science and engineering, which has applications in the design of computer networks, algorithms, telecommunications networks and so forth. Figure 2.1 is an example of a MST obtained from the distance matrix of eight objects. One thing that needs to be mentioned here is not only the lengths of the edges between the objects in this MST but also the lengths of the removed edges during the building process of the MST are proportional to the actual distances between the objects. There are some algorithms designed for extracting the MST from a connected, undirected and weighted graph, such as Prim's algorithm (invented by Jarník (1930) and rediscovered by Prim (1957)), Kruskal's algorithm (Kruskal (1956)) and the provably optimal algorithm (Pettie and Ramachandran (2002)).

After the brief introduction to the concept of minimum spanning trees, we can start to summarize the algorithm of Gowda and Ravi (1995a). Let an interval-valued dataset consist of n symbolic observations.

Step 1:

Compute the dissimilarities between all pairs of symbolic data in this dataset based on their proposed dissimilarity measure and thus obtain the dissimilarity matrix.

Step 2:

Find the minimum spanning tree based on the dissimilarity matrix.

Step 3:

Remove the edge with the largest dissimilarity value from the minimum spanning tree to obtain two groups of symbolic objects.

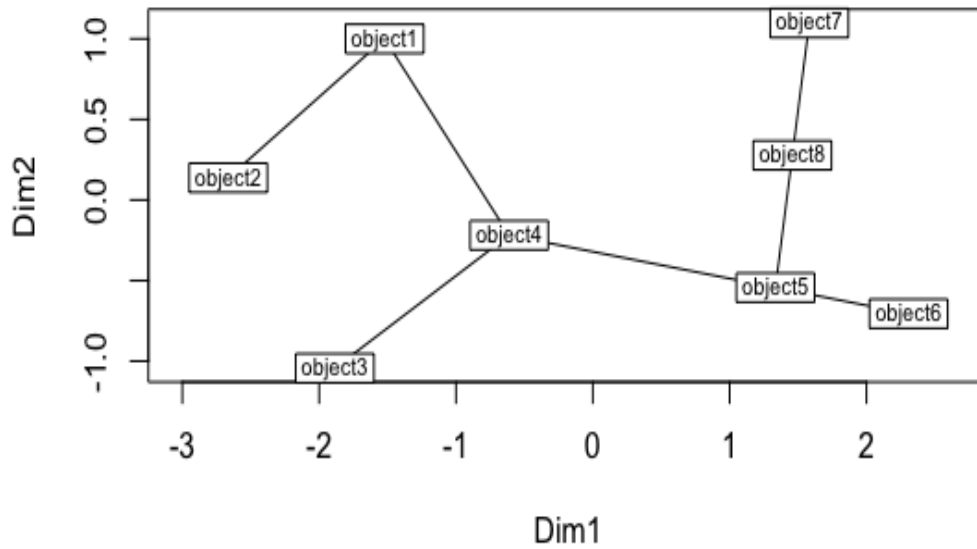


Figure 2.1: An example of a minimum spanning tree.

Step 4:

Merge the objects within each group to form two symbolic objects and calculate the dissimilarity and similarity between these two symbolic objects based on their proposed dissimilarity and similarity measure. If the dissimilarity is larger than the similarity, then the two groups of objects obtained in Step 3 are treated as two clusters. If the dissimilarity is smaller than the similarity, then the two groups of symbolic objects are still treated as one single cluster.

Step 5:

Repeat Step 1 to 4 until there is no group that can be further partitioned by the inconsistent edge (edge with the largest dissimilarity value) within each group.

This algorithm divides the objects into two subsets and subdivides each subset repeatedly until there is no subset that can be further divided by the inconsistent edge of that subset. One disadvantage of this algorithm is that we need to calculate the dissimilarities and similarities and find the MST repeatedly, which is not efficient. The second disadvantage is that Step 4 of this algorithm requires merging the objects within each group to form two symbolic objects but how to merge the objects is not clearly explained in this algorithm. For example, if we merge two intervals by overlapping, then merging $\{[1, 3], [2, 4]\}$ and merging $\{[1, 2], [2, 4]\}$ leads to the same result $[1, 4]$. However, the internal variations of the group $\{[1, 3], [2, 4]\}$ and the group $\{[1, 2], [2, 4]\}$ are different.

One fact that needs to be noticed here is that removing one edge from the MST leads to two disjoint clusters of the objects. Thus, we can use the MST to determine the bipartitions. Suppose we have n objects and a MST for these objects. If we only consider the partitions which result from removing the edges from the MST, the number of possible bipartitions is the number of the edges, i.e., $n - 1$. However, if we consider all the possible bipartitions to find the globally optimal bipartition, the number would be $(2^{n-1} - 1)$, which exponentially increases as the number of observations increases. Thus, the MST can serve as a tool to significantly reduce the number of possible partitions. Furthermore, although we can use some other connected and undirected network to represent the relationships (dissimilarities) between the objects, the MST gives us the underlying structure of the relationship network without redundant connections and it is possible that the globally optimal bipartition can be found in the underlying structure. Therefore, we apply the MST to one of our proposed algorithms.

2.3.2 MONOTHETIC DIVISIVE HIERARCHICAL CLUSTERING ALGORITHMS FOR SYMBOLIC DATA

A monothetic divisive hierarchical clustering method for symbolic data was developed by Chavent (1998, 2000). Chavent (1998, 2000) extended the definitions of the within-cluster

variation and the between-cluster variation from classical quantitative data to symbolic data. Based on the definitions, they proposed the clustering criterion in Eq.(2.17) which is used to choose the cluster to be split at each clustering stage. This clustering criterion is widely used in symbolic divisive clustering algorithms.

For clustering interval-valued data, Chavent's monothetic divisive clustering algorithm gives the intervals an initial order by the midpoints of the intervals at each clustering stage. Let an interval-valued dataset Ω consist of n p -dimensional interval-valued objects $\{Y_u = (y_{u1}, \dots, y_{up}) = ([a_{u1}, b_{u1}), \dots, [a_{up}, b_{up})), u = 1, \dots, n\}$. Let $P_r = (C_1, \dots, C_r)$, $(r > 1)$ represent that Ω be partitioned into r clusters $C_1, \dots, C_r$ after the $(r-1)^{th}$ clustering stage. Chavent's algorithm can be summarized as:

Step 1:

Stage $r = 1$: Start with $\Omega = C_1 = \{Y_1, \dots, Y_n\}$.

Step 2:

Stage r : $\Omega = P_r = (C_1, \dots, C_r)$. For any variable $y_j, j = 1, \dots, p$, and any C_v in $P_r, v = 1, \dots, r$, order the observations in C_v by increasing (or decreasing) midpoint values of y_j . Let n_v be the number of observations in C_v and $\{Y_{(1)j}, \dots, Y_{(n_v)j}\}$ be the observations in C_v after the ordering by the midpoint values of y_j . Then, the possible bipartitions based on this ordering are $\{Y_{(1)j}, \dots, Y_{(m)j}\}$ and $\{Y_{(m+1)j}, \dots, Y_{(n_v)j}\}$, $m = 1, \dots, n_v - 1$.

Step 3:

For each possible bipartition of each cluster C_v in $P_r, v = 1, \dots, r$, by midpoint values of each variable $y_j, j = 1, \dots, p$, calculate the change in the total within-cluster variation Δ_v by Eq.(2.17).

Step 4:

Select the cluster C_v to be partitioned and the position m where C_v is to be partitioned by the maximum Δ_v calculated in Step 3.

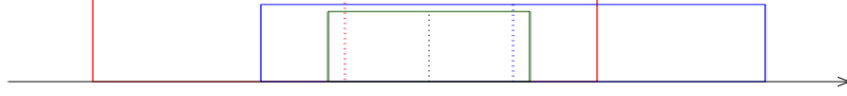


Figure 2.2: An example of a possible limitation of the monotheistic method.

Step 5:

Now, $r = r + 1$. Repeat Steps 2 – 4 until $r = n$ or stop this process by some stopping rules.

One feature of this algorithm is the number of possible bipartition is reduced by ordering the observations by increasing (or decreasing) midpoint values of intervals. Suppose we are at the r^{th} clustering stage and Ω is partitioned into $P_r = (C_1, \dots, C_r)$. Then, the total number of possible partitions from P_r to P_{r+1} is $\sum_{v=1}^r (2^{n_v-1} - 1)$, which exponentially increases as the number of observations increases. However, by giving order to the observations, in total there are $\sum_{v=1}^r p * (n_v - 1) = pn - pr$ possible bipartitions, which is linearly related to the sample size n . Although ordering is necessary, the ordering by mean values (midpoints for intervals) may lead to some problems. For example, in Figure 2.2, we can see three intervals: two long intervals (the blue and the red one) and the narrow green interval. Suppose the dimension $p = 1$. If we order these three intervals by the values of the midpoints (the dash lines), the green interval would always be the second one no matter they were ordered by increasing or decreasing midpoint values. In that case, if the red interval and the blue interval were clustered into the same group, the green one would always be in that group. However, based on the lengths of the intervals, it is highly possible that the blue one and the red one are from the same cluster while the green one is from another cluster.

Another monothetic algorithm is proposed by Pirçon (2004), see also Rasson et al. (2008). For interval-valued data, the algorithm is based on the assumption that the distribution of points in the space $(m, l) \subseteq R \times R^+$ spanned by midpoint values and lengths of intervals can be modelled by a non-homogeneous Poisson process. The difference between this algorithm and Chavent's monothetic algorithm is the clustering criterion used at each clustering stage. Pirçon's algorithm is based on the generalized hypervolumes clustering criterion while Chavent's algorithm is based on the minimum total within-cluster variation criterion described in Section 2.2.2. One similarity is this algorithm still orders the observations by increasing (or decreasing) midpoint values of intervals to find the possible bipartitions and considers the same set of possible bipartitions as Chavent's monothetic algorithm does. Thus, this algorithm has the same limitation which is shown in Figure 2.2 and the same number of possible bipartitions $pn - pr$ at the r^{th} clustering stage as Chavent's monothetic algorithm does. Using midpoint values is beneficial to the explanation of a dendrogram since we can set a binary question based on the midpoints of the interval-valued variables at each partition node of the dendrogram. Thus, the dendrogram can serve as a decision tree. However, as long as the divisive algorithm is monothetic, the dendrogram can serve as a decision tree no matter whether the midpoints are used or not.

One extension of the monothetic algorithm (Chavent (1998, 2000)) is proposed by Brito and Chavent (2012), which is a monothetic divisive clustering algorithm for interval-valued and histogram-valued data. In this algorithm, each observed histogram from a histogram-valued variable is rewritten by the intersections of the subintervals of all the observed histograms from that variable and a uniform distribution is assumed within each subinterval. The difference between this algorithm and the monothetic algorithm proposed by Chavent (1998, 2000) is that this algorithm uses the upper bounds of the subintervals of the rewritten histograms to find the possible bipartitions at each clustering stage instead of using the midpoints (means) of intervals. This algorithm is highly dependent on the form in which the

histograms are rewritten and it requires each interval-valued variable to be rewritten into the form of histograms before the clustering process.

2.3.3 A POLYTHETIC DIVISIVE HIERARCHICAL CLUSTERING ALGORITHM FOR SYMBOLIC DATA

Kim (2009) and Kim and Billard (2011) proposed a polythetic divisive clustering algorithm for symbolic data. Although this algorithm was only applied to some histogram-valued datasets in Kim (2009) and Kim and Billard (2011), it can be directly used to cluster interval-valued data. In Kim (2009) and Kim and Billard (2011), they also proposed two indices for hierarchical clustering which were introduced in Section 2.2.3.

One simplified interpretation of this algorithm is that it first applies an average weighted dissimilarity to find one observation which is “farthest” to the other observations in its cluster in a weighted average sense. Suppose we are at the r^{th} clustering stage and Ω is partitioned into $P_r = (C_1, \dots, C_r)$. Then, the average weighted dissimilarity for an observation Y_u^v , $u = 1, \dots, n_v$, in cluster C_v is defined as

$$\bar{D}_v(Y_u^v) = \frac{1}{\lambda - w_u} \sum_{u_1 \neq u=1}^{n_v} w_u w_{u_1} d(Y_u^v, Y_{u_1}^v), \quad u = 1, \dots, n_v, \quad (2.20)$$

where $d(Y_{u_1}, Y_{u_2})$ is a distance measure for symbolic data, and where n_v is the number of observations in C_v , and where w_u is the weight for Y_u^v with $\lambda = \sum_{u=1}^{n_v} w_u$.

After finding the observation with the largest average weighted dissimilarity, this observation is used as the seed for the partition and the cluster to which the seed belongs is going to be partitioned at this clustering stage. Let Y^* be the seed and C^* be the cluster to which the seed belongs. Then, the algorithm divides C^* into two parts $C^1 = \{Y^*\}$ and $C^2 = C^* - \{Y^*\}$. Each time, the algorithm moves one observation from C^2 to C^1 and computes the sums of the within-cluster variations by Eq.(2.14) before and after the movements, respectively. Then, it is put back to C^2 . The observation which results in the largest decrease of the sum of the within-cluster variations is permanently moved from C^2 to C^1 . Now there are two observations in C^1 and the process is repeated to find the third, fourth and fifth

observations of C^1 until no movement can further reduce the sum of the within-cluster variations. In other words, those observations which are “similar” to the “farthest” observation Y^* are permanently moved from C^2 to C^1 . Thus, C^* is partitioned into C^1 and C^2 at this clustering stage. We move on to the next stage and use Eq.(2.20) to find the next seed and the next C^* to be partitioned. The whole process is repeated until the bottom of the hierarchical tree is reached or the process is stopped by some stopping rules.

Suppose the number of the observations in the cluster to which the seed belongs is n_r^* at the r^{th} clustering stage. Then, there are at most $n_r^*(n_r^* - 1)/2 - 1$ possible bipartitions at the r^{th} clustering stage of this algorithm. Although n_r^* tends to decrease as r increases, n_r^* equals the sample size n at the first clustering stage. Thus, the algorithm may be not very efficient at the first several clustering stages if the sample size is very large.

2.4 PRINCIPAL COMPONENT ANALYSIS METHODOLOGIES FOR INTERVAL-VALUED DATA

2.4.1 PRINCIPAL COMPONENT ANALYSIS AND WEIGHTED PRINCIPAL COMPONENT ANALYSIS

Principal component analysis (PCA) is a popular linear transformation technique which is widely used to analyze data patterns by reducing the dimension of the dataset. The PCA was invented by Pearson (1901). The PCA aims to find a set of linearly uncorrelated combinations of variables which account for as much of the variance of the original data as possible. These linearly uncorrelated combinations of variables are called principal components.

Suppose the data matrix $\mathbf{X}$ consists of n samples (rows) and m variables (columns). We first do zero mean scaling on each variable of $\mathbf{X}$. Depending on the application, we may also choose to do unit variance scaling. After the scaling, we run an eigenvalue decomposition on the sample covariance matrix $\mathbf{X}^T\mathbf{X}$. Then, the loadings (the weights by which the original variables are multiplied to get the principal components) of the first principal component are the elements of the eigenvector corresponding to the largest eigenvalue of $\mathbf{X}^T\mathbf{X}$, the loadings of the second principal component are the elements of the eigenvector corresponding to the

second largest eigenvalue, and so forth. We can use part of these principal components to represent the sample matrix $\mathbf{X}$. In other words, we project the samples onto a new coordinate system where each axis is a principal component. In that new system, the dimension is lower, but the variations between different samples are more obvious than those of the original coordinate system, which is an important merit for feature selections in clustering methods.

Many researchers have extended classical PCA to weighted PCA to solve different tasks such as Yue and Tomoyasu (2004), Tamuz et al. (2005), Agahian et al. (2008), Da Costa et al. (2009) and Delchambre (2014). There are many ways to weight a PCA model. For example, we can assign weights to different samples (observations), variables or even elements (cells) of the data matrix. In this research, we focus on the sample-wise weighted PCA.

Take the data matrix $\mathbf{X}$. Suppose each observation x_i has an associated weight w_i and assume $\sum_i w_i = 1$. We first do zero mean scaling on each variable of $\mathbf{X}$ and then perform an eigenvalue decomposition on the weighted covariance matrix $\mathbf{X}^T \mathbf{W}^T \mathbf{W} \mathbf{X}$ where $\mathbf{W}$ is the diagonal weighting matrix with w_i as the diagonal elements. Then, the k^{th} principal component of the weighted PCA is $\mathbf{Y}_k = \mathbf{W} \mathbf{X} \mathbf{v}_k$ where $\mathbf{v}_k$ is the eigenvector corresponding to the k^{th} largest eigenvalue from the eigenvalue decomposition.

2.4.2 CURRENT PCA METHODOLOGIES FOR INTERVAL-VALUED DATA

Some PCA methods for interval-valued data have been developed by researchers. Two intuitive methods are the centers and the vertices methods proposed by Cazes et al. (1997), see also Chouakria et al. (2000) and Billard et al. (2011). The basic idea of the centers method is to run a classical PCA on the centers of the intervals of the observations, which does not consider the lengths of intervals. The vertices method is to run the classical PCA on the vertices of the interval-valued observations, which can be seen as the vertices of some hyper-rectangles. Because these vertices include the information of both the center and the length of each interval, the vertices method accounts for the variation in the data and gives us more accurate PCA results. Lauro and Palumbo (2000) developed a symbolic object principal

component analysis (SO-PCA), a principal component analysis on the range transformation (RT-PCA) of interval variables and a mixed strategy combining SO-PCA and RT-PCA. They also developed a midpoints and radii method (Palumbo and Lauro (2003)), in which the midpoints and the ranges of intervals are not treated together as one single interval-valued variable but two separate variables. Gioia and Lauro (2006) applied interval algebra to the PCA for interval-valued data. Le-Rademacher (2008) proposed the symbolic covariance PCA based on the symbolic sample covariance matrix. Le-Rademacher and Billard (2012) gave a detailed review of the above PCA methods and pointed out their deficiencies. There are also some other symbolic PCA methods which can be applied to interval-valued data such as the quantile method for symbolic principal component analysis (Ichino (2001)), the complete-information-based principal component analysis (CIPCA) proposed by Wang et al. (2012), and the probabilistic symbolic principal component analysis (PSPCA) proposed by Chen et al. (2015).

Classical PCA takes the eigenvectors of the sample covariance matrix as the coefficients of the linear combinations of variables (principal components) to account for as much of the variability in the data as possible. One natural extension from classical PCA to symbolic PCA is the symbolic covariance PCA method developed by Le-Rademacher (2008) and Le-Rademacher and Billard (2012) since this method follows the way of the classical PCA to obtain the principal components from the eigenvectors of the sample covariance matrix. In symbolic data analysis, multivariate interval-valued observations are represented by hyper-rectangles. However, these hyper-rectangles may not keep their shapes in principal components spaces. Le-Rademacher (2008) and Le-Rademacher and Billard (2013) proposed using some polytopes to represent the projections of the hyper-rectangles. They project the polytopes onto some principal components planes and get some polygons for visualization of the principal components. Furthermore, they divide the polygons into small pieces and calculate the area of each piece to get a histogram-valued representation of each principal component. As a result, their works mainly focus on constructing the true structures of

the observations in principal components spaces. However, in this dissertation, what we care more about are the loadings of principal components and the component scores (the elements of the principal components) since they are used to give initial orders to the observations in our proposed hierarchical clustering algorithms.

CHAPTER 3

MONOTHETIC DIVISIVE CLUSTERING FOR INTERVAL-VALUED DATA BASED ON SYMBOLIC PRINCIPAL COMPONENT ANALYSIS AND ENDPOINTS OF INTERVALS

In contrast to polythetic divisive algorithms for which all variables are used simultaneously for finding the bipartition at each clustering stage, monothetic divisive algorithms only choose one variable at a time to find the bipartition. We can set a binary question at each bipartition node of the dendrogram based on the chosen variable to clearly explain the bipartition criterion at each node. Therefore, the dendrogram from a monothetic algorithm can serve as a decision support tool like a decision tree. In this chapter, we focus on monothetic divisive clustering algorithms for interval-valued data.

In Section 3.1, we first propose three monothetic divisive clustering algorithms for interval-valued data and a weighted symbolic principal component analysis (PCA) method for interval-valued data. The first algorithm is based on the symbolic covariance PCA proposed by Le-Rademacher (2008) and Le-Rademacher and Billard (2012). The second algorithm is based on our proposed weighted symbolic PCA method. The third one is based on the endpoints of intervals. Furthermore, we propose two mixed-strategy algorithms which are based on not only the two symbolic PCA method mentioned above but also on the endpoints of intervals. In Section 3.2, the five proposed monothetic algorithms are evaluated and compared with the monothetic divisive algorithm proposed by Chavent (1998, 2000) in a series of simulations. The simulation results show that the two proposed mixed-strategy algorithms outperform all the other monothetic algorithms in clustering accuracies. In Section 3.3, the two mixed-strategy algorithms are applied to real-world datasets and the performances of these two algorithms are summarized.

3.1 MONOTHETIC DIVISIVE CLUSTERING ALGORITHMS

In divisive clustering, we start with the whole data set as a single cluster and then proceed to recursively divide a cluster into two sub-clusters. A big question is which particular cluster is to be divided and then how do we divide that particular cluster into two sub-clusters. If we have a partition $P_r = (C_1, \dots, C_r)$ and suppose the size of each cluster is $n_v, v = 1, \dots, r$, the number of possible bipartitions of P_r is $\sum_{v=1}^r (2^{n_v-1} - 1)$, which exponentially increases as the number of observations increases. Therefore, it may not be computationally efficient to consider all the bipartitions in many cases. To reduce the total number of possible bipartitions to be considered, one solution is to give an ordering to the observations. Then, we only need to consider the $(n_v - 1), v = 1, \dots, r$, possible bipartitions of the ordered observations in $C_v, v = 1, \dots, r$; so the number of possible bipartitions of P_r is reduced to $\sum_{v=1}^r (n_v - 1) = n - r$.

For multimodal-valued data, an ordering criterion is based on an association measure (Kim and Billard (2012)). For histogram-valued data, a double ordering criterion is based on the internal standard deviation and the internal mean of each histogram (Kim and Billard (2018)). For interval realizations, one ordering criterion is based on midpoint values of intervals (Chavent (1998, 2000)). However, If we only consider the midpoints of intervals, we lose the information of the lengths of intervals so we have the possible limitations on the clustering as were discussed in Section 2.3.2. The situation in Figure 2.2 usually happens when an interval-valued hyper-rectangle is overlapped with or even covered by some other hyper-rectangles in some dimensions.

Instead of ordering interval-valued observations by the midpoints of intervals, we propose five monothetic algorithms which order the interval-valued observations by some other ordering criteria. In Section 3.1.1, we propose an ordering criterion for interval-valued data based on the symbolic covariance principal component analysis method which was proposed by Le-Rademacher (2008) and Le-Rademacher and Billard (2012). In Section 3.1.2, we propose a weighted symbolic principal component analysis method for interval-valued data and

an ordering criterion based on this method. In Section 3.1.3, we propose two divisive clustering algorithms based on the two ordering criteria described in Section 3.1.1 and Section 3.1.2, respectively. In Section 3.1.4, one double monothetic divisive clustering algorithm in which observations are ordered by increasing values of either the upper endpoints or the lower endpoints of the intervals of each variable is proposed. In Section 3.1.5, we propose two mixed-strategy algorithms which are combinations of the two PCA-based algorithms proposed in Section 3.1.3 and the endpoints-based algorithm proposed in Section 3.1.4.

3.1.1 ORDERING CRITERION BASED ON THE SYMBOLIC COVARIANCE PRINCIPAL COMPONENT ANALYSIS METHOD

Some methods have been developed to extend classical principal component analysis (PCA) to symbolic principal component analysis (SPCA). A brief review can be found in Section 2.4. One SPCA method which extends PCA from classical data to interval-valued data is the symbolic covariance PCA method proposed by Le-Rademacher (2008) and Le-Rademacher and Billard (2012). It is a natural extension from classical PCA to SPCA since this method follows the way of the classical PCA to obtain the principal components from the eigenvectors of the sample covariance matrix. Furthermore, Le-Rademacher (2008) developed a method to construct the true structures of the projections of interval-valued observations onto the principal components space, i.e., the true structures of the principal components. They found that the multivariate interval-valued observations which are represented by hyper-rectangles in their original space can no longer keep their shapes in the principal components space. Thus, Le-Rademacher (2008) and Le-Rademacher and Billard (2012, 2013) proposed using some polytopes to represent the true structures of the principal components. Besides, the polytopes are projected onto some principal components planes to obtain some polygons for better visualization of the principal components.

The symbolic covariance PCA method relies on the matrix of vertices from the vertices PCA method (Cazes et al. (1997)) to construct the p -dimensional polytopes which represent

the projections of the interval-valued objects onto the principal components space. However, the construction of the polytopes involves a lot of computation and cannot be directly used to order the objects in the principal components space for the purpose of hierarchical clustering. In this dissertation, we still use the same idea as the symbolic covariance PCA method which obtains the principal components loadings from the eigenvectors of the symbolic sample covariance matrix. However, instead of projecting the whole hyper-rectangles, we project the centers of hyper-rectangles. We project the centers onto a certain one-dimensional principal components space (the line or axis to represent a certain principal component) based on the loadings of this principal component. The projections are the component scores for the hyper-rectangle centers (one score for one hyper-rectangle center). Since one hyper-rectangle corresponds to one original observation, we can give an initial ordering criterion to the original observations by increasing or decreasing component score values during the hierarchical clustering process. Figure 3.1 is an example in which the centers C_1 and C_2 of two hyper-rectangles are projected onto a one-dimensional principal components space (the blue line) to obtain two component scores P_1 and P_2 .

Let an interval-valued dataset consist of n p -dimensional interval-valued observations $\{Y_u = ([a_{u1}, b_{u1}), \dots, [a_{up}, b_{up})], u = 1, \dots, n\}$. The algorithm to obtain the projection values (component scores) of the centers is:

Step 1:

Use Eq.(2.2) and Eq.(2.6) to calculate the variances and covariances of the variables, respectively, and thus obtain the symbolic sample covariance matrix $\mathbf{S}$. Furthermore, use Eq.(2.7) to obtain the sample correlation matrix $\mathbf{R}$.

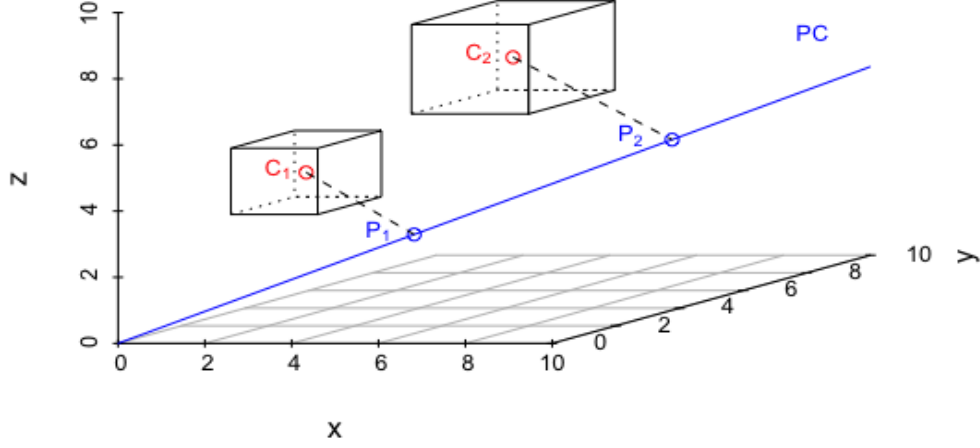


Figure 3.1: Projections of the centers of hyper-rectangles onto a principal component line.

Step 2:

Obtain the matrix of centers $\mathbf{X}^c$, i.e.,

$$\mathbf{X}^c = \begin{bmatrix} X_{11}^c & X_{12}^c & \dots & X_{1p}^c \\ X_{21}^c & X_{22}^c & \dots & X_{2p}^c \\ \vdots & \vdots & \vdots & \vdots \\ X_{n1}^c & X_{n2}^c & \dots & X_{np}^c \end{bmatrix} \quad (3.1)$$

where $X_{uj}^c = (a_{uj} + b_{uj})/2$. Each row of $\mathbf{X}^c$ represents the center of each hyper-rectangle.

Step 3:

We first do zero mean scaling on each variable of $\mathbf{X}^c$. Depending on the application, we may also choose to do unit variance scaling.

Step 4:

Perform an eigenvalue decomposition on the sample covariance matrix $\mathbf{S}$; if unit variance scaling is performed in Step 3, the decomposition should be performed on the sample correlation matrix $\mathbf{R}$ here.

Step 5:

Then calculate the projection values of the centers onto the k^{th} principal component axis (the k^{th} principal components for the centers) by the formula $\mathbf{Y}_k^c = \mathbf{X}^c \mathbf{v}_k^c$, where $\mathbf{v}_k^c$ is the eigenvector corresponding to the k^{th} largest eigenvalue from the eigenvalue decomposition in Step 4.

By Step 5, we obtain the projection values of the centers onto the k^{th} principal component axis. Then, we can give an initial ordering criterion to the interval-valued observations by increasing or decreasing projection values during the hierarchical clustering process. Details of how to find the polytopes which are the whole projections of the interval-valued observations onto the principal components space can be found in Le-Rademacher (2008) and Le-Rademacher and Billard (2012).

3.1.2 A WEIGHTED SYMBOLIC PRINCIPAL COMPONENT ANALYSIS METHOD AND THE CORRESPONDING ORDERING CRITERION

Another SPCA method which extends PCA from classical data to interval-valued data is the centers method proposed by Cazes et al. (1997), see also Chouakria et al. (2000) and Billard et al. (2011). The basic idea of the centers method is to run a classical PCA on the centers of the interval-valued objects or on the centers (midpoints) of intervals. The method assumes the projections of the interval-valued objects (the hyper-rectangles) onto the principal components space are also hyper-rectangles, i.e., each principal component of each object is an interval. One disadvantage of this method is the lengths of intervals are not considered during the analysis process and thus leads to information loss. Although this method suffers from loss of information, it is actually much more efficient than other SPCA

methods for interval-valued data. In the hierarchical clustering process of a large dataset, efficiency is an aspect which cannot be ignored since hierarchical clustering process involves some repetitive calculations and nested loops. Thus, the second SPCA method which we apply in our divisive algorithms is based on the centers method. However, instead of directly running PCA on the centers of intervals as in the centers method, we propose a weighted centers method which assigns a weight to each object based on the lengths of the intervals of each object and then runs a sample-wise weighted PCA on the centers. In other words, the proposed weighted centers method not only takes the centers of intervals into account, but also the lengths of intervals into account. It can be imagined that an interval is well-defined in the real line if we know the information about the length and the center of the interval. Thus, we choose to use the center and a weight based on the length to represent each interval and run a weighted PCA. Furthermore, this method is not much more complicated in its calculation than is the centers method since it is a weighted version of the centers method.

First, we illustrate how to assign the weights. Let an interval-valued dataset consist of n p -dimensional interval-valued observations $\{Y_u = ([a_{u1}, b_{u1}), \dots, [a_{up}, b_{up})], u = 1, \dots, n\}$. For the j^{th} variable, the observations are described by the set of intervals $Z = \{[a_{uj}, b_{uj}), u = 1, \dots, n\}$, and we suppose that the joint set of the upper bounds and the lower bounds of these intervals contain z different bound values. Then, the joint set can be written as $\{c_k, k = 1, \dots, z\} = \{a_{uj}, u = 1, \dots, n\} \cup \{b_{uj}, u = 1, \dots, n\}$. We can stack these n intervals from the set Z into one single histogram $\{[c_k, c_{k+1}), n_k; k = 1, \dots, z - 1\}$ where the frequency n_k is the number of the intervals from the set Z which contain the subinterval $[c_k, c_{k+1})$. Figure 3.2 is an example of how to stack three intervals into a single histogram. We stack the green, blue and red intervals into a histogram $\{[c_1, c_2), n_1 = 1; [c_2, c_3), n_2 = 2; [c_3, c_4), n_3 = 3; [c_4, c_5), n_4 = 2; [c_5, c_6), n_5 = 1\}$. The reason why $n_3 = 3$ is that all the three colored intervals contain the subinterval $[c_3, c_4)$.

We can use the area of the corresponding rectangle in the histogram to define the weight for the subinterval $[c_k, c_{k+1})$, which is $n_k * (c_{k+1} - c_k)$. Then, the weight w_{uj} for an interval

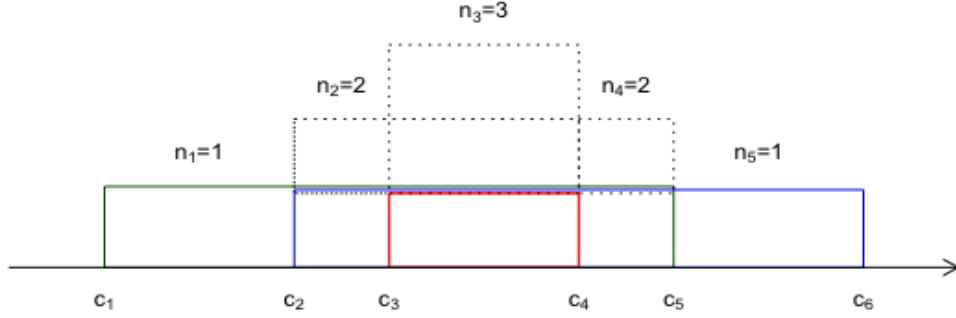


Figure 3.2: An example of how to stack intervals into a histogram.

$[a_{uj}, b_{uj})$ from the set Z is defined as

$$w_{uj} = \sum_k n_k * (c_{k+1} - c_k), \quad a_{uj} \leq c_k \leq c_{k+1} \leq b_{uj}. \quad (3.2)$$

In other words, the weight for an interval from the set Z is the sum of the weights for the subintervals of this interval. We can imagine that if an interval is far from the other intervals, each subinterval of this interval (most probably just itself) will be assigned a low frequency and thus a low weight. Then, this interval will be given a low weight. Therefore, we tend to assign low weights to those intervals which are far from others and assign high weights to those intervals which are frequently overlapped with other intervals. Besides, we tend to assign low weights to those narrow intervals since narrow intervals usually contain fewer or narrower subintervals and we use the length of each subinterval of those narrow intervals to multiply its frequency to obtain the weight for each subinterval.

