FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Nadia Farhanaz AZAM
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S.
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Spectral Clustering: An Explorative Study of Proximity Measures

TITRE DE LA THÈSE / TITLE OF THESIS

Dr. Herna Victor
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Dr. Shirley Mills

Dr. Eric Paquet

Dr. Nathalie Japkowicz

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Spectral Clustering: An Explorative Study of Proximity Measures

by

Nadia Farhanaz Azam

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Computer Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-59872-6
Our file Notre référence
ISBN: 978-0-494-59872-6

Abstract

In cluster analysis, data are clustered into meaningful groups so that the objects in the same group are very similar, and the objects residing in two different groups are different from one another. One such cluster analysis algorithm is called the spectral clustering algorithm, which originated from the area of graph partitioning. The input, in this case, is a similarity matrix, constructed from the pair-wise similarity between data objects. The algorithm uses the eigenvalues and eigenvectors of a normalized similarity matrix to partition the data. The pair-wise similarity between the objects is calculated from the proximity (e.g. similarity or distance) measures. In any clustering task, the proximity measures often play a crucial role. In fact, one of the early and fundamental steps in a clustering process is the selection of a suitable proximity measure. A number of such measures may be used for this task. However, the success of a clustering algorithm partially depends on the selection of the proximity measure. While, the majority of prior research on the spectral clustering algorithm emphasizes on the algorithm-specific issues, little research has been performed on the evaluation of the performance of the proximity measures.

To this end, we perform a comparative and exploratory analysis on several existing proximity measures to evaluate their performance when applying the spectral clustering algorithm to a number of diverse data sets. To accomplish this task, we use a ten-fold cross validation technique, and assess the clustering results using several external cluster evaluation measures. The performances of the proximity measures are then compared using the quantitative results from the external evaluation measures and analyzed further to determine the probable causes that may have led to such results.

In essence, our experimental evaluation indicates that the proximity measures, in general, yield comparable results. That is, no measure is clearly superior, or inferior, to the others in its group. However, among the six similarity measures considered for the binary data, one measure (Russell and Rao similarity coefficient) frequently performed poorer than the others. For numeric data, our study shows that the distance measures based on the relative distances (i.e. the Pearson correlation coefficient and the Angular distance) generally performed better than the distance measures based on the absolute distances (e.g. the Euclidean or Manhattan distance). When considering the proximity measures for mixed data, our results indicate that the choice of distance measure for the numeric data has the highest impact on the final outcome.

Acknowledgements

I would like to express my deep gratitude to my Supervisor, Dr. Herna L. Viktor for her unconditional support and inspiration. Without her help and endless efforts to make topics interesting and easily understandable to me, I would not have been able to complete this thesis. Her suggestions and help with finding material on different topics related to my work and most importantly her encouragement throughout the entire study, will never be forgotten and always be appreciated.

I would like to thank my parents and my sister; without them I would have been lost. Their love, advice, support and encouragement have brought me where I am today.

My warm thanks also go to my friends and colleagues who have encouraged me over the course of this endeavour.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The amount of data collected from various sources is increasing. With the invention of new technologies, preserving this enormous volume of data for future reference and analysis has become more manageable. In contrast, the task of discovering underlying patterns and hidden information from data has become more challenging and complex. According to Witten et al. [76] - "*As the volume of data increases, inexorably, the proportion of it that people understand decreases, alarmingly*". As such, we need automated and practical tools and techniques to take full advantage of the information lying hidden in the data. This is where *Data Mining* techniques come to aid.

Data Mining is defined as the process of automatic discovery of hidden, interesting, and previously unknown patterns in data stored electronically [76]. Some of the benefits of mining data are to extract previously unknown information and use it to predict future trends, make decisions, categorize or group data to discover common characteristics, amongst others. In a widely used data mining technique, called *Clustering*, formally known as *Cluster Analysis*, data is analyzed from different perspectives and clustered into natural groups (clusters, subsets or partitions) of objects [25]. Objects located in the same cluster share many common characteristics. In contrast, objects located in two different groups are very different. One of the main objectives of a cluster analysis algorithm is to maximize the *within* cluster similarity and minimize the *between* cluster similarity. The difference between the objects is often measured by the *proximity* measures, such as, *similarity* or *distance* functions. Here, *similarity* is a measure that is used to determine how similar or close two objects are from one another, whereas *distance* is a measure that is used to determine the differences between two objects. Cluster analysis has been successfully applied within a wide spectrum of areas, including data mining, statistical data analysis, machine learning, pattern recognition, image analysis, and bioinformatics [35].

In recent years, a new family of cluster analysis algorithms, collectively known as the *Spectral Clustering* algorithms, has gained much interest in research communities. The algorithms in this category often combine well-defined steps, may be solved by standard linear algebra, and frequently outperform traditional cluster analysis algorithms such as K-means or Single Linkage [46]. Moreover, these algorithms have shown significant practical successes when applied to problems regarding image segmentation [58], speech separation [5], and cluster analysis of protein sequences [52]. One of the most important advantages of the spectral clustering algorithm is that the algorithm may be applied to a wide range of data types (i.e. numeric, categorical, binary, and mixed). Therefore, the spectral clustering algorithms are not sensitive to any particular data type. These algorithms consider the pair-wise similarity between the data objects to construct the *similarity* (also known as *proximity, affinity,* or *weight*) matrix. The eigenvectors and eigenvalues[1] of the similarity matrix are used to find the clusters [58], [50], [46]. A number of algorithms from this family are discussed collectively by Luxburg [46] and Meila et al. [66]. These algorithms mainly differ with respect to how the similarity matrix is manipulated and/or which eigenvalue(s) and eigenvector(s) are used to partition the objects into disjoint clusters. While significant theoretical progress has been made regarding the improvement of the spectral clustering algorithms as well as the proposal of new methods, or the application in new domains, little research has been performed on the selection of the proximity measures, which is a crucial step in constructing the similarity matrix. In this study, we evaluate the performance of a number of such proximity measures and perform an explorative study on their behaviour when applied to spectral clustering algorithms.

In this chapter we discuss the motivation of our study. We also present the thesis outline that briefly highlights the content of each chapter.

1.1 Motivation

Proximity measures, such as, similarity and distance functions, often play a fundamental role in cluster analysis [35]. Early steps of the majority of cluster analysis algorithms may require the selection of proximity measure and the construction of a similarity

[1]Definitions of several linear algebra terms and notations are provided in Appendix A.

matrix (if necessary). As previously mentioned, a spectral clustering algorithm considers a similarity matrix as an input to the algorithm. Most of the time, the similarity matrix is constructed from an existing similarity or distance measure, or by introducing a new measure specifically suitable for a particular domain or task. Selection of such measures, particularly when existing measures are applied, often require careful consideration as the success of these algorithms relies heavily on the choice of the proximity function [4], [46], [19].

Most of the previous studies on the spectral clustering algorithm use the *Euclidean distance* measure, a distance measure based on linear differences, to construct the similarity matrix for numeric feature type [58], [50], [66] without explicitly stating the consequences of selecting the distance measure. However, there are several different proximity measures available for numeric variable types. The *Manhattan distance* and the *Pearson Correlation Coefficient*[2] are some examples of proximity measures. To our knowledge, no in-depth evaluation of the performance of these proximity measures on spectral clustering algorithms, specifically showing that the *Euclidean distance* measure outperforms, has been carried out. As such, an evaluation and an exploratory study that compares and analyzes the performance of various proximity measures may help to provide important guidelines for researchers when selecting a proximity measure for future studies in this area. This study endeavors to evaluate and compare the performance of these measures and to imply the conditions under which these measures may be expected to perform well.

Objects in a dataset are represented by *attributes* (also referred to as *features* or *variables*). The attributes denote various properties and characteristics of the objects. Variables or attributes may be of different types. The most commonly used variable types in cluster analysis are *numeric* (also known as *continuous* or *interval-scaled*) and *binary* variables. Apart from these two variable types, there are several other variable types including *nominal*, *ratio-scaled*, and *ordinal* variables [39]. Different proximity measures are also introduced for each of the attribute types. In this study, we focus on the proximity functions of three types of variables. In addition to numeric and binary variables, we consider mixed variables, which are capable of incorporating more than one of the types of variables in their functions. Datasets with mixed variable types are common in practical applications. There are several ways to handle mixed variables during cluster analysis, some of them are discussed by Kaufman et al. in [39]. The authors suggest that the most practical way to deal with mixed variables is to combine

[2]Chapter 3 defines the similarity and distance measures in detail.

them together in one function and to perform single cluster analysis on the entire set of variables [39]. Spectral clustering algorithms are suitable in such cases where the dataset consists of attributes of variable types. The algorithms from this family are applied on the similarity matrix, not directly on the dataset. Therefore, as long as the dataset is converted into a similarity matrix, the algorithm may be able to find the clusters regardless of the attribute types.

In this study, we compare the performance of the proximity measures for datasets with three different variable types. There are several different algorithms available from this family. They may either form a hierarchical tree of clusters (hierarchical method) or may directly divide the dataset into a predefined number of clusters (partitional method). For this study, we consider an algorithm from each method. To compare the performance of the similarity and distance measures, we follow the fundamental steps of cluster analysis [35], [80]. As such, we perform spectral cluster analysis with the combination of each of the proximity measures on a number of datasets. We then evaluate the solutions from the spectral clustering algorithms by applying several cluster evaluation measures. The proximity measures are then compared and analyzed from the results of the cluster evaluation measures. We also perform a manual inspection of the clusters to ensure accuracy, verify the results, and analyze the composition of the clusters.

1.2 Thesis Outline

This thesis includes eight chapters that are organized as follows. In Chapter 2, we provide an overview of cluster analysis methods. We also include a detailed discussion of spectral clustering algorithms in this chapter. Chapter 3 presents an overview of the proximity measures for numeric, binary and mixed variable types. In Chapter 4, we present an experimental design that includes information about parameter settings as well as the methods used, and the approach followed, to perform the experiments successfully. The cluster evaluation measures used to assess the validity of clustering results are also introduced in this chapter. In Chapter 5, we evaluate and examine the results of our experiments when datasets consisting of binary variables are considered. Chapter 6 presents the result analysis of the datasets of mixed variable type. In Chapter 7, we present the results when the experiments are performed on the numeric datasets. Finally in Chapter 8, we conclude by providing a summary that highlights of our contributions and provide possible directions for future research.

Chapter 2

Introduction to Cluster Analysis

Cluster Analysis is an exploratory data analysis method in which objects are clustered into several natural groupings. This chapter presents an introductory review of cluster analysis as well as the methods used to perform this task. The chapter begins in Section 2.1 with a background study on cluster analysis. An introduction to several cluster analysis methods is presented in Section 2.2. Section 2.3 presents a review of spectral clustering algorithms which is an efficient graph-based cluster analysis method. We conclude the chapter with a summary in Section 2.4.

2.1 Cluster Analysis

The word *clustering* is defined as: *"a grouping of a number of similar things"* [65]. Here, the word *similar* refers to the objects present in the same group, which possess like characteristics. In data mining, the goal of cluster analysis methods is to cluster unlabeled data, with no or little prior information about the class labels, into groups, such that objects in the same subgroup are very similar to one another and objects in two different subgroups are very different [76] [32] [16]. Let D be a dataset with n objects. When a cluster analysis algorithm is applied to this dataset D, it groups the data in $C_1, C_2 ... C_k$ clusters given that the total number of clusters is k. The main objective of a cluster analysis method is to minimize the distance between the objects located in the same cluster and to maximize the distance between the objects located in different clusters. Figure 2.1 (a) depicts a sample dataset in a 2-dimensional space and Figure 2.1 (b) shows the clusters marked with circles when $k = 3$. The results after applying a cluster analysis algorithm show that the clusters are generated in such a way

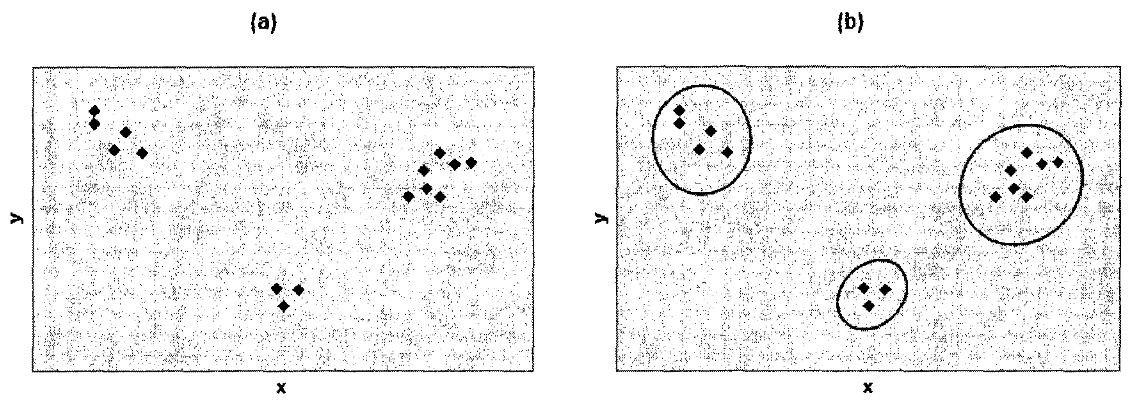


Figure 2.1: Clustering example: (a) input data and (b) the clusters.

so that the objects in each cluster are very close to one another. However, in the real world, the datasets are not as simple as the one depicted above. The objects are not always so clearly separated and the clusters are not usually as well-defined. Moreover, the datasets may contain hundreds or even thousands of objects and the feature space of these objects may also be very high dimensional. As a result, the task of clustering is often more complex and challenging.

In the following subsection we discuss the fundamental steps of a typical cluster analysis task.

2.1.1 Cluster Analysis Procedure

Cluster analysis methods usually follow a number of sequential steps [35], [80]. Figure 2.2 illustrates the basic steps of a cluster analysis procedure as discussed in [80]. According to Xu et al. [80], the four main steps that most clustering algorithms follow are: 1) feature selection or feature extraction, 2) design or selection of cluster analysis algorithm, 3) cluster validation, and 4) interpretation of results. We briefly discuss each of the four components below.

Feature Selection or Extraction: In practical applications, datasets often contain a large number of features to represent the objects. However, not all the features are useful for the learning process. Most of the time, there are several features that are irrelevant or redundant to the cluster analysis process. According to

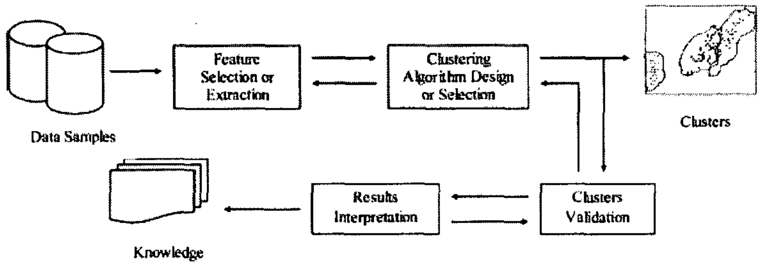

Data Samples
Feature Selection or Extraction
Clustering Algorithm Design or Selection
Clusters
Clusters Validation
Results Interpretation
Knowledge

Figure 2.2: Sequential procedure of a cluster analysis process [80].

Witten et al. [76] experimental studies show that, adding such features to the cluster analysis process usually deteriorates the performance of the algorithms. As such, techniques such as *Feature Selection* and *Feature Extraction* often prove to be useful to carefully reduce the size of the original feature set. According to Jain et al. [35] *Feature Selection* is the process of identifying the most effective subset of features from the original feature set. In contrast, *Feature Extraction* is the process of producing a new set of features by performing transformations on the original feature set [80], [35]. Both of the processes reduce the feature size by removing the redundant or irrelevant features and in doing so, simplify the clustering process. For more information on feature selection we suggest [40], [82] and for feature extraction, we refer to [24], [13].

Design or Selection of Cluster Analysis Algorithm: This step involves the selection of a proximity measure and a cluster analysis algorithm. The selection of a proximity measure directly affects the formation of the clusters. As mentioned previously in Chapter 1, one of the commonly used distance measures is the *Euclidean distance* measure. There are, however, a number of other proximity measures available in the literature which we discuss in detail in Chapter 3. In addition to the selection of a proximity measure, the results from cluster analysis also vary depending on the clustering algorithm that has been selected [35]. Several algorithms partition the data into a predefined number of groups (i.e. K-means), whereas other algorithms output a nested series of clusters [35]. Some of the algorithms are suitable for large datasets, whereas other methods handle outliers better. We discuss various cluster analysis methods in Section 2.2.

Cluster Validation: Given a dataset, a cluster analysis algorithm will always produce clusters [35], [80]. Moreover, as we mentioned above, different algorithms and proximity functions may produce different results. Therefore, it is necessary to assess the results to compare, evaluate, and measure the goodness of the cluster analysis methods. There are several evaluation and validation measures proposed in the literature that help to perform such an assessment. According to Jain et al. [35] and Xu et al. [80], these cluster validation measures are categorized into three groups: 1) *external measures*, 2) *internal measures*, and 3) *relative measures.* The external measures consider the prior knowledge about the data (i.e. class labels) against the cluster analysis results for the assessments. In contrast, the internal measures compute the assessment without any reference to the external information; they only consider the information present in the original dataset. The relative measures perform the evaluation by comparing the results from various cluster analysis methods with one another.

Interpretation of Results: The ultimate goal of any cluster analysis task is to partition the data into *meaningful* groups. As such, in this step, domain experts often analyze the clusters to discover the hidden patterns among the objects in a cluster and to assign a label to the clusters based on the underlying patterns.

2.1.2 Limitations

In Chapter 1, we mentioned a number of application domains to which the cluster analysis algorithms are often applied. The areas include data mining, machine learning, pattern recognition, bioinformatics, image processing, and many others. Nevertheless, when the cluster analysis techniques are applied to real-world datasets, several problems arise. In this section, we briefly state the drawbacks of cluster analysis as addressed by Dunham in [16].

- One of the main difficulties that arise with respect to a cluster analysis task is to correctly and automatically determine the number of clusters k. In cluster analysis, most of the time the prior knowledge or additional information about the data is not available to the users. As such, the algorithms that require the number of clusters k as input need special consideration. Intuitively, providing an incorrect value for k may result in unsatisfactory results. For instance, selecting a smaller value for k may over-generalize the results as it will try to combine natural clusters to achieve the user-specified number of clusters. In contrast, if k is set to a very high value it may decompose the natural clusters into many smaller subsets to achieve the desired number of clusters. Both the cases will have significant impact on the results.

- Interpreting the clustering results or more specifically, interpreting the clusters, is also considered to be one of the major problems in cluster analysis. As class labels are not available during the process, it may not always be possible to correctly interpret the semantic meaning of each of the individual clusters without any domain-specific knowledge.
- Handling outliers is another fundamental problem in cluster analysis. In a dataset, outliers are objects that are very different from the other objects in the dataset, and as such, they usually form their own clusters. Placing an outlier in a cluster that contains objects that are very different from it (i.e. to achieve the desired number of clusters), may result in the formation of poor clusters [16].
- Because dynamic data change over time, cluster membership may also change over time and therefore requires careful consideration to accommodate the changes.
- Another problem that may be encountered during the cluster analysis process, is that there may be no exact or correct answer to the clustering solution. Given a dataset, different algorithms may return different sets of clusters. Moreover, different users may also have different views and therefore may interpret the clusters differently. These difficulties may make the decision making task more complex and ambiguous.
- Finally, with the increasing amount of data, problems surrounding high dimensionality and handling of large datasets have also become a point of concern.

However, these problems also open the door to new research ideas. Various algorithms have been proposed to solve one or more of these problems efficiently. In the next section, we provide an overview of the cluster analysis methods and briefly address their advantages and disadvantages.

2.2 Overview of Cluster Analysis Methods

There have been many cluster analysis algorithms proposed in the literature. A number of these algorithms are particularly suitable for a certain type of data (e.g. numeric or nominal). Several algorithms are also suitable for a particular purpose or the application domain [32], [39]. We briefly present several cluster analysis methods as discussed in [16] and [32]. We place particular emphasis on the first two methods, partitional and hierarchical, as they are strongly related to this study. For a detailed review of cluster analysis methods we suggest [35], [7], [16], [32].

2.2.1 Partitional Methods

The partitional (also referred to as *partitioning*) cluster analysis methods divide the data into k groups, where each group contains at least one object and where an object may reside in at most one group. Once an initial partitioning is created these algorithms use an *iterative relocation technique* to move the objects to different clusters for better partitioning result [32]. In addition, a distance function is used to measure the quality of partitioning result so that the goal of the cluster analysis is satisfied. Recall from Chapter 1 that the main goal of any cluster analysis task is to minimize the intra-cluster distance and maximize the inter-cluster distance. Thus, for partitional algorithms, this acts as an objective function that needs to be satisfied. The *K-means* algorithm is the best-known algorithm from this category [32]. We discuss the algorithm in detail below.

K-means Algorithm
K-means is an iterative algorithm where a cluster is represented by the *centroids* (the mean value of the objects in a cluster). Given a dataset and the number of clusters k, the algorithm works as follows - the first step of this algorithm is to initialize the centroids. There are a number of different ways to assign the initial values to the centroids. We may either randomly select any k objects from the data, or select the first k objects and assign them as the centroids of the clusters. Once the algorithm is initialized with the centroids, the next step is to calculate the distance from each centroid to all the objects in the dataset. A distance measure, such as the Euclidean distance, is often used to calculate this distance. Next, the objects are assigned to the respective clusters based on the minimum distance from the centroids. Therefore, an object will be assigned to a cluster if the distance between its centroid and the object is minimum (compared to the distances between the centroids of other clusters and this object). Once all the objects are assigned to their respective clusters, we recalculate the centroids with the new cluster assignments. The centroid, as mentioned above, is the mean value of all the objects in a cluster. We then iterate the process a number of times until the stopping criterion is satisfied. This is usually satisfied when the objects are no longer reallocated to different

clusters or when the maximum number of iterations is reached. The pseudocode for the algorithm is given in Table 2.1. Figure 2.3 depicts the results of the K-means algorithm when applied to a sample dataset with $k = 3$. The clusters are marked with circles for clear visualization. Example 2.2.1 illustrates the K-means algorithm when applied to a sample dataset.

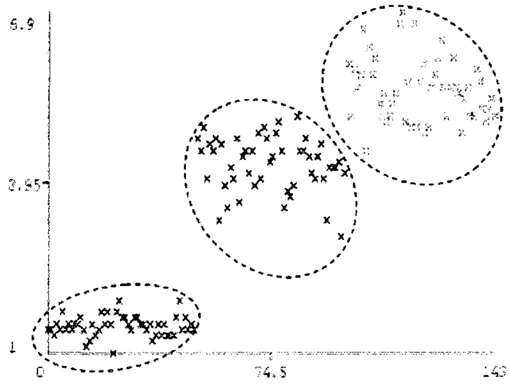


Figure 2.3: Example of the K-means algorithm applied to a sample dataset. Each cluster is marked with a dashed circle.

Example 2.2.1. In this example, the dataset contains 9 items:
$D = \{2, 4, 10, 12, 3, 20, 30, 11, 25\}$. Let $k = 2$, the desired number of clusters. We use the Euclidean distance as the distance measure. The first step of the algorithm provided in Table 2.1 consists in assigning any two items as the cluster centroids. These items are either selected randomly or the first k items are selected. We used the later approach for this example. Below we show the calculations for each phase.

Iteration 1: centroid1 = 2 and centroid2 = 4

The distance between centroid1 and each item in D:
$\{0, 2, 8, 10, 1, 18, 28, 9, 23\}$

The distance between centroid2 and each item in D:
$\{2, 0, 6, 8, 1, 16, 26, 7, 21\}$

According to the minimum distance between the centroids and each of the items, the clusters are:

$Cluster1 = \{2, 3\}$ Since the item 3 is equally close to centroid1 and centroid2, we arbitrarily selected cluster1.

$Cluster2 = \{4, 10, 12, 20, 30, 11, 25\}$

Iteration 2: centroid1 $= \frac{2+3}{2} = 2.5$ and centroid2 $= \frac{4+10+12+20+30+11+25}{7} = 16$

The distance between centroid1 and each item in D:

$\{0.5, 1.5, 7.5, 9.5, 0.5, 17.5, 27.5, 8.5, 22.5\}$

The distance between centroid2 and each item in D:

$\{14, 12, 6, 4, 13, 4, 14, 5, 9\}$

The clusters are:

$Cluster1 = \{2, 3, 4\}$ and $Cluster2 = \{10, 12, 20, 30, 11, 25\}$

Iteration 3: centroid1 $= \frac{2+3+4}{3} = 3$ and centroid2 $= \frac{10+12+20+30+11+25}{6} = 18$

The distance between centroid1 and each item in D:

$\{1, 1, 7, 9, 0, 17, 27, 8, 22\}$

The distance between centroid2 and each item in D:

$\{16, 14, 8, 6, 15, 2, 12, 7, 7\}$

The clusters are:

$Cluster1 = \{2, 3, 4, 10\}$ and $Cluster2 = \{12, 20, 30, 11, 25\}$

Iteration 4: centroid1 $= \frac{2+3+4+10}{4} = 4.75$ and centroid2 $= \frac{12+20+30+11+25}{5} = 19.6$

The distance between centroid1 and each item in D:

$\{2.75, 0.75, 5.25, 7.25, 1.75, 15.25, 25.25, 6.75, 20.25\}$

The distance between centroid2 and each item in D:

$\{17.6, 15.6, 9.6, 7.6, 16.6, 0.4, 11.4, 8.6, 5.4\}$

The clusters are:

$Cluster1 = \{2, 3, 4, 10, 11, 12\}$ and $Cluster2 = \{20, 30, 25\}$

Iteration 5: centroid1 $= \frac{2+3+4+10+11+12}{6} = 7$ and centroid2 $= \frac{20+30+25}{3} = 25$

The distance between centroid1 and each item in D:

$\{5, 3, 3, 5, 4, 13, 23, 6, 18\}$

The distance between centroid2 and each item in D:

$\{23, 21, 15, 13, 22, 5, 5, 14, 0\}$

The clusters are:

$Cluster1 = \{2, 3, 4, 10, 11, 12\}$ and $Cluster2 = \{20, 30, 25\}$

Since none of the items were relocated in iteration 5 (iteration 4 and 5 are identical), the algorithm terminates at this point. The result for this example, which is returned at the end of the process is: *Cluster1* $= \{2, 3, 4, 10, 11, 12\}$ and *Cluster2* $= \{20, 30, 25\}$.

Advantages of the K-means Algorithm

According to Han et al. [32], the K-means algorithm works well for compact clusters in which the clusters are well separated from one another. Moreover, the algorithm also works well for large datasets, since the computational complexity of the algorithm is *O(n)*, where n is the number of objects present in the dataset [35], [32].

Limitations of the K-means Algorithm

One of the disadvantages of the K-means algorithm is that it only considers numeric attribute types and is therefore not applicable to datasets with nominal or categorical attributes. However, the *K-modes* algorithm, a variation of the K-means algorithm, has been proposed to deal with this shortcoming [32]. Unlike the K-means algorithm, which uses the *means* to compute the cluster centroids, the K-modes algorithm as the name suggests, uses the *modes* of the cluster centroids. There is another issue with the K-means algorithm with regard to the number of clusters. We have previously discussed the consequences of incorrectly selecting the number of clusters. The performance of the K-means algorithm depends in part on the initial values selected as the cluster centroids in the initialization stage that may later affect the quality of the clusters. Dunham [32] also states that, the K-means algorithm is very sensitive to outliers. Moreover, the clusters produced by this algorithm are usually convex shaped and as such, the K-means algorithm usually fails to discover clusters of arbitrary shapes (Figure 2.5 (b)) [2].

