Preserving nearest neighbor consistency in cluster analysis

by

Jong-Seok Lee

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Industrial Engineering

Program of Study Committee:
Sigurdur Olafsson, Major Professor
John Jackman
Lizhi Wang
Dianne Cook
Dan Zhu

Iowa State University

Ames, Iowa

2009

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this dissertation.

First and foremost, I would like to thank Dr. Sigurdur Olafsson for his guidance and support throughout this research and the writing of this dissertation. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education toward a Ph.D. I would also like to thank my committee members for their efforts and helpful comments to this work.

I would additionally like to thank Dr. Chi-Hyuck Jun for his guidance throughout the initial stages of my graduate career and Dr. Kyoung-Shim Lee for her inspirational teaching style.

ABSTRACT

The two main streams in finding cluster structure from data could be to identify the number of natural clusters and, of course, to group the objects in a reasonable way. In order to achieve good results for these two, measuring goodness of clustering is required prior to beginning any related studies because it helps to establish a definition of cluster that could be ambiguous by individuals having different opinions on it. In this research we are concerned about the compactness and the connectivity of cluster as our goodness measurements. The former has been regarded as one of the most important properties that should be accomplished in a clustering task, whereas the latter that we think as a significant factor has received less attention. Since we believe that both are individually important, we employ them for better estimating the number of clusters and clustering objects. A new estimating method produces a set of promising estimates by measuring compactness and connectivity from clustered datasets which look similar to the original data but have an amount of perturbation, and then determines a single optimal number by majority voting scheme. The connectivity measure newly introduced in our research is also used as an objective to be achieved in clustering objects. We propose a new clustering algorithm, named as *CNCLUST*, that works in a way to optimize the quantity of connectivity. The proposed clustering algorithm is a greedy heuristic that looks like a single linkage method, but it is distinguishable by the fact that it first considers local compactness of objects and later incorporates it into global connectivity. We conducted numerical experiments in order to evaluate the performances of the proposed methods based on simulated datasets and a real data. The results seem optimistic.

CHAPTER 1. INTRODUCTION

1.1 Clustering and Its Applications

In the last decades, a variety methods for data mining, also called as machine learning or pattern recognition, have been developed and successfully applied to various fields, such as bioinformatics, computer vision, power system, information retrieval, fraud detection, financial prediction, and etc (Chan et al., 1999; Haussler, 1999; Brown et al., 2000; Forsyth and Ponce, 2003; Moulin et al., 2004; Lee et al., 2005). Generally, the scenarios of data mining are divided into two main streams which are classification (with a discrete type of response variables) including prediction (with a continuous type of response variables) and clustering. Clustering is distinguished from classification by the fact that there is no *a priori* output, and that is why it is called unsupervised learning while classification is called supervised learning. Therefore, clustering algorithms are expected to produce partitions that reflect the internal structure of the data and identify natural groups. Because of such a nature of clustering, it is acknowledged that data clustering is a challenging task and an ill-posed problem when prior information about the underlying data distributions is not well defined. Nevertheless, it has been successfully applied, as an exploratory pattern analysis, to many different applications including image segmentation, constructing prototypes of classifiers, understanding genomic data, market segmentation, and etc (Jain et al., 1999). Some of those applications will be briefly mentioned in the rest of this section.

One of the practical motivations of image segmentation is to make special effects in making films. This thus means that the image segmentation is used to locate objects and boundaries (lines, curves, etc.) in images, so, for example, we can remove the current background of an image and apply another new background in order to artificially give a designated special

effect on the image. Again, image segmentation is the process of assigning a label to every single pixel in an image such that pixels with the same label share certain common visual properties (Shapiro and Stockman, 2001). This process of assigning labels is therefore defined as a clustering problem, and various approaches to this problem have been developed in its literature (Pham et al., 2000; Liew and Yan, 2005; Levin et al., 2008).

Market segmentation that enables us to plan a specific promotion or target at a certain market is another major application of clustering, since segmentation means grouping customers with similar needs and responses (Punj and Stewart, 1983). In general, people make a survey to collect customer responses and apply an appropriate clustering method based on datasets observed on some demographic variables, psychographic variables, behavioral variables, and etc., in order to group customers. Some previously unknown but interesting knowledge can be found from this analysis and such findings are then used to more accurately meet the needs of selected customers in a more profitable way (May et al., 2001).

Spatial data clustering shows several applications of cluster analysis to criminal detection, traffic incidence detection, defects classification in manufacturing systems, and etc. (Nath, 2006; Sheu, 2002; Lee et al., 2005). In this field of applications, clustering typically constructs prototypes of classifiers to discriminate a new object such as incidence or defect into one of conducted classes. Since the objects in the initial learning stage are generally unlabeled, a clustering method can be applied to the dataset to generate specific types of patterns.

In this research, the application of clustering under our consideration is about foodborne disease outbreaks. In most states, the diagnosed cases of certain serious infections are reported to health department that also reports to Center for Disease Control and Prevention (CDC) through the National Outbreak Reporting System (NORS). Since foodborne outbreak investigations made in a timely manner can lead us to rapid identification of corresponding etiologies and thus to taking an appropriate action for prevention and control of diseases, it is necessary to come up with a methodology for identification of patterns of outbreaks. However, it has not been yet investigated through cluster analysis; nonetheless cluster analysis has been successfully applied to many other fields as mentioned above. With this application-oriented

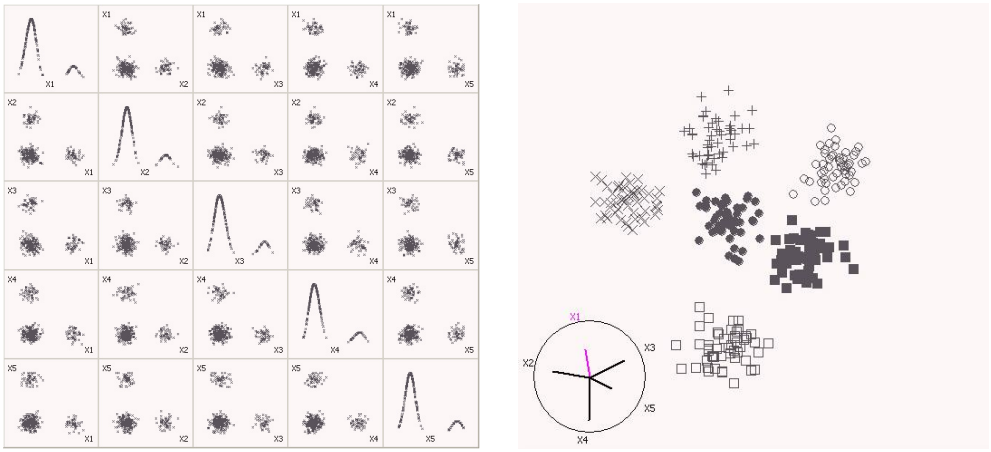
motivation we, therefore, apply our clustering method to the dataset from CDC in Chapter 5.

1.2 Why Cluster Analysis

In this section we briefly mention the answer to the question in the section title with regard to data visualization. Let us take a look at Figure 3.1 containing two examples of clusters. It is clear to notice that the first example has three clusters and the second one has two clusters, because they are observed in two-dimensional space. We also recognize which object should belong to which cluster without any difficulties. This means that if the dataset we are investigating lies on up to three dimensions, we then notice its cluster structure by visualizing the dataset and we do not have to use any clustering algorithms or cluster validation methods. However, almost all data we obtain from real world domains is typically much more high-dimensional.

Due to some difficulties of high-dimensional data visualization, people have made a tremendous amount of efforts on developing better visualization methods. Scatter plot matrix is one of the most often used and good visualization methods for a dataset observed in a high-dimensional space. Given a set of variables, the scatter plot matrix contains all the pair-wise scatter plots of the variables on a single page in a matrix format. Figure 1.1a shows an example. The dataset in this example was generated as follows. 50 objects were generated from standard multivariate normal distribution with the center of $\mu_1 = (0, 0, 0, 0, 0)$. Another 50 objects were then generated again from the same distribution but with a different center, $\mu_2 = (1, 0, 0, 0, 0)$, meaning that we moved the center of the density along with the first axis. We repeated this procedure for $\mu_3 = (0, 1, 0, 0, 0)$, $\mu_4 = (0, 0, 1, 0, 0)$, $\mu_5 = (0, 0, 0, 1, 0)$, and $\mu_6 = (0, 0, 0, 0, 1)$. Hence, this dataset has clearly six clusters in five dimensions. However, its scatter plot matrix failed to reveal the true cluster structure as shown in the figure. The figure actually misleads us to three clusters rather than six clusters. However, more sophisticated visualization methods may overcome the visualization difficulty of this dataset, and Figure 1.1b shows a good example. We used the 2D tour provided in GGobi software (http://www.ggobi.org/) to visualize the same dataset. The idea behind this 2D tour is that we may be able to see interesting

two-dimensional projections of high dimensional data if we look a sequence of projections long enough. As can be seen in Figure 1.1b it clearly shows the structure of six clusters.



(a) Scatter plot matrix (b) 2D tour plot

Figure 1.1: Two different visualizations of a same dataset

Nevertheless, there still exist limitations of visualizing high-dimensional datasets. For example, a dataset observed in a high-dimensional and discrete space may be very difficult to be shown graphically with its full representation of variation or distribution. Moreover, we cannot entirely rely on the visualization methods to discover the structure of clusters in data. Clustering has here its own reason for existing and that is why people have been working on developing cluster algorithms, improving its performance, and validating the resulting clusters.

1.3 Research Initiative

A myriad number of clustering algorithms have been proposed during the last half of decade. Some have their origins in graph theory, whereas the others are based on statistical pattern recognition, optimization and more. Comparing the relative benefits of various clustering methods is quite difficult to be made due to the fact that when applied to the same dataset, different clustering algorithms often lead to remarkably different clustering results. In some cases such differences are expected, since different algorithms make different assumptions about the structure of the data. An ideal one may be able to group objects so that most of human

eyes or opinions can agree on the result as a 'good' clustering, but it may be very difficult to come up with such a beautiful method.

Although some of clustering methods generate some special partitions, it has been observed that a majority of existing methods tends to cluster objects so that they form a cluster in which the objects are spherically distributed. In the optimization point of view, this phenomenon is caused by their objective that pursues the compactness as a goodness of clustering. For instance, the objective of the k-means clustering, which is one of the most famous clustering algorithms, is to minimize the sum of squared distances between the centroid in a cluster and the other objects in the same cluster. A similar objective can be found in many other clustering studies.

Based on the objective pursuing the compactness, people have also validated a clustering result in order to decide whether the result is good or not. For estimating the number of natural clusters, many estimating methods have looked for the point where it is meaningless to increase the number of clusters by having improved compactness. In other words, people deicide the optimal number of clusters when the improvement of compactness is not significant although we increase the number of clusters. It is done by seeking for an elbow point on the plot of the compactness against the number of clusters. As for this compactness measure, people have used the same computation with the objective in clustering, which is the within-cluster sum of squares.

However, there may be another goodness measure to evaluate a clustering result. Our research motivation comes from this initiative. Since we believe that a good clustering result should preserve the close relationships among objects such that the close objects are clustered together, two intuitive concepts called as 'cluster k-NN consistency' and 'cluster k-MN consistency' are first considered to provide a new goodness measurement which is the connectivity. We believe that these concepts are simple but powerful to conduct a good clustering paradigm. We propose this goodness measure for a better cluster analysis, including cluster validation and clustering algorithm itself, which hopefully can handle any arbitrarily shaped datasets. Hence, a new clustering method and estimating method will be studied in this research.

This research article has the following structure. We introduce a basic foundation of cluster analysis in the following chapter. It includes both clustering methods and cluster validation. The criteria, compactness and connectivity, that we pursue in this research are mentioned in Chapter 3. Based on those objectives, a new estimating method and a heuristic approach for data clustering will be proposed in Chapter 4 and Chapter 5 respectively. We conclude and suggest our future studies in Chapter 6.

CHAPTER 2. REVIEW OF LITERATURE

This chapter provides a basic understanding of both cluster analysis and cluster validation through a review of related literatures. The first section briefly introduces several well-known clustering methods categorized by their characteristics, and we will then take a look at how to validate the resulting cluster solutions and how to determine the optimal number of clusters in the proceeding section. At the end of each section, some claimable issues which are our research motivation will be stated.

2.1 Cluster Analysis

Cluster analysis, which is broadly called 'unsupervised classification' in the machine learning society due to the absence of data labels, is a technique used for combining objects into several groups or clusters such that each group is homogeneous. Suppose that there are n objects in a space and let O_i be the ith object. A set S which consists of the n objects can be expressed as below.

$$S = \{O_1, O_2, \ldots, O_n\} \tag{2.1}$$

Cluster analysis can be described as a task that partitions the set S into mutually exclusive g subsets $C_1, C_2, \ldots, C_g$ such that

$$C_i \cap C_j = \emptyset, 1 \leq i \neq j \leq g \tag{2.2}$$

and

$$\bigcup_{i=1}^{g} C_i = S. \tag{2.3}$$

In the above expression, C_j is the jth resulting cluster. An aggregation of C_j $(j = 1, 2, \ldots, g)$ is called a clustering result or a clustering solution and it can be stated as following.

$$P = \{C_1, C_2, \ldots, C_g\} \tag{2.4}$$

A great number of clustering methods have been proposed for the last half-century. Although there may be antoher reason to categorize the clustering methods in a different way, conventional literatures classify the huge majority of algorithms into two types, *hierarchical* methods and *non-hierarchical* (partitioning) methods (Kaufman and Rousseeuw, 1990).

Hierarchical methods, which do not build a single partition with g clusters but deal with all values of g in a single run, are also subdivided into two kinds of techniques: the *agglomerative* methods and the *divisive* methods. Whereas the former begins with n clusters meaning that each object makes one cluster, which are united step by step until we get one single cluster, the latter starts when all objects are together and in each following step a cluster is split up, until there are n objects. Hence, their constructions of data hierarchy have the opposite direction.

It is important to note that the number of clusters g is predefined in non-hierarchical methods. The procedure sets g number of representative objects or values and each object is then assigned to one of those in a sense that this assignment maximizes the intra-cluster similarity and minimizes the inter-cluster similarity. We typically repeat this procedure by repositioning new representatives and assigning objects again at each step, until the clustering solution converges.

As we might have noticed, now the task is to quantitatively measure the 'similarity' or the 'homogeneity'. In the cluster analysis, the dissimilarity measure rather than similarity measure is generally used. One of the most well-known dissimilarity measures is the distance measure. Assume that one object has p attributes or variables and let x_{ij} $(i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, p)$ denote the jth variate of the ith object. In other words, the data matrix $\mathbf{X} = (x_{ij})$ consists of p features measured on n independent observations. The ith object, therefore, has the coordinate $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^\top$. The Euclidean distance between the ith and jth object is

given by

$$d(O_i, O_j) = d(x_i, x_j) = \sqrt{\sum_{a=1}^{p}(x_{ia} - x_{ja})^2} = \sqrt{(x_i - x_j)^\top (x_i - x_j)}, \tag{2.5}$$

and its generalized version, which is called Minkowski distance, is defined as below.

$$d(x_i, x_j) = \left(\sum_{a=1}^{p} \mid x_{ia} - x_{ja} \mid^m \right)^{\frac{1}{m}} \tag{2.6}$$

The case of $m = 2$ is equivalent to the Euclidean distance as shown above, and the Manhattan distance can be defined by setting $m = 1$. Since the most general selection of $d(x_i, x_j)$ is the squared Euclidean distance in the clustering literature, we also adopt it and the 'distance' thus refers to the squared Euclidean distance from hence in this research.

2.1.1 Hierarchical Methods

Although the hierarchical methods consist of the agglomerative and the divisive as mentioned above, we only focus on understanding the former in this section because of its broad use in many application areas.

The linkage methods which construct a hierarchical cluster tree in an agglomerative manner are based on various measurements of proximity between two groups of objects. In order to explain each of linkage methods we need the following notation.

C_i:	ith cluster
$\mid C_i \mid$:	number of objects in the ith cluster
$d(u, v)$:	distance measure between the uth object and the vth object
$D(C_i, C_j)$:	distance measure between the ith cluster and the jth cluster

1. *Single linkage* uses the smallest distance between objects in the two clusters.

$$D(C_i, C_j) = \min_{u \in C_i, v \in C_j} d(u, v) \tag{2.7}$$

2. *Complete linkage* uses the largest distance between objects in the two clusters.

$$D(C_i, C_j) = \max_{u \in C_i, v \in C_j} d(u, v) \tag{2.8}$$

3. *Average linkage* uses the average distance between all pairs of objects in cluster C_i and cluster C_j.

$$D(C_i, C_j) = \frac{1}{\mid C_i \mid\mid C_j \mid} \sum_{u \in C_i, v \in C_j} d(u, v) \tag{2.9}$$

4. *Centroid linkage* uses the distance between the centroids of the two clusters

$$D(C_i, C_j) = d(c_i, c_j) \tag{2.10}$$

where

$$c_i = (\bar{x}_1^{(i)}, \bar{x}_2^{(i)}, \ldots, \bar{x}_p^{(i)}), \tag{2.11}$$

$$\bar{x}_a^{(i)} = \frac{1}{\mid C_i \mid} \sum_{j \in C_i} x_{ja}, a = 1, \ldots, p. \tag{2.12}$$

5. *Ward linkage* uses the incremental sum of squares; that is, the increase in the total within-cluster sum of squares as a result of joining clusters C_i and C_j. It is given by

$$D(C_i, C_j) = \frac{\mid C_i \mid\mid C_j \mid d(c_i, c_j)^2}{\mid C_i \mid + \mid C_j \mid}. \tag{2.13}$$

The within-cluster sum of squares of a cluster is defined as the sum of squares of the distance between all objects in the cluster and the centroid of the cluster.

The five linkage methods mentioned above use different measurements of proximity between two groups of objects, but follow the same algorithm in order to create a hierarchical cluster tree, as described below.

Linkage Clustering Algorithm

Step 0.	(1)	Start by assigning each object into a cluster, so that if you have n objects, you now have n clusters.
	(2)	$g \leftarrow n$.
Step 1.	(1)	Compute the distance $D(C_i, C_j), 1 \leq i \neq j \leq g$ for all pairs of current resulting clusters.
	(2)	Find the closest pair of clusters and merge them into a single cluster, so that you now have one cluster less.
	(3)	$g \leftarrow g - 1$.
Step 2.	(1)	If $g = 1$ then stop. Otherwise repeat Step 1.

2.1.2 Non-hierarchical Methods

Non-hierarchical method, which is also called partitioning method, predefines the number of clusters g and then assigns each object into an appropriate one of clusters so that the resulting clusters are non-overlapped each other. As indicated in the previous studies (Vinod, 1969; Olafsson et al., 2008), the non-hierarchical clustering can be given as an integer programming formulation by defining the binary decision variable,

$$a_{ij} = \begin{cases} 1, & \text{if the } i\text{th object is assigned to the } j\text{th cluster} \\ 0, & \text{otherwise} \end{cases} . \tag{2.14}$$

Let c_{ij} be some cost of assigning the ith object into the jth cluster then the optimization problem which objective is to minimize the total cost of the assignment can be formulated as below.

$$\begin{aligned} \min \quad & Z = \sum_{i=1}^{n} \sum_{j=1}^{g} c_{ij} a_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{g} a_{ij} = 1, \;\; i = 1, 2, \ldots, n \\ & \sum_{i=1}^{n} a_{ij} \geq 1, \;\; j = 1, 2, \ldots, g \end{aligned} \tag{2.15}$$

In this optimization problem the first set of constraints is to build the non-overlapping clusters by assigning one object into exactly one cluster, and the second one is to prevent an empty

cluster meaning that each cluster should have at least one object. Since the c_{ij} in the objective function is usually not a constant but a function of a_{ij}, this problem, in general, cannot be solved directly. We therefore employ some heuristic-like algorithms in order to find the optimal solution. In this section we briefly mention the two of the most famous partitioning algorithms, k-means method and PAM method that is a k-medoids clustering.

k-means clustering

For the identical usage of the notation, the descriptions of k-means clustering use 'g' instead of 'k' which is mostly used in the k-means literatures, to indicate the number of clusters. Given the input parameter g, k-means clustering works with g centroids. Once the g centroids are determined, the corresponding g clusters are routinely structured by assigning each object to the closest centroid, vice versa, once the g clusters are formed, g new centroids are calculated by taking the mean value of objects for each cluster. This process iterates until its convergence. Hence, this method can be interpreted as finding g centroids denoted by c_j, $j = 1, 2, \ldots, g$, and the below optimization problem can be formulated in order to achieve the goal.

$$\begin{aligned} \min \quad & Z = \sum_{i=1}^{n} \sum_{j=1}^{g} d(x_i, c_j) a_{ij} \\ \text{subject to} \quad & c_j = \frac{\sum_{i=1}^{n} x_i a_{ij}}{\sum_{i=1}^{n} a_{ij}}, \;\; j = 1, 2, \ldots, g \\ & \sum_{j=1}^{g} a_{ij} = 1, \;\; i = 1, 2, \ldots, n \\ & \sum_{i=1}^{n} a_{ij} \geq 1, \;\; j = 1, 2, \ldots, g \end{aligned} \tag{2.16}$$

In order to obtain the optimal solution to the above problem, the following algorithm is widely used, because, again, the problem is a nonlinear integer programming that is thus very difficult to obtain an optimal solution directly.

k-means Clustering Algorithm

Step 0.	(Initialization) Set initial g centroids to random values or arbitrarily chosen g objects in the given data.
Step 1.	(Assignment) (Re)assign each object to the cluster which has the closest centroid to the object.
Step 2.	(Update) Update the cluster centroids, i.e., calculate the mean value of the objects for each cluster.
Step 3.	(Convergence check) If the updated centroids are equal to the centroids in the previous iteration then stop. Otherwise, repeat Step 1 – Step 2.

k-medoids clustering

One drawback of k-means clustering is that it is quite sensitive to outliers, and k-medoids clustering is sometimes used for this reason. Since it uses the most centrally located object (medoid) instead of a mean valued object (centroid) in a cluster, it is less sensitive to outliers than the k-means method.

A medoid is an object in a cluster that has the smallest average or total distance to other objects in the cluster, that is, if we let $x_m^{(i)}$ denote the medoid of the cluster C_i, then

$$\sum_{j \in C_i} d(x_m^{(i)}, x_j^{(i)}) = \min_{k \in C_i} \left(\sum_{j \in C_i} d(x_k^{(i)}, x_j^{(i)}) \right). \tag{2.17}$$

By defining an additional decision variable

$$b_i = \begin{cases} 1, & \text{if the } i\text{th object is a medoid} \\ 0, & \text{otherwise} \end{cases}, \tag{2.18}$$

k-medoids clustering problem is also given by the following optimization formulation (Vinod,

1969; Kaufman and Rousseeuw, 1990).

$$\begin{aligned} \min \quad & Z = \sum_{i=1}^{n}\sum_{j=1}^{n} d(x_i, x_j) a_{ij} \\ \text{subject to} \quad & \sum_{i=1}^{n} a_{ij} = 1, \;\; j = 1, 2, \ldots, n \\ & a_{ij} \leq b_i, \;\; i, j = 1, 2, \ldots, n \\ & \sum_{i=1}^{n} b_i = g \end{aligned} \tag{2.19}$$

PAM (Partitioning Around Medoids) is one of the early-introduced k-medoids algorithms. This method consists of two parts which are 'Build' and 'Swap'. The first one selects the initial representative objects carefully, and the other one repeatedly tries to make a better choice of cluster representatives. This procedure analyzes all possible pairs of objects where one object in each pair is regarded as a representative object (medoid) and the other is not. The description of detailed PAM algorithm is shown in Kaufman and Rousseeuw (1990).

2.1.3 Other Methods

Apart from the classical clustering methods mentioned above, many other approaches have been developed in solving the clustering problem.

A model-based clustering technique using Expectation and Maximization (EM) algorithm is one of the most often used methods (Banfield and Raftery, 1993; Celeux and Govaert, 1995; Fraley and Raftery, 2002). Since this method computes probabilities of cluster memberships based on one or more probability distributions rather than cluster labels, it is called a 'soft' clustering. The EM algorithm begins with initial parameters of g Gaussian mixtures and calculates the likelihood that each object is drawn from a particular density function. The algorithm then updates the parameters in order to maximize the likelihood of the given dataset. This procedure repeats iteratively. Therefore, this method can be said a probabilistic variant of k-means method. Let μ_j and Σ_j be the mean and the covariance of the jth Gaussian density function which will be corresponding to the jth cluster. The probability density function evaluated at x_i is the sum of all densities:

$$P(x_i) = \sum_{j=1}^{g} p_j f_j(x_i \mid \mu_j, \Sigma_j) \tag{2.20}$$

where the *a priori* probability p_j is the fraction of the objects in cluster j such that $\sum p_j = 1$, and

$$f_j(x_i \mid \mu_j, \Sigma_j) = \frac{1}{(2\pi \mid \Sigma_j \mid)^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x_i - \mu_j)^\top \Sigma_j^{-1}(x_i - \mu_j)\right\}. \tag{2.21}$$

By defining the binary variable a_{ij} shown in Equation 2.14, the resulting log-likelihood is

$$L = \sum_{i=1}^{n} \log P(x_i) = \sum_{i=1}^{n}\sum_{j=1}^{g} a_{ij} \log\Big(p_j f_j(x_i \mid \mu_j, \Sigma_j)\Big). \tag{2.22}$$

The E step in EM algorithm for mixture models is given from Bayes' theorem by

$$\hat{a}_{ij} = \frac{\hat{p}_j f_j(x_i \mid \hat{\mu}_j, \hat{\Sigma}_j)}{\sum_{j=1}^{g} \hat{p}_j f_j(x_i \mid \hat{\mu}_j, \hat{\Sigma}_j)}, \tag{2.23}$$

while the M step involves maximizing Equation 2.22 in terms of p_j, μ_j, and Σ_j with a_{ij} fixed at the values calculated in Equation 2.23. Therefore, the resulting estimates have simple closed-form expressions involving the given data and $\hat{a}_{ij}$ as below:

$$\hat{\mu}_j = \frac{\sum_{i=1}^{n} \hat{a}_{ij} x_i}{\sum_{i=1}^{n} \hat{a}_{ij}}, \;\; \hat{\Sigma}_j = \frac{\sum_{i=1}^{n} \hat{a}_{ij}(x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^\top}{\sum_{i=1}^{n} \hat{a}_{ij}}, \;\; \hat{p}_j = \frac{1}{n}\sum_{i=1}^{n} \hat{a}_{ij}. \tag{2.24}$$

Computation of the covariance estimate $\hat{\Sigma}_j$ in fact depends on its parameterization. Details of parameterization by eigenvalue decomposition are shown in Celeux and Govaert (1995).

Due to good performance as a stochastic search procedure, some evolutionary techniques have recently been proposed for cluster representation. Especially, Genetic Algorithm (GA) has been broadly employed in data clustering literatures (Murthy and Chowdhury, 1996; Maulik and Bandyopadhyay, 2000; Garai and Chaudhuri, 2004). Although many different types of string (or chromosome) representation are allowed in order to define cluster memberships, most of GA-based clustering methods have focused on defining g number of cluster centroids, because once g cluster centroids has been decided, the next step, which is the cluster labeling, is a straight-forward step. It could be a binary representation or a sequence of real numbers standing for coordinates of centroids in p-dimensional space. Once a chromosome type is defined, the general GA-based clustering algorithms intrinsically follow the canonical GA procedures and use basic operators of GA. However, it has been found that the GA-based methods could be inefficient if we fail to make a compromise between two conflicting facets; the maintenance of population diversity and the optimality guarantee with fewer changes in

the bits of the present best strings as the GA goes nearer to the optimum. It may be difficult to satisfy those, since we usually cannot reflect the structure of clustering problem into the GA procedures after encoding the original solution to the chromosome representation. In order to overcome such difficulties, a recently developed heuristic optimization method called the Nested Partitions method (Shi and Olafsson, 2008) was employed to solve an optimization problem formulated for clustering in the one recent research (Kim et al., 2009). One advantage of this method is that we can reflect the special structure of clustering problem into the method unlike the GA. Computational time can therefore be reduced especially for large-scale clustering problems. Details are described in Kim et al. (2009).

2.1.4 Objectives of Clustering

As mentioned earlier, the final goal of clustering is to group objects in which the resulting clusters satisfy as much homogeneity within each cluster as well as much separation between the clusters as possible. As for the homogeneity within a cluster, the majority of existing approaches introduced in this chapter adopts the objective of compactness in terms of variation between data objects in the same cluster. For instance, k-means clustering tried to minimize the sum of squared distances between a cluster mean and the other objects in the same cluster, while k-medoids clustering uses a discrete type of cluster representatives, i.e. medoids, instead of cluster centers to minimize the sum of squared distances in a cluster. Average or complete linkage agglomerative approach is a heuristic algorithm that implicitly tries to achieve a similar objective. Since the model-based clustering attempts to fit the best mixture of Gaussian at a fixed number of components and the Gaussian distribution basically assumes the convex-shaped dispersion of data objects, this approach does also pursue a very similar objective. An example having a quite different criterion, namely one-nearest-neighbor-based connectivity, is the single linkage agglomerative algorithm. This approach does consider a more local concept of clustering in its procedure.

Having said that different clustering objectives can lead more than one structure of clusters, we, in this research, would like to introduce a new clustering criterion that is basically based

on a concept of connectivity, which has been received less awareness than the compactness for which the majority of existing approaches strives, and to suggest a new clustering approach to optimize the proposed objective. This objective is similar to what the single linkage method tries to accomplish but different in the respect that it has a more global concept of clustering than the single linkage. Details for this criterion and its implementation as a clustering algorithm will be described in Chapter 3 and Chapter 5 respectively.

2.2 Cluster Validation

The term 'cluster validation' usually refers to the ability of a given method to recover the true clustering structure in a dataset. Once we obtain a clustering solution stated by Equation 2.4, we should be able to assess quantitatively if the clustering algorithm recover the true cluster labels, since the clustering results are not completely reliable in the most of situations. In many validation studies, clustering methods are evaluated on their performance on empirical datasets with or without *a priori* known cluster labels. In the 'with' case, cluster validity can be investigated by *external indices*, while *internal indices* have been used for the latter case where we do not have the true cluster labels.

2.2.1 External Indices

In order to assess the ability of a clustering method for recovering true cluster labels, it is necessary to define a measure of agreement between two partitions; the first partition being the *a priori* known cluster structure of the data, and the second partition resulting from the cluster procedure. In the clustering literature the measurements of agreement between two different partitions have been referred to as the external indices. In this section we will briefly review several well-known indices.

Consider two partitions P^1 and P^2 of n objects: U-group partition $P^1 = \{C_1^1, \ldots, C_U^1\}$ and V-group partition $P^2 = \{C_1^2, \ldots, C_V^2\}$. External indices of agreement between two partitions can be expressed in terms of a confusion matrix as shown in Table 2.1. The cell statistic n_{uv} denotes the number of objects that belong to both clusters C_u^1 and C_v^2, $u = 1, \ldots, U$,

$v = 1, \ldots, V$. Let

$$n_{u\cdot} = \sum_{v=1}^{V} n_{uv} \text{ and } n_{\cdot v} = \sum_{u=1}^{U} n_{uv} \tag{2.25}$$

denote the row and column sum of the confusion matrix respectively.