Since for a given variable each observation is an interval, we can use w_{uj} , $u = 1, \dots, n$, $j = 1, \dots, p$, to denote the weight for the j^{th} variable of the u^{th} observation Y_u and obtain w_{uj} from Eq.(3.2). Besides, the variables may be measured on different scales, which leads to w_{uj} on different scales for different j . Therefore, we may choose to do unit variance scaling on

each of the vectors $(w_{1j}, \dots, w_{nj}), j = 1, \dots, p$. After the scaling, we use the obtained w_{uj} to define the weight for Y_u , which is

$$w_u = \sum_j w_{uj} / \sum_u \sum_j w_{uj}. \quad (3.3)$$

Now we have the weight for each interval-valued observation based on the lost information. Then, we can run a sample-wise weighted PCA to obtain the projection of the center of each observation. Same as the ordering criterion based on the symbolic covariance PCA, we use the projections of the centers onto a given principal component axis to order the observations.

Let an interval-valued dataset consist of n p -dimensional interval-valued observations $\{Y_u = ([a_{u1}, b_{u1}), \dots, [a_{up}, b_{up})], u = 1, \dots, n\}$. The weighted symbolic PCA method is based on six steps:

Step 1:

Use Eq.(3.3) to calculate the weights $w_1, \dots, w_n$ for these n interval-valued observations. Let $\mathbf{W}$ be the diagonal weighting matrix with $w_u, u = 1, \dots, n$, as the diagonal elements.

Step 2:

Obtain the matrix of centers $\mathbf{X}^c$, i.e.,

$$\mathbf{X}^c = \begin{bmatrix} X_{11}^c & X_{12}^c & \dots & X_{1p}^c \\ X_{21}^c & X_{22}^c & \dots & X_{2p}^c \\ \vdots & \vdots & \vdots & \vdots \\ X_{n1}^c & X_{n2}^c & \dots & X_{np}^c \end{bmatrix}$$

where $X_{uj}^c = (a_{uj} + b_{uj})/2$. Further, use $\bar{X}_j^c = \frac{1}{n} \sum_{u=1}^n X_{uj}^c$ to denote the mean of the j^{th} variable (column) of $\mathbf{X}^c$.

Step 3:

We first do zero mean scaling on each variable of $\mathbf{X}^c$. Depending on the application, we may also choose to do unit variance scaling.

Step 4:

Perform an eigenvalue decomposition on the weighted sample covariance matrix $\mathbf{X}^{cT}\mathbf{W}^T\mathbf{W}\mathbf{X}^c$.

Step 5:

Calculate the k^{th} principal components for the centers of the objects by the formula $\mathbf{Y}_k^* = \mathbf{W}\mathbf{X}^c\mathbf{v}_k^c$, where $\mathbf{v}_k^c = (v_{1k}^c, \dots, v_{pk}^c)$ is the eigenvector corresponding to the k^{th} largest eigenvalue from the eigenvalue decomposition in Step 4.

Step 6:

Obtain the k^{th} principal component for Y_u . The k^{th} principal component is an interval $[\underline{y}_{uk}, \overline{y}_{uk}]$ where

$$\begin{aligned}\underline{y}_{uk} &= w_u \sum_{j:v_{jk}^c < 0} (b_{uj} - \bar{X}_j^c)v_{jk}^c + w_u \sum_{j:v_{jk}^c > 0} (a_{uj} - \bar{X}_j^c)v_{jk}^c, \\ \overline{y}_{uk} &= w_u \sum_{j:v_{jk}^c < 0} (a_{uj} - \bar{X}_j^c)v_{jk}^c + w_u \sum_{j:v_{jk}^c > 0} (b_{uj} - \bar{X}_j^c)v_{jk}^c.\end{aligned}$$

By Step 5, we obtain the projection values of the centers of the interval-valued objects onto the k^{th} principal component axis. Then, we can give an initial ordering criterion to the interval-valued objects by increasing or decreasing projection values during the hierarchical clustering process. By Step 6, we obtain the projection of each object onto the k^{th} principal component axis, i.e., the k^{th} principal component for each object.

3.1.3 DIVISIVE CLUSTERING ALGORITHMS BASED ON TWO ORDERING CRITERIA

In Section 3.1.1 and Section 3.1.2, we proposed two ordering criteria. One criterion is based on the projection values of the centers by the symbolic covariance PCA and the other one is

based on the projection values of the centers by the weighted symbolic PCA. In this section, we propose two divisive clustering algorithms for interval-valued objects based on these two criteria, respectively.

To obtain the projection values, the centers are projected onto the k^{th} principal component axis. However, the k^{th} principal component axis could be any principal component axis. Thus, one of our choices is to use the projections onto the first principal component axis to order the observations. The reason why we use the first principal component axis is that the first principal component accounts for the largest possible variation in the data. By ordering the observations by increasing (or decreasing) projection values of centers onto the first principal component axis, we give a reasonable initial order to the observations by a single synthetic variable (the first principal component of each center), which is more efficient than considering all the variables and more precise than just considering a single variable in the original dataset.

Another choice is to project the centers onto multiple principal component axes to obtain multiple ordering criteria for the objects and compare all the possible bipartitions caused by the criteria at each clustering stage. However, we can control the number of principal component axes to be projected onto by a given cumulative contribution rate of principal components (the cumulative proportion of variance explained by the principal components). For example, if we want a 90% cumulative contribution rate and the cumulative contribution rates of the first two principal components and the first three principal components are respectively 85% and 92%, we select the first three principal components axes and use them to give three different initial orders to the observations. If we set a fairly small cumulative contribution rate, it is equivalent to only using the first principal component axis.

Let an interval-valued dataset Ω consist of n p -dimensional interval-valued objects $\{Y_u = ([a_{u1}, b_{u1}), \dots, [a_{up}, b_{up})], u = 1, \dots, n\}$. Let $P_r = (C_1, \dots, C_r), (r > 1)$, represent that Ω is partitioned into r clusters $C_1, \dots, C_r$ after the $(r - 1)^{th}$ stage of the divisive hierarchical clustering algorithm. The algorithm involves the following steps:

Step 1:

Stage $r = 1$: Start with $\Omega = C_1 = \{Y_1, \dots, Y_n\}$.

Step 2:

Run a symbolic PCA method described in Section 3.1.1 or Section 3.1.2 on the centers of the interval-valued objects and obtain all the p principal components for the centers $\{\mathbf{Y}_1^*, \dots, \mathbf{Y}_p^*\}$.

Step 3:

Based on the contribution proportion of each principal component from the PCA results in Step 2 and a given cumulative contribution rate λ , determine k which is the number of principal components to be used by finding the smallest k such that the cumulative contribution rate of the first k principal components reaches λ . Then, the selected principal components for the centers are $\{\mathbf{Y}_1^*, \dots, \mathbf{Y}_k^*\}$.

Step 4:

Stage r : $\Omega = P_r = (C_1, \dots, C_r)$. Each vector $\mathbf{Y}_i^*$, $i = 1, \dots, k$, consists of n elements and each element is the i^{th} principal component value for the center of each object. For any $\mathbf{Y}_i^*$, $i = 1, \dots, k$, and any C_v in P_r , $v = 1, \dots, r$, order the objects in C_v by increasing (or decreasing) principal component values of $\mathbf{Y}_i^*$. Let n_v be the number of objects in C_v and $\{Y_{(1)i}, \dots, Y_{(n_v)i}\}$ be the objects in C_v after the ordering by the principal component values of $\mathbf{Y}_i^*$. Then, the possible bipartitions of C_v based on this ordering are $(\{Y_{(1)i}, \dots, Y_{(m)i}\}, \{Y_{(m+1)i}, \dots, Y_{(n_v)i}\})$, $m = 1, \dots, n_v - 1$.

Step 5:

For each possible bipartition of each cluster C_v in P_r , $v = 1, \dots, r$, ordered by the principal component values of each $\mathbf{Y}_i^*$, $i = 1, \dots, k$, calculate the change in the total within-cluster variation Δ_v by Eq.(2.17).

Step 6:

Select the cluster C_v to be partitioned and the position m where C_v is to be partitioned by the maximum Δ_v calculated in Step 5.

Step 7:

Now, $r = r + 1$. Repeat Steps 4 – 6 until $r = n$, the total number of objects, or stop this process by some stopping rules.

Depending on the symbolic PCA method used in Step 2, we obtain the divisive algorithm based on the symbolic covariance PCA and the divisive algorithm based on the weighted symbolic PCA. Other symbolic PCA methods for interval-valued data can also be applied in Step 2 to obtain different algorithms. Besides, these two proposed divisive algorithms are monothetic since we use one synthetic variable (one principal component) at a time to conduct the bipartition at each stage. Furthermore, our proposed weighted symbolic PCA method takes the lengths of intervals into consideration since the weights are dependent on the lengths of intervals. The symbolic covariance PCA method also takes the lengths of intervals into consideration since the calculation of the symbolic sample covariance matrix automatically involves the information about the lengths of intervals. Last but not least, we use the cumulative contribution rate of the principal components to decide how many principal components to be kept. Suppose we keep n_{pc} principal components based on the pre-specified cumulative contribution rate λ . Then, we only need to consider $\sum_{v=1}^r n_{pc}(n_v - 1) = n_{pc}(n - r)$, ($n_{pc} \leq p$), possible bipartitions at Stage r .

3.1.4 A DIVISIVE CLUSTERING ALGORITHM BASED ON THE ENDPOINTS OF INTERVALS

The monothetic divisive algorithm proposed by Chavent (1998, 2000) gives the interval-valued objects initial orders by the midpoints of intervals. Our proposed algorithms in Section 3.1.3 give the interval-valued objects initial orders by the projections of the centers onto the principal component axes, i.e., linear combinations of the midpoints of intervals. Thus, these algorithms are all based on the midpoints of intervals and may have the same limitation

shown in Figure 2.2 when some interval-valued hyper-rectangles are overlapped with some other hyper-rectangles in some dimensions.

For histogram-valued data, Kim and Billard (2018) proposed a divisive algorithm with a double ordering criterion. The double ordering criterion is based on the standard deviation and the mean of each histogram. However, the standard deviations and the means of histograms only contain the information about the first two moments of histograms. For intervals, if we know the information about both the lengths and the midpoints, which is equivalent to knowing the information about the two endpoints, we can obtain not only the first two moments but also higher moments of intervals. This is the first reason why we decide to use the information about the endpoints to order the observations in our divisive monothetic algorithm. The second reason is that endpoints work better than midpoints in differentiating different clusters of intervals with dissimilar lengths. Since monothetic algorithms only use one variable at a time for the bipartition at each clustering stage, we can compare the performance of endpoints with the performance of midpoints in differentiating intervals from different clusters in some one-dimensional cases.

In Figure 3.3, there are two intervals (the blue interval A and the red interval B) from two different clusters and with dissimilar lengths. They move toward each other along the axis, which leads to six different types of relative positions. In scenario (a) and scenario (f), the two intervals are well-separated. The two intervals are partially overlapped in scenario (b) and scenario (e) while one interval is covered by the other one in scenario (c) and scenario (d). Suppose the length of interval B is l_b and the length of interval A is l_a ($l_b > l_a$). Then, the distances between the upper or lower endpoints of these two intervals (d_u or d_l) and the distance between the midpoints of these two intervals (d_m) have the following relationships:

1. In scenarios (a), (b) and (c), $d_u > d_m$ and $d_u - d_m = (l_b - l_a)/2$.
2. In scenarios (d), (e) and (f), $d_l > d_m$ and $d_l - d_m = (l_b - l_a)/2$.

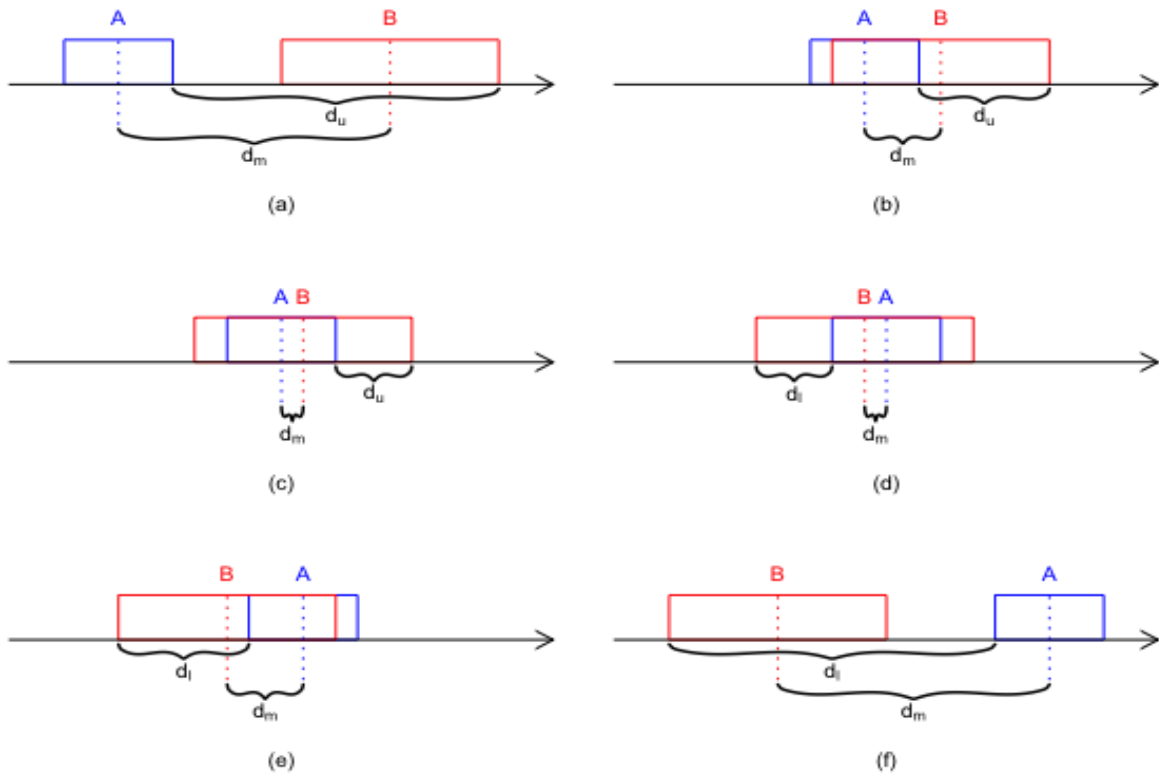


Figure 3.3: Six scenarios when two intervals move toward each other along the axis.

Thus, no matter in which scenario, one of d_u and d_l is always larger than d_m . In other words, if we consider both the difference between the upper endpoints and the difference between the lower endpoints of the intervals from different clusters, we will achieve a clearer clustering result than just considering the difference between the midpoints. The possible limitation of Chavent's algorithm shown in Figure 2.2 can also be easily solved if the three intervals in Figure 2.2 are ordered by increasing or decreasing endpoints values. Thus, in divisive clustering, the ordering criteria based on the upper and lower endpoints may give us better possible bipartitions at each clustering stage than the criteria based on the midpoints when the lengths of intervals from different clusters are distinguishable. Therefore, we propose a monothetic divisive algorithm using a double ordering criterion based on the upper endpoint and the lower endpoint of each interval.

Let an interval-valued dataset Ω consist of n p -dimensional interval-valued objects $\{Y_u = ([a_{u1}, b_{u1}), \dots, [a_{up}, b_{up})], u = 1, \dots, n\}$. Let $P_r = (C_1, \dots, C_r), (r > 1)$, represent that Ω is partitioned into r clusters $C_1, \dots, C_r$ after the $(r - 1)^{th}$ stage of the divisive hierarchical clustering algorithm. The algorithm involves the following steps:

Step 1:

Stage $r = 1$: Start with $\Omega = C_1 = \{Y_1, \dots, Y_n\}$.

Step 2:

Stage r : $\Omega = P_r = (C_1, \dots, C_r)$. For any variable $y_j, j = 1, \dots, p$, and any C_v in $P_r, v = 1, \dots, r$, order the observations in C_v by increasing (or decreasing) values of b_{uj} . Let n_v be the number of observations in C_v and $\{Y_{(1)j}^b, \dots, Y_{(n_v)j}^b\}$ be the observations in C_v after the ordering by b_{uj} . Then, the possible bipartitions of C_v based on this ordering are $(\{Y_{(1)j}^b, \dots, Y_{(m)j}^b\}, \{Y_{(m+1)j}^b, \dots, Y_{(n_v)j}^b\}), m = 1, \dots, n_v - 1$.

Step 3:

For any variable $y_j, j = 1, \dots, p$, and any C_v in $P_r, v = 1, \dots, r$, order the observations in C_v by increasing (or decreasing) a_{uj} values of variable y_j to obtain the possible

bipartitions $(\{Y_{(1)j}^a, \dots, Y_{(m)j}^a\}, \{Y_{(m+1)j}^a, \dots, Y_{(n_v)j}^a\})$, $m = 1, \dots, n_v - 1$, of C_v in the same way as in Step 2.

Step 4:

For each possible bipartition of each cluster C_v in P_r , $v = 1, \dots, r$, based on the lower endpoints a_{uj} and the upper endpoints b_{uj} of each variable y_j , $j = 1, \dots, p$, i.e., each possible bipartition obtained in Step 2 and Step 3, calculate the change in the total within-cluster variation Δ_v by Eq.(2.17).

Step 5:

Select the cluster C_v to be partitioned and the position m where C_v is to be partitioned by the maximum Δ_v calculated in Step 4.

Step 6:

Now, $r = r + 1$. Repeat Steps 2 – 5 until $r = n$ or stop this process by some stopping rules.

Since this algorithm applies a double ordering criterion based on the upper endpoint and the lower endpoint of each interval, there are $2 \sum_{v=1}^r p(n_v - 1) = 2p(n - r)$ possible bipartitions in total at Stage r .

3.1.5 DIVISIVE CLUSTERING ALGORITHMS BASED ON BOTH ENDPOINTS OF INTERVALS AND SYMBOLIC PRINCIPAL COMPONENT ANALYSIS

In Section 3.1.4, we proposed a divisive algorithm which applies a double ordering criterion based on the upper endpoint and the lower endpoint of each interval. In other words, the algorithm is based on the coordinates of two vertices of each interval-valued object (hyper-rectangle), i.e., the vertex with the smallest coordinate on each axis and the vertex with the largest coordinate on each axis. Based on the discussion in Section 3.1.4, we know this algorithm may outperform other midpoints-based algorithms in differentiating clusters of intervals with dissimilar lengths. However, two different clusters of intervals may have similar

lengths but dissimilar midpoint values, or even worse, similar lengths and similar midpoint values. In classical data analysis, PCA is a commonly-used technique to make similar observations distinguishable by projecting them onto the principal components axes. In Section 3.1.3, we proposed two divisive algorithms based on symbolic PCA, in which the centers of interval-valued objects are projected onto the principal component axes. Thus, these two algorithms may work better in differentiating clusters of intervals with similar lengths.

To cover both of the situations when the lengths of intervals are dissimilar or similar, we propose two mixed-strategy algorithms which are combinations of the two PCA-based algorithms proposed in Section 3.1.3 and the endpoints-based algorithm proposed in Section 3.1.4. Instead of projecting the centers of interval-valued objects onto the principal components axes, the mixed strategy projects two vertices of each interval-valued object, i.e., the vertex V_s with the smallest coordinate on each axis and the vertex V_l with the largest coordinate on each axis. The projection values are actually some linear combinations of the endpoint values of the one-dimensional intervals of each object. After obtaining the projection values of the V_s 's of the objects onto a principal component axis, we can give an initial ordering criterion to the interval-valued objects by increasing or decreasing order of these projection values during the hierarchical clustering process. We also use the projection values of the V_l 's of the objects to give another initial ordering criterion. Thus, the mixed strategy is also a kind of algorithm with a double ordering criterion. Figure 3.4 is an example in which the V_s 's and V_l 's of two hyper-rectangles are projected onto a principal component axis. The mixed strategy also has two versions, one based on the symbolic covariance PCA method proposed by Le-Rademacher (2008) and Le-Rademacher and Billard (2012) and the other one based on the weighted symbolic PCA method proposed in Section 3.1.2.

Let an interval-valued dataset Ω consist of n p -dimensional interval-valued objects $\{Y_u = ([a_{u1}, b_{u1}), \dots, [a_{up}, b_{up})], u = 1, \dots, n\}$. The vertex V_{su} of Y_u with the smallest coordinate on each axis and the vertex V_{lu} of Y_u with the largest coordinate on each axis are defined as $V_{su} = (a_{u1}, \dots, a_{up})$ and $V_{lu} = (b_{u1}, \dots, b_{up})$. Let $P_r = (C_1, \dots, C_r), (r > 1)$, represent that Ω

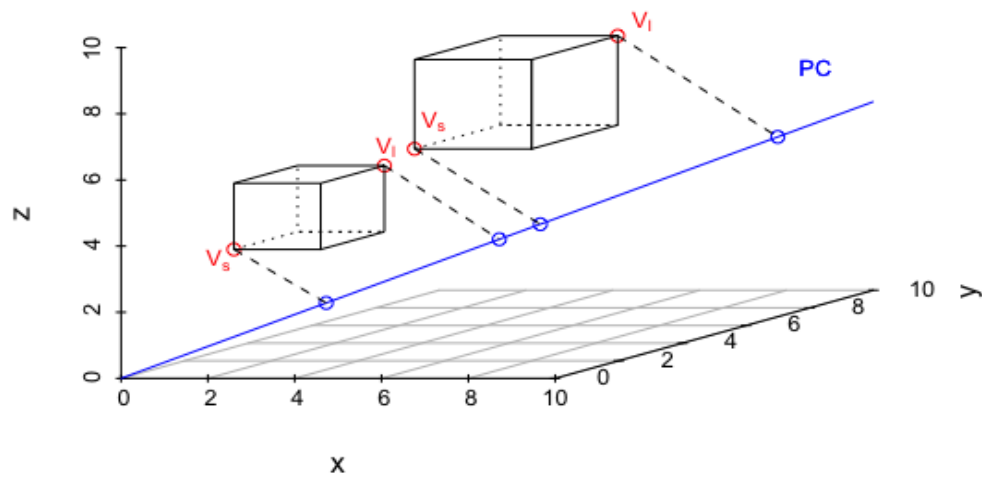


Figure 3.4: Projections of the V_s 's and V_l 's of two hyper-rectangles onto a principal component line.

is partitioned into r clusters $C_1, \dots, C_r$ after the $(r-1)^{th}$ stage of the divisive hierarchical clustering algorithm. The algorithm involves the following steps:

Step 1:

Stage $r = 1$: Start with $\Omega = C_1 = \{Y_1, \dots, Y_n\}$.

Step 2:

Run a symbolic PCA method described in Section 3.1.1 or Section 3.1.2 on the $V_{su}, u = 1, \dots, n$, i.e., replace all the $\mathbf{X}^c$'s ($\mathbf{X}^c$ is the matrix of centers) in the algorithms proposed in Section 3.1.1 or Section 3.1.2 with the matrix

$$\mathbf{X}^s = \begin{bmatrix} V_{s1} \\ V_{s2} \\ \vdots \\ V_{sn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{np} \end{bmatrix}, \quad (3.4)$$

and obtain all the p principal components for the $V_{su}, u = 1, \dots, n$. Denote all the p principal components as $\{\mathbf{Y}_1^s, \dots, \mathbf{Y}_p^s\}$.

Step 3:

Based on the contribution proportion of each principal component from the PCA results in Step 2 and a given cumulative contribution rate λ , determine k^s which is the number of principal components to be used by finding the smallest k^s such that the cumulative contribution rate of the first k^s principal components reaches λ . Then, the selected principal components for $\mathbf{X}^s$ are $\{\mathbf{Y}_1^s, \dots, \mathbf{Y}_{k^s}^s\}$.

Step 4:

Run a symbolic PCA method described in Section 3.1.1 or Section 3.1.2 on the $V_{lu}, u = 1, \dots, n$, i.e., replace all the $\mathbf{X}^c$'s in the algorithms proposed in Section 3.1.1 or Section

3.1.2 with the matrix

$$\mathbf{X}^1 = \begin{bmatrix} V_{l1} \\ V_{l2} \\ \vdots \\ V_{ln} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{np} \end{bmatrix}, \quad (3.5)$$

and obtain all the p principal components for the $V_{lu}, u = 1, \dots, n$. Denote all the p principal components as $\{\mathbf{Y}_1^1, \dots, \mathbf{Y}_p^1\}$.

Step 5:

Obtain k^l and the selected principal components for $\mathbf{X}^1$, i.e., $\{\mathbf{Y}_1^1, \dots, \mathbf{Y}_{k^l}^1\}$, in the same way as in Step 3.

Step 6:

Stage r : $\Omega = P_r = (C_1, \dots, C_r)$. Let $\mathbf{Y}_i^*, i = 1, \dots, k^s + k^l$, represent the principal components $\mathbf{Y}_1^s, \dots, \mathbf{Y}_{k^s}^s, \mathbf{Y}_1^l, \dots, \mathbf{Y}_{k^l}^l$, which are selected in Step 3 and Step 5. For each principal component $\mathbf{Y}_i^*, i = 1, \dots, k^s + k^l$, and any C_v in $P_r, v = 1, \dots, r$, order the objects in C_v by increasing (or decreasing) principal component values of $\mathbf{Y}_i^*$. Let n_v be the number of objects in C_v and let $\{Y_{(1)i}, \dots, Y_{(n_v)i}\}$ be the objects in C_v after the ordering by the principal component values of $\mathbf{Y}_i^*$. Then, the possible bipartitions of C_v based on this ordering are $(\{Y_{(1)i}, \dots, Y_{(m)i}\}, \{Y_{(m+1)i}, \dots, Y_{(n_v)i}\}), m = 1, \dots, n_v - 1$.

Step 7:

For each possible bipartition of each cluster C_v in $P_r, v = 1, \dots, r$, ordered by the principal component values of each $\mathbf{Y}_i^*, i = 1, \dots, k^s + k^l$, calculate the change in the total within-cluster variation Δ_v by Eq.(2.17).

Step 8:

Select the cluster C_v to be partitioned and the position m where C_v is to be partitioned by the maximum Δ_v calculated in Step 7.

Step 9:

Now, $r = r + 1$. Repeat Steps 6 – 8 until $r = n$, or stop this process by some stopping rules.

Suppose we keep k^s principal components in Step 3 and k^l principal components in Step 5 based on the pre-specified cumulative contribution rate λ . Then, we only need to consider $\sum_{v=1}^r (k^s + k^l)(n_v - 1) = (k^s + k^l)(n - r)$, ($1 \leq k^s, k^l \leq p$), possible bipartitions at Stage r . If we set $\lambda = 1$, the number of possible bipartitions at Stage r is $\sum_{v=1}^r 2p(n_v - 1) = 2p(n - r)$, which means V_s 's and V_l 's are projected onto all the PC axes. If we set $\lambda = 0$, the number of possible bipartitions at Stage r is $\sum_{v=1}^r 2(n_v - 1) = 2(n - r)$, which means V_s 's and V_l 's are only projected onto the first PC axis. Furthermore, $k^s = k^l$ if the symbolic covariance PCA method is applied in Step 2 and Step 4.

3.2 SIMULATION STUDIES

In Section 3.1, we proposed three basic monothetic divisive clustering algorithms for interval-valued data, which are the divisive algorithm based on the symbolic covariance PCA proposed by Le-Rademacher (2008) and Le-Rademacher and Billard (2012), the divisive algorithm based on our proposed weighted symbolic PCA method and the divisive algorithm based on the endpoints of intervals. These three algorithms are named as DIVCPCA, DIVWPCA and DIVE, respectively. Furthermore, we proposed two mixed-strategy algorithms based on both the endpoints of intervals and the symbolic PCA. Depending on which of the symbolic covariance PCA method and the weighted symbolic PCA method is used, the two mixed-strategy algorithms are named as DIVECPCA and DIVEWPCA. In this section, the five proposed algorithms are evaluated and compared with the monothetic divisive algorithm proposed by Chavent (1998, 2000) on some simulated data. We use DIVM (the divisive algorithm based on the midpoints of intervals) to denote Chavent's algorithm. We divide these six algorithms into different types which are summarized in Table 3.1. For PCA-based algorithms, we can control the number of principal components axes to be used in these

Table 3.1: Types of the monothetic algorithms.

	Non-PCA-based Algorithms	PCA-based Algorithms
Midpoints-based Algorithms	DIVM	DIVWPCA, DIVCPCA
Endpoints-based Algorithms	DIVE	DIVEWPCA, DIVECPCA

algorithms by the cumulative contribution rates of the principal components. In the simulations, we consider two situations for each of these algorithms, only using the first principal component axis and using all the three principal components axes of the three-dimensional simulated datasets. We use DIVCPCA (1 PC) and DIVCPCA (3 PCs) to denote these two situations for DIVCPCA, and the same for DIVWPCA, DIVEWPCA and DIVECPCA. For DIVECPCA and DIVEWPCA, using one principal component axis means we use two principal components of each interval-valued observation. One principal component is the projection of the vertex V_l of the observation onto the principal component axis and the other one is the projection of the vertex V_s .

The six scenarios described in Figure 3.3 are for different clusters of intervals with dissimilar lengths and we can divide them into three cases, the well-separated case (scenario (a) and scenario (f)), the covering case (scenario (b) and scenario (e)) and the overlapping case (scenario (c) and scenario (d)). We simulate these three cases in Simulation 1, Simulation 2 and Simulation 3, respectively, to compare the accuracies of the algorithms in each case. Specifically, we simulate two different clusters of multi-dimensional interval-valued observations and in each dimension the intervals of different clusters of observations have dissimilar lengths and are well-separated in Simulation 1, overlapping in Simulation 2 and covering in Simulation 3. We can also simulate the three cases for different clusters of intervals with similar lengths. However, if two clusters of intervals with similar lengths are covered by each

Table 3.2: Different simulation cases.

	Dissimilar Lengths	Similar Lengths
Well-separated	Simulation 1	Simulation 4
Overlapping	Simulation 2	Simulation 5
Covering	Simulation 3	NA

other, it is highly possible that these two clusters of intervals are from the same cluster. If the lengths of these intervals are the same and they are covered by each other, the two clusters are not distinguishable unless we have more information. Thus, we only consider the well-separated case and the overlapping case for different clusters of intervals with similar lengths in Simulation 4 and Simulation 5, respectively. The first five simulations cases are summarized in Table 3.2.

In Simulation 6, we consider a more complex case. We simulate two different clusters of multi-dimensional interval-valued observations where the intervals from different clusters are overlapped in some dimensions and are covered by each other in some other dimensions. Simulation 7 is a mixed case of the first five simulations. We consider four different clusters of interval-valued observations. In each dimension, the relationships between the intervals from different clusters include all the three cases, i.e., the well-separated case, the overlapping case and the covering case.

In these seven simulations, we compare the accuracies of the six algorithms summarized in Table 3.1. In these simulations, the true numbers of clusters in the simulated datasets are pre-specified (two for the first six simulations and four for Simulation 7) and two distance or dissimilarity measures for interval-valued data are used, specifically, the span normalized Euclidean Hausdorff distance and the Ichino-Yaguchi distance. The definitions of those two distance or dissimilarity measures are given in Section 2.2.1. Furthermore, we conduct a simulation to compare the computation times of the algorithms on datasets with different

numbers of variables and different sample sizes in Simulation 8 to find the relationship between the numbers of possible bipartitions caused by different ordering criteria of the algorithms and the computation times of the algorithms.

Before the simulations, we need to explain how to simulate a p -variate interval-valued dataset which consists of several different clusters. First, we randomly draw 100 classical samples $\{X_i = (x_{i1}, \dots, x_{ip}), i = 1, \dots, 100\}$ from a p -variate normal distribution $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. We aggregate these 100 samples into one p -dimensional interval-valued observation $Y = ([a_1, b_1], \dots, [a_p, b_p])$ where $a_j = \min_i(x_{ij})$ and $b_j = \max_i(x_{ij})$, $i = 1, \dots, 100$, $j = 1, \dots, p$. Then, we draw another 100 classical samples from $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and aggregate them into the second observation in the same way. We repeat this process until we have enough interval-valued observations aggregated from the samples drawn from the distribution $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. These observations compose the first cluster. The next cluster can be generated in the same way but with the observations aggregated from the samples drawn from another p -variate normal distribution $N_p(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$, and so on.