Other Partitional-based Algorithms

The *K-medoids* algorithm [32] is another partitional-based algorithm in which a cluster is represented by one of the real objects, located close to the center point. The center point, commonly known as the *medoids*, leads its name to the algorithm. As opposed to the K-means algorithm, the K-medoids algorithms are less sensitive to the outliers [7]. The most well-known K-medoids algorithms are *PAM* (Partitioning Around Medoids) [39], *CLARA* (Clustering LARge Applications) [39], and *CLARANS* (Clustering Large Applications based upon RANdomized Search) [51]. The PAM algorithm works best when the dataset is small, whereas CLARA and CLARANS both deal with larger datasets. The PAM algorithm performs a search over the entire dataset to find k medoids, whereas CLARA performs the search over a fixed sample of the dataset [32]. Therefore, CLARA may not provide a good solution, as the results considering the samples may not nec-

The K-means Algorithm

Input:

Data objects and k (number of clusters)

Output:

K (the set of clusters)

Algorithm:

```
1. Initialize the algorithm with $k$ objects which will the act as
   the centroids for the first run.
2. For each object calculate the distance (i.e. Euclidean Dis-
   tance) from centroid and assign the object to the cluster with
   closest centroid.
3. Recalculate the centroid once all the objects are assigned into
   one of the $k$ clusters.
4. Repeat step 2 and 3 till the stopping criterion is satisfied.
   The stopping criteria may be: when the maximum number
   of iteration is reached or when no objects are assigned to
   different clusters.
```

Table 2.1: Pseudocode for the K-means algorithm.

essarily represent a good solution for the entire dataset. The CLARANS algorithm, on the other hand, is an improvement to the CLARA algorithm. The CLARANS algorithm selects the sample with some randomness in which every object is regarded as a potential candidate. One of the drawbacks of the CLARANS algorithm is the computational complexity, which is $O(n^2)$ (where n is the number of objects in a given dataset). According to Han et al. [32], practical experiments show that the CLARANS algorithm performs better than both the PAM and CLARA algorithms.

2.2.2 Hierarchical Methods

In this section, we discuss another type of cluster analysis method known as the *Hierarchical Clustering* methods. A hierarchical method builds a hierarchy or a tree of clusters.

The tree is also commonly referred to as a *dendrogram* [16]. The root of a tree often contains all the data objects in one cluster, whereas the leaves of the tree usually contain each object in a single cluster. There are two variations of this method discussed in the literature: *agglomerative or bottom-up approach* and *divisive or top-down approach* [32], [35]. In the first approach, the algorithm starts from the bottom of the tree where each object has its own unique cluster. It gradually groups these clusters by recursively merging two or more similar clusters together. This process is continued until all the clusters are merged into a single cluster (the root) or a given termination criterion is satisfied. The divisive hierarchical approach, on the other hand, initially holds all the objects in one single cluster. It then gradually splits them into smaller clusters until all the objects belong to their own individual clusters or a termination criterion is satisfied. The merging and splitting is performed depending on the distance associated with each level of the hierarchical tree, which is given in the form of a distance matrix as an input to the algorithm [16]. Two clusters are merged when the distance is low and a cluster is split into smaller clusters when the distance is large (when the elements are not close enough).

Example 2.2.2. Figure 2.4 depicts a simple example of a hierarchical clustering algorithm, performed on a dataset consisting of six objects [16]. In this example, when the agglomerative approach is used, $\{B\}$ and $\{D\}$ are grouped together to form the cluster $\{B, D\}$, whereas $\{E\}$ and $\{F\}$ are merged together to form the cluster $\{E, F\}$. Next $\{A\}$ and $\{B, D\}$ are merged to create the cluster $\{A, B, D\}$ which is then merged with $\{C\}$ to create the cluster $\{A, B, D, C\}$. In the very last step $\{A, B, D, C\}$ and $\{E, F\}$ are grouped together and the root cluster $\{A, B, C, D, E, F\}$ is created. The divisive approach, as discussed earlier works in a similar way, but in the reversed order. It starts from the root $\{A, B, C, D, E, F\}$ and in each step splits the cluster into two smaller sets until all the elements have formed their own individual cluster.

Advantages of the Hierarchical Clustering Methods

Hierarchical methods are suitable for datasets that possess natural nesting relationships between the clusters. Examples of such datasets include datasets from biology and animal taxonomies [16]. Moreover, since the distance or similarity is presented through a matrix to these algorithms, the algorithms are able to handle different attribute types [7].

Limitations of the Hierarchical Clustering Methods

One of the weaknesses of the hierarchical methods is that, once a cluster is formed, the

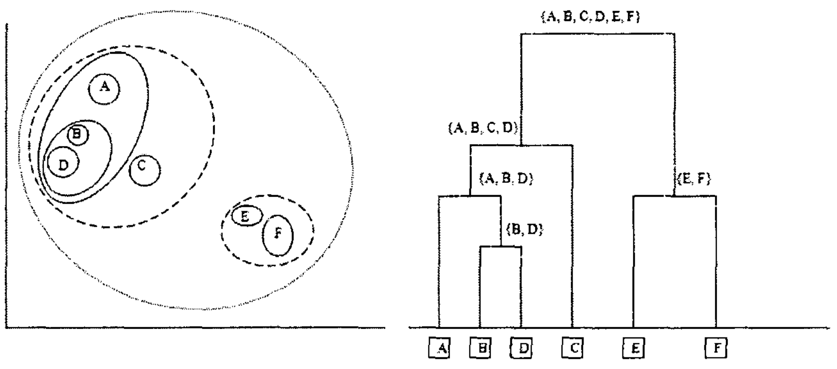


Figure 2.4: An example of the hierarchical clustering algorithm applied to a sample data: (Left) representation of the result as a nested subset, (Right) representation of the result as a dendrogram.

objects in the clusters may not be relocated to improve the results. As such, unlike the K-means algorithm where objects are iteratively relocated to improve the result, the hierarchical algorithms lack such possibility. The algorithms are also sensitive to outliers [80]. Dunham [16] also noted that, due to the time and space complexity (which is $O(n^2)$ for a dataset with n objects) of these algorithms, they may not be suitable for large datasets.

Variations of Hierarchical Algorithms

A variety of hierarchical cluster analysis algorithms are available in literature. For instance, *AGNES* (AGglomerative NESting) is an agglomerative clustering approach and *DIANA* (DIvisive ANAlysis) is an example of divisive hierarchical approach. The *ROCK* (RObust Clustering using linKs) algorithm [29], another example of agglomerative hierarchical algorithm, is suitable for datasets with categorical attributes. The *BIRCH* (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm [83] handles both large datasets and outliers well and also has the ability to perform incremental or dynamic cluster analysis by accommodating dataset updates [32]. The *CURE* (Clustering Using REpresentatives) algorithm [28] integrates hierarchical and partitional methods together in one algorithm. The algorithm is suitable for large datasets, handles outliers and has the ability to discover arbitrary-shaped clusters. *CHAMELEON* [38] is another example of the agglomerative hierarchical algorithm which is also suitable for large datasets and has the ability to discover arbitrary shaped clusters. An excellent and detailed review of these algorithms is presented in [32].

Apart from the hierarchical and partitional methods, there are a number of other methods that are frequently mentioned in the literature. We briefly discuss these methods in the subsequent sections to keep the readers informed about the wide variety of possible approaches from the family of cluster analysis methods.

2.2.3 Density-based Methods

Unlike the hierarchical and partitional cluster analysis algorithms, which consider the distance or similarity between the objects to find the clusters, *density-based* methods are based on the notion of *density*. According to Dunham [16], the term *density* is defined as the minimum number of objects located within a certain distance of one another. Thus, the clusters are represented by the dense areas of the data objects and are usually separated by the areas with low density. In this approach, the clusters may take any arbitrary shape and grow in any direction, as long as the density in the neighboring area exceeds a certain threshold [32]. Examples of algorithms from this family are: *DBSCAN* (Density-Based Spatial Clustering Algorithm with Noise) [18] and *DENCLUE* (DENsity-based CLUstEring) [34]. As the name implies, the DBSCAN algorithm is suitable for spatial datasets with noise. The algorithm also discovers clusters of arbitrary shape [32]. However, this algorithm is very sensitive to the choice of user-defined parameters (e.g. the radius of the neighborhood) [32]. The DENCLUE algorithm is suitable for high dimensional datasets. Similar to the DBSCAN algorithm, this algorithm also discovers arbitrary shaped clusters and handles datasets with large amount of noise [32].

2.2.4 Grid-based Methods

In the *Grid-based cluster analysis* [32] methods, the entire data space is first divided into a finite number of cells that form a grid structure. The cluster analysis is then performed on this grid data, instead of the original data points. Since the number of cells in the grid data is usually much less than the number of original data points, the computation and processing time of this algorithm are relatively faster than many other cluster analysis algorithms. The algorithms from this family are mostly suitable for spatial datasets. *STING* (STatistical INformation Grid) [68], *WaveCluster* [57], and *CLIQUE* [1] are an example of algorithms based on this method. The STING algorithm manipulates the statistical information (e.g. count, maximum, minimum, and standard deviation) of the grid cells to process the queries. The algorithm is query-independent as the statistical information regarding the attributes are pre-computed and stored in each cell. STING is also very efficient. Moreover, when a given dataset is updated, this algorithm is able to perform incremental updates without re-computing all the statistical information [32]. However, the user-specific parameters (e.g. the number of grids and number of layers) need to be provided by the users and therefore the selection of parameters may have impact on the end result. The WaveCluster algorithm, in contrast, applies a signal processing technique called *wavelet transform*, to find the clusters. More information regarding *wavelet transform* and WaveCluster are presented in [57], [32]. The algorithm is not sensitive to outliers, discovers clusters of arbitrary shapes, and performs well for large datasets. However, one of the drawbacks of this algorithm is that it may only be applied to low-dimensional datasets. On the other hand, the CLIQUE algorithm, which integrates density-based and grid-based algorithms together, is suitable for large, highly dimensional datasets.

2.2.5 Model-based Methods

Model-based approaches assume that all the data is generated by a mixture of underlying statistical distributions. For example, the *EM* (Expectation-Maximization) algorithm is a popular model-based approach that performs expectation-maximization analysis based on statistical modeling [32]. The *COBWEB* and *SOM* (Self-Organized Map) algorithms also fall into this category, where the former is a conceptual learning algorithm and the later is a neural network-based algorithm. A detailed discussion of these algorithms is presented in [32].

2.2.6 Clustering High Dimensional Data

Highly dimensional datasets consist of several hundreds or even thousands of attributes. For instance, objects in a text dataset are usually regarded as a collection of documents and each document consists of hundreds or even thousands of words and terms. Thus, the attributes for this type of datasets are the collection of these words and terms gathered from the documents. In such cases, the previously discussed clustering algorithms may not work well as the data become very sparse with the increase of the number of dimensions. As a result, when the similarity between the data points is calculated, the result is usually a very small value which may not contribute to the computation. Moreover, as Han and Kamber [32] noted, the average density of these points is also likely to be very low. Therefore, new or modified algorithms that handle the problem of high dimensionality are necessary. Two such methods for clustering high-dimensional datasets are *Subspace Clustering* and *Frequent Pattern-based Clustering.* The subspace clustering algorithms such as *CLIQUE* and *PROCLUS*, tend to find the clusters from a subset of dimensions of the original set of attributes. On the other hand, the frequent pattern-based clustering algorithms search for frequently occurring patterns from the dataset and use these patterns to find the clusters [32]. With a growing number of domains containing high dimensional data, performing cluster analysis on highly dimensional datasets has become challenging. Therefore, special care is needed to successfully perform cluster analysis on this type of datasets.

2.2.7 Constraint-based Clustering

The *Constraint-based* methods consist of cluster analysis algorithms that heavily rely on user guidance. Users provide various constraints and information to the algorithms so that the clusters may be generated based on the preferences given by the users. Yin et al. [81] proposed one such user-guided clustering algorithm called *CrossClus*. The algorithm is suitable for multi-relational datasets. The algorithm starts with selecting a set of relevant features from multiple relations to construct a single object type, based on the user interest and domain specific knowledge. Next, the K-medoids based algorithm, *CLARANS* is applied to the selected features to find the clusters.

In the next section, we introduce the *Spectral Clustering* algorithms, a graph-based clustering algorithm.

2.3 Introduction to Spectral Clustering

In the previous section, we discussed cluster analysis algorithms based on the concepts of distance, density, grids, and statistical models. Recently, another family of cluster analysis method has gained much interest in the research community. The algorithms from this group originated from the area of *graph partitioning* and are collectively known as the *Spectral clustering* algorithms [58], [46], [50], [48], [37]. The spectral methods manipulate the eigenvector(s) and eigenvalue(s) of a similarity matrix to find the clusters.

There are several advantages to applying the spectral clustering algorithms as stated in [46]. One of the advantages is that the spectral clustering algorithms do not make any assumptions on the shape of the clusters. For instance, algorithms such as K-means, usually form convex shaped clusters. Aiello et al. in their work in [2] provided an example illustrating this difference as given in Figure 2.5 when applied on a sample dataset. Figure 2.5(a) shows the ring clusters obtained from the spectral clustering algorithm and Figure

2.5(b) depicts the results from the K-means algorithm when applied on the same sample dataset. Fischer et al. [22] presented more illustrations and examples of various forms of clusters that are achievable by the spectral clustering algorithms. The algorithms

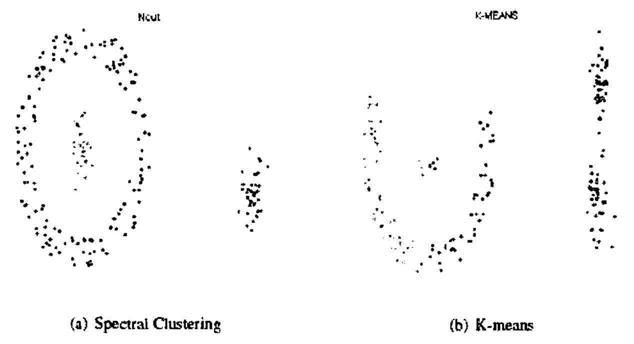


Figure 2.5: Comparison of results from spectral clustering and the K-means algorithm. (a) Results from the spectral clustering algorithm, (b) Results from the K-means algorithm.

based on the spectral method also do not suffer from local minima [46]. Therefore, it may not be necessary to restart the algorithms with various initializations. The spectral clustering algorithms also work very well for large datasets. According to Luxburg [46], *"spectral clustering can be implemented efficiently even for large data sets, as long as we make sure that the similarity graph is sparse"*. Moreover, the algorithms may be solved efficiently with standard linear algebra methods [46]. The algorithms are also more stable than some algorithms in terms of initializing the user-specific parameters (i.e. the number of clusters). The user-specific parameters may often be estimated accurately with the help of theories related to the algorithms. As mentioned previously in Chapter 1, the spectral clustering algorithms often outperform traditional algorithms such as K-means and Single Linkage [46]. Prior studies also show that the algorithms have had significant practical success in the areas including, image segmentation [58], [67], speech separation [5], biological sequence datasets [52], social network analysis [42], [70], and high-dimensional spaces [8], amongst others. Finally, the algorithms from this family are able to handle different types of data (i.e. numeric, nominal, binary, or mixed), since one needs to convert the dataset into a similarity matrix to be able to apply this algorithm on a given dataset.

Even though these algorithms are new to the world of cluster analysis, the notion behind it was already a well-established research topic in the literature. As mentioned, these algorithms are closely related to the area called *Graph Partitioning Theory* [58] [50]. A graph consists of a set of nodes or vertices and the set of edges that connects the nodes. Therefore, partitioning the graph denotes dividing the graph into several disjoint subsets. This began in the early 1970s when Fielder [20] discovered that the graph bipartitioning problem is correlated with the eigenvectors of a matrix called the *Laplacian matrix.* The Laplacian matrix is a matrix representation of a given graph which will be discussed in detail later in the chapter. Given a $n \times n$ matrix A, then λ is an eigenvalue of A if there exists a non-zero vector x such that, $Ax = \lambda x$. Here, x is called an eigenvector of A corresponding to eigenvalue λ [9]. Fielder noticed that the eigenvector associated with the second smallest eigenvalue of the Laplacian matrix contains useful information about the connectivity of a given graph. As such, this vector is often referred to as the *Fielder Vector* in literature [46], [20]. The techniques associated with graph partitioning and the exploitation of eigenvalues and eigenvectors of the Laplacian matrix are equally applicable to the cluster analysis problem [48]. Therefore, the spectral cluster analysis algorithms also use the eigenvalues and eigenvectors of the Laplacian matrix (which is constructed from the similarity matrix) to find the partitions. In graph theory, the eigenvalues and eigenvectors of the Laplacian matrix or the adjacency matrix for a given graph, are known as the *graph spectrum* or the *spectrum of the graph.* As such, the algorithms in this group are called the *Spectral Clustering* algorithms.

The algorithms in this family require many theories and properties from graph theory and linear algebra. In fact, an area called *Spectral Graph Theory* [11] is fully dedicated to the study of graph connectivity and graph spectrum. As a result, even though the algorithm seems less complex and efficient to solve, it requires good understanding of various graph and eigenvector related topics (i.e. graph spectrum, graph partitioning, graph cut, random walk, and linear algebra). In the subsequent sections, we discuss several notations from graph theory that are later applied to explain the spectral clustering algorithms. A brief summary of several linear algebra terms and notations related to our discussion are presented in Appendix A.

2.3.1 Graph Notations

Let G be a weighted, undirected graph denoted as $G = (V, E)$. Here, V is the set of vertices such that $V = v_1, v_2 .... v_n$, assuming that the graph G has $n = |V|$ nodes or vertices. Let E be the set of edges connecting the vertices in graph G. It is not necessary that every two vertices in the graph should be connected through an edge. Since G is a weighted graph, we may also assume that if two vertices are connected through an edge then there will be a positive value assigned to that edge. In graph theory, this is known as the *edge weight.* Therefore, if v_i and v_j have an edge connecting them, then the weight is denoted as $w_{i,j}$ and $w_{i,j} \geq 0$. Let W be a $n \times n$ matrix, known as the *adjacency matrix* of the graph G. Each entry of W contains the edge weight of any two vertices connected to one another. Since G is an undirected graph, the adjacency matrix W is a symmetric matrix, where for all pair of vertices v_i and v_j, $w_{i,j} = w_{j,i}$. If the vertices v_i and v_j are not connected through an edge then $w_{i,j} = w_{j,i} = 0$.

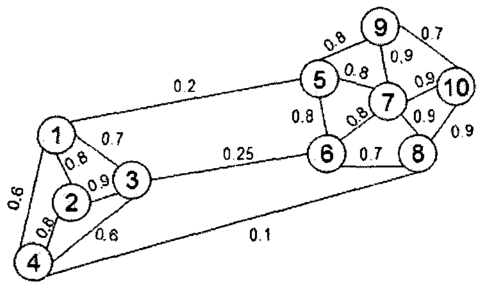


Figure 2.6: An example of a simple, undirected and weighted graph.

The *degree* of a vertex v_i is the sum of the total edge weight from vertex v_i to all other vertices that are connected to v_i. Thus, for vertex v_i, it is same as the sum of the elements of row i in the adjacency matrix W. Mathematically, this is defined as Equation 2.1.

$$d_i = \sum_{j=1}^{n} w_{i,j} \tag{2.1}$$

Let D be another $n \times n$ diagonal matrix called the *degree matrix* with elements of d on its diagonal. The matrix D is also symmetric.

The *volume* of a subset of vertices is denoted as Equation 2.2, where A is a subset of V. *Volume* is often used to measure the size of a subset of vertices V.

$$Vol(A) = \sum_{i \in A} d_i \tag{2.2}$$

Example 2.3.1. Figure 2.6 depicts a simple graph with 10 vertices.
$V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and
$E = \{$ {1,2},{1,3},{1,4},{1,5},{2,1},{2,3},{2,4},{3,1},{3,2},{3,4},{3,6},{4,1},{4,2},{4,3}, {4,8},{5,1},{5,6},{5,7},{5,9},{6,3},{6,5},{6,7},{6,8},{7,5},{7,6},{7,8},{7,9},{7,10},{8,4},

{8,6},{8,7},{8,10},{9,5},{9,7},{9,10},{10,7},{10,8},{10,9}}.
We arbitrarily assigned edge weights between several vertices based on the visual distance between the vertices, for explanation purpose. The edge weights are high when the vertices are relatively close to one another, whereas the weights are small when the vertices are located relatively far from one another. Then, according to the above notations, the matrix W and D will have entries as given in Figure 2.7 and Figure 2.8, respectively. For instance, the edge weight between vertex v_1 and v_2 is $w_{1,2} = 0.8 = w_{2,1}$, vertex v_3 and v_6 is $w_{3,6} = 0.25 = w_{6,3}$, and vertex v_1 and v_6 is $w_{1,6} = 0 = w_{6,1}$ (since there is no edge present in between vertex v_1 and v_6).

$$W = \begin{pmatrix} 0 & 0.8 & 0.7 & 0.6 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0.8 & 0 & 0.9 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.7 & 0.9 & 0 & 0.6 & 0 & 0.25 & 0 & 0 & 0 & 0 \\ 0.6 & 0.8 & 0.6 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0.2 & 0 & 0 & 0 & 0 & 0.8 & 0.8 & 0 & 0.8 & 0 \\ 0 & 0 & 0.25 & 0 & 0.8 & 0 & 0.8 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0.8 & 0 & 0.9 & 0.9 & 0.9 \\ 0 & 0 & 0 & 0.1 & 0 & 0.7 & 0.9 & 0 & 0 & 0.9 \\ 0 & 0 & 0 & 0 & 0.8 & 0 & 0.9 & 0 & 0 & 0.7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.9 & 0.9 & 0.7 & 0 \end{pmatrix}$$

Figure 2.7: The *weight matrix* W for the graph given in Figure 2.6.

The degree matrix D is given as the sum of each row of matrix W on its main diagonal. For instance, the degree of vertex v_1 is: $0.8 + 0.7 + 0.6 + 0.2 = 2.3$. The matrix D for the graph G is given in Figure 2.8.

To calculate the volume of a subset A, assume that A contains vertices $\{1, 2, 3, 4\}$. Then according to the definition of volume, the volume of subset A is calculated as:

$$Vol(A) = 2.3 + 2.5 + 2.45 + 2.1 = 9.35$$

In the next subsection we will define a term, called *Graph Cuts* which is one of the key terms in graph partitioning and is used as the objective function for spectral clustering algorithms.

2.3.2 Graph Cuts

The *Graph Cuts*, or simply the *Cuts*, partition a graph into two sets. In graph theory, graph cuts tend to find partitions such that the total edge weights in between the parti-

$$D = \begin{pmatrix} 2.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.45 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.55 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 \end{pmatrix}$$

Figure 2.8: The *degree matrix* D for the graph given in Figure 2.6.

tions have a very small value and the total edge weights within the each partition have a high value [46]. For graph G, a cut will partition the set of vertices V into sets A and B such that $A \cup B = V$ and $A \cap B = \phi$ [58]. This is achieved by removing the set of edges that connect the two partitions A and B. This set of edges between the partition A and B are called the *edge cut.* In graph theory, a cut is denoted as Equation 2.3.

$$cut(A, B) = \sum_{i \in A, j \in B} w_{i,j} \tag{2.3}$$

An example of a graph cut is depicted in Figure 2.9. The dashed line in the figure shows where the cut is performed so that the *cut* value between the two partition is minimum. However, there may be a number of different possible solutions to bi-partition a graph.

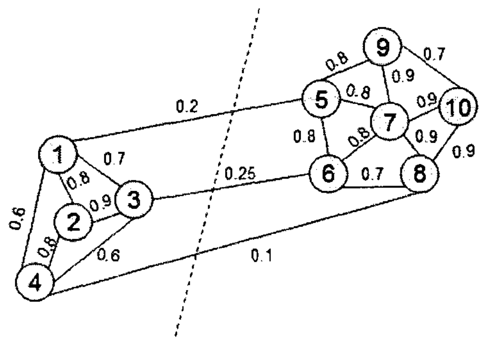


Figure 2.9: An example of graph cuts.

For this reason, in graph theory the optimum solution to this bi-partitioning problem is the one that minimizes the cut value among all the possible pairs of partitions. This cut method is often referred to as the *minimum cut* problem. However, as Wu and Leahy [77] noticed, a common problem with the minimum cut method is that it tends to cut small sets of isolated objects in the graph. Notice that the cut value increases with the number of edges crossing the partitions. Therefore, when the two sets are balanced they will have a higher cut value. On the other hand, if one of the sets contains isolated points then the cut value in between this set and the rest will be the lowest, as there will be fewer edges crossing the partitions. To overcome this problem, Shi and Malik [58] proposed a new cut method called the *Normalized Cut* or *NCut* method. The formula for *NCut* is denoted in Equation 2.4.

$$NCut(A,B) = \frac{Cut(A,B)}{Vol(A)} + \frac{Cut(A,B)}{Vol(B)} \tag{2.4}$$

In this case also, one may achieve the desired partitions by minimizing the NCut value. Unlike the minimum cut method, the normalized cut method will usually have a high value when the partitions are imbalanced (when one of the partitions contains small sets of isolated nodes). Moreover, minimizing the NCut value means finding a cut that will have a relatively small weight in between the partitions while maintaining high internal edge weights within each partition [58]. Therefore, it satisfies the goal of graph cuts. Another version of the graph cut method is known as the *ratio-cut* method [58] [46]. The ratio-cut is similar to the normalized cut method. However, as opposed to the normalized cut method that takes the volume of the subsets to measure the cut weight, ratio-cut considers the size of the partitions as denoted by the total number of vertices present in the subsets.

Example 2.3.2. This example uses the graph given in Figure 2.6. We consider two cases: Case 1 computes the normalized cut for the cut position depicted in Figure 2.9 and Case 2 computes the NCut when the partitions are not balanced.

Case 1: In this case we will assume that the partitions are: $A = \{1,2,3,4\}$ and $B = \{5,6,7,8,9\}$
$Vol(A) = 2.3 + 2.5 + 2.45 + 2.1 = 9.35$
$Vol(B) = 2.6 + 2.55 + 4.3 + 2.6 + 2.4 + 2.5 = 16.95$
$Cut(A,B) = 0.2 + 0.25 + 0.1 = 0.55$
$NCut(A,B) = \frac{0.55}{9.35} + \frac{0.55}{16.95} = 0.6204$

Case 2: In this case we will assume that the partitions are: $A = \{4\}$ and $B = \{1,2,3,5,6,7,8,9\}$

$$Vol(A) = 2.1$$
$$Vol(B) = 2.5 + 2.45 + 2.1 + 2.6 + 2.55 + 4.3 + 2.6 + 2.4 + 2.5 = 24.0$$
$$Cut(A, B) = 0.6 + 0.8 + 0.6 + 0.1 = 2.1$$
$$NCut(A, B) = \frac{2.1}{2.1} + \frac{2.1}{24} = 1.0875$$

The NCut value for Case 1 where the partition is balanced is 0.6204. The NCut value for an unbalanced partition is 1.0875. Since Case 1 gives the minimum NCut value, this partition will be the solution to the bi-partitioning problem for the graph depicted in Figure 2.9.