Table 2.1: Confusion matrix for two partitions of n objects

	C_1^2	C_2^2	$\cdots$	C_V^2	total
C_1^1	n_{11}	n_{12}	$\cdots$	n_{1V}	$n_{1\cdot}$
C_2^1	n_{21}	n_{22}	$\cdots$	n_{2V}	$n_{2\cdot}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
C_U^1	n_{U1}	n_{U2}	$\cdots$	n_{UV}	$n_{U\cdot}$
total	$n_{\cdot 1}$	$n_{\cdot 2}$	$\cdots$	$n_{\cdot V}$	n

Rand Index (Rand, 1971) is simply a matching coefficient. If we newly define four simpler statistics in order to express agreement between two resulting cluster solutions as below:

a: *number of pairs of objects in the same cluster in both* P^1 *and* P^2
b: *number of pairs of objects in the same cluster in* P^1 *but not in the same cluster in* P^2
c: *number of pairs of objects in the same cluster in* P^2 *but not in the same cluster in* P^1
d: *number of pairs of objects in different clusters in both* P^1 *and* P^2

the Rand Index is then given by

$$RI(P^1, P^2) = \frac{a+d}{a+b+c+d}. \tag{2.26}$$

The Rand Index lies between 0 and 1, and the perfect agreement of two partitions is indicated by '1' of RI. Equation 2.26 is equivalent to the following equation expressed by the cell statistics in Table 2.1.

$$RI(P^1, P^2) = \frac{1 + \left[\sum_{u=1}^{U} \sum_{v=1}^{V} n_{uv}^2 - \frac{1}{2} \left(\sum_{u=1}^{U} n_{u\cdot}^2 + \sum_{v=1}^{V} n_{\cdot v}^2 \right) \right]}{\binom{n}{2}} \tag{2.27}$$

A problem with the Rand Index is that the expected value of the Rand Index of two random partitions does not take the value of zero. The Adjusted Rand Index proposed by Hubert and Arabie (1985) follows the form of adjustment in Equation 2.28 so that it could be bounded

above by '1' and take the value of '0' when the observed index equals to the expected index.

$$\frac{\text{observed index} - \text{expected index}}{\text{maximum index} - \text{expected index}} \tag{2.28}$$

The numerator of Equation 2.28 represents the observed improvement over chance and the denominator says the maximum possible improvement over chance. The maximum index is always unity. By following the form in Equation 2.28, the Adjusted Rand Index is given by

$$ARI(P^1, P^2) = \frac{\sum_{u=1}^{U}\sum_{v=1}^{V}\binom{n_{uv}}{2} - \left[\sum_{u=1}^{U}\binom{n_{u\cdot}}{2}\sum_{v=1}^{V}\binom{n_{\cdot v}}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_{u=1}^{U}\binom{n_{u\cdot}}{2} + \sum_{v=1}^{V}\binom{n_{\cdot v}}{2}\right] - \left[\sum_{u=1}^{U}\binom{n_{u\cdot}}{2}\sum_{v=1}^{V}\binom{n_{\cdot v}}{2}\right]/\binom{n}{2}} \tag{2.29}$$

Jaccard's coefficient (Jain and Dubes, 1988) also shows a similarity between two partitions as the Rand Index does, but does not take account into 'conjoint absence' as shown in Equation 2.30. Hence, this coefficient measures the portion of pairs in the same cluster in both P^1 and P^2 from those in the same cluster either in P^1 or P^2. Another expression in terms of n_{uv} is also given in Equation 2.30. For more information about relationship between $\{a, b, c, d\}$ statistics and n_{uv} statistics, see Collins and Dent (1988).

$$JC(P^1, P^2) = \frac{a}{a+b+c} = \frac{\sum_{u=1}^{U}\sum_{v=1}^{V} n_{uv}^2 - n}{\sum_{u=1}^{U} n_{u\cdot}^2 + \sum_{v=1}^{V} n_{\cdot v}^2 - \sum_{u=1}^{U}\sum_{v=1}^{V} n_{uv}^2 - n} \tag{2.30}$$

As in the Rand Index, the value of this coefficient lies between '0' and '1'. If the value is close to '1', high agreement between two partitions appears.

The Fowlkes and Mallows Index (Fowlkes and Mallows, 1983) is a geometric mean of two measurements; the probability that a pair of objects in the same cluster in P^1 belong to the same cluster in P^2, and the probability that a pair of objects in the same cluster in P^2 belong to the same cluster in P^1. This measure basically comes from the Wallace Index (Wallace, 1983). Since the Wallace Index for two partitions is asymmetric, the Fowlkes and Mallows

Index takes the geometric mean of the asymmetric Wallace Indices as shown in Equation 2.31.

$$FM(P^1, P^2) = \left(\frac{a}{a+b} \cdot \frac{a}{a+c}\right)^{\frac{1}{2}} = \frac{\sum_{u=1}^{U}\sum_{v=1}^{V} n_{uv}^2 - n}{2\left[\sum_{u=1}^{U}\binom{n_{u\cdot}}{2}\sum_{v=1}^{V}\binom{n_{\cdot v}}{2}\right]^{\frac{1}{2}}} \tag{2.31}$$

This index also lies between '0' and '1' and a higher value means a higher agreement between two partitions.

It is important to note that the significance of an observed external index is usually assessed under assumption that the two partitions P^1 and P^2 to be compared are independent. Let one of the two partitions be the true cluster labels and the other partition be the resulting cluster solution. Based on the above indices we can quantitatively measure the degree of successfulness of the employed clustering method for recovering the true clustering structure in a dataset.

2.2.2 Internal Indices

While the assessment of a resulting cluster solution using external indices is straightforward because we have a clear external criterion that is *a priori* known cluster labels, the validation of resulting cluster structures without true cluster labels is a more difficult task. Let us recall the definition of cluster analysis or clustering. Because of several different definitions of cluster by researchers' different point of views, a number of definitions of cluster analysis have been observed in the literature. One point of view regards clustering as the segmentation of a heterogeneous group into a number of more homogeneous sub-groups such that the resulting sub-groups maximize intra-cluster similarity and minimize inter-cluster similarity. The majority of existing researches attempt to look for the clustering structure where summary statistics of interest are optimal, and these statistics are typically the within-cluster sum of squares $tr\mathbf{W}_g$ where

$$\mathbf{W}_g = \sum_{j=1}^{g}\sum_{i\in C_j}(x_i - c_j)(x_i - c_j)^\top, \tag{2.32}$$

and the between-cluster sum of squares $tr\mathbf{B}_g$ where $\mathbf{B}_g$ is given as below:

$$\mathbf{B}_g = \sum_{j=1}^{g}|C_j|(c_j - c)(c_j - c)^\top, \quad c = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{2.33}$$

for the representation of intra-cluster similarity and inter-cluster similarity respectively. A function of those statistics is usually referred to as an internal index that is to estimate the number of clusters in a dataset. Estimation of the correct number of clusters is regarded as one challenging problem in cluster analysis. A myriad number of approaches has been proposed for estimating the optimal number of clusters g^* and they may fall into one of the following categories that are summary-statistic-based methods, resampling methods, density-based methods, and the other methods. The two main streams in this line of research are the first two categories.

One earliest method was proposed by Calinski and Harabasz (1974). Let us denote $CH(g)$ the index computed by their method. It decides the optimal number of clusters by

$$g^* = \underset{g \geq 2}{\operatorname{argmax}} \left[CH(g) = \frac{tr\mathbf{B}_g/(g-1)}{tr\mathbf{W}_g/(n-g)} \right]. \tag{2.34}$$

Because it searches for the maximum ratio between inter-cluster similarity and intra-cluster similarity, it is also called Variance Ratio Criterion (VRC).

Hartigan (1975) looked for the greatest improvement on intra-cluster similarity when one more number of clusters is allowed, and suggested the following index.

$$g^* = \underset{g \geq 1}{\operatorname{argmin}} \left[Hart(g) = \left(\frac{tr\mathbf{W}_g}{tr\mathbf{W}_{g+1}} - 1 \right) / (n-g-1) \right] \tag{2.35}$$

Instead of comparing the intra-cluster similarity at g number of clusters to that at $g+1$ number of clusters, Krzanowski and Lai (1985) considered three consecutive values of within-cluster sum of squares at $g-1$, g, and $g+1$. They believed that if the difference between the current within-cluster sum of squares and the previous within-cluster sum of squares are big enough when compared to the difference between the current value and the next value, more partitioning is meaningless and the optimal number of clusters may have to be decided at this point. The equation is defined as below:

$$g^* = \underset{g \geq 2}{\operatorname{argmax}} \left[KL(g) = \left| \frac{DIFF(g)}{DIFF(g+1)} \right| \right], \tag{2.36}$$

where

$$DIFF(g) = (g-1)^{\frac{2}{p}} tr\mathbf{W}_{g-1} - g^{\frac{2}{p}} tr\mathbf{W}_g. \tag{2.37}$$

The Average Silhouette Width proposed with the clustering method PAM (Kaufman and Rousseeuw, 1990) is also built on the comparison of the inter-cluster similarity with the intra-cluster similarity, although a bit different metrics are used. For the ith object, let $a(i)$ be the average distance to other objects in its cluster and $b(i)$ be the smallest $d(i, C)$ where $d(i, C)$ is the average distance to all objects in any other cluster C. The silhouette width of the ith object is given by

$$Sil(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \tag{2.38}$$

and the overall average silhouette width is simply the mean value of $Sil(i)$. Intuitively, objects with large silhouette width values are well-clustered, whereas those with small $Sil(i)$ values are likely to lie between clusters. Kaufman and Rousseeuw suggested an estimated number of clusters g^* by observing the largest average silhouette width as shown in Equation 2.39.

$$g^* = \underset{g \geq 2}{\operatorname{argmax}} \left[\overline{Sil}(g) = \frac{1}{n} \sum_{i=1}^{n} Sil(i) \right] \tag{2.39}$$

One recent approach for estimating number of clusters called '*Gap* statistic' was proposed by Tibshirani et al. (2001). This method compares an observed internal index, such as the within-cluster sum of squares, to its expectation under a reference null distribution. Its procedure is as follows. For each number of clusters $g \geq 1$, compute the within-cluster sum of squares $tr\mathbf{W}_g$. Generate B number of reference datasets under the null distribution (They generated each reference feature uniformly in Tibshirani et al.) and apply the clustering algorithm to each reference dataset for calculating a set of within-cluster sum of squares that are $tr\mathbf{W}_g^1, tr\mathbf{W}_g^2, \ldots, tr\mathbf{W}_g^B$. Based on the computed set of sum of squares we calculate the *Gap* statistic

$$Gap(g) = \frac{1}{B} \sum_{b=1}^{B} \log tr\mathbf{W}_g^b - \log tr\mathbf{W}_g \tag{2.40}$$

and the standard deviation $SD(g)$ of $\log tr\mathbf{W}_g^b$, $b = 1, 2, \ldots, B$, that is

$$SD(g) = \left[\frac{1}{B} \sum_{b=1}^{B} \left(\log tr\mathbf{W}_g^b - \left(\frac{1}{B} \sum_{b=1}^{B} \log tr\mathbf{W}_g^b \right) \right)^2 \right]^{\frac{1}{2}} \tag{2.41}$$

Let $S(g) = SD(g)\sqrt{1 + 1/B}$. The estimated number of clusters via *Gap* statistic is chosen by

$$g^* = \text{smallest } \ g \ \text{ such that } \ Gap(g) \geq Gap(g+1) - S(g). \tag{2.42}$$

We are be able to say that the above five methods belong to the summary-statistic-based approaches, since they first compute one statistic or more which summarizes the given clustering result and then decide the number of clusters by observing its trend at varying number of clusters.

Clest, a prediction-based resampling method was proposed by Dudoit and Fridlyand (2002). In order to estimate the number of clusters in a dataset this method repeatedly and randomly divide the original data set into two non-overlapping sets, a learning set and a test set. At each iteration and each number of clusters g, a clustering solution of the learning set is obtained and a prediction model is built using the cluster labels from the clustering. The prediction model is then applied to the test set and the predicted labels are compared to those produced by applying the clustering procedure to the test set, using one of the external indices mentioned in the previous section. The number of clusters is estimated by comparing the observed similarity statistic for each g to its expected value under a suitable null distribution. In their research they also used the uniform distribution. In order to have the expected similarity the procedure that generates the agreement statistic between two sets of cluster labels from both the clustering method and the prediction model is repeated B times. In the respect of obtaining an expected value of a statistic, this method can be said to take the framework of *Gap* statistic method, but the main difference is that an external index for obtaining consistency is used. The entire *Clest* procedure is shown in Dudoit and Fridlyand (2002).

Ben-Hur et al. (2002) proposed another resampling-based estimating method (BH). This method also exploits measurements of stability of clustering solutions obtained by perturbing the given data set, and the stability is characterized by the distribution of pair-wise similarities between clustering solutions obtained from two sub-samples of the data. The pair-wise similarities are also acquired from one of the external indices. For each number of clusters g two sub-samples are generated from the given data set at a given sampling rate, and an identical clustering method is then applied to both sub-samples in order to obtain two partitions. Measuring the degree of agreement between two partitions using a suitable external criterion gives us a clue in estimating the number of clusters. The intuition behind this approach is that,

at the optimal number of clusters, many sub-samples have similar clustering results and the distribution of similarities will be concentrated close to 1. One common thing between *BH* and *Clest* is that they use external indices to measure consistency of clustering solutions. *Clest*, however, generates predicted class labels from a classification model for its external criterion, whereas *BH* compares cluster labels from resampled datasets.

Kernel Density Estimation, which is a non-parametric density estimation method, has been used for determining the number of clusters (Nakamura and Kehtarnavaz, 1998; Herbin et al., 2001). A basic idea of this approach is that the modes of the probability density function of a dataset leads to the construction of influence zones that are intrinsically related to the number of clusters. Because the number of clusters depends on the scale factor which is the window width, a careful selection of the factor is needed. Moreover, due to difficulty in density estimation for high dimensional data set, either dimensionality reduction technique should be employed as a preprocessing step, or its application may be restricted to low dimensional cases such as image segmentation.

There have been several other estimating methods that were combined with specific clustering algorithms. Kothari and Pitts (1999) modified the cost (objective) function in k-means clustering. As can be seen in Equation 2.16, the cost function tries to distribute the cluster centers so as to minimize the within-cluster sum of squares. They inserted an additional term, the regularization term that requires one more objective that the sum of squared distances from a cluster to its nearby cluster should be minimized. Hence, the concept of $CH(g)$ index is embedded to the objective function in the k-means clustering framework. A similar research which is called x-means clustering is shown in Pelleg and Moore (2000). After each run of the traditional k-means algorithm, it decides which subset of current centroids should split themselves in order to better fit the given dataset by computing Bayesian Information Criterion. Lukashin and Fuchs (2001) employed a heuristic optimization approach, which is simulated annealing, not only to minimize the within-cluster sum of squares but also to optimize the cut-off distance value that is critical to decide the number of clusters. Salvador and Chan (2004) used a regression fit on the space of the number of clusters versus the within-cluster sum of

squares from hierarchical clustering methods to find the best number, and Sun et al. (2004) embedded a cluster validity index into the existing Fuzzy C-means clustering method.

As mentioned above, the majority of previous research has focused on how much compact clusters can be obtained by varying the number of clusters with the measurement the within-cluster sum of squares. They decide the number of clusters when the measurement of within-cluster sum of squares is not improved significantly although we increase the number of clusters by one. This mechanism may like to assume that the current structure of clustering consists of a number of clusters where each cluster is convex-shaped, e.g. spherically shaped or ellipsoidal. What they want to assert under this assumption is that it is meaningless to divide one of such clusters into two groups and now we thus have the most promising number of clusters. Because of this assumption, however, they may fail to identify the correct number of clusters for a dataset that has more complicated cluster structure, e.g. a structure having not only convex-shaped but also arbitrarily non-convex-shaped clusters, since sometimes the within-cluster sum of squares is not having an important effect for such a structure of clustering. Here is our research motivation. Measuring the quality of clusters using the within-cluster sum of squares for some arbitrarily shaped clusters means that we already fail to figure out the organization or the hierarchy of clusters. Therefore, selecting a best number from several candidate numbers based on this measurement of cluster quality may be unsuccessful. By indicating these negative aspects that the most of previous approaches has, we would like to propose a more general type of estimating method. This research thus starts with suggesting a new goodness measurement of clustering that may work for not only convex-shaped but also arbitrarily non-convex-shaped clusters.

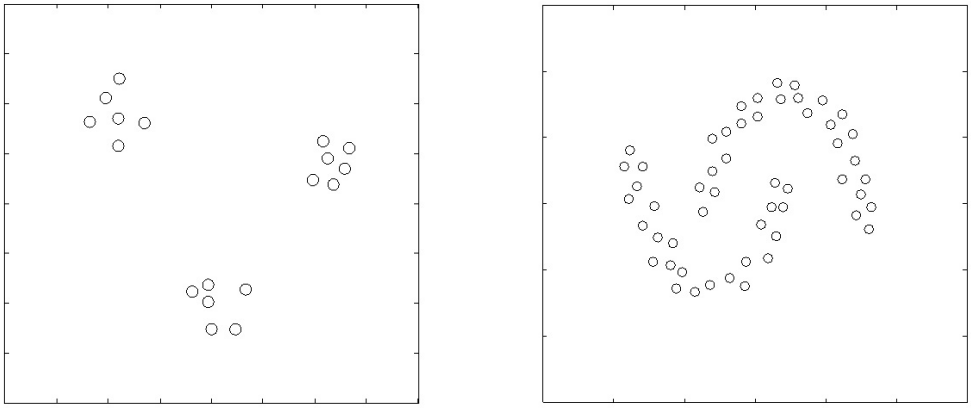
CHAPTER 3. MEASURING CLUSTER QUALITY

This chapter is an extended content of the section 2.2.2 in the respect of that we mention about evaluating goodness of clustering. Let us recall the definition of clustering. Clustering is commonly defined as a task of finding natural groups in a dataset such that the objects in the same group are more similar than those in the other groups. Since there is no completely agreed definition of the natural groups and the similarity between objects or groups yet, it is generally acknowledged that data clustering is a challenging task. Discovery on definitions of natural groups and similarity between objects or groups is the problem that we may have to solve perpetually in the clustering field.

Distance, especially the squared Euclidean distance for objects placed in a continuous space, has mostly been used in order to define the similarity (in fact, dissimilarity) between objects, and the within-cluster and/or between-cluster sum of squares have also been contributing on the definition of the natural group, as mentioned in chapter 2. The measure of within-cluster sum of squares represents the amount of how compact the objects in the same cluster are. Although it is not doubtful that more compact is the better, meaning that the degree of compactness can measure the goodness of clustering, its inability to detect clusters with diverse shapes and sizes is a fundamental limitation. More trials to overcome this limitation are needed for better cluster analysis, and our research starts from this point.

Figure 3.1 shows two simple examples where each has a different type of clusters. Even though the size of each example is very small, they are good enough to show that it will be difficult to have one single measure of cluster quality that works for both of types. We may need to use different types of measurement for different types of cluster structures. In order to quantify the cluster quality, two measurement metrics will be introduced in this chapter. The

first is surely the compactness being considered as the most important measure in a numerous number of previous research studies and the second one is the connectivity that has been received less concern than the compactness, but that we believe as an important factor.



(a) 3 clusters (b) 2 clusters

Figure 3.1: Two different types of clusters in two dimensions

3.1 Compactness

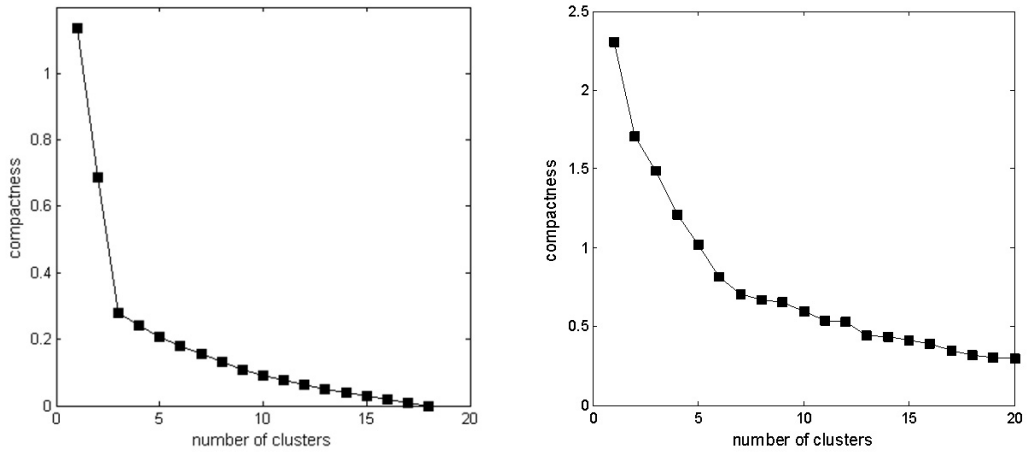
As the most of previous research studies have been doing, we would also like to define the compactness using the within-cluster sum of squares around centroids. Again, suppose that we have a partition P consisting of g number of clusters as shown in Equation 2.4 under assumption that the data matrix $\mathbf{X} = (x_{ij})$ with p features measured on n independent observations is available. Then, the compactness is given by

$$\mathcal{CP}(P) = tr\left[\sum_{j=1}^{g}\sum_{i\in C_j}(x_i - c_j)(x_i - c_j)^\top\right] = tr\mathbf{W}_g = \sum_{j=1}^{g}\frac{1}{2|C_j|}\left(\sum_{i,i^{'}\in C_j, i\neq i^{'}} d(i,i^{'})\right). \quad (3.1)$$

where c_j is defined in Equation 2.11. As shown in the above equation we take the trace of within-cluster cross product matrix as a scalar measure. In some literature the determinant instead of the trace has been used to obtain the scalar measure of the scatter matrix. However, since the scatter matrix will be singular if the number of clusters is greater than or equal to

the dimensionality, the determinant option is obviously a poor choice in such a case. Note that this quantity should be minimized for a good clustering result.

It is important to note its several properties that are related to the number of clusters. At first, the compactness tends to decrease as the number of clusters increases. Second, the maximum value of $\mathcal{CP}(P)$ obviously occurs when the partition P has a single cluster, saying that the partition P is the original dataset itself. In reverse, if the partition P consists of n singleton clusters, meaning that each object forms its own cluster, the $\mathcal{CP}(P)$ has its minimum value that equals to '0'. Figure 3.2 illustrates a trend of data compactness at varying number of clusters for two-dimensional data examples shown in Figure 3.1.



(a) (b)

Figure 3.2: Trend of compactness at varying number of clusters

The first example consists of 18 instances forming three clusters as shown in Figure 3.1a. At each number of clusters the hierarchical Ward's clustering method was employed in order to obtain each partition, and the compactness was then calculated using Equation 3.1. Figure 3.2a shows its trend. The maximum value of compactness is at one number of clusters and its minimum value is zero at 18 clusters. Because of the shown special structure of clusters of the second example consisting of 58 objects, we applied the single linkage hierarchical algorithm to this dataset at each number of clusters. A set of compactness was obtained by the same equation, and its trend was graphed in Figure 3.2b. The same properties, the maximum

compactness at one cluster and the decreasing trend at increasing number of clusters, appear in this figure. Due to space limitation we truncated the graph at 20 clusters but observed the zero compactness at 58 clusters. Through these two simple examples, we have confirmed the mentioned properties of compactness.

One more imperative aspect from Figure 3.2a is that the compactness steeply decreases from one to three clusters (A true number of clusters in this example is regarded as three.), while it slopes gently downwards toward the zero. This characteristic has been used to decide the number of clusters in many clustering studies. Such a characteristic, however, does not appear in Figure 3.2b, so it could be risky to determine the number of clusters from the trend of compactness for the similar or same type of dataset with the second example. We will discuss about this in Chapter 4 in more detail.

3.2 Connectivity

We believe that a new goodness measurement that will be proposed in this section is simple but very practical. A fundamental intuition behind this measure is that we will give some penalties for disconnection of objects which are actually supposed to be grouped together but belonged to different clusters.

3.2.1 Nearest Neighbor Consistency

Before defining the measure of connectivity we need to declare two concepts in data clustering that is based on k-Nearest Neighbor (k-NN) classification technique. k-NN classification proposed about 50 years ago basically assumes that data objects in each class are distributed consistently. Under this assumption this classification technique classifies a new object by the majority voting on class labels of its k nearest neighbors. The new object is thus classified to the most consistent class with its neighbors. Motivated by this principle Ding and He (2004) proposed a new concept of cluster nearest neighbor consistency as stated below.

Cluster k-NN Consistency

For any data object in a cluster, its k-Nearest Neighbors should also be in the same cluster.

Again, data clustering partitions a given data into sub-groups such that the objects in the same cluster are very similar each other and those in different clusters are different. If the objects in the same cluster are very similar each other, then it is likely that the nearest neighbors of any object in a cluster are also in the same cluster. Therefore, the cluster k-NN consistency has the same purpose with data clustering.

Another concept proposed in Ding and He (2004) is the 'k-Mutual Nearest Neighbor consistency'. If an object i's nearest neighbor is the object j and the object j's nearest neighbor is the object i, then we say that the object i's mutual nearest neighbor is the object j and the object j's mutual nearest neighbor is the object i. In general, if we assume that the object i is in the pth nearest neighbor and the object j is in the qth nearest neighbor and k is the maximum value between p and q, we say that the object i is in the k-mutual nearest neighbors of the object j and vice versa. According to this concept, the cluster k-Mutual Nearest Neighbor Consistency can be stated as below.

Cluster k-MN Consistency

For any data object in a cluster, its k-Mutual Nearest Neighbors should also be in the same cluster.

The concept of k-MN consistency is stronger and more interactive among objects in a dataset than the k-NN consistency concept, and it is a better representation of the natural grouping in the definition of clustering. If we can create a measure that includes both k-NN and k-MN consistency concepts, then we believe that it will greatly contribute on a good cluster analysis.

3.2.2 Proposed Connectivity Measure

In addition to the compactness measurement, we introduce another one through the k-NN consistency concept and the k-MN consistency concept as well, in order to evaluate cluster quality. Let $\sigma_i(k)$ denote a set of k nearest neighbors of the object i as below:

$$\sigma_i(k) = \{j | d(i,j) \leq d(i,k), k\text{th nearest neighbor of object } i\} \tag{3.2}$$

The connectivity at the given partition $P = \{C_1, C_2, \ldots, C_r, \ldots, C_g\}$ can be defined as a penalty for violation of both k-NN and k-MN consistency. It is given by

$$\mathcal{CN}(P) = \sum_{r} \sum_{i \in C_r} \sum_{j \notin C_r} \left(b_{ij}^{(1)} \cdot \frac{1}{d(i,j)} + b_{ij}^{(2)} \cdot \frac{1}{d(i,j)} \right), \tag{3.3}$$

where,

$$b_{ij}^{(1)} = \begin{cases} 1, & \text{if } j \in \sigma_i(k) \\ 0, & \text{otherwise} \end{cases} \text{ and } b_{ij}^{(2)} = \begin{cases} 1, & \text{if } i \in \sigma_j(k) \\ 0, & \text{otherwise} \end{cases}. \tag{3.4}$$

This metric measures the amount of penalties for the objects that violate the cluster k-NN consistency and the cluster k-MN consistency. If the neighbor object j of the object i does not belong to the same cluster, then its penalty, which is an inverse of the distance between the object i and j, will be imposed. Conversely, if the neighbor object i of the object j is not grouped in the cluster to which the object j belongs, then the same amount of penalty will also be imposed. Either the first term or the second term in the parenthesis in Equation 3.3 forces either the object j or the object i to satisfy the cluster k-NN consistency. Assume that the object i and j are in the relationship of k mutual nearest neighbors but do not belong to the same cluster. Then, both of the indicator variables, $b_{ij}^{(1)}$ and $b_{ij}^{(2)}$, will be one, and so the penalty will be paid twice because they do not hold the cluster k-MN consistency. It means that we impose more penalty for violating the k-MN consistency than violating the k-NN consistency. Since the penalty term is an inverse distance between two objects, this measurement gives more penalty for closer neighbor objects than further neighbor objects. Note that although the proposed measurement is a penalty function, we named it as 'connectivity' rather than 'disconnectivity'. It is not difficult to notice that this quantity, similarly to the compactness, should be minimized for a good clustering solution.

Figure 3.3 shows a conceptual illustration of the connectivity measure. All objects in the figure are supposed to be in the same cluster. Let us separate them into two different clusters by the bold curve, then the connectivity measure will be increased due to the penalties on the grey-shaded objects.

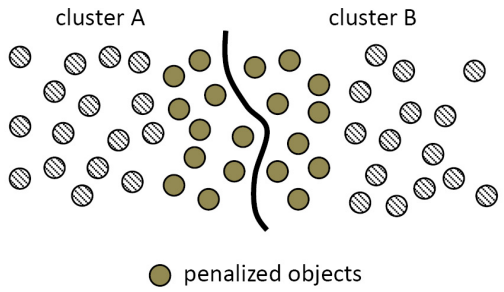
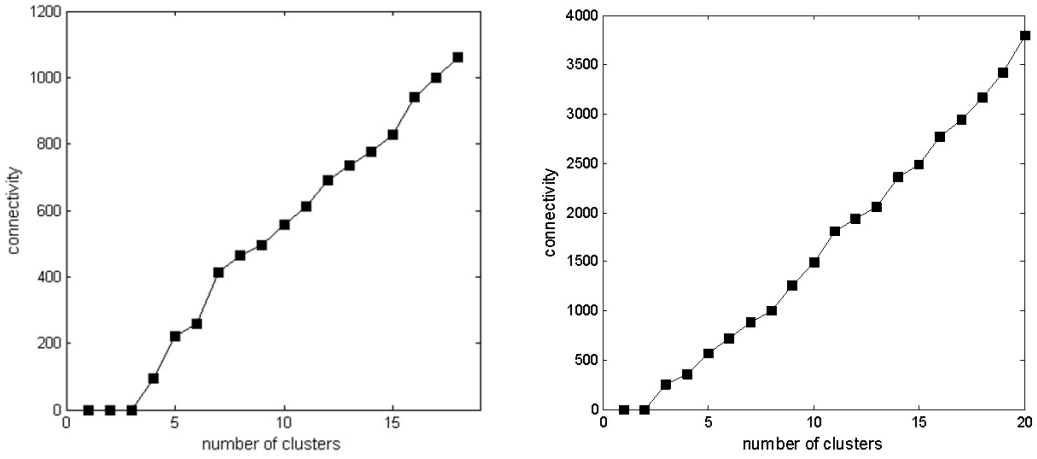


Figure 3.3: Illustration of penalized objects

Recall the example of 18 objects shown in Figure 3.1a. We employed the same settings of clustering procedure, applying Ward's method at different number of clusters from 1 to 18. We then computed the connectivity measure using Equation 3.3 for each partition. Figure 3.4a shows the trend of the connectivity measure.



(a) (b)

Figure 3.4: Trend of connectivity at varying number of clusters

At a fixed number of nearest neighbors of an object the maximum value of connectivity is already decided, and it occurs when there are n singleton clusters. In reverse, its minimum value which is zero will obviously be happened on a single cluster of the dataset regardless of the value of k. Also, the connectivity value will be increased by the penalized objects as we partition a data into more and more sub-groups. Figure 3.4a shows those properties of the connectivity through a simple example. This metric also gives us a hopeful indication for estimating the optimal number of clusters as the compactness does. As can be seen in Figure 3.4a, the connectivity value stays at zero, which is the minimum value, from 1 to 3 cluster number, whereas it increases from 4 cluster number, since the partitions at 1 to 3 cluster number hold the cluster k-NN and k-MN consistency. With the single linkage clustering method based on the dataset in Figure 3.1b, we also obtained Figure 3.4b and it shows a very similar trend to Figure 3.4a. As for the matter of determining the number of clusters, this graph also indicates the zero connectivity at 1 and 2 clusters and the connectivity value increases as the number of clusters increases. It means that the connectivity may be able to select a good number of clusters from data and this is a better symptom than Figure 3.2b. This also implies the potential of this metric that we may be able to desirably handle datasets which look like not only Figure 3.1a but also Figure 3.1b containing more complicated structure of clusters.