3.2.1 SIMULATION 1: TWO WELL-SEPARATED CLUSTERS OF INTERVALS WITH DISSIMILAR LENGTHS

In Simulation 1, we consider a dataset with two well-separated clusters and each cluster consists of 10 three-dimensional interval-valued objects. In each dimension, the lengths of intervals from different clusters are dissimilar. Then, 1000 classical samples are randomly drawn from each of the three-variate normal distributions

$$N_3 \left(\begin{pmatrix} -5 \\ -5 \\ -5 \end{pmatrix}, \begin{pmatrix} 1 & 0.1 & 0.1 \\ 0.1 & 1 & 0.1 \\ 0.1 & 0.1 & 1 \end{pmatrix} \right)$$

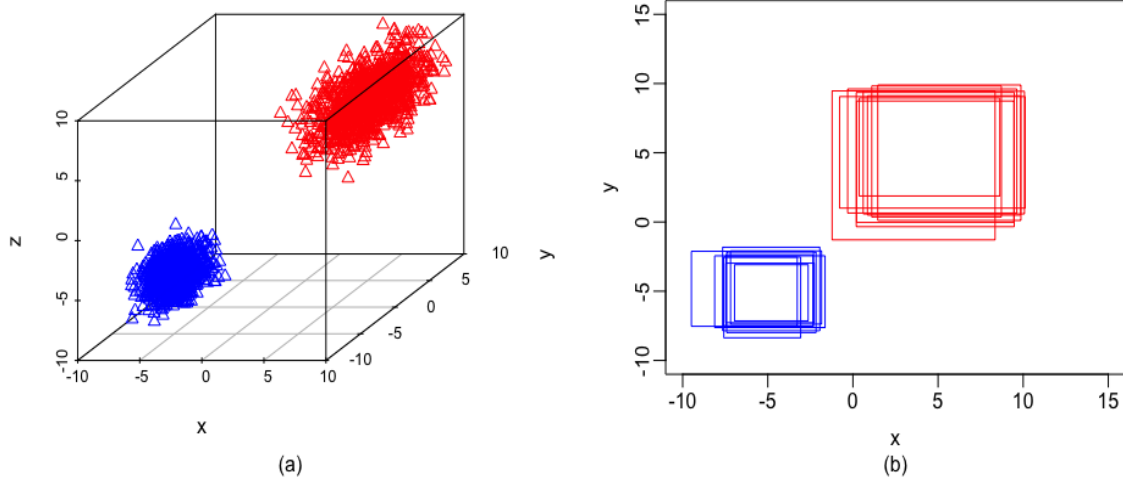


Figure 3.5: (a) 3D scatter plot of the classical samples from Simulation 1; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.

and

$$N_3 \left(\begin{pmatrix} 5 \\ 5 \\ 5 \end{pmatrix}, \begin{pmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix} \right),$$

respectively. There are 2000 samples in total and the 3D scatter plot of these samples is shown in Figure 3.5(a), which shows two well-separated clusters of samples. Based on the way in which we obtain the interval-valued objects, the two well-separated clusters of samples lead to two well-separated clusters of interval-valued objects.

We aggregate every 100 samples into one single interval-valued object and thus we obtain a dataset which consists of two clusters of interval-valued objects

$$\{Y_{(1)}, \dots, Y_{(10)}\} \sim N_3 \left(\begin{pmatrix} -5 \\ -5 \\ -5 \end{pmatrix}, \begin{pmatrix} 1 & 0.1 & 0.1 \\ 0.1 & 1 & 0.1 \\ 0.1 & 0.1 & 1 \end{pmatrix} \right) \quad (3.6)$$

and

$$\{Y_{(11)}, \dots, Y_{(20)}\} \sim N_3 \left(\begin{pmatrix} 5 \\ 5 \\ 5 \end{pmatrix}, \begin{pmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix} \right). \quad (3.7)$$

Figure 3.5(b) is the projections of the interval-valued objects onto the xy-plane. On both the x-axis and the y-axis, the two clusters are well-separated and the lengths of intervals from different clusters are quite different. After obtaining the dataset, we apply the six algorithms summarized in Table 3.1 on this dataset. For each algorithm, we also consider using the two distance measures for interval-valued data mentioned above. This process is repeated for 500 times and we count the number of times that the observations are correctly clustered into the two clusters described in Eq.(3.6) and Eq.(3.7) by each algorithm.

Table 3.3 records the clustering accuracies of these algorithms for the 500 repetitions. All the algorithms can easily classify the observations from well-separated clusters. Furthermore, there is no difference between these two distance measures in Simulation 1.

3.2.2 SIMULATION 2: TWO OVERLAPPING CLUSTERS OF INTERVALS WITH DISSIMILAR LENGTHS

In Simulation 2, we consider a dataset with two overlapping clusters and each cluster consists of 10 three-dimensional interval-valued objects. In each dimension, the lengths of intervals from different clusters are dissimilar. To simulate this dataset, 1000 classical samples are randomly drawn from the normal distributions given in Eq.(3.8) and Eq.(3.9), respectively.

Table 3.3: Clustering accuracies of the monothetic algorithms (Simulation 1).

Algorithm	Span normalized Euclidean Hausdorff distance	Ichino-Yaguchi distance
DIVM	100.0%	100.0%
DIVWPCA (1 PC)	100.0%	100.0%
DIVWPCA (3 PCs)	100.0%	100.0%
DIVCPCA (1 PC)	100.0%	100.0%
DIVCPCA (3 PCs)	100.0%	100.0%
DIVE	100.0%	100.0%
DIVEWPCA (1 PC)	100.0%	100.0%
DIVEWPCA (3 PCs)	100.0%	100.0%
DIVECPCA (1 PC)	100.0%	100.0%
DIVECPCA (3 PCs)	100.0%	100.0%

Every 100 samples are aggregated into one single interval-valued object and thus we obtain a dataset which consists of two clusters of interval-valued objects

$$\{Y_{(1)}, \dots, Y_{(10)}\} \sim N_3 \left(\begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 & 0.1 & 0.1 \\ 0.1 & 1 & 0.1 \\ 0.1 & 0.1 & 1 \end{pmatrix} \right) \quad (3.8)$$

and

$$\{Y_{(11)}, \dots, Y_{(20)}\} \sim N_3 \left(\begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix} \right). \quad (3.9)$$

The 3D scatter plot of these classical samples is shown in Figure 3.6(a) which shows two overlapping clusters of samples. Figure 3.6(b) shows the projections of the interval-valued objects onto the xy-plane. On both the x-axis and the y-axis, the two clusters are overlapped and the lengths of intervals from different clusters are quite different.

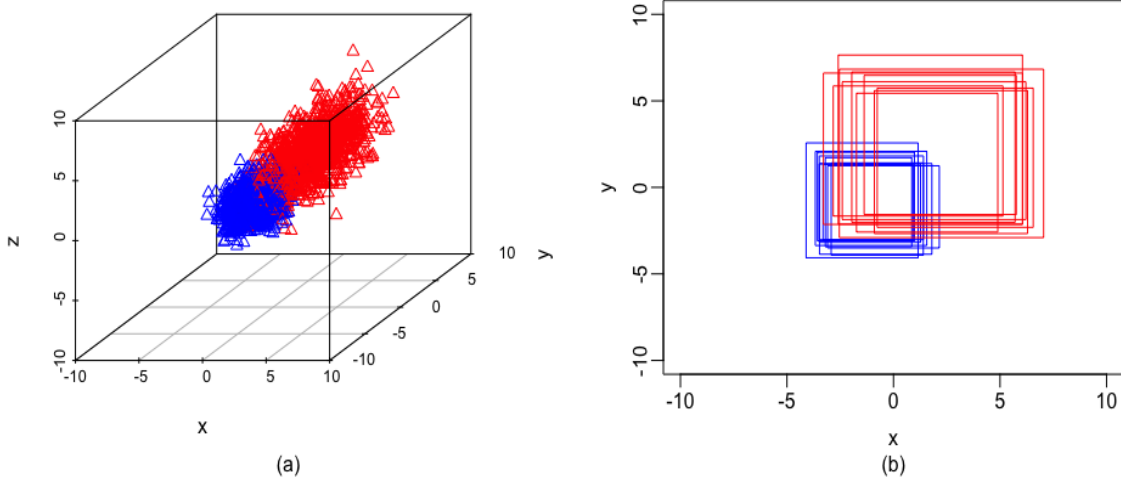


Figure 3.6: (a) 3D scatter plot of the classical samples from Simulation 2; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.

We simulate this dataset for 500 times and the clustering accuracies of all the monothetic algorithms for the 500 repetitions in Simulation 2 are the same as those in Simulation 1. Although the two clusters are partially overlapped, all the algorithms can classify the observations from the two overlapping clusters and give 100% clustering accuracies in this simulation. One possible reason is the midpoints, endpoints and lengths of the intervals from the two clusters are quite different in the simulated data.

3.2.3 SIMULATION 3: TWO COVERING CLUSTERS OF INTERVALS WITH DISSIMILAR LENGTHS

In Simulation 3, we consider a dataset with two clusters and one of the clusters is covered by the other one. Each cluster consists of 10 three-dimensional interval-valued objects. In each dimension, the lengths of intervals from different clusters are dissimilar. To simulate

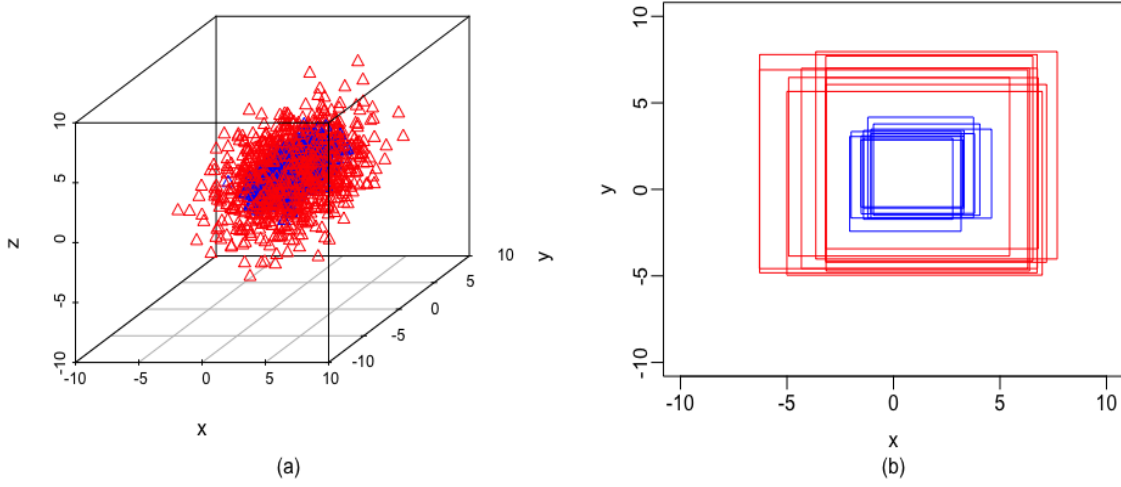


Figure 3.7: (a) 3D scatter plot of the classical samples from Simulation 3; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.

this dataset, 1000 classical samples are randomly drawn from the normal distributions given in Eq.(3.10) and Eq.(3.11), respectively, and we aggregate them into two clusters of interval-valued objects

$$\{Y_{(1)}, \dots, Y_{(10)}\} \sim N_3 \left(\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 & 0.6 & 0.3 \\ 0.6 & 1 & 0.3 \\ 0.3 & 0.3 & 1 \end{pmatrix} \right) \quad (3.10)$$

and

$$\{Y_{(11)}, \dots, Y_{(20)}\} \sim N_3 \left(\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 5 & 0.8 & 0.8 \\ 0.8 & 5 & 0.8 \\ 0.8 & 0.8 & 5 \end{pmatrix} \right). \quad (3.11)$$

The 3D scatter plot of these classical samples is shown in Figure 3.7(a). Figure 3.7(b) shows the projections of the interval-valued objects onto the xy-plane. On both the x-axis and

Table 3.4: Clustering accuracies of the monothetic algorithms (Simulation 3).

Algorithm	Span normalized Euclidean Hausdorff distance	Ichino-Yaguchi distance
DIVM	0.0%	0.2%
DIVWPCA (1 PC)	0.0%	0.0%
DIVWPCA (3 PCs)	0.0%	0.0%
DIVCPCA (1 PC)	0.0%	0.0%
DIVCPCA (3 PCs)	0.0%	0.0%
DIVE	100.0%	100.0%
DIVEWPCA (1 PC)	100.0%	100.0%
DIVEWPCA (3 PCs)	100.0%	100.0%
DIVECPCA (1 PC)	100.0%	100.0%
DIVECPCA (3 PCs)	100.0%	100.0%

the y-axis, the lengths of intervals from different clusters are quite different and the intervals from the blue cluster are covered by the intervals from the red cluster, which generalizes the case shown in Figure 2.2.

Table 3.4 records the clustering accuracies of all the monothetic divisive algorithms for the 500 repetitions. The pattern of the clustering accuracies is quite obvious in this simulation. No matter which of the two distance measures is used, endpoints-based algorithms outperform the midpoints-based algorithms. All the endpoints-based algorithms give 100% clustering accuracies while the clustering accuracies of the midpoints-based algorithms are all close to 0%. The huge difference between the performance of these two kinds of algorithms on this simulated data proves our discussion in Section 3.1.4 that endpoints work better than midpoints in differentiating different clusters of intervals with dissimilar lengths. The clustering accuracies of the midpoints-based algorithms will decrease if the average midpoint values of the intervals from different clusters are closer. In this simulated data, the average midpoint values of the intervals from the two clusters in each dimension are all set to 1 (both

of the means of the two three-variate normal distributions in Eq.(3.10) and Eq.(3.11) are $(1, 1, 1)$ and thus the midpoints-based algorithms can barely make one correct clustering among the 500 repetitions. Besides, increasing the number of principal components used in DIVWPCA and DIVCPCA cannot increase the clustering accuracies in this case.

3.2.4 SIMULATION 4: TWO WELL-SEPARATED CLUSTERS OF INTERVALS WITH SIMILAR LENGTHS

After the simulations of the well-separated, overlapping and covering cases for intervals with dissimilar lengths and from different clusters, we can also simulate the three cases for intervals with similar lengths but from different clusters. We consider a dataset with two well-separated clusters and each cluster consists of 10 three-dimensional interval-valued objects. In each dimension, the lengths of intervals from different clusters are similar. To simulate this dataset, we randomly draw 1000 classical samples from the normal distributions given in Eq.(3.12) and Eq.(3.13), respectively. Every 100 samples are aggregated into one single interval-valued object and thus we obtain a dataset which consists of two clusters of interval-valued objects

$$\{Y_{(1)}, \dots, Y_{(10)}\} \sim N_3 \left(\begin{pmatrix} -5 \\ -5 \\ -5 \end{pmatrix}, \begin{pmatrix} 2 & 0.1 & 0.1 \\ 0.1 & 2 & 0.1 \\ 0.1 & 0.1 & 2 \end{pmatrix} \right) \quad (3.12)$$

and

$$\{Y_{(11)}, \dots, Y_{(20)}\} \sim N_3 \left(\begin{pmatrix} 5 \\ 5 \\ 5 \end{pmatrix}, \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \right). \quad (3.13)$$

The 3D scatter plot of these classical samples is shown in Figure 3.8(a), which shows two well-separated clusters of samples. Figure 3.8(b) shows the projections of the interval-valued objects onto the xy-plane. On both the x-axis and the y-axis, the two clusters are well-separated and the lengths of intervals from different clusters are similar.

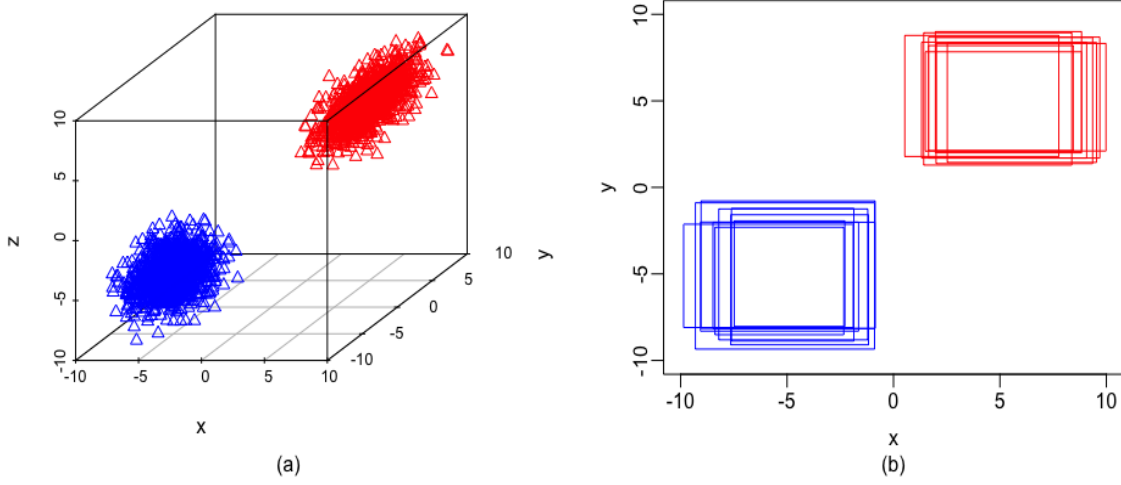


Figure 3.8: (a) 3D scatter plot of the classical samples from Simulation 4; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.

We simulate this dataset for 500 times and the clustering accuracies of all the monothetic algorithms for the 500 repetitions in this simulation are the same as those in Simulation 1. All the algorithms give 100% clustering accuracies in this simulation. Thus, the conclusion is no matter whether the lengths of the intervals from different clusters are similar or not, all the algorithms can correctly classify the observations from well-separated clusters.

3.2.5 SIMULATION 5: TWO OVERLAPPING CLUSTERS OF INTERVALS WITH SIMILAR LENGTHS

In Simulation 5, we consider a dataset with two overlapping clusters and each cluster consists of 10 three-dimensional interval-valued objects. In each dimension, the lengths of the intervals from the two different clusters are similar. To simulate this dataset, we draw 1000 classical samples from the normal distributions given in Eq.(3.14) and Eq.(3.15), respectively. Every

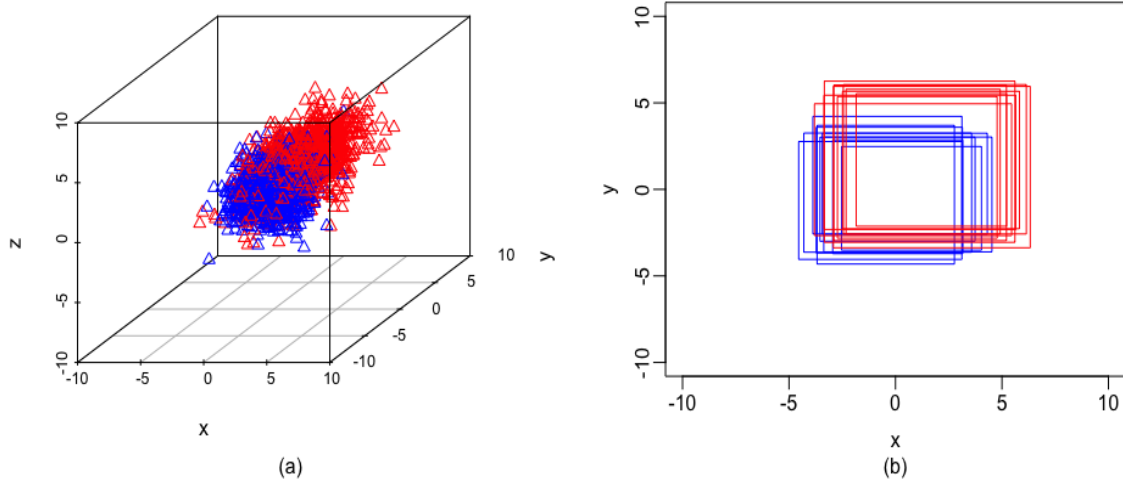


Figure 3.9: (a) 3D scatter plot of the classical samples from Simulation 5; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.

100 samples are aggregated into one single interval-valued object and thus we obtain a dataset which consists of two clusters of interval-valued objects

$$\{Y_{(1)}, \dots, Y_{(10)}\} \sim N_3 \left(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 & 0.1 & 0.1 \\ 0.1 & 2 & 0.1 \\ 0.1 & 0.1 & 2 \end{pmatrix} \right) \quad (3.14)$$

and

$$\{Y_{(11)}, \dots, Y_{(20)}\} \sim N_3 \left(\begin{pmatrix} 1.5 \\ 1.5 \\ 1.5 \end{pmatrix}, \begin{pmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix} \right). \quad (3.15)$$

The 3D scatter plot of these classical samples is shown in Figure 3.9(a), which shows two overlapping clusters of samples. Figure 3.9(b) shows the projections of the interval-valued

Table 3.5: Clustering accuracies of the monothetic algorithms (Simulation 5).

Algorithm	Span normalized Euclidean Hausdorff distance	Ichino-Yaguchi distance
DIVM	87.8%	89.4%
DIVWPCA (1 PC)	94.8%	96.4%
DIVWPCA (3 PCs)	92.2%	94.4%
DIVCPCA (1 PC)	95.0%	94.6%
DIVCPCA (3 PCs)	94.6%	96.8%
DIVE	66.4%	69.4%
DIVEWPCA (1 PC)	94.2%	96.2%
DIVEWPCA (3 PCs)	96.0%	97.4%
DIVECPCA (1 PC)	95.8%	96.4%
DIVECPCA (3 PCs)	95.0%	96.2%

objects onto the xy-plane. On both the x-axis and the y-axis, the two clusters of intervals are overlapped and the lengths of intervals from different clusters are similar.

Table 3.5 records the clustering accuracies of all the monothetic divisive algorithms for 500 repetitions. It is quite clear that all the PCA-based algorithms give higher clustering accuracies than the two non-PCA-based algorithms, i.e., DIVM and DIVE. The accuracies of the PCA-based algorithms are all around 95% while the accuracies of DIVM and DIVE are around 88% and 68%, respectively. The reason is when we project the interval-valued objects onto the x-axis or the y-axis, the two clusters of projected intervals are not easy to classify since they have similar lengths and each interval is almost covered by the intervals from the other cluster, which is evident from Figure 3.9(b). However, the PCA-based algorithms project the objects onto some principal components axes to maximize the variations between them, which makes it easier to classify the two clusters. Besides, the results in Table 3.5 show that the numbers of principal components used in the PCA-based algorithms may not interfere with the clustering accuracies in this simulation.

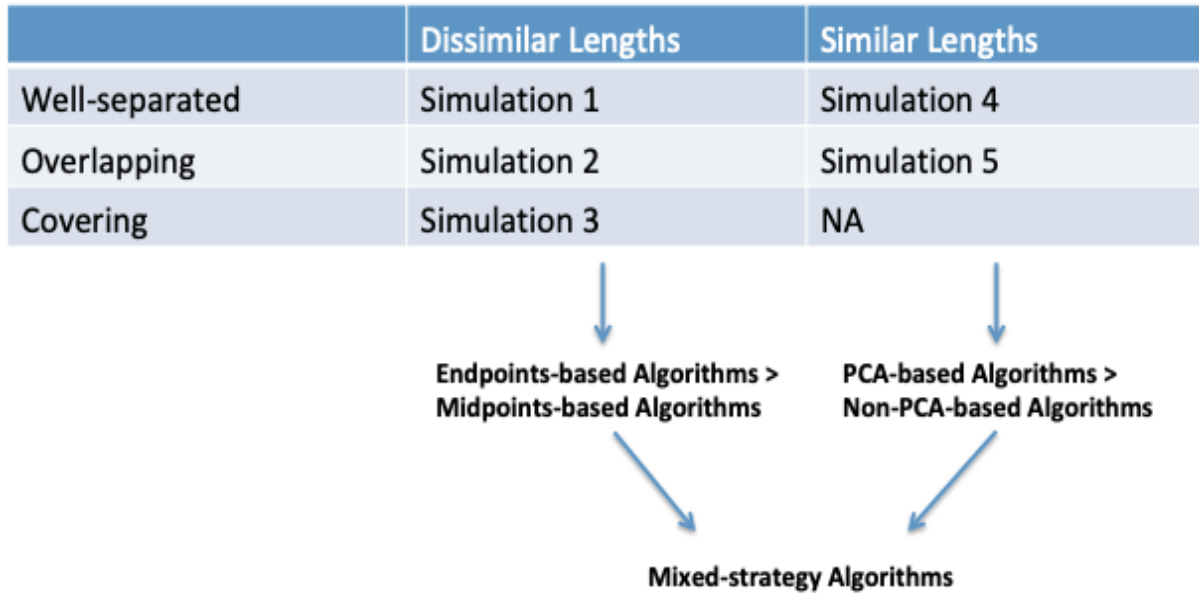


Figure 3.10: Summary of the first five simulations.

Thus, we finish all the simulations described in Table 3.2 and a summary of the simulation results is shown in Figure 3.10. Among all the monothetic divisive algorithms, the endpoints-based algorithms work better than do the midpoints-based algorithms for intervals with dissimilar lengths and from different clusters. The PCA-based algorithms work better than do the non-PCA-based algorithms for intervals with similar lengths but from different clusters. More importantly, our proposed mixed-strategy algorithms DIVECPCA and DIVEWPCA are not only endpoints-based algorithms but also PCA-based algorithms and thus they can handle all the situations described in Table 3.2. The results of the first five simulations demonstrate that the two mixed-strategy algorithms outperform the other monothetic divisive algorithms in these simulations.

3.2.6 SIMULATION 6: TWO CLUSTERS OF INTERVAL-VALUED OBJECTS WITH IRREGULAR RELATIONSHIPS

In Simulation 6, we consider two clusters of multi-dimensional interval-valued objects with irregular relationships. We simulate two clusters of interval-valued observations where intervals of different clusters are overlapped in some dimensions and are covered in some other dimensions. To simulate this dataset, 1000 classical samples are drawn from the normal distributions given in Eq.(3.16) and Eq.(3.17), respectively. Every 100 samples are aggregated into one single interval-valued object and thus we obtain a dataset which consists of two clusters of interval-valued objects

$$\{Y_{(1)}, \dots, Y_{(10)}\} \sim N_3 \left(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 & 0.1 & 0.1 \\ 0.1 & 2 & 0.1 \\ 0.1 & 0.1 & 1 \end{pmatrix} \right) \quad (3.16)$$

and

$$\{Y_{(11)}, \dots, Y_{(20)}\} \sim N_3 \left(\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 & 0.2 & 0.2 \\ 0.2 & 0.1 & 0.2 \\ 0.2 & 0.2 & 1 \end{pmatrix} \right). \quad (3.17)$$

The 3D scatter plot of these classical samples is shown in Figure 3.11(a). Figure 3.11(b), Figure 3.12(a) and Figure 3.12(b) are the projections of the interval-valued objects onto the xy-plane, the xz-plane and the yz-plane, respectively, and it is clear that the two clusters of projected intervals are partially overlapped by each other on both the x-axis and the z-axis while the red cluster is covered by the blue cluster on the y-axis.

The clustering accuracies of all the monothetic divisive algorithms for 500 repetitions are recorded in Table 3.6. From the table, the accuracies of DIVECPCA (1PC) are much lower than the accuracies of the other mixed-strategy algorithms. Recall DIVECPCA (1PC) means only the first principal components of the V_s 's and the V_l 's are used to give the initial ordering criterion to the objects in the mixed-strategy algorithm based on the symbolic covariance

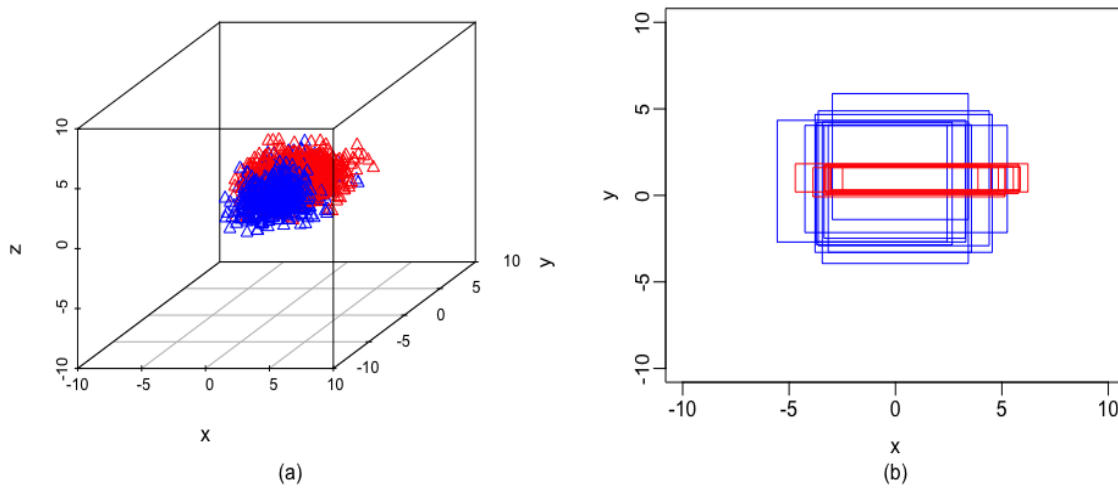


Figure 3.11: (a) 3D scatter plot of the classical samples from Simulation 6; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.

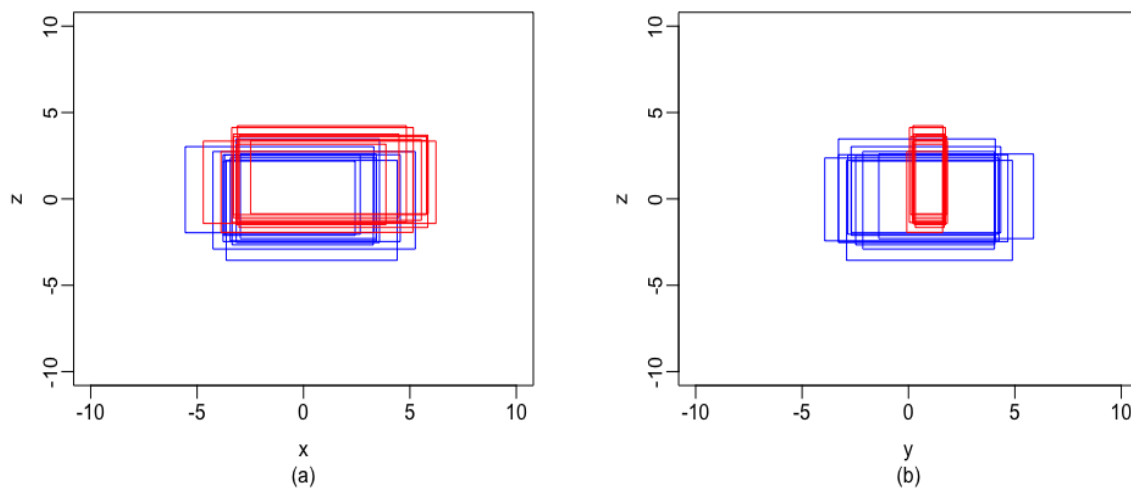


Figure 3.12: (a) Projections of the interval-valued objects aggregated by the classical samples onto the xz-plane; (b) projections onto the yz-plane.

Table 3.6: Clustering accuracies of the monothetic algorithms (Simulation 6).

Algorithm	Span normalized Euclidean Hausdorff distance	Ichino-Yaguchi distance
DIVM	60.8%	68.0%
DIVWPCA (1 PC)	64.0%	62.8%
DIVWPCA (3 PCs)	61.6%	61.8%
DIVCPCA (1 PC)	45.2%	43.8%
DIVCPCA (3 PCs)	48.8%	44.8%
DIVE	41.2%	36.4%
DIVEWPCA (1 PC)	100.0%	100.0%
DIVEWPCA (3 PCs)	100.0%	100.0%
DIVECPCA (1 PC)	55.8%	56.0%
DIVECPCA (3 PCs)	99.8%	100.0%

PCA method. However, if we increase the number of principal components of each V_s and V_l used in the algorithm from one to three by increasing the cumulative contribution rate λ , the accuracies of DIVECPCA increase significantly. In the previous simulations, only considering the first principal components is enough for the mixed-strategy algorithms to reach high accuracies and there is no difference between the accuracies of the 1PC version and the 3PCs version of each PCA-based algorithm. For some more complex situations such as this simulation, only considering the first principal components may not be enough since the correct bipartition may not be included in the possible bipartitions which result from the first principal components values. Therefore, DIVECPCA needs more principal components to achieve a higher clustering accuracy in this simulation. However, DIVEWPCA can achieve 100% accuracies even if only the first principal components are considered, which means the performance of the mixed-strategy algorithms also depends on which symbolic PCA method is applied in the algorithms. Also, the results in Table 3.6 show that all the algorithms except the mixed-strategy algorithms perform poorly in this simulation. Furthermore, we simulated

two clusters of interval-valued observations which are overlapped in some dimensions and are covered in some other dimensions in this simulation. However, if we make the two clusters well-separated in some dimensions, the accuracies of all the algorithms will improve.