The spectral methods manipulate the eigenvalues and eigenvectors of the graph Laplacian to find the partition. In the next subsection, we define the graph Laplacians and discuss their properties.

2.3.3 Graph Laplacian

Let W be the similarity matrix constructed from a given dataset. We build the degree matrix D from W by using Equation 2.1. Then the Laplacian matrix is defined as Equation 2.5.

$$L = D - W \tag{2.5}$$

The Laplacian matrix L for the graph depicted in Figure 2.6 is given in Figure 2.10. Notice that the main diagonal of the Laplacian matrix is always non-negative and the row sum for each row i is 0. This is true for all the Laplacian matrices and from this a number of important properties arise as discussed shortly.

Recall that the spectral clustering algorithm is usually performed on this Laplacian matrix and not directly on the similarity matrix. The eigenvectors and eigenvalues of the Laplacian matrix possess many important properties [11], [20] that ultimately help to find the partitions. The use of eigenvectors of the Laplacian matrix for partitioning the graph is not new to the area of graph partitioning. In early 1970's Fielder [20], Donath and Hoffman [15], and a number of other researchers discovered that the eigenvectors of the adjacency matrix or the Laplacian matrix, possess many interesting properties which lead to the solution of the problem of partitioning a given graph. According to Donath and Hoffman, it was the eigenvectors of the adjacency matrix that partitions the graph. In contrast, according to Fielder, the eigenvector associated with the second smallest eigenvalue of the Laplacian matrix gives the partitions. Since then, there have been numerous publications on this topic where various authors have proposed different

$$L = \begin{pmatrix} 2.3 & -0.8 & -0.7 & -0.6 & -0.2 & 0 & 0 & 0 & 0 & 0 \\ -0.8 & 2.5 & -0.9 & -0.8 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.7 & -0.9 & 2.45 & -0.6 & 0 & -0.25 & 0 & 0 & 0 & 0 \\ -0.6 & -0.8 & -0.6 & 2.1 & 0 & 0 & 0 & -0.1 & 0 & 0 \\ -0.2 & 0 & 0 & 0 & 2.6 & -0.8 & -0.8 & 0 & -0.8 & 0 \\ 0 & 0 & -0.25 & 0 & -0.8 & 2.55 & -0.8 & -0.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.8 & -0.8 & 4.3 & -0.9 & -0.9 & -0.9 \\ 0 & 0 & 0 & -0.1 & 0 & -0.7 & -0.9 & 2.6 & 0 & -0.9 \\ 0 & 0 & 0 & 0 & -0.8 & 0 & -0.9 & 0 & 2.4 & -0.7 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.9 & -0.9 & -0.7 & 2.5 \end{pmatrix}$$

Figure 2.10: The *Laplacian matrix* L for the graph given in Figure 2.6.

combinations of these matrices (Laplacian/adjacency) and eigenvectors to bi-partition the graph. An excellent review on graph Laplacian and their spectral theories is given by Chung in [11].

In Chapter 1, we mentioned that there are several different variations of spectral clustering algorithms available in the literature. These algorithms mostly differ on the choice of graph Laplacian and the associated eigenvalues and eigenvectors. However, Luxburg [46] also noted that there are a number of matrices present in the literature that are also referred to as the graph Laplacians. These matrices are commonly known as the *Normalized Laplacian* matrices. The two variants of the normalized Laplacian are given in Equation 2.6 and Equation 2.7 [46].

$$L = D^{\frac{-1}{2}}(D - W)D^{\frac{-1}{2}} \tag{2.6}$$

$$L = D^{\frac{-1}{2}}WD^{\frac{-1}{2}} \tag{2.7}$$

The matrix in Equation 2.6 is used by Shi and Malik in [58], where they discovered that the second smallest eigenvector is the solution to the partitioning problem. This method is similar to the one given by Fielder [20]. One of the differences with Fielder's method and Shi and Malik's work, is that Fielder used the original Laplacian matrix, whereas Shi and Malik used the normalized Laplacian matrix to find the eigenvector. Ng, Jordan and Weiss [50], used the matrix in Equation 2.7 to define their version of spectral clustering algorithm and used the k largest eigenvectors to find the clusters. The eigenvectors of the graph Laplacians contain important information about the cluster assignments. They are the characteristic vector of a matrix and the components of this vector indicate the cluster assignments. Thus, the points that are similar will usually have similar eigenvector components and by thresholding the eigenvector(s) we discover the clusters. There are several properties of the Laplacian matrix that are often addressed to explain the spectral methods theoretically and are presented below.

Properties of the Laplacian matrix [11], [46]:

Property 2.3.1. *Laplacian matrices are symmetric real matrices. This follows directly from the definition of D and W.*

Property 2.3.2. *The matrix is positive semi-definite. A real valued matrix M is positive semi-definite if, for all $f \in \Re^n, f'Mf \geq 0$. The mathematical proof of this property is given in [46].*

Property 2.3.3. *L has n non-negative, real valued eigenvalues such that $0 = \lambda_0 \leq \lambda_1 \leq \lambda_2 ... \leq \lambda_n$. This follows from the definition of the Laplacian matrix (a real symmetric matrix will have exactly n real eigenvalues) and the properties of the positive semi-definite functions (the eigenvalues of the positive semi-definite functions are always non-negative).*

Property 2.3.4. *The smallest eigenvalue is 0 and the eigenvector corresponding to the smallest eigenvalue is a constant vector containing all 1's. If eigenvalue $\lambda = 0$ then we need to find an eigenvector v such that $(D - W)v = 0v = 0$. Recall, that the row sum for each row i of matrix $D - W$ is 0 which follows directly from the definition of D and W. Thus $(D - W)v = 0$ only when v is a vector with all 1's.*

Property 2.3.5. *The eigenvectors and eigenvalues provide important information about the graph connectivity [20]. The multiplicity k of the eigenvalue 0 gives the number of connected components in the graph [46].*

The spectral clustering algorithms are often divided into two types: 1) *recursive algorithms* and 2) *multi-way algorithms* [66]. The algorithms in the first group, as the name suggest, recursively bi-partition the data at each step until a stopping criterion is satisfied. The most widely used algorithm from this group is, the *Normalized Cut Spectral Clustering (SM(NCut))* by Shi and Malik [58], [66], [46]. In contrast, multi-way spectral clustering algorithms directly partition the data into k groups. The best-known algorithms from this group are: 1) the Ng, Jordan and Weiss algorithm [50] and 2) the

Meila - Shi algorithm [48]. In this study, to evaluate the performance of the proximity measures on the spectral clustering algorithms, we consider two algorithms, one from each group. From the first group, we select the normalized cut spectral clustering algorithm as this algorithm proved to have had several practical successes in a variety of fields. We refer to this algorithm as *SM(NCut)* in the rest of the thesis. The Ng, Jordan and Weiss algorithm is an improvement to the algorithm proposed by Meila and Shi [50]. The algorithm proposed by Meila and Shi normalizes the rows of the weight matrix and uses its eigenvectors to find the clusters. In contrast, the algorithm proposed by Ng, Jordan and Weiss uses the eigenvectors of the Laplacian matrix and normalizes the rows of the eigenvectors to unit length. The authors suggest that the former method might not produce good clustering solutions in situations when the between cluster *degree* varies substantially across the clusters [50]. Thus, we select Ng, Jordan and Weiss algorithm from the second group for this study. We will refer to this algorithm as the *NJW(K-means)* from now on.

In the next two subsections we present these two algorithms in detail.

2.3.4 Algorithm 1: SM(NCut)

The SM(NCut) spectral clustering algorithm is one of the most widely used recursive spectral clustering algorithm, which manipulates the eigenvectors to find the clusters. The algorithm was proposed by Shi and Malik [58] for image segmentation tasks. The main intuition behind this algorithm is the optimization of an objective function called the Normalized Cut, or NCut, as discussed in Section 2.3.2. The authors introduced this new graph cut method and showed that, by minimizing this objective function, one achieve good partitions of the data. They also provided necessary mathematical proofs to show that the NCut is optimized by solving a generalized eigenvalue problem, and then use the eigenvector of this generalized eigenvalue system to find the underlying clusters. The algorithm is discussed in detail below.

The normalized cut is defined in Equation 2.4 (the definition of the terms (e.g. *Vol(A)*, *Cut(A,B)*) are given in Section 2.3.1 and Section 2.3.2). Minimizing the NCut is the same as finding a cut such that the total connection in between two groups is weak, whereas the total connection within each group is strong. The authors found that minimizing the NCut in this way is a NP-complete problem. A proof for this claim is given in [58]. Thus, they proposed a different strategy that minimizes the NCut but uses a generalized eigenvalue problem to solve this. The mathematical proof for how this NCut problem is transformed into a generalized eigenvalue problem is also given in [58]. We continue with the algorithm, assuming that the NCut problem is transformed into a generalized eigenvalue problem by using the proof given by Shi and Malik. At this point, we will introduce several terms, to help us clarify the algorithm proposed in [58].

Let A and B be the two partitions and $\mathbf{x}$ be an n dimensional (where n is the number of objects) indicator vector with $x_i = 1$ if object $i \in A$ and $x_i = -1$ when object $i \in B$. (The terms W, D and d_i are defined in Section 2.3.1). Then minimizing the NCut problem is equivalent to minimizing the following expression with two constraints:

$$min_{\mathbf{x}} NCut(\mathbf{x}) = min_{\mathbf{y}} \frac{\mathbf{y}^T(D-W)\mathbf{y}}{\mathbf{y}^T D \mathbf{y}} \tag{2.8}$$

The constraints are $\mathbf{y}_i \in \{1, -b\}$ and $\mathbf{y}^T D\mathbf{1} = 0$. In the above expression, $\mathbf{y}$ has a similar meaning as the indicator vector $\mathbf{x}$. However, it represents the information in a slightly different way as $\mathbf{y} = (\mathbf{1}+\mathbf{x}) - b(\mathbf{1}-\mathbf{x})$. Here, $b = \frac{\sum_{x_i>0} d_i}{\sum x_i{<}0d_i}$. Recall, $x_i > 0 \approx x_i = 1$ (according to the definition of $\mathbf{x}$) which represents the objects in group A and $\sum_{x_i>0} d_i \approx (1+\mathbf{x})D\mathbf{1}$ denotes the total connection from nodes in A to all other nodes. Similarly, $x_i < 0 \approx x_i = -1$ which represents the objects in group B and $\sum_{x_i<0} d_i \approx (1-\mathbf{x})D\mathbf{1}$ denotes the total connection from B to all other nodes. Thus, b is the ratio of the two groups A and B as measured by their degree d.

The constraints, as discussed above, are the direct consequences of the condition that the vector $\mathbf{x}$ may only take two discrete values, as defined by the indicator vector $\mathbf{x}$. Below we show that the second constraint also holds.

Proof. $\mathbf{y}^T D\mathbf{1}$
$= [(1+\mathbf{x}) - b(1-\mathbf{x})]D\mathbf{1}$
$= (1+\mathbf{x})D\mathbf{1} - b(1-\mathbf{x})D\mathbf{1} = \sum_{x_i>0} d_i - b\sum_{x_i<0} d_i = 0$ □

However, the authors proposed that if $\mathbf{y}$ is relaxed to take real, continuous values, then Equation 2.8 may be minimized by solving a generalized eigenvalue system as given in Equation 2.9.

$$(D-W)\mathbf{y} = \lambda D\mathbf{y} \tag{2.9}$$

Moreover, they also suggested to transform this generalized eigenvalue system into a standard eigenvalue system. By doing so, the authors showed that the constraints on $\mathbf{y}$ are satisfied by this standard eigenvalue problem, which in turn, satisfy the constraints of generalized eigenvalue system automatically. The generalized eigenvalue system given in Equation 2.9 is transformed into a standard eigenvalue system as follows [58]:

Proof. $(D - W)\mathbf{y} = \lambda D\mathbf{y}$
$=> (D - W)\mathbf{y} = \lambda D^{\frac{1}{2}} D^{\frac{1}{2}}\mathbf{y}$
$=> \frac{(D-W)}{D^{\frac{1}{2}}}\mathbf{y} = \lambda D^{\frac{1}{2}}\mathbf{y}$
$=> D^{\frac{-1}{2}}(D - W)\mathbf{y} = \lambda D^{\frac{1}{2}}\mathbf{y}$
$=> D^{\frac{-1}{2}}(D - W)D^{\frac{-1}{2}}D^{\frac{1}{2}}\mathbf{y} = \lambda D^{\frac{1}{2}}\mathbf{y}$ □

Let $z = D^{\frac{1}{2}}\mathbf{y}$, then the above equation is equivalent to:

$$D^{\frac{-1}{2}}(D - W)D^{\frac{-1}{2}}z = \lambda z \tag{2.10}$$

Equation 2.10 is the standard eigenvalue system of the generalized system 2.9. Here z is the eigenvector of matrix $D^{\frac{-1}{2}}(D - W)D^{\frac{-1}{2}}$ and λ is the eigenvalue.

Properties of the matrix $D^{\frac{-1}{2}}(D - W)D^{\frac{-1}{2}}$ (from Equation 2.10):

Property 2.3.6. $z_0 = D^{\frac{1}{2}}\mathbf{1}$ *(where **1** is a vector of all ones) is an eigenvector of this matrix with eigenvalue* 0. *We may verify this as follows.*

Proof. $D^{\frac{-1}{2}}(D - W)D^{\frac{-1}{2}}z_0$
$= D^{\frac{-1}{2}}(D - W)D^{\frac{-1}{2}}D^{\frac{1}{2}}\mathbf{1} = D^{\frac{-1}{2}}(D - W)\mathbf{1} = 0$
Recall from previous discussion that the term $(D-W)$ is known as the Laplacian matrix. According to the **Property 2.3.4** of the Laplacian matrix $(D - W)\mathbf{1} = 0$. Thus, the term $D^{\frac{-1}{2}}(D - W)\mathbf{1} = 0$. □

Property 2.3.7. *The expression $D^{\frac{-1}{2}}(D - W)D^{\frac{-1}{2}}$ is symmetric positive semi-definite. This again follows from the properties of the Laplacian matrix (**Property 2.3.1** and **Property 2.3.2**), symmetric matrix and positive semi-definiteness.*

Property 2.3.8. *The eigenvalues are all real and non-negative. z_0 is the smallest eigenvector of this matrix all the eigenvectors are perpendicular to one another. This follows directly from the properties of positive semi-definiteness.*

Having summarized the properties of the eigenvalue problem given in Equation 2.10, we now use these properties to find the solution to the original generalized eigensystem given in Equation 2.9. We will also show how the constraints placed on $\mathbf{y}$ are satisfied through these properties. Notice that the smallest eigenvalue of the generalized eigenvalue system is 0 and the eigenvector associated with this eigenvalue is $\mathbf{1}$. The proof is similar to the one for the standard eigenvalue system in Equation 2.10. Next, $0 = z_1^T z_0 = y_1^T D\mathbf{1}$ (y_1 is the second smallest eigenvector). Recall from **Property 2.3.8**

of the matrix given in Equation 2.10 (Page 31) that all the eigenvectors of this matrix are perpendicular to one another. Therefore, if z_0 and z_1 are two eigenvectors of that matrix and if they are perpendicular to one another then their inner product is 0 (according to the properties of perpendicular vectors) and thus $z_1^T z_0 = 0$. Also recall that $z = D^{\frac{1}{2}}y$. Then, $z_1 = D^{\frac{1}{2}}y_1$ and $z_0 = D^{\frac{1}{2}}\mathbf{1}$. Replacing them back to $z_1^T z_0$, we get
$z_1^T z_0 = D^{\frac{1}{2}}y_1^T D^{\frac{1}{2}}\mathbf{1} = y_1^T D\mathbf{1} = 0$.

Therefore, the constraints are satisfied by the eigenvectors of the generalized eigenvalue system. However, we still need to find which eigenvector actually minimizes the normalized cut problem. The authors noted that the expression $min_{\mathbf{y}}\frac{\mathbf{y}^T(D-W)\mathbf{y}}{\mathbf{y}^T D\mathbf{y}}$ is known as the *Rayleigh quotient* [27] and found that according to the properties of this quotient, the second smallest eigenvector of the generalized eigenvalue system is the real valued solution to their Normalized Cut problem. The partitions are found by thresholding this eigenvector. There are a number of ways this grouping may be performed. One may use a particular point (i.e. zero, mean, median) as the splitting criteria or may use any existing algorithms such as the K-means for this purpose. Components with similar values usually reside in the same cluster. Since, this algorithm bi-partition the data, we get two disjoint clusters. To find more clusters we need to re-partition the segments by recursively applying the algorithm on each of the partitions. The summary of the SM(NCut) algorithm is given in Table 2.2.

Example 2.3.3. We use the same graph as illustrated in Figure 2.6 to show how this algorithm works. We assume that the nodes in this graph are the objects from a datasets and the edge weights represent the similarity in between the objects. Table 2.2 contains the steps of the normalized cut spectral clustering algorithm. According to step 1 and step 2, we need to construct the matrix W, D and L, as we have done for calculations in previous sections. We continue from step 3 by solving the eigensystem. The first few eigenvalues and their corresponding eigenvectors are given in Figure 2.11 and Figure 2.12. We used MATLAB® to calculate the eigenvectors and eigenvalues.

Notice that the eigenvalues are sorted in increasing order (Figure 2.11). The smallest eigenvalue is 0 and the eigenvector associated with this eigenvalue is a constant vector of all ones. In this example it is the first vector with all 0.1950's and is similar to the vector $0.1950 * evec_1$ where $evec_1$ is a vector of all ones. The second smallest eigenvalue is 0.0842 as marked with a circle in the Figure 2.11 and the corresponding eigenvector is the second column from left and is marked with a square box in Figure 2.12. According to the algorithm, this eigenvector contains the information about the cluster assignments. Figure 2.14 plots the components of the eigenvectors associated with the smallest and the

Normalized Cut Spectral Clustering Algorithm (SM(NCut))

Input:

The Dataset of N objects

Algorithm:

1. Form the $N \times N$ symmetric weight matrix W and degree matrix D.
2. Construct the Laplacian matrix $D - W$.
3. Solve the eigensystem $(D - W)x = \lambda Dx$ for eigenvectors with smallest eigenvalues.
4. Find the second smallest eigenvalue and use the eigenvector associated with this eigenvalue to bipartition the dataset.
5. Recursively repartition the partitions if necessary.

Table 2.2: The normalized cut spectral clustering algorithm (SM(NCut)).

```
-0.0000         0         0         0         0         0
      0    0.0842         0         0         0         0
      0         0    0.7646         0         0         0
      0         0         0    0.8426         0         0
      0         0         0         0    1.2544         0
      0         0         0         0         0    1.2736
```

Figure 2.11: The eigenvalues of the Laplacian matrix of the graph in Figure 2.6. The second smallest eigenvalue is marked with a circle.

second smallest eigenvalues, respectively. To find the clusters we now map the original objects to the corresponding components of this eigenvector. The objects are mapped according to their order in matrix W. We then split this eigenvector at 0 to find the clusters. This is depicted in Figure 2.13. This algorithm found two clusters where *Cluster1* contains objects $\{1, 2, 3, 4\}$ and *Cluster2* contains objects $\{5, 6, 7, 8, 9, 10\}$. In the original graph, also, the objects in the first cluster are located very close to one another and so do the objects in *Cluster2*.

0.1950	0.2526	-0.0379	0.0428	0.0007	0.5401
0.1950	0.2788	0.0272	0.0403	-0.0211	-0.1013
0.1950	0.2492	-0.0016	-0.0455	-0.4197	-0.1929
0.1950	0.2651	0.0597	0.0282	0.4881	-0.2365
0.1950	-0.1225	-0.3850	-0.0091	0.0865	0.0902
0.1950	-0.1114	-0.1386	-0.4030	-0.0516	-0.0628
0.1950	-0.1552	0.0059	0.0077	0.0212	-0.0505
0.1950	-0.1432	0.3149	-0.2133	0.0746	0.0861
0.1950	-0.1612	-0.1523	0.3799	-0.0413	-0.1020
0.1950	-0.1666	0.3094	0.2056	-0.0901	0.0575

Figure 2.12: The eigenvectors of the Laplacian matrix of the graph in Figure 2.6. The eigenvector associated with the second smallest eigenvalue is marked with a square box.

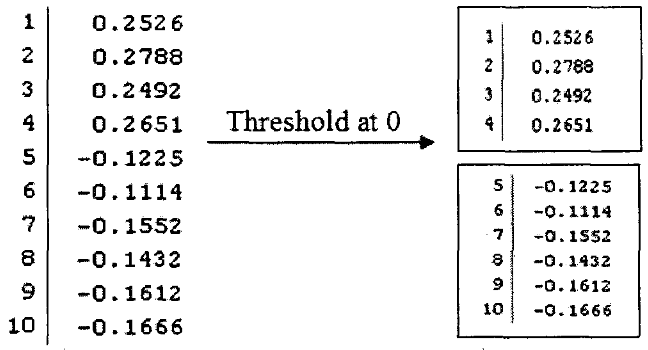


Figure 2.13: Spectral bi-partitioning of graph in Figure 2.6. On the left we first map the original objects to their corresponding components in the eigenvector. Next we partition them at position 0 to find the clusters.

2.3.5 Algorithm 2: NJW(K-means)

This section explains the algorithm proposed by Ng, Jordan and Weiss [50]. Unlike the *SM(NCut)* algorithm that minimizes the NCut objective function and recursively bi-partitions the data, this algorithm directly partitions the data into k groups. The algorithm manipulates the normalized Laplacian matrix given in Equation 2.7 to find the clusters. As mentioned in Section 2.3.3, the algorithm is an improvement to the prior work performed by Shi and Meila in [48]. Shi and Meila proposed an algorithm called the *MNCut* algorithm that relates the theories from *Markov Random Walk* to the eigenvectors and the eigenvalues of the normalized Laplacian matrix. We present the algorithm from the perspective of the *Random Walk Theory* [56] and show their

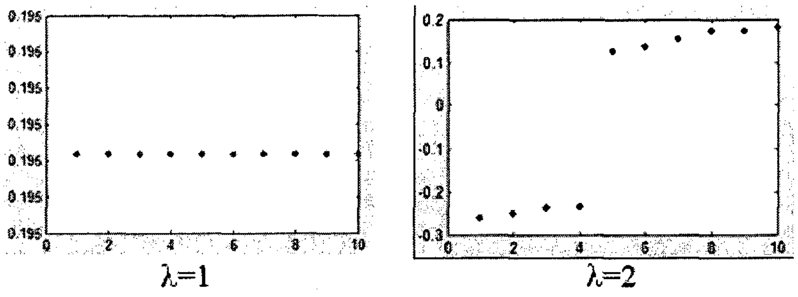


Figure 2.14: The eigenvalues and eigenvectors. (Left) The components of eigenvector associated with the smallest eigenvalue 0. (Right) The components of the eigenvector associated with the second smallest eigenvalue.

relationship with the normalized Laplacian matrix [48], [52]. Then together with the theories of random walk as well as the properties of the normalized Laplacian matrix we find the perturbed matrix which serve as the solution to the partitioning problem. This matrix is then solved using the theories from *Matrix Perturbation Theory* [50], [46].

Assume that the graph G represents a random walk problem. The nodes are the states and the edges are considered as the path that connects two states. Also, assume that we have placed some particles on the nodes and they have the liberty to move around and jump from one node to another. At any particular time the particles are more likely to jump to a node where the edge weight is high (when nodes are very similar to one another). The particles are also likely to stay within the nodes in which the similarity values of the neighboring nodes are high and are unlikely to jump to a node where the edge weight is low (when nodes are very dissimilar). The main idea behind the spectral partitioning from the random walk point of view is to separate the clusters such that the group of nodes where the particles stay for a longer period are separated from the group of nodes where the particles are unlikely to travel. We relate both the random walk and the spectral theory together to explain the algorithm [48], [52].

Let the probability of jumping from node i to node j be P_{ij}. As mentioned above, the probability P_{ij} depends on the edge weights from node i to node j and thus depends on the similarity between the nodes denoted as $w_{i,j}$ previously.

$$P_{ij} = \frac{\textit{the edge weight from node i to node j}}{\textit{total edge weights from node i to all other nodes}} \tag{2.11}$$

$Pij = \frac{w_{i,j}}{\sum_l w_{i,l}} = \frac{w_{i,j}}{d_i}$ [from the definitions of w and d]
Then, according to the definitions of W and D, the *Markov Transition Matrix* P is

defined as:

$$P = \frac{W}{D} = D^{-1}W \tag{2.12}$$

All the entries of this matrix are positive. The row sum for each row i is 1.

Properties of the Markov matrix [62]:

Property 2.3.9. *The largest eigenvalue of this matrix is* $\lambda_1 = 1$.

Property 2.3.10. *All other eigenvalues for* $i = [2...n]$ *satisfy* $\lambda_i \leq 1$.

Next, according to the properties of random walk [56], if we start at node i_0 then,
after 1 step we will be at i_1 and the probability $= P_{i_0 i_1}$
after 2 steps we will be at i_2 and the probability $= \sum_{i_1} P_{i_0 i_1} P_{i_1 i_2} = (P^2)_{i_0 i_2}$
after 3 steps we will be at i_3 and the probability $= \sum_{i_2} P_{i_0 i_1} P_{i_1 i_2} P_{i_2 i_3} = (P^3)_{i_0 i_3}$
and after n steps we will be at i_n and the probability $= (P^n)_{i_0 i_n}$
Therefore, P^n is the probability distribution of a particle after n steps. According to [52], for an undirected, non-negative and connected graph, any particle that starts at a particular state will reach the same stationary distribution after an infinite number of steps. Stationary distribution is defined as $\pi^{\infty} = \frac{d}{\sum_i d_i}$ and $P^T \pi^{\infty} = \pi^{\infty}$. Thus, as time evolves and progresses toward infinity, the probability of being at a particular state becomes more independent of the initial state and as it reaches the stationary distribution the information about the initial state will be completely lost. The stationary distribution in this case will give us little information about the areas in which the particle has already traveled, which in turn will not be able to give us information about the areas in the graph where the nodes are closely connected. However, the authors also noted that during the *Markov Relax* process a particle will spend longer time at this closely connected area before reaching the stationary distribution. Thus, we are looking for those nodes in which the particle spends the most time before jumping to a different location. By analyzing the eigenvalues and eigenvectors of the matrix given in Equation 2.12, one may find the nodes where the particle spends the most time and in turn find the partitions where the similarity in between the objects are high.