An algorithmic procedure of Equation 3.3 is as follows. For each object in a dataset, we first look for its k-NN objects. After that, we confirm whether each nearest neighbor belongs to the same cluster or not, and then penalize some neighbor objects that do not belong to the same cluster with the object under consideration. Since we sequentially consider all the objects in the dataset for this procedure, we can implement Equation 3.3 that verifies both the cluster k-NN consistency and the cluster k-MN consistency.

The goodness of clustering is typically refereed by a clustering algorithm that generates relative clusters and as an internal assessment criterion as well. A more general goodness function that can be adopted for a large range of clustering algorithms will be necessary. One or more well-established goodness functions of clustering can greatly affect not only for better clustering but also for determining the optimal number of clusters in a dataset. In other words,

if such goodness functions are available, we may be able to discover the true cluster structure in data, although some clusters in the dataset are arbitrarily shaped. Two measurements, the compactness and the connectivity, introduced in this chapter will be used for constructing a good clustering method and estimating the correct number of clusters as well. They will be mentioned in Chapter 4 and Chapter 5 correspondingly.

CHAPTER 4. IDENTIFYING THE NUMBER OF NATURAL CLUSTERS IN DATA

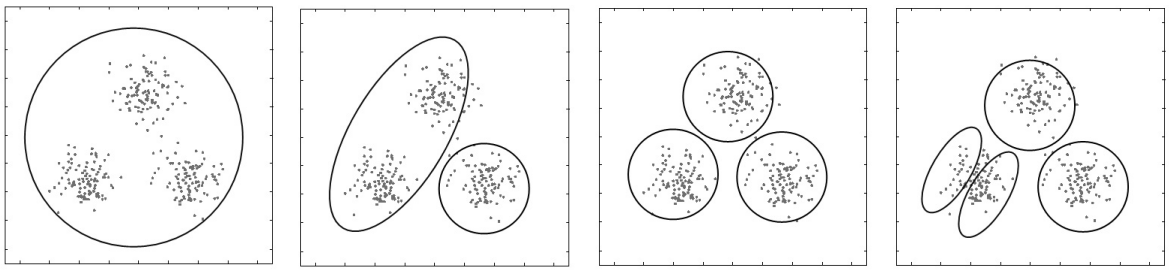
4.1 Introduction

Identifying the correct number of clusters from a dataset by examining the partitions that a clustering algorithm generates has been known as one of the most challenging tasks in cluster analysis. As mentioned earlier people generally use the internal indices in order to examine the clustering solutions, and various approaches to this problem have been suggested over the years. In Section 2.2.2 we have briefly reviewed several prominent estimating methods. Although each of those seems to pursue different actions, most of them can be characterized as an approach that looks for the location of 'elbow' on the plot of compactness versus the number of clusters. It is expected for them that, in a typical plot, the compactness decreases monotonically as the number of clusters increases, but from some point onwards the decrease flattens remarkably. For example, the 3 cluster number in Figure 3.2a is the typical elbow point, and the correct number of clusters in this example is truly 3. The $Hart(g)$, $KL(g)$, and $Gap(g)$ were invented to point out the location of elbow, 3 in this example, by detecting the minimum ratio of the 3rd compactness to the 4th compactness, the maximum ratio of the difference between 2nd and 3rd compactness to the difference between 3rd and 4th compactness, and the biggest difference between the expected and observed compactness that occurs at 3 respectively. The $CH(g)$ and $\overline{Sil}(g)$ follow the intrinsic definition of cluster analysis which is homogeneity within a cluster and heterogeneity between clusters by utilizing the compactness with the separation concept.

However, we believe that none of those may be able to detect the correct number of clusters for a dataset that includes non-spherical clusters, since the measurements for evaluating cluster quality consider the amount of compactness only in the most cases. The *Clest* proposed by

Dudoit and Fridlyand (2002) also falls into the same situation, because the employed classifier in their approach was the discriminant analysis and this classification technique works under assumption that the data is Gaussian-distributed, meaning that the classifier does not work properly for a non-Gaussian dataset.

In this chapter we propose a new estimating method to detect the correct number of clusters for a dataset that has not only spherically shaped clusters but also non-spherically distributed objects in a cluster. In order to achieve our purpose we incorporate both the compactness and the connectivity in our method. A resampling-like framework that can be seen as a sort of meta-learning will help to give more reliability to the chosen number of clusters. Details of the proposed procedure are described in Section 4.2 and we experiment with the proposed method based on several synthetic data-generating models in Section 4.3 in order to evaluate the performance of the new method.



(a) 1 cluster (b) 2 cluster (c) 3 cluster (d) 4 cluster

Figure 4.1: Desirable clustering behavior at increasing number of clusters

An assumption over the task of determining the correct number of clusters is that we always occupy a good clustering method that either generates a desirable hierarchy of data clusters or preserves an identical characteristic for each cluster. For example, since k-means clustering tends to generate convex-shaped clusters, although we increase the number of clusters from g to $g+1$, any cluster generated at either g or $g+1$ should be preserved to have the spherical shape. Figure 4.1 shows what is the cluster hierarchy at varying number of clusters that we expect. If the clustering method that generates the results in the figure conducts the case of 5 clusters, then one of two circles in the last plot should be split into two sub-groups as described by

two ellipsoids. This assumption should be made on any arbitrarily distributed dataset. Again, our objective is to detect the optimal number of clusters under a good clustering algorithm's working, as the other estimating approaches that have been suggested so far have the same purpose.

4.2 Proposed Estimating Method

4.2.1 Location of Elbow by Compactness and Connectivity

Our estimating method is on the line of the studies of the elbow phenomenon. However, one major difference between the proposed method and the previously developed approaches is that we do not assume the shape of data clusters such as Gaussian-distributed objects. We thus look for the location of elbow not only from the plot of the compactness but also from the connectivity plot.

As for the compactness measure we adopt Krzanowski and Lai's (1985) idea in order to select the elbow point. Therefore, we search for the maximum ratio between two consecutive differences of compactness from the conducted compactness plot. From the given values of $\mathcal{CP}$'s, the compactness measure chooses the promising number of clusters by

$$g^*(\mathcal{CP}) = \underset{g \geq 2}{\operatorname{argmax}} \left| \frac{(g-1)^{\frac{2}{p}}\mathcal{CP}(g-1) - g^{\frac{2}{p}}\mathcal{CP}(g)}{g^{\frac{2}{p}}\mathcal{CP}(g) - (g+1)^{\frac{2}{p}}\mathcal{CP}(g+1)} \right|. \tag{4.1}$$

As discussed in Section 3.2.2 and shown in Figure 3.4, the connectivity plot may also have the elbow point on its plot, but the bended direction is opposite to the compactness plot due to its increment characteristic at increasing number of clusters. It is very likely that the optimal number of clusters is found at the location where the connectivity is significantly increased which is the elbow, since the penalties are given to the objects that is supposed be grouped together. Therefore we would like to select the most promising number of clusters from the connectivity plot by

$$g^*(\mathcal{CN}) = \underset{g \geq 1}{\operatorname{argmax}} \left(\theta_g(\mathcal{CN}) \right), \tag{4.2}$$

where

$$\theta_g(\mathcal{CN}) = \begin{cases} \operatorname{atan}\left(\mathcal{CN}(g+1) - \mathcal{CN}(g)\right), & \text{if } g = 1 \\ \operatorname{atan}\left(\mathcal{CN}(g+1) - \mathcal{CN}(g)\right) - \operatorname{atan}\left(\mathcal{CN}(g) - \mathcal{CN}(g-1)\right), & \text{otherwise} \end{cases} . \quad (4.3)$$

Figure 4.2 illustrates θ_g's in a hypothetical connectivity plot which looks like Figure 3.4. The $\theta_g = 0$ means that the cluster k-NN and k-MN consistency are satisfied. As the number of clusters increases, it is more difficult to hold those consistency conditions. Hence, the basic idea of Equation 4.2 expects that the location we are looking for is the point right before breaking the consistency condition, or increasing the sum of penalties significantly. We believe that this idea works with a good clustering algorithm that can consistently generate desirable clustering solutions at varying number of clusters, as mentioned at the end of the previous section. The desirable clustering results should also be guaranteed for any arbitrarily shaped dataset. Again, it is important to have a good clustering algorithm in order to detect the true number of clusters. Let us take a look at the difference between θ_4 and $\theta_4^{'}$ in the figure. The intention that we measure θ_4 instead of $\theta_4^{'}$ is to ignore previously cumulated sum of penalties. We can measure the additionally increased amount of penalties by observing the degree of θ_4. It means that the additional increment can have a negative sign so as θ_5 and θ_6 do, and we do not expect that such negative values of angles can come up with a correct number of clusters.

The geometrical meaning of g^* in Equation 4.1 is that although we split one cluster into two parts there is no significant improvement on compactness, because the objects forming that cluster are almost evenly distributed over the space that the cluster occupies. On the other hand, the g^* from the connectivity says that if we divide one cluster, that is really supposed to be one cluster, into two clusters, the number of objects violating the k-NN and k-MN consistency will be radically increased.

Consequently, our decision will be made based on both the compactness and the connectivity. If they say an agreed number as the optimal number of clusters, we will be able to make a decision with doubled reliability. In some easy cases such as Model 1 in our experiments as shown in Section 4.3, both of them arrived at an agreement of the same number of clusters. However, for some reasons, it may possibly appear that sometimes they come into conflict with each other. Hopefully, we may be able to address this issue by incorporating the concept of

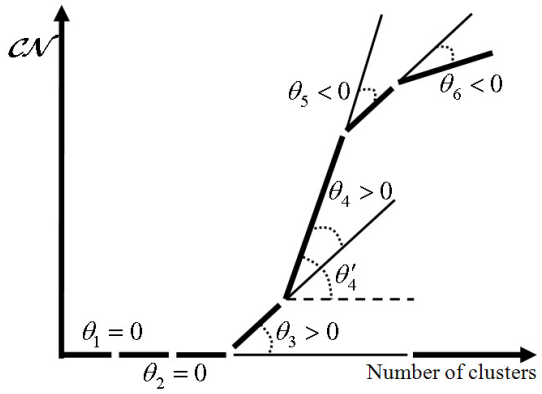


Figure 4.2: A hypothetical plot of $\mathcal{CN}$ versus number of clusters

stability. A meta-learning framework will be designed for this purpose. The proposed method will thus examine a set of partitions resulted from clustering randomly perturbed datasets rather than one single partition of a single dataset. Detailed description will be shown in the following two sections.

4.2.2 Populating Synthetic Samples

In supervised learning, a diversity of techniques for combining classifiers has been developed and theoretical foundations and behavior of those techniques have been studied, proving their validity and providing some guidelines for designing such meta-models (Lam and Suen, 1997; Kittler, 1998; Kittler et al., 1998; Jain et al., 2000). One of the most famous meta-learning methods is the Bootstrap Aggregating called as Bagging (Breiman, 1996) that manipulates the training samples. Briefly, the Bagging is as follows. Each of multiple classifiers built on sets of bootstrapped samples casts a vote for the target value, and one final prediction value is decided by the majority voting principle. The resampling procedure in Bagging has been applied for cluster validation (Levine and Domany, 2001; Fred, 2001). In order to find consistent or stable clusters from a dataset in their research, the training set is bootstrapped and a clustering algorithm is then applied to obtain multiple partitions. Evaluation of those partitions using some external indices can recover consistent clusters from the training dataset. The method we propose in this research also follows a similar scheme. However, instead of resampling the

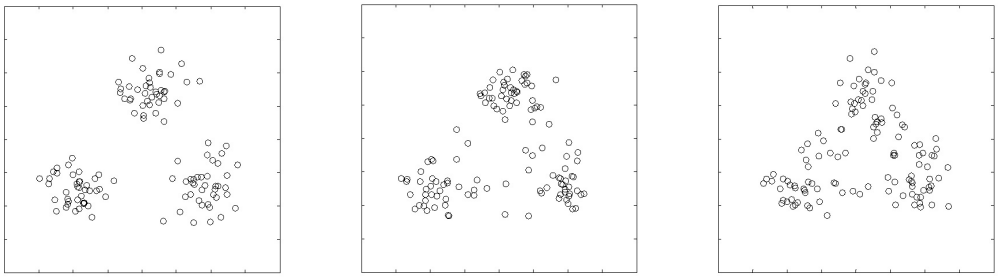
Table 4.1: Decsription of *Populate*() function

1	**Function** *Populate*($\mathbf{X} = (x_{ij}), K$)
2	**Input**
3	Given data table $\mathbf{X} = (x_{ij}), (i = 1, 2, \ldots, n; j = 1, 2, \ldots, p)$
4	Maximum value of nearest neighbors K
5	**For** $i \leftarrow 1$ **To** n
6	Generate a random number k, where $k \sim U[1, K]$.
7	Select $x_{i'} \leftarrow k$th nearest neighbor of x_i from $\mathbf{X}$.
8	Generate a random number α, where $\alpha \sim U[0, 1]$.
9	Create ith new object by $x_i^{new} \leftarrow \alpha \cdot x_i + (1 - \alpha) \cdot x_{i'}$.
10	**End For**
11	**End Function**

given dataset, we generate a number of synthetic datasets by some amount of data distortion, which may be able to reflect the true structure of given dataset. Therefore, it can be said in this study that the proposed method is also based on the stability of clustering with respect to perturbation that is the addition of noise.

SMOTE (Synthetic Minority Over-sampling Technique) was basically invented for unbalanced data classification (Chawla et al., 2002). This method enlarges the mass of the minority class by populating artificial minority class instances in order to cope with the data unbalance, that is, to obtain improved accuracy over the minority class. We adopt one part of the SMOTE that generates synthetic objects, since we believe that it can distort the given dataset efficiently while preserving the true data structure. Let us call the part we employ as 'Populate', and this, with a little modification for its fitness to our research, is described in Table 4.1 in terms of a function. Its implementation is to introduce synthetic objects along the line segments joining one existing object with one of its k-nearest neighbors. Hence, the newly generated object is a convex combination of an existing object and its one of k-nearest neighbors.

In the *Populate*() function, there are two locations inserting randomness; the first part choosing the kth nearest neighbor of the ith object, and the second part creating a new object that lies on the line segment between the ith object and its kth nearest neighbor. Of course, the first element gives more perturbation to the new synthetic dataset than the second part, because the range of randomness in the second part is restricted by the chosen number of k. Assume that the selected number of k is very small, saying that it equals to 1. Then, the newly produced object will be placed nearby the ith object, since it lost a chance to reach to a further object. Figure 4.3 demonstrates how much distortion can be introduced to the new dataset by setting of the maximum value of nearest neighbors K.



(a) Original data ($n = 120$) (b) A sample ($K = 50$) (c) A sample ($K = 80$)

Figure 4.3: Data distortion by *Populate*() function at different number of K

The most important reason that we did not choose a resampling method but chose this scheme of generating objects is to cope with sparseness of a particular cluster. A sampling method such as bootrstrap resampling may not be able to preserve the original cluster structure in some cases. For example, suppose that one cluster in Figure 4.3a has a much smaller number of objects than the other clusters. Since the objects in such a cluster with a lower density have a relatively lower chance to be selected, resampling may result in a structure of two clusters in some sampled datasets. However, the scheme generating synthetic objects is free from this kind of risk in spite of randomness insertion. Furthermore, we can easily control the amount of perturbation by choosing the number K.

4.2.3 Procedure of the Proposed Method

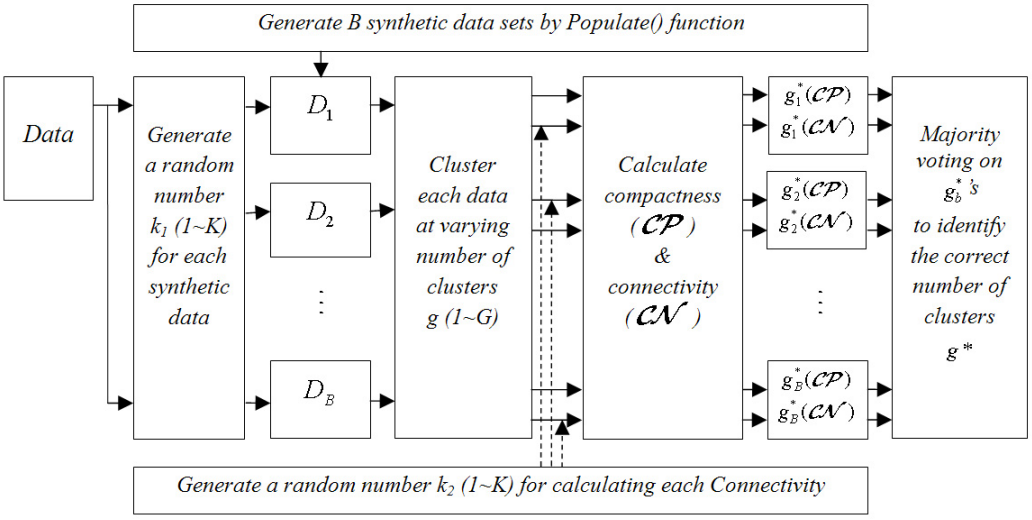


Figure 4.4: Procedure of the proposed estimating method

The whole procedure of the proposed method for indentifying the correct number of clusters is described in Figure 4.4. Based on the given dataset, an artificial dataset D_b, $b = 1, 2, \ldots, B$, is generated by the *Populate*() function with a randomly selected number k_1 that is to give a different amount of distortion. Note that the number k_1 in this step is the maximum number of nearest neighbors denoted by K in the description of Table 4.1 An employed clustering algorithm constructs a partition P_b^g, $g = 1, 2, \ldots, G$, for each date set D_b. We need to predefine the maximum possible number of clusters G to be tested. Before going through the step measuring cluster quality, we introduce another randomness by deciding k_2 which is also the number of nearest neighbors for calculating the connectivity. Since the connectivity measure is affected by the parameter k_2, we leave the problem of setting this number to the randomness in this step. The intention behind inserting randomness into the two places for generating a synthetic dataset and calculating the connectivity was based on our expectation that the true and unknown number of clusters should consistently be able to appear over various degrees of perturbation. In other words, the true number should cover a majority of the space of the parameters, while a false number may occupy a specific range of the parameters to be appeared.

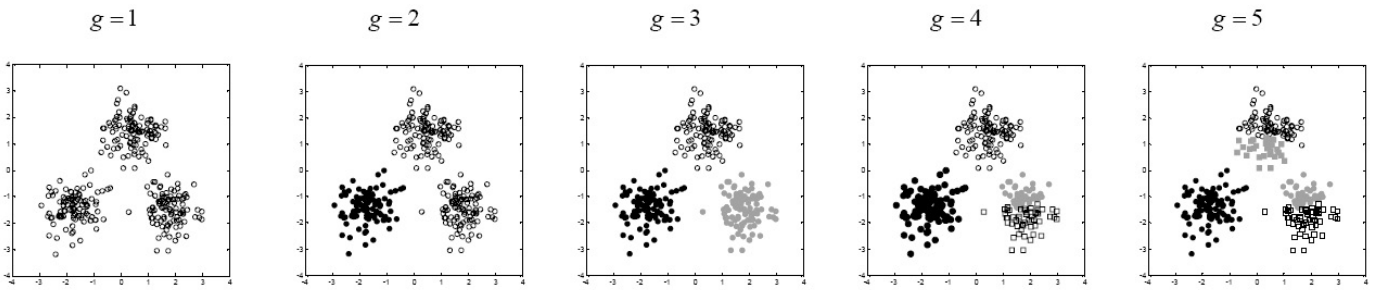
The computed compactness $\mathcal{CP}_b^g$ and connectivity $\mathcal{CN}_b^g$ by Equation 3.1 and 3.3 respectively induce their own optimal number of clusters $g_b^*(\mathcal{CP})$ and $g_b^*(\mathcal{CN})$ according to Equation 4.1 and 4.2. The aggregated g_b^*'s finally choose a single g^* by majority voting scheme.

4.3 Experiments

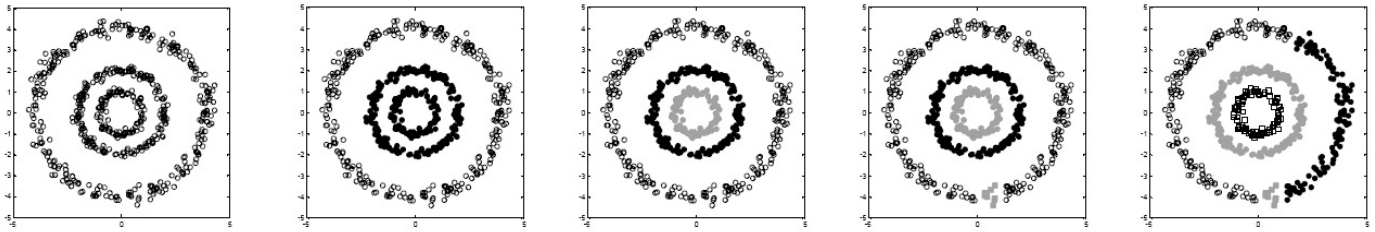
A rigorous simulation study based on 13 data-generating models was conducted to show the effectiveness of the proposed method. Those models were designed with consideration of various types of variation sources such as dimension, number of objects, number of clusters, noise objects, noise variables, shapes of clusters, and different degrees of sparseness for clusters. The description of each model is depicted in Section 4.3.1. We compared the performance of the proposed method with those of 7 existing approaches presented in Section 2.2.2 that are $CH(g)$, $KL(g)$, $Hart(g)$, $\overline{Sil}(g)$, $Gap(g)$, BH, and $Clest$. The experimental results in terms of accuracy are shown in Section 4.3.2. This section also provides some interesting findings and interpretations.

4.3.1 Simulation Models

In order to evaluate the procedures for estimating the number of clusters in a dataset, simulated datasets were used from a variety of data-generating models including some of those commonly considered by Tibshirani et al. (2001) and Dudoit and Fridlyand (2002). The models used for our experiments have different number of clusters, different number of variables, different number of objects, different shapes of clusters, a wide range of covariance structure for Gaussian-distributed clusters. We also included a number of irrelevant noise variables in order to obscure the underlying cluster structure to be recovered. In addition to the noise variables, noise objects uniformly distributed over the space were also under our consideration with the same intention of adding noise variables.



(a) Model 1 clustered by Ward's method



(b) Model 3 clustered by single linkage method

Figure 4.5: Clustering behaviors based on different models and clustering algorithms

The choice of clustering algorithm for each model is important to cluster datasets properly, because it is indeed meaningless to attempt finding the correct number of clusters based on wrong clustering results as mentioned in Section 4.1. We selected an appropriate clustering algorithm for each model to create a desirable cluster hierarchy at sequentially increasing number of clusters. Figure 4.5 shows representative examples of a good combination of simulation model and clustering method. Hierarchical Ward's clustering method was employed to acquire the desirable clustering solutions at different number of clusters for Model 1. Its clustering capability for this model is illustrated in Figure 4.5a. However, it is obvious that the Ward's method does not well cluster the dataset which has the non-spherically shaped clusters such as the three rings in Figure 4.5b. The dataset in this figure was generated from Model 3. Hierarchical Single linkage method was therefore employed to cluster the data and the resulting cluster hierarchy over the different number of clusters was well-constructed as shown in Figure 4.5b. The selected clustering algorithm will be mentioned with the description of each simulation model. The 13 simulation models are as below, and Figure 4.6 shows some examples from two-dimensional models only.

Model 1: *Three clusters in two dimensions*

The clusters are Gaussian-distributed by the variance-covariance matrix of $\Sigma_j = 0.3 \cdot \mathbf{I}_2$, $j = 1, 2, 3$, where $\mathbf{I}_2$ denotes a 2×2 identity matrix. They have $(100, 100, 100)$ instances centered at

$$\left\{ \mu_1 = \begin{pmatrix} -1.8 \\ -1.5 \end{pmatrix}, \mu_2 = \begin{pmatrix} 0.3 \\ 1.5 \end{pmatrix}, \mu_3 = \begin{pmatrix} 1.8 \\ -1.5 \end{pmatrix} \right\}.$$

Model 2: *Two elongated clusters in three dimensions*

Each cluster is generated as follows. Set $x_1 = x_2 = x_3 = t$ with t taking 100 equal spaced values from -0.5 to 0.5, and then Gaussian noise with standard deviation 0.1 is added to each variable. Cluster 2 is generated in the same way, except that the value 2 is added to each variable at the end. The result is two elongated clusters stretching out along the main diagonal of a three dimensional cube. This model is adopted with intention to break down the $Gap(g)$

method. Hierarchical Ward's method was employed to cluster datasets generated from this model.

Model 3: *Three rings in two dimensions*

The angular coordinates (θ) of the data instances are selected from a uniform distribution, and the radial coordinates (r) are Gaussian-distributed around three different radii as shown below.

$$\theta \sim U[0, 2\pi], \;\; r \sim N[R, \sigma^2]$$

The outer ring parameterized by $R = 4.0$ and $\sigma = 0.2$ consists of 300 data instances. The bigger inner ring consisting of 200 data instances has $R = 1.0$ and $\sigma = 0.1$. Lastly, the smaller inner ring with $R = 1.0$ and $\sigma = 0.1$ consists of 100 instances. This model was to produce an example of non-spherically shaped clusters, and it will be very hard for some cluster methods such as k-means algorithm to cluster this example properly. We employed the hierarchical single-linkage method to cluster the datasets generated from this model, since the results by this method were well-conducted as we confirmed from Figure 4.5b.

Model 4: *Three clusters in two dimensions with noise objects*

Three clusters are generated from Model 1 and another 100 noise objects distributed uniformly over the two-dimensional space are added. The variables of noise object are independently simulated from a uniform distribution $U[min_i, max_i]$ where min_i is the minimum value and max_i is the maximum value of ith variable ($i = 1, 2$). In this case, it is very likely to see that Ward's method, complete linkage method, average linkage method, or model-based algorithm properly clusters the dataset. We used the Ward's method.

Model 5: *Two elongated clusters in ten dimensions with seven noise variables*

The two elongated clusters are generated as in Model 2, but, in addition, seven noise variables are simulated independently from the Gaussian distribution with mean 0 and variance v^2 for the vth variable, $v = 4, 5, \ldots, 10$. Hierarchical Ward's method was also employed for

this model.

Model 6: *Three nested spheres in three dimensions*

This model is a three-dimensional version of Model 3. The angular coordinates (θ) and the radial coordinates (r) are the same as in Model 3. In order to create a sphere we need to define one more coordinates for the zenith angle. The third coordinates (φ) are generated from the uniform distribution which is $U[0, 2\pi]$. These three coordinates can be converted into Cartesian coordinates by the following transformation, $x_1 = r\sin\varphi\cos\theta$, $x_2 = r\sin\varphi\sin\theta$, $x_3 = r\cos\varphi$. Single linkage algorithm will cluster the datasets generated from this model.

Model 7: *Five clusters in ten dimensions*

Each cluster is randomly chosen to contain either 25 or 50 objects, with means also randomly selected from the Gaussian distribution $N[\mathbf{0}_{10}, 3.6\mathbf{I}_{10}]$ where $\mathbf{0}_{10}$ denotes a 10×1 vector consisting of zeros and $\mathbf{I}_{10}$ denotes a 10×10 identity matrix. Such a design indicates that each simulation of this model generates a different set of cluster centers. The objects in a given cluster are independently drawn from another Gaussian distribution with an identity covariance matrix and the selected mean vector. In order to preserve the structure of five clusters, any simulation, in which the Euclidean distance between the two closest objects belonging to different clusters is less than 1, will be removed. Ward's method was adopted in this simulation.

Model 8: *Five clusters in ten dimensions with noise objects*

The design for setting cluster centers is the same as in Model 7, but it is different that each cluster has 50 objects only, and the Gaussian distribution for generating the cluster means is $N[\mathbf{0}_{10}, 5\mathbf{I}_{10}]$ in order to introduce more possibility of further separation of the clusters. The 50 objects in each cluster are also generated from the standard multivariate normal distribution. At this point, we check if the condition that the distance between two closest objects belonging to different clusters should be greater than 1 is satisfied. If it does not hold, we discard the simulation and repeat the above procedure. For the simulated dataset satisfying the condi-

tion, we then add 50 more objects uniformly distributed over the ten-dimensional hypercube. We also employed the Ward's clustering algorithm to cluster the datasets simulated from this model.

Model 9: *Two dense and three sparse clusters in two dimensions*

All five clusters are drawn from Gaussian distributions with the following setting of means and one identical covariance matrix which is $0.1\mathbf{I}_2$.

$$\left\{\mu_1 = \begin{pmatrix} -2 \\ -2 \end{pmatrix}, \mu_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \mu_3 = \begin{pmatrix} -2 \\ 2 \end{pmatrix}, \mu_4 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mu_5 = \begin{pmatrix} 2 \\ -2 \end{pmatrix}\right\}$$

Two dense cluster corresponding to μ_1 and μ_2 consist of 100 objects each, and the other three clusters which means are μ_3, μ_4, and μ_5 respectively have 10 objects each to make them sparser than the other two clusters. Ward's linkage method is suitable to cluster the datasets generated from this model, so it was adopted.

Model 10: *Nine small clusters forming three large clusters in two dimensions*

First three clusters are created from Model 1 and we then add 7 to the first variable to move them along with the first axis. The second and third three clusters are also generated from the same model, but then -7 is added to the first variable of the second three clusters to shift them into the opposite direction to the first three clusters while we add 10 to the second variable of the last three clusters to move them away from the other two clusters. Consequently, each of three clusters forms one large cluster in a global view of the clusters structure. Ward's method is also good for clustering the datasets.

Model 11: *Four clusters drawing turbine blades in two dimensions*

The objects in each cluster are randomly generated from the Gaussian distribution with μ_j and Σ_j where j denotes the jth cluster. The μ_j and Σ_j are defined as below:

$$\left\{\mu_1 = \begin{pmatrix} 0 \\ 5 \end{pmatrix}, \Sigma_1 = \begin{pmatrix} 0.01 & 0 \\ 0 & 2 \end{pmatrix}\right\}, \left\{\mu_2 = \begin{pmatrix} -4 \\ -4 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 1.05 & 1 \\ 1 & 1.05 \end{pmatrix}\right\},$$

$$\left\{\mu_3 = \begin{pmatrix} 4 \\ -4 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 1.05 & -1 \\ -1 & 1.05 \end{pmatrix}\right\}, \left\{\mu_4 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma_4 = \begin{pmatrix} 0.4 & 0 \\ 0 & 0.4 \end{pmatrix}\right\}.$$

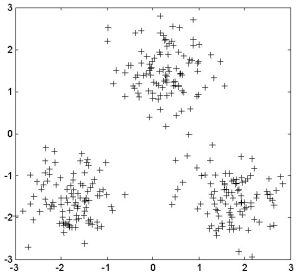
Each cluster consists of 100 objects and these four clusters form the shape of turbine blades.