3.2.7 SIMULATION 7: A MIXED CASE

Simulation 7 is a mixed case of the first five simulations. We consider four different clusters of interval-valued observations. In each dimension, the relationships between the intervals from different clusters include all the three cases, i.e., the well-separated case, the overlapping case and the covering case. Each cluster consists of 5 three-dimensional interval-valued objects. To simulate this dataset, 500 classical samples are randomly drawn from the normal distributions given in Eq.(3.18), Eq.(3.19), Eq.(3.20) and Eq.(3.21), respectively. Every 100 samples are aggregated into one single interval-valued object and thus we obtain a dataset which consists of four clusters of interval-valued objects

$$\{Y_{(1)}, \dots, Y_{(5)}\} \sim N_3 \left(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 & 0.1 & 0.1 \\ 0.1 & 2 & 0.1 \\ 0.1 & 0.1 & 2 \end{pmatrix} \right), \quad (3.18)$$

$$\{Y_{(6)}, \dots, Y_{(10)}\} \sim N_3 \left(\begin{pmatrix} 1.5 \\ 1.5 \\ 1.5 \end{pmatrix}, \begin{pmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix} \right), \quad (3.19)$$

$$\{Y_{(11)}, \dots, Y_{(15)}\} \sim N_3 \left(\begin{pmatrix} -7 \\ -7 \\ -7 \end{pmatrix}, \begin{pmatrix} 0.5 & 0.1 & 0.1 \\ 0.1 & 0.5 & 0.1 \\ 0.1 & 0.1 & 0.5 \end{pmatrix} \right) \quad (3.20)$$

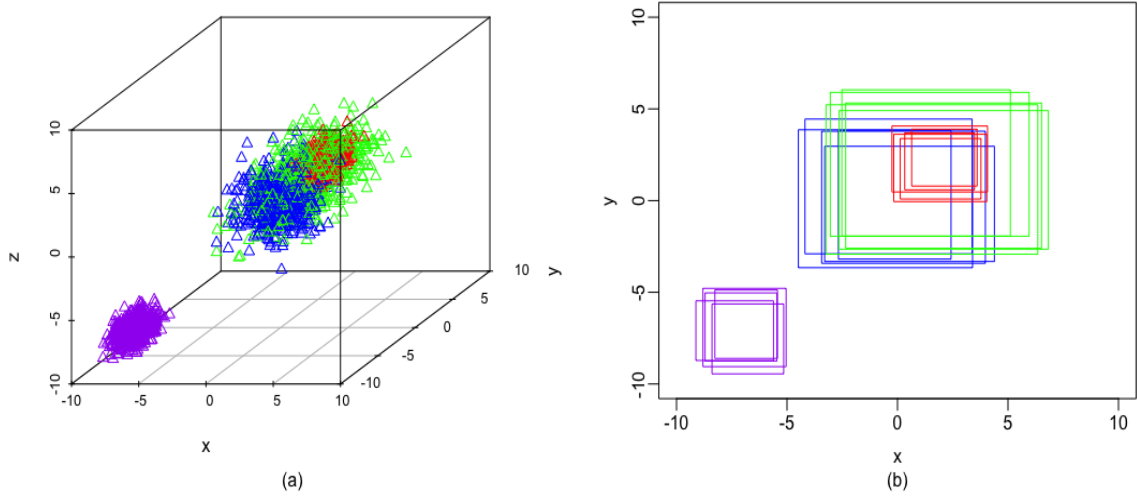


Figure 3.13: (a) 3D scatter plot of the classical samples from Simulation 7; (b) projections of the interval-valued objects aggregated by the classical samples onto the xy-plane.

and

$$\{Y_{(16)}, \dots, Y_{(20)}\} \sim N_3 \left(\begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 0.5 & 0.1 & 0.1 \\ 0.1 & 0.5 & 0.1 \\ 0.1 & 0.1 & 0.5 \end{pmatrix} \right). \quad (3.21)$$

The 3D scatter plot of these classical samples is shown in Figure 3.13(a), which shows four clusters of samples. Figure 3.13(b) shows the projections of the interval-valued objects onto the xy-plane. On both the x-axis and the y-axis, the blue cluster and the green cluster of intervals are overlapped while the red cluster is covered by both of them. The purple cluster is well-separated from the other clusters. The lengths of the intervals from the purple cluster and the red cluster are similar and the lengths of the intervals from the blue cluster and the green cluster are also similar.

Table 3.7: Clustering accuracies of the monothetic algorithms (Simulation 7).

Algorithm	Span normalized Euclidean Hausdorff distance	Ichino-Yaguchi distance
DIVM	41.0%	39.2%
DIVWPCA (1 PC)	90.2%	90.2%
DIVWPCA (3 PCs)	88.6%	88.8%
DIVCPCA (1 PC)	44.8%	44.8%
DIVCPCA (3 PCs)	49.8%	49.2%
DIVE	85.8%	88.6%
DIVEWPCA (1 PC)	97.0%	97.4%
DIVEWPCA (3 PCs)	97.6%	97.6%
DIVECPCA (1 PC)	96.6%	97.8%
DIVECPCA (3 PCs)	96.0%	97.0%

Table 3.7 records the clustering accuracies of all the monothetic divisive algorithms for 500 repetitions. In this simulation, all the five proposed algorithms (DIVWPCA, DIVCPCA, DIVE, DIVEWPCA and DIVECPCA) give higher accuracies than does DIVM. DIVWPCA works much better than DIVCPCA in this simulation, which means not only the performance of the mixed-strategy algorithms but also the performance of the other PCA-based algorithms depend on which symbolic PCA method is applied, and our proposed weighted symbolic PCA method works better than the symbolic covariance PCA method when applied to these algorithms. Also, Table 3.7 shows that the accuracies of the mixed-strategy algorithms are higher than the accuracies of the other algorithms and DIVM and DIVCPCA perform poorly in this simulation.

The results of all the above simulations show that the two distance measures applied in these monothetic algorithms are similar in most simulation cases. Furthermore, only the mixed-strategy algorithms can obtain high accuracies in all the simulations.

Table 3.8: Numbers of possible bipartitions caused by the different ordering criteria of the monothetic algorithms.

Algorithm	The number of possible bipartitions at Stage r	The total number of possible bipartitions
DIVM	$p(n - r)$	$pn(n - 1)/2$
DIVWPCA (1 PC)	$n - r$	$n(n - 1)/2$
DIVWPCA (p PCs)	$p(n - r)$	$pn(n - 1)/2$
DIVCPCA (1 PC)	$n - r$	$n(n - 1)/2$
DIVCPCA (p PCs)	$p(n - r)$	$pn(n - 1)/2$
DIVE	$2p(n - r)$	$pn(n - 1)$
DIVEWPCA (1 PC)	$2(n - r)$	$n(n - 1)$
DIVEWPCA (p PCs)	$2p(n - r)$	$pn(n - 1)$
DIVECPCA (1 PC)	$2(n - r)$	$n(n - 1)$
DIVECPCA (p PCs)	$2p(n - r)$	$pn(n - 1)$

3.2.8 SIMULATION 8: THE COMPUTATION TIMES OF THE MONOTHETIC ALGORITHMS

Simulation 8 is conducted to enable us to find the relationship between the numbers of possible bipartitions caused by different ordering criteria of the monothetic algorithms and the computation times of these algorithms. The numbers of possible bipartitions caused by the different ordering criteria of the monothetic algorithms are shown in Table 3.8. Table 3.8 shows both the number of possible bipartitions at Stage r , $r = 1, \dots, n - 1$, and the total number of possible bipartitions across all the clustering stages of each algorithm.

In this simulation, “ p PCs” means we keep all the p principal components in the PCA-based algorithms by setting the cumulative contribution rate $\lambda = 1$. Here, “1PC” means we keep only the first principal components in the PCA-based algorithms by setting the cumulative contribution rate $\lambda = 0$.

The algorithms are run on simulated datasets with three different sample sizes $n = 10, 20$ and 40 and three different numbers of variables $p = 5, 10$ and 15 and thus we obtain nine kinds of datasets with different combinations of sample sizes and numbers of variables. Each

Table 3.9: Average computation times of the monothetic algorithms (in seconds).

Algorithm	n=10			n=20		
	p=5	p=10	p=15	p=5	p=10	p=15
DIVM	0.1069	0.1581	0.2150	0.8606	1.2926	1.8413
DIVWPCA (1 PC)	0.0825	0.0973	0.1103	0.6511	0.6879	0.7513
DIVWPCA (p PCs)	0.1178	0.1682	0.1850	0.9341	1.4028	1.9050
DIVCPCA (1 PC)	0.0775	0.1015	0.1271	0.6165	0.6669	0.7294
DIVCPCA (p PCs)	0.1138	0.1784	0.2080	0.8932	1.3299	1.8844
DIVE	0.2120	0.3474	0.4728	1.7399	2.9640	4.3652
DIVEWPCA (1 PC)	0.0873	0.1047	0.1216	0.7017	0.7672	0.8354
DIVEWPCA (p PCs)	0.1620	0.2473	0.2804	1.2363	2.1295	3.1309
DIVECPCA (1 PC)	0.0851	0.1044	0.1370	0.6573	0.7353	0.8075
DIVECPCA (p PCs)	0.1584	0.2651	0.2971	1.2316	2.1323	3.079
	n=40					
	p=5	p=10	p=15			
DIVM	7.4189	11.6568	17.3903			
DIVWPCA (1 PC)	5.6375	6.0285	6.7051			
DIVWPCA (p PCs)	7.4977	11.8055	17.6201			
DIVCPCA (1 PC)	5.4431	5.9076	6.4708			
DIVCPCA (p PCs)	7.4084	11.6457	17.4296			
DIVE	15.4323	28.5432	44.6726			
DIVEWPCA (1 PC)	5.8029	6.5601	7.3652			
DIVEWPCA (p PCs)	10.4409	18.6794	29.7368			
DIVECPCA (1 PC)	5.7767	6.3912	7.2217			
DIVECPCA (p PCs)	10.3070	18.5135	29.4670			

kind of datasets is simulated for 200 times and Table 3.9 records the average computation times of the monothetic algorithms in seconds on each kind of simulated datasets. The distance measure applied in these algorithms is the span normalized Euclidean Hausdorff distance.

Comparing Table 3.8 with Table 3.9, it is obvious that the algorithm with a larger number of possible bipartitions usually has a longer computation time. The number of possible bipartitions of DIVWPCA (1 PC), DIVCPCA (1 PC), DIVEWPCA (1 PC) and DIVECPCA (1

PC) are uncorrelated with p and thus they are faster than the other algorithms. However, only considering the first principal components in the mixed-strategy algorithms may decrease the clustering accuracies such as for DIVECPCA (1 PC) in Simulation 6. One possible solution to this problem is that we tune the cumulative contribution rate λ used in the DIVECPCA algorithm to a moderate value between 0 and 1 such that the algorithm is not only as accurate as DIVECPCA (p PCs) but also not much slower than DIVECPCA (1 PC). Furthermore, Table 3.9 shows that the average computation times of the monothetic algorithms increase as n and p increase and the average computation times increase more significantly as n increases because of the quadratic relationship between n and the total number of possible bipartitions considered in each of the algorithms.

3.3 APPLICATIONS

In the simulation studies, only the two mixed-strategy algorithms DIVEWPCA and DIVECPCA obtained high accuracies in all the simulation cases. Therefore, in this section, these two mixed-strategy algorithms are applied to real-world data to further demonstrate their effectiveness. Two interval-valued datasets are applied in this section, namely, the fats and oils dataset from Ichino and Yaguchi (1994) and the face recognition dataset from Leroy et al. (1996). Dendrograms are drawn to display the divisive clustering trees of the algorithms. Furthermore, one advantage of the monothetic algorithms over the polythetic algorithms is that we can set a binary question at each partition node of the dendrograms of these monothetic algorithms and thus these dendrograms can serve as decision trees. Thus, we also show in this section how to set the binary questions for the dendrograms of the two mixed-strategy algorithms.

3.3.1 FATS AND OILS DATASET

The first dataset comes from Ichino and Yaguchi (1994), and is given in Table 3.10. Eight different fats or oils are described by four quantitative features which take interval values.

Table 3.10: Table of fats and oils.

Sample	Specific Gravity	Freezing Point	Iodine Value	Saponification Value
linseed oil	[0.930, 0.935]	[-27, -18]	[170, 204]	[118, 196]
perilla oil	[0.930, 0.937]	[-5, -4]	[192, 208]	[188, 197]
cottonseed oil	[0.916, 0.918]	[-6, -1]	[99, 113]	[189, 198]
sesame oil	[0.920, 0.926]	[-6, -4]	[104, 116]	[187, 193]
camellia oil	[0.916, 0.917]	[-21, -15]	[80, 82]	[189, 193]
olive oil	[0.914, 0.919]	[0, 6]	[79, 90]	[187, 196]
beef tallow	[0.860, 0.870]	[30, 38]	[40, 48]	[190, 199]
hog fat	[0.858, 0.864]	[22, 32]	[53, 77]	[190, 202]

We take this dataset as an example to illustrate the clustering procedures of the two mixed-strategy algorithms DIVEWPCA and DIVECPA. Since DIVEWPCA and DIVECPA are combinations of the three basic algorithms DIVE, DIVWPCA and DIVCPA, this example can also be used as a reference when one of the three basic algorithms is applied.

CLUSTERING PROCEDURES OF DIVEWPCA

Let Y_u , $u = 1, \dots, 8$, represent the eight different oils and fats given in Table 3.10. To run DIVEWPCA on this dataset, the first step is to run the proposed weighted symbolic PCA method on this dataset. For each variable, the eight observations take interval values and we stack these eight intervals into a single histogram. Then, we apply Eq.(3.2) to obtain the weight w_{uj} for the j^{th} variable of Y_u and the result is given in Table 3.11. From Table 3.10 and Table 3.11, we can find that the four variables are measured on different scales. Therefore, we choose to do unit variance scaling on each column of Table 3.11 and the result is given in Table 3.12.

Table 3.11: w_{uj} values before the unit variance scaling.

Sample	Specific Gravity	Freezing Point	Iodine Value	Saponification Value
Y_1	0.010	12	46	124
Y_2	0.012	3	28	56
Y_3	0.005	8	23	55
Y_4	0.006	5	21	37
Y_5	0.003	9	4	30
Y_6	0.008	6	13	55
Y_7	0.014	10	8	51
Y_8	0.010	12	24	54

Table 3.12: w_{uj} values after the unit variance scaling.

Sample	Specific Gravity	Freezing Point	Iodine Value	Saponification Value
Y_1	1.0191	1.2929	1.7762	1.8241
Y_2	1.2229	0.3232	1.0811	0.8238
Y_3	0.5096	0.8619	0.8881	0.8091
Y_4	0.6115	0.5387	0.8109	0.5443
Y_5	0.3057	0.9697	0.1545	0.4413
Y_6	0.8153	0.6465	0.5020	0.8091
Y_7	1.4267	1.0774	0.3089	0.7502
Y_8	1.0191	1.2929	0.9267	0.7944

After the scaling, we apply Eq.(3.3) on the w_{uj} values in Table 3.12 to obtain the weights for the eight observations, namely,

$$w_1 = 0.2175, w_2 = 0.1270, w_3 = 0.1129, w_4 = 0.0922,$$

$$w_5 = 0.0688, w_6 = 0.1020, w_7 = 0.1311, w_8 = 0.1484.$$

Let $\mathbf{W}$ be the diagonal weighting matrix with $w_u, u = 1, \dots, 8$, as the diagonal elements.

After obtaining the weights for the eight observations, we run the weighted symbolic PCA method on the matrix of lower endpoints, namely,

$$\mathbf{X}^s = \begin{bmatrix} V_{s1} \\ V_{s2} \\ \vdots \\ V_{s8} \end{bmatrix} = \begin{bmatrix} 0.930 & -27 & 170 & 118 \\ 0.930 & -5 & 192 & 188 \\ 0.916 & -6 & 99 & 189 \\ 0.920 & -6 & 104 & 187 \\ 0.916 & -21 & 80 & 189 \\ 0.914 & 0 & 79 & 187 \\ 0.860 & 30 & 40 & 190 \\ 0.858 & 22 & 53 & 190 \end{bmatrix}. \quad (3.22)$$

We do zero mean scaling on each variable of $\mathbf{X}^s$ and we also do unit variance scaling on each variable since the four variables are measured on different scales. Then, we perform an eigenvalue decomposition on the weighted sample covariance matrix $\mathbf{X}^{sT}\mathbf{W}^T\mathbf{W}\mathbf{X}^s$ to obtain the principal components for $\mathbf{X}^s$. The principal component coefficients and the contribution rate for each principal component are given in Table 3.13.

Table 3.13: Principal component coefficients and contribution rate for each principal component of $\mathbf{X}^s$ based on the DIVEWPCA algorithm.

Variable	PC1	PC2	PC3	PC4
Specific Gravity	0.4203	-0.5560	0.2422	0.6749
Freezing Point	-0.4911	0.1936	-0.5371	0.6580
Iodine Value	0.4666	-0.2777	-0.8078	-0.2295
Saponification Value	-0.6038	-0.7591	-0.0189	-0.2426
Contribution Rate	0.8180	0.1412	0.0368	0.0040
Cumulative Contribution Rate	0.8180	0.9592	0.9960	1.0000

Then, we run the weighted symbolic PCA method on the matrix of upper endpoints, namely,

$$\mathbf{X}^1 = \begin{bmatrix} V_{l1} \\ V_{l2} \\ \vdots \\ V_{l8} \end{bmatrix} = \begin{bmatrix} 0.935 & -18 & 204 & 196 \\ 0.937 & -4 & 208 & 197 \\ 0.918 & -1 & 113 & 198 \\ 0.926 & -4 & 116 & 193 \\ 0.917 & -15 & 82 & 193 \\ 0.919 & 6 & 90 & 196 \\ 0.870 & 38 & 48 & 199 \\ 0.864 & 32 & 77 & 202 \end{bmatrix}. \quad (3.23)$$

We also conduct zero mean scaling and unit variance scaling on each variable of $\mathbf{X}^1$ and perform an eigenvalue decomposition on $\mathbf{X}^{1T}\mathbf{W}^T\mathbf{W}\mathbf{X}^1$ to obtain the principal components for $\mathbf{X}^1$. The principal component coefficients and the contribution rate for each principal component are given in Table 3.14.

In this example, we set $\lambda = 0.95$. From Table 3.13, the cumulative contribution rate of the first principal component is $0.8180 < 0.95$ and the cumulative contribution rate of the first two principal components is $0.9592 > 0.95$. Thus, we find the smallest $k = 2$ such that the cumulative contribution rate of the first k principal components reaches λ . Therefore,

Table 3.14: Principal component coefficients and contribution rate for each principal component of $\mathbf{X}^1$ based on the DIVEWPCA algorithm.

Variable	PC1	PC2	PC3	PC4
Specific Gravity	0.5463	0.1280	0.1894	0.8058
Freezing Point	-0.5452	-0.0702	0.8136	0.1896
Iodine Value	0.5351	-0.6696	0.3815	-0.3461
Saponification Value	-0.3435	-0.7283	-0.3959	0.4415
Contribution Rate	0.8450	0.1317	0.0143	0.0090
Cumulative Contribution Rate	0.8450	0.9767	0.9910	1.0000

Table 3.15: First two PCs of $\mathbf{X}^s$ (Y_1^s and Y_2^s) and first two PCs of $\mathbf{X}^l$ (Y_1^l and Y_2^l).

Sample	Y_1^s	Y_2^s	Y_1^l	Y_2^l
Y_1	0.6698	0.1754	0.4218	-0.1349
Y_2	0.1297	-0.1543	0.1940	-0.1202
Y_3	0.0012	-0.0573	0.0112	-0.0229
Y_4	0.0147	-0.0506	0.0860	0.0939
Y_5	0.0155	-0.0385	0.0510	0.0966
Y_6	-0.0303	-0.0250	-0.0050	0.0534
Y_7	-0.2941	0.1559	-0.3380	-0.0074
Y_8	-0.2902	0.1602	-0.3878	-0.1661

we keep the first two principal components of $\mathbf{X}^s$. Likewise, from Table 3.14, the smallest k such that the cumulative contribution rate of the first k principal components of $\mathbf{X}^1$ reaches $\lambda = 0.95$ is 2 and thus we keep the first two principal components of $\mathbf{X}^1$. The first two principal components of $\mathbf{X}^s$ (Y_1^s and Y_2^s) and the first two principal components of $\mathbf{X}^l$ (Y_1^l and Y_2^l) are given in Table 3.15. Then, we can give four ordering criteria to the interval-valued objects by increasing values of the elements of each of these four principal components during the hierarchical clustering process.

Now, we obtain four ordering criteria based on the weighted symbolic PCA method to reduce the number of possible bipartitions at each clustering stage. Then, we need to find the “best” bipartition at each clustering stage by the minimum total within-cluster variation criterion described in Section 2.2.2. To calculate the change in the total within-cluster variation Δ_v , we first need to calculate the distance matrix. In this example, we consider the span normalized Euclidean Hausdorff distance and the Ichino-Yaguchi distance. The distance matrix based on the span normalized Euclidean Hausdorff distance is

$$\mathbf{D}_{\text{SNEH}} = \begin{bmatrix} 0.0000 & 0.9093 & 1.0763 & 1.0342 & 1.1411 & 1.1614 & 1.7751 & 1.6518 \\ 0.9093 & 0.0000 & 0.6163 & 0.5673 & 0.8303 & 0.7544 & 1.4527 & 1.3597 \\ 1.0763 & 0.6163 & 0.0000 & 0.1297 & 0.3017 & 0.1776 & 1.0061 & 0.9349 \\ 1.0342 & 0.5673 & 0.1297 & 0.0000 & 0.3283 & 0.2382 & 1.0786 & 1.0131 \\ 1.1411 & 0.8303 & 0.3017 & 0.3283 & 0.0000 & 0.3295 & 1.1087 & 1.0484 \\ 1.1614 & 0.7544 & 0.1776 & 0.2382 & 0.3295 & 0.0000 & 0.8794 & 0.8316 \\ 1.7751 & 1.4527 & 1.0061 & 1.0786 & 1.1087 & 0.8794 & 0.0000 & 0.2280 \\ 1.6518 & 1.3597 & 0.9349 & 1.0131 & 1.0484 & 0.8316 & 0.2280 & 0.0000 \end{bmatrix} \quad (3.24)$$

and the distance matrix based on the Ichino-Yaguchi distance is

$$\mathbf{D}_{\text{IY}} = \begin{bmatrix} 0.00 & 77.20 & 150.02 & 143.41 & 159.42 & 175.42 & 250.47 & 226.07 \\ 77.20 & 0.00 & 100.62 & 96.41 & 138.02 & 127.62 & 200.27 & 175.87 \\ 150.02 & 100.62 & 0.00 & 10.81 & 45.20 & 34.00 & 104.25 & 79.86 \\ 143.41 & 96.41 & 10.81 & 0.00 & 45.41 & 38.41 & 113.46 & 89.06 \\ 159.42 & 138.02 & 45.20 & 45.41 & 0.00 & 30.60 & 95.65 & 71.26 \\ 175.42 & 127.62 & 34.00 & 38.41 & 30.60 & 0.00 & 78.45 & 54.06 \\ 250.47 & 200.27 & 104.25 & 113.46 & 95.65 & 78.45 & 0.00 & 34.40 \\ 226.07 & 175.87 & 79.86 & 89.06 & 71.26 & 54.06 & 34.40 & 0.00 \end{bmatrix}. \quad (3.25)$$

Now, we begin the divisive hierarchical clustering and suppose the span normalized Euclidean Hausdorff distance is used. At each clustering stage, we first use the four ordering criteria based on the PC values given in Table 3.15 to reduce the number of possible

bipartitions and then use the distance matrix given in Eq.(3.24) to calculate the change in the total within-cluster variation Δ_v for each possible bipartition. By the minimum total within-cluster variation criterion, we can find the “best” bipartition by finding the bipartition which can maximize Δ_v . At the first clustering stage, we start with a single cluster $\Omega = C_1 = \{Y_1, \dots, Y_8\}$. For example, if we order them by increasing values of the elements of Y_1^s , the eight observations after the ordering are $Y_7, Y_8, Y_6, Y_3, Y_4, Y_5, Y_2, Y_1$. By this ordering, the number of possible bipartitions of C_1 are reduced to seven and the seven bipartitions are $(\{Y_7\}, \{Y_8, Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$, $(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$, $(\{Y_7, Y_8, Y_6\}, \{Y_3, Y_4, Y_5, Y_2, Y_1\})$, $(\{Y_7, Y_8, Y_6, Y_3\}, \{Y_4, Y_5, Y_2, Y_1\})$, $(\{Y_7, Y_8, Y_6, Y_3, Y_4\}, \{Y_5, Y_2, Y_1\})$, $(\{Y_7, Y_8, Y_6, Y_3, Y_4, Y_5\}, \{Y_2, Y_1\})$ and $(\{Y_7, Y_8, Y_6, Y_3, Y_4, Y_5, Y_2\}, \{Y_1\})$.

Likewise, we order the observations by increasing values of the elements of the other PCs given in Table 3.15 and thus we obtain 28 possible bipartitions in total at the first clustering stage. The 28 possible bipartitions and the corresponding Δ_v 's are shown in Table 3.16. Notice that the possible bipartitions obtained by different PCs may be duplicate. Based on the result given in Table 3.16, the bipartition which maximizes Δ_v is $(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$ and this bipartition can be achieved by Y_1^s or Y_1^l . Thus, we can select one of them as the cut criterion at the first clustering stage. From Table 3.15, the value of Y_1^s for Y_8 is -0.2902 and the value of Y_1^s for Y_6 is -0.0303. Thus, the cut criterion at the first clustering stage can be written as “Is $Y_1^s \leq (-0.2902 - 0.0303)/2 = -0.16025$?”.

At the first clustering stage, Ω is divided into $(C_1, C_2) = (\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$. At the second stage, we need to further divide one of these two clusters. Likewise, we order the observations by increasing values of the elements of the PCs given in Table 3.15 and thus we obtain 24 possible bipartitions in total at the second clustering stage. The 24 possible bipartitions and the corresponding Δ_v 's are shown in Table 3.17.

Based on the result given in Table 3.17, the bipartition which maximizes Δ_v is $(\{Y_8, Y_7\}, \{Y_1\}, \{Y_2, Y_3, Y_6, Y_4, Y_5\})$ and this bipartition can be achieved by all the four PCs. Thus, we can select one of them as the cut criterion at the second clustering stage.

Table 3.16: Possible bipartitions and the corresponding Δ_v 's at the first clustering stage of DIVEWPCA based on the span normalized Euclidean Hausdorff distance.

Criterion	Possible Bipartition	Δ_v
Y_1^s	$(\{Y_7\}, \{Y_8, Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$	0.8945
	$(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$	1.8574
	$(\{Y_7, Y_8, Y_6\}, \{Y_3, Y_4, Y_5, Y_2, Y_1\})$	1.5314
	$(\{Y_7, Y_8, Y_6, Y_3\}, \{Y_4, Y_5, Y_2, Y_1\})$	1.2917
	$(\{Y_7, Y_8, Y_6, Y_3, Y_4\}, \{Y_5, Y_2, Y_1\})$	1.1507
	$(\{Y_7, Y_8, Y_6, Y_3, Y_4, Y_5\}, \{Y_2, Y_1\})$	1.4274
	$(\{Y_7, Y_8, Y_6, Y_3, Y_4, Y_5, Y_2\}, \{Y_1\})$	1.1937
Y_2^s	$(\{Y_2\}, \{Y_3, Y_4, Y_5, Y_6, Y_7, Y_8, Y_1\})$	0.5023
	$(\{Y_2, Y_3\}, \{Y_4, Y_5, Y_6, Y_7, Y_8, Y_1\})$	0.3885
	$(\{Y_2, Y_3, Y_4\}, \{Y_5, Y_6, Y_7, Y_8, Y_1\})$	0.4934
	$(\{Y_2, Y_3, Y_4, Y_5\}, \{Y_6, Y_7, Y_8, Y_1\})$	0.6418
	$(\{Y_2, Y_3, Y_4, Y_5, Y_6\}, \{Y_7, Y_8, Y_1\})$	0.7780
	$(\{Y_2, Y_3, Y_4, Y_5, Y_6, Y_7\}, \{Y_8, Y_1\})$	0.4220
	$(\{Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8\}, \{Y_1\})$	1.1937
Y_1^l	$(\{Y_8\}, \{Y_7, Y_6, Y_3, Y_5, Y_4, Y_2, Y_1\})$	0.7272
	$(\{Y_8, Y_7\}, \{Y_6, Y_3, Y_5, Y_4, Y_2, Y_1\})$	1.8574
	$(\{Y_8, Y_7, Y_6\}, \{Y_3, Y_5, Y_4, Y_2, Y_1\})$	1.5314
	$(\{Y_8, Y_7, Y_6, Y_3\}, \{Y_5, Y_4, Y_2, Y_1\})$	1.2917
	$(\{Y_8, Y_7, Y_6, Y_3, Y_5\}, \{Y_4, Y_2, Y_1\})$	1.2979
	$(\{Y_8, Y_7, Y_6, Y_3, Y_5, Y_4\}, \{Y_2, Y_1\})$	1.4274
	$(\{Y_8, Y_7, Y_6, Y_3, Y_5, Y_4, Y_2\}, \{Y_1\})$	1.1937
Y_2^l	$(\{Y_8\}, \{Y_1, Y_2, Y_3, Y_7, Y_6, Y_4, Y_5\})$	0.7272
	$(\{Y_8, Y_1\}, \{Y_2, Y_3, Y_7, Y_6, Y_4, Y_5\})$	0.4220
	$(\{Y_8, Y_1, Y_2\}, \{Y_3, Y_7, Y_6, Y_4, Y_5\})$	0.5103
	$(\{Y_8, Y_1, Y_2, Y_3\}, \{Y_7, Y_6, Y_4, Y_5\})$	0.4159
	$(\{Y_8, Y_1, Y_2, Y_3, Y_7\}, \{Y_6, Y_4, Y_5\})$	0.3103
	$(\{Y_8, Y_1, Y_2, Y_3, Y_7, Y_6\}, \{Y_4, Y_5\})$	0.2498
	$(\{Y_8, Y_1, Y_2, Y_3, Y_7, Y_6, Y_4\}, \{Y_5\})$	0.1996

Table 3.17: Possible bipartitions and the corresponding Δ_v 's at the second clustering stage of DIVEWPCA based on the span normalized Euclidean Hausdorff distance.

Criterion	Possible Bipartition	Δ_v
Y_1^s	$(\{Y_7\}, \{Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$	0.0260
	$(\{Y_7, Y_8\}, \{Y_6\}, \{Y_3, Y_4, Y_5, Y_2, Y_1\})$	0.1537
	$(\{Y_7, Y_8\}, \{Y_6, Y_3\}, \{Y_4, Y_5, Y_2, Y_1\})$	0.2513
	$(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4\}, \{Y_5, Y_2, Y_1\})$	0.3719
	$(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5\}, \{Y_2, Y_1\})$	0.8299
	$(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2\}, \{Y_1\})$	0.8719
Y_2^s	$(\{Y_7\}, \{Y_8\}, \{Y_2, Y_3, Y_4, Y_5, Y_6, Y_1\})$	0.0260
	$(\{Y_7, Y_8\}, \{Y_2\}, \{Y_3, Y_4, Y_5, Y_6, Y_1\})$	0.2881
	$(\{Y_7, Y_8\}, \{Y_2, Y_3\}, \{Y_4, Y_5, Y_6, Y_1\})$	0.1581
	$(\{Y_7, Y_8\}, \{Y_2, Y_3, Y_4\}, \{Y_5, Y_6, Y_1\})$	0.1871
	$(\{Y_7, Y_8\}, \{Y_2, Y_3, Y_4, Y_5\}, \{Y_6, Y_1\})$	0.2703
	$(\{Y_7, Y_8\}, \{Y_2, Y_3, Y_4, Y_5, Y_6\}, \{Y_1\})$	0.8719
Y_1^l	$(\{Y_8\}, \{Y_7\}, \{Y_6, Y_3, Y_5, Y_4, Y_2, Y_1\})$	0.0260
	$(\{Y_8, Y_7\}, \{Y_6\}, \{Y_3, Y_5, Y_4, Y_2, Y_1\})$	0.1537
	$(\{Y_8, Y_7\}, \{Y_6, Y_3\}, \{Y_5, Y_4, Y_2, Y_1\})$	0.2513
	$(\{Y_8, Y_7\}, \{Y_6, Y_3, Y_5\}, \{Y_4, Y_2, Y_1\})$	0.5300
	$(\{Y_8, Y_7\}, \{Y_6, Y_3, Y_5, Y_4\}, \{Y_2, Y_1\})$	0.8299
	$(\{Y_8, Y_7\}, \{Y_6, Y_3, Y_5, Y_4, Y_2\}, \{Y_1\})$	0.8719
Y_2^l	$(\{Y_8\}, \{Y_7\}, \{Y_1, Y_2, Y_3, Y_6, Y_4, Y_5\})$	0.0260
	$(\{Y_8, Y_7\}, \{Y_1\}, \{Y_2, Y_3, Y_6, Y_4, Y_5\})$	0.8719
	$(\{Y_8, Y_7\}, \{Y_1, Y_2\}, \{Y_3, Y_6, Y_4, Y_5\})$	0.8299
	$(\{Y_8, Y_7\}, \{Y_1, Y_2, Y_3\}, \{Y_6, Y_4, Y_5\})$	0.4670
	$(\{Y_8, Y_7\}, \{Y_1, Y_2, Y_3, Y_6\}, \{Y_4, Y_5\})$	0.2139
	$(\{Y_8, Y_7\}, \{Y_1, Y_2, Y_3, Y_6, Y_4\}, \{Y_5\})$	0.1905

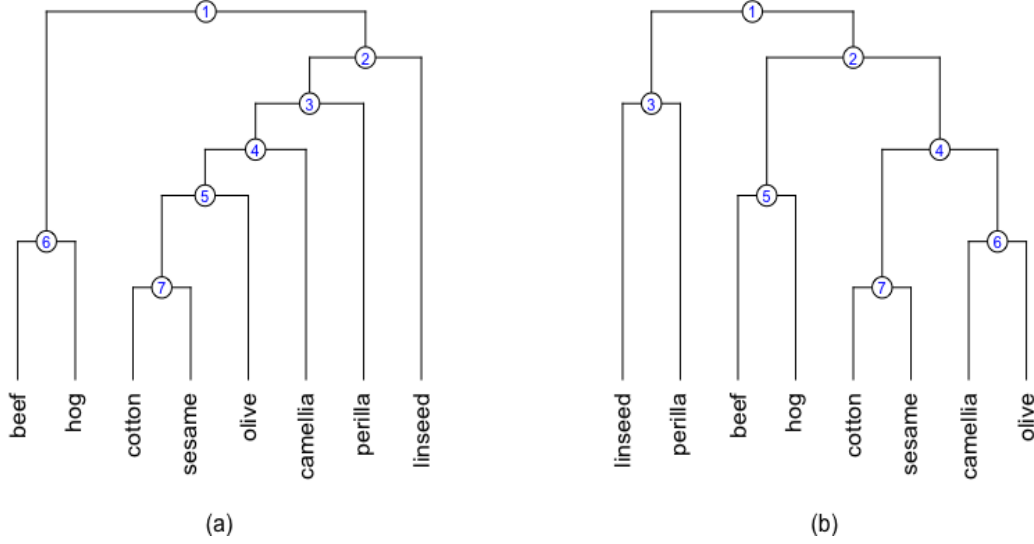


Figure 3.14: Dendrograms for the fats and oils dataset using the DIVEWPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

At this stage, $C_2 = \{Y_1, Y_2, Y_3, Y_6, Y_4, Y_5\}$ is divided into $\{Y_1\}$ and $\{Y_2, Y_3, Y_6, Y_4, Y_5\}$. From Table 3.15, the value of Y_2^l for Y_1 is -0.1349 and the value of Y_2^l for Y_2 is -0.1202 . Thus, the cut criterion at the second clustering stage can be written as “Is $Y_2^l \leq (-0.1349 - 0.1202)/2 = -0.12755$?”.