Recall that the original algorithm as given in [50] uses the normalized Laplacian matrix $L = D^{\frac{-1}{2}} W D^{\frac{-1}{2}}$ to find the partitions, whereas, we started our discussion with the Markov Transition Matrix $P = D^{-1}W$. The matrices are, however, similar [52]. This is proved as below:

Proof. $D^{-1}Wv = \lambda v$
$=> D^{\frac{1}{2}}D^{-1}Wv = \lambda D^{\frac{1}{2}}v$
$=> D^{\frac{-1}{2}}Wv = \lambda D^{\frac{1}{2}}v$
$=> D^{\frac{-1}{2}}WD^{\frac{-1}{2}}x = \lambda D^{\frac{1}{2}}D^{\frac{-1}{2}}x$ [by letting, $v = D^{\frac{-1}{2}}x$]
$=> D^{\frac{-1}{2}}WD^{\frac{-1}{2}}x = \lambda x$ $\square$

Notice that, both the matrices $D^{-1}W$ and $D^{\frac{-1}{2}}WD^{\frac{-1}{2}}$ have the same eigenvalues and the eigenvectors corresponding to these eigenvalues are v and $x = D^{\frac{1}{2}}v$, respectively. Therefore, the matrices are similar. Moreover, the matrix $D^{\frac{-1}{2}}WD^{\frac{-1}{2}}$ is symmetric and has several interesting properties that will help us to find our solution. As such, we will use this matrix from now on rather than the Markov matrix.

One of the interesting properties of a real symmetric matrix is that it is often decomposed and re-written as the sum of its eigenvalue and eigenvector pairs [62]. Let A be a real symmetric matrix and the eigenvalues of this matrix are λ_i and the eigenvectors are e_i. Then according to this property, A is re-written as the equation given in Equation 2.13.

$$A = \sum_{i=1}^{k} \lambda_i e_i e_i^T \tag{2.13}$$

As the matrix $D^{\frac{-1}{2}}WD^{\frac{-1}{2}}$ is also symmetric, let the eigenvalues and eigenvectors of this matrix be λ_i and z_i respectively with $\lambda_1 \geq \lambda_2 \geq .... \geq \lambda_t$. We may re-write this as Equation 2.14 or Equation 2.15:

$$D^{\frac{-1}{2}}WD^{\frac{-1}{2}} = \sum_{i=1}^{k} \lambda_i z_i z_i^T \tag{2.14}$$

$$D^{\frac{-1}{2}}WD^{\frac{-1}{2}} = \lambda_1 z_1 z_1^T + \lambda_2 z_2 z_2^T + ... + \lambda_t z_t z_t^T \tag{2.15}$$

Moreover, the eigenvectors of the symmetric matrices are also orthogonal to one another. Thus, $z_i^T z_j = 0$ for $i \neq j$.

As such, we have showed that the Markov transition matrix $D^{-1}W$ is similar to the symmetric matrix $D^{\frac{-1}{2}}WD^{\frac{-1}{2}}$. Recall, we also showed that P^n is the probability distribution of a particle after n finite steps. Since $P = D^{-1}W$, this is re-written as Equation 2.16.

$$P = D^{\frac{-1}{2}}WD^{\frac{-1}{2}} \tag{2.16}$$

Then,

$$P^n = (D^{\frac{-1}{2}}WD^{\frac{-1}{2}})^n \tag{2.17}$$

Since $DP = W$ (from Equation 2.12),

$$\begin{aligned}(D^{\frac{-1}{2}}WD^{\frac{-1}{2}})^n &= (D^{\frac{-1}{2}}WD^{\frac{-1}{2}})......(D^{\frac{-1}{2}}WD^{\frac{-1}{2}})\\ &= (D^{\frac{-1}{2}}DPD^{\frac{-1}{2}})......(D^{\frac{-1}{2}}DPD^{\frac{-1}{2}})\\ &= (D^{\frac{1}{2}}PD^{\frac{-1}{2}})......(D^{\frac{1}{2}}PD^{\frac{-1}{2}})\\ &= (D^{\frac{1}{2}}P^nD^{\frac{-1}{2}})\end{aligned}$$

$(D^{\frac{-1}{2}}WD^{\frac{-1}{2}})^n = (D^{\frac{1}{2}}P^nD^{\frac{-1}{2}})$
$=> D^{\frac{-1}{2}}(D^{\frac{-1}{2}}WD^{\frac{-1}{2}})^n = D^{\frac{-1}{2}}(D^{\frac{1}{2}}P^nD^{\frac{-1}{2}})$
$=> D^{\frac{-1}{2}}(D^{\frac{-1}{2}}WD^{\frac{-1}{2}})^n = P^nD^{\frac{-1}{2}}$
Thus,

$$P^n = D^{\frac{-1}{2}}(D^{\frac{-1}{2}}WD^{\frac{-1}{2}})^nD^{\frac{1}{2}} \tag{2.18}$$

Recall $D^{\frac{-1}{2}}WD^{\frac{-1}{2}} = \lambda_1 z_1 z_1^T + \lambda_2 z_2 z_2^T + ... + \lambda_t z_t z_t^T$ and by replacing it in the Equation 2.18 we get,

$$\begin{aligned}P^n &= D^{\frac{-1}{2}}(D^{\frac{-1}{2}}WD^{\frac{-1}{2}})^nD^{\frac{1}{2}}\\ &= D^{\frac{-1}{2}}(\lambda_1 z_1 z_1^T + \lambda_2 z_2 z_2^T + ... + \lambda_t z_t z_t^T)^nD^{\frac{1}{2}}\\ &= D^{\frac{-1}{2}}(\lambda_1^n z_1 z_1^T + \lambda_2^n z_2 z_2^T + ... + \lambda_t^n z_t z_t^T)D^{\frac{1}{2}}\end{aligned}$$

As such,

$$P^n = \sum_{i=1}^{t} D^{\frac{-1}{2}}\lambda_i^n z_i z_i^T D^{\frac{1}{2}} \tag{2.19}$$

As mentioned earlier, P^n gives the dynamics of the particle after n finite steps. Thus, P_n is also the dynamics before we reach the stationary distribution. During this time the particle will stay for a longer period within the cluster before jumping to any other location [52]. When $n = \infty$, P^∞ becomes the distribution after an infinite steps and is thus called the stationary distribution.

$$P^\infty = \sum_{i=1}^{t} D^{\frac{-1}{2}}\lambda_i^\infty z_i z_i^T D^{\frac{1}{2}} \tag{2.20}$$

Recall from the properties of Markov matrix (**Property 2.3.9**), that the largest eigenvalue λ_1 is always 1. We already showed that (Page(36)) this matrix and the normalized Laplacian matrix are similar and they have the same eigenvalues. Also, from the **Property 2.3.10** of the Markov matrix all the other eigenvalues are less or equal to one. We have $\lambda_1^\infty z_1 z_1^T = z_1 z_1^T$ and $\lambda_2^\infty z_2 z_2^T + ... + \lambda_t^\infty z_t z_t^T \approx 0$ and $\lambda_i^\infty \approx 0$. Incorporating all the information in Equation 2.20 we get:

$$\begin{aligned} P^\infty &= \sum_{i=1}^{t} D^{\frac{-1}{2}} \lambda_i^\infty z_i z_i^T D^{\frac{1}{2}} \\ &= D^{\frac{-1}{2}} (\lambda_1^\infty z_1 z_1^T + \lambda_2^\infty z_2 z_2^T + ... + \lambda_t^\infty z_t z_t^T) D^{\frac{1}{2}} \\ &= D^{\frac{-1}{2}} z_1 z_1^T D^{\frac{1}{2}} \end{aligned}$$

From Equation 2.19 we have,

$$\begin{aligned} P^n &= \sum_{i=1}^{t} D^{\frac{-1}{2}} \lambda_i^n z_i z_i^T D^{\frac{1}{2}} \\ &= (D^{\frac{-1}{2}} \lambda_1^n z_1 z_1^T D^{\frac{1}{2}}) + \sum_{i=2}^{t} D^{\frac{-1}{2}} \lambda_i^n z_i z_i^T D^{\frac{1}{2}} \\ &= P^\infty + \sum_{i=2}^{t} D^{\frac{-1}{2}} \lambda_i^n z_i z_i^T D^{\frac{1}{2}} \end{aligned}$$

Therefore we have:

$$P^\infty = D^{\frac{-1}{2}} z_1 z_1^T D^{\frac{1}{2}} \tag{2.21}$$

and

$$P^n = P^\infty + \sum_{i=2}^{t} D^{\frac{-1}{2}} \lambda_i^n z_i z_i^T D^{\frac{1}{2}} \tag{2.22}$$

In this case, the first term P^∞ in Equation 2.22, is the distribution where the particle will end up after an infinite number of steps. From the definition of this term, we know that this manipulates the largest eigenvector of the normalized Laplacian matrix. On the other hand, the second term tells us about the probability distribution of the particle after n finite steps. This term manipulates the leading eigenvectors associated with the eigenvalues $\lambda \in [2...n]$ of the normalized Laplacian matrix. This term is called the perturbation to the stationary distribution [52]. As of now, we showed why the algorithm uses the normalized Laplacian matrix as given in Equation 2.7. We explained the main idea behind using this matrix, relating the problem to the Markov Random Walk problem. However, the matrix that ultimately gives the partition is a perturbed matrix and need to be solved by applying the properties of matrix perturbation theory.

According to [46], [50], perturbation theory studies how the eigenvalues and eigenvectors of a matrix $\hat{A}$ change when a little perturbation H is added, resulting a perturb matrix A such that $A = \hat{A} + H$. The perturbation theory is often applied to solve problems which may not be solved exactly, by first solving a related problem for which the exact solution is known. As given above, the equation $P^n = P^\infty + \sum_{i=2}^{t} D^{\frac{-1}{2}} \lambda_i^n z_i z_i^T D^{\frac{1}{2}}$ also falls into this category if we take, $A = P^n$, $\hat{A} = P^\infty$ and $H = \sum_{i=2}^{t} D^{\frac{-1}{2}} \lambda_i^n z_i z_i^T D^{\frac{1}{2}}$. According to [50], $\hat{A}$, in this case, is the *ideal* solution to the problem in which all the objects in different clusters are located infinitely far apart and between cluster similarity is 0. Proof for this claim is given in [50].

We will use an example to illustrate how the clusters may be found by using this method. Figure 2.15 shows a similarity matrix and the first three eigenvectors of its Laplacian matrix. Objects $\{1, 2, 3\}$ belong to cluster 1, similarly, objects $\{4, 5, 6\}$ and $\{7, 8, 9\}$ belong to cluster 2 and cluster 3 respectively. We assume the similarity between the objects in a cluster are all the same (in this case 1). The entries with 0 in the similarity matrix represents the between cluster similarity. According to Luxburg [46],

1	1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1	1

Similarity Matrix

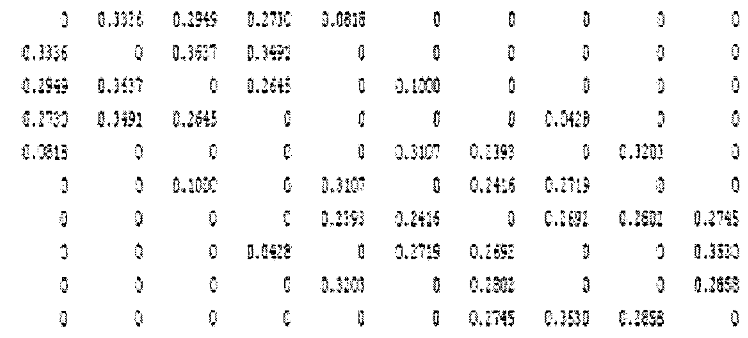
First 3 eigenvectors $\longrightarrow$

0.5774	0	0
0.5774	0	0
0.5774	0	0
0	0.5774	0
0	0.5774	0
0	0.5774	0
0	0	0.5774
0	0	0.5774
0	0	0.5774
y1	y2	y3

Figure 2.15: The *ideal case* of K-means spectral clustering algorithm.

the object $v_i \in \Re^k$ will have the form $(0, 0, 0..1, 0, 0...)$ where 1 represents the cluster where the object belongs to and k is the number of clusters. In our example, when $k = 3$ the object $v_1 = (1, 0, 0)$ (since all the non zero entries for the eigenvectors in the figure have the same value 0.5774, we may use 1 instead of 0.5774). Thus, all the objects that belong to the same cluster will have the same value for v. For example, $v_1 = v_2 = v_3 = (1, 0, 0)$ as they all reside in cluster 1. This also holds for cluster 2 and cluster 3. The K-means algorithm is applied to this set of objects v_i to find the correct partitions [50], [46]. This will be the case for a perfectly *ideal* situation. In a nearly ideal case when the between cluster similarity is not necessarily 0, the normalized Laplacians are considered as the perturbed version of the ideal case [46] [50]. Thus, by solving $\sum_{i=2}^{t} D^{\frac{-1}{2}} \lambda_i^n z_i z_i^T D^{\frac{1}{2}}$, as discussed above one may find the clusters. This is due to the fact that the perturbation theory suggests that the eigenvectors of this matrix should be very close to the ideal case, if not the same, and the K-means or any other clustering algorithm should be able to find the clusters. A more detailed explanation is presented in [46]. The algorithm is given in Table 2.3.

Example 2.3.4. In this section, we provide an example (similar to the one given for the *SM(NCut)* algorithm) to show how the algorithm works. In this case also, the weight matrix (W), and the degree matrix (D) will be same as the one calculated previously (step 1-3 in the algorithm). However, matrix L in step 4 will be different than the one we calculated earlier. According to step 4 of this algorithm, L is defined as: $L = D^{\frac{-1}{2}} W D^{\frac{-1}{2}}$. This will give us the matrix L as depicted in Figure 2.16. To keep this example as close as



0	0.3336	0.2949	0.2730	0.0815	0	0	0	0	0
0.3336	0	0.3637	0.3491	0	0	0	0	0	0
0.2949	0.3637	0	0.2645	0	0.1030	0	0	0	0
0.2730	0.3491	0.2645	0	0	0	0	0.0428	0	0
0.0815	0	0	0	0	0.3107	0.2393	0	0.3203	0
0	0	0.1030	0	0.3107	0	0.2416	0.2719	0	0
0	0	0	0	0.2393	0.2416	0	0.2692	0.2802	0.2745
0	0	0	0.0428	0	0.2719	0.2692	0	0	0.3530
0	0	0	0	0.3203	0	0.2802	0	0	0.2858
0	0	0	0	0	0	0.2745	0.3530	0.2858	0

Figure 2.16: The matrix $L = D^{\frac{-1}{2}} W D^{\frac{-1}{2}}$.

the one we provided for the Normalized cut algorithm, we assume that $k = 2$. Then, the two largest eigenvalues and their associated eigenvectors are (step 5) given in Figure 2.17 (the upper part contains the eigenvalues and the lower part contains the eigenvectors). The matrix X, as given in step 6, is same as the lower part of the Figure 2.17. Next in step 7, we construct the matrix Y by renormalizing the rows of X. Figure 2.19 plots the components of matrix X and matrix Y.

Finally, in step 8, we run the K-means algorithm with $k = 2$ on the matrix Y. The cluster assignments are given in Figure 2.20, where the column on the left contains the object ids and the column on the right shows the cluster assignments.

The first cluster contains objects $\{1, 2, 3, 4\}$ and the second cluster contains objects $\{5, 6, 7, 8, 9\}$.

Algorithm: Spectral Clustering with K-means (NJW(K-means))

Input:

The Dataset of N objects

k the number of clusters

Algorithm:

1. Form the $N \times N$ symmetric weight matrix W from the dataset of size N. Here $W(i,j) = w_{ij}$, the similarity between object i and j.

2. Let d be a $N \times 1$ matrix with $d = \sum_j w_{ij}$ (the total similarity value from object i to all other object or the sum of the elements of W's ith row).

3. Let D be another $N \times N$ diagonal matrix with d on its diagonal.

4. Construct the $N \times N$ matrix $L = D^{\frac{-1}{2}} W D^{\frac{-1}{2}}$.

5. Find the k (same as the number of clusters) eigenvectors associated with k largest eigenvalues, let these eigenvectors be $x_1, x_2, ..., x_k$

6. Form the $N \times k$ matrix $X = [x_1 x_2 .... x_k]$ by stacking the eigenvectors in columns.

7. Form another $N \times k$ matrix Y from X by normalizing each row of X so that it has a unit length. This may be done by calculating $Y_{ij} = \frac{X_{ij}}{(\sum_j X_{ij}^2)^{\frac{1}{2}}}$ for each row of X.

8. Use K-means on Y to find k clusters, by treating each row of Y as a point in k-dimensions.

9. Assign the original points back to the clusters formed from Y such that if a point i from Y belongs to j cluster then the original point i from the input dataset will also belong to cluster j.

Table 2.3: The spectral clustering algorithm with K-means (NJW(K-means)).

```
1.0000         0
     0    0.9158
------------------
0.2957   -0.3831
0.3083   -0.4409
0.3052   -0.3900
0.2826   -0.3842
0.3144    0.1975
0.3114    0.1778
0.4043    0.3217
0.3144    0.2309
0.3021    0.2497
0.3083    0.2634
```

Figure 2.17: The two largest eigenvalues and their eigenvectors of matrix L.

```
1.0000   -0.7916
1.0000   -0.8195
1.0000   -0.7875
1.0000   -0.8055
1.0000    0.5319
1.0000    0.4960
1.0000    0.6226
1.0000    0.5920
1.0000    0.6371
1.0000    0.6495
```

Figure 2.18: The matrix Y.

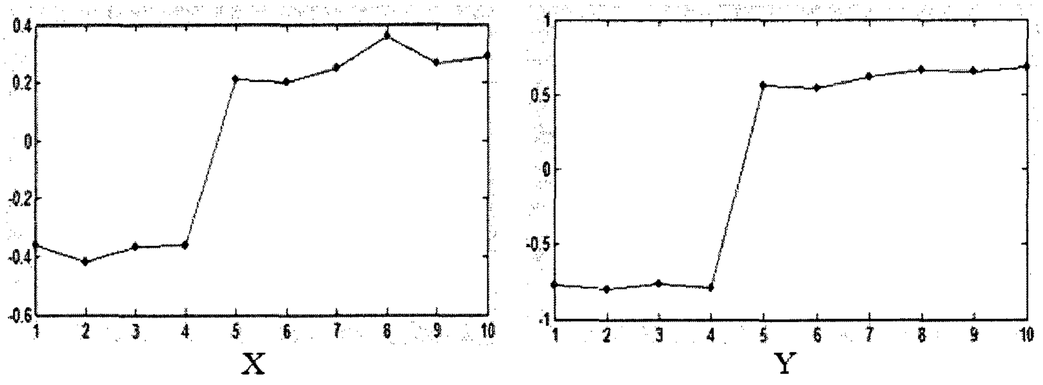


Figure 2.19: (Left) The matrix X. (Right) The matrix Y after renormalizing the rows of the matrix X.

1	2
2	2
3	2
4	2
5	1
6	1
7	1
8	1
9	1
10	1

Figure 2.20: The cluster assignments.

2.4 Chapter Summary

In this chapter, we provided an overview of one of the exploratory data mining techniques, called *Cluster Analysis.* The vast amount of data collected, processed, stored and preserved electronically, needs automated processes so that they may be explored and analyzed. The cluster analysis algorithms have proved to be a very useful method to perform this task. In this chapter, we discussed various cluster analysis methods that are present in the literature.

We focused our discussion on a graph-based cluster analysis method that considers the pair-wise similarity between the objects. The algorithms, known as the *Spectral Clustering Algorithms* often perform better than several traditional cluster analysis algorithms [46]. The spectral methods are unique, in that they manipulate the eigenvalues and eigenvectors of the similarity matrix to partition the data into groups. Moreover, they are not sensitive to the attribute type. They may be applied to datasets with numeric, nominal, binary, or mixed variables as long as the dataset is convertible into a similarity matrix. In this chapter, we presented two well-known algorithms from this category and discussed the theories associated with each of the algorithms, in detail.

Recall from our discussion that the proximity measures, such as the *similarity* and *distance* functions, play an important role in cluster analysis. Therefore, we present a detailed discussion of similarity, dissimilarity, and distance measures in the next chapter.

Chapter 3

Proximity Measures

Similarity, dissimilarity, and distance measures are crucial concepts in cluster analysis. In essence, they are also known as the *proximity measures* that quantify the distance or closeness between two data objects. As Everitt [19] says, *"the majority of clustering techniques begin with the calculation of a matrix of similarities and distances between entities, and therefore careful consideration is needed of the possible ways of defining these quantities."*. In Chapter 2, we presented the fundamental steps of a cluster analysis process (Figure 2.2). As discussed, one of the early steps of a cluster analysis method is the selection of proximity measure. However, in spectral clustering, the first step (after preprocessing the data) is not only to select a proximity measure, but also to construct a similarity matrix from the objects present in a given dataset. The similarity matrix is often created from the proximity measure. In this chapter, we present and discuss the proximity measures used in our experiments. We define the functions and present their properties. Therefore, the chapter begins in Section 3.1 with the theoretical definitions of the terms, namely, *similarity*, *dissimilarity*, and *distance.* The subsequent sections are devoted to the measures suitable for a specific data type (binary, mixed, and numeric type). The proximity measures for binary data are introduced in Section 3.2. Recall from Chapter 1, datasets may also contain attributes of mixed variable types. As such, the proximity measures for mixed variable types are introduced in Section 3.3. In Section 3.4, the discussion focuses on defining the proximity measures for numeric variable types. We conclude the chapter with a brief summary in Section 3.5.

3.1 Similarity, Dissimilarity, and Distance

In data mining, particularly in cluster analysis, *similarity*, *dissimilarity*, and *distance* measures play an important role to calculate the *proximity* between data objects. As for the spectral clustering algorithm, the entire dataset is first converted into a *similarity* matrix (also called the *affinity matrix*). The similarity matrix is constructed from the proximity measure. According to Everitt [19], the results from a cluster analysis task depend heavily on the choice of proximity measures. Therefore, different measures, when applied on the same dataset as well as the same algorithm, may provide different solutions. Moreover, the measures also impose different requirements on the same cluster analysis algorithm. There are many different measures available in the literature to calculate the proximity between data objects. One of the reasons for this variety is that these measures differ on the data type of the objects present in a given dataset. For instance, it follows that the proximity measures that are suitable for numeric variables may not be suitable for nominal data, as the attribute values from these two data types are represented differently. Therefore, a different set of measures is required to handle binary or nominal data. Moreover, the measures also differ on the properties they exhibit. Several measures consider the correlation between the variables under consideration, whereas several others discover clusters of a particular shape. In the subsequent sections we define the similarity and distance measures and discuss their properties.

Similarity is a numerical measure that represents the similarity (i.e. how alike the objects are) between two objects. The objects are represented as a collection of various features and attributes. This measure usually returns a non-negative value that falls in between 0 and 1. However, in some cases similarity may also range from -1 to $+1$. The *Pearson Coefficient Correlation* and the *Angular Separation*, as discussed later in the chapter, are two examples where the similarity may take a negative value. When the similarity takes a value zero (0), it means that there is no similarity between the objects and the objects are very different from one another. In contrast, one (1) denotes complete similarity, emphasizing that the objects are identical and possess the same attribute values. In homogeneous datasets (where all the objects are of identical type), measuring the similarity between two objects is less complex. We may use the feature vectors of the objects for calculating the similarity score. Most of the measures discussed here are designed for homogeneous objects. As for heterogeneous objects (e.g. in multi-relational dataset where the objects may be of different types), these similarity measures need special care and may need to be re-defined or modified to incorporate all the information from various object types. Our work, however, considers datasets with homogeneous objects and calculates the similarity value directly from the attribute values.

In contrast to the similarity measure, the *dissimilarity* measure [69], [32] is also a numerical measure, which represents the discrepancy or the difference between a pair objects. If two objects are very similar then the dissimilarity measure will have a lower value, whereas if the objects are very different from one another, this measure will return a higher numeric value. Therefore, the measure is reversely related to the similarity measure. As such, when the similarity between two objects is high, the dissimilarity will be low and vise versa. As with the similarity score, the dissimilarity value also fall into the interval $[0, 1]$, but it may also take values ranging from -1 to $+1$.

The term *distance*, which is also commonly used as a synonym for the *dissimilarity measure* [47], computes the distance between two data points in a multi-dimensional space. Unlike dissimilarity metrics, which usually have a value in between 0 and 1 (but sometimes ranges from -1 to $+1$), the distance measures always take a positive value between 0 and ∞. The distance measures also satisfy the following four properties [32], [44]:

1. $d(x, y) = d(y, x)$, for all points x and y. For instance, the distance from point x to point y is same as the distance from point y to point x.

2. $d(x, y) = 0$, if $x = y$. Distance is only 0 when both the coordinates are same.

3. $d(x, y) \geq 0$, for all points x and y. The distance is always non-negative.

4. $d(x, y) \leq d(x, z) + d(z, y)$, for all points x, y and z. This is also known as the *Triangle Inequality.* This implies that introducing a third point may never shorten the distance between the two other points [44].

Similarity and distance are, in a sense, inversely related to one another. When the distance in between two objects is large (meaning that the objects are different from one another), the similarity will be low. Conversely, when the distance is low the similarity will be high. Since it is inversely related, a common way to transform a distance measure to a similarity measure is by using the equation $s(i, j) = \frac{1}{d(i,j)}$, where i and j are two objects. However, one of the problems with this equation is that the similarity value will not always fall into the range $[0, 1]$. For instance, if the distance between two objects is very small, such as 0.25, then the similarity value for these two objects will be 4 ($\frac{1}{0.25} = 4$). There are various other ways to transform a distance or dissimilarity measure

to a similarity measure such that the values for similarity measure ranges from 0 to 1. We will refer to these functions as the *external* or *secondary* functions to distinguish them from the functions discussed in this study.
If dissimilarity scores fall in between 0 and 1 then similarity is calculated using the following formula:

$$similarity = 1 - dissimilarity \tag{3.1}$$

However, if the value for a distance measure is greater than 1, then there are different ways to transform a distance measure into a similarity measure. One such adaptation that has been used in this study, is the function given in Equation 3.2. One of the reasons for selecting this function, is that the function monotonously falls with the increasing distance, which later simplifies the analysis of eigenvalues [58]. Recall from Chapter 2, eigenvalues and eigenvectors are the two important concepts in spectral clustering algorithm. We discuss the properties of this function in detail in Chapter 4. This function is also known as the *Gaussian* function.

$$s(x,y) = \exp(\frac{-d(x,y)^2}{2 * \sigma^2}) \tag{3.2}$$

In the above equation,

$s(x,y)$ = similarity between points x and y

$d(x,y)$ = distance between points x and y

σ = a user specified scaling variable

There are several other ways to convert a distance measure into a similarity measure, as stated below:

$$s(x,y) = \frac{1}{1 + d(x,y)^2} \tag{3.3}$$

$$s(x,y) = \frac{1}{1 + d(x,y)} \tag{3.4}$$

$$s(x,y) = 1 - d(x,y)(d(x,y) \in [0,1]) \tag{3.5}$$

As mentioned previously, the distance and similarity measures vary on the type of data. In the next section, we discuss the proximity measures suitable for the datasets with binary attributes.

3.2 Proximity Measures for Binary Variables

Binary variables take only two values, such as: 0 (negative) and 1 (positive), *yes* (positive) and *no* (negative), or *agree* and *disagree.* These variables are usually categorized into two types: 1) *symmetric binary variables* where both the positive and the negative values carry equal weight and 2) *asymmetric binary variables* where the positive and the negative values do not carry equal weight, and one (usually the positive value) carries more weight than the other. Let x and y be two binary data points. Each proximity measure for binary data is represented by four variables (a, b, c, d):
a = number of occurrences of $x_i = 1$ and $y_i = 1$ (positive matches),
b = number of occurrences of $x_i = 0$ and $y_i = 1$ (disagreement),
c = number of occurrences of $x_i = 1$ and $y_i = 0$ (disagreement),
d = number of occurrences of $x_i = 0$ and $y_i = 0$ (negative matches),
and $a + b + c + d = p$ (total number of attributes in x and y).