Model 12: *Two clusters forming Taeguk in two dimensions*

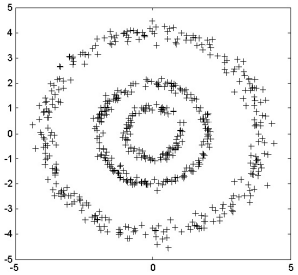
Two overlapped half-circles form the Taeguk shape. In addition to Model 3 and 6, this is another simulation model for generating non-spherically shaped clusters where the compactness-oriented estimating methods may fail to find the true cluster structure. The radial coordinates $r \sim N[3, 0.3^2]$ are the same for both upper half-circle and lower half-circle, but the angular coordinates are generated from $\theta_{upper} \sim U[0, \pi]$ for the upper and $\theta_{lower} \sim U[\pi, 2\pi]$ for the lower correspondingly. By adding positive numbers 3 and 1.5 to the first variable and the second variable of the lower half-circle respectively, we shift the lower half-circle for the purpose of overlapping and the Taeguk mark is finally formed.

Model 13: *Null (single cluster) data in ten dimensions*

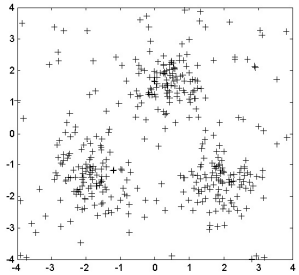
200 data objects are uniformly distributed over the unit square in 10 dimensions. This model is designed in order to show that some of the estimating methods in this simulation study are able to detect the non-cluster structure meaning a single cluster whereas some of those are not.



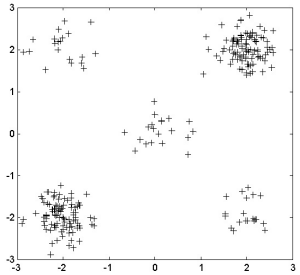
(a) Model 1



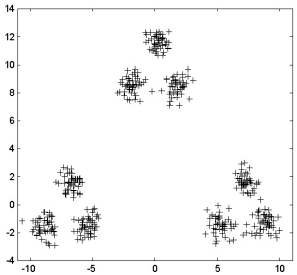
(b) Model 3



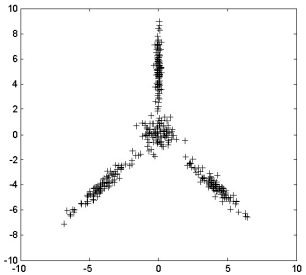
(c) Model 4



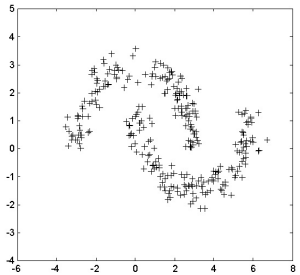
(d) Model 9



(e) Model 10



(f) Model 11



(g) Model 12

Figure 4.6: Two-dimensional simulation models

4.3.2 Experimental Results

We performed 50 simulations for each model to generate 50 datasets from one data-generating model, and the 8 estimating approaches were then applied to find the correct number of clusters. We considered a range of 1 to 10 as the possible number of clusters. Table 4.2, 4.3, 4.4, and 4.5 provide a detailed account of the experimental results. Numbers in the tables are counts out of 50 trials, and g^* is the designed number of clusters for each model that is regarded as a true number of clusters. In addition to the tables, the results are summarized in Figure 4.7 in terms of the percentage of the correct estimates. Note that we considered both 3 and 9 as correct estimates for Model 10.

The overall appearance in this simulation study was that there is no single approach always outperforming the others across the 13 simulation models, meaning that some methods showed good performances for some simulation models but they failed to discover the true cluster structures for another models. However, it is noticeable that the proposed method showed good accuracies in general although it was not the best for all the models. In order to help our understanding the results in this section, some comments and interesting results are highlighted as below.

For the simulation model 1 which was provided as the easiest case, all methods except $Hart(g)$ and $Clest$ estimated the true number of clusters in the most trials out of 50 simulations. The proposed method, $CH(g)$, $\overline{Sil}(g)$, and $Gap(g)$ perfectly recovered 3 number of clusters during all 50 trials. In the experiments of Model 2, five methods among the eight, which are the proposed method, $KL(g)$, $\overline{Sil}(g)$, BH, and $Clest$, were successful to detect the correct number of clusters. It can be seen that $CH(g)$, $Hart(g)$, and $Gap(g)$ entirely failed to distinguish between one and the other cluster. This simulation model pointed to the weakness of those three approaches. Such a weakness of those against this model is a consistent finding with the experimental results shown in Tibshirani et al. (2001) and Dudoit and Fridlyand (2002). In a majority of the simulations from Model 3 that is a non-spherically shaped case, only the proposed method was successful to estimate the correct number of clusters. It is interesting to note that $Gap(g)$ consistently indicated a single cluster for this model, and this

result agrees with the working attitude of the $Gap(g)$ when we consider that the 3 rings occupy all around of the space that encloses those 3 rings.

Based on the results from the first three models, we would like to mention one feature of the proposed estimating method. Model 1 and Model 2, in fact, belong to somehow easy clustering problems. We observed that the g_b^*'s from the both compactness and connectivity were quite consistent, although we did not report here. In such easy cases, we may be able to remove the meta-learning scheme, saying that we do not have to generated multiple data sets, since just one calculations of the $\mathcal{CP}$ and $\mathcal{CN}$ will probably agree upon the estimate of a single cluster number which is correct. However, in the case of Model 3, a quite large range of g_b^*'s was observed, while the majority voting finally found the correct answer in around 80% of trials. It might be due to either wrong clustering solutions from the clustering algorithm itself or randomness from calculating $\mathcal{CN}$. What we believe from this observation is that the meta-learning scheme used in this research may give a chance to correctly cluster objects to the employed clustering algorithm and/or to properly calculate the $\mathcal{CN}$. In the meantime, the compactness did not contribute to decide the optimal number at all in Model 3. The estimated $g_b^*(\mathcal{CP})$'s were wrong in many cases out of the 50 trials and also inconsistent as well.

It appears that the noise objects in Model 4 might obfuscate the detection capability of BH. Since this approach relies on the sampled subsets of the given data, some of those might not be able to reflect the real clusters by introducing noise objects. $Hart(g)$ was also totally unsuccessful to detect the three clusters in data. For model 5, $Gap(g)$ and $Clest$ performed poorly, while $CH(g)$, $\overline{Sil}(g)$, and the proposed method showed good performance. It was anticipated that in majority of the simulations of Model 6 all the existing approaches failed to uncover the three spheres in the three-dimensional space. This result is similar to what we found in the simulation of Model 3. For Model 7, the proposed method, $CH(g)$, $KL(g)$, and $Gap(g)$ completely succeeded to figure out five clusters from this model and $\overline{Sil}(g)$ and BH showed a good performance, while $Hart(g)$ and $Clest$ entirely and almost failed. Adding noise objects to Model 7 that is Model 8 was a more difficult task to all of the estimating methods. In general, the performance of all approaches was worse than that in the simulation study of

Model 7. The proposed method, however, showed the least degeneration. $\overline{Sil}(g)$ and $Gap(g)$ tended to select as a large number of clusters as possible. If we consider the larger number of clusters in the simulation, they may choose the largest one. The simulation result of the proposed method, $CH(g)$, and $KL(g)$ are satisfactory in this model.

We designed Model 9 for the purpose of seeing how BH works for the datasets having different degrees of density in clusters. The experimental results in this model are saying that the resampling based approaches that are BH and $Clest$ were worse than the others except $Hart(g)$. The $Hart(g)$ actually could not find the correct number of clusters for any of models designed in this simulation study. The worse results of the resampling-based approaches could be due to the fact that once they fail to preserve the original cluster structure when resampling objects, it is apparent that the next procedures lose their working abilities. In this example, since the objects in the sparser clusters have a low probability to be chosen in the sampling step, some or all of the sparser clusters might not appear in the sampled datasets and this could thus result in the poor performance of BH and $Clest$. However, the proposed method could be more robust to such a situation, because it generates similar artificial objects rather than resamples the existing objects.

One of the most interesting experiments in this research was Model 10. Let us take a look at Figure 4.6e, and Model 10 generates this kind of structure of clusters. Two possible numbers of clusters are shown in this figure. Someone may say nine clusters, whereas another may see three clusters or someone may agree on both three and nine clusters. It depends on two different points of view for this cluster structure. This example reminds us of the definition of clusters and shows the fact that there is no single ideal definition of cluster. As we mentioned very earlier in this paper, we are only following what the most of human eyes or opinions agree on in order to describe something about cluster, for example, the number of clusters in this research, and in this simulation model we may agree upon either three or nine. An estimating method should then be able to tell us that two kinds of answer will be available for this model. In the resulting table, the most approaches among eight, $KL(g)$, $\overline{Sil}(g)$, $Gap(g)$, BH, and $Clest$, resulted in showing three clusters, while $CH(g)$ completely

came up with nine clusters. None of these did suggest two possibilities of three and nine, unlike the human eyes do. However, in the fifty simulations, the proposed method voted on three 19 times and on nine clusters 31 times. This result asserts that the proposed method was able to more mimic human eyes than any other methods. Since this simulation model generates two-dimensional datasets, we were actually able to see the datasets generated from the *Populate*() function based on this model and confirmed that a large k_1 resulted in three clusters while a small number of k_1 tried to preserve the original structure of clusters. Hence, if the *Populate*() function inserts more perturbation into the original dataset, then the resulting dataset will be closer to the three clusters structure and the compactness and the connectivity will have the same opinion on the three rather than nine. There is one more reason that the proposed method could say both three and nine. Since we adopt the Krzanowski and Lai's (1985) idea in order to select the elbow point from the compactness, the most of $g_b^*(CP)$'s was three. The contributor to nine clusters was the connectivity due to the fact that more violating objects appear when we go to from nine to ten clusters than the point from three to four clusters. We observed all detailed results from the fifty simulations, and the rationales of such results of the proposed method are, in summary, as follows: First, the compactness tended to come up with three clusters whereas the connectivity did with nine clusters. Second, if the *Populate*() trick generated a synthetic dataset that looked more like to three clusters, then both compactness and connectivity became to agree upon the three clusters. Lastly, if the artificially generated dataset was closer to the original one, nine clusters structure was asserted by the proposed method to which the connectivity governed more by its consistent argument, which was nine, than the compactness. We have attached the detailed results of the proposed method in Model 10 to Appendix A, in order to show how the method arrived at both three and nice clusters.

$CH(g)$ and $Hart(g)$ completely failed to catch up the structure of four clusters with Model 11, but the proposed approach, $KL(g)$, $\overline{Sil}(g)$, $Gap(g)$, and BH was generally successful to do that. *Clest* failed to distinguish between three and four clusters, and it is a possible reasoning that this method could not figure out the spherically distributed cluster at the center because of a poor classifier, which was learned by DLDA (Diagonal Linear Discriminant Analysis),

not handling a various types of covariance structure. For Model 12, the proposed method, $CH(g)$, $\overline{Sil}(g)$, $Gap(g)$, and $Clest$ performed well. Only three approaches, which are the proposed method, $Gap(g)$, and $Clest$, were able to detect the null cluster generated from Model 13, because the others, in fact, do not have the capability of doing that due to their own assumptions on the minimum number of clusters which is two.

Through this simulation study considering various types of variation source of data, we have established that the proposed method showed regularly good results over the range of 13 models, while some of existing methods were not performing well for some specific cases.

Table 4.2: Estimated number of clusters (Model 1 $\sim$ Model 3)

Method	Number of clusters									
	1	2	3	4	5	6	7	8	9	10
Model 1 ($g^* = 3$)										
Proposed	0	0	**50**	0	0	0	0	0	0	0
CH	0	0	**50**	0	0	0	0	0	0	0
KL	0	0	**34**	0	0	3	3	6	3	1
Hart	0	0	1	0	0	3	3	5	16	**22**
Sil	0	0	**50**	0	0	0	0	0	0	0
Gap	0	0	**50**	0	0	0	0	0	0	0
BH	0	0	**36**	12	2	0	0	0	0	0
Clest	0	**29**	21	0	0	0	0	0	0	0
Model 2 ($g^* = 2$)										
Proposed	0	**50**	0	0	0	0	0	0	0	0
CH	0	1	0	7	3	**21**	8	9	0	1
KL	0	**45**	0	1	0	1	0	1	0	2
Hart	0	0	0	0	0	0	0	3	12	35
Sil	0	**50**	0	0	0	0	0	0	0	0
Gap	0	0	0	11	5	**23**	10	1	0	0
BH	0	**50**	0	0	0	0	0	0	0	0
Clest	0	**50**	0	0	0	0	0	0	0	0
Model 3 ($g^* = 3$)										
Proposed	0	9	**38**	0	0	1	1	1	0	0
CH	0	0	2	3	2	5	**11**	8	**11**	8
KL	0	3	4	2	**10**	5	7	7	5	7
Hart	0	**25**	7	2	1	3	1	1	4	6
Sil	0	**28**	4	3	3	0	2	1	3	6
Gap	**50**	0	0	0	0	0	0	0	0	0
BH	0	**42**	7	1	0	0	0	0	0	0
Clest	0	**50**	0	0	0	0	0	0	0	0

Table 4.3: Estimated number of clusters (Model 4 $\sim$ Model 7)

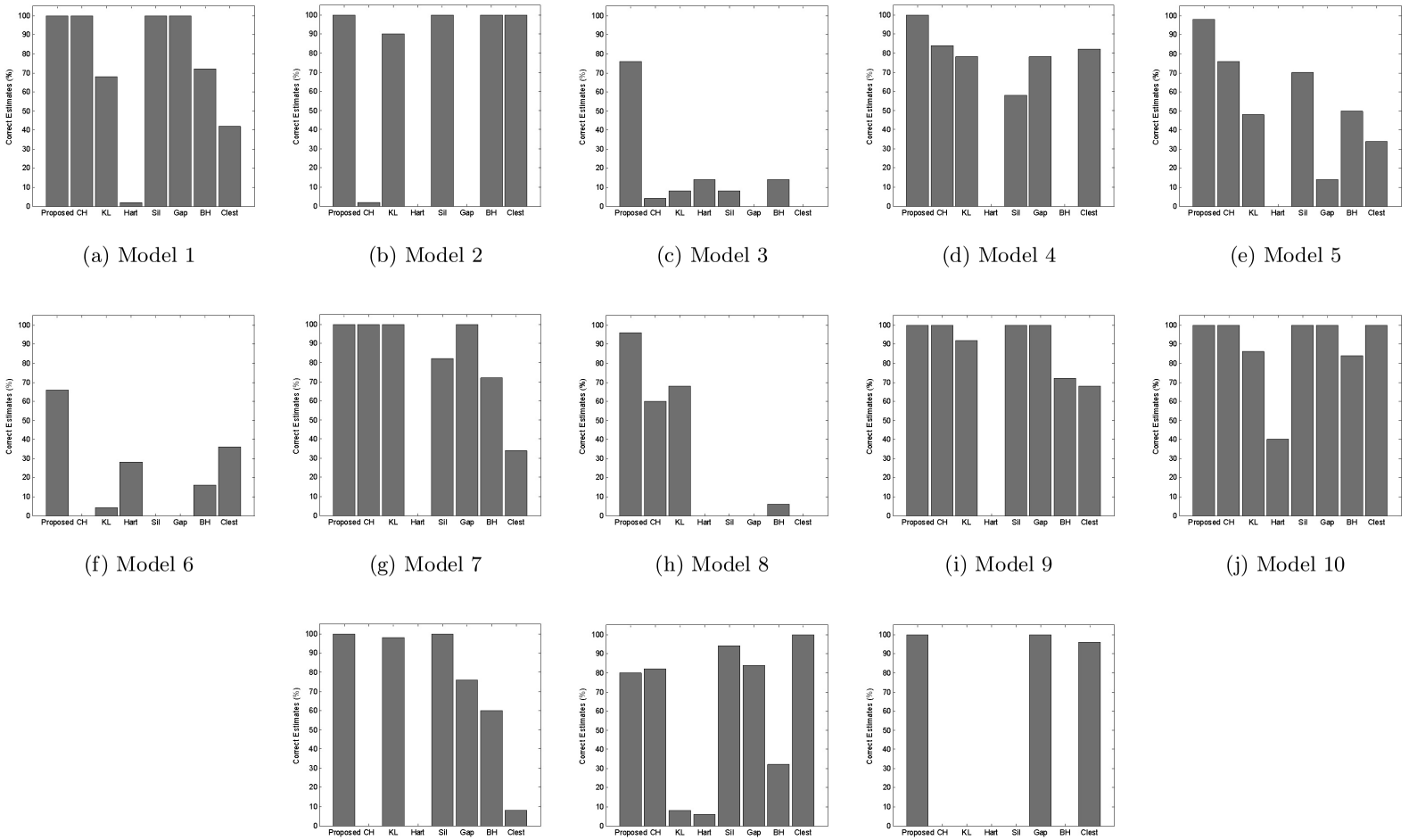
Method	Number of clusters									
	1	2	3	4	5	6	7	8	9	10
Model 4 ($g^* = 3$)										
Proposed	0	0	**50**	0	0	0	0	0	0	0
CH	0	0	**42**	7	1	0	0	0	0	0
KL	0	0	**39**	1	0	1	1	2	0	6
Hart	0	0	0	0	0	3	3	10	11	**23**
Sil	0	0	**29**	11	10	0	0	0	0	0
Gap	11	0	**39**	0	0	0	0	0	0	0
BH	0	0	0	2	**31**	10	6	0	1	0
Clest	0	9	**41**	0	0	0	0	0	0	0
Model 5 ($g^* = 2$)										
Proposed	1	**49**	0	0	0	0	0	0	0	0
CH	0	**38**	12	0	0	0	0	0	0	0
KL	0	**24**	10	3	4	5	0	1	2	1
Hart	0	0	0	0	0	1	2	2	12	**33**
Sil	0	**35**	5	3	0	2	2	0	0	3
Gap	**43**	7	0	0	0	0	0	0	0	0
BH	0	**25**	19	3	3	0	0	0	0	0
Clest	3	17	**20**	6	1	1	1	0	0	1
Model 6 ($g^* = 3$)										
Proposed	0	4	**33**	2	1	1	5	2	2	0
CH	0	1	0	3	4	5	6	10	7	**14**
KL	0	3	2	4	3	7	6	8	3	**14**
Hart	0	**30**	14	1	2	2	0	0	0	1
Sil	0	**50**	0	0	0	0	0	0	0	0
Gap	**50**	0	0	0	0	0	0	0	0	0
BH	0	**33**	8	3	0	1	3	1	1	0
Clest	0	0	18	**28**	4	0	0	0	0	0
Model 7 ($g^* = 5$)										
Proposed	0	0	0	0	**50**	0	0	0	0	0
CH	0	0	0	0	**50**	0	0	0	0	0
KL	0	0	0	0	**50**	0	0	0	0	0
Hart	0	0	0	0	0	3	4	4	16	**23**
Sil	0	0	0	9	**41**	0	0	0	0	0
Gap	0	0	0	0	**50**	0	0	0	0	0
BH	0	2	2	0	**36**	9	1	0	0	0
Clest	0	**25**	5	3	17	0	0	0	0	0

Table 4.4: Estimated number of clusters (Model 8 $\sim$ Model 11)

Method	Number of clusters									
	1	2	3	4	5	6	7	8	9	10
Model 8 ($g^* = 5$)										
Proposed	0	0	0	1	**48**	1	0	0	0	0
CH	0	1	0	2	**30**	12	3	1	0	1
KL	0	1	3	4	**34**	7	1	0	0	0
Hart	0	0	0	0	0	1	0	5	18	**26**
Sil	0	0	0	0	0	0	0	0	0	**50**
Gap	0	0	0	0	0	0	0	0	0	**50**
BH	0	**20**	16	9	3	0	0	1	1	0
Clest	0	**26**	4	5	0	0	5	2	4	4
Model 9 ($g^* = 5$)										
Proposed	0	0	0	0	**50**	0	0	0	0	0
CH	0	0	0	0	**50**	0	0	0	0	0
KL	0	0	0	0	**46**	2	2	0	0	0
Hart	0	0	0	0	0	0	0	2	11	**37**
Sil	0	0	0	0	**50**	0	0	0	0	0
Gap	0	0	0	0	**50**	0	0	0	0	0
BH	0	0	0	0	**36**	14	0	0	0	0
Clest	0	11	5	0	**34**	0	0	0	0	0
Model 10 ($g^* = 3$ or 9)										
Proposed	0	0	**19**	0	0	0	0	0	**31**	0
CH	0	0	0	0	0	0	0	0	**50**	0
KL	0	0	**43**	7	0	0	0	0	0	0
Hart	0	0	0	0	0	0	0	0	20	**30**
Sil	0	0	**50**	0	0	0	0	0	0	0
Gap	0	0	**50**	0	0	0	0	0	0	0
BH	0	0	**42**	2	1	4	1	0	0	0
Clest	0	0	**50**	0	0	0	0	0	0	0
Model 11 ($g^* = 4$)										
Proposed	0	0	0	**50**	0	0	0	0	0	0
CH	0	0	0	0	0	0	7	2	1	**40**
KL	0	0	0	**49**	0	0	1	0	0	0
Hart	0	0	0	0	0	0	12	6	13	**19**
Sil	0	0	0	**50**	0	0	0	0	0	0
Gap	9	0	0	**38**	2	0	0	1	0	0
BH	0	0	0	**30**	6	2	5	3	4	0
Clest	0	1	**45**	4	0	0	0	0	0	0

Table 4.5: Estimated number of clusters (Model 12 $\sim$ Model 13)

Method	Number of clusters									
	1	2	3	4	5	6	7	8	9	10
Model 12 ($g^* = 2$)										
Proposed	1	**40**	4	1	2	1	0	0	1	0
CH	0	**41**	2	0	2	1	0	1	2	1
KL	0	4	8	3	7	8	3	**9**	4	4
Hart	0	3	4	4	7	5	4	4	**10**	9
Sil	0	**47**	3	0	0	0	0	0	0	0
Gap	5	**42**	2	1	0	0	0	0	0	0
BH	0	**16**	12	8	3	9	2	0	0	0
Clest	0	**50**	0	0	0	0	0	0	0	0
Model 13 ($g^* = 1$)										
Proposed	**50**	0	0	0	0	0	0	0	0	0
CH	0	**39**	9	1	1	0	0	0	0	0
KL	0	9	**32**	9	0	0	0	0	0	0
Hart	0	0	0	0	0	0	2	5	10	**33**
Sil	0	10	0	0	0	1	1	4	6	**28**
Gap	**50**	0	0	0	0	0	0	0	0	0
BH	0	**50**	0	0	0	0	0	0	0	0
Clest	**48**	0	1	0	1	0	0	0	0	0



(a) Model 1 (b) Model 2 (c) Model 3 (d) Model 4 (e) Model 5

(f) Model 6 (g) Model 7 (h) Model 8 (i) Model 9 (j) Model 10

(k) Model 11 (l) Model 12 (m) Model 13

Figure 4.7: Percentage of correct estimates

4.4 Chapter Summary and Discussions

In this chapter, we have proposed a new estimating method in order to discover the best number of clusters in a dataset. Not only the compactness that, in the most of previous research studies, has been used but also the connectivity newly introduced from the cluster k-NN and k-MN consistency concepts were used to suggest the method working for both convex-shaped and non-convex-shaped clusters. The proposed method was designed in a framework of meta-learning through a function that generates synthetic datasets by inserting some amount of perturbation for the purpose of giving more reliability to the finally estimated number of clusters. One more advantage of the proposed method is that it is designed to be applicable with almost any clustering methods. Hence, the proposed method is independent from an employed clustering algorithm in cluster analysis.

We have examined the proposed method with the other existing approaches throughout an intensive simulation study based on 13 data-generating models. The models tried to vary geometry of data used for testing, with consideration of many different types of clustering. The experimental results seem positive. For the simulated datasets, the proposed approach performed well across a wide range of models with varying the number of clusters including null cluster, different numbers of variables, various shapes of clusters, noise objects, noise variables, and different covariance matrix structures for Gaussian-distributed objects. In more detail, it showed better performance than the others especially for the non-convex shaped clusters in Model 3 and 6 with around 70% of the correct estimates, while all of existing approaches was worse than a chance which is 10% (selecting one out of 10 numbers of clusters in our study). The proposed method was also robust to some cases having noise objects or noise variables where some of existing method could not correctly estimate the true number of clusters. In addition, we showed one particular example tempting two different answers simultaneously which is Model 10 where the proposed method accurately suggested both numbers as possible estimates of cluster numbers. It was satisfactory that the method could behave itself like human eyes' judgment. Based on the results in this simulation study, we arrived at a conclusion that the proposed method is a promising tool in order to address an important and challenging

problem which is to identify the best number of clusters.

However, due to several characteristics of procedures in the proposed method, the method may have some limitations and weakness. First of all, the proposed method needs a large amount of computational time according to the large size of clustering problem because of its meta-learning framework like another resampling based methods and the *Gap* statistic, since we need to apply the chosen clustering algorithm to a number of datasets either resampled or generated and then validate the same number of clustering results. Second, since the proposed method is a sort of *ad hoc* approach to estimate the number of clusters although it is built on reasonable rationales such as the characteristics of compactness and connectivity, synthetic datasets, and the majority voting scheme, there is no theoretical guarantee that the proposed method always works for any types of data. One potential example that we can expect where the proposed method may fail to discover the true number is the dataset of two spirals that will be shown in the next chapter. If the characteristic of a cluster is a kind of very thin and long as a line-like cluster is, it may deteriorate the detection capability of the proposed method. Lastly, in order to operate the proposed method, we need to predetermine three parameters that are the number of synthetic datasets, the number of nearest neighbors as an input argument of *Populate*() function, and another number of nearest neighbors for calculating the connectivity, and wrong settings of those parameters may result in a poor performance.

Therefore, the following issues may need to be addressed as for the further investigation of this research. A sensitivity analysis on different settings of parameters of the proposed method will need to be performed in order to obtain a good range of each. Throughout this analysis we will be able to suggest what parameter setting are practically reasonable for what types of dataset. If we can derive some key parts of theoretical bases of the method, it will also fortify the weakness of the proposed method. However, we may have to make some assumptions about several things such as distributions of clusters and shapes of clusters. Additionally, an empirical data analysis based on real datasets from different applications will give us good examples of an appropriate use of the proposed method.

CHAPTER 5. CONNECTIVITY-BASED CLUSTERING APPROACH

Measuring cluster quality is useful not only for validating clustering solutions, i.e. identifying the promising number of clusters but also for developing or improving a clustering algorithm itself, because it can be used as an objective of a clustering method. For example, the well-known k-means clustering has the objective of having the optimal 'compactness' at given number of clusters. We have demonstrated how the 'connectivity' measure can be valuable to the problem of deciding the optimal number of clusters. In this chapter we propose a new clustering method in which the connectivity measure plays an important role. In order to incorporate the connectivity into the frame of a clustering algorithm, we developed a new heuristic approach that attempts to optimize the connectivity objective.

5.1 Optimization Point of View on Clustering

As can be seen in Section 2.1, a clustering problem is generally an optimization problem and its integer programming formulation is given in Equation 2.15 which is the most typical form, Equation 2.16 being formulated for k-means clustering, and Equation 2.19 that is the case of k-medoids clustering. Besides such classical formulations first introduced by Vinod (1969) and shown in Kaufman and Rousseeuw (1990), people have been trying to interpret the clustering problem as many different mathematical programming formulations such as linear programming (Olafsson et al., 2008).

A constrained optimization problem to which most of clustering problems belong consists of two parts; the objective function part and the second part of one or more constraints. The inherent nature of clustering that partitions a dataset into several sub-groups toward the maximized within-cluster homogeneity and between-cluster heterogeneity formulates the objective

function part. Based on the meaning of partitioning which corresponds to the second part, the clustered sub-groups should be non-overlapped in the case of 'hard' clustering into which the most of existing clustering methods fall. The formulation of this requirement conducts the corresponding constraints in terms of the expressions that one object should be assigned into no more than one cluster and each cluster should have at least one object. Unfortunately, due to the addition of those conditions into a clustering problem, the number of constraints increases with respect to the number of objects and the number of clusters, although its increment is linear. Moreover, the objective function is also difficult to evaluate directly, because it usually forms both non-linearity and non-convexity that may have many local optimum solutions in the space of integer. This kind of formulation is generally considered as a difficult and large-scale optimization problem.

With respect to this constraint issue, there is one more concern that makes a clustering problem extremely difficult. Suppose that we have a clustering problem where its size is very small with four objects {1, 2, 3, 4} and two clusters, meaning that we want to cluster those four objects into 2 sub-groups. All possible solutions are then {(1), (2, 3, 4)}, {(2), (1, 3, 4)}, {(3), (1, 2, 4)}, {(4), (1, 2, 3)}, {(1, 2), (3, 4)}, {(1, 3), (2, 4)}, and {(1, 4), (2, 3)}. Hence, the number of all solutions to be evaluated is finite and we can obtain the global optimum solution by assessing all of those partitions. However, obtaining the exact solution of the problem is theoretically possible, not yet feasible in practice, because the number of possible solutions increases incredibly with respect to the number of objects and the number of clusters. Again, assume that we have a dataset consisting of n objects and we wish for obtaining g partitions. If exhaustive enumeration is used to solve this clustering problem, then one requires the evaluation of $S(n, g)$ partitions (Anderberg, 1973; Spath, 1980) which is given by

$$S(n,g) = \frac{1}{g!}\left(\sum_{j=1}^{g}(-1)^{g-j}\binom{g}{j}j^n\right) \tag{5.1}$$

Some examples showing the possible number of partitions are given as follows.

$$S(15,3) = 2,375,101 \tag{5.2}$$

$$S(20,4) = 45,232,115,901 \tag{5.3}$$

$$S(100, 5) = 10^{68} \tag{5.4}$$

It is clearly indicated that exhaustive enumeration cannot lead to the required solution for most practical clustering problems in a reasonable time.

Due to such difficulties mentioned above, approximate heuristic techniques looking for a compromise or a near optimal solution which performance is reasonable to be accepted have usually been adopted. As a short comment, the k-means clustering and PAM algorithm introduced in Section 2.1 also fall into the class of heuristic approaches. Genetic Algorithms (GA) have widely been employed in the research line of heuristics-based clustering (Murthy and Chowdhury, 1996; Maulik and Bandyopadhyay, 2000; Garai and Chaudhuri, 2004), because of its multi-dimensional and stochastic search capability. Hence, the GA-based clustering approaches have been proposed with an assertion that they could be good alternatives of the well-known k-means clustering in respect of better reaching to the global optimum solution in a reasonable computational time. However, it has been found that the GA-based methods could be inefficient if we fail to make a compromise between two conflicting facets; the maintenance of population diversity and the optimality guarantee with fewer changes in the bits of the present best strings as the GA goes nearer to the optimum. It may be difficult to satisfy those, since we usually cannot reflect the structure of clustering problem into the GA procedures after encoding the original solution to the chromosome representation. Moreover, none of the previous studies did consider the connectivity concept that we have shown in Chapter 3 and 4, as their objective to be optimized.

The objective function part in a clustering formulation is also tremendously important, because the same data can have more than one relevant structure, each one in accordance with a different cluster criterion. Most of traditional clustering algorithms strive for compact clusters, a criterion which is usually implemented by keeping intra-cluster homogeneity. They include algorithms like k-means clustering, average linkage agglomerative clustering, self-organizing maps (Kohonen, 2001), or model-based clustering (Banfield and Raftery, 1993). The resulting methods tend to be very effective for convex-shaped, spherically shaped, and/or well-separated clusters, but they may fail for more complicated cluster structures. In the meantime, clustering

approaches based on a criterion of a general meaning of connectivity have been received less attention than those of compactness objective and few algorithms have thus been mentioned in the literature. Briefly, they employ a more local concept of clustering based on the idea that neighbored data items should belong to the same cluster. The single linkage agglomerative hierarchical clustering is the most well-known method implementing the objective function of connectivity. Methods belonging to this category are well-suited to detect clusters of arbitrary shapes; however they can lack robustness when there is little spatial separation between clusters.