At the remaining stages, we repeat this process to continue the bipartition until Ω is divided into eight clusters and there is only one observation in each cluster. The dendrogram for the whole process is given in Figure 3.14(a). Besides, if we use the distance matrix based on the Ichino-Yaguchi distance given in Eq.(3.25) to calculate the change in the total within-cluster variation Δ_v for each possible bipartition at each clustering stage, we achieve the dendrogram which is shown in Figure 3.14(b). The cut criterion at each clustering stage (partition node) of the two dendrograms is summarized in Table 3.18.

Table 3.18: Cut criterion at each clustering stage (partition node) of the two dendrograms given in Figure 3.14.

Stage	Dendrogram (a)	Dendrogram (b)
Stage 1	" $Y_1^s \leq -0.16025?$ "	" $Y_1^l \leq 0.14000?$ "
Stage 2	" $Y_2^l \leq -0.12755?$ "	" $Y_2^s \leq 0.06545?$ "
Stage 3	" $Y_1^s \leq 0.0726?$ "	" $Y_1^s \leq 0.39975?$ "
Stage 4	" $Y_1^s \leq 0.01510?$ "	" $Y_2^s \leq -0.04455?$ "
Stage 5	" $Y_1^s \leq -0.01455?$ "	" $Y_1^s \leq -0.29215?$ "
Stage 6	" $Y_1^s \leq -0.29215?$ "	" $Y_1^s \leq -0.00740?$ "
Stage 7	" $Y_1^s \leq 0.00795?$ "	" $Y_1^s \leq 0.00795?$ "

Table 3.19: Validity indices for the clustering result of the fats and oils dataset using the DIVEWPCA algorithm.

Stage	Dendrogram (a)		Dendrogram (b)	
	DI_r^s	DB_r^s	DI_r^s	DB_r^s
Stage 1	1.3795	0.7389	3.1734	0.3952
Stage 2	1.8374	0.4814	3.0812	0.3799
Stage 3	3.6021	0.2443	1.5303	0.4650
Stage 4	1.9429	0.3378	2.4005	0.3602
Stage 5	1.0244	0.4990	1.2639	0.4397
Stage 6	3.0920	0.2128	8.0189	0.0791
Stage 7	$+\infty$	0.0000	$+\infty$	0.0000

After the hierarchical clustering, we need to choose the result of one clustering stage as the final clustering result of each dendrogram by comparing the Dunn index values, DI_r^s , or the Davis-Bouldin index values, DB_r^s , of all the clustering stages of each dendrogram. These two indices were extended by Kim (2009) and Kim and Billard (2011) from classical data to symbolic data and the definitions are given in Eq.(2.18) and Eq.(2.19). Usually, a larger DI_r^s or a smaller DB_r^s means a better clustering outcome. We calculate the two indices values for the outcome of each clustering stage and the values are given in Table 3.19. From the table, we find that the results of Stage 3, Stage 6 and Stage 7 are often better than the results of the other clustering stages. However, Stage 6 and Stage 7 are the last two stages, which means most of the clusters from the results of these two stages contain only one observation. Thus, we prefer to choose the result of Stage 3 as the final clustering result. For the dendrogram based on the span normalized Euclidean Hausdorff distance, the result of Stage 3 is $(\{Y_7, Y_8\}, \{Y_1\}, \{Y_6, Y_3, Y_4, Y_5\}, \{Y_2\})$. For the dendrogram based on the Ichino-Yaguchi distance, the result of Stage 3 is $(\{Y_3, Y_4, Y_5, Y_6\}, \{Y_2\}, \{Y_7, Y_8\}, \{Y_1\})$. Thus, the two dendrograms give the same final clustering result though the two dendrograms are different. Therefore, the final clustering result of the DIVEWPCA algorithm on this dataset is $(\{Y_7, Y_8\}, \{Y_1\}, \{Y_6, Y_3, Y_4, Y_5\}, \{Y_2\})$, no matter which of the two distance measures is used in the algorithm. The result is reasonable since Y_1 and Y_2 are drying oils which are key components of oil paint, Y_7 and Y_8 are fat of animals, and Y_6, Y_3, Y_4 and Y_5 are mainly for cooking.

CLUSTERING PROCEDURES OF DIVECPCA

To run DIVECPCA on this dataset, the first step is to calculate the Pearson correlation coefficient between the four variables by Eq.(2.7) and thus obtain the symbolic correlation

matrix

$$\mathbf{R} = \begin{bmatrix} 1.0000 & -0.8643 & 0.7843 & -0.3144 \\ -0.8643 & 1.0000 & -0.6160 & 0.5654 \\ 0.7843 & -0.6160 & 1.0000 & -0.4438 \\ -0.3144 & 0.5654 & -0.4438 & 1.0000 \end{bmatrix}. \quad (3.26)$$

The reason that we use the (sample) correlation matrix instead of the sample covariance matrix here is the four variables in this dataset are measured on different scales. Then, we perform an eigenvalue decomposition on the correlation matrix $\mathbf{R}$ and use the eigenvectors to obtain the principal components of $\mathbf{X}^s$ and $\mathbf{X}^l$. The $\mathbf{X}^s$ and $\mathbf{X}^l$ are given in Eq.(3.22) and Eq.(3.23) and we also need to perform zero mean scaling and unit variance scaling on the two matrices because of the different scales of the variables. The principal component coefficients and the contribution rate for each principal component based on the eigenvalue decomposition are given in Table 3.20. In this example, we set $\lambda = 0.95$. From Table 3.20, the cumulative contribution rate of the first two principal components is $0.8929 < 0.95$ and the cumulative contribution rate of the first three principal components is $0.9895 > 0.95$. Therefore, we keep both the first three principal components of $\mathbf{X}^s$ and the first three principal components of $\mathbf{X}^l$. The first three principal components of $\mathbf{X}^s$ (Y_1^s , Y_2^s and Y_3^s) and the first three principal components of $\mathbf{X}^l$ (Y_1^l , Y_2^l and Y_3^l) are given in Table 3.21. Then, we can give six ordering criteria to the interval-valued objects by increasing values of the elements of each of these six principal components during the hierarchical clustering process.

Now, we obtain six ordering criteria based on the six principal components given in Table 3.21. We begin the divisive hierarchical clustering process and suppose the span normalized Euclidean Hausdorff distance is used. At each clustering stage, we first use the six ordering criteria to order the observations and then use the distance matrix given in Eq.(3.24) to calculate the change in the total within-cluster variation Δ_v for each possible bipartition based on the ordering criteria. At the first clustering stage, we start with a single cluster $\Omega = C_1 = \{Y_1, \dots, Y_8\}$. The 42 possible bipartitions and the corresponding Δ_v 's are shown in Table 3.22. Based on the result given in Table 3.22, the bipartition which maximizes Δ_v

Table 3.20: Principal component coefficients and contribution rate for each principal component of $\mathbf{X}^s$ and $\mathbf{X}^l$ based on the DIVECPCA algorithm.

Variable	PC1	PC2	PC3	PC4
Specific Gravity	0.5421	-0.4266	0.1800	0.7012
Freezing Point	-0.5462	0.0036	-0.6047	0.5796
Iodine Value	0.5093	-0.2078	-0.7703	-0.3225
Saponification Value	-0.3852	-0.8802	0.0922	-0.2614
Contribution Rate	0.7078	0.1851	0.0966	0.0105
Cumulative Contribution Rate	0.7078	0.8929	0.9895	1.0000

Table 3.21: First three principal components of $\mathbf{X}^s$ (Y_1^s , Y_2^s and Y_3^s) and first three principal components of $\mathbf{X}^l$ (Y_1^l , Y_2^l and Y_3^l).

Sample	Y_1^s	Y_2^s	Y_3^s	Y_1^l	Y_2^l	Y_3^l
Y_1	2.7666	1.5514	-0.2643	1.9089	-0.4599	-0.3481
Y_2	1.2769	-0.9969	-1.0094	1.4812	-0.7943	-0.7701
Y_3	0.1448	-0.4672	0.2801	0.0832	-0.4614	0.2953
Y_4	0.2972	-0.4743	0.2251	0.9831	0.8672	0.2427
Y_5	0.3864	-0.3961	1.0219	0.8077	1.1223	0.9552
Y_6	-0.2210	-0.2888	0.3615	-0.0275	0.1905	0.3355
Y_7	-2.4817	0.5475	-0.3324	-2.5741	0.2124	-0.2804
Y_8	-2.1693	0.5245	-0.2824	-2.6625	-0.6767	-0.4301

is $(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$ and this bipartition can be achieved by both Y_1^s and Y_1^l . This bipartition is the same as the result of the first clustering stage of DIVEWPCA. From Table 3.21, the value of Y_1^s for Y_8 is -2.1693 and the value of Y_1^s for Y_6 is -0.2210. Thus, if we select Y_1^s as the cut criterion at the first clustering stage, the cut criterion can be written as “Is $Y_1^s \leq (-2.1693 - 0.2210)/2 = -1.19515$?”.

At the second stage, we start with the partition $(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$. We order the observations within each cluster by increasing values of the elements of the six PCs, which leads to 36 possible bipartitions. The 36 possible bipartitions and the corresponding Δ_v 's are shown in Table 3.23. The bipartition which maximizes Δ_v is $(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2\}, \{Y_1\})$ and this bipartition can be achieved by Y_1^s , Y_2^s or Y_1^l . If we select Y_1^s as the cut criterion at this clustering stage, the cut criterion can be written as “Is $Y_1^s \leq (2.7666 + 1.2769)/2 = 2.02175$?”.

At the remaining stages, we continue the bipartition until Ω is divided into eight clusters. Based on the partition result of each stage, we achieve the dendrogram given in Figure 3.15(a). The dendrogram is the same as the dendrogram given in Figure 3.14(a), which was achieved by the DIVEWPCA algorithm based on the span normalized Euclidean Hausdorff distance. In addition, if we use the distance matrix based on the Ichino-Yaguchi distance given in Eq.(3.25) to calculate Δ_v for each possible bipartition at each clustering stage, we achieve the dendrogram in Figure 3.15(b), which is the same as the dendrogram achieved by the DIVEWPCA algorithm and shown in Figure 3.14(b). Furthermore, the same dendrograms achieved by the two algorithms lead to the same values of the two validity indices DI_r^s and DB_r^s which were given in Table 3.19. Therefore, we also choose the result of Stage 3 as the final clustering result of the DIVECPCA algorithm on this dataset and the final clustering result is also $(\{Y_7, Y_8\}, \{Y_1\}, \{Y_6, Y_3, Y_4, Y_5\}, \{Y_2\})$, no matter which of the two distance measures is used in the algorithm. The only difference between the corresponding dendrograms achieved by the two algorithms is the cut criterion at each clustering stage (partition node) of the corresponding dendrograms since the cut criterion is based on the

Table 3.22: Possible bipartitions and the corresponding Δ_v 's at the first clustering stage of DIVEPCPA based on the span normalized Euclidean Hausdorff distance.

Criterion	Possible Bipartition	Δ_v
Y_1^s	$(\{Y_7\}, \{Y_8, Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$	0.8945
	$(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$	1.8574
	$(\{Y_7, Y_8, Y_6\}, \{Y_3, Y_4, Y_5, Y_2, Y_1\})$	1.5314
	$(\{Y_7, Y_8, Y_6, Y_3\}, \{Y_4, Y_5, Y_2, Y_1\})$	1.2917
	$(\{Y_7, Y_8, Y_6, Y_3, Y_4\}, \{Y_5, Y_2, Y_1\})$	1.1507
	$(\{Y_7, Y_8, Y_6, Y_3, Y_4, Y_5\}, \{Y_2, Y_1\})$	1.4274
	$(\{Y_7, Y_8, Y_6, Y_3, Y_4, Y_5, Y_2\}, \{Y_1\})$	1.1937
Y_2^s	$(\{Y_2\}, \{Y_4, Y_3, Y_5, Y_6, Y_8, Y_7, Y_1\})$	0.5023
	$(\{Y_2, Y_4\}, \{Y_3, Y_5, Y_6, Y_8, Y_7, Y_1\})$	0.4602
	$(\{Y_2, Y_4, Y_3\}, \{Y_5, Y_6, Y_8, Y_7, Y_1\})$	0.4934
	$(\{Y_2, Y_4, Y_3, Y_5\}, \{Y_6, Y_8, Y_7, Y_1\})$	0.6418
	$(\{Y_2, Y_4, Y_3, Y_5, Y_6\}, \{Y_8, Y_7, Y_1\})$	0.7780
	$(\{Y_2, Y_4, Y_3, Y_5, Y_6, Y_8\}, \{Y_7, Y_1\})$	0.3355
	$(\{Y_2, Y_4, Y_3, Y_5, Y_6, Y_8, Y_7\}, \{Y_1\})$	1.1937
Y_3^s	$(\{Y_2\}, \{Y_7, Y_8, Y_1, Y_4, Y_3, Y_6, Y_5\})$	0.5023
	$(\{Y_2, Y_7\}, \{Y_8, Y_1, Y_4, Y_3, Y_6, Y_5\})$	0.2228
	$(\{Y_2, Y_7, Y_8\}, \{Y_1, Y_4, Y_3, Y_6, Y_5\})$	0.8345
	$(\{Y_2, Y_7, Y_8, Y_1\}, \{Y_4, Y_3, Y_6, Y_5\})$	0.4473
	$(\{Y_2, Y_7, Y_8, Y_1, Y_4\}, \{Y_3, Y_6, Y_5\})$	0.2931
	$(\{Y_2, Y_7, Y_8, Y_1, Y_4, Y_3\}, \{Y_6, Y_5\})$	0.2189
	$(\{Y_2, Y_7, Y_8, Y_1, Y_4, Y_3, Y_6\}, \{Y_5\})$	0.1996
Y_1^l	$(\{Y_8\}, \{Y_7, Y_6, Y_3, Y_5, Y_4, Y_2, Y_1\})$	0.7272
	$(\{Y_8, Y_7\}, \{Y_6, Y_3, Y_5, Y_4, Y_2, Y_1\})$	1.8574
	$(\{Y_8, Y_7, Y_6\}, \{Y_3, Y_5, Y_4, Y_2, Y_1\})$	1.5314
	$(\{Y_8, Y_7, Y_6, Y_3\}, \{Y_5, Y_4, Y_2, Y_1\})$	1.2917
	$(\{Y_8, Y_7, Y_6, Y_3, Y_5\}, \{Y_4, Y_2, Y_1\})$	1.2979
	$(\{Y_8, Y_7, Y_6, Y_3, Y_5, Y_4\}, \{Y_2, Y_1\})$	1.4274
	$(\{Y_8, Y_7, Y_6, Y_3, Y_5, Y_4, Y_2\}, \{Y_1\})$	1.1937
Y_2^l	$(\{Y_2\}, \{Y_8, Y_3, Y_1, Y_6, Y_7, Y_4, Y_5\})$	0.5023
	$(\{Y_2, Y_8\}, \{Y_3, Y_1, Y_6, Y_7, Y_4, Y_5\})$	0.2020
	$(\{Y_2, Y_8, Y_3\}, \{Y_1, Y_6, Y_7, Y_4, Y_5\})$	0.1335
	$(\{Y_2, Y_8, Y_3, Y_1\}, \{Y_6, Y_7, Y_4, Y_5\})$	0.4159
	$(\{Y_2, Y_8, Y_3, Y_1, Y_6\}, \{Y_7, Y_4, Y_5\})$	0.3049
	$(\{Y_2, Y_8, Y_3, Y_1, Y_6, Y_7\}, \{Y_4, Y_5\})$	0.2498
	$(\{Y_2, Y_8, Y_3, Y_1, Y_6, Y_7, Y_4\}, \{Y_5\})$	0.1996
Y_3^l	$(\{Y_2\}, \{Y_8, Y_1, Y_7, Y_4, Y_3, Y_6, Y_5\})$	0.5023
	$(\{Y_2, Y_8\}, \{Y_1, Y_7, Y_4, Y_3, Y_6, Y_5\})$	0.2020
	$(\{Y_2, Y_8, Y_1\}, \{Y_7, Y_4, Y_3, Y_6, Y_5\})$	0.5103
	$(\{Y_2, Y_8, Y_1, Y_7\}, \{Y_4, Y_3, Y_6, Y_5\})$	0.4473
	$(\{Y_2, Y_8, Y_1, Y_7, Y_4\}, \{Y_3, Y_6, Y_5\})$	0.2931
	$(\{Y_2, Y_8, Y_1, Y_7, Y_4, Y_3\}, \{Y_6, Y_5\})$	0.2189
	$(\{Y_2, Y_8, Y_1, Y_7, Y_4, Y_3, Y_6\}, \{Y_5\})$	0.1996

Table 3.23: Possible bipartitions and the corresponding Δ_v 's at the second clustering stage of DIVECPA based on the span normalized Euclidean Hausdorff distance.

Criterion	Possible Bipartition	Δ_v
Y_1^s	$(\{Y_7\}, \{Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2, Y_1\})$	0.0260
	$(\{Y_7, Y_8\}, \{Y_6\}, \{Y_3, Y_4, Y_5, Y_2, Y_1\})$	0.1537
	$(\{Y_7, Y_8\}, \{Y_6, Y_3\}, \{Y_4, Y_5, Y_2, Y_1\})$	0.2513
	$(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4\}, \{Y_5, Y_2, Y_1\})$	0.3719
	$(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5\}, \{Y_2, Y_1\})$	0.8299
	$(\{Y_7, Y_8\}, \{Y_6, Y_3, Y_4, Y_5, Y_2\}, \{Y_1\})$	0.8719
Y_2^s	$(\{Y_8\}, \{Y_7\}, \{Y_2, Y_4, Y_3, Y_5, Y_6, Y_1\})$	0.0260
	$(\{Y_8, Y_7\}, \{Y_2\}, \{Y_4, Y_3, Y_5, Y_6, Y_1\})$	0.2881
	$(\{Y_8, Y_7\}, \{Y_2, Y_4\}, \{Y_3, Y_5, Y_6, Y_1\})$	0.1754
	$(\{Y_8, Y_7\}, \{Y_2, Y_4, Y_3\}, \{Y_5, Y_6, Y_1\})$	0.1871
	$(\{Y_8, Y_7\}, \{Y_2, Y_4, Y_3, Y_5\}, \{Y_6, Y_1\})$	0.2703
	$(\{Y_8, Y_7\}, \{Y_2, Y_4, Y_3, Y_5, Y_6\}, \{Y_1\})$	0.8719
Y_3^s	$(\{Y_7\}, \{Y_8\}, \{Y_2, Y_1, Y_4, Y_3, Y_6, Y_5\})$	0.0260
	$(\{Y_7, Y_8\}, \{Y_2\}, \{Y_1, Y_4, Y_3, Y_6, Y_5\})$	0.2881
	$(\{Y_7, Y_8\}, \{Y_2, Y_1\}, \{Y_4, Y_3, Y_6, Y_5\})$	0.8299
	$(\{Y_7, Y_8\}, \{Y_2, Y_1, Y_4\}, \{Y_3, Y_6, Y_5\})$	0.5300
	$(\{Y_7, Y_8\}, \{Y_2, Y_1, Y_4, Y_3\}, \{Y_6, Y_5\})$	0.3488
	$(\{Y_7, Y_8\}, \{Y_2, Y_1, Y_4, Y_3, Y_6\}, \{Y_5\})$	0.1905
Y_1^l	$(\{Y_8\}, \{Y_7\}, \{Y_6, Y_3, Y_5, Y_4, Y_2, Y_1\})$	0.0260
	$(\{Y_8, Y_7\}, \{Y_6\}, \{Y_3, Y_5, Y_4, Y_2, Y_1\})$	0.1537
	$(\{Y_8, Y_7\}, \{Y_6, Y_3\}, \{Y_5, Y_4, Y_2, Y_1\})$	0.2513
	$(\{Y_8, Y_7\}, \{Y_6, Y_3, Y_5\}, \{Y_4, Y_2, Y_1\})$	0.5300
	$(\{Y_8, Y_7\}, \{Y_6, Y_3, Y_5, Y_4\}, \{Y_2, Y_1\})$	0.8299
	$(\{Y_8, Y_7\}, \{Y_6, Y_3, Y_5, Y_4, Y_2\}, \{Y_1\})$	0.8719
Y_2^l	$(\{Y_8\}, \{Y_7\}, \{Y_2, Y_3, Y_1, Y_6, Y_4, Y_5\})$	0.0260
	$(\{Y_8, Y_7\}, \{Y_2\}, \{Y_3, Y_1, Y_6, Y_4, Y_5\})$	0.2881
	$(\{Y_8, Y_7\}, \{Y_2, Y_3\}, \{Y_1, Y_6, Y_4, Y_5\})$	0.1581
	$(\{Y_8, Y_7\}, \{Y_2, Y_3, Y_1\}, \{Y_6, Y_4, Y_5\})$	0.4670
	$(\{Y_8, Y_7\}, \{Y_2, Y_3, Y_1, Y_6\}, \{Y_4, Y_5\})$	0.2139
	$(\{Y_8, Y_7\}, \{Y_2, Y_3, Y_1, Y_6, Y_4\}, \{Y_5\})$	0.1905
Y_3^l	$(\{Y_8\}, \{Y_7\}, \{Y_2, Y_1, Y_4, Y_3, Y_6, Y_5\})$	0.0260
	$(\{Y_8, Y_7\}, \{Y_2\}, \{Y_1, Y_4, Y_3, Y_6, Y_5\})$	0.2881
	$(\{Y_8, Y_7\}, \{Y_2, Y_1\}, \{Y_4, Y_3, Y_6, Y_5\})$	0.8299
	$(\{Y_8, Y_7\}, \{Y_2, Y_1, Y_4\}, \{Y_3, Y_6, Y_5\})$	0.5300
	$(\{Y_8, Y_7\}, \{Y_2, Y_1, Y_4, Y_3\}, \{Y_6, Y_5\})$	0.3488
	$(\{Y_8, Y_7\}, \{Y_2, Y_1, Y_4, Y_3, Y_6\}, \{Y_5\})$	0.1905

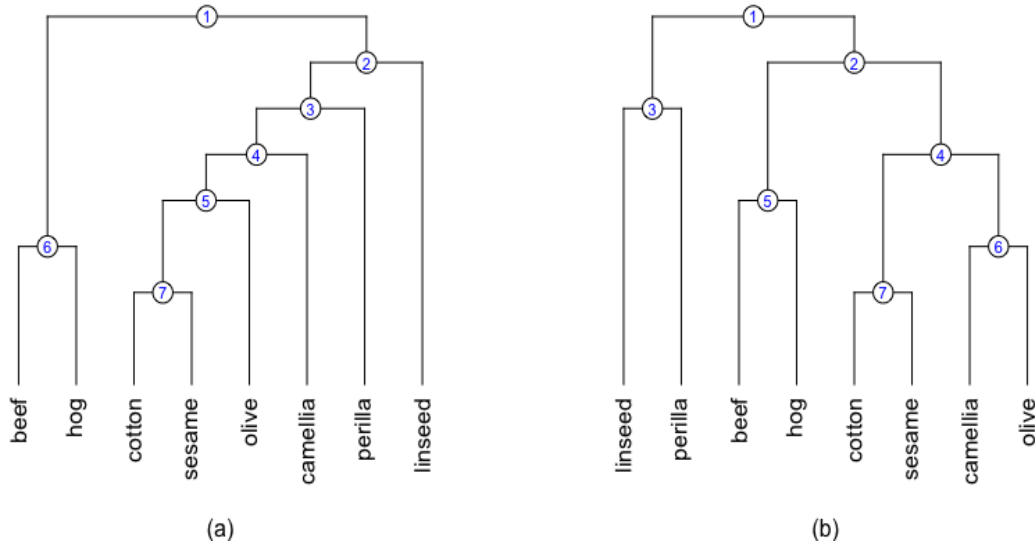


Figure 3.15: Dendrograms for the fats and oils dataset using the DIVECPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

Table 3.24: Cut criterion at each clustering stage (partition node) of the two dendrograms given in Figure 3.15.

Stage	Dendrogram (a)	Dendrogram (b)
Stage 1	" $Y_1^s \leq -1.19515?$ "	" $Y_1^l \leq 1.23215?$ "
Stage 2	" $Y_1^s \leq 2.02175?$ "	" $Y_1^s \leq -1.19515?$ "
Stage 3	" $Y_1^s \leq 0.83165?$ "	" $Y_1^s \leq 2.02175?$ "
Stage 4	" $Y_3^s \leq 0.6917?$ "	" $Y_2^s \leq -0.43165?$ "
Stage 5	" $Y_1^s \leq -0.0381?$ "	" $Y_1^s \leq -2.3255?$ "
Stage 6	" $Y_1^s \leq -2.3255?$ "	" $Y_1^s \leq 0.0827?$ "
Stage 7	" $Y_1^s \leq 0.2210?$ "	" $Y_1^s \leq 0.2210?$ "

principal components and the principal components obtained by these two algorithms are different. Based on the principal components given in Table 3.21, we obtain the cut criterion at each clustering stage of the dendrograms achieved by the DIVECPCA algorithm and the cut criterion is given in Table 3.24.

3.3.2 FACE RECOGNITION DATASET

The face recognition dataset is from Leroy et al. (1996), see also Douzal-Chouakria et al. (2011). The dataset is given in Table 3.25. In this dataset, there are nine subjects and three interval-valued observations for each subject, which leads to 27 observations in total. The three observations for the same subject are labelled by three identical letters and a number from one to three to distinguish them. The observations are interval-valued because each of them is aggregated from a sequence of face images from the same subject. Thus, the underlying real number of clusters in this dataset is nine. There are six variables in this dataset, namely, AD = the length between the outer right eye and the outer left eye, BC = the length between the inner right eye and the inner left eye, AH = the length between the outer right eye and the upper middle lip, DH = the length between the outer left eye and the upper middle lip, EH = the length between the upper middle lip and the right side of the mouth and GH = the length between the upper middle lip and the left side of the mouth. All the lengths are measured by the number of pixels on each face image and thus the variables are measured on the same scale.

We run the two mixed-strategy algorithms DIVEWPCA and DIVECPCA on this dataset and we set $\lambda = 0.9$ for both of them. The clustering procedures are similar to the procedures illustrated by the fats and oils dataset. The difference is when applying DIVEWPCA and DIVECPCA on this dataset, we do zero mean scaling on each variable of the matrix of lower endpoints $\mathbf{X}^s$ and the matrix of upper endpoints $\mathbf{X}^1$ but we do not conduct unit variance scaling on them because the six variables in this dataset are measured on the same scale.

Table 3.25: Face recognition dataset.

id	Subject	$X_1 = AD$	$X_2 = BC$	$X_3 = AH$	$X_4 = DH$	$X_5 = EH$	$X_6 = GH$
1	FRA1	[155.00, 157.00]	[58.00, 61.01]	[100.45, 103.28]	[105.00, 107.30]	[61.40, 65.73]	[64.20, 67.80]
2	FRA2	[154.00, 160.01]	[57.00, 64.00]	[101.98, 105.55]	[104.35, 107.30]	[60.88, 63.03]	[62.94, 66.47]
3	FRA3	[154.01, 161.00]	[57.00, 63.00]	[99.36, 105.65]	[101.04, 109.04]	[60.95, 65.60]	[60.42, 66.40]
4	HUS1	[168.86, 172.84]	[58.55, 63.39]	[102.83, 106.53]	[122.38, 124.52]	[56.73, 61.07]	[60.44, 64.54]
5	HUS2	[169.85, 175.03]	[60.21, 64.38]	[102.94, 108.71]	[120.24, 124.52]	[56.73, 62.37]	[60.44, 66.84]
6	HUS3	[168.76, 175.15]	[61.40, 63.51]	[104.35, 107.45]	[120.93, 125.18]	[57.20, 61.72]	[58.14, 67.08]
7	INC1	[155.26, 160.45]	[53.15, 60.21]	[95.88, 98.49]	[91.68, 94.37]	[62.48, 66.22]	[58.90, 63.13]
8	INC2	[156.26, 161.31]	[51.09, 60.07]	[95.77, 99.36]	[91.21, 96.83]	[54.92, 64.20]	[54.41, 61.55]
9	INC3	[154.47, 160.31]	[55.08, 59.03]	[93.54, 98.98]	[90.43, 96.43]	[59.03, 65.86]	[55.97, 65.80]
10	ISA1	[164.00, 168.00]	[55.01, 60.03]	[120.28, 123.04]	[117.52, 121.02]	[54.38, 57.45]	[50.80, 53.25]
11	ISA2	[163.00, 170.00]	[54.04, 59.00]	[118.80, 123.04]	[116.67, 120.24]	[55.47, 58.67]	[52.43, 55.23]
12	ISA3	[164.01, 169.01]	[55.00, 59.01]	[117.38, 123.11]	[116.67, 122.43]	[52.80, 58.31]	[52.20, 55.47]
13	JPL1	[167.11, 171.19]	[61.03, 65.01]	[118.23, 121.82]	[108.30, 111.20]	[63.89, 67.88]	[57.28, 60.83]
14	JPL2	[169.14, 173.18]	[60.07, 65.07]	[118.85, 120.88]	[108.98, 113.17]	[62.63, 69.07]	[57.38, 61.62]
15	JPL3	[169.03, 170.11]	[59.01, 65.01]	[115.88, 121.38]	[110.34, 112.49]	[61.72, 68.25]	[59.46, 62.94]
16	KHA1	[149.34, 155.54]	[54.15, 59.14]	[111.95, 115.75]	[105.36, 111.07]	[54.20, 58.14]	[48.27, 50.61]
17	KHA2	[149.34, 155.32]	[52.04, 58.22]	[111.20, 113.22]	[105.36, 111.07]	[53.71, 58.14]	[49.41, 52.80]
18	KHA3	[150.33, 157.26]	[52.09, 60.21]	[109.04, 112.70]	[104.74, 111.07]	[55.47, 60.03]	[49.20, 53.41]
19	LOT1	[152.64, 157.62]	[51.35, 56.22]	[116.73, 119.67]	[114.62, 117.41]	[55.44, 59.55]	[53.01, 56.60]
20	LOT2	[154.64, 157.62]	[52.24, 56.32]	[117.52, 119.67]	[114.28, 117.41]	[57.63, 60.61]	[54.41, 57.98]
21	LOT3	[154.83, 157.81]	[50.36, 55.23]	[117.59, 119.75]	[114.04, 116.83]	[56.64, 61.07]	[55.23, 57.80]
22	PHI1	[163.08, 167.07]	[66.03, 68.07]	[115.26, 119.60]	[116.10, 121.02]	[60.96, 65.30]	[57.01, 59.82]
23	PHI2	[164.00, 168.03]	[65.03, 68.12]	[114.55, 119.60]	[115.26, 120.97]	[60.96, 67.27]	[55.32, 61.52]
24	PHI3	[161.01, 167.00]	[64.07, 69.01]	[116.67, 118.79]	[114.59, 118.83]	[61.52, 68.68]	[56.57, 60.11]
25	ROM1	[167.15, 171.24]	[64.07, 68.07]	[123.75, 126.59]	[122.92, 126.37]	[51.22, 54.64]	[49.65, 53.71]
26	ROM2	[168.15, 172.14]	[63.13, 68.07]	[122.33, 127.29]	[124.08, 127.14]	[50.22, 57.14]	[49.93, 56.94]
27	ROM3	[167.11, 171.19]	[63.13, 68.03]	[121.62, 126.57]	[122.58, 127.78]	[49.41, 57.28]	[50.99, 60.46]

The dendrograms for the face recognition dataset using the DIVEWPCA algorithm based on the span normalized Euclidean Hausdorff distance and the Ichino-Yaguchi distance are given in Figure 3.16(a) and Figure 3.16(b), respectively. The two corresponding dendrograms using the DIVECPCA algorithm are given in Figure 3.17(a) and Figure 3.17(b), respectively. From these figures, we can find the two dendrograms obtained from the DIVEWPCA algorithm are the same as are the two corresponding dendrograms obtained from the DIVECPCA algorithm. In other words, the two algorithms give the same clustering result at each clustering stage when they are applied on this dataset.

After the hierarchical clustering, we need to choose the clustering result of one clustering stage as the final clustering result for each dendrogram by comparing the Dunn index values, DI_r^s , or the Davis-Bouldin index values, DB_r^s , of all the clustering stages of each dendrogram. These two indices were extended by Kim (2009) and Kim and Billard (2011) from classical data to symbolic data. The validity indices for the two dendrograms obtained by the DIVEWPCA algorithm are given in Figure 3.18(a) and Figure 3.18(b), respectively. The validity indices for the two dendrograms obtained by the DIVECPCA algorithm are given in Figure 3.19(a) and Figure 3.19(b), respectively. Since a larger DI_r^s or a smaller DB_r^s means a better clustering outcome, we choose the clustering result of Stage 8 of each dendrogram as the final result of each dendrogram by Figure 3.18 and Figure 3.19. More importantly, the clustering results of Stage 8 of these four dendrograms are all $(\{FRA1, FRA2, FRA3\}, \{HUS1, HUS2, HUS3\}, \{INC1, INC2, INC3\}, \{ISA1, ISA2, ISA3\}, \{JPL1, JPL2, JPL3\}, \{KHA1, KHA2, KHA3\}, \{LOT1, LOT2, LOT3\}, \{PHI1, PHI2, PHI3\}, \{ROM1, ROM2, ROM3\})$, which means both of the two algorithms correctly divide the twenty-seven observations into nine clusters, one for each subject, and each cluster contains the three observations from the same subject. Besides, the cut criterion at each clustering stage (partition node) of these four dendrograms can be obtained by the principal components calculated in the corresponding clustering procedures. Let us take the dendrogram given in Figure 3.16(a) as an example and suppose Y_i^s denotes the i^{th}

principal component of $\mathbf{X}^s$ and Y_i^l denotes the i^{th} principal component of $\mathbf{X}^l$. At the seventh stage of this dendrogram, the cluster $\{LOT1, LOT2, LOT3, KHA1, KHA2, KHA3\}$ is divided into $\{KHA1, KHA2, KHA3\}$ and $\{LOT1, LOT2, LOT3\}$ by the DIVEWPCA algorithm. This bipartition maximizes Δ_v at this stage and this bipartition can be achieved by ordering the observations by increasing values of the elements of Y_1^l . The values of Y_1^l for $LOT1, LOT2, LOT3, KHA1, KHA2$ and $KHA3$ are $-0.0819, -0.0598, -0.0431, 0.1263, 0.2034$ and 0.2512 and thus the cut criterion at the seventh stage of this dendrogram is “Is $Y_1^l \leq (-0.0431 + 0.1263)/2 = 0.0416$?”. At the twentieth stage of this dendrogram, the cluster $\{ISA1, ISA2, ISA3\}$ is divided into $\{ISA1, ISA2\}$ and $\{ISA3\}$ by the DIVEWPCA algorithm. This bipartition maximizes Δ_v at this stage and this bipartition can be achieved by ordering the observations by increasing values of the elements of Y_1^s . The values of Y_1^s for $ISA3, ISA2$ and $ISA1$ are $-0.4100, -0.3653$ and -0.3479 and thus the cut criterion at the twentieth stage of this dendrogram is “Is $Y_1^s \leq (-0.3653 - 0.4100)/2 = -0.38765$?”.