The similarity coefficients used for the binary variables mostly originated from the area of *numerical taxonomy.* According to Sokal and Sneath [59], numerical taxonomy is *"the numerical evaluation of the affinity or similarity between taxonomic units and the ordering of these units into taxa on the basis of there affinities.".* Therefore, numerous similarity coefficients were proposed by various researchers to calculate the proximities, which are also equally applicable to fields including data mining and statistics. A number of such coefficients give equal weight to the positive and negative values, whereas several coefficients ignore the negative matches. As such, for the same set of data, different coefficients may give different similarity values [19]. According to Everitt [19], *"The number of proposed association coefficients is large, mainly because of the uncertainty over how to incorporate negative matches into the coefficients".* The coefficients differ mostly on the preference given by the researchers regarding issues related to the weights given on the positive and negative values as well as the degree of weight given to each of the four variables. In this study, we present the coefficients that are suitable for clustering binary data, according to the discussion presented in [39],[69],[19]. All the coefficients in this study ranges from 0 to 1. We use the sample dataset given in Table 3.1 to compute the coefficients.

Object ID	Attribute 1	Attribute 2	Attribute 3	Attribute 4
Object 1	0	1	1	1
Object 2	1	1	1	1
Object 3	1	0	0	0

Table 3.1: Sample dataset for binary data type.

3.2.1 Jaccard Coefficient

The *Jaccard* coefficient does not consider the negative matches. In terms of the four variables defined above, the Jaccard similarity coefficient is defined by Equation 3.7. Recall that, a denotes the number of *positive matches* whereas, b and c denote the total number of *disagreements*.

$$dis_{Jaccard} = \frac{b+c}{a+b+c} \tag{3.6}$$

$$sim_{Jaccard} = \frac{a}{a+b+c} \tag{3.7}$$

The values range from 0 to 1. The maximum similarity is achieved when $b = c = 0$ and the minimum similarity is achieved when there are no positive matches (when $a = 0$). The above equation does not contain d, which represents the negative matches. The argument behind not including d in the measurement is that, if the negative values are not important, then counting the occurrences where $x_i = 0$ and $y_i = 0$ may not have meaningful contribution toward the calculation of similarity measure [64], [55]. For the asymmetric binary variables where either positive or negative matches are given more preference, the Jaccard coefficient may perform best. This is because, most of the time positive values are given more weight, whereas negative values are considered unimportant [32]. However, for the symmetric binary variables, where the positive and the negative matches carry equal weight, the Jaccard coefficient may not give correct result. Moreover, the Jaccard coefficient is very sensitive to the direction of coding which implies that interchanging the meaning of *1* and *0* may affect the similarity between the objects [55].

Example 3.2.1. The dissimilarity and the similarity between Object 1 and Object 2:

$$dis_{1,2} = \frac{1+0}{3+1+0} = \frac{1}{4} = 0.25$$

$$sim_{1,2} = 1 - 0.25 = 0.75$$

The dissimilarity and the similarity between Object 1 and Object 3:

$$dis_{1,3} = \frac{1+3}{0+1+3} = \frac{4}{4} = 1.0$$

$$sim_{1,3} = 1 - 1 = 0$$

3.2.2 Czekanowski Coefficient

The *Czekanowski* similarity coefficient is also known as the *Dice* or *Sorenson* coefficient. The function is given in Equation 3.9. Recall that, a denotes the total number of *positive matches*. The total numbers of *disagreements* are denoted with the variables b and c.

$$dis_{czekanowski} = \frac{b+c}{2a+b+c} \tag{3.8}$$

The equation may be modified to represent similarity as follows:

$$sim_{czekanowski} = \frac{2a}{2a+b+c} \tag{3.9}$$

The coefficient is similar to the *Jaccard* coefficient. However, double weight is given to the variable a which denotes the total number of occurrences of the positive matches. By giving twice the weight to a, the function gives more emphasis to the positive matches. Variable d (when $x = 0$ and $y = 0$) is not present in this measure. The Czekanowski coefficient is also suitable for the asymmetric binary variables for the similar reasons as the Jaccard coefficient. Interchanging the meaning of positive and negative values may also affect the score.

Example 3.2.2. The dissimilarity and the similarity between Object 1 and Object 2:

$$dis_{1,2} = \frac{1+0}{2*3+1+0} = \frac{1}{7} = 0.1429$$

$$sim_{1,2} = 1 - 0.1429 = 0.8571$$

The dissimilarity and the similarity between Object 1 and Object 3:

$$dis_{1,3} = \frac{1+3}{2*0+1+3} = \frac{4}{4} = 1.0$$

$$sim_{1,3} = 1 - 1 = 0$$

3.2.3 Sokal and Sneath Coefficient

Sokal and Sneath proposed a similarity coefficient that is similar to the ones proposed by Jaccard and Czekanowski. This measure is defined as:

$$dis_{SokalandSneath} = \frac{b+c}{\frac{a}{2}+b+c} \tag{3.10}$$

$$sim_{SokalandSneath} = \frac{\frac{a}{2}}{\frac{a}{2}+b+c} = \frac{a}{a+2(b+c)} \tag{3.11}$$

However, in contrast to the Czekanowski coefficient which gives double weight to the positive matches (a), the Sokal and Sneath coefficient gives double weight to the disagreements in the denominator. The disagreements are represented by the variables b and c as denoted earlier. Thus, the Sokal and Sneath coefficient gives twice the weight on the combined disagreements denoted by $b+c$. By doing so, the coefficient actually gives slightly less weight to the positive matches compared to the Jaccard and Czekanowski coefficients. Similar to the Jaccard and Czekanowski coefficients, this coefficient is also suitable for the asymmetric binary variables and is also sensitive to the direction of coding.

Example 3.2.3. The dissimilarity and the similarity between Object 1 and Object 2:

$$dis_{1,2} = \frac{1+0}{1/2*3+1+0} = \frac{2}{5} = 0.4$$

$$sim_{1,2} = 1 - 0.4 = 0.6$$

The dissimilarity and the similarity between Object 1 and Object 3:

$$dis_{1,3} = \frac{1+3}{1/2*0+1+3} = \frac{4}{4} = 1.0$$

$$sim_{1,3} = 1 - 1 = 0$$

3.2.4 Simple Matching Coefficient

The *Simple matching coefficient* [69], also known as the *Hamming distance*, denotes the proportion of variables for which two variables have the same value [69]. As mentioned earlier, the variables a and d denote the total number of positive and negative matches, respectively. The variables b and c denote the total number of disagreement.

$$dis_{SimpleMatchingCoefficient} = \frac{b+c}{a+b+c+d} \tag{3.12}$$

$$sim_{SimpleMatchingCoefficient} = \frac{a+d}{a+b+c+d} \tag{3.13}$$

The Simple Matching Coefficient considers both, the positive matches (a) and the negative matches (d). Moreover, it gives equal weight to the positive and negative matches. Therefore, the Simple Matching Coefficient is suitable for datasets with symmetric binary attribute values where both the positive and the negative matches posses equal weight. Unlike the Jaccard, Czekanowski, and Sokal and Sneath coefficients, which are sensitive when the meanings of positive and negative values are interchanged, the Simple Matching Coefficient is not sensitive to such a situation as it gives equal weigh to both the values. The coefficient achieves the maximum similarity value when $b = c = 0$ and minimum similarity score when $a = d = 0$.

Example 3.2.4. The dissimilarity and the similarity between Object 1 and Object 2:

$$dis_{1,2} = \frac{1+0}{3+1+0+0} = \frac{1}{4} = 0.25$$

$$sim_{1,2} = 1 - 0.25 = 0.75$$

The dissimilarity and the similarity between Object 1 and Object 3:

$$dis_{1,3} = \frac{1+3}{0+1+3+0} = \frac{4}{4} = 1.0$$

$$sim_{1,3} = 1 - 1 = 0$$

3.2.5 Russell and Rao Coefficient

The Russell and Rao similarity coefficient is sometimes known as the *Positive matching coefficient* [69]. The similarity function is defined in Equation 3.15.

$$dis_{RussellandRao} = \frac{b+c+d}{a+b+c+d} \tag{3.14}$$

$$sim_{RussellandRao} = \frac{a}{a+b+c+d} \tag{3.15}$$

Unlike the Simple Matching Coefficient, which gives the proportion of both the positive (a) and the negative matches (d), the Russell and Rao coefficient gives the proportion of the positive matches against the total number of variables (including the negative matches). The coefficient is also sensitive to the meaning of positive and negative values. If the values are interchanged, then it will represent the proportion of the negative matches. The Russell and Rao coefficient achieves the maximum similarity when $b = c = d = 0$ (when there are only positive matches present) and scores the minimum when $a = 0$ (when there are no positive matches).

Example 3.2.5. The dissimilarity and the similarity between Object 1 and Object 2:

$$dis_{1,2} = \frac{1+0+0}{0+3+1+0} = \frac{1}{4} = 0.25$$
$$sim_{1,2} = 1 - 0.25 = 0.75$$

The dissimilarity and the similarity between Object 1 and Object 3:

$$dis_{1,3} = \frac{1+3+0}{0+1+3+0} = \frac{4}{4} = 1.0$$
$$sim_{1,3} = 1 - 1 = 0$$

3.2.6 Rogers and Tanimoto Coefficient

The coefficient proposed by Rogers and Tanimoto is defined in Equation 3.17.

$$dis_{RogersandTanimoto} = \frac{b+c}{\frac{(a+d)}{2} + b + c} \tag{3.16}$$

$$sim_{RogersandTanimoto} = \frac{\frac{(a+d)}{2}}{\frac{(a+d)}{2} + b + c} = \frac{a+d}{(a+d) + 2(b+c)} \tag{3.17}$$

The Rogers and Tanimoto coefficient is similar to the Simple Matching Coefficient. In this case also, the similarity coefficient considers both the positive and negative matches in the equation and gives equal weight to them. However, in contrast to the Simple Matching Coefficient, the Rogers and Tanimoto coefficient gives double weight to the variables that represent the disagreements in the denominator (i.e. the variable b and c). Therefore, the Rogers and Tanimoto coefficient always scores less than the Simple Matching Coefficient, except when $b + c = 0$ [59].

Example 3.2.6. The dissimilarity and the similarity between Object 1 and Object 2:

$$dis_{1,2} = \frac{1+0}{1/2 * (3+0) + 1 + 0} = \frac{2}{5} = 0.4$$
$$sim_{1,2} = 1 - 0.4 = 0.6$$

The dissimilarity and the similarity between Object 1 and Object 3:

$$dis_{1,3} = \frac{1+3}{1/2 * (0+0) + 1 + 3} = \frac{4}{4} = 1.0$$
$$sim_{1,3} = 1 - 1 = 0$$

A summary of the results from different similarity measures for the sample data is given in Table 3.2. In the following subsection we compare these similarity coefficients for the binary variables and show the interrelation between each of these similarity measures.

Similarity Measures	Similarity between Object 1 and Object 2	Similarity between Object 1 and Object 3
Czekanowski	0.8571	0.0
Jaccard	0.75	0.0
Sokal and Sneath	0.6	0.0
Simple Matching Coefficient	0.75	0.0
Russell and Rao	0.75	0.0
Rogers and Tanimoto	0.6	0.0

Table 3.2: Sample result from different similarity measures (binary variables).

3.2.7 Comparison of Similarity Measures

The formulas given in the previous sections for the binary data types show that the equations may be categorized into three subgroups based on the presence and absence of the four terms (a, b, c and d) in each equation. The groups are given below. Abbreviations for each of these measures are also given and will be used in later chapters.

First Group: Czekanowski (CZE), Jaccard (JAC), Sokal and Sneath (SAS)
Coefficients in this category do not consider d, which denotes the negative matches. If 1 represents *presence* and 0 represents *absence*, then this group does not include the cases when values for both the objects are 0. Usually, in binary data, the values represent *presence - absence*, *agree - disagree* or *yes - no* relationships, assuming that the positive answers (*presence, agree, yes*) carry more information than the negative ones (*absence, disagree or no*). Therefore, coefficients from this group work well for situations where the positive matches (the ones with the most important outcome) are given more value (i.e. asymmetric binary variables). An example of this would be the results from a medical diagnosis in which the positive and negative outcomes are not equal. However, if the data gives equal weight to both positive and negative values then these equations are not expected to do well, e.g. the Gender attribute where male and female may carry equal weight. Similarity measures in this group are also correlated in the sense that both the Czekanowski and Sokal and Sneath coefficients may be written as a function of the Jaccard coefficient (Figure 3.1). Moreover, all three coefficients from this group are very sensitive if the values are interchanged. For instance, if 1 (which usually contains more

Czekanowski re-written as a function of Jaccard	*Sokal and Sneath re-written as a function of Jaccard*
$\frac{2a}{2a+b+c}$ $=\frac{2a}{a+b+c}*\frac{a+b+c}{2a+b+c}$ $=\frac{2a}{a+b+c}/\frac{2a+b+c}{a+b+c}$ $=\frac{2a}{a+b+c}/(\frac{a}{a+b+c}+\frac{a+b+c}{a+b+c})$ $=\frac{2a}{a+b+c}/(\frac{a}{a+b+c}+1)$ $=2*jac/(jac+1)\ [jac=\frac{a}{a+b+c}]$ $=\frac{2*jac}{jac+1}$	$\frac{\frac{1}{2}a}{\frac{1}{2}a+b+c}$ $=\frac{\frac{1}{2}a}{a+b+c}*\frac{a+b+c}{\frac{1}{2}a+b+c}$ $=\frac{\frac{1}{2}a}{a+b+c}*\frac{2}{2}*\frac{a+b+c}{\frac{1}{2}a+b+c}$ $=\frac{a}{a+b+c}*\frac{a+b+c}{a+2b+2c}$ $=\frac{a}{a+b+c}/\frac{a+2b+2c}{a+b+c}$ $=\frac{a}{a+b+c}/(\frac{a}{a+b+c}+\frac{2b+2c}{a+b+c})$ $=\frac{a}{a+b+c}/(\frac{a}{a+b+c}+\frac{2b+2c+2a-2a}{a+b+c})$ $=\frac{a}{a+b+c}/(\frac{a}{a+b+c}+\frac{2a+2b+2c}{a+b+c}-\frac{2a}{a+b+c})$ $=jac/(jac+2-2*jac)\ [jac=\frac{a}{a+b+c}]$ $=\frac{jac}{2-jac}$

Figure 3.1: The *Czekanowski* and *Sokal and Sneath* coefficient re-written as a function of the *Jaccard* coefficient.

weight) is interchanged with 0 (which usually denote less weight) then, these measures will not give correct answer.

Second Group: Simple matching coefficient (SIM), Roger and Tanimoto (RAT)

Similarity measures in this subgroup give equal weight to both the variables a and d. In contrast to other similarity measures, it will give equal importance to both positive and negatives values. Therefore, these coefficients are particularly suitable for symmetric binary attributes. For example, if an attribute represents 1 as a female population and 0 as a male population then these measures will do justice to the attribute values by

Rogers and Tanimoto re-written as a function of Simple Matching Coefficient

$$\frac{\frac{1}{2}(a+d)}{\frac{1}{2}(a+d)+b+c}$$

$$=\frac{\frac{1}{2}(a+d)}{a+b+c+d}*\frac{2}{2}*\frac{a+b+c+d}{\frac{1}{2}(a+d)+b+c}$$

$$=\frac{(a+d)}{a+b+c+d}*\frac{a+b+c+d}{a+d+2b+2c}$$

$$=\frac{(a+d)}{a+b+c+d}/\frac{(a+d)+2b+2c}{a+b+c+d}$$

$$=\frac{(a+d)}{a+b+c+d}/(\frac{(a+d)}{a+b+c+d}+\frac{2b+2c}{a+b+c+d})$$

$$=\frac{(a+d)}{a+b+c+d}/(\frac{(a+d)}{a+b+c+d}+\frac{2b+2c+2a-2a+2d-2d}{a+b+c+d})$$

$$=\frac{(a+d)}{a+b+c+d}/(\frac{(a+d)}{a+b+c+d}+\frac{2a+2b+2c+2d}{a+b+c+d}-\frac{2a+2d}{a+b+c+d})$$

$$=\frac{(a+d)}{a+b+c+d}/(\frac{(a+d)}{a+b+c+d}+2-\frac{2(a+d)}{a+b+c+d})$$

$$=sim/(sim+2-2*sim)\quad [sim=\frac{(a+d)}{a+b+c+d}]$$

$$=\frac{sim}{2-sim}$$

Figure 3.2: The *Rogers and Tanimoto* coefficient re-written as a function of the *Simple Matching Coefficient.*

giving them equal weight. However, in situations where 1 strictly considers cases with *presence* or *agree* and have more importance, this will not give expected result. This means that two objects with no common pairs of 1 ($a = 0$) but with many common pairs of 0 ($d > 0$), will have a higher value for these two coefficients. The Rogers and Tanimoto coefficient may be re-written as a function of the *Simple Matching Coefficient* as given in Figure 3.2 below. Since, both the coefficients give equal weight to the positive and negative matches, they are not sensitive if the values are interchanged.

Third Group: Russell and Rao (RAR)

The Russell and Rao coefficient gives the proportion of the matching cases where both the attribute values are 1*s*. In contrast to the Simple Matching Coefficient, which gives the matching cases for the instances that have values 00*s* and 11*s*, it will give a higher value for a pair of objects when there are many 11*s*. The Russell and Rao similarity coefficient is also suitable for the asymmetric binary variables where more weight is given to the positive matches. Since the equation does not consider d (the negative matches) in the numerator, it may not be suitable for the symmetric binary attribute. The coefficient is also sensitive to the meaning of positive and negative values for the same reason.

In the next section, we discuss the proximity measures for the mixed variable types.

3.3 Proximity Measures for Mixed Variables

In the previous section, the discussion mostly focused on datasets of a particular variable type (e.g. binary). Nevertheless, in practical applications, it is possible to have more than one type of attribute in the same dataset. For instance, a dataset may have numeric and binary attributes to describe the objects. In such cases, the conventional proximity measures for these two data types may not work well, as they are suitable to deal with one kind of variable at a time. In cluster analysis, one way to deal with the mixed data type is to perform separate cluster analysis for each kind of variable and then combine the results if the conclusions all agree [39]. However, research suggests that if the conclusions from different variable types do not agree then it would be difficult to reconcile them. As such, a more practical approach is to process all the variables of different types together and then perform a single cluster analysis [39]. Therefore, some similarity measures are proposed that incorporate information from various data types into a single similarity coefficient. The coefficients present in literature to calculate the similarity for mixed data type are, the *Gower's General Dissimilarity Coefficient* [39], [32] and the *Laflin's General Coefficient* [43], [39].

3.3.1 Gower's General Dissimilarity Coefficient

The dissimilarity measure was originally introduced by Gower and discussed in [32]. The function is defined as follows (where i and j are two data objects and f is the set of attributes):

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}} \tag{3.18}$$

Where the indicator $\delta_{ij}^{(f)} = 0$ if either,

1. x_{if} or x_{jf} is missing (x is the attribute value) or

2. $x_{if} = x_{jf} = 0$ and f is Asymmetric Binary.

Otherwise, $\delta_{ij}^{(f)} = 1$. The distance between object i and j for a variable f is calculated using different distance measures that already exist for various attribute types. For example:

- If f is Numeric: $d_{ij}^{(f)} = \frac{|x_{if}-x_{jf}|}{max_h x_{hf} - min_h x_{hf}}$, where h runs over all the non-missing objects of attribute f. Therefore, this measure is not scale sensitive for the numeric variables.

- If f is Binary or Nominal: if $x_{if} = x_{jf}$ then $d_{ij}^{(f)} = 0$; otherwise $d_{ij}^{(f)} = 1$.

- If f is Ordinal: First we compute the rank r_{if} for object i assuming that the attribute f has M_f ordered states and $r_{if} \in 1, ...M_f$. This is done by replacing x_{if} by its corresponding rank. The rank is determined by the significance of the attribute value. For example: an attribute with values *First*, *Second* and *Third* will have a rank 3, 2 and 1 respectively. Thus, a value with the most weight will have the highest rank and an attribute with less weight will have a lower rank. Once ranking is performed on the objects, the values need to be normalized so that each of them falls in the interval $[0.0, 1.0]$. This is done by using the following formula: $z_{if} = \frac{r_{if}-1}{M_f-1}$. Here, z_{if} is treated as a numeric variable and a distance metric for the numeric variable (as discussed in Section 3.4) is used to calculate the distance between the objects.

- If f is Ratio-scaled (According to Han and Kamber [32], *"a ratio-scaled variable makes a positive measurement on a nonlinear scale, such as an exponential scale."*), then the distance between its objects may be calculated in one of two ways. Firstly by performing a logarithmic transformation and treating this transformed data as numeric values, or secondly, by treating f as continuous ordinal data and calculating the distance as mentioned above.

Example 3.3.1. For this example, we used a dataset (Table 3.3) similar to the one given in [32].
To calculate the similarity between Object 1 and Object 2; we proceed as follows:
Attribute 1 (Numeric): $max = 12$ and $min = 3$

$$d_{1,2}^{(Attribute1)} = \frac{|12-9|}{|12-3|} = 0.3333$$

	Attribute 1 (Numeric)	Attribute 2 (Numeric)	Attribute 3 (Nominal)	Attribute 4 (Nominal)	Attribute 5 (Ordinal)	Attribute 6 (Ordinal)
Object 1	12	10	A	A	Good	First
Object 2	9	12	A	A	Excellent	First
Object 3	3	4	B	C	Fair	Third

Table 3.3: Sample dataset for mixed data type.

Attribute 2 (Numeric): $max = 12$ and $min = 4$

$$d_{1,2}^{(Attribute2)} = \frac{|10 - 12|}{|12 - 4|} = 0.25$$

Attribute 3 (Nominal):

$$d_{1,2}^{(Attribute3)} = 0$$

Attribute 4 (Nominal):

$$d_{1,2}^{(Attribute4)} = 1$$

Attribute 5 (Ordinal): rank: $Fair = 1$, $Good = 2$ and $Excellent = 3$ and $M_f = 3$
The normalized values for Attribute 5 will be:
$Object1 = \frac{2-1}{3-1} = 0.5$, $Object2 = \frac{3-1}{3-1} = 1$, $Object3 = \frac{1-1}{3-1} = 0$

$$d_{1,2}^{(Attribute5)} = \frac{|0.5 - 1|}{1 - 0} = 0.5$$

Attribute 6 (Ordinal):
rank: $Third = 1$, $Second = 2$ and $First = 3$ and $M_f = 3$
The normalized values for Attribute 6 will be:
$Object1 = \frac{3-1}{3-1} = 1$, $Object2 = \frac{3-1}{3-1} = 1$, $Object3 = \frac{1-1}{3-1} = 0$

$$d_{1,2}^{(Attribute6)} = \frac{|1 - 1|}{1 - 0} = 0$$

The total dissimilarity between Object 1 and Object 2 are thus calculated as,

$$d_{1,2} = \frac{(1 * 0.3333) + (1 * 0.25) + (1 * 0) + (1 * 1) + (1 * 0.5) + (1 * 0)}{1 + 1 + 1 + 1 + 1 + 1} = \frac{2.0833}{6} = 0.3472$$

Next, the similarity may be derived by using Equation 3.5 as follows:

$$similarity_{1,2} = 1 - 0.3472 = 0.6528$$

Now, to calculate the similarity between Object 1 and Object 3, we follow a similar method. Therefore, the total dissimilarity and similarity between Object 1 and Object 3 are as follows.

$$d_{1,3} = \frac{(1*1)+(1*0.75)+(1*1)+(1*1)+(1*0.5)+(1*1)}{1+1+1+1+1+1} = \frac{5.25}{6} = 0.8750$$

$$similarity_{1,3} = 1 - 0.8750 = 0.1250$$

3.3.2 Laflin's General Coefficient

The Laflin's coefficient is measured as follows. Let there be $N1$ Binary attributes and $N2$ Numeric attributes in a dataset. Let $s1$ and $s2$ be the similarity measures calculated for the Binary and the Numeric data respectively using some existing similarity measures (as discussed in Section 3.2 and Section 3.4 respectively). Then Laflin's coefficient [43] is calculated as follows:

$$s(i,j) = \frac{N1.s1 + N2.s2}{N1 + N2} \tag{3.19}$$

This function may be extended to include additional data types in a similar manner. For example, if each instance in a dataset contains four types of variables (i.e. *Binary*, *Numeric*, *Ordinal* and *Nominal*) then $N1$, $N2$, $N3$ and $N4$ will represent the total number of attributes for these four types of variables, respectively. Next, we calculate the similarity between each pair of objects using existing similarity measures, as discussed earlier, for each of these set of attributes separately. Let $s1$, $s2$, $s3$ and $s4$ be the similarity measure associated with the set of attributes $N1$, $N2$, $N3$ and $N4$, respectively. All these similarity values should be scaled so that they fall in between 0 and 1. The general similarity coefficient for this mixed set of attributes is calculated as:

$$s(i,j) = \frac{N1.s1 + N2.s2 + N3.s3 + N4.s4}{N1 + N2 + N3 + N4} \tag{3.20}$$

This equation ensures that each attribute makes an equal contribution to the measure of similarity between two objects i and j [43].

Example 3.3.2. For the dataset given in Table 3.3, Laflin's coefficient is calculated as follows.

There are three different variable types in this dataset each type containing 2 variables. Thus, $N1 = N2 = N3 = 2$. To calculate the distance between nominal variables we use the formula given in [32]:

$$d(i,j) = \frac{p-m}{p} \tag{3.21}$$

Here p is the total number of variables and m is the number of variables for which i and j have the same value.

For numeric variables, the Euclidean distance measure as defined in Equation 3.22 is used and for all the cases distance measure is converted into a similarity measure by using Equation 3.4.

The similarity between Object 1 and Object 2 is calculated as follows.

Numeric variables:

$$d1 = \sqrt{(12-9)^2 + (10-12)^2} = 3.4641$$
$$S1 = \frac{1}{1+d1} = \frac{1}{1+3.4641} = 0.2240$$

Nominal variables: $P = 2$ (total number of variables of type nominal)

$$d2 = \frac{2-2}{2} = 0$$
$$S2 = \frac{1}{1+d2} = \frac{1}{1+0} = 1$$

Ordinal variables:

$d3 = 0.5$ (same as the example given for Gower's Coefficient)

$$S3 = \frac{1}{1+d3} = \frac{1}{1+0.5} = 0.6667$$

When substituting the values of S1, S2 and S3 in Equation 3.20, we obtain:

$$similarity_{1,2} = \frac{(2*0.2240) + (2*1) + (2*0.6667)}{2+2+2} = 0.6302$$

To calculate the similarity between Object 1 and Object 3, we proceed as follows.