In this chapter, we attempt to formulate a clustering problem pursuing the connectivity objective mentioned in Section 3.2 and to develop a new heuristic approach for solving the problem with consideration of the constraints stated above. Detailed description of the proposed clustering approach will be mentioned in the following section that is also followed by the section of numerical experiments.

5.2 Proposed Clustering Algorithm

The proposed method falls into the category of the 'hard' clustering which means that resulting clusters should satisfy the meaning of constraints in Equation 2.15 that generates non-overlapping clusters. Hence, each object should not belong to more than one cluster and each cluster should have at least one object. Based on these constraints of non-overlapping, what we attempt to optimize is the connectivity described in Equation 3.3. This quantity should be minimized or reach to the value of zero ideally. The connectivity measure varies at different number of nearest neighbors. It can easily reach to the value of zero with a large number of clusters at a very small number of nearest neighbors, because it is not difficult to satisfy the cluster k-NN and k-MN consistencies. On the other hand, a large number of nearest neighbors make the quantity of connectivity relatively large, and thus it will be very difficult or may be impossible to reach to either quite minimized or zero of the connectivity. It is very likely that a small number of clusters are produced at a given value of k which is large. Therefore, we need to wisely choose the number of nearest neighbors, or it may be a good

way to explore the resulting clusters at sequentially increasing values of k. In our proposed algorithm, we choose the latter strategy for obtaining as good clustering results as possible. This issue will be addressed later.

In order to cluster objects with the objective function and constraints mentioned above, we propose a greedy heuristic approach, named as *CNCLUST* (connectivity-based clustering), and its algorithm is described as below. Following notation will be needed to present it. Note that this method starts with $k = 1$.

Notation

- n : total number of objects in the given dataset
- $\mathcal{A}$: a set containing all objects, i.e. $\mathcal{A} = \{1, 2, \ldots, n\}$
- m : number of resulting clusters by the algorithm
- k : number of nearest neighbors
- g^* : user-defined number of clusters
- $d(i, j)$: distance between object i and j
- $\sigma_i(k)$: a set of k nearest neighbors of object i

Algorithm CNCLUST

1. Initial partition

(a) Set $\Omega = \emptyset$, $P = \emptyset$, and $h = 1$.

(b) Find an object $\hat{i}$ such that

$$\hat{i} = \underset{i \in \mathcal{A} \setminus \Omega}{\operatorname{argmin}} \sum_{j \in \sigma_i(k)} d(i, j).$$

(c) Construct a cluster $C(\hat{i}) = \sigma_{\hat{i}}(k)$, and this cluster can be divided into two subsets $C_1(\hat{i})$ and $C_2(\hat{i})$ where

$$C_1(\hat{i}) = \left\{ j | \sigma_j(k) \subseteq C(\hat{i}) \right\} \text{ and } C_2(\hat{i}) = C(\hat{i}) \setminus C_1(\hat{i}).$$

(d) For an object $\tilde{j} \in C_2(\hat{i})$, add its nearest neighbors into $C(\hat{i})$, i.e.

$$C(\hat{i}) \leftarrow C(\hat{i}) \cup \sigma_{\tilde{j}}(k)$$

until $C_2(\hat{i}) = \emptyset$.

(e) Update the current Ω by adding the objects in $C(\hat{i})$, i.e. $\Omega \leftarrow \Omega \cup C(\hat{i})$.

(f) Let $C(\hat{i})$ denote C_h, and also let C_h be an element of P, i.e. $P = \{C_h\}$.

(g) If $\mathcal{A} \setminus \Omega = \emptyset$, then stop. Otherwise, $h \leftarrow h + 1$ and then move to Step 1(b).

2. Merge

(a) In the resulting partition $P = \{C_h\}$ (from Step 1.), $h = 1, 2, \ldots, m$ and $m > g^*$, select a pair of clusters $C_{\hat{s}}$ and $C_{\hat{t}}$ such that

$$\left(\hat{s}, \hat{t}\right) = \underset{(s,t)\in\{1,2,\ldots,m\}}{\operatorname{argmax}} \eta_{st}$$

where

$$\eta_{st} = \sum_{i\in C_s} \sum_{j\in C_t \wedge j\in\sigma_i(k)} \frac{1}{d(i,j)},\ |C_s| \leq |C_t|.$$

(b) Merge $C_{\hat{s}}$ into $C_{\hat{t}}$, and then $m \leftarrow m - 1$.

3. Check stopping criteria

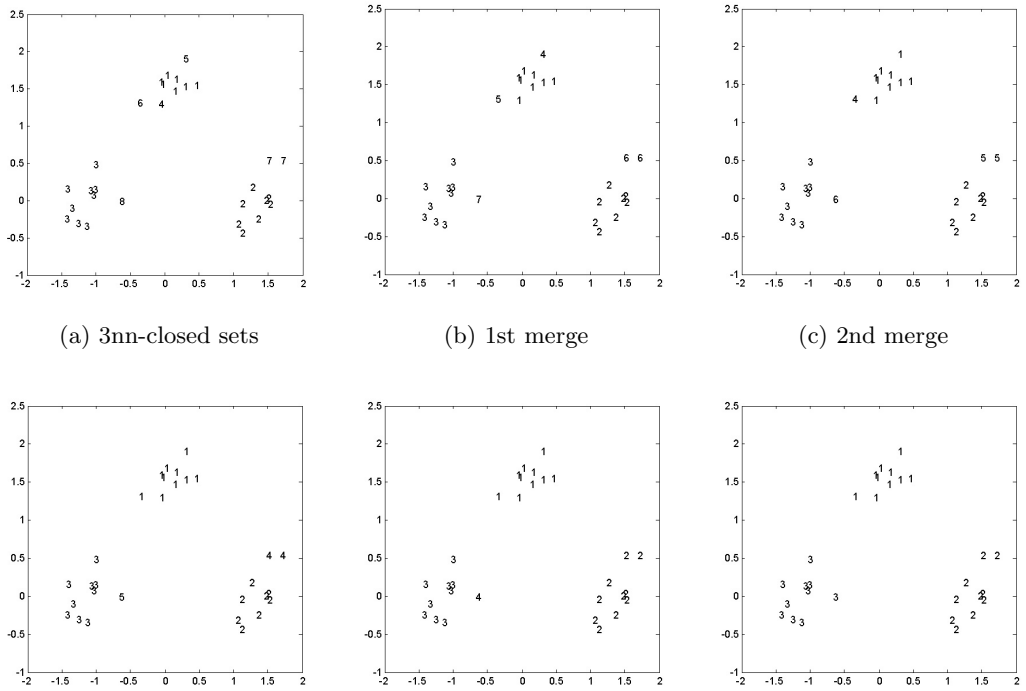
(a) If $m = g^*$, then stop. Otherwise, check if $\eta_{st} = 0$, $\forall(s,t) \in \{1, 2, \ldots, m\}$. If so, $k \leftarrow k + 1$ and then move to Step 1(a). If $\eta_{st} \neq 0$, $\forall(s,t) \in \{1, 2, \ldots, m\}$, then move to Step 2(a).

Step 1, which initially partitions the given objects at a given number of k, constructs groups of objects, namely called as $k$nn-closed sets. When constructing a $k$nn-closed set, we start with the object $\hat{i}$ chosen in Step 1(b) for the purpose of reducing the amount of connectivity as much as possible in earlier stages of algorithm. This step considers a concept of local compactness in data. Step 1(d) is a sort of chain operator that attempts to satisfy the cluster k-NN and k-MN consistency, that is, to minimize our objective function. Hence, successively grouping the neighbor objects of already grouped objects tries to secure as many consistencies of objects as possible. If there is neighbor-distinction between objects, this step stops grouping objects and results in one $k$nn-closed set. We repeat this procedure until all objects in the given dataset are exhausted. Since we consider the constraints of non-overlapping clusters, an object, which belongs to one of constructed sets in earlier steps, cannot be grouped into other closed sets in later steps.

When grouping two clusters together in Step 2, we also try to minimize our objective function as quickly as possible. Therefore, Step 2(a) selects a pair of clusters by measuring the largest amount of violation of cluster k-NN and k-MN consistencies between clusters. We first define a major cluster C_t and a minor cluster C_s by the inequality of $|C_s| \leq |C_t|$, and then compute η_{st}. Its magnitude is relative to both the number of pairs of objects, where one object is in the minor cluster and the other one is its neighbor object in the major cluster, and the distance between those. Since one step merge reduces the number of clusters by one, we can control the number of clusters in Step 2. This merge step is stopped by either $m = g^*$ or $\eta_{st} = 0$. The former means we have arrived at the number of clusters that we defined, while the latter indicates that further merging steps are impossible. Note that $\eta_{st} = 0$ sometimes means the zero objective function value, but sometimes does not. If the further merge is impossible before reaching to the pre-defined the number of clusters g^*, we repeat Step 1 and 2 with an increased number of k by 1.

There are two possible strategies with regard to the resulting number of clusters. The first one is to set the number of clusters ahead as we do in other partitioning methods. We begin the proposed algorithm with $k = 1$ and repeat Step 1 and 2 until we reach to g^*. Since another g^* could appear in the subsequent procedures, it is recommended to go further with a larger k until the algorithm creates one single cluster. Then, we can see which clustering among several results, where all results have g^* clusters, works better than the others for our application. The second way is to obtain the m number of clusters when the merge step is stopped by $\eta_{st} = 0$. It is very likely that a small number of k on a large number of n results in many small clusters by this strategy. Therefore, although the connectivity is either zero or minimized at the given k, if the resulting number of clusters is (much) greater than expected from the given dataset, we need to repeat Step 1 and 2 with larger k's. What we believe from this strategy is that if there exists a clear structure of clusters then we perhaps can reach to zero or possible minimum connectivity at consecutive k's with the same number of clusters. This strategy is reasonable in respect that cluster analysis is basically an exploratory pattern analysis. We explain this using the following example. It also illustrates how the proposed

algorithm works. As can be seen in Figure 5.1, this example has the structure of three clusters ($g^* = 3$) with 30 objects where each cluster contains 10 objects.



(a) 3nn-closed sets (b) 1st merge (c) 2nd merge

(d) 3rd merge (e) 4th merge (f) 5th merge–stop

Figure 5.1: Illustration of *CNCLUST* through a synthetic example

This example starts with $k = 3$ and the resulting eight 3nn-closed sets are merged sequentially as the merge step goes. Each object in the figure is marked by a number that indicates its cluster membership. This number also shows the sequence of grouping. When we started with $k = 2$, the resulting number of clusters was 4 at zero connectivity meaning both that no more merge was allowed and that all objects satisfied the two consistencies. Figure 5.1f shows the first clustering result of $g^* = 3$. When the algorithm stopped, we arrived at zero connectivity and obtained the correct clustering. We resulted in the same clustering by $\{k = 4, 2\text{nd merge}\}$, $\{k = 5, \text{no merge}\}$, $\{k = 6, \text{no merge}\}$, ... , and $\{k = 9, \text{no merge}\}$. At $k = 10$, Step 1 created 2 clusters, and a single cluster was generated by only Step 1 with $k = 11$.

As shown in the description of the algorithm and the above example, the proposed method generates a series of cluster hierarchies and each hierarchy corresponds to each number of

nearest neighbors which is k. The example in Figure 5.1 shows an ideal case that several ends of cluster hierarchies met the pre-defined number of clusters g^*. However, with regard to the 2nd strategy for the resulting number of clusters, we do not have to define the g^* ahead, because it is in fact unknown in cluster analysis. Suppose that we do not know the true number of clusters in the above example. Since many of k's resulted in 3 clusters with the same structure of clusters at the end of each hierarchy, we are able to choose this cluster result with high reliability.

Again, the proposed method is flexible in respect that we can either predefine the number of clusters or come up with one or several promising number of clusters. Since cluster analysis is an exploratory pattern discovery method, we may want to go with the 2nd way; in sum, generate several cluster hierarchies, obtain several cluster results from the ends of hierarchies, and select a good clustering from those that has a reasonable number of clusters and provides a suitable explorability into the data.

5.3 Numerical Experiments

The goal of this section is to illustrate the proposed clustering approach on several simulated examples and to compare it to five existing clustering methods mentioned in Section 2.1 that are single linkage hierarchical clustering, complete linkage hierarchical clustering, k-means clustering, Partitioning Around Medoids (PAM), and model-based approach.

5.3.1 Clustering Methods

CNCLUST: The proposed clustering method. This method will group the given datasets by following its algorithm described in Section 5.2. The resulting cluster numbers will be determined as the algorithm suggests, rather than the predefined number of clusters, that is, the 2nd strategy. This method will therefore propose one or more of the most promising clustering results in our experiments.

Single linkage method: It is expected that this clustering method shows the most similar behaviors with *CNCLUST*, since it basically constructs the hierarchy of clusters by neighboring

objects in an agglomerative manner. We would like to see the difference between this method and the proposed method on our simulated datasets in respect that both of them are based on the nearest neighbor principle.

Complete linkage method: Another hierarchical approach which clustering behavior is quite different from the single linkage was adopted for the purpose of comparison. Since this method uses the largest distance between objects in two different clusters, it is called Furthest Neighbor approach. It is likely that *CNCLUST* may show its behavior lying between this method and single linkage.

***k*-means clustering**: One of the most often used clustering methods was included in our experiments. For the purpose of avoiding poor clustering results due to wrong selection of initial centroids in its algorithm, we perform a preliminary clustering phase on a random 10% subsample of the given dataset and then choose the initial centroids which are the mean values of clusters.

Partitioning Around Medoids (PAM): Since k-medoids clustering can be distinguished from k-means clustering by the fact that it selects data objects as centers and is thus more robust to outliers, PAM was selected as a k-medoids clustering approach. For this clustering algorithm, we used the function 'pam' in the package of *cluster* in R version 2.7.1 (Maechler et al., 2005).

Model-based clustering (Mclust): This method is distinctive from the other methods in our experiments in respect that it is a parametric clustering approach with Gaussian mixtures, so it clusters objects based on estimated densities rather than distances between objects. The function 'Mclust' in the package of *mclust* in R version 2.7.1 was employed to cluster objects (Fraley and Raftery, 2006).

Squared Euclidean distance was used as a distance measure between two objects for all clustering methods used in our experiments except the model-based clustering. So, we used a prepared dissimilarity matrix which elements are the squared Euclidean distances as the input argument of the 'pam' function.

5.3.2 Simulated Datasets

All simulated datasets are two-dimensional, since we believe that they are easy to understand their own special features, and thus make us convenient to recognize the results and interpret the resulting clusters. Moreover, it is superior to visually show the differences between the results of clustering from different approaches. Eventually, we considered 11 datasets and their short descriptions are provided as below. Illustration of each dataset can be seen at its corresponding result in Section 5.3.3.

Data 1: This dataset consists of four Gaussian-distributed clusters and it was generated from the Gaussian distributions with an identical covariance matrix, $\Sigma = 2.5\mathbf{I}_2$. Each cluster contains 80 objects centered at

$$\left\{ \mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mu_2 = \begin{pmatrix} 0 \\ 10 \end{pmatrix}, \mu_3 = \begin{pmatrix} 10 \\ 0 \end{pmatrix}, \mu_4 = \begin{pmatrix} 10 \\ 10 \end{pmatrix} \right\}.$$

We prepared this dataset as the most formal structure of clusters that may be easy for most of the clustering algorithms to figure out the true clustering.

Data 2: This dataset also contains Gaussian-distributed clusters. In addition to the three major clusters, we added several noise objects to make this dataset different from Data 1. This dataset was generated from Model 4 described in Section 4.3.1, but it is different in respect that each cluster has 50 objects and the number of noise objects is also 50. It may be difficult for neighboring-based approaches such as single linkage to discover the true structure of clusters due to noise objects.

Data 3: Two paralleled and elongated clusters are included in this dataset. The layered part of two elongated clusters may obfuscate some of clustering algorithms to cluster the objects correctly. The procedure for generating this dataset is very similar to Model 2 in Section 4.3.1. Let x_1 take 200 equal spaced values from -2 to 2 and set $x_2 = 0$ for all the resulting 201 objects. Gaussian noise with standard deviation 0.1 is then added to each variable. The second cluster is also generated in the same way except the setting of $x_2 = 1$. In order to shift each cluster to opposite directions, we add -1 to the first variable of the first cluster and 1 to the first variable of the second cluster.

Data 4: It is the same to Data 1 and 2 that we generate clusters from a set of Gaussian distributions. The difference, however, comes from that we consider various shapes, volumes, and orientations of Gaussian distribution for each cluster. This dataset consists of five clusters generated from the following Gaussian distributions. For each cluster C_i, $i = 1, 2, \ldots, 5$,

$$|C_1| = 120, |C_2| = |C_3| = |C_4| = |C_5| = 70,$$
$$\left\{ \mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mu_2 = \begin{pmatrix} -5 \\ -4 \end{pmatrix}, \mu_3 = \begin{pmatrix} 5 \\ 4 \end{pmatrix}, \mu_4 = \begin{pmatrix} -5 \\ 4 \end{pmatrix} \mu_5 = \begin{pmatrix} 5 \\ -4 \end{pmatrix} \right\},$$
$$\left\{ \Sigma_1 = \begin{pmatrix} 0.2 & 0 \\ 0 & 10 \end{pmatrix}, \Sigma_2 = \Sigma_3 = \begin{pmatrix} 0.8 & 0 \\ 0 & 0.8 \end{pmatrix}, \Sigma_4 = \begin{pmatrix} 0.8 & 0.6 \\ 0.6 & 0.8 \end{pmatrix}, \Sigma_5 = \begin{pmatrix} 0.8 & -0.6 \\ -0.6 & 0.8 \end{pmatrix} \right\}.$$

Data 5: This data is called 'Wreath data' because of its forming shape. It consists of 1000 objects generated from a fourteen-component Gaussian mixture where the covariance matrices of the components are of equal size and shape, but differ in orientation. We obtained this dataset from the *mclust* package in R software (Fraley and Raftery, 2006).

Data 6: A special structure of clusters is shown in this dataset which contains two spirals in two dimensions. Set $r_1 = r_2 = t$, with t taking 200 equal spaced values from 0.5 to 4.5. Also let θ_1 take the 200 equal spaced values from 0 to 4π, while θ_2 takes another 200 equal spaced values from π to 5π. The subscript i ($i = 1, 2$) indicates the cluster i. In order to convert the polar coordinates into Cartesian coordinates, we used the equations that are $x_1 = r\cos\theta$ and $x_2 = r\sin\theta$. It is expected that only the approaches of nearest neighboring will work for this dataset.

Data 7: This dataset includes three nested rings in two dimensions. From Model 3 in Section 4.3.1, we obtained this dataset. Similar to Data 6, non-neighboring methods may not perform well for this dataset.

Data 8: The shape of this dataset does look like a smile. Two clusters forming eyes were generated from Gaussian distributions, and the other cluster forming a mouth is a part of a generated circle. The circle can be generated in the same way of generating a ring in Model 3 in Section 4.3.1 or Data 7. Each eye has 50 objects and the mouth consists of 100 objects.

Data 9: Five arbitrarily shaped clusters are included in this dataset. We generated the coordinates of each object manually, not from a model. Total number of objects in this dataset

is 200.

Data 10: Each spherically shaped cluster, C_i, was generated from the following Gaussian distribution,

$$|C_1| = |C_2| = |C_3| = |C_4| = 80,$$
$$\left\{ \mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mu_2 = \begin{pmatrix} -5 \\ -4 \end{pmatrix}, \mu_3 = \begin{pmatrix} 5 \\ 4 \end{pmatrix}, \mu_4 = \begin{pmatrix} -5 \\ 4 \end{pmatrix} \mu_5 = \begin{pmatrix} 5 \\ -4 \end{pmatrix} \right\},$$
$$\Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_4 = \mathbf{I}_2,$$

and we then connected those so they form a rectangular figure. The x_1 coordinates of objects forming two horizontal lines and the x_2 coordinates of objects forming two vertical lines were generated from uniform distributions, and the x_2 coordinates of objects forming two horizontal lines and the x_1 coordinates of objects forming two vertical lines were generated from Gaussian distributions with a very small variance. Each line consists of 10 objects. Similar to Data 2, this dataset may obfuscate nearest-neighboring-based approaches such as single linkage method because of the connecting lines.

Data 11: This dataset is a simple combination of Data 1 and Data 7. Total number of cluster is, therefore, 7, including 4 Gaussian distributed clusters and 3 nested rings. Each Gaussian distributed cluster has 50 objects, and the number of objects for 3 rings is the same as in Data 7. We constructed this dataset for the purpose of having both spherically and non-spherically shaped clusters in one dataset.

5.3.3 Results

All clustering results will be shown in 2D plots where each object is represented by a number that indicates the corresponding cluster membership. Therefore, we can easily recognize each clustering result. They are provided in Figure 5.2 $\sim$ Figure 5.12. The below descriptions highlights the results in order to help us understand the figures.

Data 1: Four Gaussian-distributed clusters

Complete linkage, k-means clustering, PAM, and Model-based approach showed a good

clustering result, whereas single linkage method grouped two lower clusters into one and made a singleton cluster which is cluster 3. The result of single linkage shows its well-known drawback that this method is very sensitive to outliers because of its one-nearest-neighboring property. Although we increase the number of clusters to 5 or 6 based on the cluster hierarchy of single linkage, it is likely that another singleton clusters may be created. The 4th hierarchy of *CNCLUST* exactly found the four clusters from this dataset. As a reference, we also reported the 3rd and 5th hierarchy. The 3rd hierarchy of clusters generated 14 clusters after 51st merge, and it is shown that this result also figured out three major clusters.

Data 2: Three clusters with noise objects

k-means clustering, PAM, and model-based approach performed well as shown in Figure 5.3, although their results are a bit different each other. Complete linkage method grouped two major clusters together, and single linkage grouped all three clusters into one group and generated 2 very small clusters. However, although *CNCLUST* is basically built on a linkage framework, its results from the 3rd and 4th hierarchy have a good quality. It appears that the proposed method is more robust to noise objects than the two linkage methods on this dataset.

Data 3: Two paralleled and elongated clusters

CNCLUST, single linkage, and model-based clustering found two clusters correctly, while the other methods were confused on grouping objects in the layered part of two clusters. It is noticeable that the model-based approach estimated two narrow paralleled Gaussian distributions, so it clustered the objects correctly. As can be seen in the connectivity plot, the 6th, 7th, ..., 10th hierarchy identically resulted in the same two clusters.

Data 4: Five clusters with different shape, volume, and orientaion

Most of clustering algorithms failed to discover the true structure of clusters because of the long and thin cluster which is located at the center of the space. Single linkage was confused on distinguishing the cluster at the left and bottom from the middle cluster due to three ob-

jects at the bottom. Complete linkage, k-means, and PAM failed in a similar way due to the centered cluster. However, *CNCLUST* and model-based approach was completely successful to distinguish all five clusters from each other.

Data 5: Wreath data (14 clusters)

Due to the large number of objects, it is difficult to read the results directly from the figures. Alternatively, Table 5.1 shows the performances of clustering methods in terms of Adjusted Rand Index. See the table. In this case, model-based clustering performed best. Complete linkage and PAM, which are followed by k-means clustering, is comparable with model-based clustering. Although *CNCLUST* showed the worst performance, the clustering was not completely wrong. At the left side in the last plot of *CNCLUST*, it is shown that this method grouped four clusters into two clusters labeled as cluster 11 and 5.

Data 6: Two spirals

All clustering methods, except single linkage and *CNCLUST*, completely failed to reveal the true clusters structure form this dataset due to its very special distribution. Complete linkage, k-means, PAM, and model-based clustering cut the whole data into halves, whereas single linkage and CNCLUST figured out two spirals formed by linked objects. The connectivity plot shows that CNCLUST sequentially merged objects and stopped merging at 4 clusters when we set the number of neighbors to 1. With settings of $k = 2$ or $k = 3$, this method figured out 2 clusters only by constructing the $k$nn-closed sets, which is Step 1 in its algorithm.

Data 7: Three rings

Similar to the results on Data 6, this dataset was also difficult for four clustering methods, which are complete linkage, k-means, PAM, and model-based clustering, to figure out three rings. Nearest-neighboring-based approaches, which are single linkage and *CNCLUST*, performed well as shown in Figure 5.8.

Data 8: Smile data (3 clusters)

The results from complete linkage, k-means clustering, and PAM are very similar each other. It seems that they tried to group objects into a left, middle, and right cluster. We can notice that model-based approach tried to find three Gaussian distributions, which are the biggest left one, small right-upper one, and long right-lower one. Single linkage and *CNCLUST* found the correct clusters from this dataset.

Data 9: Hand-made data (5 clusters)

Complete linkage, k-means clustering, and PAM resulted in a very similar and poor clustering, whereas single linkage and *CNCLUST* performed perfectly. Model-based clustering also grouped well although it was not ideal. It seems that although the five clusters are shaped arbitrarily, the separation between clusters is spacious enough to cluster objects correctly by a good mixture of five Gaussian distributions in the case of model-based.

Data 10: Four connected clusters

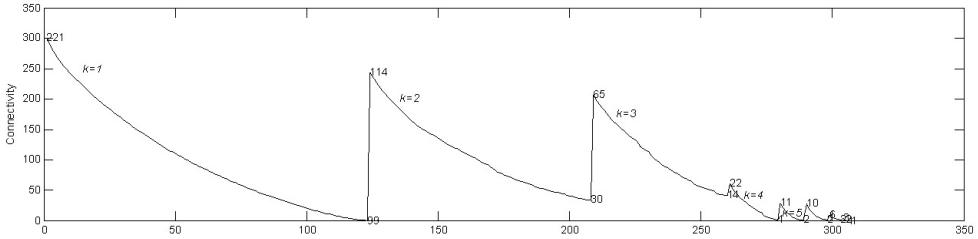
Single linkage divided the objects into two large clusters and two singleton clusters, while the other methods, which are complete linkage, k-means, PAM, and model-based clustering, correctly found the four major clusters. *CNCLUST* found the true clusters structure from its 4th hierarchy. The four connecting lines were unable to confuse the proposed method in this case, whereas the single linkage was obfuscated by them.

Data 11: Data 1 & Data 7 (7 clusters)

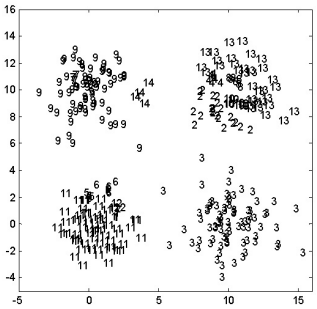
Single linkage grouped two inner rings together and generated one singleton cluster that is cluster 2, whereas *CNCLUST* correctly found all three rings and four spherically shaped clusters from its 7th cluster hierarchy. The results from 8th, 9th, and 10th hierarchy were same as shown in Figure 5.12. The other methods completely failed to discover the true 7 clusters. It is interesting to see that k-means clustering grouped the four left-upper clusters into one cluster due to the three rings located at the left-lower side, although each of them is

spherically distributed.

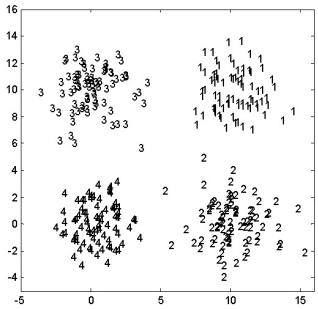
Above clustering results shown in the figures are also summarized in terms of Adjusted Rand Index (ARI), computed by Equation 2.29, in Table 5.1. Noise objects in Data 2 and the objects forming lines in Data 10 were excluded for calculating ARI. In general, *CNCLUST* performed well over the range of 11 simulated datasets, whereas some of other methods performed very poorly in some special cases. It was revealed from this simulation study that the proposed method not only grouped well any arbitrarily shaped clusters as single linkage method did, but also showed more robustness to noise objects than the single linkage.