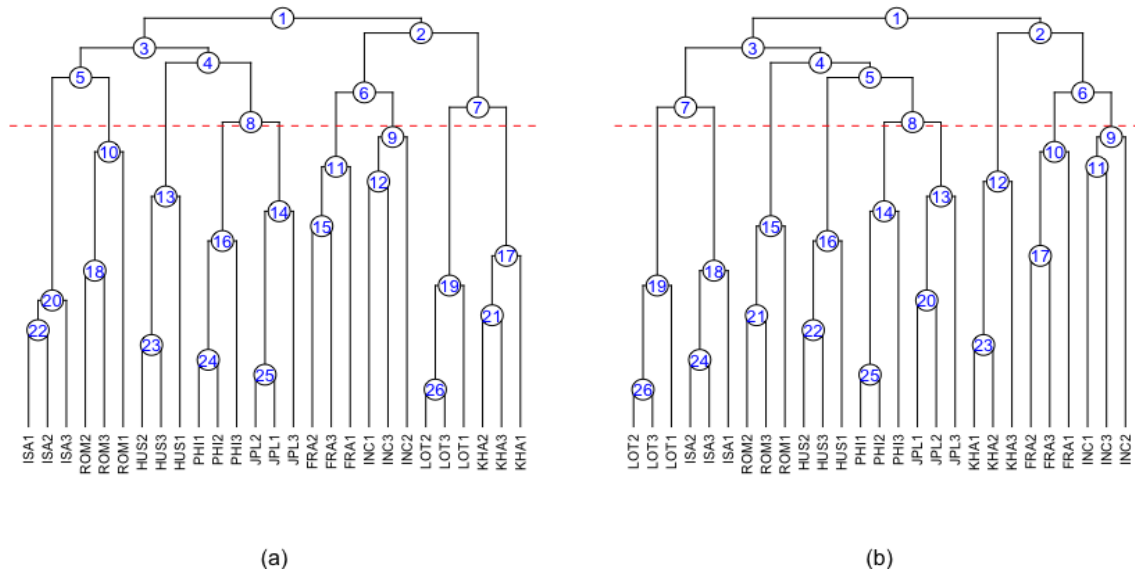


Figure 3.16: Dendrograms for the face recognition dataset using the DIVEWPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

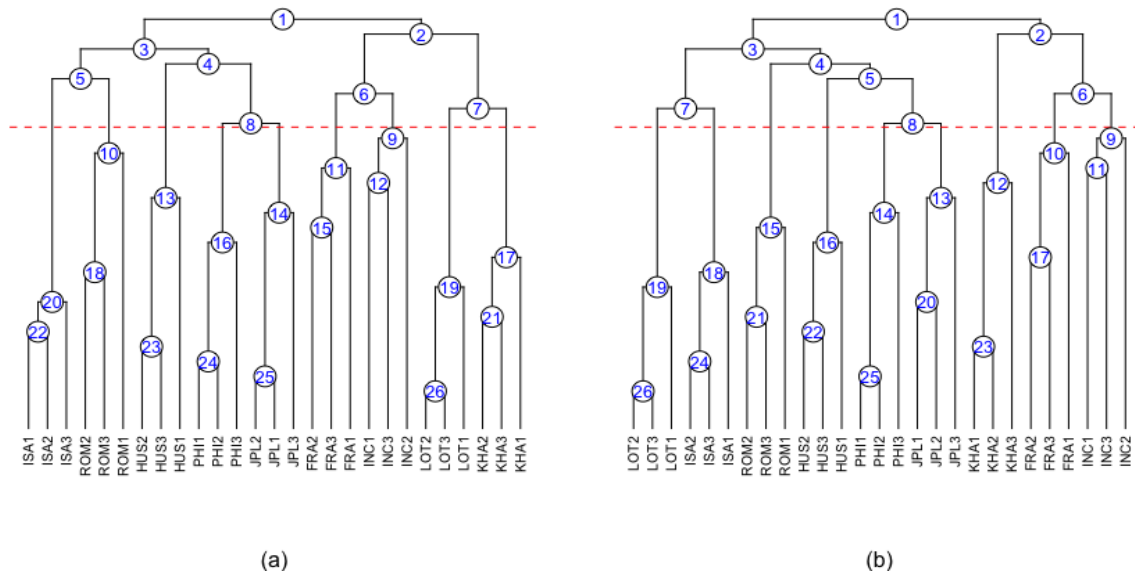


Figure 3.17: Dendrograms for the face recognition dataset using the DIVECPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

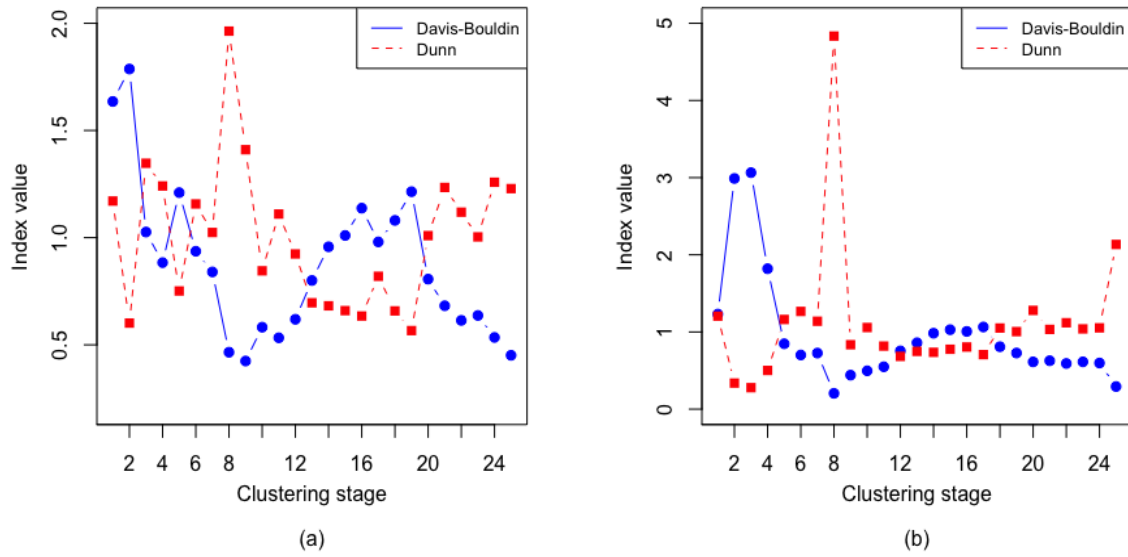


Figure 3.18: Validity indices for the clustering result of the face recognition dataset using the DIVEWPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

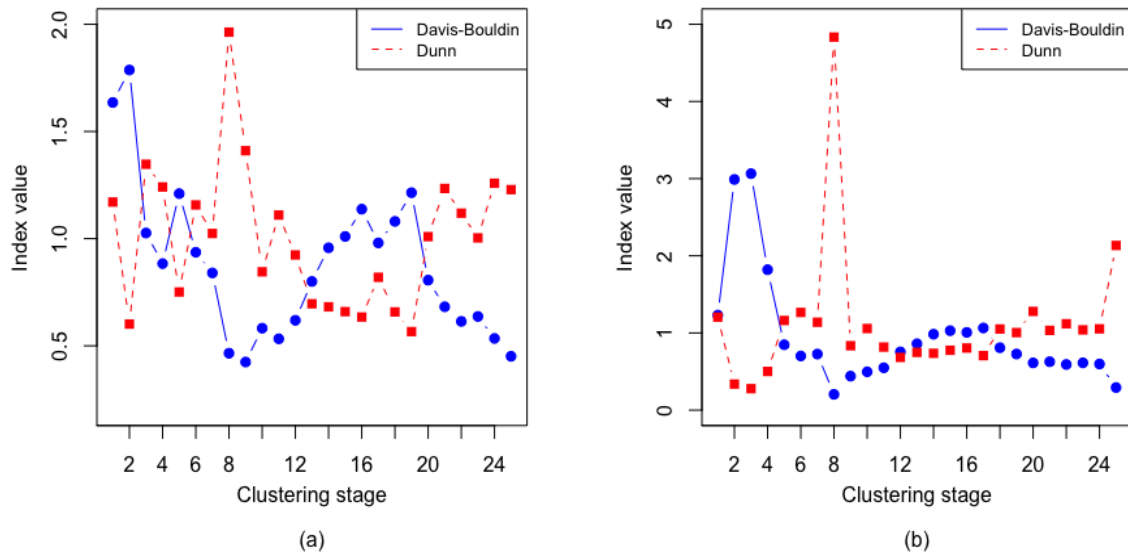


Figure 3.19: Validity indices for the clustering result of the face recognition dataset using the DIVECPCA algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

CHAPTER 4

POLYTHETIC DIVISIVE CLUSTERING FOR INTERVAL-VALUED DATA BASED ON MINIMUM SPANNING TREES

In monothetic divisive algorithms, one variable at a time is used for the bipartition at each clustering stage. In contrast to monothetic divisive algorithms, all variables are used simultaneously in polythetic divisive algorithms for the bipartition at each clustering stage, which may lead to more comprehensive and more reliable clustering results.

In Section 4.1, a polythetic divisive algorithm based on minimum spanning trees is proposed for the divisive clustering of interval-valued data. In Section 4.2, the proposed polythetic algorithm is evaluated by a series of simulations. The simulation results demonstrate the effectiveness of the proposed polythetic algorithm. Real data applications are also conducted to evaluate this algorithm in Section 4.3.

4.1 A POLYTHETIC DIVISIVE ALGORITHM BASED ON MINIMUM SPANNING TREES

Given a weighted, connected and undirected graph, the minimum spanning tree (MST) of the graph is the subgraph which spans every vertex of the graph with the minimum total edge weight. The MST is an important tool in computer science and engineering, which is applied in the design of computer networks, cluster analysis, image segmentation, telecommunications networks and so forth. Two classical algorithms for finding the MST from a connected, undirected and weighted graph are Prim's algorithm (invented by Jarník (1930) and rediscovered by Prim (1957)) and Kruskal's algorithm (Kruskal (1956)).

One reason that MST can be applied in cluster analysis is that the MST gives us an underlying structure of the relationships between the objects without redundant connections.

Gowda and Ravi (1995a) developed a divisive hierarchical clustering algorithm for symbolic objects and the algorithm involves a step which is to find the MST based on the dissimilarity matrix of the symbolic data set. A more detailed introduction to this algorithm can be found in Section 2.3.1. Another step of this algorithm requires merging the symbolic objects within a group to form a new symbolic object, but how to merge the objects is not clearly explained in this algorithm.

In graph theory, one of the defining properties of trees is that removing an edge from a tree leads to two separate subtrees and each subtree contains one end of the edge. Thus, suppose T is a tree for which the vertices compose the set C and then we can divide C into two disjoint subsets by removing an edge from the corresponding tree T and using the vertices of the two subtrees as the two subsets of C . In our proposed divisive hierarchical clustering algorithm, we use the same idea to obtain one possible bipartition of a cluster of interval-valued objects by removing an edge from the corresponding tree for which the vertices compose the cluster. To be more specific, we remove each edge of the trees (subtrees) corresponding to the clusters of the interval-valued objects at each clustering stage and compare all the resulting bipartitions by the minimum total within-cluster variation criterion to find the “best” bipartition at each stage.

Let an interval-valued dataset Ω consist of n p -dimensional interval-valued objects $\{Y_u = ([a_{u1}, b_{u1}), \dots, [a_{up}, b_{up})], u = 1, \dots, n\}$. Let $P_r = (C_1, \dots, C_r), (r > 1)$, represent that Ω is partitioned into r clusters $C_1, \dots, C_r$ after the $(r - 1)^{th}$ stage of the divisive hierarchical clustering algorithm. The algorithm involves the following steps:

Step 1:

Stage $r = 1$: Start with $\Omega = C_1 = \{Y_1, \dots, Y_n\}$.

Step 2:

Obtain the distance matrix $\mathbf{D}$ for all the objects in Ω based on the selected distance measure.

Step 3:

Connect each pair of objects Y_{u_1} and Y_{u_2} in Ω by a weighted and undirected edge where the weight is the distance between Y_{u_1} and Y_{u_2} in the distance matrix $\mathbf{D}$, and thus obtain an undirected and edge-weighted graph G which connects all the interval-valued objects.

Step 4:

Find one MST of G by Prim's algorithm or Kruskal's algorithm.

Step 5:

Stage r : $\Omega = P_r = (C_1, \dots, C_r)$. At the same time, the objects in $C_v, v = 1, \dots, r$, are the vertices of a subtree of the MST obtained in Step 4 and here we use T_v to denote the subtree. Let n_v be the number of objects in C_v . Then, the number of edges in T_v is $(n_v - 1)$ and denote the set of edges of T_v as $\{e_v^i, i = 1, \dots, (n_v - 1)\}$. Removing an edge $e_v^i, i = 1, \dots, (n_v - 1)$, from T_v leads to two separate subtrees of T_v (the two subtrees are also subtrees of the MST obtained in Step 4) and denote them as $T_{v_1}^i$ and $T_{v_2}^i$. Suppose the vertices of $T_{v_1}^i$ and $T_{v_2}^i$ compose the sets $C_{v_1}^i$ and $C_{v_2}^i$, respectively. Then, the bipartition of C_v based on removing e_v^i from T_v is $(C_{v_1}^i, C_{v_2}^i)$. Next, put e_v^i back to T_v , remove the next edge from T_v and record the resulting bipartition. This process is repeated until all the $(n_v - 1)$ bipartitions which correspond to removing each of the $(n_v - 1)$ edges of T_v are recorded.

Step 6:

For each bipartition of each cluster C_v in $P_r, v = 1, \dots, r$, based on removing each edge $e_v^i, i = 1, \dots, (n_v - 1)$, from the corresponding T_v , calculate the change in the total within-cluster variation Δ_v by Eq.(2.17).

Step 7:

Select the cluster C_v to be partitioned and its bipartition $(C_{v_1}^i, C_{v_2}^i)$ by the maximum Δ_v calculated in Step 6.

Step 8:

Now, $r = r + 1$. Repeat Steps 5 – 7 until $r = n$ or stop this process by some stopping rules.

In this algorithm, we start from the MST with n vertices and $(n - 1)$ edges. We select one edge to be removed from the MST at Stage 1 and we obtain two subtrees with $(n - 2)$ edges in total. Then, we select one edge to be removed from the two subtrees at Stage 2 and we obtain three subtrees with $(n - 3)$ edges in total. Since one edge corresponds to one possible bipartition, there are $(n - r)$ possible bipartitions to select at Stage r . Therefore, the total number of possible bipartitions is $n(n - 1)/2$, which is the same as DIVWPCA (1 PC) and DIVCPCA (1 PC) developed in Section 3.1.3 are. In this sense, the MST in this algorithm plays the same role as the ordering criteria in the monothetic divisive algorithms, which is to significantly reduce the number of possible partitions. Another fact that needs to be noticed here is that the Prim's algorithm or the Kruskal's algorithm is only used once during the whole clustering process, which is to find the MST based on the distance matrix of the whole dataset in Step 4 of this algorithm. Furthermore, this algorithm is polythetic since the possible bipartitions at each stage are all based on the MST which connects all the interval-valued objects and all the variables in the dataset are used simultaneously in calculating the distance matrix and then finding the MST. Last but not least, this polythetic algorithm can not only be applied in the divisive clustering of interval-valued data but also in the divisive clustering of the other types of symbolic data.

4.2 SIMULATION STUDIES

In Section 4.1, we proposed a polythetic divisive algorithm for interval-valued data based on minimum spanning trees. In this section, we redo the first seven simulations which were described in Section 3.2 to evaluate the clustering accuracy of this polythetic algorithm. We

simulate two clusters of multi-dimensional interval-valued observations and in each dimension the intervals of different clusters of observations have dissimilar lengths and are well-separated in Simulation 1, overlapping in Simulation 2 and covering in Simulation 3. Besides, we simulate the well-separated case and the overlapping case for two clusters of interval-valued observations with similar lengths in each dimension in Simulation 4 and Simulation 5, respectively. In Simulation 6, we simulate two clusters of observations of which the intervals are overlapped in some dimensions and are covered by each other in some other dimensions. In Simulation 7, a dataset with four clusters of multi-dimensional interval-valued observations is simulated. In each dimension, the relationships between the intervals from different clusters include the well-separated case, the overlapping case and the covering case.

We use the same method which was described in Section 3.2 to simulate the datasets which are used in these seven simulations. To be more specific, we aggregate 100 classical samples $\{X_i = (x_{i1}, \dots, x_{ip}), i = 1, \dots, 100\}$ which are randomly drawn from a p -variate normal distribution $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ into one p -dimensional interval-valued observation $Y = ([a_1, b_1], \dots, [a_p, b_p])$ where $a_j = \min_i(x_{ij})$, $b_j = \max_i(x_{ij})$, $i = 1, \dots, 100$, $j = 1, \dots, p$. Likewise, we randomly draw another 100 classical samples points from $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and aggregate them into the second observation. We repeat this process to obtain the rest of the observations in the first cluster. The next cluster can be generated in the same way but with observations aggregated from samples drawn from another p -variate normal distribution $N_p(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$, and so on. Moreover, we use the same p -variate normal distributions as described in the first seven simulations of Section 3.2 to draw the classical samples.

The span normalized Euclidean Hausdorff distance and the Ichino-Yaguchi distance are applied in these seven simulations. The definitions of those two distance measures are given in Section 2.2.1.

The clustering accuracies of the polythetic algorithm in these seven simulations are given in Table 4.1. The performance of the polythetic algorithm in these seven simulation cases is close to the performance of the two mixed-strategy monothetic algorithms proposed in

Table 4.1: Clustering accuracies of the polythetic algorithm in the seven simulations.

Simulation	Span normalized Euclidean Hausdorff distance	Ichino-Yaguchi distance
Simulation 1	100.0%	100.0%
Simulation 2	100.0%	100.0%
Simulation 3	100.0%	100.0%
Simulation 4	100.0%	100.0%
Simulation 5	93.4%	95.4%
Simulation 6	100.0%	100.0%
Simulation 7	96.4%	96.8%

Section 3.1.5. In other words, the polythetic algorithm can obtain high clustering accuracies in all the seven simulations. The results of Simulation 3 also show that the possible limitation of Chavent’s algorithm shown in Figure 2.2 can be easily solved by this polythetic algorithm since all the variables are used simultaneously for the bipartition at each clustering stage in this algorithm. The results of Simulation 6 and Simulation 7 show that the polythetic algorithm can handle complex and mixed cases.

4.3 APPLICATIONS

In this section, the polythetic divisive algorithm based on minimum spanning trees is applied to two real-world datasets to further demonstrate its effectiveness. The two datasets are the Iris dataset from Fisher (1936) and the face recognition dataset from Leroy et al. (1996).

4.3.1 THE IRIS DATASET

The first dataset to which we apply our proposed polythetic algorithm is the well-known Iris dataset from Fisher (1936). This well-known dataset is frequently used in the research of pattern recognition. We take this dataset as an example to illustrate the clustering procedure

Table 4.2: Table of the interval-valued Iris dataset.

ID	Species	Sepal Length	Sepal Width	Petal Length	Petal Width
1	Iris-setosa	[4.4, 5.4]	[2.9, 3.9]	[1.3, 1.7]	[0.1, 0.4]
2	Iris-setosa	[4.3, 5.8]	[3.0, 4.4]	[1.1, 1.7]	[0.1, 0.4]
3	Iris-setosa	[4.6, 5.4]	[3.0, 3.7]	[1.0, 1.9]	[0.2, 0.5]
4	Iris-setosa	[4.4, 5.5]	[3.0, 4.2]	[1.2, 1.6]	[0.1, 0.4]
5	Iris-setosa	[4.4, 5.3]	[2.3, 3.8]	[1.3, 1.9]	[0.2, 0.6]
6	Iris-versicolor	[4.9, 7.0]	[2.3, 3.3]	[3.3, 4.9]	[1.0, 1.6]
7	Iris-versicolor	[5.0, 6.7]	[2.0, 3.1]	[3.5, 4.7]	[1.0, 1.5]
8	Iris-versicolor	[5.7, 6.8]	[2.5, 3.2]	[3.5, 5.0]	[1.0, 1.8]
9	Iris-versicolor	[5.4, 6.7]	[2.3, 3.4]	[3.7, 5.1]	[1.0, 1.6]
10	Iris-versicolor	[5.0, 6.2]	[2.3, 3.0]	[3.0, 4.6]	[1.0, 1.4]
11	Iris-virginica	[4.9, 7.6]	[2.5, 3.6]	[4.5, 6.6]	[1.7, 2.5]
12	Iris-virginica	[5.7, 7.7]	[2.2, 3.8]	[5.0, 6.9]	[1.5, 2.4]
13	Iris-virginica	[5.6, 7.7]	[2.7, 3.3]	[4.8, 6.7]	[1.6, 2.3]
14	Iris-virginica	[6.0, 7.9]	[2.6, 3.8]	[4.8, 6.4]	[1.4, 2.4]
15	Iris-virginica	[5.8, 6.9]	[2.5, 3.4]	[5.0, 5.9]	[1.8, 2.5]

of the polythetic algorithm. This dataset consist of 150 flowers of three species: Iris-setosa, Iris-versicolor and Iris-virginica. Each species contains 50 instances. Four variables are measured (in cm) for each instance, which are sepal length, sepal width, petal length and petal width.

To obtain an interval-valued Iris dataset, we aggregate each set of 10 consecutive instances $\{X_i = (x_{i1}, \dots, x_{i4}), i = 1, \dots, 10\}$ of the Iris dataset into one single interval-valued observation $Y = ([a_1, b_1], \dots, [a_4, b_4])$ where $a_j = \min_i(x_{ij})$, $b_j = \max_i(x_{ij})$, $i = 1, \dots, 10$, $j = 1, \dots, 4$. Thus, the resulting dataset is an interval-valued dataset with five observations from each of the three species. We use an ID number to label each of the fifteen observations and the resulting interval-valued dataset is given in Table 4.2.

In this application, we apply two distance measures, i.e., the span normalized Euclidean Hausdorff distance and the Ichino-Yaguchi distance, to calculate the distance matrices for

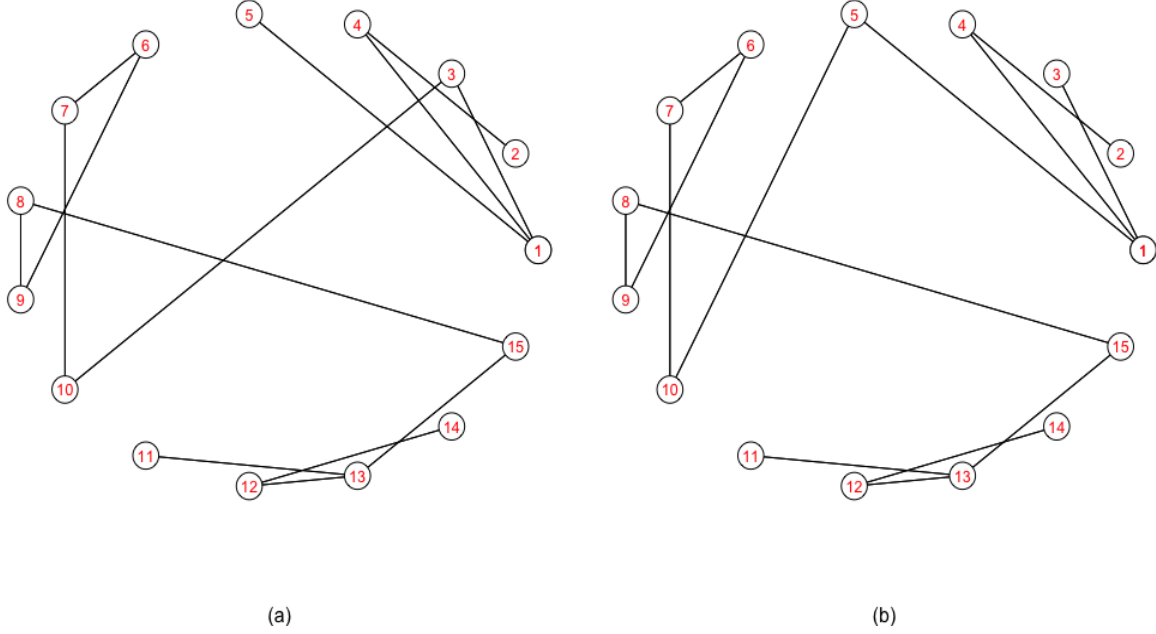


Figure 4.1: Minimum spanning trees for the interval-valued Iris dataset: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

the fifteen observations and then obtain the two MSTs based on these two distance matrices, respectively. Figure 4.1 shows the MSTs based on the two distance matrices. The numbers in the nodes of the two MSTs represent the ID numbers of the observations.

After obtaining the MSTs, we can apply the minimum total within-cluster variation criterion to find the best bipartition at each clustering stage. Suppose the span normalized Euclidean Hausdorff distance is used and we start the clustering from the MST given in Figure 4.1(a). The MST has fourteen edges and thus we have fourteen possible bipartitions to compare at the first clustering stage. The possible bipartitions and the corresponding Δ_v 's are shown in Table 4.3. The maximum Δ_v is corresponding to the bipartition $(\{1, 2, 3, 4, 5\}, \{6, 7, 8, 9, 10, 11, 12, 13, 14, 15\})$. Thus, we choose this bipartition as the divisive clustering result of the first stage. This bipartition clearly divides the first five observations from the Iris-setosa species into one cluster and is achieved by removing the edge between Observation

Table 4.3: Removed edges, possible bipartitions and the corresponding Δ_v 's at the first clustering stage of the polythetic algorithm based on the span normalized Euclidean Hausdorff distance.

Removed Edge	Possible Bipartition	Δ_v
$4 \longleftrightarrow 2$	$(\{2\}, \{1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\})$	0.6191
$1 \longleftrightarrow 3$	$(\{1, 2, 4, 5\}, \{3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\})$	2.7972
$1 \longleftrightarrow 4$	$(\{2, 4\}, \{1, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\})$	1.3252
$1 \longleftrightarrow 5$	$(\{5\}, \{1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\})$	0.4829
$7 \longleftrightarrow 6$	$(\{1, 2, 3, 4, 5, 7, 10\}, \{6, 8, 9, 11, 12, 13, 14, 15\})$	3.1839
$10 \longleftrightarrow 7$	$(\{1, 2, 3, 4, 5, 10\}, \{6, 7, 8, 9, 11, 12, 13, 14, 15\})$	3.4595
$9 \longleftrightarrow 8$	$(\{1, 2, 3, 4, 5, 6, 7, 9, 10\}, \{8, 11, 12, 13, 14, 15\})$	2.8045
$6 \longleftrightarrow 9$	$(\{1, 2, 3, 4, 5, 6, 7, 10\}, \{8, 9, 11, 12, 13, 14, 15\})$	2.9344
$3 \longleftrightarrow 10$	$(\{1, 2, 3, 4, 5\}, \{6, 7, 8, 9, 10, 11, 12, 13, 14, 15\})$	3.7447
$13 \longleftrightarrow 11$	$(\{11\}, \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15\})$	0.4401
$13 \longleftrightarrow 12$	$(\{12, 14\}, \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15\})$	0.9744
$15 \longleftrightarrow 13$	$(\{11, 12, 13, 14\}, \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15\})$	2.1285
$12 \longleftrightarrow 14$	$(\{14\}, \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15\})$	0.4425
$8 \longleftrightarrow 15$	$(\{11, 12, 13, 14, 15\}, \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\})$	2.6772

3 and Observation 10. Now, the observations are divided into two clusters. At the same time, the MST given in Figure 4.1(a) is divided into two subtrees corresponding to the two clusters by removing the edge between Observation 3 and Observation 10. The first subtree connects the first five observations and the second subtree connects the rest of the observations.

At the second clustering stage, we continue to bipartition one of the two clusters obtained at the first stage by removing one edge from one of the two subtrees. There are thirteen edges left in these two subtrees and thus we have thirteen possible bipartitions to compare at the second stage. The possible bipartitions and the corresponding Δ_v 's are given in Table 4.4. The maximum value of Δ_v is 0.7192, which is corresponding to the partition $(\{1, 2, 3, 4, 5\}, \{6, 7, 8, 9, 10\}, \{11, 12, 13, 14, 15\})$. This partition clearly divides the fifteen observations of the three species into three clusters, each cluster with one species. Besides, the edge to be removed at this stage is the edge between Observation 8 and Observation 15. Now, the MST given in Figure 4.1(a) is divided into three subtrees corresponding to the

Table 4.4: Removed edges, possible bipartitions and the corresponding Δ_v 's at the second clustering stage of the polythetic algorithm based on the span normalized Euclidean Hausdorff distance.

Removed Edge	Possible Bipartition	Δ_v
$4 \longleftrightarrow 2$	$(\{2\}, \{1, 3, 4, 5\}, \{6, 7, 8, 9, 10, 11, 12, 13, 14, 15\})$	0.0396
$1 \longleftrightarrow 3$	$(\{3\}, \{1, 2, 4, 5\}, \{6, 7, 8, 9, 10, 11, 12, 13, 14, 15\})$	0.0331
$1 \longleftrightarrow 4$	$(\{2, 4\}, \{1, 3, 5\}, \{6, 7, 8, 9, 10, 11, 12, 13, 14, 15\})$	0.0590
$1 \longleftrightarrow 5$	$(\{5\}, \{1, 2, 3, 4\}, \{6, 7, 8, 9, 10, 11, 12, 13, 14, 15\})$	0.0621
$7 \longleftrightarrow 6$	$(\{1, 2, 3, 4, 5\}, \{7, 10\}, \{6, 8, 9, 11, 12, 13, 14, 15\})$	0.3554
$10 \longleftrightarrow 7$	$(\{1, 2, 3, 4, 5\}, \{10\}, \{6, 7, 8, 9, 11, 12, 13, 14, 15\})$	0.2101
$9 \longleftrightarrow 8$	$(\{1, 2, 3, 4, 5\}, \{6, 7, 9, 10\}, \{8, 11, 12, 13, 14, 15\})$	0.5869
$6 \longleftrightarrow 9$	$(\{1, 2, 3, 4, 5\}, \{6, 7, 10\}, \{8, 9, 11, 12, 13, 14, 15\})$	0.4590
$13 \longleftrightarrow 11$	$(\{1, 2, 3, 4, 5\}, \{11\}, \{6, 7, 8, 9, 10, 12, 13, 14, 15\})$	0.1166
$13 \longleftrightarrow 12$	$(\{1, 2, 3, 4, 5\}, \{12, 14\}, \{6, 7, 8, 9, 10, 11, 13, 15\})$	0.2909
$15 \longleftrightarrow 13$	$(\{1, 2, 3, 4, 5\}, \{6, 7, 8, 9, 10, 15\}, \{11, 12, 13, 14\})$	0.5656
$12 \longleftrightarrow 14$	$(\{1, 2, 3, 4, 5\}, \{14\}, \{6, 7, 8, 9, 10, 11, 12, 13, 15\})$	0.1420
$8 \longleftrightarrow 15$	$(\{1, 2, 3, 4, 5\}, \{6, 7, 8, 9, 10\}, \{11, 12, 13, 14, 15\})$	0.7192

three clusters by first removing the edge between Observation 3 and Observation 10 and then removing the edge between Observation 8 and Observation 15.

At the remaining stages, we repeat this process to continue the divisive clustering until the observations are divided into fifteen clusters with one observation in each cluster. The dendrogram for the whole process is given in Figure 4.2(a). Moreover, if we apply the Ichino-Yaguchi distance in this algorithm, we achieve the dendrogram shown in Figure 4.2(b).