Numeric variables:

$$d1 = \sqrt{(12-3)^2 + (10-4)^2} = 10.8167$$
$$S1 = \frac{1}{1+d1} = \frac{1}{1+10.8167} = 0.0846$$

Nominal variables:

$$d2 = \frac{2-0}{2} = 1$$
$$S2 = \frac{1}{1+d2} = \frac{1}{1+1} = 0.5$$

Nominal variables:

$$d3 = 1.118$$
$$S3 = \frac{1}{1 + d3} = \frac{1}{1 + 1.118} = 0.4721$$

When substituting the values of S1, S2 and S3 in Equation 3.20, we obtain:

$$similarity_{1,3} = \frac{(2 * 0.0846) + (2 * 0.5) + (2 * 0.4721)}{2 + 2 + 2} = 0.3522$$

Table 3.4 summarizes the results for the Similarity Measures for Mixed Type:

Similarity Coefficient	Similarity between Object 1 and Object 2	Similarity between Object 1 and Object 3
Gower's General Coefficient	0.6528	0.1250
Laflin's General Coefficient	0.6302	0.3522

Table 3.4: Sample result from different similarity measures (mixed variable type).

As before, both the similarity measures for the mixed data types have been able to distinguish between objects that are similar to one another as well as the objects that are different from one another. This is denoted by a higher value for objects that are comparatively similar to one another and a lower value for the objects that are different from one another. It is shown in Table 3.4 that Object 1 and Object 2 represent two such objects that are closely related, whereas Object 1 and Object 3 are very different from one another scoring a lower similarity value.

In the next section, we discuss the distance measures that are applicable to the objects with numeric, continuous, or interval-scaled variables.

3.4 Proximity Measures for Numeric Variables

There exist several distance measures for numeric or real-valued data. We present our discussion based on the measures presented in [69], [54], [64]. For the purpose of clarification, we provide an example that shows the calculations for each of these distance measures. We use the sample dataset given in Table 3.5, which contains three data objects, and each of the objects is represented with four features.

Object ID	Attribute 1	Attribute 2	Attribute 3	Attribute 4
Object 1	10	5	8	2
Object 2	11	6	9	1
Object 3	1	20	0	8

Table 3.5: Sample dataset for numeric data type.

3.4.1 Euclidean Distance

The *Euclidean distance* is one of the most widely used distance measures in the area of cluster analysis [19]. The distance, in this case, is the straight-line distance between a given pair of data points. The distance is calculated as the summation of the differences between the coordinates of the data points x_i and x_j. The function is denoted as Equation 3.22.

$$d_{x_i,x_j} = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2} \tag{3.22}$$

Jain et al.[35], stated that the Euclidean distance works well for compact or isolated clusters and discovers clusters of spherical shape. The K-means algorithm often uses this distance measure to compute the distance between the objects and the cluster centroids. Since *triangle inequality* holds for the distance measure, the distance between any two objects may not be influenced by the addition of a new object (i.e. outliers) [60]. However, one drawback is, when the underlying clusters are not isolated, compact, or clusters of spherical shape, this distance measure may not be suitable. Moreover, the distance measure is very sensitive to the scales of the variables. As a result, the variables with the largest values may always dominate the distance score. For instance, let there be two variables p and q, where p ranges between 0 and 100 and q ranges in $[0 - 1]$. When the distance is calculated, taking these two variables into consideration, the Euclidean distance will usually be dominated by the variable p. Therefore, it is necessary to scale all the variables in the dataset into similar units.

Example 3.4.1. The distance between Object 1 and Object 2 is calculated as:

$$d_{1,2} = \sqrt{(10-11)^2 + (5-6)^2 + (8-9)^2 + (2-1)^2} = 2$$

The distance between Object 1 and Object 3 is calculated as:

$$d_{1,3} = \sqrt{(10-1)^2 + (5-20)^2 + (8-0)^2 + (2-8)^2} = 20.1494$$

3.4.2 Manhattan Distance

The *Manhattan distance* is also commonly known as the *city-block* distance. Unlike the Euclidean distance, which takes the straight-line distance between the objects, the Manhattan distance measure would travel from one point to another as if a grid-like path is followed. It is the summation of absolute differences between the coordinates of two data points (x_i and x_j).

$$d_{x_i,x_j} = \sum_{k=1}^{n} |x_{ik} - x_{jk}| \tag{3.23}$$

The Euclidean and Manhattan distance measures are very similar to one another. However, compared to the Euclidean distance, the Manhattan distance is computationally cheaper (as the variables are not squared) and may be selected over the Euclidean distance when the speed of an application is the main concern [54]. For similar reasons, it is less likely that variables with large differences may dominate the total distance [60]. However, the weaknesses of the Euclidean distance (i.e. scale-dependent) also persist for the Manhattan distance. Everitt [19] noted that this measure is particularly suitable for situations when *"two entities are specified by two variables whose scale units are of equal value, they should have the same distance whether (a) they are two units apart on each variable or (b) they are one unit apart on one variable and three units apart on the other"*. For instance, let there be two objects, p denoted by variables $(2, 4)$ and q denoted by variables $(4, 6)$. Then, the Manhattan distance between p and q is: $2 + 2 = 4$. Now, if $p = (2, 4)$ and $q = (3, 7)$, then the distance is $1 + 3 = 4$. In contrast, for these two situations, the Euclidean distance would score, $\sqrt{2^2 + 2^2} = \sqrt{8}$ and $\sqrt{1^2 + 3^2} = \sqrt{10}$, respectively.

Example 3.4.2. The distance between Object 1 and Object 2 is calculated as:

$$d_{1,2} = |10 - 11| + |5 - 6| + |8 - 9| + |2 - 1| = 4$$

The distance between Object 1 and Object 3 is calculated as:

$$d_{1,3} = |10 - 1| + |5 - 20| + |8 - 0| + |2 - 8| = 38$$

3.4.3 Minkowski Distance

The *Minkowski distance* is defined in Equation 3.24.

$$d_{x_i,x_j} = \left(\sum_{k=1}^{n} |x_{ik} - x_{jk}|^{\lambda} \right)^{\frac{1}{\lambda}} \tag{3.24}$$

In Equation 3.24, λ may take any value greater than 0. Depending on the value of λ, the Minkowski distance may take several different forms. For instance, when $\lambda = 1$, the Minkowski distance is similar to the Manhattan distance, whereas when $\lambda = 2$, the Minkowski distance is similar to the Euclidean distance. Therefore, one advantage of using this distance measure is the flexibility of controlling the amount of emphasis one may want to place on the larger differences [69]. A large value of λ indicates larger difference. However, a larger value of λ also indicates that the largest scale would dominate the total distance. Moreover, computationally, the Minkowski distance may cost more than the Euclidean and the Manhattan distance when $\lambda > 2$.

Example 3.4.3. The distance between Object 1 and Object 2 is calculated as (when $\lambda = 3$):

$$d_{1,2} = \sqrt[3]{|10-11|^3 + |5-6|^3 + |8-9|^3 + |2-1|^3} = 1.587$$

The distance between Object 1 and Object 3 is calculated as (when $\lambda = 3$):

$$d_{1,3} = \sqrt[3]{|10-1|^3 + |5-20|^3 + |8-0|^3 + |2-8|^3} = 16.9061$$

3.4.4 Chebyshev Distance

The *Chebyshev distance* is a special case of the Minkowski distance with $\lambda = \infty$. In this case, the distance is measured as the distance between the coordinates of two data points where the absolute distance between the points in any single dimension is maximized.

$$d_{x_i,x_j} = max_k|x_{ik} - x_{jk}| \tag{3.25}$$

Since the function returns the maximum difference across all the variables, the Chebyshev distance is suitable for situations where the computation time is very crucial. As the Chebyshev distance considers the largest difference in any dimension, it is also very sensitive to the scale of the variables.

Example 3.4.4. The distance between Object 1 and Object 2 is calculated as:

$$d_{1,2} = max(|10-11|, |5-6|, |8-9|, |2-1|) = max(1,1,1,1) = 1$$

The distance between Object 1 and Object 3 is calculated as:

$$d_{1,3} = max(|10-1|, |5-20|, |8-0|, |2-8|) = max(9,15,8,6) = 15$$

3.4.5 Canberra Distance

The *Canberra distance* is the summation of the series of fractional differences between coordinates of two data points (x_i and x_j). The Canberra distance is defined as Equation 3.26.

$$d_{x_i,x_j} = \sum_{k=1}^{n} \frac{|x_{ik} - x_{jk}|}{|x_{ik} + x_{jk}|} \tag{3.26}$$

The numerator of this equation signifies the difference between the objects, whereas the denominator normalizes the difference. Thus, the distance for each dimension may at most be 1. Therefore, in contrast to the distance measures such as, Euclidean, Manhattan, Minkowski, and Chebyshev, it is not scale sensitive. The Canberra distance is suitable for non-negative values. One of the drawbacks of the Canberra distance measure, is that it is very sensitive to changes near the origin. In a special case, it is possible that both coordinates may take the value 0 and will create a situation when both the numerator and denominator are 0. Thus, the distance will return a value which is undefined.

Example 3.4.5. The distance between Object 1 and Object 2 is calculated as:

$$d_{1,2} = \frac{|10-11|}{|10+11|} + \frac{|5-6|}{|5+6|} + \frac{|8-9|}{|8+9|} + \frac{|2-1|}{|2+1|} = 0.5307$$

The distance between Object 1 and Object 3 is calculated as:

$$d_{1,3} = \frac{|10-1|}{|10+1|} + \frac{|5-20|}{|5+20|} + \frac{|8-0|}{|8+0|} + \frac{|2-8|}{|2+8|} = 3.0182$$

3.4.6 Mahalanobis Distance

The *Mahalanobis distance* [74], considers the correlation between variables. The Mahalanobis distance measure uses the *covariance matrix* to measure the variance and the correlation between the objects. Let $\vec{x}$ and $\vec{y}$ be two vectors and C^{-1} be the inverse covariance matrix. Then the Mahalanobis distance is calculated as:

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})C^{-1}(\vec{x} - \vec{y})^T} \tag{3.27}$$

The main difference between the distance measures discussed so far and the Mahalanobis distance measure is that it considers the correlation between the variables. Moreover, the distance measure is not scale-dependent. When applied to a cluster analysis problem, the distance measure favors the clusters of hyper-ellipsoidal shape [7], [80].

However, one of the drawbacks of using the Mahalanobis distance measure is its high computational cost, which is due to the calculation required to construct the inverse covariance matrix. The Mahalanobis distance may not be suitable for high-dimensional datasets as covariance estimation may be inaccurate [26].

Example 3.4.6. The covariance matrix (C^{-1}) for the data in Table 3.5 is (as calculated by MATLAB):

$$\begin{pmatrix} 30.3333 & -45.6667 & 27.1667 & -20.8333 \\ -45.6667 & 70.3333 & -40.8333 & 31.1667 \\ 27.1667 & -40.8333 & 24.3333 & -18.6667 \\ -20.8333 & 31.1667 & -18.6667 & 14.3333 \end{pmatrix}$$

The distance between Object 1 and Object 2 is calculated as:

$$(\vec{x} - \vec{y}) = \begin{bmatrix} -1 & -1 & -1 & 1 \end{bmatrix}$$

$$d_{1,2} = \sqrt{\begin{bmatrix} -1 & -1 & -1 & 1 \end{bmatrix} * C^{-1} * \begin{bmatrix} -1 & -1 & -1 & 1 \end{bmatrix}^T} = 1.9828$$

The distance between Object 1 and Object 3 is calculated as:

$$(\vec{x} - \vec{y}) = \begin{bmatrix} 9 & -15 & 8 & -6 \end{bmatrix}$$

$$d_{1,3} = \sqrt{\begin{bmatrix} 9 & -15 & 8 & -6 \end{bmatrix} * C^{-1} * \begin{bmatrix} 9 & -15 & 8 & -6 \end{bmatrix}^T} = 3.8609$$

3.4.7 Angular Distance

The *Angular Separation* or *Cosine Distance* [64], measures the angular distance between the coordinates of two data points (x_i and x_j). Even though, this measure is called the Angular distance, it is a similarity measure rather than a distance measure. It represents the cosine angle between the unit vectors in the direction of the two pattern vectors [69] and thus the value lies between 1 and -1 as of the range of cosine angle. Though the angles are measured, it is meant to give the linear distance between the data points. A higher value of this function denotes that the data objects are very similar to one another. The similarity and distance measure between object x_i and x_j is given below:

$$s_{x_i,x_j} = \frac{\sum_{k=1}^{n} x_{ik} \cdot x_{jk}}{\left(\sum_{k=1}^{n} x_{ik}^2 \cdot \sum_{k=1}^{n} x_{jk}^2\right)^{\frac{1}{2}}} \tag{3.28}$$

$$d_{x_i,x_j} = 1 - \frac{\sum_{k=1}^{n} x_{ik} \cdot x_{jk}}{\left(\sum_{k=1}^{n} x_{ik}^2 \cdot \sum_{k=1}^{n} x_{jk}^2\right)^{\frac{1}{2}}} \tag{3.29}$$

The Angular distance measure is widely applied to Text Document cluster analysis, where the data is usually highly dimensional. One of the reasons for its popularity in document cluster analysis is that the Angular distance does not depend on the length: $s_{x_i,x_j} = s_{\alpha x_i,x_j}, \alpha > 0$ [26]. Therefore, documents with similar term frequency proportion but different totals are treated as similar. In addition to this property, the Angular distance is also scale invariant, and thus, the different units do not affect the result. The Angular distance considers the relative distance between the objects from a fixed point (the origin).

Example 3.4.7. The distance between Object 1 and Object 2 is calculated as:

$$d_{1,2} = 1 - \frac{(10*11)+(5*6)+(8*9)+(2*1)}{\sqrt{(10^2+5^2+8^2+2^2)*(11^2+6^2+9^2+1^2)}} = 0.0036$$

The distance between Object 1 and Object 3 is calculated as:

$$d_{1,3} = 1 - \frac{(10*1)+(5*20)+(8*0)+(2*8)}{\sqrt{(10^2+5^2+8^2+2^2)*(1^2+20^2+0^2+8^2)}} = 0.5794$$

3.4.8 Pearson Correlation Distance

The *Pearson correlation coefficient* [64] measures similarity between data points. The values of this function ranges from +1 to -1. Since this measurement shows whether two data points are linearly related or not, a value of 1 shows that the points are lying on the same line and are positively correlated. A value of -1 indicates that the points are negatively correlated, whereas 0 means there is no linear correlation between the data points.

$$s_{ij} = \frac{\sum_{k=1}^{n}(x_{ik}-\bar{x}_i)\cdot(x_{jk}-\bar{x}_j)}{\left(\sum_{k=1}^{n}(x_{ik}-\bar{x}_i)^2\cdot\sum_{k=1}^{n}(x_{jk}-\bar{x}_j)^2\right)^{\frac{1}{2}}} \tag{3.30}$$

where, $\bar{x}_i = \frac{1}{n}\sum_{k=1}^{n} x_{ik}$ and $\bar{x}_j = \frac{1}{n}\sum_{k=1}^{n} x_{jk}$. The similarity function may be changed to correlation distance measure by subtracting from 1.

$$d_{ij} = 1 - \frac{\sum_{k=1}^{n}(x_{ik}-\bar{x}_i)\cdot(x_{jk}-\bar{x}_j)}{\left(\sum_{k=1}^{n}(x_{ik}-\bar{x}_i)^2\cdot\sum_{k=1}^{n}(x_{jk}-\bar{x}_j)^2\right)^{\frac{1}{2}}} \tag{3.31}$$

The Pearson correlation coefficient is scale invariant. Therefore, two curves with identical shape but different magnitude will have similar similarity. The *Pearson Correlation*

Coefficient takes the mean of the variables into consideration, and thus, the measure gives the similarity or distance in terms of the mean of the variables. This coefficient is also similar to the *Angular Distance* where the mean is zero. The Pearson coefficient correlation is used in the areas of microarray analysis and the document cluster analysis, amongst others. Since this distance measure considers the correlation between the objects, the outliers may affect the end results.

Example 3.4.8. $\bar{x_1} = \frac{10+5+8+2}{4} = 6.25$, $\bar{x_2} = \frac{11+6+9+1}{4} = 6.75$ and $\bar{x_3} = \frac{1+20+0+8}{4} = 7.25$
The distance between Object 1 and Object 2 is calculated as:

$$d_{1,2} = 1 - \frac{[(10-6.25)*(11-6.75)]+[(5-6.25)*(6-6.75)]+[(8-6.25)*(9-6.75)]+[(2-6.25)*(1-6.75)]}{\sqrt{[(10-6.25)^2+(5-6.25)^2+(8-6.25)^2+(2-6.25)^2]*[(11-6.75)^2+(6-6.75)^2+(9-6.75)^2+(1-6.75)^2]}}$$
$$= 1 - \frac{45.2498}{45.6680} = .0092$$

The distance between Object 1 and Object 3 is calculated as:

$$d_{1,3} = 1 - \frac{[(10-6.25)*(1-7.25)]+[(5-6.25)*(20-7.25)]+[(8-6.25)*(0-7.25)]+[(2-6.25)*(8-7.25)]}{\sqrt{[(10-6.25)^2+(5-6.25)^2+(8-6.25)^2+(2-6.25)^2]*[(1-7.25)^2+(20-7.25)^2+(0-7.25)^2+(8-7.25)^2]}}$$
$$= 1 - \frac{-55.25}{96.7586} = 1.57$$

Table 3.6 contains the results from each of the distance measures when the distance is calculated in between the objects in Table 3.5. The distance between Object 1 and Object 2 is less than the distance between Object 1 and Object 3 in all cases. The values in the table shows that each distance measure gives a different value for these two pairs, as the way they calculate the distance is different from one another. Table 3.7 provides a summary of each of the distance measures based on the discussion provided in this section.

Distance Measure	Distance between Object 1 and Object 2	Distance between Object 1 and Object 3
Euclidean Distance	2	20.1494
Manhattan Distance	4	38
Minkowski Distance	1.5874	16.9061
Chebyshev Distance	1	15
Canberra Distance	0.5307	3.0182
Mahalanobis Distance	1.9828	3.8609
Angular Distance	0.0036	0.5794
Pearson Coefficient Distance	0.0092	1.571

Table 3.6: Sample result from different distance measures (numeric variables).

Distance Measure	Comments	Limitations
Euclidean Distance	Works well for compact or isolated clusters; Discovers clusters of spherical shape; Any two objects may not be influenced by the addition of a new object (i.e. outliers).	Very sensitive to the scales of the variables; Not suitable for clusters of different shapes; The variables with the largest values may always dominate the distance.
Manhattan Distance	Computationally cheaper than the Euclidean distance.	Scale dependent.
Minkowski Distance	One may control the amount of emphasis given on the larger differences.	The Minkowski distance may cost more than the Euclidean and Manhattan distance when $\lambda > 2$.
Chebyshev Distance	Suitable for situations where the computation time is very crucial.	Very sensitive to the scale of the variables.
Canberra Distance	Not scale sensitive; Suitable for non-negative values.	Very sensitive to the changes near the origin; Undefined when both the coordinates are 0.
Mahalanobis Distance	Considers the correlation between the variables; Not scale-dependent; Favors the clusters of hyper ellipsoidal shape.	Computational cost is high; May not be suitable for high-dimensional datasets.
Angular Separation	Calculates the relative distance between the objects from the origin; Suitable for semi-structured datasets (i.e. Widely applied in Text Document cluster analysis where data is highly dimensional); Does not depend on the vector length; Scale invariant.	Absolute distance between the data objects is not captured.
Pearson Correlation Coefficient	Scale invariant; Considers the correlation between the variables; Calculates the relative distance between the objects from the mean of the data; Suitable for semi-structured data analysis (i.e. in microarray analysis, document cluster analysis).	Outliers may affect the results.

Table 3.7: Summary of distance measures for the numeric data type.

3.5 Chapter Summary

Similarity, dissimilarity, and distance are the three fundamental terms directly related to the concept of cluster analysis. As such, the measures that define and calculate similarity, dissimilarity, and distance often need special consideration. This chapter provided an introduction to each of the three terms. We discussed the proximity measures that are particularly suitable for datasets with numeric, binary, and mixed attributes. There are several different measures available for each of the data types. We noticed that several proximity measures for the numeric variables are very sensitive to the scale of the attribute. A number of these measures also consider the correlation between the variables into account. For the binary datasets, the similarity measures differ mainly on the amount of weight given to positive matches, negative matches, and disagreements. On the other hand, similarity and dissimilarity coefficients for the mixed variable combine measures from various data types into a single equation.

In the next chapter, we discuss the experimental approach observed in this study. That is, we present the techniques and methods used in this work to compare the performance of the proximity measures presented in this chapter.

Chapter 4

Experimental Approach

In this thesis, a comparative study of several proximity measures is performed against the spectral clustering algorithm. Therefore, in Chapter 2, we discussed two variants of spectral clustering algorithms and in Chapter 3; we presented the proximity measures suitable for three types of attributes (i.e. binary, numeric, and mixed). This chapter presents the experimental approach that we followed. The chapter begins with a discussion regarding the preparation of data and the construction of the similarity matrix in Section 4.1. Section 4.2 presents the methods that are applied to address several algorithm-specific issues. This is followed by Section 4.3, where we present the procedures that are used to ensure the accuracy of the experiments, such as the cross-validation technique. Recall from Chapter 2 that, once the selection is made and the algorithm is performed on a given dataset, the next fundamental step is the validation or assessment of the results obtained from the cluster analysis method. Thus, Section 4.4 is dedicated to the cluster evaluation methods used in this study. We conclude our discussion with a brief summary in Section 4.6.

4.1 Data Preparation

As mentioned in Chapter 2, the spectral clustering algorithms take the similarity matrix as input (Step 1 in Figure 4.1). Before the data is converted to a similarity matrix, it is necessary to process the data to ensure that the data is compatible with the distance measures and that they do not contain any missing values. Therefore, to ensure the accuracy and usefulness of the results, it is important to pre-process the data [16]. In this section, we present the methods adopted in this study to prepare the data for our

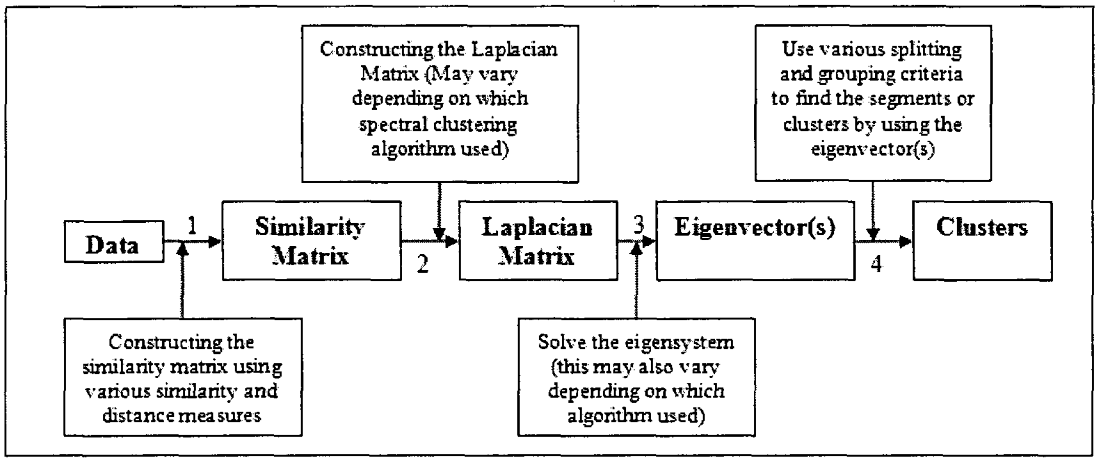

Constructing the Laplacian Matrix (May vary depending on which spectral clustering algorithm used)
Use various splitting and grouping criteria to find the segments or clusters by using the eigenvector(s)
Data
1
Similarity Matrix
2
Laplacian Matrix
3
Eigenvector(s)
4
Clusters
Constructing the similarity matrix using various similarity and distance measures
Solve the eigensystem (this may also vary depending on which algorithm used)

Figure 4.1: Steps of spectral clustering algorithm.

experiments.

4.1.1 Preprocessing

We used several datasets in this study. A number of these datasets contained missing values. According to Witten et al. [76], the missing values may be replaced by the mean or the mode of the attribute values depending on the attribute type. For instance, the numeric missing values were replaced by the mean, whereas the binary and nominal missing values were replaced by the mode of the attribute values.

Recall from Chapter 3 that several distance measures (i.e. Euclidean, Manhattan, and Minkowski) are scale-dependent for the numeric variable type. Therefore, the numeric datasets that contain attributes with different scales need to be preprocessed. Data is standardized by converting the original measurements to unitless variables [32]. The procedure is as follows [32].

Standardization

Given the measurements for a variable f for a dataset, we first calculate the mean absolute deviation, s_f:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + ... + |x_{nf} - mf|) \tag{4.1}$$

where, $x_{1f}, ..., x_{nf}$ are n measurements of variable f and m_f is the mean value of f such that $m_f = \frac{1}{n}(x_{1f} + x_{2f} + ... + x_{nf})$. Next, we calculate the standardized measurement

z_{if}.

$$z_{if} = \frac{x_{if} - m_f}{s_f} \quad (4.2)$$

4.1.2 Constructing the Similarity Matrix

Recall from Chapter 2, that the first step of the spectral clustering algorithm is the construction of the similarity matrix (Step 1 in Figure 4.1). A similarity matrix is a n x n symmetric non-negative matrix, where n is the number of objects in a given dataset. Let, i and j be any two objects in a given dataset, located at row i and row j, respectively. If the similarity (i.e. calculated from a similarity measure) between these two objects is S_{ij}, then it will be located at the cell at row i and column j in the similarity matrix. Since, the similarity matrix is symmetric, the entry at row i and column j will be the same as the entry at row j and column i. As mentioned in Chapter 3, the similarity matrix is built from the proximity measure that depends on the type of the attributes in a dataset. There are several choices to be made regarding the similarity and the distance measures. For instance, a number of different user-defined scaling parameters may need to be set during the experimental stage. Therefore, construction of the similarity matrices for the data types (numeric, binary, and mixed) considered in this study, vary from one another. Below we discuss the procedures for creating the similarity matrix for each of the data types.