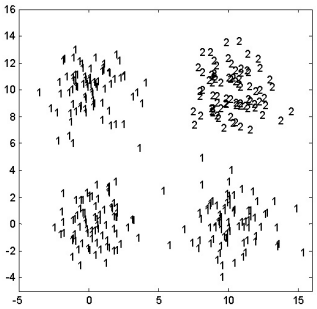
(a) Connectivity plot



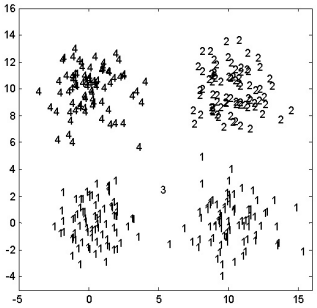
(b) *CNCLUST* (14 clusters); $k = 3$, after 51st merge



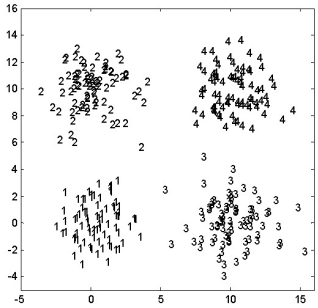
(c) *CNCLUST* (4 clusters); $k = 4$, after 18th merge



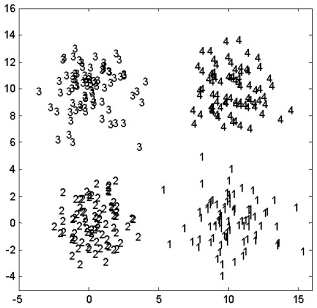
(d) *CNCLUST* (2 clusters); $k = 5$, after 9th merge



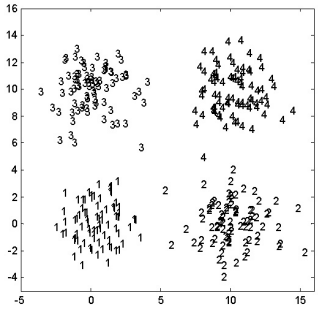
(e) Single ($g = 4$)



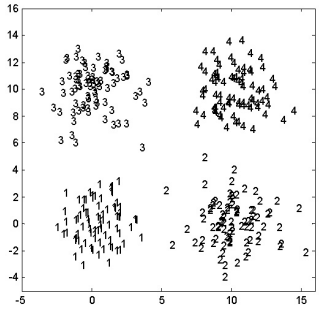
(f) Complete ($g = 4$)



(g) k-means ($g = 4$)

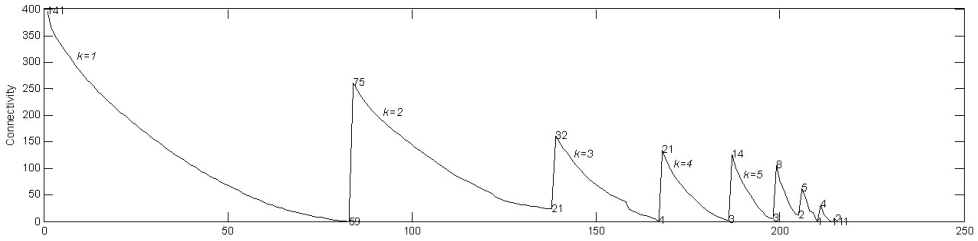


(h) PAM ($g = 4$)

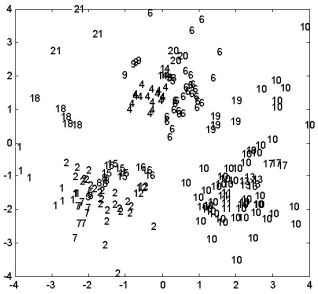


(i) Model-based ($g = 4$)

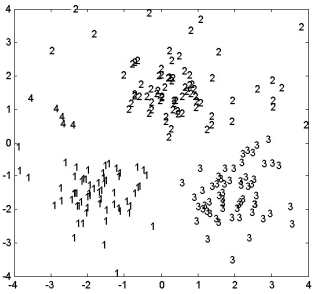
Figure 5.2: Clustering results on Data 1



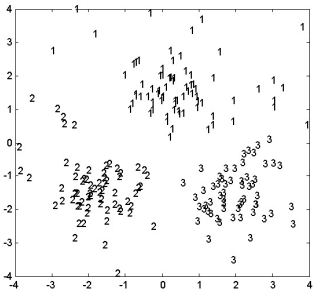
(a) Connectivity plot



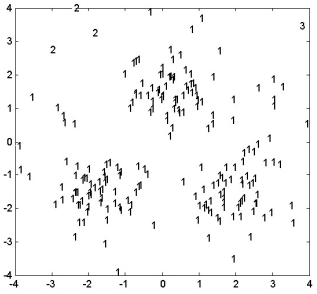
(b) *CNCLUST* (21 clusters); $k = 2$, after 54th merge



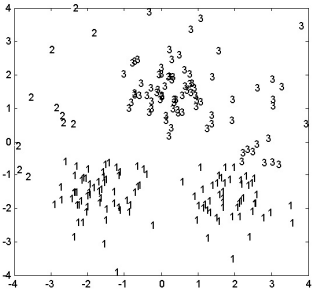
(c) *CNCLUST* (4 clusters); $k = 3$, after 28th merge



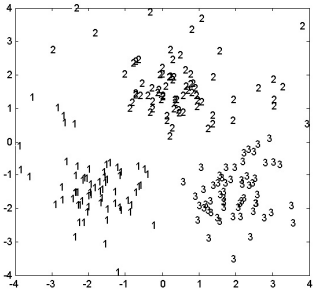
(d) *CNCLUST* (3 clusters); $k = 4$, after 18th merge



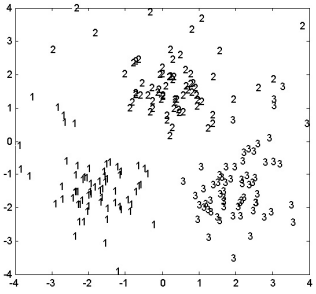
(e) Single ($g = 3$)



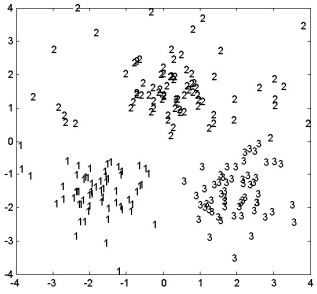
(f) Complete ($g = 3$)



(g) k-means ($g = 3$)

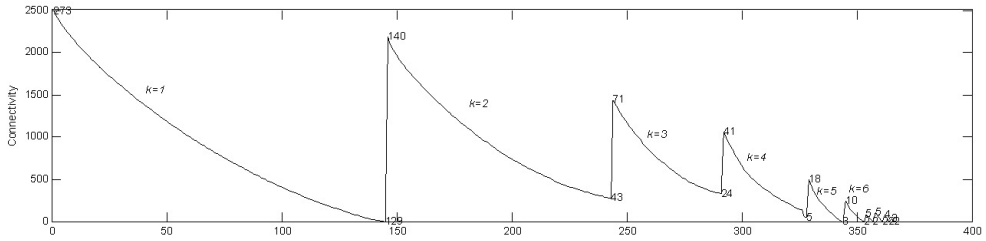


(h) PAM ($g = 3$)

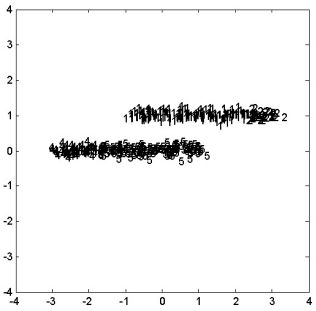


(i) Model-based ($g = 3$)

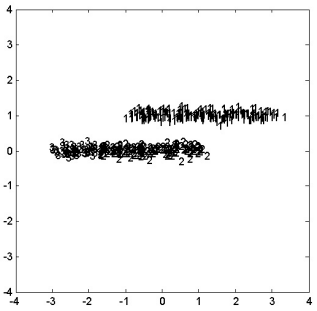
Figure 5.3: Clustering results on Data 2



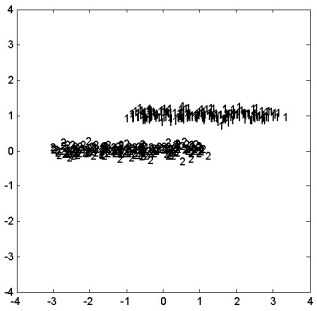
(a) Connectivity plot



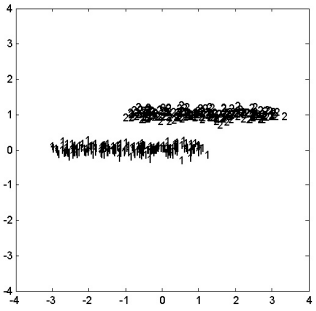
(b) *CNCLUST* (5 clusters); $k = 4$, after 36th merge



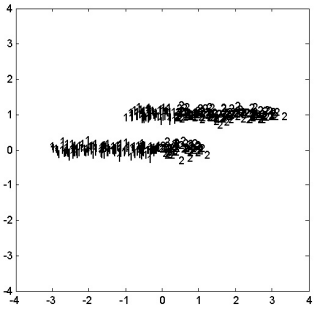
(c) *CNCLUST* (3 clusters); $k = 5$, after 15th merge



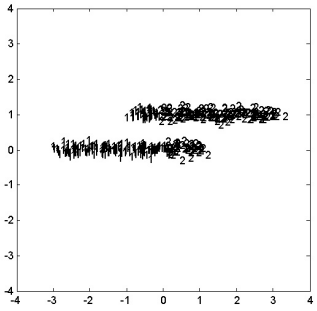
(d) *CNCLUST* (2 clusters); $k = 6$, after 8th merge



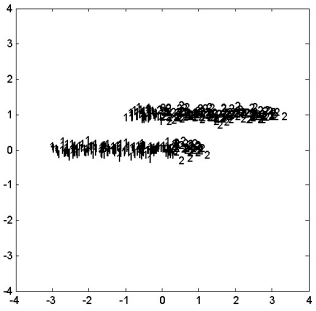
(e) Single ($g = 2$)



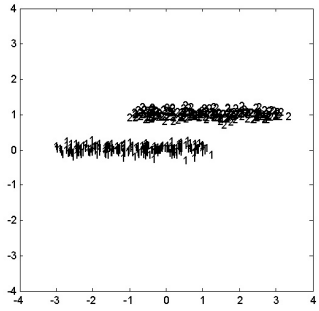
(f) Complete ($g = 2$)



(g) k-means ($g = 2$)

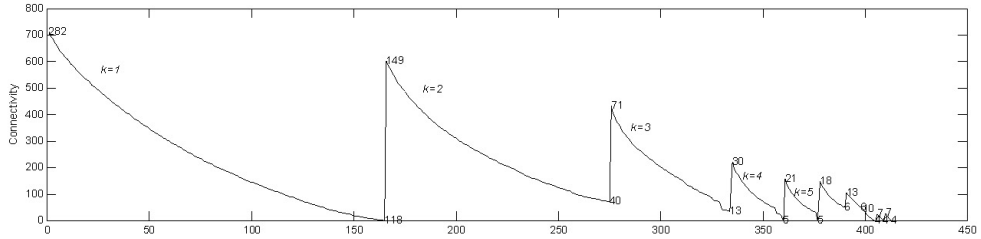


(h) PAM ($g = 2$)

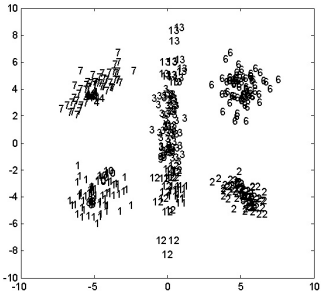


(i) Model-based ($g = 2$)

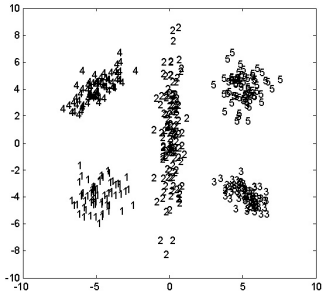
Figure 5.4: Clustering results on Data 3



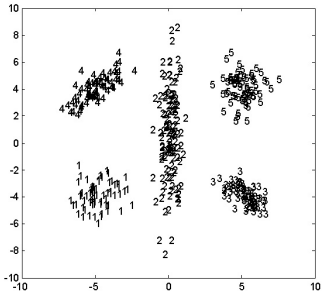
(a) Connectivity plot



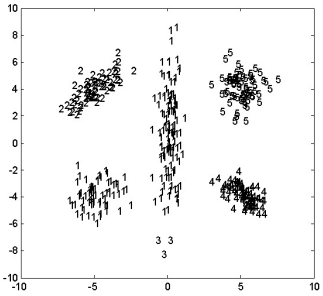
(b) *CNCLUST* (13 clusters); $k = 3$, after 58th merge



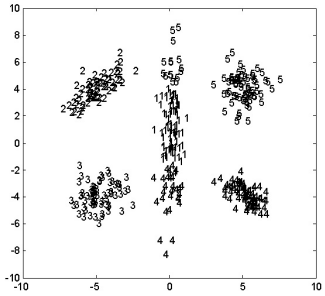
(c) *CNCLUST* (5 clusters); $k = 4$, after 25th merge



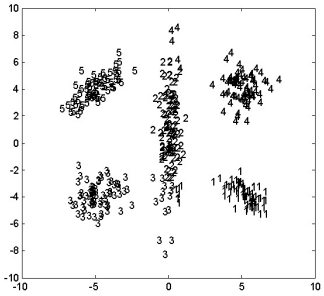
(d) *CNCLUST* (5 clusters); $k = 5$, after 16th merge



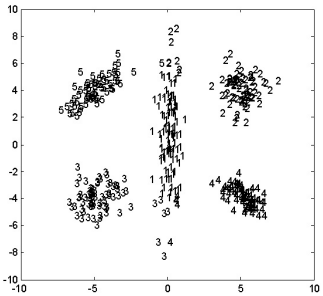
(e) Single ($g = 5$)



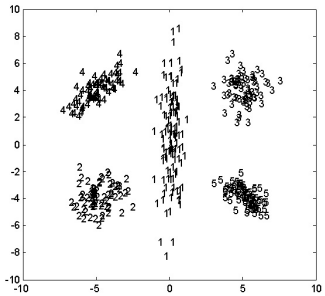
(f) Complete ($g = 5$)



(g) k-means ($g = 5$)

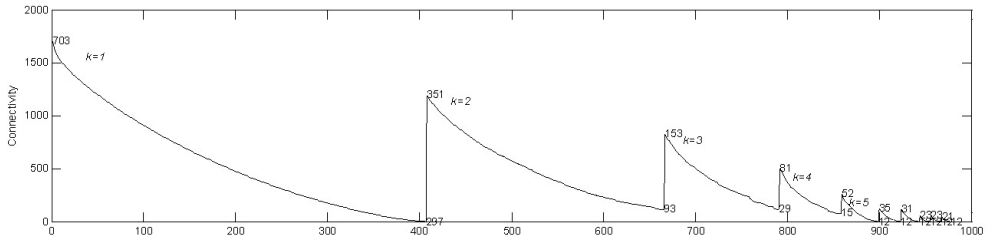


(h) PAM ($g = 5$)

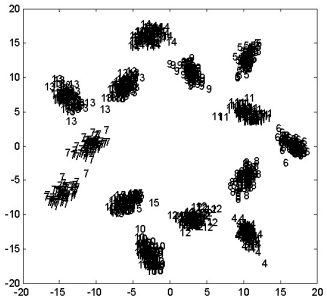


(i) Model-based ($g = 5$)

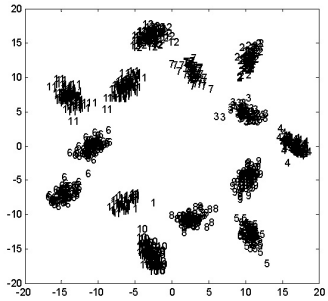
Figure 5.5: Clustering results on Data 4



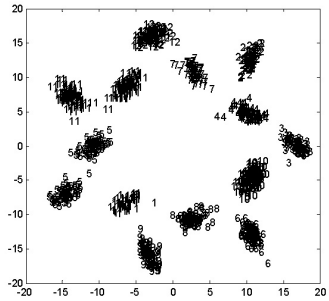
(a) Connectivity plot



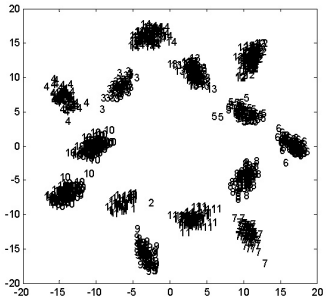
(b) *CNCLUST* (15 clusters); $k = 4$, after 66th merge



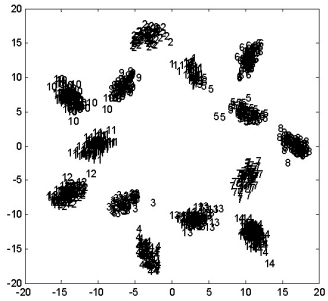
(c) *CNCLUST* (12 clusters); $k = 5$, after 40th merge



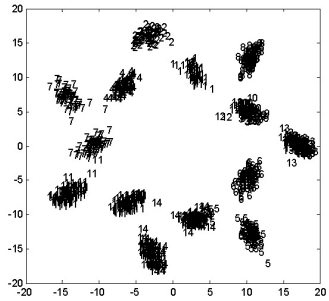
(d) *CNCLUST* (12 clusters); $k = 6$, after 23th merge



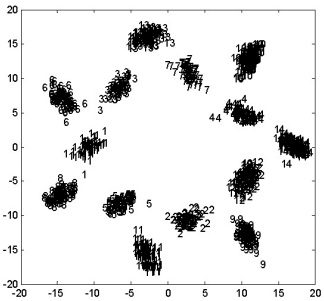
(e) Single ($g = 14$)



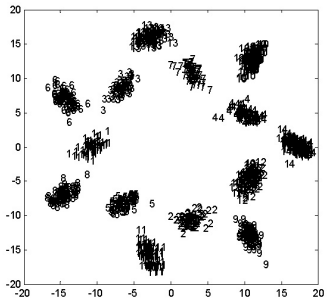
(f) Complete ($g = 14$)



(g) k-means ($g = 14$)

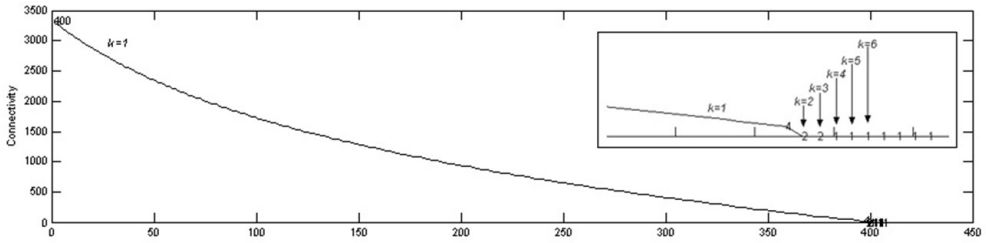


(h) PAM ($g = 14$)

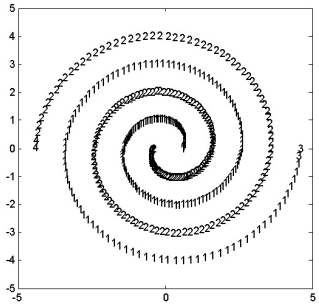


(i) Model-based ($g = 14$)

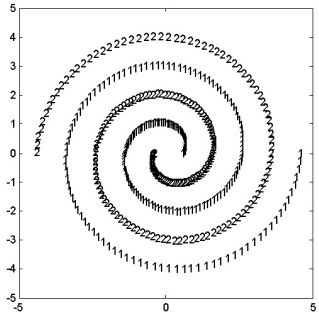
Figure 5.6: Clustering results on Data 5



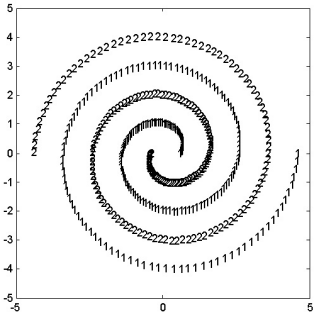
(a) Connectivity plot



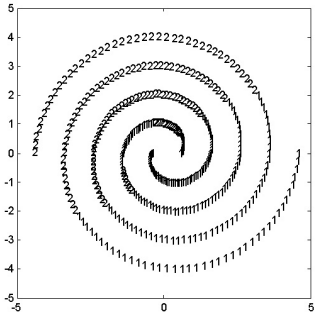
(b) *CNCLUST* (4 clusters); $k = 1$, after 396th merge



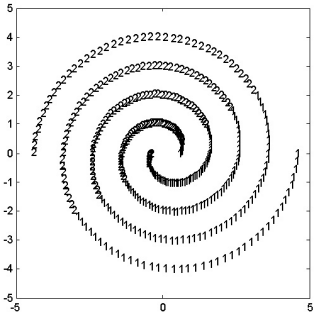
(c) *CNCLUST* (2 clusters); $k = 2$, no merge



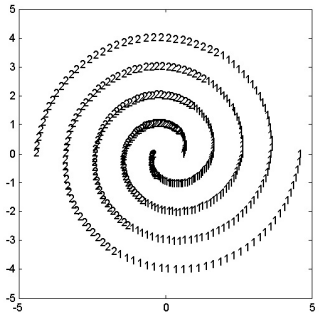
(d) Single ($g = 2$)



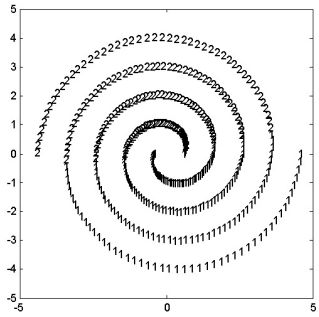
(e) Complete ($g = 2$)



(f) k-means ($g = 2$)

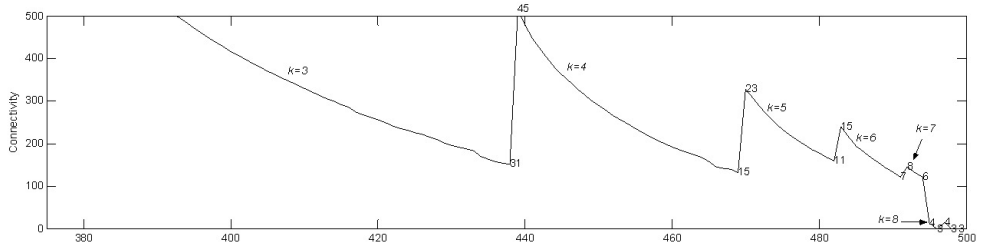


(g) PAM ($g = 2$)

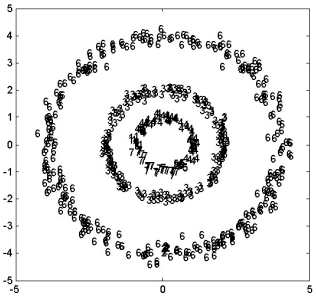


(h) Model-based ($g = 2$)

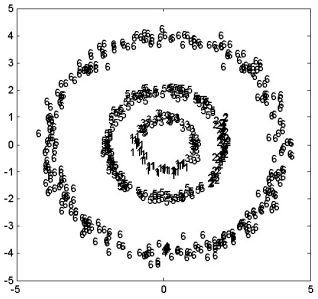
Figure 5.7: Clustering results on Data 6



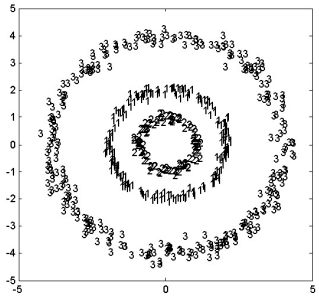
(a) Connectivity plot



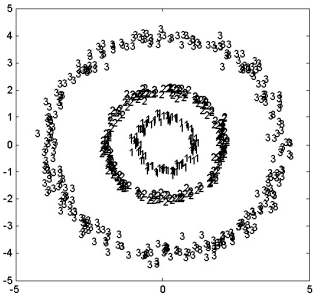
(b) *CNCLUST* (7 clusters); $k = 6$, after 8th merge



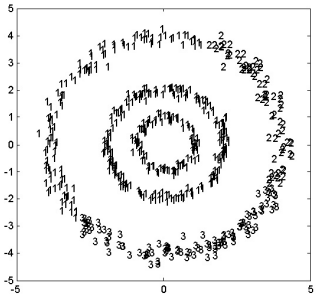
(c) *CNCLUST* (6 clusters); $k = 7$, after 2nd merge



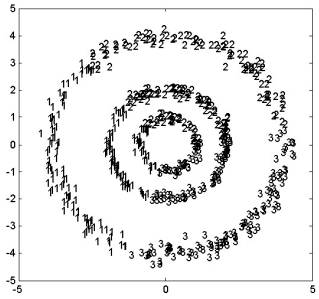
(d) *CNCLUST* (3 clusters); $k = 8$, after 1st merge



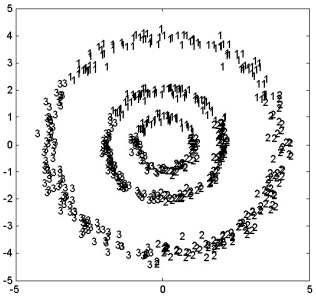
(e) Single ($g = 3$)



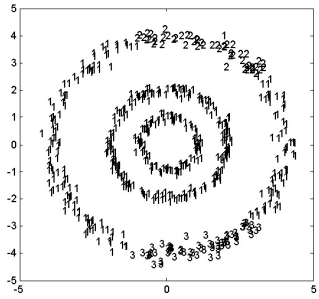
(f) Complete ($g = 3$)



(g) k-means ($g = 3$)

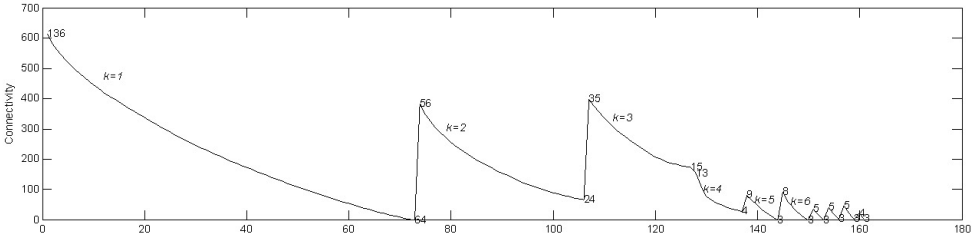


(h) PAM ($g = 3$)

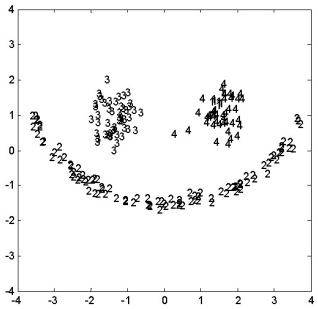


(i) Model-based ($g = 3$)

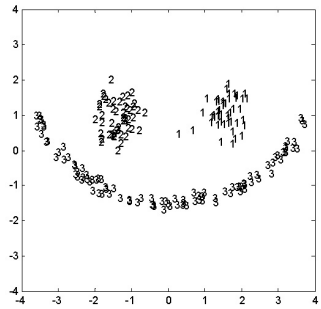
Figure 5.8: Clustering results on Data 7



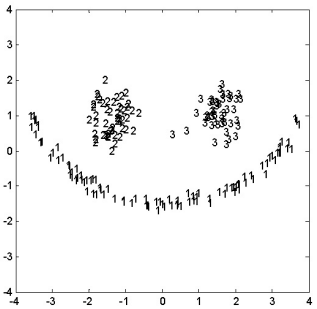
(a) Connectivity plot



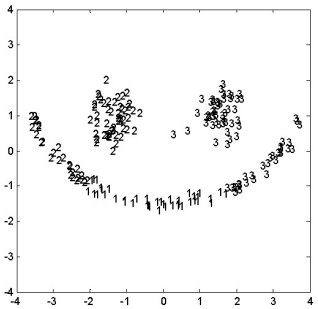
(b) *CNCLUST* (4 clusters); $k = 4$, after 9th merge



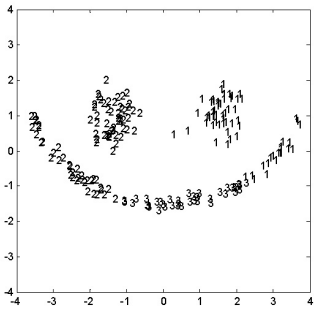
(c) *CNCLUST* (3 clusters); $k = 5$, after 6th merge



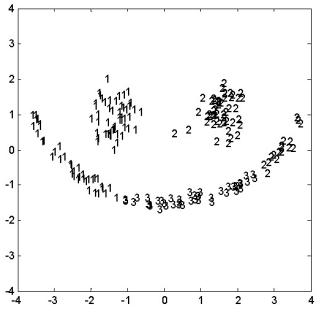
(d) Single ($g = 3$)



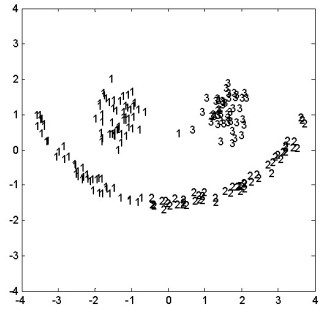
(e) Complete ($g = 3$)



(f) k-means ($g = 3$)

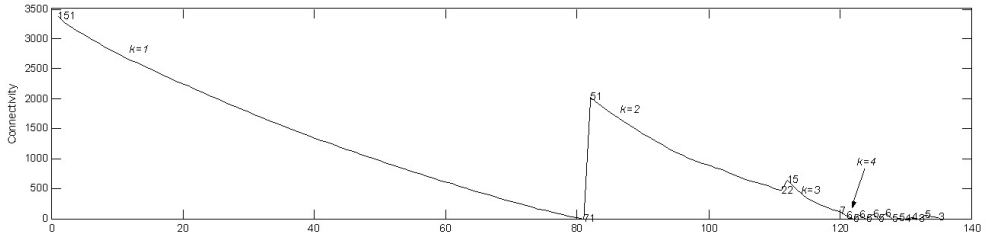


(g) PAM ($g = 3$)

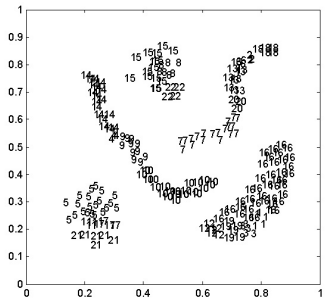


(h) Model-based ($g = 3$)

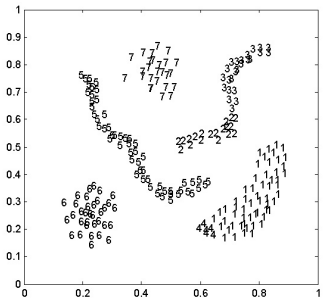
Figure 5.9: Clustering results on Data 8



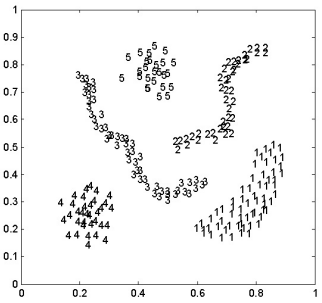
(a) Connectivity plot



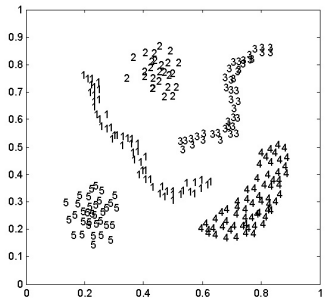
(b) *CNCLUST* (22 clusters); $k = 2$, after 29th merge



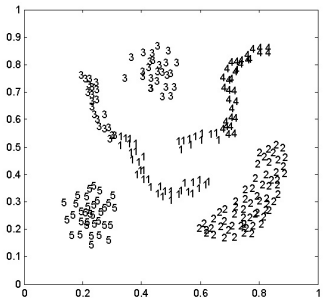
(c) *CNCLUST* (7 clusters); $k = 3$, after 8th merge



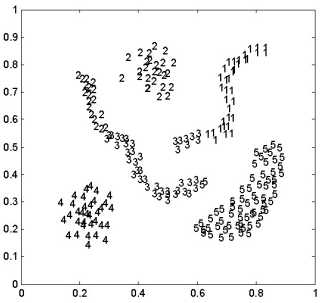
(d) *CNCLUST* (5 clusters); $k = 4$, after 1st merge



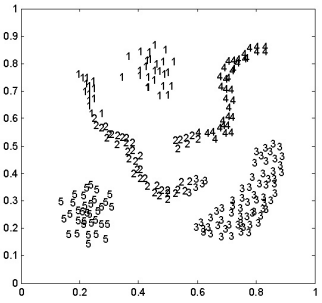
(e) Single ($g = 5$)



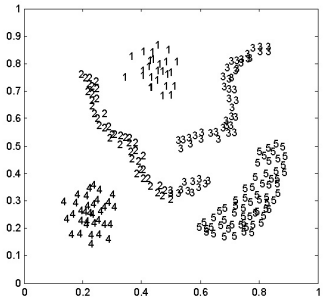
(f) Complete ($g = 5$)



(g) k-means ($g = 5$)

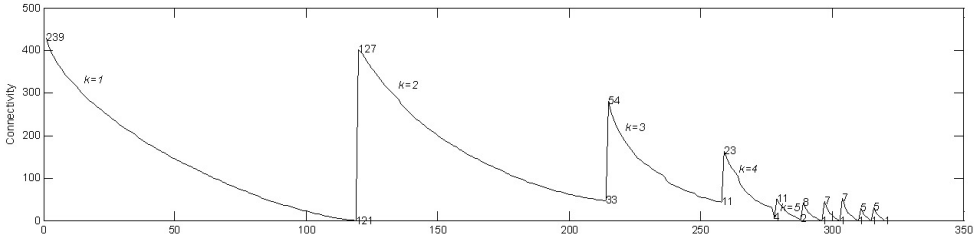


(h) PAM ($g = 5$)

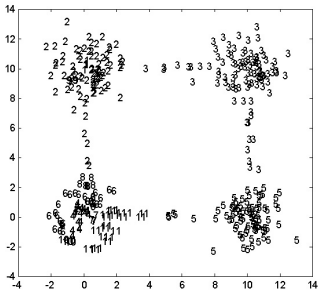


(i) Model-based ($g = 5$)

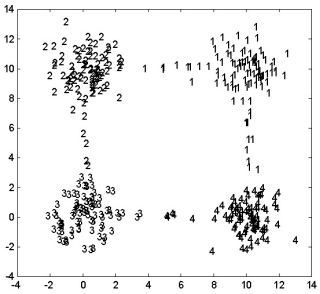
Figure 5.10: Clustering results on Data 9