After the divisive clustering, we select the result of one clustering stage as the final clustering result of each dendrogram by comparing the Dunn index values, DI_r^s , or the Davis-Bouldin index values, DB_r^s , of all the clustering stages of each dendrogram. These two validity indices were extended by Kim (2009) and Kim and Billard (2011) from classical data to symbolic data. The definitions are given in Eq.(2.18) and Eq.(2.19), respectively. The validity indices of all the clustering stages of the two dendrograms given in Figure 4.2(a) and Figure 4.2(b) are shown in Figure 4.3(a) and Figure 4.3(b), respec-

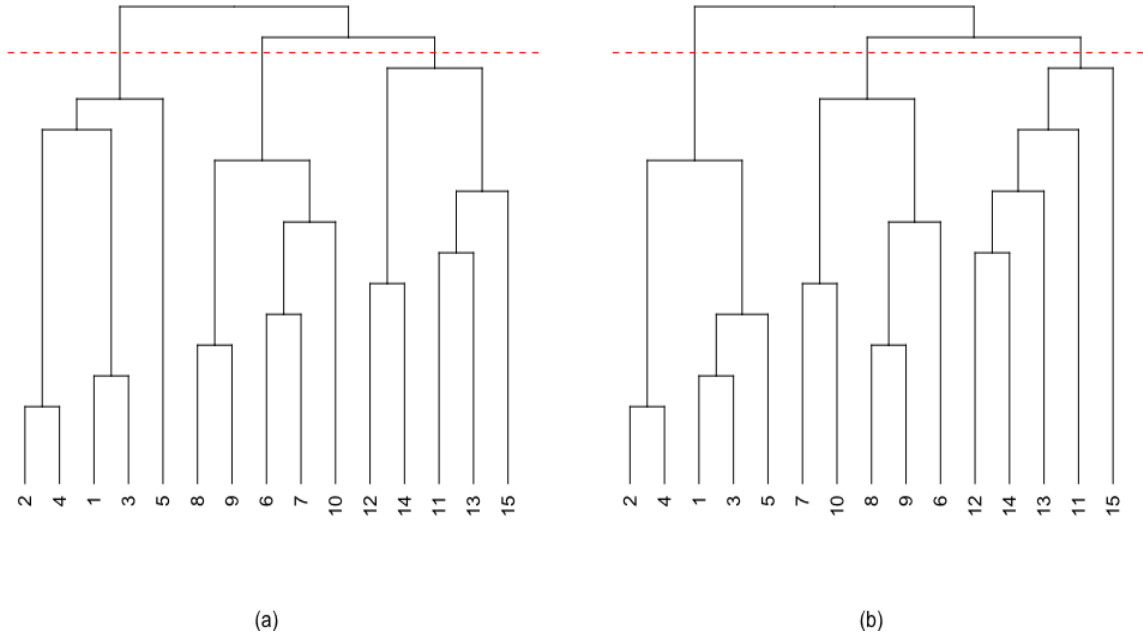


Figure 4.2: Dendrograms for the interval-valued Iris dataset using the polythetic algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

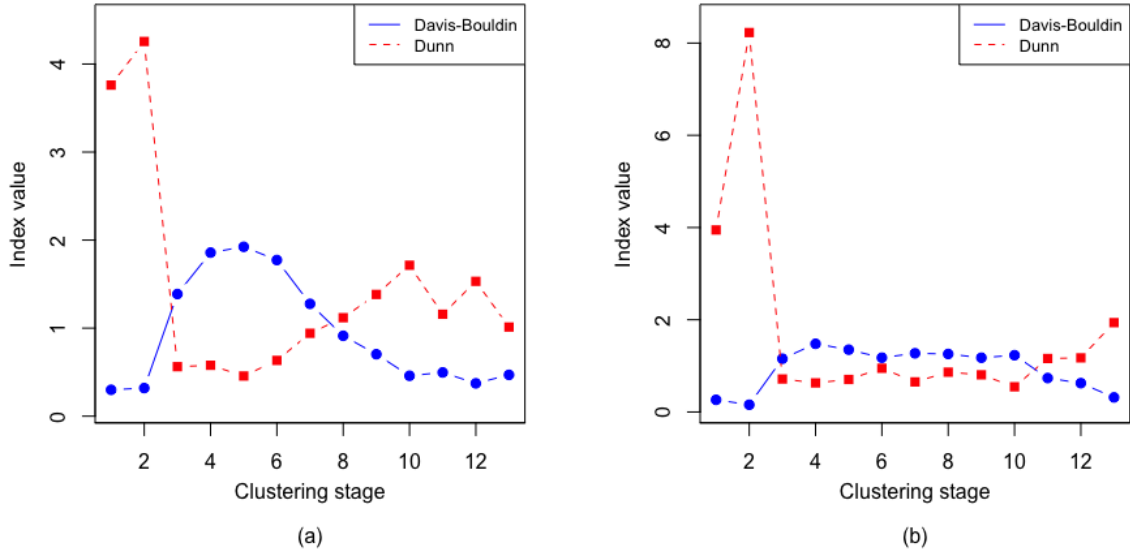


Figure 4.3: Validity indices for the clustering result of the interval-valued Iris dataset using the polythetic algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

tively. Since a larger DI_r^s or a smaller DB_r^s usually means a better clustering outcome, it is quite clear that we achieve the best clustering outcome at Stage 2 of each of these two dendrograms and thus we select the result of this stage as the final clustering result of each dendrogram. Based on Figure 4.2, the results of Stage 2 of the two dendrograms are both $(\{1, 2, 3, 4, 5\}, \{6, 7, 8, 9, 10\}, \{11, 12, 13, 14, 15\})$, which correctly divide the observations from the three different species into three different clusters. Therefore, our proposed polythetic algorithm works well in this application.

4.3.2 THE FACE RECOGNITION DATASET

The second dataset is the face recognition dataset collected by Leroy et al. (1996), see also Douzal-Chouakria et al. (2011). This dataset consists of twenty-seven observations and six variables. More details about this dataset can be found in Section 3.3.2 and the dataset is given in Table 3.25.

We use the ID numbers of the twenty-seven observations to represent them. Then, we can plot the observations on a circle to clearly show the MSTs of the observations. Figure 4.4 shows the MSTs of the observations. For any two observations, the weight of the edge between them is the distance between them. We apply two distance measures, i.e., the span normalized Euclidean Hausdorff distance and the Ichino-Yaguchi distance, to obtain the distance matrices and then obtain the two MSTs based on the two distance matrices, respectively.

The dendrograms for the face recognition dataset using the polythetic algorithm based on the span normalized Euclidean Hausdorff distance and the Ichino-Yaguchi distance are given in Figure 4.5(a) and Figure 4.5(b), respectively. The validity indices for the two dendrograms are given in Figure 4.6(a) and Figure 4.6(b), respectively. Since a larger DI_r^s or a smaller DB_r^s usually means a better clustering outcome, we select Stage 8 of each dendrogram as the final clustering result of each dendrogram by Figure 4.6(a) and Figure 4.6(b). The clustering results of Stage 8 of the two dendrograms shown in Figure 4.5 are both

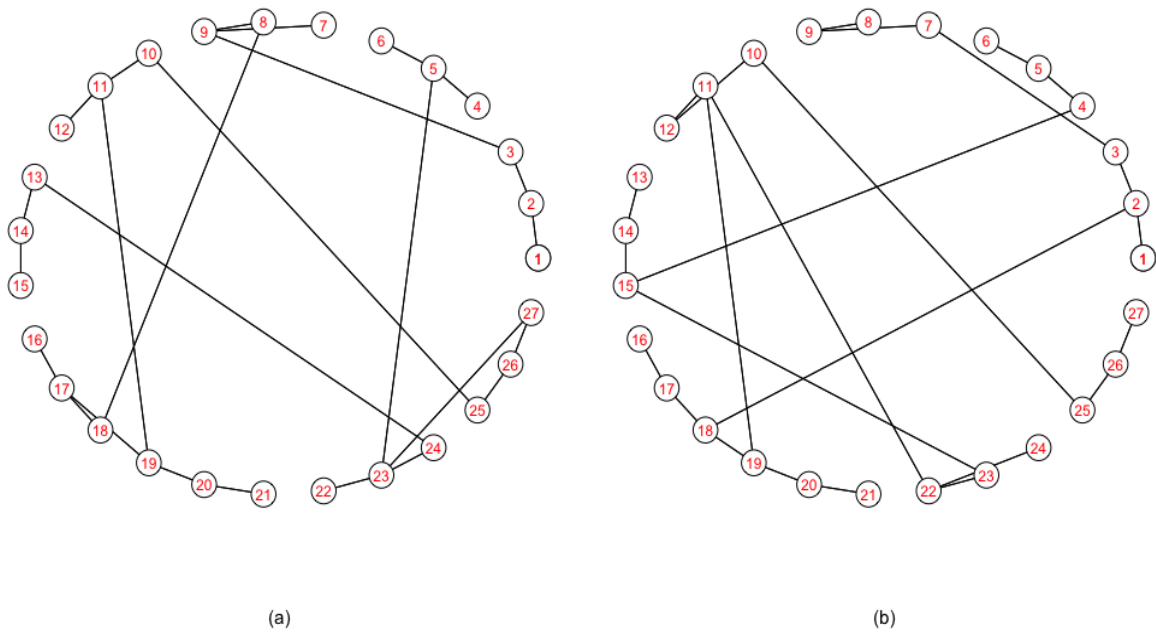


Figure 4.4: Minimum spanning trees for the face recognition dataset: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

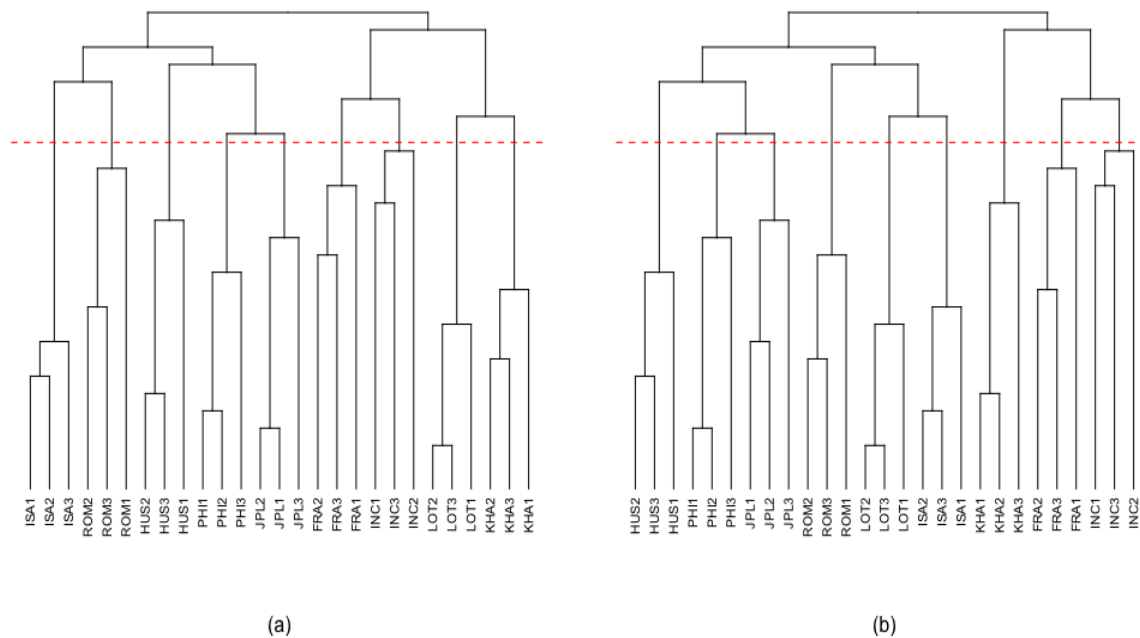


Figure 4.5: Dendrograms for the face recognition dataset using the polythetic algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

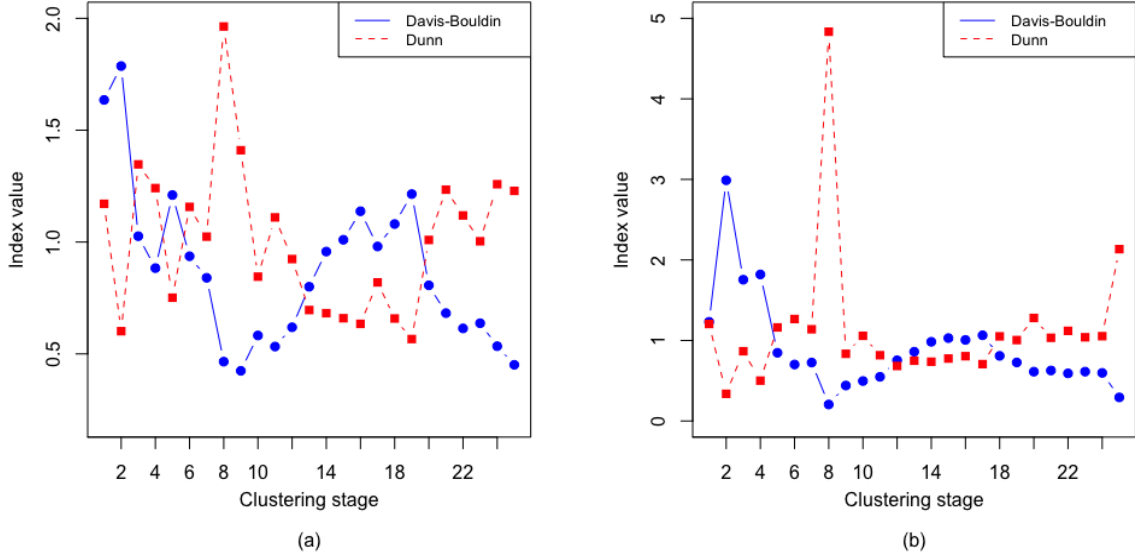


Figure 4.6: Validity indices for the clustering result of the face recognition dataset using the polythetic algorithm: (a) Based on the span normalized Euclidean Hausdorff distance, (b) based on the Ichino-Yaguchi distance.

($\{FRA1, FRA2, FRA3\}, \{HUS1, HUS2, HUS3\}, \{INC1, INC2, INC3\}, \{ISA1, ISA2, ISA3\}, \{JPL1, JPL2, JPL3\}, \{KHA1, KHA2, KHA3\}, \{LOT1, LOT2, LOT3\}, \{PHI1, PHI2, PHI3\}, \{ROM1, ROM2, ROM3\}$). Thus, our proposed polythetic algorithm correctly divides the twenty-seven observations into nine clusters, one for each subject, and each cluster contains the three observations from the same subject.

By these two real data applications, the effectiveness of the polythetic divisive algorithm is further demonstrated.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this dissertation, we propose three basic divisive clustering algorithms and two mixed-strategy algorithms combining these three basic algorithms for interval-valued data. All these five algorithms are monothetic algorithms. We design a series of simulations to compare these algorithms with an existing monothetic divisive algorithm proposed by Chavent (1998, 2000), and show that the two mixed-strategy algorithms outperform all the other monothetic algorithms in these simulations. The effectiveness of these two mixed-strategy algorithms is also demonstrated through real-world data applications.

Furthermore, we propose a polythetic divisive clustering algorithm for interval-valued data. The effectiveness of this polythetic algorithm is also validated through several simulation studies and real data applications. Besides, this polythetic algorithm can also be directly applied to the divisive clustering of the other types of symbolic data.

We divide all the monothetic algorithms into several types which are summarized in Table 3.1. Some of our proposed monothetic algorithms are endpoints-based algorithms or PCA-based algorithms while the two mixed-strategy algorithms are both endpoints-based algorithms and PCA-based algorithms. The endpoints-based algorithms are proposed to solve the possible limitation of Chavent's algorithm shown in Figure 2.2. The PCA-based algorithms are proposed because of a property of monothetic divisive algorithms, i.e., monothetic divisive algorithms only choose one variable at a time to find the bipartition at each clustering stage. The advantage is that we can set a binary question at each partition node of the dendrogram based on the chosen variable to clearly explain the bipartition criterion at each node. The disadvantage is that monothetic divisive algorithms may perform poorly

when the correct bipartition at each node relies on more than one variable. Our solution to this problem is to use a single synthetic variable which combines all the original variables to set the binary question at each partition node and this synthetic variable is obtained by symbolic PCA methodologies. This idea leads to our PCA-based algorithms which not only take all the variables into account to decide the correct bipartition at each node but also use a single (synthetic) variable to set the binary question at each node. Besides, we design a series of simulations to compare the performance of the algorithms in the simulation studies. In fact, this series of simulations can also be used to comprehensively evaluate the performance of any other clustering algorithm for interval-valued data.

In this dissertation, we combine the symbolic PCA methodologies with the monothetic divisive clustering algorithms for interval-valued data and thus obtain our PCA-based algorithms. We choose two symbolic PCA methodologies to be combined with our monothetic algorithms. One is the symbolic covariance PCA method proposed by Le-Rademacher (2008) and Le-Rademacher and Billard (2012), and the other one is a weighted symbolic PCA method which is proposed in this dissertation. Both of them work well when they are applied in our algorithms. In fact, any other well-designed symbolic PCA method for interval-valued data can be combined with our monothetic algorithms and this is one proposal for future work. On the other hand, these PCA-based algorithms can be extended to the other types of symbolic data as long as the symbolic PCA methods for the other types of symbolic data are proposed and proved to be effective. Then, we can propose some monothetic algorithms for the divisive clustering of the other types of symbolic data based on these symbolic PCA methods. A lot of future work can be generated from this direction.

Another potential problem is how to decide the number of principal components used as the ordering criteria in the PCA-based algorithms. The simulations results of Section 3.2 show that a larger number of principal components used in each PCA-based algorithm sometimes leads to a higher accuracy. On the other hand, a larger number of principal components means a larger number of possible bipartitions to be considered at each clustering stage of

each algorithm, which decreases the efficiency of each algorithm. In this dissertation, we use a given cumulative contribution rate λ of the principal components to decide how many principal components to keep. Our suggestion is to set λ to a moderate value (we set $\lambda = 0.95$ and $\lambda = 0.9$, respectively, in the two applications of Section 3.3) so that at least one-third of all the principal components obtained by the symbolic PCA are used as the ordering criteria in the PCA-based algorithms to ensure the accuracy at first and then to improve the efficiency.

Last but not least, any interval-valued variable can be seen as a two-dimensional variable $(m, l) \subseteq R \times R^+$, where m is the midpoint value and l is the length of the interval. It can also be seen as another two-dimensional variable $[a, b]$, where a and b are the endpoints values of the interval. Any histogram-valued variable consists of a number of three-dimensional variables since a histogram is represented by a sequence of intervals and corresponding frequencies. Therefore, the actual dimension of a symbolic dataset is sometimes not small. In this dissertation, we use symbolic PCA as a dimension-reduction technique to give ordering criteria to the interval-valued observations. Instead of PCA, some other dimension-reduction techniques such as factor analysis and linear discriminant analysis can also be extended to symbolic data and be applied in the divisive clustering algorithms for symbolic data. Some future work may arise from this idea.

BIBLIOGRAPHY

- [1] Agahian, F., Amirshahi, S. A., and Amirshahi, S. H. (2008). Reconstruction of reflectance spectra using weighted principal component analysis. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, 33(5), 360–371.
- [2] Bertrand, P. and Goupil, F. (2000). Descriptive statistics for symbolic data. In: *Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data*, ed. H.H. Bock, and E. Diday, Springer, 106–124.
- [3] Billard, L. (2006). Symbolic data analysis: what is it? In: *Compstat 2006-Proceedings in Computational Statistics*, Physica-Verlag HD, 261–269.
- [4] Billard, L. (2007). Dependencies and variation components of symbolic interval-valued data. In: *Selected Contributions in Data Analysis and Classification*, Springer, 3–12.
- [5] Billard, L. (2008). Sample covariance functions for complex quantitative data. In: *Proceedings of World IASC Conference, Yokohama, Japan*, 157–163.
- [6] Billard, L. and Diday, E. (2003). Symbolic data analysis: Definitions and examples. Technical Report, 62 pages.
- [7] Billard, L. and Diday, E. (2006). *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Wiley, Chichester.

- [8] Billard, L., Douzal-Chouakria, A., and Diday, E. (2011). Symbolic principal components for interval-valued observations. *Statistical Analysis and Data Mining*, 4 (2), 229-246.
- [9] Brito, P. and Chavent, M. (2012). Divisive monothetic clustering for interval and histogram-valued data. In: *ICPRAM 2012-1st International Conference on Pattern Recognition Applications and Methods*, 229–234.
- [10] Cazes, P., Chouakria, A., Diday, E., and Schektman, Y. (1997). Extension de l’analyse en composantes principales à des données de type intervalle. *Revue de Statistique Appliquée*, 45(3), 5–24.
- [11] Chavent, M. (1998). A monothetic clustering method. *Pattern Recognition Letters*, 19(11), 989–996.
- [12] Chavent, M. (2000). Criterion-based divisive clustering for symbolic data. In: *Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data*, ed. H.H. Bock, and E. Diday, Springer, 299–311.
- [13] Chen, M., Wang, H., and Qin, Z. (2015). Principal component analysis for probabilistic symbolic data: a more generic and accurate algorithm. *Advances in Data Analysis and Classification*, 9(1), 59–79.
- [14] Chen, Y. (2014). *Symbolic Data Regression and Clustering Methods*. PhD dissertation, University of Georgia.
- [15] Chouakria, A., Cazes, P., and Diday, E. (2000). Symbolic principal component analysis. In: *Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data*, ed. H.H. Bock, and E. Diday, Springer, 200–212.
- [16] Da Costa, J. F. P., Alonso, H., and Roque, L. (2009). A weighted principal component analysis and its application to gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(1), 246–252.

- [17] Davis, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2), 224–227.
- [18] de Carvalho, F. A. T. (1994). Proximity coefficients between Boolean symbolic objects. In: *New Approaches in Classification and Data Analysis*, Springer, Berlin, Heidelberg, 387–394.
- [19] de Carvalho, F. A. T. (1998). Extension based proximities between constrained boolean symbolic objects. In: *Data Science, Classification, and Related Methods*, Springer, Tokyo, 370–378.
- [20] Delchambre, L. (2014). Weighted principal component analysis: a weighted covariance eigendecomposition approach. *Monthly Notices of the Royal Astronomical Society*, 446(4), 3545–3555.
- [21] Diday, E. (1989). Introduction à l’approche symbolique en analyse des données. *RAIRO-Operations Research-Recherche Opérationnelle*, 23(2), 193–236.
- [22] Douzal-Chouakria, A., Billard, L., and Diday, E. (2011). Principal component analysis for interval-valued observations. *Statistical Analysis and Data Mining*, 4(2), 229–246.
- [23] Dunn, J. C. (1974). Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1), 95–104.
- [24] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2), 179–188.
- [25] Gioia, F. and Lauro, C. N. (2006). Principal component analysis on interval data. *Computational Statistics*, 21(2), 343–363.
- [26] Gowda, K. C. and Diday, E. (1991a). Symbolic clustering using a new dissimilarity measure. *Pattern Recognition*, 24(6), 567–578.

- [27] Gowda, K. C. and Diday, E. (1991b). Unsupervised learning through symbolic clustering. *Pattern Recognition Letters*, 12(5), 259–264.
- [28] Gowda, K. C. and Ravi, T. V. (1995a). Divisive clustering of symbolic objects using the concepts of both similarity and dissimilarity. *Pattern Recognition*, 28(8), 1277–1282.
- [29] Gowda, K. C. and Ravi, T. V. (1995b). Agglomerative clustering of symbolic objects using the concepts of both similarity and dissimilarity. *Pattern Recognition Letters*, 16(6), 647–652.
- [30] Guru, D. S. and Kiranagi, B. B. (2005). Multivalued type dissimilarity measure and concept of mutual dissimilarity value for clustering symbolic patterns. *Pattern Recognition*, 38(1), 151–156.
- [31] Guru, D. S., Kiranagi, B. B., and Nagabhushan, P. (2004). Multivalued type proximity measure and concept of mutual similarity value useful for clustering symbolic patterns. *Pattern Recognition Letters*, 25(10), 1203–1213.
- [32] Hausdorff, F. (1957). *Set Theory* (transl. J.R. Aumann). Chelsea, New York.
- [33] Ichino, M. (2001). The quantile method for symbolic principal component analysis. *Statistical Analysis and Data Mining*, 4(2), 184–198.
- [34] Ichino, M. and Yaguchi, H. (1994). Generalized Minkowski metrics for mixed feature-type data analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4), 698–708.
- [35] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3), 264–323.
- [36] Jarník, V. (1930). O jistém problému minimálním. *Práce Moravské Přírodovědecké Společnosti*, 6, 57–63.

- [37] Kim, J. (2009). *Dissimilarity Measures for Histogram-valued Data and Divisive Clustering of Symbolic Objects*. PhD dissertation, University of Georgia.
- [38] Kim, J. and Billard, L. (2011). A polythetic clustering process and cluster validity indexes for histogram-valued objects. *Computational Statistics & Data Analysis*, 55(7), 2250–2262.
- [39] Kim, J. and Billard, L. (2012). Dissimilarity measures and divisive clustering for symbolic multimodal-valued data. *Computational Statistics & Data Analysis*, 56(9), 2795–2808.
- [40] Kim, J. and Billard, L. (2018). Double monothetic clustering for histogram-valued data. *Communications for Statistical Applications and Methods*, 25(3), 263–274.
- [41] Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1), 48–50.
- [42] Lauro, C. N. and Palumbo, F. (2000). Principal component analysis of interval data: a symbolic data analysis approach. *Computational Statistics*, 15(1), 73–87.
- [43] Le-Rademacher, J. G. (2008). *Principal Component Analysis for Interval-valued and Histogram-valued Data and Likelihood Functions and Some Maximum Likelihood Estimators for Symbolic Data*. PhD dissertation, University of Georgia.
- [44] Le-Rademacher, J. G. and Billard, L. (2012). Symbolic covariance principal component analysis and visualization for interval-valued data. *Journal of Computational and Graphical Statistics*, 21(2), 413–432.
- [45] Le-Rademacher, J. G. and Billard, L. (2013). Principal component histograms from interval-valued observations. *Computational Statistics*, 28(5), 2117–2138.

- [46] Leroy, B., Chouakria, A., Herlin, I., and Diday, E. (1996). Approche géométrique et classification pour la reconnaissance de visage. In *Congrès de Reconnaissance des Formes et Intelligence Artificielle*, 548–557.
- [47] Palumbo, F. and Lauro, C. N. (2003). A PCA for interval-valued data based on mid-points and radii. In: *New Developments in Psychometrics*, Springer, Tokyo, 641–648.
- [48] Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11), 559–572.
- [49] Pettie, S. and Ramachandran, V. (2002). An optimal minimum spanning tree algorithm. *Journal of the ACM (JACM)*, 49(1), 16–34.
- [50] Pirçon, J. Y. (2004). *Le Clustering et les Processus de Poisson pour de Nouvelles Méthodes Monothétiques*. PhD dissertation, FUNDP-University of Namur, Namur, Belgium.
- [51] Prim, R. C. (1957). Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6), 1389–1401.
- [52] Rasson, J. P., Pirçon, J. Y., Lallemand, P., and Adans, S. (2008). Unsupervised divisive classification. *Symbolic Data Analysis and the Sodas Software*, 149–156.
- [53] Ravi, T. V. and Gowda, K. C. (1999). Clustering of symbolic objects using gravitational approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(6), 888–894.
- [54] Tamuz, O., Mazeh, T., and Zucker, S. (2005). Correcting systematic effects in a large set of photometric light curves. *Monthly Notices of the Royal Astronomical Society*, 356(4), 1466–1470.

- [55] Wang, H., Guan, R., and Wu, J. (2012). CIPCA: Complete-Information-based Principal Component Analysis for interval-valued data. *Neurocomputing*, 86, 158–169.
- [56] Yue, H. H. and Tomoyasu, M. (2004). Weighted principal component analysis and its applications to improve FDC performance. In *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, volume 4, 4262–4267. IEEE.

APPENDIX

R CODE

```
#####
#
#   Divisive Hierarchical Clustering for Interval-valued Data   #
#                               By Jiankun Zhu                   #
#                                                                #
#####

#####
#                               R Code for DIVCPCA and DIVWPCA   #
#####

PCAdiv.Interval<-function(distance="SNEH",pcamethod="weightedPCA",cp=0.9,data)
# The DIVCPCA algorithm based on the symbolic covariance PCA proposed
# by Le-Rademacher (2008) and Le-Rademacher and Billard (2012), and the
# DIVWPCA algorithm based on our proposed weighted symbolic PCA method.
# distance = "SNEH" (the span normalized Euclidean Hausdorff distance)
# or "IY" (the Ichino-Yaguchi distance).
# pcamethod = "symb_covPCA" (the symbolic covariance PCA method)
# or "weightedPCA" (the weighted symbolic PCA method).
# cp = a given cumulative contribution rate of principal components
# data = an interval-valued data matrix

{ nob=nrow(data) # the number of the observations
  nvar=ncol(data)/2 # the number of the variables

  # the weighted symbolic PCA method
  wPCA.Interval=function(data,cp)
  {
    # obtain the weights
    w=matrix(rep(0,nob*nvar),nrow=nob,ncol=nvar,byrow=T)
    for (col in 1:nvar){
      a=sort(unique (c(data[,2*col-1],data[,2*col])))
      # define the subintervals, duplicate elements removed.
      t=b=rep(0,length(a)-1)
      # b denotes the frequency of each subinterval.
```

```

# t indicates whether the interval contains each subinterval.
for(i in 1:(length(a)-1))
  {b[i]=sum((data[,2*col-1]<=a[i])&(data[,2*col]>=a[i+1]))}
for(j in 1:nob){
  for(i in 1:(length(a)-1)) {
    t[i]=(data[j,2*col-1]<=a[i])&(data[j,2*col]>=a[i+1])}
    for(i in 1:(length(a)-1)) {
      w[j,col]=t[i]*b[i]*(a[i+1]-a[i])+w[j,col]
    }
  }
}
weight=scale(w, center = FALSE, scale = TRUE)
weight=rowSums(weight)
weight=weight/sum(weight)
weight=diag(weight)

Xc=matrix(rep(0,nob*nvar),nrow=nob,ncol=nvar,byrow=T)
# Xc is the matrix of centers
for(i in 1:nvar){
Xc[,i]=(data[,2*i-1]+data[,2*i])/2
}

Xc=scale(Xc, center = TRUE, scale = FALSE)
# scale = TRUE if the unit variance scaling is needed.
pc=eigen(cov(weight%*%Xc))
pcvalue=weight%*%Xc%*%pc$vectors
numpc=sum(cumsum(pc$values)<cp*sum(pc$values))+1
# the smallest number of PCs such that the cumulative contribution
# rate is larger than cp.
return(as.matrix(pcvalue[,1:numpc]))
} # wPCA.Interval ends

if(pcamethod=="weightedPCA") {pcavalues=wPCA.Interval(data=data,cp=cp)}

# the symbolic covariance PCA method
symb_covPCA.Interval=function(data,cp){
  p=nvar
  m=nob
  cov=cor=matrix(0, p, p)
  x_mean=rep(0, p)

  # the variable means
  for (i in 1:p){
    x_mean[i]=(mean(data[,2*i-1])+mean(data[,2*i]))/2}

  # the covariance matrix
  for (k in 1:p){

```

```

    for (l in 1:p){
      q = 0
      for (h in 1:m){
        q = q + 2*(data[h,2*k-1]-x_mean[k])*(data[h,2*1-1]-x_mean[1])
        + (data[h,2*k-1]-x_mean[k])*(data[h,2*1]-x_mean[1])
        + (data[h,2*k]-x_mean[k])*(data[h,2*1-1]-x_mean[1])
        + 2*(data[h,2*k]-x_mean[k])*(data[h,2*1]-x_mean[1])
      }
      cov[k,l] = q/6/m
    }
  }

# the correlation matrix
for (k in 1:p){
  for (l in 1:p){
    cor[k,l]=cov[k,l]/sqrt(cov[k,k])/sqrt(cov[l,l])
  }
}

Xc=matrix(rep(0,nob*nvar),nrow=nob,ncol=nvar,byrow=T)
for(i in 1:nvar){
  Xc[,i]=(data[,2*i-1]+data[,2*i])/2
}
Xc=scale(Xc, center = TRUE, scale = FALSE)
# scale = TRUE if the unit variance scaling is needed.
pc=eigen(cov)
# pc=eigen(cor) if the unit variance scaling is performed on Xc.
pcvalue=Xc%*%pc$vectors
numpc=sum(cumsum(pc$values)<cp*sum(pc$values))+1
# the smallest number of PCs such that the cumulative contribution
# rate is larger than cp.
return(as.matrix(pcvalue[,1:numpc]))
} # symb_covPCA.Interval ends

if(pcamethod=="symb_covPCA") {pcavalues=symb_covPCA.Interval(data=data,cp=cp)}

# the span normalized Euclidean Hausdorff distance
DSNEH=function(a,b,data)
{
  dd=yj=rep(0,nvar)
  dis=0
  for(i in 1:nvar)
  {yj[i]=max(data[,2*i])-min(data[,2*i-1])}
  for(i in 1:nvar)
  {dd[i]=max(abs(a[2*i-1]-b[2*i-1]),abs(a[2*i]-b[2*i]))}
  for(i in 1:nvar)
  {dis=dis+dd[i]^2/(yj[i])^2}
}

```

```

    return(sqrt(dis))
}

# the Ichino-Yaguchi distance
DIY=function(a,b,data,gamma=0.4){
  d1j=d2j=d3j=d4j=rep(0,nvar)
  dis=0
  for(i in 1:nvar)
  {
    d1j[i]=max(a[2*i],b[2*i])-min(a[2*i-1],b[2*i-1])
    d2j[i]=max(0,min(a[2*i],b[2*i])-max(a[2*i-1],b[2*i-1]))
    d3j[i]=a[2*i]-a[2*i-1]
    d4j[i]=b[2*i]-b[2*i-1]
  }
  for(i in 1:nvar)
  {dis=dis+d1j[i]-d2j[i]+gamma*(2*d2j[i]-d3j[i]-d4j[i])
  }
  return(dis)
}

# obtain the distance matrix d
d=matrix(rep(0,nob^2),ncol=nob,nrow=nob)
for(i in 1:nob){
  for(j in 1:nob){
    if(distance=="SNEH") d[i,j]=DSNEH(data[i,],data[j,],data)
    if(distance=="IY") d[i,j]=DIY(data[i,],data[j,],data)
  }
}

# the function to calculate the within-cluster variation (equal weight)
I=function(index){
  withinvari=0
  weight=rep(1,nob)
  lamd=sum(weight[index])
  for(i in 1:length(index)){
    for(j in 1:length(index)){
      withinvari=withinvari+
        1/(2*lamd)*weight[index[i]]*weight[index[j]]*(d[index[i],index[j]]^2)
    }
  }
  return(withinvari)
}

# the function to find the position where the cluster is to be partitioned
Isplit=function(index){
  if(length(index)==1) {aindex=0
  avalue=-Inf}