Similarity Matrix for Binary Data

Constructing the similarity matrix for ***binary data*** is less complicated than any other data types. Most of the coefficients for the binary data as discussed in Chapter 3, directly calculate either the similarity or the dissimilarity between the object pairs. Therefore, no external or secondary function (Gaussian function for numeric data) is necessary. However, if the coefficient gives the dissimilarity rather than the similarity measure, we need to convert the coefficient into a similarity measure. We get the similarity between the two objects by subtracting the dissimilarity value from 1. The values from the similarity measures for the binary data type ranges from 0 to 1. Therefore, we do not need to be concerned about the cases when the similarity value is between -1 and $+1$. Once the similarity between the objects is calculated, these values become the entries of the similarity matrix. In Chapter 3, we defined six similarity measures for the binary data. Recall that the similarity coefficients are: 1. *Czekanowski (CZE)*, 2. *Jaccard (JAC)*, 3. *Sokal and Sneath (SAS)*, 4. *Simple Matching Coefficient (SIM)*, 5. *Russell and Rao*

(RAR) and 6) *Rogers and Tanimoto (RAT).*

Similarity Matrix for Numeric Data

Constructing the similarity matrix for ***numeric data*** is more complicated than for binary data. There are a number of issues related to computing the similarity between the objects of numeric attribute type. First, most of the measures that calculate the proximity between the numeric objects are distance measures. In contrast, the spectral clustering algorithms take the similarity matrix as an input, rather than the distance matrix. Therefore, we cannot use them directly to construct the similarity matrix. They need to be converted to a similarity measure before they may be used. For this reason, an external or secondary function that converts the distance measure to the similarity measure is necessary. A number of functions that convert the distance measures into the similarity measures are discussed in Chapter 3. In this work, we use the function given by Equation 3.2. Recall that the function is commonly known as the *Gaussian* function.

$$s(x, y) = \exp(\frac{-d(x, y)^2}{2 * \sigma^2}) \tag{4.3}$$

In the above equation,

$s(x, y)$ = similarity between points x and y

$d(x, y)$ = distance between points x and y

σ = a user specified scaling variable

Regarding the selection of a function that calculates the similarity between the objects for numeric data, Fischer and Poland [22] said, *"any symmetrical, non-negative function monotonously falling with increasing distance can be applied"*. However, they also note that using the function given in Equation 4.3 is preferable because it simplifies the computation of the eigenvalues. It may be clarified as follows. Recall that $d(x, y) \geq 0$, according to the properties of distance measures discussed in Chapter 3. The value of σ is also greater than 0, since, if $\sigma = 0$ then $\frac{-d(x,y)^2}{2*\sigma^2}$ becomes undefined as the denominator becomes 0. Moreover, σ cannot be negative because of the way a value for σ is selected, which we will discuss shortly. Therefore, it directly follows from the above consequences that the term $\frac{d(x,y)^2}{2*\sigma^2} \geq 0$ in Equation 4.3 and from this we can deduce that $\frac{-d(x,y)^2}{2*\sigma^2} \leq 0$ (true for all the values of $d(x, y)$ and σ). Next, let $y = e^x$ be an exponential function where x is any real number, then according to the properties of exponential function *exp* [73]:

- y is always positive and lies above the x-axis.
- The function never touches the x-axis, even though it passes very close to the x-axis. Therefore, y cannot have a value of 0.
- The function climbs quickly for the positive values of x.
- The function climbs slowly for the negative values of x.
- When the negative value of x is very small, the function will have a value very close to 0. These properties may be depicted through Figure 4.2.

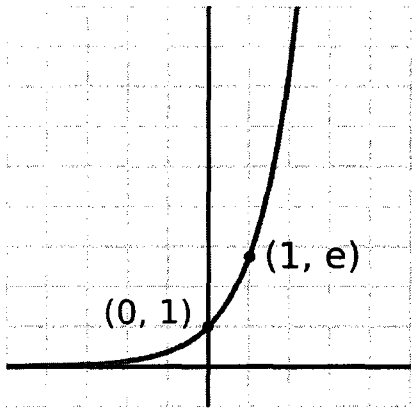


Figure 4.2: The exponential function as given in [73]

Therefore, according to the properties of exponential function, when the distance between two objects i and j is large (the objects are very different from one another), the Gaussian function will return a very small value (a value very close to zero). This is depicted in Figure 4.2. Since the Gaussian function is the similarity function for numeric data, the returned value will represent the similarity between the objects i and j. Thus, when the distance is large between any two objects, the similarity will be low, and vise versa. An example is presented in Table 4.1. In this example, to show the difference, we assigned two values to $d(i,j)$. The first column contains the value as distance and the last column in the table gives the similarity as calculated by the Gaussian function. The first distance value (2) being comparatively larger than the second value (0.25), has a smaller similarity value ($1.9287 * 10^{(-22)}$, very close to 0). Since, the Gaussian function never takes the value zero and is always positive, the similarity matrix constructed for a dataset using this function always has positive entries, irrespective of the distance value. According to

$d(x,y)$	$d(x,y)^2$	σ	σ^2	$\frac{d(x,y)^2}{2*\sigma^2}$	$s(i,j)\exp(\frac{-d(x,y)^2}{2*\sigma^2})$
2	4	0.2	0.04	50	$1.9287 * 10^-22$
0.25	0.0625	0.2	0.04	0.7813	0.4578

Table 4.1: Example showing the behaviour of the Gaussian function with small and large distance value.

[22], when this matrix is used in a spectral clustering algorithm, it simplifies the analysis and computation of the eigenvalues. Shi and Malik in [58], also compared a number of functions (as presented in Chapter 3) that convert a distance measure into a similarity measure and found that the Gaussian function outperforms most of the time. In their work, they showed that the Gaussian function has the fastest decreasing rate and only the objects that are located close to one another will have a significantly larger value. We use this Gaussian function in all the experiments for the datasets of the numeric variable type.

However, there is a drawback associated with the Gaussian function. The function is very sensitive to the choice of σ [46]. The significance of σ in Equation 4.3, is that it controls the portion of area that serves as the neighborhood of the points in a dataset. If the value of σ is large then more points are included in the neighborhood, which results in a more spread-out area. Therefore, a large σ may merge the natural clusters [23]. In contrast, a small value for σ considers a smaller area, or neighborhood of points. However, a smaller value for σ may also present problems. If the σ is too small, it may neglect some points in the neighborhood, which may produce an incorrect clustering result. This is shown in the Figure 4.3 where the area of the neighborhood increases with the value of σ. In this figure, we plot the similarity matrices after applying the normalized cut spectral clustering algorithm on a sample dataset, keeping the distance measure constant for all four cases. Therefore, it is necessary to choose a σ where the similarity of closely connected points is high [23]. Initially the value is set to 10 to 20 percent of the total range of the values obtained from the distance function $d(x,y)$ as suggested by Shi and Malik [58]. However, to ensure that we take the correct value that gives the best result for σ, we consider a range of values for σ in addition to the values obtained from the method suggested by the authors.

In Chapter 3, we discussed the distance measures considered in this study for the datasets with numeric variables. Recall that these measures are, 1. *Euclidean Distance (EUC)*, 2. *Manhattan Distance (MAN)*, 3. *Minkowski Distance (MIN)*, 4. *Canberra*

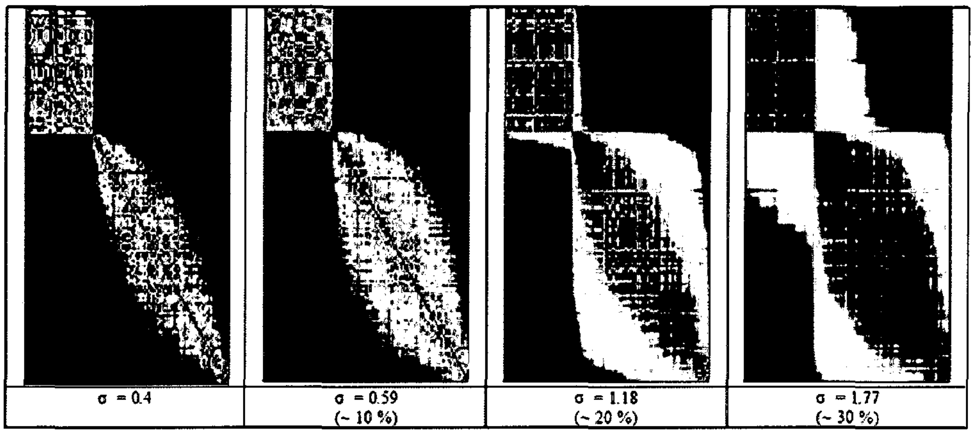


Figure 4.3: Similarity matrices using various values of σ

Distance (CAN), 5. *Mahalanobis Distance (MAH)*, 6. *Chebyshev Distance (CHEB)*, 7. *Angular Distance (COS)*, and 8. *Pearson Correlation Distance (COR)*. It is important to note that, for each distance measure, there may be a different set of values of σ. This is because, different distance measures give different distance values for the same pair of objects, which in turn depends on how the distance function is defined. Furthermore, the value of σ depends on the values of these distance measures. Therefore, it is necessary to consider different sets of values of σ for the various distance measures.

Similarity Matrix for Mixed Data

To construct the similarity matrix for ***mixed data***, we calculate the similarity directly from the data as we did for the binary data. Therefore, we do not need any secondary function or any user defined parameter (i.e. σ for numeric data). As mentioned in Chapter 1, the most common attribute types are numeric, nominal, and binary [39]. As such, the datasets considered in this study contain attributes from these three types. Compared to the number of proximity measures for the numeric and binary attribute types, there are not many similarity measures proposed for the mixed data types. In this work we used two measures as defined in Chapter 3, namely, 1. *Gower's General Dissimilarity Coefficient (GOWER)* and 2. *Laflin's General Coefficient (LAFLIN)*. Recall that the *GOWER* coefficient is a dissimilarity measure that ranges from 0 to 1. Thus, we subtracted this value from 1 to get the similarity score using the Equation 3.1 defined in Chapter 3. On the other hand, *LAFLIN* gives the similarity value rather than the distance and we do not need any additional steps for this measure. However, as the definition suggests, we need a distance measure to compute the distance between the numeric objects in a mixed dataset. We used the *Euclidean Distance* measure to calculate this distance, which is a widely used distance measure in spectral clustering algorithms (defined in Equation 3.3). Since this coefficient only considers the similarity between the objects, rather than the distance, we used Equation 3.4 to convert the Euclidean distance to the similarity score. Recall that the definitions for each of these proximity measures are presented in Chapter 3 (Section 3.3).

4.2 Algorithm-specific Issues

Once the similarity matrix is created, the spectral clustering algorithm is applied to this matrix. In Chapter 2, we discussed the steps for the spectral method. In this section, we address several additional issues related to the algorithms, such as the stopping and splitting criteria that are applied in this study.

4.2.1 Stopping Criteria for SM(NCut) Algorithm

In Chapter 2, we discussed the *SM(NCut)* spectral clustering algorithm. Recall that the *SM(NCut)* algorithm is a recursive algorithm that bi-partitions the data at each level. Therefore, the result of this algorithm when applied to a given dataset forms a hierarchical tree of clusters. Since, the algorithm recursively continues until a stopping criterion is satisfied, the authors of [58] suggested a method to terminate partitioning when a stability criterion is satisfied. The stability is measured by taking the ratio of the minimum and maximum bin sizes. This is done by drawing the histogram of the components of the eigenvector and then computing the ratio of the minimum and maximum bin sizes. According to [58], if the values of the eigenvector change frequently, the histogram bins will stay relatively the same and thus the ratio will be high. In contrast, when the values of the eigenvector do not change much, the histogram bins will vary and the ratio will be small. In such cases, when stability is smaller than a certain threshold, the algorithm would recursively repartition the segments. Another method that we used in this work keeps track of the size of each cluster. When the size reaches a certain user specified threshold, the algorithm stops repartitioning that cluster. This is particularly helpful, since we are able to control the minimum size of the clusters acceptable by the algorithm. However, it is important to note that, since we are comparing the performance of various similarity measures, it is essential to keep the stopping criteria constant for a given dataset. By doing so, we ensure that all the parameters and settings are the same for a particular dataset and the only place they vary is in the similarity matrices itself.

4.2.2 Splitting Criteria and Grouping Algorithm

Recall from Chapter 2 that, in spectral clustering, clusters are generated from the eigenvectors of the Laplacian matrix, which is a modified version of the similarity matrix. The eigenvectors are split at different positions to find the clusters (step 4 of Figure 4.1). Therefore, the choice of splitting method may also influence the results as it partitions the eigenvectors at different positions. Shi and Malik [58], suggested a number of splitting methods in their work. In this study, we performed an evaluation on these splitting points to compare their performance. We discuss the splitting methods proposed in [58] as well as another method as proposed in [66].

Let the eigenvector associated with the second smallest eigenvalue be e. Assume that there are $i = 1...n$ objects in the dataset which was initially provided as input for constructing the similarity matrix. Then the eigenvector $e = e_1, e_2....e_n$ will have exactly n elements, where element 1 in e corresponds to the object 1 in the dataset, element 2 corresponds to the object 2, and so on. Now, the main idea behind finding an optimal cut at c (splitting at c) is to split e in such a way so that all the components $e_i < c$ ($i \in [1...n]$) are in one partition and the elements where $e_i \geq c$ are in a different group. Then,

- **Split at Zero:** In this case, $c = 0$. Therefore, we will have one partition that contains the components with negative values of e ($e_i < 0$) and another partition with positive elements of e ($e_i \geq 0$).

- **Split at Mean:** This method uses the mean value of eigenvector e (average of $e_1..e_n$) as the splitting criteria. The elements in e with values greater or equal to the mean value will be in one group and the rest will be in a different group.

- **Split at Median:** The splitting point c is located at the median of $e_1..e_n$. In this case, the elements in e are sorted in ascending order and the element located at the middle is used as c, separating the elements in the higher half from the lower half. Thus, it gives two partitions of almost the same sizes.

- **Split using normalized cut method 1 (NCut1):** This method optimizes the normalized cut [58] (as elaborated in Chapter 2). There are two methods proposed in two different studies. One such method as discussed in [66] performs a search over $n-1$ partitions to find the best one. Recall that this is the value that minimizes the NCut as defined by the Equation 2.4 in Chapter 2. This partitioning algorithm works as follows:

  - Sort the eigenvector associated with the second smallest eigenvalue in ascending order.
  - For $i = 1..n - 1$ (here i is the index of sorted eigenvector)
    Partition A = $1...i$
    Partition B = $i + 1...n$
    Compute NCut(A,B);
  - Return the partition that gives the minimum value for NCut.

- **Split using normalized cut method 2 (NCut2):** Another method for finding the minimum value for NCut from the sorted eigenvector is given in [58] and [49]. Unlike the previous NCut based splitting method, this method considers m evenly spacing splitting points and computes the minimum NCut among them. This partitioning method works as follows:

  - Sort the eigenvector associated with the second smallest eigenvalue in ascending order.
  - Find m uniform values between the maximum and minimum value of eigenvector e. Here, $m = \lceil \log_2(n) + 1 \rceil$. This will return m points from the eigenvector of equal interval. Let the points be denoted as $e_{1m}, e_{2m}...e_{mm}$.
  - For $j = 1...m$
    Partition A = $e_i < e_{jm}$
    Partition B = $e_i \geq e_{jm}$
    Compute NCut(A,B);
  - Return the partition that gives minimum value for NCut.

Since the splitting methods based on the concept of normalized cut perform a search over the eigenvector space, it is comparatively slower and more time consuming than the splitting methods based on zero, mean, and median. The spectral clustering algorithm using the K-means algorithm does not need any splitting criteria, as it uses the K-means algorithm to directly cluster the objects in eigenvectors into k groups.

As of now we discussed some of the algorithm-specific issues related to the spectral clustering algorithms. In the next section we will discuss the experimental method.

4.3 Experimental Methodology

This section explains the experimental details that are not directly related to the algorithms. This includes methods, such as cross-validation, which is often applied to ensure the performance of the algorithms.

4.3.1 Cross-Validation

In order to evaluate the performance of the distance measures, we need to perform the experiments on several datasets. We apply a ten-fold cross validation method, similar to the approach proposed in [12], for cluster analysis. In this method, data is first divided into ten folds. For each iteration, one set is used as the test set and the rest of the nine sets are used as training sets. The cluster analysis algorithm is then applied on the training set. Next, we calculate the cluster centroid for each of the clusters obtained from the result. Then, for each object in the test set, we calculate the proximity between the centroids of each cluster and the object. The object is assigned to the cluster with the nearest centroid. Next, with the help of the external class labels, we compute the external evaluation measures as discussed in the next section. It is important to note that the external class labels are not used prior to this step. They are stored in a different file and are not used when the cluster analysis algorithm is applied to the training set. Therefore, they are only used for the purpose of evaluation. This method is similar to the *classes to clusters* evaluation method [76], where we ignore the class attribute first and then use them later to compute the evaluation measures. Once the evaluation measures are computed, the entire process is repeated ten times, each time with a new test set. The results from the evaluation measures are then averaged over the ten folds to get the final result. When the dataset is subdivided into folds, it is important to stratify the folds, so that each fold has approximately similar distribution of the class labels as the original dataset, to ensure accuracy. We also restore the original index numbers of the similarity matrix to correctly identify each object. Since we partition the eigenvector(s) to find the clusters, if the indexes are not restored then once the eigenvectors are sorted, there will be no way to determine the cluster membership. However, the ordering of rows is unimportant at the time of building the similarity matrix. Once the similarity matrix is created, the ordering information must be restored in order to correctly map the objects to their corresponding components in the eigenvector(s).

4.3.2 Experiments

As mentioned, we considered several distance measures for each of the three data types. In order to compare the performance of the proximity measures for a particular data type, we performed ten-fold cross validation and classes to clusters evaluation on each of the datasets. Therefore, it is important that we divide a dataset into ten folds first and then apply the cross validation and classes to clusters evaluation for each of the proximity and spectral clustering algorithm combinations. For instance, we have six similarity coefficients for the binary attribute types and we have two spectral clustering algorithms for each experiment. Then, we first subdivide the dataset into ten sets and for a similarity and algorithm combination; we perform the cross validation and classes to clusters evaluation. For the next similarity coefficient, we use the same folds and repeat the procedure. In that way, we ensure that, for each algorithm, the only reasons the results may vary, is because of the choice of similarity coefficient.

4.3.3 Implementation and Settings

For all the experiments in this study, the data preprocessing was performed using WEKA [76], an open-source Java-based machine learning software developed at the University of Waikato in New Zealand. The spectral cluster analysis algorithms are implemented in MATLAB ®. Since the spectral clustering algorithm manipulates the similarity matrix of a dataset, computation of eigenvalues and eigenvectors of the similarity matrix may be inefficient for a large matrix. However, MATLAB has a function called *eigs* that efficiently solves the eigensystem of large matrices. We used this function in this study to find the eigenvalues and eigenvectors. The cluster evaluation measures have been implemented in Java.

4.4 Cluster Evaluation Measures

In Chapter 2, we presented the four fundamental components of the cluster analysis process. The first two steps deal with the preprocessing of data, the selection of the proximity measure, and the selection of the clustering algorithm. Apart from the selection of measure and the execution of the cluster analysis algorithm, the next fundamental step in a given cluster analysis process is the verification or evaluation of the results to assess the quality of the clusters. In this section we present the evaluation measures that are used in this study.

Cluster evaluation methods measure the *goodness* or the quality of the clusters obtained from a cluster analysis method. In this study, we applied the external quality measures. The external evaluation measures may be calculated when the true class labels are known to the user. In this study, we consider the external measures when evaluating our results, as the external class labels for each of the datasets used were available to us. Moreover, this allows us to perform a fair comparison against the known *true clusters* for all the proximity measures. We present the three external methods that we used for the evaluation purpose in this thesis. These measures have been previously used in numerous studies regarding cluster analysis and have proved successful in representing the quality of clusters numerically. For instance, the F-measure is widely used to evaluate the quality of clusters when hierarchical algorithms are applied [6].

We define a number of terms here before presenting the evaluation measures in the section. Let C be the set of external class labels available to users and D be the set of clusters obtained after an algorithm is applied on a given dataset. Then according to [61], $|C_i \cap D_j|$ denotes the number of members of external class i (C_i) that are also in cluster j (D_j), $|C_i|$ denotes the number of members that are in external class C_i according to C, $|D_j|$ denote the number of members that are in cluster D_j according to D, and $n = |C|$ denotes the total number of objects present in the dataset. The cluster evaluation methods such as *F-measure*, *G-means*, and *Entropy* are defined as follows.

4.4.1 F-measure

F-measure [52], [61], [10] is one of the external cluster evaluation methods that are widely used when the class information is known. The F-measure incorporates *precision* and *recall* from the information theory into a single number. The precision denotes the proportion of correctly clustered objects out of all the objects in a cluster D_j. The *precision* for a cluster j (D_j) and external class i (C_i) is defined as:

$$P_{i,j} = \frac{|C_i \cap D_j|}{|D_j|} \tag{4.4}$$

In contrast, the recall denotes the proportion of correctly clustered objects out of all the objects in an external class C_i. The *recall* for cluster j (D_j) and external class i (C_i) is defined as:

$$R_{i,j} = \frac{|C_i \cap D_j|}{|C_j|} \tag{4.5}$$

Then, F-measure combines both the precision and recall together. The F-measure for cluster j (D_j) and external class i (C_i) is defined as:

$$F_{i,j} = \frac{2 * P_{i,j} * R_{i,j}}{P_{i,j} + R_{i,j}} \tag{4.6}$$

For each external class i, the F-measure is the maximum value attained at any cluster:

$$F_i = max_j F_{i,j} \tag{4.7}$$

Therefore, the *F-measure* for an entire solution is defined as:

$$F = \sum_i \frac{|C_i|}{|C|} . F_i \tag{4.8}$$

The range for F-measure is [0,1]. A higher value for F-measure implies a better solution, whereas a lower value implies that the results from cluster analysis do not match with the external class information. For hierarchical cluster analysis, each node in the hierarchical tree is considered as a cluster and each cluster is treated as if it were the result of a query. For an entire hierarchical clustering, the F-measure of any class is the maximum value it attains at any node in the tree, and the overall value for the F-measure is then calculated by taking the weighted average of all values for the F-measure as given above [61].

4.4.2 G-means

G-means is defined as the geometric mean of *recall* and *precision.*

$$G - means = \sqrt{recall * precision} \tag{4.9}$$

As the equation above suggests, G-means will have a high value when precision and recall are both high. G-means may be used in a similar way as F-measure described earlier [17]. For a more detail introduction of G-means we suggest [41].

4.4.3 Entropy

Entropy [10], [61] is another cluster evaluation method used to measure the quality of clusters. This measure considers the distribution of classes in a cluster to assess the results. For each cluster D_j, the Entropy is calculated as:

$$E(D_j) = \sum_{i=1}^{k} - \left(\frac{|C_i \cap D_j|}{|D_j|}\right) \log \left(\frac{|C_i \cap D_j|}{|D_j|}\right) \tag{4.10}$$

Here, the *sum* is taken over all classes and *log* is *log* base 2. The total Entropy for a set of clusters is calculated as the sum of entropies of each cluster weighted by the size of each cluster:

$$E = \sum_{j=1}^{k} E(D_j) \cdot \frac{|D_j|}{|C|} \tag{4.11}$$

The Entropy of a cluster is a measure of disorder within the cluster and a lower value for Entropy indicates a better clustering result. Thus, a value close to 0 denotes that the cluster mostly includes objects from one class.

There are, however, several drawbacks of using the Entropy measure. The measure gives the best score when a cluster contains a single object. Therefore, this measure is biased to favor a large number of clusters [26]. Prior research on several hierarchical clustering algorithms also suggest that the Entropy may not be a suitable measure where the algorithms produce clusters of different sizes [78]. In such cases, the authors noticed that the Entropy measures more heavily penalizes a large impure cluster (a cluster is impure when it contains members from several external classes). In contrast, the F-measure and G-means do not favor larger number of clusters and attain the best score when each external class is contained in a single cluster [78]. Therefore, in our study, we rely more on the results from these two measures than the results from the Entropy measure. As such, when evaluating the clustering results from the *SM(NCut)* algorithm (i.e. a recursive, bi-partitioning, and hierarchical spectral clustering algorithm), we do not consider the Entropy scores as they may provide biased information.

Example 4.4.1. Let a given dataset contain 10 objects such as $1, 2, 3, \ldots\ldots, 10$. Suppose, the objects in this datasets are categorized in two classes: C_1 and C_2. The members of class $C_1 = 1, 4, 6, 9$ and class $C_2 = 2, 3, 5, 7, 8, 10$. Also assume that, after cluster analysis has been performed on this data, the dataset is divided into to two clusters, $D_1 = 1, 2, 6, 9, 10$ and $D_2 = 3, 4, 5, 7, 8$. Then according to the definitions given above, we have:
Total number of members in external class C_1, $|C_1| = 4$,
Total number of members in external class C_2, $|C_2| = 6$,
Total number of members in cluster D_1, $|D_1| = 5$,

Total number of members in cluster D_2, $|D_2| = 5$.

External Class	Cluster	Precision	Recall	F-measure	G-means
C_1	D_1	$P_{1,1} = 0.6$	$R_{1,1} = 0.75$	$F_{1,1} = 0.6667$	$G_{1,1} = 0.6708$
C_1	D_2	$P_{1,2} = 0.2$	$R_{1,2} = 0.25$	$F_{1,2} = 0.2222$	$G_{1,2} = 0.2236$
C_2	D_1	$P_{2,1} = 0.4$	$R_{2,1} = 0.33$	$F_{2,1} = 0.3616$	$G_{2,1} = 0.3633$
C_2	D_2	$P_{2,2} = 0.8$	$R_{2,2} = 0.67$	$F_{2,2} = 0.7293$	$G_{2,2} = 0.7321$

Table 4.2: The F-measure and G-means scores for the external class i and cluster j

Table 4.2 gives the F-measure and G-means scores for each external class i and cluster j. Then, the maximum F-measure score for external class C_1 is 0.6667 and C_2 is 0.7293. Therefore, the overall F-measure (Equation 4.8) is 0.7043. Similarly, the maximum G-means score for external class C_1 is 0.6708 and class C_2 is 0.7321. Thus, the overall G-means (Equation 4.9) score is 0.7076. The overall Entropy as calculated from Equation 4.11 is 0.8254. In this example, 3 out of 10 objects are placed incorrectly. A high value for the F-measure, G-means, and Entropy indicate that the clusters are not pure and contain objects from different *true* clusters or *classes*.

4.5 Statistical Significance Testing

In many practical applications, the scores obtained by evaluation methods are usually sufficient to compare the performance of different learning methods [76]. However, in several cases, there is a possibility that the difference may have occurred by chance and may not be statistically significant. As such, statistical tests may be applied to confirm that the results are statistically significant. Nonetheless, the use of statistical tests in machine learning has been widely debated [36], [14]. Several researchers suggest against the usage of such tests, as they believe that the tests lead to overvalued results. According to Japkowicz et al. [36], *"the use of available statistical tools for such testing in the fields of machine learning and data mining has been limited at best"*. The authors also suggest that not all the tests are appropriate for machine learning tasks. The most commonly used test in this area is called the t-test[1] [36], [76]. However, the t-test is suitable for pairwise comparison. In our study, we compare a number of proximity measures against several datasets. Therefore, to perform statistical tests on our experiments, we need

[1]For a detailed description of the t-test we refer to [36], [76].

tests that are appropriate for performing multiple comparisons on multiple datasets. There are two tests, as mentioned in [36], that are most appropriate for our experiments. The first is known as the *Analysis of Variance (ANOVA)* and the second is known as the *Friedman Test* [36]. However, ANOVA assumes that the samples are drawn from a normal distribution. In contrast, the Friedman test does not make any assumption about the distribution of data. As such, we select this test to explore the statistical significance of our experimental results. The Friedman test is available in MATLAB as a built-in function. In our case, we choose the null hypothesis as, "the performance of all the proximity measures is equivalent". The Friedman test returns the *p-value*, as a result. A significantly small *p-value* (a value close to zero) suggests that there is a significant difference between the proximity measures on the datasets used. It is common to reject the null hypothesis when the *p-value* is less than 0.05 [36]. According to [75], *"If the level is 0.05, then the results are only* 5% *likely to be as extraordinary as just seen, given that the null hypothesis is true".*

4.6 Chapter Summary

In this chapter, we discussed the experimental process that we adapted for our experiments. Recall that in Chapter 2, we discussed the four fundamentals of the cluster analysis process. For the experiments in this study, we closely followed the fundamental steps. We start with the preprocessing of the dataset so that it is ready to be used as an input to the spectral clustering algorithm. We also presented several algorithm and experimentation specific details and provided the approaches that are best suitable for the spectral clustering algorithm and our study. We applied a ten-fold cross-validation technique to ensure the accuracy of the results. We then applied these methods and approaches on two spectral methods using several proximity measures depending on the types of the attributes (i.e. binary, numeric, and mixed) present in the dataset. The experiments are tested against a number of datasets in each case.