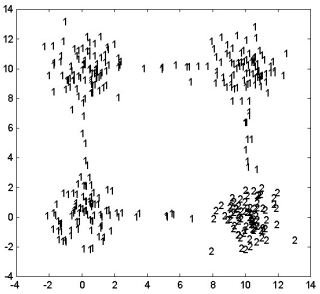
(a) Connectivity plot



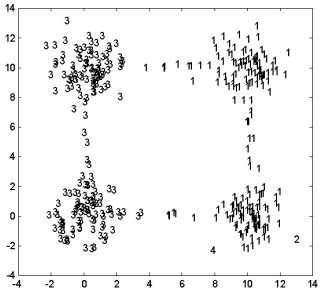
(b) *CNCLUST* (11 clusters); $k = 3$, after 43rd merge



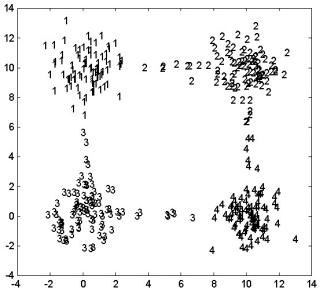
(c) *CNCLUST* (4 clusters); $k = 4$, after 19th merge



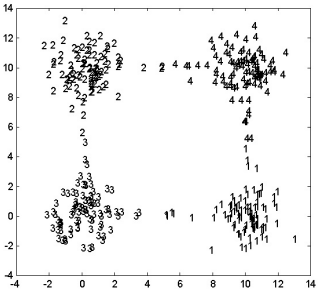
(d) *CNCLUST* (2 clusters); $k = 5$, after 9th merge



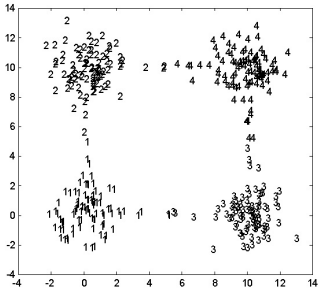
(e) Single ($g = 4$)



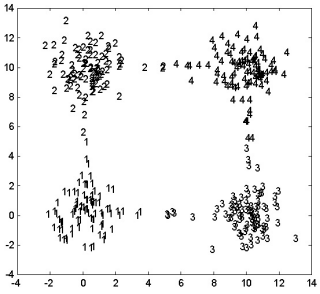
(f) Complete ($g = 4$)



(g) k-means ($g = 4$)

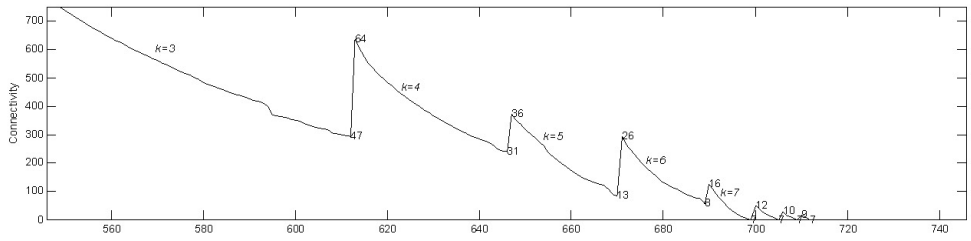


(h) PAM ($g = 4$)

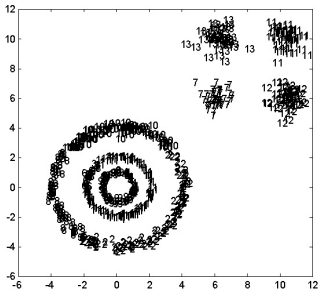


(i) Model-based ($g = 4$)

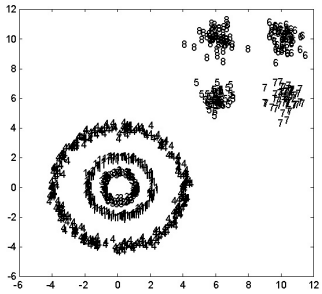
Figure 5.11: Clustering results on Data 10



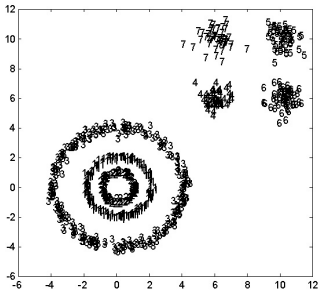
(a) Connectivity plot



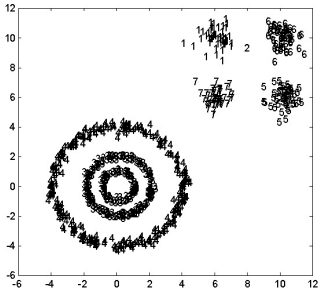
(b) *CNCLUST* (13 clusters); $k = 5$, after 23rd merge



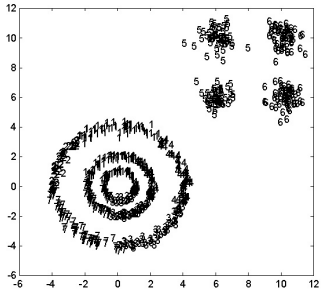
(c) *CNCLUST* (8 clusters); $k = 6$, after 18th merge



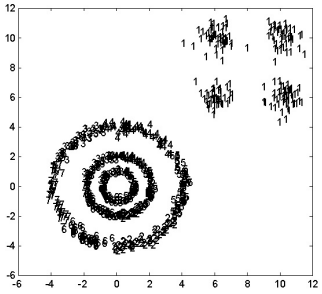
(d) *CNCLUST* (7 clusters); $k = 7$, after 9th merge



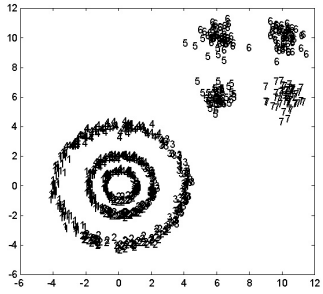
(e) Single ($g = 7$)



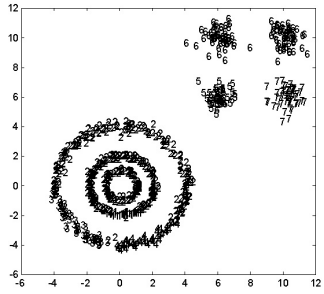
(f) Complete ($g = 7$)



(g) k-means ($g = 7$)



(h) PAM ($g = 7$)



(i) Model-based ($g = 7$)

Figure 5.12: Clustering results on Data 11

Table 5.1: Clustering results in terms of Adjusted Rand Index

	CNCLUST						Single	Complete	k-means	PAM	Model
	g^*	g^1	Adj. Rand	g^2	Adj. Rand	g^3	Adj. Rand			Adj. Rand	
Data 1	4	14	0.6705	4	0.9832	2	-0.1252	0.6887	0.9832	0.9832	0.9916	0.9832
Data 2	3	21	-0.1819	4	1.0000	3	1.0000	N/A	0.4233	1.0000	0.9799	1.0000
Data 3	2	5	0.4234	3	0.7544	2	1.0000	1.0000	0.3975	0.3088	0.3372	1.0000
Data 4	5	13	0.6306	5	1.0000	5	1.0000	0.7267	0.7280	0.8372	0.8619	1.0000
Data 5	14	15	0.8665	12	0.8839	12	0.8839	0.9326	0.9815	0.9111	0.9979	1.0000
Data 6	2	4	0.9800	2	1.0000	N/A	N/A	1.0000	0.0025	-0.0025	-0.0025	-0.0025
Data 7	3	7	0.9321	6	0.8268	3	1.0000	1.0000	0.0104	-0.0494	-0.0495	-0.4131
Data 8	3	4	0.9625	3	1.0000	N/A	N/A	1.0000	0.3146	0.3190	0.3384	0.4761
Data 9	5	22	-0.0656	7	0.8867	5	1.0000	1.0000	0.7556	0.7588	0.7637	0.8756
Data 10	4	11	0.7803	4	0.9916	2	-0.1252	0.3788	1.0000	1.0000	1.0000	1.0000
Data 11	7	13	0.3449	8	0.9816	7	1.0000	0.8401	-0.0149	-0.0915	0.0304	0.2944

5.4 Chapter Summary and Discussions

A new linkage-like clustering approach, namely *CNCLUST*, was proposed in this chapter. We employed a new measurement of cluster quality, which is the connectivity introduced in Chapter 3, as our objective function. In order to minimize this quantity we developed a greedy heuristic that reduces the amount of connectivity, which is directly determined when the number of nearest neighbors is given, as quickly as the algorithm goes. The method first constructs the nearest-neighbor-closed sets, and then sequentially merges two selected sets (clusters) in the way that this merge can reduce the possible largest amount of connectivity at each merge step. Since the structure of nearest neighbors of objects and the connectivity objective value vary at different number of nearest neighbors k, the method starts from $k = 1$ and patiently repeats the same steps, which are Step 1 and Step 2, with increment of k, until all objects in data are clustered together. Because of such a nature of the proposed method, it generates a series of cluster hierarchies where earlier hierarchies need more merge steps to reach the minimum connectivity while later hierarchies quickly stops within few merge steps. Therefore, the method tends to create many clusters in earlier hierarchies and few clusters in later hierarchies. Among those clustering results, we can select a good one that has a reasonable number of clusters, that is identically discovered by several k's, and that provides an interesting interpretation of data from the created cluster patterns. The proposed method is flexible in respect that we can also predefine the number of clusters, similar to the partitioning methods such as k-means.

Based on 11 two-dimensional datasets that were artificially generated, we examined the proposed method with 5 existing clustering approaches. We tried to design the 11 datasets with consideration of various shapes of clusters, number of clusters, and robustness to noise objects. It has been observed that *CNCLUST* in general performed well over the range of 11 simulated datasets, whereas some of other methods performed very poorly in some special cases.

In respect that the proposed method constructs clustering by connecting nearest neighbors, it is similar to the single linkage method. However, the major difference between these two

methods is that the proposed method first considers local compactness when forming $k$nn-closed sets and then attempts to realize global connectivity of objects, whereas single linkage only looks for the closest distance between objects. Moreover, since the proposed method tries to investigate a given dataset with different neighboring scheme, it examines various clustering hierarchies and thus may result in the best possible clustering solution in data that single linkage could neglect. Such differences sometimes give the proposed method more robustness than the single linkage, and it is shown in our experiments of Data 1, Data 2, and Data 10.

One of the known drawbacks of *CNCLUST* is that since it does not create one cluster hierarchy but come up with a series of hierarchies, it needs more computational efforts. A large number of objects may threaten the proposed method with this point. One possible strategy that we suggest to deal with such a problem is to start with a big enough number of k so that we do not waste our efforts to investigate some meaningless cluster structures. However, more rigorous approaches or modification of the algorithm will be needed.

CHAPTER 6. APPLICATION TO REAL DATA–FOODBORNE DISEASE OUTBREAKS

In this chapter we show an application of the proposed clustering method to a real dataset that contains information about foodborne disease outbreaks. This chapter consists of the following sections; an introductory section for briefly mentioning about the background of this application, a preliminary for describing the dataset and employed clustering methods, the association rule mining section, and the last section that shows the summarized results.

6.1 Background

Food safety is in fact very important part of public health, and although several advanced surveillance and monitoring systems exist in developed countries, outbreaks of foodborne diseases continue to be commonplace. Such foodborne diseases are caused by consumption of contaminated foods or beverages. Hence, an outbreak of foodborne illness occurs when a group of people consume the same contaminated food, and two or more of them come down with the same illness. The Center for Disease Control and Prevention (CDC) estimates that foodborne diseases cause 76 million illnesses, 325,000 hospitalizations, and around 5,000 deaths in the United States every year.

In most states, the diagnosed cases of certain serious infections are reported to the health department that also reports to CDC through the National Outbreak Reporting System (NORS). The reported data is investigated by CDC to obtain information regarding the role of food in the outbreaks. The surveillance of foodborne disease outbreaks can establish prevention and control measures in the food industry by identification of critical control points by the public health officials. The investigation of outbreaks also provides critical means for identifying new

and emerging pathogens, as well as maintains awareness about ongoing problems. Therefore, foodborne outbreak investigations made in a timely manner can lead us to rapid identification of corresponding etiologies and thus make us to take appropriate actions for prevention and control of diseases.

Although it is an important issue to come up with a methodology for identification of patterns of outbreaks, it has not been yet investigated through data mining techniques; nonetheless they have been successfully applied to many other areas. We therefore perform a cluster analysis for the purpose of extracting out previous unknown but interesting patterns that connect specific types of foodborne diseases outbreaks with associated foods and consumption locations. Five most common disease causing etiologies will be under our consideration, and they are namely *salmonella enteritidis*, *salmonella typhimurium*, *e. coli*, *norovirus*, and *scombroid toxin*. In order to improve our understanding of the resulting clusters, we also apply another data mining method, which is Association Rule Mining, to the same dataset.

6.2 Data Preparation and Clustering Methods

The data for this study was obtained from the Outbreak Surveillance Data from the CDC for the year 2006 (http://www.cdc.gov/). All the data was collected in a text form electronically through the Electronic Foodborne Outbreak Reporting System (EFORS) and all etiologies were as reported by the states. The raw Outbreak Surveillance Data was preprocessed in order to set up a clustering problem by converting a single text word into a binary variable. Hence, a text description of outbreak having several relative words was converted into a binary string, where 1 indicates that the reported record has the corresponding word and 0 says it does not. We finally acquired a flat data matrix consisting of 1's and 0's. In addition to these binary variables we have one more nominal variable which name is 'confirmed etiology', but this variable was not used for clustering. Total number of instances in this dataset is 1,167 and total number of variables is 107, where the variables are 106 binary variables and 1 nominal variable. All variable names are described in Appendix B with the zero-one distribution of each variable. The description of the nominal variable is also given in Appendix B. Note that the

etiologies in which we are interested are, again, *salmonella enteritidis*, *salmonella typhimurium*, *e. coli*, *norovirus*, and *scombroid toxin*.

Let $\mathbf{X} = (x_{ij})$ denote our binary dataset which is available for clustering. In order to cluster instances accordingly, we first need to define a distance measure between two instances. When defining the distance measure, we also need to consider the variation of dataset. We therefore counted total number of 1's in this dataset and found that the given dataset used for clustering is extremely sparse. Total number of 1's is 1,528, and the sparseness of the dataset is thus 98.76% ($= 1 - \frac{1,528}{1,167 \times 106}$). With consideration of these we define the distance between instance x_a and x_b as below:

$$d(x_a, x_b) = \frac{1}{1 + \sum_j x_{aj} \cdot x_{bj}}. \tag{6.1}$$

Hence, the distance in Equation 6.1 does not take account into conjoint absences, i.e. 0's, and it lies in the range of 0 to 1.

Table 6.1 and Table 6.2 show other descriptions of the dataset. As shown in Table 6.1, most of instances have 1 or 2 one's and only 62 instances among 1,167 have more than 3 one's. Due to this extreme sparseness of dataset, many of pairs of instances either have only single one in common or do not have any ones in common. Therefore, we needed to modify our clustering method in order to make it suitable to this dataset.

Table 6.1: Distribution of the dataset

Number of ones	Number of instances
0	104
1	681
2	320
3	44
4	16
5	1
6	1
Total	1,167

Table 6.2: Number of pairs of instances by ones in common

Ones in common (distance)	Number of pairs
0 (1.000)	488,689
1 (0.500)	190,500
2 (0.333)	1,161
3 (0.250)	10
4 (0.200)	1
Total	608,361

First, we did not cluster objects together if the distance between those is equal to or greater than 0.5, because grouping such objects does not give us any meaningful pattern. As we can infer from Appendix B, most of pairs have a one in common at the variable of 'restaurant/deli'. Second, we used only Step 1 of the proposed method, that is, we did not apply the merge step. In Step 1, we in fact stop the neighboring objects by not the number of neighbors which is k but the distance which is equal to or greater than 0.5. Hence, for some objects $\tilde{j} \in C_2(\hat{i})$ in Step 1(d) of the algorithm, we do not add all of its k neighbors but some of those having the distance (< 0.5), to $C(\hat{i})$. The stopping criterion of forming closed-sets, which is 'until $C_2(\hat{i}) = \emptyset$', is the same. Again, this modification does not enable us to define the number of nearest neighbors k, and thus we cannot fairly calculate η_{st} for merging two clusters together. Therefore, the modified algorithm is simply to construct nn-closed sets based on the distance constraint.

In addition to the proposed method, we also attempt to apply other clustering approaches. As for the hierarchical clustering methods, which are single linkage and complete linkage, we used a specified cutting criterion rather than the typical tree depth criterion, which is the distance constraint (< 0.5), for forming clusters from the resulting cluster hierarchy. In other words, we cut the connections, which are equal to or greater than 0.5, in dendrogram for generating clusters. It was to make those methods as comparable to the modified *CNCLUST* as possible by setting a similar criterion for forming clusters. Results will be reported in Section

6.4.

6.3 Association Rule Mining

Association Rule Mining finds interesting associations and correlated relationships among large sets of data items (Agrawal et al., 1993). In order to investigate what variables are related to each etiology, we applied another data mining method, which is ARM, to the foodborne disease dataset. Then, the generated association rules will be compared to our clustering results in order to improve our understanding of the results from two different pattern analyses.

According to the problem of ARM defined in Agrawal et al. (1993), let $I = \{i_1, i_2, \ldots, i_q\}$ denote a set of q binary variables called 'items' and $T = \{t_1, t_2, \ldots, t_p\}$ denote a set of transactions called the database. Each transaction in T has a unique transaction ID and contains a subset of items in I. A resulting rule is then defined as an implication of the form $A \Rightarrow B$ where $A, B \subset I$ and $A \cap B = \emptyset$. The sets of items A and B (itemsets) are called antecedent and consequent of the rule respectively. If we consider the raw format of our dataset where each outbreak is described as a set of words, we can notice that it is an appropriate input to ARM. The resulting rules are expected to have a relationship between relative variables and etiologies, e.g. 'location is wedding reception & food is potato salad $\Rightarrow$ etiology is *salmonella typhimurium*'.

An association rule has three measures that express the degree of uncertainty about the rule, and those numbers are used to select interesting rules from the set of all possible rules. The first measure as a probability is called the support for the rule that is given in Equation 6.2, and it is simply the portion of transactions that contain all items in the antecedent and consequent part of the rule.

$$Support(A \Rightarrow B) = P(A \cap B) \tag{6.2}$$

The confidence of the rule, which is the ratio of the number of transactions that include all items in the consequent as well as the antecedent to the number of transactions that include all items in the antecedent, can, by its definition, be interpreted as the probability of finding

the consequent part of the rule in transactions under the condition that these transactions also include the antecedent part. Therefore, the confidence is given by

$$Confidence(A \Rightarrow B) = P(B|A) = \frac{P(A \cap B)}{P(A)}. \tag{6.3}$$

The last measure, which is the lift of the rule, is the ratio of the confidence to the expected confidence. The expected confidence means the confidence if the antecedent part does not enhance the probability of occurrence of the consequent part. It is the number of transactions that include the consequent part divided by the total number of transactions. Hence, the lift value gives us information about the increase in probability of the consequent part given the antecedent part. By such a definition of the lift, a meaningful rule should have the lift value that is greater than one. A lift value that is greater than one means that when the consequent part happens it is more likely that the antecedent happens (positive association), whereas a lift value of less than 1 means that if the consequent happens it is less likely that the antecedent happens (negative association). The lift is calculated by

$$Lift(A \Rightarrow B) = \frac{Confidence(A \Rightarrow B)}{P(B)} = \frac{P(A \cap B)}{P(A) \cdot P(B)}. \tag{6.4}$$

Association rules are required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. To achieve this, association rule generation is a two-step process. First, minimum support is applied to find all frequent itemsets in a database. In a second step, these frequent itemsets and the minimum confidence constraint are used to form rules. While the second step is straight forward, the first step needs more attention. In order to implement this two-step process, *a-priori* algorithm is the most often used (Agrawal and Srikant, 1994).

Considering the sparseness of the dataset, we allowed enough tolerance for the support of a rule by setting the minimum support to 3. Only the rules having the lift value that is greater than 1 were under our consideration. Since our expectation is that the most useful rules are of the type 'if X and Y then Z', where X is a location information, Y is a food vehicle that caused the corresponding outbreak, and Z is a type of etiology, we chose three as the maximum

number of items for generating frequent item sets. No lower limit of the confidence was decided to prevent losing some interesting rules due to the sparseness of dataset.

Recall that association rule mining is an unsupervised learning method, that is, it will find relationships called association rules between any attributes. Some of those relationships will therefore not describe the etiologies of interest, so after generating all association rules, we prune them to only include those rules that include one of the target etiologies in the consequent (e.g. *salmonella enteritidis*, *salmonella typhimurium*, *e. coli*, *norovirus*, and *scombroid toxin* in the results). Hence, we expect these patterns to provide insights into what types of outbreaks (etiologies) are caused by specific types of food items and/or locations.

6.4 Results

An overall description of the results of clustering by our method is that we came up with greatly many singleton clusters and several small groups having few instances. We selected the most interpretable clusters among those, and they are shown in Table 6.3. The variables that describe the characteristics of each corresponding cluster are also included in the table. Based on our clustering results we could not claim any clusters that can represent either *salmonella enteritidis* or *salmonella typhimurium* due to somehow evenly distributed instances of those over the resulting clusters. However, it appeared that there are some clusters that mostly consist of outbreaks of the other etiologies. Major instances in Cluster 1 are of *e. coli* etiology. Three clusters, which are Cluster 2, 3, and 4, consist of the outbreaks of *norovirus* only. Cluster 5 and 6 has the instances of the *scombroid toxin* etiology except only one outbreak in those two clusters.

Only hierarchical methods could find several meaningful clusters, whereas the others, which are k-means, PAM, and model-based, completely failed to discover the hidden patterns of outbreaks. The result of single linkage was exactly same with the one of the proposed method. Similar to the proposed method, it might be undemanding for the single linkage to group the neighbored objects by its algorithm. Complete linkage found less number of meaningful clusters than the single linkage and the proposed. It might be due to its distance measure

Table 6.3: Major clusters created by *CNCLUST* and single linkage method

Cluster	Major etiology	Number of instances	Variables
1	*e. coli*	37	Ground beef, Lettuce, Private home, Restaurant or deli
2	*norovirus*	35	Ice tea, Sandwich submarine, Sandwich deli, Sandwich turkey, Salads multiple, Lettuce, Spring rolls, Vegetables, Restaurant or deli, Office setting, Private home, Workplace not cafeteria, School
3	*norovirus*	11	Ice, Restaurant or deli, Banquet facility
4	*norovirus*	9	Salad, Restaurant or deli
5	*Scombroid toxin*	8	Tuna, Restaurant or deli
6	*Scombroid toxin*	10	Fish mahi, Fish escolar, Restaurant or deli

between clusters which is the largest distance between two objects in different clusters. Hence, it grouped objects into one cluster in the case that the objects commonly share the same features.

What we tried with PAM was to observe the average silhouette width at varying number of clusters from 2 to 500 in order to first find a good number of clusters. However, the average silhouette width never met a positive value, meaning that the objects were not appropriately classified. Model-based clustering basically assumes a mixture of several Gaussian distributions and uses the Bayesian Information Criterion (BIC) to select the best model. Due to the binary variables of the foodborne dataset, it was impossible to find a good number of clusters by observing the BIC. k-means clustering also faced the same difficulty when we were trying to apply it to the given dataset. In summary, PAM, model-based, and k-means clustering commonly need to define the number of clusters first, and then run their own algorithms to create clusters. Since we failed to have a good cluster number, we could not obtain meaningful cluster results. In fact, we run PAM and k-means with several cluster numbers that we guessed, but the resulting clusters were very difficult to read and tricky to find some interesting patterns.

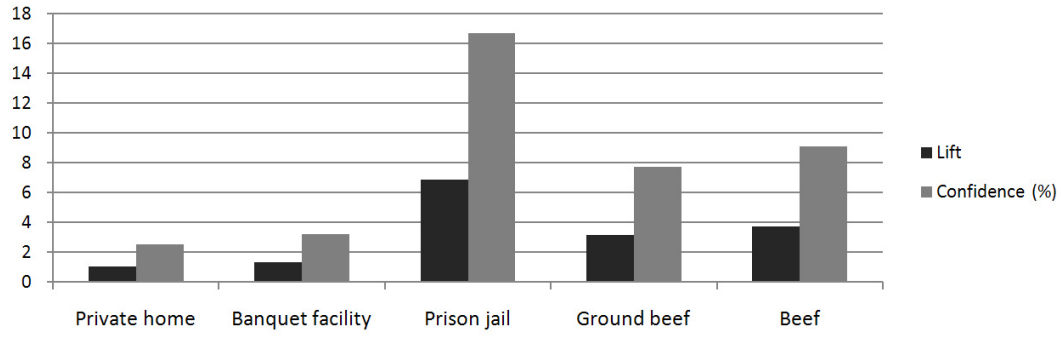


Figure 6.1: Selected variables with high confidence and lift (*salmonella enteritidis*)

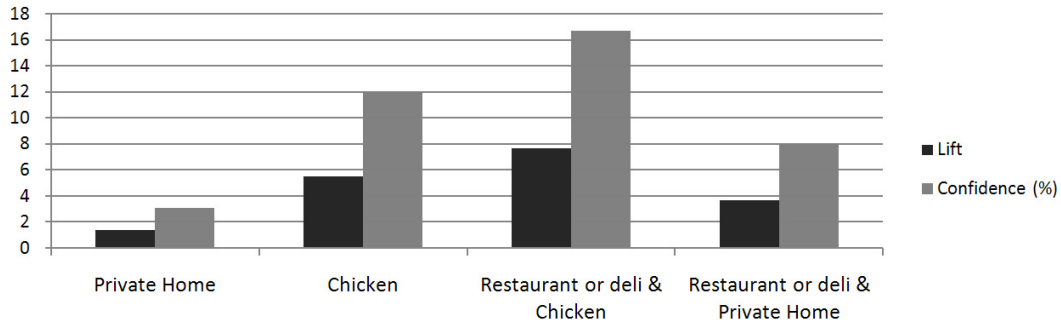


Figure 6.2: Selected variables with high confidence and lift (*salmonella typhimurium*)

Association Rule Mining generated several association rules linking the target etiologies in the consequent with location and/or food variables in the antecedent. We described some selected rules having the highest lift and confidence values in terms of bar chart in Figure 6.1 $\sim$ Figure 6.5. We can read the rules from the figure, for example, 'food is beef $\Rightarrow$ etiology is *salmonella typhimurium*' for the last two bars of beef in Figure 6.1.

A review of Figure 6.1 reveals that the lift value of prison/jail in which *salmonella enteritidis* was involved is approximately 6.5. This means that the probability that prison/jail will be involved in *salmonella enteritidis* is 6.5 times higher than the general probability of prison/jail in the dataset. Similar interpretations can be made on the other attributes, private home, banquet facility, ground beef, and beef. More noticeable patterns of outbreaks can be seen on two etiologies that are *e. coli* and *scombroid toxin*. The lift value of spinach is reaching to 30.0

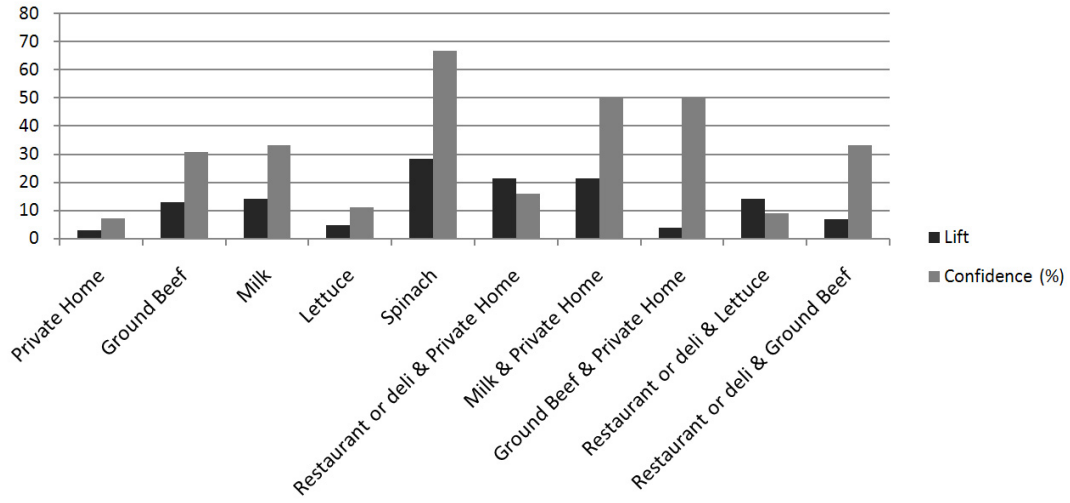


Figure 6.3: Selected variables with high confidence and lift (*e. coli*)

and the confidence of spinach is greater than 60% in Figure 6.3. It means that the rule, 'food is spinach ⇒ etiology is *e. coli*', is highly promising. The other selected attributes overall have very high lift values with good confidence numbers. As for the *scombroid toxin* etiology, all the chosen attributes have very high lift values that are greater than 20.0 and very high values of confidence as well. It is interesting to note that the selected attributes are related to fish products.

We will see how the resulting clusters of outbreaks by the proposed method are consistent with findings from the association rule mining. Representative attributes in each cluster corresponding to each etiology came from Table 6.3 and were then summarized again in the first column of Table 6.4. The table also shows the selected variables from the clustering results by single linkage and complete linkage.

Given the different nature of the clustering and ARM methods of extracting patterns, it is worth noting when the same pattern is found by two or more methods. As can be seen in the table, the results from the different approaches are quite consistent each other, in the case of *e. coli*, *norovirus*, and *scombroid toxin*. They lead us to more lucid findings for each type of etiology. As for the *e. coli*, the most commonly appeared food items are ground beef and lettuce. Private home and Restaurant/deli were chosen as the locations of

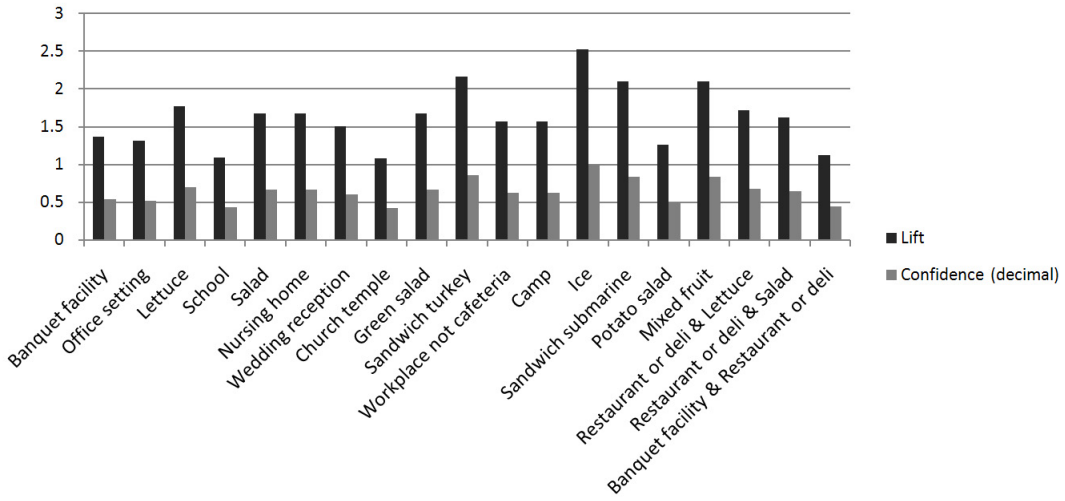


Figure 6.4: Selected variables with high confidence and lift (*norovirus*)

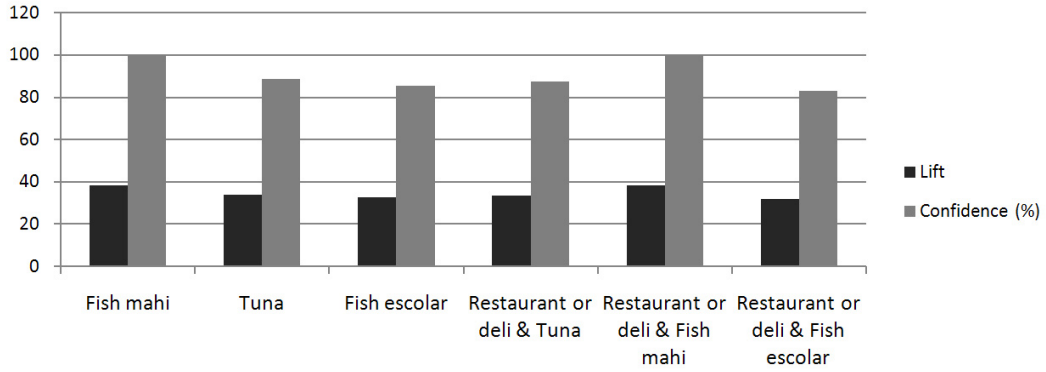


Figure 6.5: Selected variables with high confidence and lift (*scombroid toxin*)

e. coli by all three methods. It appears from the results of *CNCLUST*, single linkage, and ARM that the *norovirus* could contaminate many different food items in various places. The most frequently chosen food items are ice, lettuce, a series of sandwiches, and a category of salads. Since restaurant/deli appears frequently across the most of etiologies, except this, workspace-not-cafeteria and banquet facility seem the places of *norovirus*. It is clearly shown that the *scombroid toxin* is directly related to some marine products. In an outbreak of tuna consumption in restaurants, the disease can be attributed to the *scombroid toxin* etiology by all three clustering methods and ARM.