```

```

if(length(index)>1) {delta=rep(0,length(index)-1)
for(i in 1:(length(index)-1)){
  delta[i]=I(index)-I(index[1:i])-I(index[(i+1):length(index)])}
aindex=which.max(delta)
avalue=delta}
return(list(aindex,avalue))
}
# the bipartition: index[1:aindex] and index[(aindex+1):length(index)]

# the Dunn index (the input is a list)
dunn=function(a){
  l=length(a) # the number of clusters
  ber=I(a[[1]])
  for(i in 1:(l-1)){
    if(ber<I(a[[i+1]])) ber=I(a[[i+1]])
    else ber=ber}

  du=I(union(a[[1]],a[[2]]))-I(a[[1]])-I(a[[2]])
  for(i in 2:l){
    for (j in 1:(i-1))
      { if(du>(I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])))
        du=I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])
      }
    else du=du
  }
  return(du/ber)
}

# the Davis-Bouldin index (the input is a list)
DavB=function(a){
  l=length(a) # the number of clusters
  berr=duu=rep(0,l)
  for(i in 1:l)
  { lv=1:l
  numb=numb2=rep(0,length(lv[-i]))
  k=1
  for(j in lv[-i])
  {
    numb[k]=I(a[[j]])+I(a[[i]])
    numb2[k]=I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])
    k=k+1
  }
  berr[i]=max(numb)
  duu[i]=min(numb2)
  }
  return(mean(berr/duu))
}

```

```

# Hierarchical divisive clustering
noPC=ncol(pcavalues) # the number of PCs which will be used
result=list()
pcused=rep(0,nob-1)
clusters=list(c(1:nob))
DavBindex=dunnindex=rep(0,(nob-1))
for(j in 1:(nob-1))
{
  cz=bz=rep(0,noPC)
  for (i in 1:noPC)
  {az=rep(0,j)
   for(z in 1:j)
   {newindex=clusters[[z]][order(pcavalues[clusters[[z]],i])]
    # reorder the observations in each cluster by the ith PC
    az[z]=max(Isplit(newindex)[[2]]) # the maxdelta of each cluster by PCi
   }
   bz[i]=max(az,na.rm = TRUE) # the maxdelta by PCi
   cz[i]=which.max(az) # which cluster maximizes delta by PCi
  }
  pc.chosen=which.max(bz)
  cluster.chosen=cz[pc.chosen]
  newindex1=
clusters[[cluster.chosen]][order(pcavalues[clusters[[cluster.chosen]],pc.chosen])]
  posit=Isplit(newindex1)[[1]]
  clusters[[cluster.chosen]] <- newindex1[1:posit]
  clusters[[j+1]] <- newindex1[(posit+1):length(newindex1)]
  DavBindex[j]=DavB(clusters)
  dunnindex[j]=dunn(clusters)
  avec=c()
  for(k in 1:(j+1)){
    bvec=c(clusters[[k]],0)
    avec=c(avec,bvec)}
  result[[j]]=avec
  pcused[j]=pc.chosen
}

return(list(hierarchical_divisive_Clustering_tree=result,
           PC_used_each_stage=pcused,
           Davis_Bouldin_index=DavBindex,
           Dunn_index=dunnindex)
)
}

```

```
#####
#                               R Code for DIVE                               #
#####

vertexdiv.Interval <- function(distance="SNEH",data)
  # The DIVE algorithm based on the endpoints of the intervals or the
  # vertices of the hyper-rectangles.
  # distance = "SNEH" (the span normalized Euclidean Hausdorff distance),
  # or "IY" (the Ichino-Yaguchi distance).
  # data = an interval-valued data matrix
{
  nob=nrow(data) # the number of the observations
  nvar=ncol(data)/2 # the number of the variables

  # the span normalized Euclidean Hausdorff distance
  DSNEH=function(a,b,data)
  {
    dd=yj=rep(0,nvar)
    dis=0
    for(i in 1:nvar)
      {yj[i]=max(data[,2*i])-min(data[,2*i-1])}
    for(i in 1:nvar)
      {dd[i]=max(abs(a[2*i-1]-b[2*i-1]),abs(a[2*i]-b[2*i]))}
    for(i in 1:nvar)
      {dis=dis+dd[i]^2/(yj[i])^2}
    return(sqrt(dis))
  }

  # the Ichino-Yaguchi distance
  DIY=function(a,b,data,gamma=0.4){
    d1j=d2j=d3j=d4j=rep(0,nvar)
    dis=0
    for(i in 1:nvar)
      {
        d1j[i]=max(a[2*i],b[2*i])-min(a[2*i-1],b[2*i-1])
        d2j[i]=max(0,min(a[2*i],b[2*i])-max(a[2*i-1],b[2*i-1]))
        d3j[i]=a[2*i]-a[2*i-1]
        d4j[i]=b[2*i]-b[2*i-1]
      }
    for(i in 1:nvar)
      {dis=dis+d1j[i]-d2j[i]+gamma*(2*d2j[i]-d3j[i]-d4j[i])}
    }
    return(dis)
  }

  # obtain the distance matrix d

```

```

d=matrix(rep(0,nob^2),ncol=nob,nrow=nob)
for(i in 1:nob){
  for(j in 1:nob){
    if(distance=="SNEH") d[i,j]=DSNEH(data[i,],data[j,],data)
    if(distance=="IY") d[i,j]=DIY(data[i,],data[j,],data)
  }
}

# the function to calculate the within-cluster variation (equal weight)
I=function(index){
  withinvari=0
  weight=rep(1,nob)
  lamd=sum(weight[index])
  for(i in 1:length(index)){
    for(j in 1:length(index)){
      withinvari=withinvari+
        1/(2*lamd)*weight[index[i]]*weight[index[j]]*(d[index[i],index[j]]^2)
    }
  }
  return(withinvari)
}

# the function to find the position where the cluster is to be partitioned
Isplit=function(index){
  if(length(index)==1) {aindex=0
  avalue=-Inf}
  if(length(index)>1) {delta=rep(0,length(index)-1)
  for(i in 1:(length(index)-1)){
    delta[i]=I(index)-I(index[1:i])-I(index[(i+1):length(index)])}
  aindex=which.max(delta)
  avalue=delta}
  return(list(aindex,avalue))
}

# the bipartition: index[1:aindex] and index[(aindex+1):length(index)]

# the Dunn index (the input is a list)
dunn=function(a){
  l=length(a) # the number of clusters
  ber=I(a[[1]])
  for(i in 1:(l-1)){
    if(ber<I(a[[i+1]])) ber=I(a[[i+1]])
    else ber=ber}

  du=I(union(a[[1]],a[[2]]))-I(a[[1]])-I(a[[2]])
  for(i in 2:l){
    for (j in 1:(i-1))
      { if(du>(I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])))

```

```

        du=I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])
    else du=du    }
}
return(du/ber)
}

# the Davis-Bouldin index (the input is a list)
DavB=function(a){
  l=length(a) # the number of clusters
  berr=duu=rep(0,l)
  for(i in 1:l)
  { lv=1:l
    numb=numb2=rep(0,length(lv[-i]))
    k=1
    for(j in lv[-i])
    {
      numb[k]=I(a[[j]])+I(a[[i]])
      numb2[k]=I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])
      k=k+1
    }
    berr[i]=max(numb)
    duu[i]=min(numb2)
  }
  return(mean(berr/duu))
}

# Hierarchical divisive clustering
result=list()
clusters=list(c(1:nob))
variableused=rep(0,nob-1)
DavBindex=dunnindex=rep(0,(nob-1))
for(j in 1:(nob-1))
{
  cz=bz=rep(0,nvar)
  for (i in 1:nvar)
  {az=rep(0,j)
    for(z in 1:j)
    {newindexl=clusters[[z]][order(data[clusters[[z]],2*i-1])]
      # reorder each cluster by the lower bounds of the intervals
      newindexu=clusters[[z]][order(data[clusters[[z]],2*i)]]
      # reorder each cluster by the upper bounds of the intervals
      if(max(Isplit(newindexl)[[2]])>=max(Isplit(newindexu)[[2]]))
        {newindex=newindexl}
      else {newindex=newindexu}
      # reorder the observations in each cluster
      az[z]=max(Isplit(newindex)[[2]]) # the maxdelta of each cluster
    }
  }
}

```

```

bz[i]=max(az,na.rm = TRUE) # the maxdelta by each variable
cz[i]=which.max(az) # which cluster maximizes delta
}
variable.chosen=which.max(bz)
cluster.chosen=cz[variable.chosen]

newindexlow=
clusters[[cluster.chosen]][order(data[clusters[[cluster.chosen]],2*i-1])]
newindexup=
clusters[[cluster.chosen]][order(data[clusters[[cluster.chosen]],2*i])]
# reorder the chosen cluster by the endpoints of the intervals
if(max(Isplit(newindexlow)[[2]])>=max(Isplit(newindexup)[[2]]))
  {newindex1=newindexlow}
else {newindex1=newindexup}

posit=Isplit(newindex1)[[1]]
clusters[[cluster.chosen]] <- newindex1[1:posit]
clusters[[j+1]] <- newindex1[(posit+1):length(newindex1)]
DavBindex[j]=DavB(clusters)
dunnindex[j]=dunn(clusters)
avec=c()
for(k in 1:(j+1)){
  bvec=c(clusters[[k]],0)
  avec=c(avec,bvec)}
result[[j]]=avec
variableused[j]=variable.chosen
}

return(list(Variable_used_each_stage=variableused,
            hierarchical_divisive_Clustering_tree=result,
            Davis_Bouldin_index=DavBindex,
            Dunn_index=dunnindex
            ))
}

#####
#                               R Code for DIVM                               #
#####

monodiv.Interval <- function(distance="SNEH",data)
# the monothetic divisive algorithm proposed by Chavent (1998, 2000).
# We use DIVM (the divisive algorithm based on the midpoints of intervals)
# to denote Chavent's algortihm.
# distance = "SNEH" (the span normalized Euclidean Hausdorff distance),

```

```

# or "IY" (the Ichino-Yaguchi distance).
# data = an interval-valued data matrix
{ nob=nrow(data) # the number of the observations
  nvar=ncol(data)/2 # the number of the variables

# the span normalized Euclidean Hausdorff distance
DSNEH=function(a,b,data)
{
  dd=yj=rep(0,nvar)
  dis=0
  for(i in 1:nvar)
  {yj[i]=max(data[,2*i])-min(data[,2*i-1])}
  for(i in 1:nvar)
  {dd[i]=max(abs(a[2*i-1]-b[2*i-1]),abs(a[2*i]-b[2*i]))}
  for(i in 1:nvar)
  {dis=dis+dd[i]^2/(yj[i])^2}
  return(sqrt(dis))
}

# the Ichino-Yaguchi distance
DIY=function(a,b,data,gamma=0.4){
  d1j=d2j=d3j=d4j=rep(0,nvar)
  dis=0
  for(i in 1:nvar)
  {
    d1j[i]=max(a[2*i],b[2*i])-min(a[2*i-1],b[2*i-1])
    d2j[i]=max(0,min(a[2*i],b[2*i])-max(a[2*i-1],b[2*i-1]))
    d3j[i]=a[2*i]-a[2*i-1]
    d4j[i]=b[2*i]-b[2*i-1]
  }
  for(i in 1:nvar)
  {dis=dis+d1j[i]-d2j[i]+gamma*(2*d2j[i]-d3j[i]-d4j[i])}
  }
  return(dis)
}

# obtain the distance matrix d
d=matrix(rep(0,nob^2),ncol=nob,nrow=nob)
for(i in 1:nob){
  for(j in 1:nob){
    if(distance=="SNEH") d[i,j]=DSNEH(data[i,],data[j,],data)
    if(distance=="IY") d[i,j]=DIY(data[i,],data[j,],data)
  }
}

# the function to calculate the within-cluster variation (equal weight)
I=function(index){

```

```

withinvari=0
weight=rep(1,nob)
lamd=sum(weight[index])
for(i in 1:length(index)){
  for(j in 1:length(index)){
    withinvari=withinvari+
      1/(2*lamd)*weight[index[i]]*weight[index[j]]*(d[index[i],index[j]]^2)
  }
}
return(withinvari)
}

# the function to find the position where the cluster is to be partitioned
Isplit=function(index){
  if(length(index)==1) {aindex=0
  avalue=-Inf}
  if(length(index)>1) {delta=rep(0,length(index)-1)
  for(i in 1:(length(index)-1)){
    delta[i]=I(index)-I(index[1:i])-I(index[(i+1):length(index)])}
  aindex=which.max(delta)
  avalue=delta}
  return(list(aindex,avalue))
}

# the bipartition: index[1:aindex] and index[(aindex+1):length(index)]

# the Dunn index (the input is a list)
dunn=function(a){
  l=length(a) # the number of clusters
  ber=I(a[[1]])
  for(i in 1:(l-1)){
    if(ber<I(a[[i+1]])) ber=I(a[[i+1]])
    else ber=ber}

  du=I(union(a[[1]],a[[2]]))-I(a[[1]])-I(a[[2]])
  for(i in 2:l){
    for (j in 1:(i-1))
    { if(du>(I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])))
      du=I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])
    else du=du      }
  }
  return(du/ber)
}

# the Davis-Bouldin index (the input is a list)
DavB=function(a){
  l=length(a) # the number of clusters
  berr=duu=rep(0,l)

```

```

for(i in 1:l)
{
  lv=1:l
  numb=numb2=rep(0,length(lv[-i]))
  k=1
  for(j in lv[-i])
  {
    numb[k]=I(a[[j]])+I(a[[i]])
    numb2[k]=I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])
    k=k+1
  }
  berr[i]=max(numb)
  duu[i]=min(numb2)
}
return(mean(berr/duu))
}

# Hierarchical divisive clustering
meanvalues=matrix(rep(0,nob*nvar),nrow=nob,ncol=nvar,byrow=T)
for(i in 1:nvar){
  meanvalues[,i]=(data[,2*i-1]+data[,2*i])/2}
result=list()
clusters=list(c(1:nob))
variableused=rep(0,nob-1)
DavBindex=dunnindex=rep(0,(nob-1))
for(j in 1:(nob-1))
{
  cz=bz=rep(0,nvar)
  for (i in 1:nvar)
  {az=rep(0,j)
  for(z in 1:j)
  {newindex=clusters[[z]][order(meanvalues[clusters[[z]],i])]
  # reorder the observations in each cluster
  az[z]=max(Isplit(newindex)[[2]]) # the maxdelta of each cluster
  }
  bz[i]=max(az,na.rm = TRUE) # the maxdelta by each variable
  cz[i]=which.max(az) # which cluster maximizes delta
  }
  variable.chosen=which.max(bz)
  c.chosen=cz[variable.chosen]
  newindex1=
clusters[[c.chosen]][order(meanvalues[clusters[[c.chosen]],variable.chosen])]
  posit=Isplit(newindex1)[[1]]
  clusters[[c.chosen]] <- newindex1[1:posit]
  clusters[[j+1]] <- newindex1[(posit+1):length(newindex1)]
  DavBindex[j]=DavB(clusters)
  dunnindex[j]=dunn(clusters)
  avec=c()
}

```

```

for(k in 1:(j+1)){
  bvec=c(clusters[[k]],0)
  avec=c(avec,bvec)}
result[[j]]=avec
variableused[j]=variable.chosen
}

return(list(Variable_used_each_stage=variableused,
            hierarchical_divisive_Clustering_tree=result,
            Davis_Bouldin_index=DavBindex,
            Dunn_index=dunnindex
            ))
}

#####
#                               R Code for DIVECPCA and DIVEWPCA                               #
#####

mixeddiv.Interval<-function(distance="SNEH",pcamethod="weightedPCA",cp=0.9,data)
# the two mixed-strategy algorithms DIVECPCA and DIVEWPCA based on the
# symbolic covariance PCA method and the weighted symbolic PCA method.
# distance = "SNEH" (the span normalized Euclidean Hausdorff distance)
# or "IY" (the Ichino-Yaguchi distance).
# pcamethod = "symb_covPCA" (the symbolic covariance PCA method)
# or "weightedPCA" (the weighted symbolic PCA method).
# cp = a given cumulative contribution rate of principal components
# data = an interval-valued data matrix
{
  nob=nrow(data) # the number of the observations
  nvar=ncol(data)/2 # the number of the variables

  # the weighted symbolic PCA method
  wPCA.Interval=function(data)
  {
    # obtain the weights
    w=matrix(rep(0,nob*nvar),nrow=nob,ncol=nvar,byrow=T)
    for (col in 1:nvar){
      a=sort(unique (c(data[,2*col-1],data[,2*col])))
      # define the subintervals, duplicate elements removed.
      t=b=rep(0,length(a)-1)
      # b denotes the frequency of each subinterval.
      # t indicates whether the interval contains each subinterval.
      for(i in 1:(length(a)-1))
        {b[i]=sum((data[,2*col-1]<=a[i])&(data[,2*col]>=a[i+1]))}
    }
  }
}

```

```

    for(j in 1:nob){
      for(i in 1:(length(a)-1)) {
        t[i]=(data[j,2*col-1]<=a[i])&(data[j,2*col]>=a[i+1])}
        for(i in 1:(length(a)-1)) {
          w[j,col]=t[i]*b[i]*(a[i+1]-a[i])+w[j,col]
        }
      }
    }
  }
weight=scale(w, center = FALSE, scale = TRUE)
weight=rowSums(weight)
weight=weight/sum(weight)
weight=diag(weight)

Xu=Xl=matrix(rep(0,nob*nvar),nrow=nob,ncol=nvar,byrow=T)
for(i in 1:nvar){
  Xu[,i]=data[,2*i-1]
  Xl[,i]=data[,2*i]
}
Xu=scale(Xu, center = TRUE, scale = FALSE)
# scale = TRUE if the unit variance scaling is needed.
Xl=scale(Xl, center = TRUE, scale = FALSE)
# scale = TRUE if the unit variance scaling is needed.
pcu=eigen(cov(weight%%Xu))
pcl=eigen(cov(weight%%Xl))
pcvalueu=weight%%Xu%%pcu$vectors
pcvaluel=weight%%Xl%%pcl$vectors
numpcu=sum(cumsum(pcu$values)<cp*sum(pcu$values))+1
numpcl=sum(cumsum(pcl$values)<cp*sum(pcl$values))+1
# the smallest number of PCs such that the cumulative contribution
# rate is larger than cp.
return(as.matrix(cbind(pcvalueu[,1:numpcu],pcvaluel[,1:numpcl])))
} # wPCA.Interval ends

if(pcamethod=="weightedPCA") {pcavalues=wPCA.Interval(data=data)}

# the symbolic covariance PCA method
symb_covPCA.Interval=function(data){
  p=nvar
  m=nob
  cov=cor=matrix(0, p, p)
  x_mean=rep(0, p)

  # the variable means
  for (i in 1:p){
    x_mean[i]=(mean(data[,2*i-1])+mean(data[,2*i]))/2}

  # the covariance matrix

```

```

for (k in 1:p){
  for (l in 1:p){
    q = 0
    for (h in 1:m){
      q = q + 2*(data[h,2*k-1]-x_mean[k])*(data[h,2*l-1]-x_mean[l])
      + (data[h,2*k-1]-x_mean[k])*(data[h,2*l]-x_mean[l])
      + (data[h,2*k]-x_mean[k])*(data[h,2*l-1]-x_mean[l])
      + 2*(data[h,2*k]-x_mean[k])*(data[h,2*l]-x_mean[l])
    }
    cov[k,l] = q/6/m
  }
}

# the correlation matrix
for (k in 1:p){
  for (l in 1:p){
    cor[k,l]=cov[k,l]/sqrt(cov[k,k])/sqrt(cov[l,l])
  }
}

Xu=Xl=matrix(rep(0,nob*nvar),nrow=nob,ncol=nvar,byrow=T)
for(i in 1:nvar){
  Xu[,i]=data[,2*i-1]
  Xl[,i]=data[,2*i]
}
Xu=scale(Xu, center = TRUE, scale = FALSE)
# scale = TRUE if the unit variance scaling is needed.
Xl=scale(Xl, center = TRUE, scale = FALSE)
# scale = TRUE if the unit variance scaling is needed.
pc=eigen(cov)
# pc=eigen(cor) if the unit variance scaling is performed on Xu and Xl.
pcvalueu=Xu%*%pc$vectors
pcvaluel=Xl%*%pc$vectors
numpc=sum(cumsum(pc$values)<cp*sum(pc$values))+1
# the smallest number of PCs such that the cumulative contribution
# rate is larger than cp.
return(as.matrix(cbind(pcvalueu[,1:numpc],pcvaluel[,1:numpc])))
} # symb_covPCA.Interval ends

if(pcamethod=="symb_covPCA") {pcavalues=symb_covPCA.Interval(data=data)}

# the span normalized Euclidean Hausdorff distance
DSNEH=function(a,b,data)
{
  dd=yj=rep(0,nvar)
  dis=0
  for(i in 1:nvar)

```

```

{yj[i]=max(data[,2*i])-min(data[,2*i-1])}
for(i in 1:nvar)
{dd[i]=max(abs(a[2*i-1]-b[2*i-1]),abs(a[2*i]-b[2*i]))}
for(i in 1:nvar)
{dis=dis+dd[i]^2/(yj[i])^2}
return(sqrt(dis))
}

# the Ichino-Yaguchi distance
DIY=function(a,b,data,gamma=0.4){
  d1j=d2j=d3j=d4j=rep(0,nvar)
  dis=0
  for(i in 1:nvar)
  {
    d1j[i]=max(a[2*i],b[2*i])-min(a[2*i-1],b[2*i-1])
    d2j[i]=max(0,min(a[2*i],b[2*i])-max(a[2*i-1],b[2*i-1]))
    d3j[i]=a[2*i]-a[2*i-1]
    d4j[i]=b[2*i]-b[2*i-1]
  }
  for(i in 1:nvar)
  {dis=dis+d1j[i]-d2j[i]+gamma*(2*d2j[i]-d3j[i]-d4j[i])}
  }
  return(dis)
}

# obtain the distance matrix d
d=matrix(rep(0,nob^2),ncol=nob,nrow=nob)
for(i in 1:nob){
  for(j in 1:nob){
    if(distance=="SNEH") d[i,j]=DSNEH(data[i,],data[j,],data)
    if(distance=="IY") d[i,j]=DIY(data[i,],data[j,],data)
  }
}

# the function to calculate the within-cluster variation (equal weight)
I=function(index){
  withinvari=0
  weight=rep(1,nob)
  lamd=sum(weight[index])
  for(i in 1:length(index)){
    for(j in 1:length(index)){
      withinvari=withinvari+
        1/(2*lamd)*weight[index[i]]*weight[index[j]]*(d[index[i],index[j]]^2)
    }
  }
  return(withinvari)
}

```

```

# the function to find the position where the cluster is to be partitioned
Isplit=function(index){
  if(length(index)==1) {aindex=0
    avalue=-Inf}
  if(length(index)>1) {delta=rep(0,length(index)-1)
    for(i in 1:(length(index)-1)){
      delta[i]=I(index)-I(index[1:i])-I(index[(i+1):length(index)])}
    aindex=which.max(delta)
    avalue=delta}
  return(list(aindex,avalue))
}
# the bipartition: index[1:aindex] and index[(aindex+1):length(index)]

# the Dunn index (the input is a list)
dunn=function(a){
  l=length(a) # the number of clusters
  ber=I(a[[1]])
  for(i in 1:(l-1)){
    if(ber<I(a[[i+1]])) ber=I(a[[i+1]])
    else ber=ber}

  du=I(union(a[[1]],a[[2]]))-I(a[[1]])-I(a[[2]])
  for(i in 2:l){
    for (j in 1:(i-1))
      { if(du>(I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])))
        du=I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])
        else du=du      }
    }
  return(du/ber)
}

# the Davis-Bouldin index (the input is a list)
DavB=function(a){
  l=length(a) # the number of clusters
  berr=duu=rep(0,l)
  for(i in 1:l)
    { lv=1:l
      numb=numb2=rep(0,length(lv[-i]))
      k=1
      for(j in lv[-i])
        {
          numb[k]=I(a[[j]])+I(a[[i]])
          numb2[k]=I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])
          k=k+1
        }
      berr[i]=max(numb)
    }
}

```

```

    duu[i]=min(numb2)
  }
  return(mean(berr/duu))
}

# Mixed-strategy Hierarchical Divisive Clustering
noPC=ncol(pcavalues) # the number of PCs which will be used
result=list()
pcused=rep(0,nob-1)
clusters=list(c(1:nob))
DavBindex=dunnindex=rep(0,(nob-1))
for(j in 1:(nob-1))
{
  cz=bz=rep(0,noPC)
  for (i in 1:noPC)
  {az=rep(0,j)
  for(z in 1:j)
  {newindex=clusters[[z]][order(pcavalues[clusters[[z]],i])]
  #reorder the observations in each cluster by the ith PC
  az[z]=max(Isplit(newindex)[[2]]) # the maxdelta of each cluster by PCi
  }
  bz[i]=max(az,na.rm = TRUE) # the maxdelta by PCi
  cz[i]=which.max(az) # which cluster maximizes delta by PCi
  }
  pc.chosen=which.max(bz)
  cluster.chosen=cz[pc.chosen]
  newindex1=
clusters[[cluster.chosen]][order(pcavalues[clusters[[cluster.chosen]],pc.chosen])]
  posit=Isplit(newindex1)[[1]]
  clusters[[cluster.chosen]] <- newindex1[1:posit]
  clusters[[j+1]] <- newindex1[(posit+1):length(newindex1)]
  DavBindex[j]=DavB(clusters)
  dunnindex[j]=dunn(clusters)
  avec=c()
  for(k in 1:(j+1)){
    bvec=c(clusters[[k]],0)
    avec=c(avec,bvec)}
  result[[j]]=avec
  pcused[j]=pc.chosen
}

return(list(hierarchical_divisive_Clustering_tree=result,
           PC_used_each_stage=pcused,
           Davis_Bouldin_index=DavBindex,
           Dunn_index=dunnindex

)))}

```

```
#####
# R Code for the Polythetic Divisive Algorithm based on Minimum Spanning Trees #
#####
```

```
library(vegan)
library(igraph)
```

```
nob=nrow(data) # obtain the number of the observations
nvar=ncol(data)/2 # obtain the number of the variables
```

```
# the span normalized Euclidean Hausdorff distance
```

```
DSNEH=function(a,b,data)
```

```
{
  dd=yj=rep(0,nvar)
  dis=0
  for(i in 1:nvar)
  {yj[i]=max(data[,2*i])-min(data[,2*i-1])}
  for(i in 1:nvar)
  {dd[i]=max(abs(a[2*i-1]-b[2*i-1]),abs(a[2*i]-b[2*i]))}
  for(i in 1:nvar)
  {dis=dis+dd[i]^2/(yj[i])^2}
  return(sqrt(dis))
}
```

```
# the Gowda-Diday dissimilarity
```

```
DGD=function(a,b,data){
```

```
  yj=kj=Ij=d1j=d2j=d3j=rep(0,nvar)
  dis=0
  for(i in 1:nvar)
  {yj[i]=max(data[,2*i])-min(data[,2*i-1])
    kj[i]=max(a[2*i],b[2*i])-min(a[2*i-1],b[2*i-1])
    Ij[i]=max(0,min(a[2*i],b[2*i])-max(a[2*i-1],b[2*i-1]))
    d1j[i]=(abs(b[2*i]-a[2*i]-b[2*i-1]+a[2*i-1]))/kj[i]
    d2j[i]=(b[2*i]-b[2*i-1]+a[2*i]-a[2*i-1]-2*Ij[i])/kj[i]
    d3j[i]=abs(b[2*i-1]-a[2*i-1])/yj[i]
  }
  for(i in 1:nvar)
  {dis=dis+d1j[i]+d2j[i]+d3j[i]}
  return(dis)
}
```

```
# the Ichino-Yaguchi distance
```

```
DIY=function(a,b,data,gamma=0.4){
```

```
  d1j=d2j=d3j=d4j=rep(0,nvar)
  dis=0
  for(i in 1:nvar)
```

```

{
  d1j[i]=max(a[2*i],b[2*i])-min(a[2*i-1],b[2*i-1])
  d2j[i]=max(0,min(a[2*i],b[2*i])-max(a[2*i-1],b[2*i-1]))
  d3j[i]=a[2*i]-a[2*i-1]
  d4j[i]=b[2*i]-b[2*i-1]
}
for(i in 1:nvar)
{dis=dis+d1j[i]-d2j[i]+gamma*(2*d2j[i]-d3j[i]-d4j[i])
}
return(dis)
}

# specify the distance measure applied in this algorithm here
distance="SNEH"
# distance="IY"
# distance="GD"

# obtain the distance matrix d
d=matrix(rep(0,nob^2),ncol=nob,nrow=nob)
for(i in 1:nob){
  for(j in 1:nob){
    if(distance=="SNEH") d[i,j]=DSNEH(data[i,],data[j,],data)
    if(distance=="GD") d[i,j]=DGD(data[i,],data[j,],data)
    if(distance=="IY") d[i,j]=DIY(data[i,],data[j,],data)
  }
}

# the function to calculate the within-cluster variation (equal weight)
I=function(index){
  withinvari=0
  weight=rep(1,nob)
  lamd=sum(weight[index])
  for(i in 1:length(index)){
    for(j in 1:length(index)){
      withinvari=withinvari+
        1/(2*lamd)*weight[index[i]]*weight[index[j]]*(d[index[i],index[j]]^2)
    }
  }
  return(withinvari)
}

# the Dunn index (the input is a list)
dunn=function(a){
  l=length(a) # the number of clusters
  ber=I(a[[1]])
  for(i in 1:(l-1)){
    if(ber<I(a[[i+1]])) ber=I(a[[i+1]])
  }
}

```

```

    else ber=ber}

du=I(union(a[[1]],a[[2]]))-I(a[[1]])-I(a[[2]])
for(i in 2:l){
  for (j in 1:(i-1))
  { if(du>(I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])))
    du=I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])
    else du=du      }
}
return(du/ber)
}

# the Davis-Bouldin index (the input is a list)
DavB=function(a){
  l=length(a) # the number of clusters
  berr=duu=rep(0,l)
  for(i in 1:l)
  { lv=1:l
  numb=numb2=rep(0,length(lv[-i]))
  k=1
  for(j in lv[-i])
  {
    numb[k]=I(a[[j]])+I(a[[i]])
    numb2[k]=I(union(a[[i]],a[[j]]))-I(a[[i]])-I(a[[j]])
    k=k+1
  }
  berr[i]=max(numb)
  duu[i]=min(numb2)
}
return(mean(berr/duu))
}

# obtain the edges which connect the observations by the indices
edgesfromindex<- function(edg,index,nob){
  ade=rep(FALSE,(nob-1))
  for(i in 1:(nob-1)){
    if( is.element(edg[1,i],index) && is.element(edg[2,i],index)) ade[i]=TRUE
  }
  return(as.matrix(edg[,ade]))
}

# obtain a cluster of objects from the edges of a subtree
# edges: a matrix with two rows (the number of edges >= 3)
indexfromedges<- function(edges){
  gvec=c(edges[1,],edges[2,])
  return(unique(gvec))
}

```

```

# obtain two clusters of objects from the edges of two subtrees
# edges: a matrix with two rows (the number of edges >= 3)
clustfromedges<- function(edges, index){
  gvec=c(edges[1,],edges[2,])
  a=names(which.max(table(gvec)))
  ga <- graph_from_data_frame(data.frame(from = c(edges[1,],edges[2,]),
                                          to   = c(edges[2,],edges[1,]) ) )

  cluster1=as.numeric(labels((ego(ga, length(V(ga)), a))[[1]]))
  cluster2=setdiff(index,cluster1)
  return(list(cluster1, cluster2))
}

# the main function for the polythetic algorithm
# run MSTclust(dismatrix=d) to obtain the clustering result
MSTclust <- function(dismatrix)
  # dismatrix : the distance matrix
{ nob=nrow(dismatrix)
  DavBindex=dunnindex=rep(0,(nob-1))
  clusters=list(c(1:nob))
  # record the clusters at one single clustering stage
  tr=spantree(d, toolong = 0) # obtain the MST
  edg=rbind(tr$kid,2:nob)
  # a matrix with two rows to record the edges of the MST
  fresult=list()
  for(s in 1:(nob-1)) # the sth stage: s clusters

  {
    result=list()
    for(c in 1:s){

      if(length(clusters[[c]])==2)
      { result=c(result,c)
        result=c(result,I(clusters[[c]]))
        result=c(result,list(clusters[[c]][1]))
        result=c(result,list(clusters[[c]][2])) }

      if(length(clusters[[c]])>2){
        cedg=edgesfromindex(edg=edg, index=clusters[[c]], nob=nob)
        cc=ncol(cedg)
        for(k in 1:cc){ reedg=as.matrix(cedg[,-k])
          ptresult=clustfromedges(edges=reedg, clusters[[c]])
          result=c(result,c)
          result=c(result,I(clusters[[c]])-I(ptresult[[1]])-I(ptresult[[2]]))
          result=c(result,list(ptresult[[1]]))
          result=c(result,list(ptresult[[2]]))
        }
      }
    }
  }
}

```

```

    }
    }
    }

    delt=c()
    for(w in 1:(length(result)/4))
    {delt[w]=result[[4*w-2]]}
    ccc=which.max(delt)
    divclust=result[[4*ccc-3]]
    clusters=c(clusters,list(result[[4*ccc-1]]))
    clusters=c(clusters,list(result[[4*ccc]]))
    clusters=clusters[-(divclust)]

    avec=c()
    for(k in 1:length(clusters)){
        bvec=c(clusters[[k]],0)
        avec=c(avec,bvec)}
    fresult[[s]]=avec

    DavBindex[s]=DavB(clusters)
    dunnindex[s]=dunn(clusters)
    }
    return(list(fresult, DavBindex, dunnindex))
}

```