6.5 Chapter Summary

In this chapter, we applied the proposed clustering method to a real example which is about foodborne disease outbreaks. The purpose of the analysis of this foodborne disease dataset was to find interesting patterns that connect specific types of foodborne diseases outbreaks (etiologies) with associated foods and consumption locations. Due to the data sparseness, clustering the given dataset was a tough problem to any of clustering methods that we employed, and we thus made a modification to our algorithm in order to overcome the difficulty. The modified *CNCLUST* and single linkage method provided the best possible clustering solution in our analysis, and the result was quite consistent with the result of another data mining technique, which is Association Rule Mining. We believe that the knowledge found in this research will be useful to gain insight into the types of foods, food combinations and consumption locations that are more frequently linked to certain types of foodborne disease outbreaks.

Table 6.4: Selected variables for each etiology by four different methods

Etiology	*CNCLUST &* Single linkage	Complete linkage	Association Rule Mining
salmonella enteritidis	N/A	N/A	Ground beef Beef Prison/jail Private home Banquet facility
salmonella typhimurium	N/A	N/A	Chicken Private home Restaurant or deli
e. coli	Ground beef Lettuce Private home Restaurant or deli	Private home Restaurant or deli	Spinach Ground beef Milk Lettuce Restaurant or deli Private home
norovirus	Ice Ice tea Sandwich submarine Sandwich deli Sandwich turkey Salad Salads multiple Lettuce Spring rolls Vegetables Restaurant or deli Banquet facility Office setting Private home Workplace not cafeteria School	Ice Salad Lettuce Restaurant or deli Private home Banquet facility	Lettuce Salad Green salad Sandwich turkey Ice Sandwich submarine Potato salad Mixed fruit Banquet facility Office setting School Nursing home Wedding reception Church or temple Workplace not cafeteria Restaurant or deli
scombroid toxin	Fish mahi Tuna Fish escolar Restaurant or deli	Tuna Restaurant or deli	Fish mahi Tuna Fish escolar Restaurant or deli

CHAPTER 7. CONCLUSIONS

Since clustering is in part in the eye of beholder, we may not be able to assert a clearly agreed definition of cluster. Cluster has been considered as something that researchers believe as cluster. Despite of the ambiguity on its definition, people have agreed upon one principle that a good clustering should be compact, and they have been using the within-cluster sum of squares as its measurement that is called compactness in this research. What we would like to point out at this juncture is that most of works in clustering starts from measuring the quality of cluster. Hence, people first define what a good clustering is, and later work with it for solving other issues such as how to cluster objects and how to define the desirable number of clusters. It means that a goodness measurement of clustering should be indentified prior to developing any new clustering methods and/or any new approaches for estimating the number of clusters in data.

Apart from the compactness measure, perhaps there are several other criteria that we may want to achieve in data clustering. In this research we introduced the connectivity built on two preferred properties for a good clustering, which are called the cluster k-NN consistency and cluster k-MN consistency, and, with these measurements, also proposed new approaches for both estimating the number of clusters and clustering objects.

Both of the compactness and the connectivity are used in estimating the number of clusters. By measuring and evaluating these at different number of clusters, the proposed method suggests a set of promising candidates of cluster numbers and finally decides a single number by choosing a governing one. Since the two metrics are measured on several datasets generated from the original one with a little distortion, the final cluster number could be more reliable than a single estimate. We performed an intensive simulation study to evaluate the proposed

estimating method, and it has appeared that our approach nearly estimated correct cluster numbers not only for convex-shaped clusters but also for non-convex-shaped ones. Since none of previous studies in the research of identification of the number of clusters have considered a dataset that has the objects distributed non-spherically, we believe that this positive experimental result can lead us to a more worthwhile estimating methodologies based on and with the proposed scheme.

We were also able to see the potentialities that the connectivity could contribute on creating a new clustering algorithm. In developing a new clustering algorithm, what we were concerned about is that the resulting cluster solution should preserve the nearest neighbor consistencies of objects, that is, minimize the amount of connectivity defined in Chapter 3 as much as the algorithm can. With this clustering objective, the algorithm first considers a local compactness by forming nearest-neighbor-closed sets and later recognizes a global structure of cluster connectivity by grouping the sub-clusters. Both of the two steps attempt to minimize the connectivity, which is determined when the number of nearest neighbors is given, as much in earlier stages as possible. Based on 11 two-dimensional datasets, we investigated how the proposed method, named as *CNCLUST*, clusters objects. Similar to traditional single linkage method, *CNCLUST* was able to detect clusters of any arbitrary shapes; moreover it showed more robustness to outliers than the single linkage and the complete linkage because it considers not a single nearest neighbor object but the structure of nearest neighbors of objects. It has been observed that *CNCLUST* in general performed well over the range of 11 simulated datasets, while some existing methods failed to figure out true clusters in some cases. We also applied the proposed clustering algorithm with a little modification to a real example. In spited of very special structure of the given dataset, we found several interesting clusters that might be helpful to the application area. In addition, the clustering results were quite consistent with those from applying another data mining technique to the same dataset.

As mentioned at the end of Chapter 4 and 5, we end up with suggesting several future works for making this research wealthier. In spite of showing several benefits of the proposed estimating method and *CNCLUST* from numerical experiments, it will be required to present

how they work on some real benchmark datasets in clustering literature, for example, datasets in UCI repository (Asuncion and Newman, 2007). That may help readers/users who are interested in our methods to understand the methods. Second, both of the proposed methods are a sort of *ad hoc* approaches that do not have a theoretical foundation on each, although they are built on reasonable rationales. It must be valuable to clearly figure out analytical interpretations on the proposed methods, so we can guarantee their good performance, at least, on some particularly assumed structures of clusters. Lastly, with respect to the issue on computational efforts, a modification of algorithm or strategic suggestions may be required in order to cope with a large scale of clustering problems.

APPENDIX A. 50 SIMULATIONS ON DATA 10

We generated 20 synthetic datasets by *Populate*() function in this simulation model and counted how many times each number of clusters was chosen by the compactness ($\mathcal{CP}$) and the connectivity ($\mathcal{CN}$). The finally chosen number of clusters in each simulation was shown at the last column of the table.

Simulation		Number of clusters										g^* by	Final
		1	2	3	4	5	6	7	8	9	10	$\mathcal{CP},\mathcal{CN}$	g^*
1	$\mathcal{CP}$	0	0	20	0	0	0	0	0	0	0	3	3
	$\mathcal{CN}$	0	0	0	1	1	0	1	3	14	0	9	
2	$\mathcal{CP}$	0	0	5	0	0	0	0	0	15	0	9	9
	$\mathcal{CN}$	5	0	0	0	0	0	0	9	6	0	8	
3	$\mathcal{CP}$	0	0	19	0	0	0	0	0	1	0	3	3
	$\mathcal{CN}$	4	0	0	0	0	0	0	2	14	0	9	
4	$\mathcal{CP}$	0	0	0	0	0	0	0	0	20	0	9	9
	$\mathcal{CN}$	2	0	0	0	0	0	0	0	18	0	9	
5	$\mathcal{CP}$	0	0	11	0	0	0	0	0	9	0	3	9
	$\mathcal{CN}$	1	0	0	1	0	0	1	10	7	0	8	
6	$\mathcal{CP}$	0	0	7	0	0	0	0	0	13	0	9	9
	$\mathcal{CN}$	2	0	1	0	0	0	0	2	15	0	9	
7	$\mathcal{CP}$	0	0	0	0	0	0	0	0	20	0	9	9
	$\mathcal{CN}$	2	0	0	0	0	0	0	0	18	0	9	
8	$\mathcal{CP}$	0	0	20	0	0	0	0	0	0	0	3	3
	$\mathcal{CN}$	3	0	0	0	0	0	0	0	17	0	9	
9	$\mathcal{CP}$	0	0	7	0	0	0	0	0	13	0	9	9
	$\mathcal{CN}$	5	0	0	4	0	0	2	2	7	0	9	
10	$\mathcal{CP}$	0	0	18	0	0	0	0	0	2	0	3	9
	$\mathcal{CN}$	3	0	0	0	0	0	0	0	17	0	9	
11	$\mathcal{CP}$	0	0	19	0	0	0	0	0	1	0	3	3
	$\mathcal{CN}$	2	0	0	0	0	0	0	0	18	0	9	
12	$\mathcal{CP}$	0	0	20	0	0	0	0	0	0	0	3	3
	$\mathcal{CN}$	7	0	0	0	2	0	0	0	11	0	9	
13	$\mathcal{CP}$	0	0	18	0	0	0	0	0	2	0	3	3
	$\mathcal{CN}$	4	0	0	3	0	0	7	0	6	0	7	
14	$\mathcal{CP}$	0	0	18	0	0	0	0	0	2	0	3	3
	$\mathcal{CN}$	3	0	0	0	0	0	2	0	15	0	9	

Simulation		Number of clusters										g^* by	Final
		1	2	3	4	5	6	7	8	9	10	$\mathcal{CP}, \mathcal{CN}$	g^*
15	$\mathcal{CP}$	0	0	8	0	0	0	0	0	12	0	9	9
	$\mathcal{CN}$	2	0	0	0	0	0	0	9	9	0	8,9	
16	$\mathcal{CP}$	0	0	13	0	0	0	0	0	7	0	3	9
	$\mathcal{CN}$	4	0	0	0	2	0	0	0	14	0	9	
17	$\mathcal{CP}$	0	0	17	0	0	0	0	0	3	0	3	3
	$\mathcal{CN}$	7	0	0	0	1	0	0	0	12	0	9	
18	$\mathcal{CP}$	0	0	15	0	0	0	0	0	5	0	3	9
	$\mathcal{CN}$	2	0	0	0	2	0	0	0	16	0	9	
19	$\mathcal{CP}$	0	0	11	0	0	0	0	0	9	0	3	9
	$\mathcal{CN}$	1	0	0	0	0	0	0	0	19	0	9	
20	$\mathcal{CP}$	0	0	15	0	0	0	0	0	5	0	3	9
	$\mathcal{CN}$	3	0	0	0	4	0	0	2	11	0	9	
21	$\mathcal{CP}$	0	0	16	0	0	0	0	0	4	0	3	9
	$\mathcal{CN}$	3	0	0	0	2	0	0	0	15	0	9	
22	$\mathcal{CP}$	0	0	13	0	0	0	0	0	7	0	3	9
	$\mathcal{CN}$	1	0	0	1	0	0	3	7	8	0	9	
23	$\mathcal{CP}$	0	0	1	0	0	0	0	0	19	0	9	9
	$\mathcal{CN}$	4	0	2	0	0	0	0	0	14	0	9	
24	$\mathcal{CP}$	0	0	6	0	0	0	0	0	14	0	9	9
	$\mathcal{CN}$	2	0	0	0	0	0	1	0	17	0	9	
25	$\mathcal{CP}$	0	0	15	0	0	0	0	0	5	0	3	9
	$\mathcal{CN}$	3	0	0	0	0	0	0	1	16	0	9	
26	$\mathcal{CP}$	0	0	18	0	0	0	0	0	2	0	3	3
	$\mathcal{CN}$	3	0	0	0	1	0	0	1	15	0	9	
27	$\mathcal{CP}$	0	0	12	0	0	0	0	0	8	0	3	9
	$\mathcal{CN}$	2	0	6	1	0	0	0	0	11	0	9	
28	$\mathcal{CP}$	0	0	9	0	0	0	0	0	11	0	9	9
	$\mathcal{CN}$	5	0	0	0	0	0	0	0	15	0	9	
29	$\mathcal{CP}$	0	0	17	0	0	0	0	0	3	0	3	9
	$\mathcal{CN}$	2	0	0	0	0	0	0	0	18	0	9	
30	$\mathcal{CP}$	0	0	20	0	0	0	0	0	0	0	3	3
	$\mathcal{CN}$	6	0	1	0	0	0	0	0	13	0	9	
31	$\mathcal{CP}$	0	0	16	0	0	0	0	0	4	0	3	3
	$\mathcal{CN}$	2	0	0	1	1	0	3	1	12	0	9	
32	$\mathcal{CP}$	0	0	17	0	0	0	0	0	3	0	3	9
	$\mathcal{CN}$	4	0	0	0	0	0	0	0	16	0	9	
33	$\mathcal{CP}$	0	0	17	0	0	0	0	0	3	0	3	3
	$\mathcal{CN}$	2	0	4	0	0	0	0	5	9	0	9	
34	$\mathcal{CP}$	0	0	15	0	0	0	0	0	5	0	3	9
	$\mathcal{CN}$	3	0	0	0	0	0	0	0	17	0	9	
35	$\mathcal{CP}$	0	0	6	0	0	0	0	0	14	0	9	9
	$\mathcal{CN}$	4	0	0	0	0	0	0	0	16	0	9	
36	$\mathcal{CP}$	0	0	18	0	0	0	0	0	2	0	3	3
	$\mathcal{CN}$	2	0	0	0	2	0	0	8	8	0	8,9	
37	$\mathcal{CP}$	0	0	17	0	0	0	0	0	3	0	3	3
	$\mathcal{CN}$	5	0	0	0	2	0	0	0	13	0	9	
38	$\mathcal{CP}$	0	0	15	0	0	0	0	0	5	0	3	3
	$\mathcal{CN}$	1	0	0	0	0	0	0	9	10	0	9	

Simulation		Number of clusters										g^* by	Final
		1	2	3	4	5	6	7	8	9	10	$\mathcal{CP}, \mathcal{CN}$	g^*
39	$\mathcal{CP}$	0	0	17	0	0	0	0	0	3	0	3	9
	$\mathcal{CN}$	2	0	0	0	0	0	0	0	18	0	9	
40	$\mathcal{CP}$	0	0	12	0	0	0	0	0	8	0	3	9
	$\mathcal{CN}$	2	0	0	0	0	1	0	0	17	0	9	
41	$\mathcal{CP}$	0	0	20	0	0	0	0	0	0	0	3	3
	$\mathcal{CN}$	2	0	0	0	0	1	1	2	14	0	9	
42	$\mathcal{CP}$	0	0	19	0	0	0	0	0	1	0	3	3
	$\mathcal{CN}$	3	0	0	0	1	0	0	0	16	0	9	
43	$\mathcal{CP}$	0	0	2	0	0	0	0	0	18	0	9	9
	$\mathcal{CN}$	1	0	0	0	0	0	0	1	18	0	9	
44	$\mathcal{CP}$	0	0	12	0	0	0	0	0	8	0	3	9
	$\mathcal{CN}$	1	0	0	0	0	1	4	3	11	0	9	
45	$\mathcal{CP}$	0	0	5	0	0	0	0	0	15	0	9	9
	$\mathcal{CN}$	3	0	0	0	0	0	0	0	17	0	9	
46	$\mathcal{CP}$	0	0	5	0	0	0	0	0	15	0	9	9
	$\mathcal{CN}$	3	0	1	0	2	0	0	0	14	0	9	
47	$\mathcal{CP}$	0	0	19	0	0	0	0	0	1	0	3	3
	$\mathcal{CN}$	3	0	0	0	0	0	0	9	8	0	8	
48	$\mathcal{CP}$	0	0	17	0	0	0	0	0	3	0	3	9
	$\mathcal{CN}$	1	0	0	0	0	0	4	2	13	0	9	
49	$\mathcal{CP}$	0	0	14	0	0	0	0	0	6	0	3	9
	$\mathcal{CN}$	1	0	0	0	2	0	0	0	17	0	9	
50	$\mathcal{CP}$	0	0	19	0	0	0	0	0	1	0	3	3
	$\mathcal{CN}$	2	0	0	1	0	1	0	7	9	0	9	

APPENDIX B. VARIABLES IN FOODBORNE DISEASE DATA

Binary Variables

No.	Name	Ones	Zeros	No.	Name	Ones	Zeros
1	private home	197	970	54	French fries	2	1165
2	beef	11	1156	55	fruit	3	1164
3	black grouper	2	1165	56	fruit salad	4	1163
4	Caesar salad	3	1164	57	green salad	12	1155
5	cheese	4	1163	58	guacamole	2	1165
6	chicken	25	1142	59	ice	5	1162
7	clams	5	1162	60	ice tea	3	1164
8	coffee	1	1166	61	milkshake	5	1162
9	crab	4	1163	62	multiple salads	2	1165
10	deli meat	2	1165	63	ranch dressing	2	1165
11	fish escolar	7	1160	64	ribs pork	2	1165
12	fish mahi	9	1158	65	salsa	3	1164
13	fish roi	2	1165	66	sandwich club	2	1165
14	fried rice	3	1164	67	sandwich turkey	7	1160
15	ground beef	13	1154	68	seafood	2	1165
16	ham	3	1164	69	seafood pasta	3	1164
17	lettuce	27	1140	70	spring rolls	2	1165
18	macaroni cheese	2	1165	71	steak	2	1165
19	milk	9	1158	72	steak prime rib	2	1165
20	mixed fruit	6	1161	73	sushi	2	1165
21	mushrooms	4	1163	74	tuna	9	1158
22	peanut butter	2	1165	75	tuna salad	2	1165
23	pizza cheese	2	1165	76	vegetables	4	1163
24	pizza meat vegetable	2	1165	77	watermelon	4	1163
25	pork	17	1150	78	cake cheese	2	1165
26	potato salad	10	1157	79	chicken buffalo wings	2	1165
27	refried beans	3	1164	80	Picnic	16	1151
28	rice	5	1162	81	sausage	2	1165
29	salad	24	1143	82	workplace not cafeteria	39	1128
30	sandwich beef	3	1164	83	banquet facility	94	1073
31	sandwich chicken	4	1163	84	meatballs	2	1165
32	sandwich deli	8	1159	85	pasta salad	3	1164
33	sandwich submarine	6	1161	86	roast beef	2	1165
34	shrimp	8	1159	87	prison jail	6	1161
35	sphyraena barracuda	4	1163	88	turkey gravy	2	1165
36	spinach	3	1164	89	office setting	52	1115
37	tea	2	1165	90	chicken salad	3	1164

No.	Name	Ones	Zeros	No.	Name	Ones	Zeros
38	turkey	8	1159	91	workplace cafeteria	6	1161
39	bread	2	1165	92	School	37	1130
40	oysters	11	1156	93	beef meatball	2	1165
41	tomatoes	3	1164	94	sloppy Joe	2	1165
42	restaurant or deli	573	594	95	tortilla	2	1165
43	buffet	2	1165	96	temporary mobile	7	1160
44	chicken buffalo	2	1165	97	cafeteria	2	1165
45	chicken nuggets	2	1165	98	church temple	21	1146
46	chicken teriyaki	3	1164	99	hospital	9	1158
47	chips tortilla	2	1165	100	grocery store	5	1162
48	coleslaw	2	1165	101	nursing home	18	1149
49	crab salad	2	1165	102	camp	8	1159
50	egg rolls	3	1164	103	wedding reception	15	1152
51	ethnic style buffet	2	1165	104	prison jail	7	1160
52	fish	3	1164	105	day care center	3	1164
53	fish ahi	3	1164	106	workplace not cafeteria	8	1159

Confirmed Etiology Variable

Value	Number of instances
Bacillus cereus	13
Brucella spp	0
Campylobacter fetus	0
Campylobacter jejuni	14
Campylobacter unknown	10
Clostridium botulinum	4
Clostridium perfringens	34
E. coli., Enterohemorrhagic O121	0
E. coli., Enterohemorrhagic O157:H7	27
E. coli., Enterohemorrhagic O26	0
Listeria monocytogenes	2
Listeria unknown	0
Salmonella	2
Salmonella Agona	0
Salmonella Anatum	0
Salmonella Baildon	0
Salmonella Bareilly	2
Salmonella Berta	3
Salmonella Braenderup	0
Salmonella Enteritidis	28
Salmonella Group B	2
Salmonella Hadar	0
Salmonella Heidelberg	9
SalmonellaI 4,[5],12:i:-	4
Salmonella Java	2
Salmonella Javiana	4
Salmonella Miami	0
Salmonella Montevideo	3
Salmonella Muenster	1
Salmonella Newport	9
Salmonella Oranienburg	4
Salmonella Paratyphi B	0
Salmonella Potsdam	0
Salmonella Saintpaul	2
Salmonella Schwarzengrund	0
Salmonella Stanley	1
Salmonella Tallahassee	1

Value	Number of instances
Salmonella Tennessee	3
Salmonella Thompson	2
Salmonella Typhimurium	22
Salmonella Typhimurium var Copenhagen	3
Salmonella Uganda	1
Salmonella Weltevreden	1
Shigella flexneri	1
Shigella sonnei	8
Staphylococcus aureus	24
Vibrio parahaemolyticus	10
Bacillus other	0
E. coli., Enterotoxigenic Unspecified	0
Other bacterial	16
Salmonella Agona	2
Salmonella unknown	0
Staphylococcus unknown	5
Hepatitis A	3
Norovirus	457
Other viral	1
Ciguatoxin	10
Histamine	2
Monosodium glutamate (MSG)	0
Mushroom toxins	4
Neurotoxic Shellfish Poison	2
Other chemical	0
Plant toxins(Herbal toxins)	0
Scombroid toxin	30
Cleaning Agents	2
Puffer fish tetrodotoxin	1
Cryptosporidium parvum	4
Cyclospora cayatenensis	3
Giardia lamblia	2
Other parasitic	0
Trichinella spiralis	0
Bacillus cereus; Clostridium perfringens	17
Bacillus cereus; Staphylococcus aureus	1
Salmonella Enteritidis; Campylobacter jejuni	0
Salmonella Heidelberg; Salmonella Agona	0
Salmonella Newport; Salmonella Meleagridis	0
Salmonella unknown; E. coli., Enterohemorrhagic Unspecified	0
Unknown	349

BIBLIOGRAPHY

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 207–216.

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules in large databases. *Proceedings of the 20th International Conference on Very Large Data Bases*, 487–499.

Anderberg, M. R. (1973). *Cluster Analysis for Application*. Academic Press, New York.

Asuncion, A., & Newman, D. J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

Banfield, J. D., & Raftery, A. E. (1993). Model-based Gaussian and non-Gaussian clustering. *Biometrics*, *49*(3), 803–821.

Ben-Hur, A., Elisseeff, A., & Guyon, I. (2002). A stability based method for discovering structure in clustered data. in *Pacific Symposium on Biocomputing*, 6–17.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Brown, M. P. S., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C., Furey, T. S., Ares, M., & Haussler, D. (2000). Knowledge-based analysis of microarrary gene expression data using support vector machines. *Proceedings of the National Academy of Sciences*, *97*(1), 262–267.

Calinski, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, *3*(1), 1–27.

Casillas, A., Gonzalez de Lena, M. T., & Martinez, R. (2003). Document clustering into an unknown number of clusters using a genetic algorithm. *Lecture Notes in Computer Science*, *2807*, 43–49.

Celeux, G., & Govaert, G. (1995). Gaussian parsimonious clustering models. *Pattern Recognition*, *28*(5), 781–793.

Chan, P. K., Fan, W., Prodromidis, A. L., & Stolfo, S. J. (1999). Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems*, *14*(6), 67–74.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*(3), 321–357.

Collins, L. M., & Dent, C. W. (1988). Omega: A general formulation of the Rand Index of cluster recovery suitable for non-disjoint solutions. *Multivariate Behavioral Research*, *23*(2), 231–242.

Dudoit, S., & Fridlyand, J. (2002). A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, *3*(7), 0036.1–0036.21.

Ding, C., & He, X. (2004). K-Nearest-Neighbor consistency in data clustering: Incorporating local information into global optimization. *Proceedings of the 2004 ACM Symposium on Applied Computing*, 584–589.

Forsyth, D. A., & Ponce, J. (2003). *Computer Vision; A Modern Approach*. Prentice Hall.

Fowlkes, E. B., & Mallows, C. L. (1983). A method for comparing two hierarchical clusterings, Journal of the American Statistical Association. *Journal of the American Statistical Association*, *78*(383), 553–584.

Fraley, C., & Raftery, A. E. (2002). Model-baesd clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, *97*(458), 611–631.

Fraley, C., & Raftery, A. E. (2006). Mclust: Model-Based Clustering / Normal Mixture Modeling, R package version 3.1-5.

Fred, A. (2001). Finding consistent clusters in data partitions. *Lecture Notes in Computer Science*, *2096*, 309–318.

Hartigan, J. A. (1975). *Clustering Algorithms*. New York: Wiley.

Haussler, D. (1999). *Convolution kernels on discrete structures*. UC Santa Cruz.

Herbin, M., Bonnet, N., & Vautrot, P. (2001). Estimation of the number of clusters and influence zones. *Pattern Recognition Letters*, *22*(14), 1557–1568.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*(1), 193–218.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for Clustering Data*. New Jersey: Prentice Hall.

Jain, A. K., Duin, R., & Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(1), 4–37.

Jain, A. K., Murthy, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Reviews*, *31*(3), 264–323.

Kaufman, L., & Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. New York, Wiley.

Kim, J., Yang, J., & Olafsson, S. (2009). An optimization approach to partitional data clustering. *Journal of the Operational Research Society advance online publication*, 8 April 2009 (doi:10.1057/jors.2008.195).

Kittler, J. (1998). Pattern classification: Fusion of information. *Proceedings of International Conference on Advances in Pattern Recognition*, 13–22.

Kittler, J., Hatef, M., Duin, R. P., & Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(3), 226–239.

Kohonen, T. (2001). *Self-Organizing Maps*, Springer Series in Information Sciences, *30*, Springer.

Kothari, R., & Pitts, D. (1999). On finding the number of clusters. *Pattern Recognition Letters*, *20*(4), 405–416.

Krzanowski, W. J., & Lai, Y. T. (1985). A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics*, *44*(1), 23–34.

Lam, L., & Suen, C. Y. (1997). Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics*, *27*(5), 553–568.

Lee, J.-S., Chang, K.-S., Seo, S.-H., & Jun, C.-H. (2005). Defect pattern classification of semiconductor wafers using clustering and statistical analysis, 2005 INFORMS Annual Meeting.

Lee, J.-S., Jun, C.-H., Lee, J., & Kim, S. (2005). Classification-based collaborative filtering using market basket data. *Expert Systems with Applications*, *29*(3), 700–704.

Levin, A., Lischinski, D., & Weiss, Y. (2008). A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(2), 1–15.

Levine, E., & Domany, E. (2001). Resampling method for unsupervised estimation of cluster validity. *Neural Compuation*, *13*, 2573–2593.

Liew, A. W.-C., & Yan, H. (2005). Image segmentation based on adaptive cluster prototype estimation. *IEEE Transactions on Fuzzy Systems*, *13*(4), 444–453.

Lukashin, A. V., & Fuchs, R. (2001). Analysis of temporal gene expression profiles: Clustering by simulated annealing and determining the optimal number of clusters. *Bioinformatics*, *17*(5), 405–414.

Maechler, M., Rousseeuw, P., Struyf, A., & Hubert, M. (2005). Cluster Analysis Basics and Extensions, R package version 1-12.0.

May, J., Bastian, C. T., Taylor, D. T., & Whipple, G. T. (2001). Market segmentation of Wyoming snowmobilers. *Journal of Travel Research*, *39*(3), 292–299.

Moulin, L. S., da Silva, A. P. A., El-Sharkawi, M. A., & Marks II, R. J. (2004). Support vector machines for transient stability analysis of large-scale systems. *IEEE Transactions on Power Systems*, *19*(2), 818–825.

Nath, S. V. (2006). Crime pattern detection using data mining. *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 41–44.

Nakamura, E., & Kehtarnavaz, N. (1998). Determining number of clusters and prototype locations via multi-scale clustering. *Pattern Recognition Letters*, *19*(14), 1265–1283.

Olafsson, S., Li, X., & Wu, S. (2008). Operations research and data mining. *European Journal of Operational Research*, *187*(3), 1429–1448.

Olafsson, S., & Yang, J. (2005) Intelligent partitioning for feature selection. *INFORMS Journal on Computing*, *17*(3), 339–355.

Pelleg, D., & Moore, A. (2000). X-means: Extending K-means with efficient estimation of the number of clusters. *Proceedings of the 17th International Conference on Machine Learning*, 727–734.

Pham, D. L., Xu, C., & Prince, J. L. (2000). Current methods in medical image segmentation. *Annual Review of Biomedical Engineering*, *2*, 315–337.

Punj, G., & Stewart, D. W. (1983). Cluster analysis in marketing research: Review and suggestions for application. *Journal of Marketing Research*, *20*(2), 134–148.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, *66*(336), 846–850.

Salvador, S., & Chan, P. (2004). Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, 576–584.

Shapiro, L. G., & Stockman, G. C. (2001). *Computer Vision*, New Jersey, Prentice Hall.

Sheu, J.-B. (2002). A fuzzy clustering-based approach to automatic freeway incident detection and characterization. *Fuzzy Sets and Systems*, *128*(3), 377–388.

Shi, L., & Olafsson, S. (2008). *Nested Partitions Method: Theory and Applications*. Springer, New York.

Spath, H. (1980). *Cluster Analysis Algorithms*. Ellis Horwood, Chichester, UK.

Sugar, C. A., & James, G. M. (2003). Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association*, *98*(463), 750–763.

Sun, H., Wang, S., & Jiang, Q. (2004). FCM-based model selection algorithms for determining the number of clusters. *Pattern Recognition*, *37*(10), 2027–2037.

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society*, *63*(2), 411–423.

Tseng, G. C., & Wong, W. H. (2005). Tight clustering: A resampling-based approach for identifying stable and tight patterns in data. *Biometrics*, *61*(1), 10–16.

Vinod, H. D. (1969). Integer programming and the theory of grouping. *Journal of the American Statistical Association*, *64*(326), 506–519.

Wallace, D. L. (1983). A method for comparing two hierarchical clusterings: Comment. *Journal of the American Statistical Association*, *78*(383), 569–576.