In the next three chapters, we analyze and evaluate the performance of the proximity measures for each of the three types. In Chapter 5, we present the result analysis for datasets with binary attributes. Chapter 6 provides the result analysis for the datasets with mixed variables and Chapter 7 presents the results for the datasets with numeric attributes.

Chapter 5

Result Analysis - Binary Data

In Chapter 4, we presented the experimental approach that we have adopted to compare and evaluate the performance of several proximity measures. To compare the performance of these coefficients, we first performed spectral cluster analysis on the datasets. Next, we used the external cluster evaluation measures discussed in Chapter 4 to assess the results. These results are then used to compare the similarity coefficients. This chapter provides the analysis of our results after the experiments are performed on the datasets with binary attributes. The chapter begins with Section 5.1, with the description of datasets used in this experiment. Next, in Section 5.2, we provide the quantitative analysis of the results from the two spectral clustering algorithms, *SM(NCut)* and *NJW(K-means)*, in terms of the scores obtained from the external evaluation measures. This is followed by the discussion in Section 5.3, where we do some further analysis and suggest possible reasons that may have caused the differences in performance. We also compare the performance of the spitting methods used for the *SM(NCut)* algorithm in Section 5.4. In Section 5.5, we provide the clustering results from both the spectral clustering algorithms and we conclude the chapter with a brief summary in Section 5.6.

5.1 Binary Datasets

We use six datasets in this study. Five of the datasets are from the UCI data repository [3] and one dataset (Genes dataset) is from one of the KDD cup [53] competitions. In order to reach a conclusion with higher generality, we use more than one datasets for this task. As mentioned in Chapter 4, we use a ten-fold cross validation. The performance of the similarity coefficients are measured with the external criteria as discussed in Chapter

3. The evaluation measures use the external information (i.e. class labels) as the external criterion to evaluate the performance. Therefore, the *class labels* serve as the *true clusters* in this case. From here onward, we will refer to the class labels as the *true cluster* or *class*. The datasets are selected while keeping a number of characteristics in mind.

- The datasets are of different sizes. There are some datasets (i.e. Lenses and Balloon datasets) with only a number of objects. There are also a number of datasets (i.e. Genes, Votes, and SPECT) that are comparatively larger in size than the others.
- The datasets are based on real world problems that represent various fields and domains. For instance, the SPECT dataset is a medical dataset, whereas the Zoo dataset is based on the animal kingdom.
- We considered both imbalanced and balanced datasets. Imbalanced datasets are the datasets where the number of examples of one class is much higher than the rest of the classes [30]. The SPECT dataset is an example of a two-class imbalanced dataset in which one class contains only 36 examples, whereas the other class includes 206 examples.

Next, we describe each of the datasets. For each dataset, we also provide the information on the *true* clusters that are provided with the dataset.

SPECT Dataset: The SPECT dataset describes the diagnosing of the cardiac Single Proton Emission Computed Tomography (SPECT) images. There are 22 binary attributes and 242 instances in the dataset. The original dataset has 44 continuous attributes. These attributes are processed to create a set of 22 binary feature patterns. The UCI data repository contains both the processed dataset as well as the original numeric dataset. In this dataset, the heart condition of each patient is categorized as normal or abnormal. As such, this attribute is used as the *class label* or *true* cluster for the assessment of the results. Table 5.1 contains the information for each *true* cluster. It shows that one of the *true* clusters contains relatively fewer members than the other. Thus, it is an example of imbalanced dataset.

Votes Dataset: The dataset comprises votes for each of the US House of Representative Congressmen on 16 key votes identified by the Congressional Quarterly Almanac from the 1984 Congressional Voting Records Database. The *true* clusters represent whether a representative is Democrat or Republican (Table 5.2). There are 16 attributes and 435 instances in this dataset.

True Clusters	Number of Members
0 (Normal)	36
1 (Abnormal)	206

Table 5.1: The true cluster distribution of the SPECT dataset.

True Clusters	Number of Members
Democrats	267
Republicans	168

Table 5.2: The true cluster distribution of the Votes dataset.

Lenses Dataset: The dataset contains examples to predict the contact lens type for each of the patients. The dataset is small and includes only 24 instances and 4 attributes. However, one of the attributes is nominal with three distinct values. We use the *NominalToBinary* method from WEKA that changes a nominal attribute into several binary attributes, one for each nominal value [76]. Therefore, we have 6 binary attributes. The class attribute involves three values where 1 represents the patients who need hard contact lenses, 2 represents patients who need soft contact lenses and 3 represents patients who do not need lenses (Table 5.3). This dataset is also imbalanced as two of the categories (i.e. *hard lens* and *soft lens*) contain fewer members than the members from the category with *no lens*.

True Clusters	Number of Members
1 (Hard lens)	4
2 (Soft lens)	5
3 (No lens)	15

Table 5.3: The true cluster distribution of the Lenses dataset.

Zoo Dataset: The Zoo dataset contains 101 instances. Each of the instances in this dataset represents an animal and each animal is described with 18 attributes (including the class label). These attributes correspond to the presence and absence of various features (e.g. hair, feather, eggs, and milk) that describe each animal. The class labels present in the dataset categorizes each animal into one of the 7 categories. Since this experiment only considers the binary or boolean valued attributes, a set of 16 attributes were selected. The attributes that are not considered in this work are: 1) the unique name (string) of each of the animals and 2) the number of legs (numeric). The external class information for this dataset is given in Table 5.4. The class labels in the original dataset are represented with numbers. However, we notice that they actually represent different *classes* (a taxonomic rank in the biological classification [71]) that are used to uniquely identify the animals in the animal kingdom. We also provide the scientific class names associated with the unique class labels. The dataset is imbalanced.

True Clusters	Number of Members
1 (Mammals)	41
2 (Aves)	20
3 (Reptiles)	5
4 (Fishes)	13
5 (Amphibians)	4
6 (Insect)	8
7 (Mollusks)	10

Table 5.4: The true cluster distribution of the Zoo dataset.

Genes Dataset: This dataset was originally provided in one of the KDD cup competitions [53]. The dataset consists of 1242 genes and there are 13 binary attributes. The attributes give the information about the functions in which each of the genes is involved. The external class information denotes the genes localization based on the functions in which they are involved. Table 5.5 contains the external class information.

Balloon Dataset: A small dataset of 19 examples and 4 attributes (excluding the external class attribute). The attributes represent some of the characteristics of the balloon (i.e. size and age). The external class attribute determines whether the balloon is inflated or not. The external class distribution for this dataset is given in Table 5.6.

A summary of all of the datasets, as discussed above, is given in Table 5.7.

We perform the experiments, as outlined in Chapter 4, on these datasets to compare the performance of the six binary coefficients. Recall that all the tests are performed

True Clusters	Number of Members
Nucleus	539
Cytoplasm	257
Mitochondria	104
Cytoskeleton	88
ER	68
Plasma Membrane	61
Golgi	47
Others	78

Table 5.5: The true cluster distribution of the Genes dataset.

True Clusters	Number of Members
Inflated	8
Not Inflated	11

Table 5.6: The true cluster distribution of the Balloon dataset.

Dataset	Number of Instances	Number of True Clusters	Number of Attributes	Notes
Zoo	101	7	15	Imbalanced
Votes	435	2	16	5.3% missing values
SPECT	242	2	22	Imbalanced
Lenses	24	3	6	Imbalanced
Balloon	19	2	4	
Genes	1242	8	13	Imbalanced

Table 5.7: Summary of binary datasets used for our experiments. Given in the table (from left) the name of the datasets; the number of instances; the number of *true* clusters; the number of attributes; and some notes on the datasets.

on two different versions of the spectral clustering algorithms, *SM(NCut)* and *NJW(K-means)*. In the subsequent sections, we discuss the results obtained from each of the algorithms when combined with the similarity coefficients.

5.2 Comparison of Similarity Measures

Here we provide the results obtained from each of the datasets. Recall from Chapter 3, for binary data we used six similarity coefficients, namely *Czekanowski (CZE)*, *Jaccard (JAC)*, *Sokal and Sneath (SAS)*, *Simple Matching Coefficient (SIM)*, *Russell and Rao (RAR)* and *Rogers and Tanimoto (RAT)*. We test the performance of each of these coefficients on two different spectral clustering algorithms, *SM (NCut)* and *NJW(K-means)* and against six datasets as discussed in the previous section. In the next section, we discuss the quantitative results obtained from the external evaluation measures and the ten-fold cross validation when applied to the *SM(NCut)* algorithm.

5.2.1 Results from SM(NCut) Spectral Clustering Algorithm

The results from external evaluation measures are given in Table 5.8 and Table 5.9. Table 5.8 contains the F-measure scores for each of the datasets and Table 5.9 contains the G-means scores. Recall from Chapter 4 that the Entropy measure may not always be a suitable measure for the hierarchical clustering algorithms. Therefore, we do not consider this measure to evaluate the results from the *SM(NCut)* algorithm. As mentioned in Chapter 2, this algorithm bipartitions the objects in a given dataset by splitting the eigenvector associated with the second smallest eigenvalue of the Laplacian matrix constructed from the similarity matrix. There are five splitting points, as discussed in Chapter 4, and they are *Split Zero*, *Split Mean*, *Split Median*, *Split NCut1* and *Split NCut2*. Also recall from Chapter 4 that the first three splitting methods perform the split at a particular point (i.e. zero, mean or median) of the eigenvector, whereas, NCut-based splitting methods perform a search over the eigenvector and considers the partition that minimizes the NCut value.

The results, as given in Table 5.8 and Table 5.9, show that the average difference between the maximum and the minimum F-measure scores is 4.0% and for the G-means, the average difference is 3.9% for all the datasets and irrespective of the splitting points used. Figure 5.1 depicts the average F-measure scores for the binary datasets used in this study. Recall that the F-measure and G-means scores range from 0 to 1 and the average differences (which is calculated from the values in tables) are given as a percentage. When taking the original range into consideration, the differences are 0.04 and 0.039 respectively, which is very small. Therefore, the scores indicate that the performance of the similarity coefficients is very similar. However, the F-measure and G-means scores show that in all the cases, one or more similarity coefficients performed best for a

	Balloon						Lenses					
	CZE	JAC	SAS	SIM	RAT	RAR	CZE	JAC	SAS	SIM	RAT	RAR
Split Zero	0.62	0.62	0.63	0.72	0.70	0.59	0.56	0.59	0.56	0.57	0.54	0.54
Split Mean	0.62	0.65	0.67	0.68	0.68	0.68	0.56	0.58	0.57	0.56	0.57	0.55
Split Median	0.65	0.65	0.64	0.70	0.70	0.61	0.58	0.59	0.57	0.58	0.57	0.60
Split NCut1	0.68	0.68	0.66	0.68	0.66	0.69	0.62	0.65	0.62	0.59	0.61	0.63
Split NCut2	0.68	0.68	0.70	0.66	0.67	0.69	0.63	0.65	0.61	0.60	0.60	0.63
	SPECT						Votes					
	CZE	JAC	SAS	SIM	RAT	RAR	CZE	JAC	SAS	SIM	RAT	RAR
Split Zero	0.60	0.60	0.62	0.61	0.61	0.60	0.88	0.88	0.88	0.88	0.88	0.88
Split Mean	0.59	0.60	0.64	0.60	0.60	0.59	0.88	0.88	0.88	0.88	0.88	0.88
Split Median	0.59	0.59	0.60	0.65	0.65	0.59	0.87	0.87	0.87	0.87	0.87	0.87
Split NCut1	0.81	0.80	0.82	0.79	0.80	0.82	0.81	0.82	0.81	0.81	0.81	0.84
Split NCut2	0.80	0.81	0.81	0.76	0.79	0.81	0.80	0.80	0.81	0.80	0.84	0.82
	Zoo						Genes					
	CZE	JAC	SAS	SIM	RAT	RAR	CZE	JAC	SAS	SIM	RAT	RAR
Split Zero	0.93	0.93	0.93	0.93	0.94	0.90	0.52	0.52	0.52	0.49	0.49	0.51
Split Mean	0.92	0.92	0.92	0.94	0.94	0.90	0.52	0.52	0.52	0.49	0.48	0.51
Split Median	0.82	0.82	0.82	0.83	0.84	0.80	0.48	0.47	0.47	0.50	0.48	0.48
Split NCut1	0.74	0.76	0.78	0.77	0.78	0.75	0.46	0.49	0.50	0.48	0.49	0.50
Split NCut2	0.81	0.84	0.87	0.84	0.81	0.83	0.51	0.48	0.49	0.50	0.50	0.50

Table 5.8: The F-measure scores for the binary datasets when the SM(NCut) algorithm is used.

particular dataset. For example, the Balloon dataset performed best when the *SIM* or *RAT* coefficient is used, whereas the Lenses dataset scored best when the *JAC* coefficient is used. Here, the best performance for a similarity coefficient is measured by the number of times it achieves the highest (or close to highest) scores. For instance, if a similarity coefficient attains the highest scores for three out of five splitting methods for a given dataset, then we consider this coefficient performing the best. In this way, we ensure that the highest scores are not achieved by chance. Therefore, the performance of the similarity coefficients relies on the datasets. Below we compare the F-measure and G-means scores for each of the datasets individually and in Section 5.3 we analyze the datasets and similarity matrices to determine the causes. For each dataset we also provide the highest and lowest scores when *Split Zero* is used as the splitting point. This splitting point performed well in most of the cases and we provide a discussion on this later in the section.

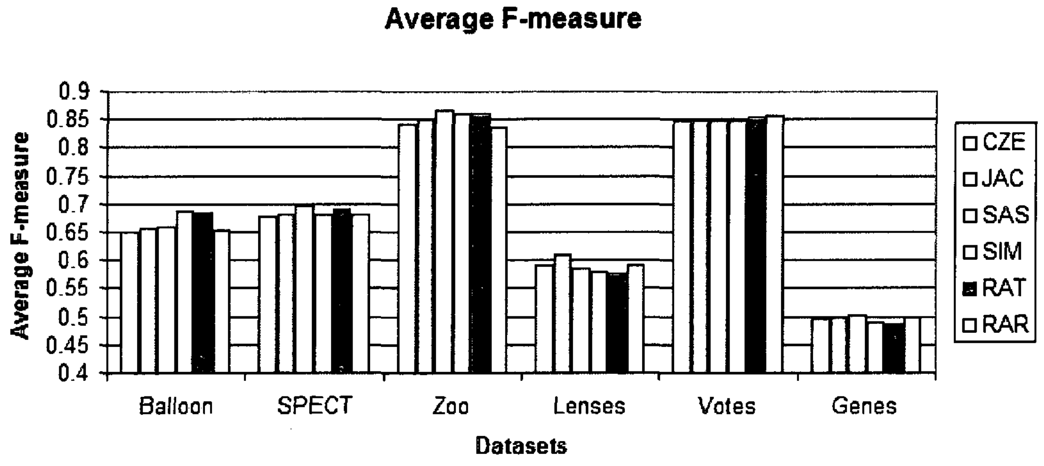


Figure 5.1: The average F-measure scores for the binary datasets.

	Balloon						Lenses					
	CZE	JAC	SAS	SIM	RAT	RAR	CZE	JAC	SAS	SIM	RAT	RAR
Split Zero	0.63	0.63	0.64	0.72	0.71	0.60	0.57	0.60	0.56	0.58	0.54	0.56
Split Mean	0.63	0.66	0.69	0.70	0.69	0.70	0.57	0.58	0.57	0.57	0.58	0.57
Split Median	0.66	0.66	0.65	0.70	0.71	0.63	0.59	0.60	0.58	0.58	0.58	0.61
Split NCut1	0.69	0.71	0.69	0.70	0.69	0.71	0.64	0.66	0.65	0.60	0.63	0.66
Split NCut2	0.70	0.69	0.71	0.68	0.69	0.70	0.65	0.66	0.64	0.61	0.61	0.67
	SPECT						Votes					
	CZE	JAC	SAS	SIM	RAT	RAR	CZE	JAC	SAS	SIM	RAT	RAR
Split Zero	0.62	0.62	0.63	0.65	0.65	0.62	0.88	0.88	0.88	0.88	0.88	0.88
Split Mean	0.61	0.62	0.66	0.63	0.63	0.61	0.88	0.88	0.88	0.88	0.88	0.88
Split Median	0.61	0.61	0.62	0.68	0.68	0.61	0.87	0.87	0.87	0.87	0.87	0.87
Split NCut1	0.82	0.82	0.84	0.80	0.80	0.84	0.82	0.83	0.82	0.82	0.81	0.84
Split NCut2	0.81	0.82	0.82	0.78	0.80	0.83	0.81	0.81	0.81	0.82	0.84	0.83
	Zoo						Genes					
	CZE	JAC	SAS	SIM	RAT	RAR	CZE	JAC	SAS	SIM	RAT	RAR
Split Zero	0.93	0.93	0.94	0.94	0.94	0.90	0.55	0.55	0.55	0.52	0.52	0.54
Split Mean	0.92	0.92	0.93	0.94	0.95	0.91	0.55	0.55	0.55	0.51	0.51	0.54
Split Median	0.83	0.83	0.83	0.84	0.84	0.81	0.50	0.50	0.50	0.52	0.50	0.51
Split NCut1	0.76	0.78	0.80	0.79	0.79	0.77	0.49	0.52	0.52	0.52	0.52	0.53
Split NCut2	0.82	0.85	0.88	0.85	0.83	0.85	0.53	0.52	0.52	0.53	0.53	0.52

Table 5.9: The G-means scores for the binary datasets when the SM(NCut) algorithm is used.

Balloon Dataset

The average F-measure score is 0.66 and the average G-means score is 0.68. The maximum F-measure and G-means scores for the Balloon dataset are 0.72 for both of the evaluation measures. The minimum scores are 0.59 (F-measure) and 0.60 (G-means), respectively. The average difference between the maximum and minimum scores over all the splitting points is 0.07. The highest scores were frequently achieved by the *SIM* and *RAT* coefficients. The lowest score was achieved by the *RAR* coefficient.

Lenses Dataset

The average F-measure and G-means scores for the Lenses dataset are, 0.59 and 0.60, respectively. The highest F-measure score is 0.59 and the minimum score is 0.54. The highest G-means score is 0.60 and the lowest score is 0.54. The average difference between the maximum and the minimum values are 0.04 (F-measure) and 0.05 (G-means). According to both the evaluation measures, the best scores were achieved by the *JAC* coefficient. In terms of both F-measure and G-means scores, the lowest scores were frequently achieved by the *RAT* and *RAR* coefficients.

SPECT Dataset

For the SPECT dataset, the average F-measure and G-means scores are, 0.68 and 0.70, respectively. The highest F-measure and G-means scores are 0.61 and 0.65 and the lowest scores are 0.59 and 0.62, respectively. The average difference between the maximum and the minimum scores are 0.04 (F-measure) and 0.05 (G-means). According to the F-measure scores, the similarity coefficient that achieved the highest scores most frequently was the *SAS* coefficient. In contrast, the G-means scores suggest that the *SIM* and *RAT* coefficients performed well. The lowest scores are achieved by the *RAR*, *CZE*, *SIM* and *JAC* coefficients.

Votes Dataset

The Votes dataset scored almost the same scores for all the coefficients. The scores are exactly the same for both the evaluation measures when *Split Zero*, *Split Mean* and *Split Median* are used as the splitting method. The NCut-based spitting methods worked well for the *RAT* and *RAR* coefficient. However, as we will see from our discussion in the next section, these methods tend to produce clusters in which one partition only contains a handful of objects. Therefore, we rely more on the results from the first three splitting methods. The average F-measure score is 0.85 and the average G-means score is 0.86. The highest scores achieved by both the measures are 0.88. The lowest score is 0.80 by F-measure and 0.81 by G-means. The average difference between the maximum and the minimum scores are 0.02 for the F-measure and 0.01 for the G-means. The average differences between the maximum and the minimum scores is low and suggest that all the similarity measures performed similarly for this dataset.

Zoo Dataset

The average F-measure score for the Zoo dataset is 0.85 and the average G-means score is 0.86, irrespective of the similarity coefficient and the splitting point used. The maximum value achieved by both the measures is 0.94. The minimum score achieved by F-measure and G-means is 0.90. The average difference between the highest score and the lowest score is 0.04 for both the evaluation measures. The *RAT* and *SIM* coefficients achieved the best scores more frequently, whereas the *RAR* coefficient scored the lowest score most of the time.

Genes Dataset

The average F-measure and G-means score for the Genes dataset are 0.50 and 0.52, respectively. The highest F-measure score is 0.52 and the lowest score is 0.49. The maximum G-means score is 0.55 and the minimum score is 0.52. The average difference between the highest and lowest score is 0.03 for both of the coefficients. The similarity coefficients that performed best for this dataset in comparison to others are: *CZE*, *SAS* and *JAC*. In contrast, the *RAT* and *SIM* coefficients scored the lowest scores.

5.2.2 Results from NJW(K-means) Algorithm

In this section, we present the results from the *NJW(K-means)* spectral clustering algorithm. Recall from Chapter 2 that this algorithm uses the k largest eigenvectors of the Laplacian matrix to find k clusters by applying the K-means algorithm on the eigenvectors. Since we use the external criteria to assess the performance of the similarity coefficients, we know the number of clusters from the number of *true* clusters. However, we use a range of values for k ($k = [2..10]$) in our experiments. This helps us to observe how the clusters are formed when k is set to a value other than the number of *true* clusters.

Table 5.10, 5.11 and 5.12 contain the results from F-measure, G-means, and Entropy, respectively. Similar to the F-measure and G-means scores for the *SM(NCut)* method described in the previous section, in this method, the scores also show that the performance of the similarity coefficient are comparable. The results indicate that the performance of a particular coefficient depends on the dataset. For example, a coefficient that works well for a certain dataset may not work well for all the datasets. The average difference between the highest and the lowest value achieved by the F-measure and G-means scores are both 6% (0.06), which indicate that the difference between the performance of the similarity coefficients are not significant. However, we noticed that there are several sit-

			Balloon								Lenses		
k	CZE	JAC	SAS	SIM	RAT	RAR	k	CZE	JAC	SAS	SIM	RAT	RAR
k2	0.63	0.63	0.63	0.69	0.68	0.62	k2	0.53	0.56	0.52	0.52	0.51	0.51
k3	0.58	0.59	0.61	0.66	0.61	0.58	k3	0.49	0.57	0.45	0.47	0.45	0.46
k4	0.52	0.50	0.49	0.57	0.54	0.53	k4	0.44	0.47	0.43	0.45	0.42	0.47
k5	0.49	0.49	0.49	0.54	0.54	0.49	k5	0.42	0.45	0.43	0.42	0.43	0.45
k6	0.49	0.48	0.44	0.50	0.49	0.44	k6	0.42	0.43	0.41	0.42	0.44	0.44
k7	0.46	0.44	0.44	0.47	0.48	0.41	k7	0.42	0.42	0.39	0.41	0.38	0.38
k8	0.45	0.44	0.41	0.47	0.46	0.42	k8	0.41	0.44	0.39	0.38	0.36	0.40
k9	0.44	0.42	0.40	0.43	0.45	0.37	k9	0.41	0.40	0.35	0.41	0.33	0.42
k10	0.43	0.39	0.36	0.41	0.37	0.36	k10	0.38	0.42	0.34	0.38	0.33	0.38
			SPECT								Votes		
k	CZE	JAC	SAS	SIM	RAT	RAR	k	CZE	JAC	SAS	SIM	RAT	RAR
k2	0.59	0.59	0.61	0.60	0.60	0.61	k2	0.88	0.88	0.88	0.88	0.88	0.88
k3	0.50	0.56	0.55	0.50	0.50	0.46	k3	0.79	0.74	0.72	0.76	0.76	0.76
k4	0.44	0.45	0.48	0.43	0.45	0.44	k4	0.64	0.65	0.65	0.66	0.66	0.64
k5	0.40	0.42	0.42	0.38	0.40	0.38	k5	0.60	0.59	0.62	0.59	0.56	0.59
k6	0.36	0.35	0.39	0.35	0.34	0.35	k6	0.53	0.54	0.55	0.55	0.54	0.54
k7	0.34	0.35	0.35	0.30	0.31	0.33	k7	0.50	0.48	0.51	0.50	0.51	0.49
k8	0.33	0.32	0.34	0.30	0.31	0.30	k8	0.47	0.47	0.47	0.48	0.47	0.45
k9	0.30	0.31	0.31	0.29	0.29	0.29	k9	0.44	0.43	0.42	0.46	0.45	0.45
k10	0.28	0.27	0.29	0.27	0.27	0.27	k10	0.42	0.40	0.41	0.44	0.41	0.43
			Zoo								Genes		
k	CZE	JAC	SAS	SIM	RAT	RAR	k	CZE	JAC	SAS	SIM	RAT	RAR
k2	0.60	0.61	0.61	0.61	0.61	0.60	k2	0.46	0.47	0.47	0.45	0.45	0.46
k3	0.72	0.67	0.70	0.67	0.70	0.69	k3	0.47	0.49	0.47	0.43	0.45	0.45
k4	0.68	0.72	0.77	0.68	0.75	0.71	k4	0.50	0.48	0.49	0.47	0.47	0.46
k5	0.70	0.71	0.76	0.72	0.67	0.64	k5	0.48	0.49	0.48	0.45	0.46	0.47
k6	0.70	0.76	0.72	0.73	0.75	0.72	k6	0.47	0.46	0.47	0.44	0.47	0.44
k7	0.76	0.75	0.74	0.70	0.70	0.75	k7	0.46	0.46	0.47	0.45	0.45	0.44
k8	0.73	0.72	0.74	0.67	0.70	0.71	k8	0.46	0.45	0.44	0.45	0.47	0.42
k9	0.70	0.71	0.74	0.68	0.73	0.69	k9	0.46	0.43	0.46	0.43	0.45	0.42
k10	0.68	0.68	0.71	0.69	0.69	0.71	k10	0.43	0.43	0.42	0.42	0.45	0.41

Table 5.10: The F-measure scores for the binary datasets when NJW(K-means) spectral clustering algorithm is used.

uations where some of the similarity measures frequently achieved either the highest or the lowest value for a given dataset. Below, we discuss the quantitative results for each individual dataset. For each dataset, we also provide the highest and lowest scores when k is set to the number of *true* clusters.

			Balloon							Lenses			
k	CZE	JAC	SAS	SIM	RAT	RAR	k	CZE	JAC	SAS	SIM	RAT	RAR
k2	0.64	0.63	0.63	0.70	0.69	0.63	k2	0.55	0.59	0.54	0.53	0.53	0.52
k3	0.60	0.60	0.62	0.68	0.64	0.59	k3	0.51	0.60	0.47	0.49	0.47	0.48
k4	0.54	0.53	0.50	0.60	0.57	0.56	k4	0.47	0.51	0.46	0.48	0.46	0.49
k5	0.53	0.54	0.55	0.58	0.59	0.54	k5	0.46	0.49	0.47	0.46	0.47	0.48
k6	0.54	0.53	0.50	0.56	0.56	0.49	k6	0.45	0.47	0.46	0.45	0.48	0.48
k7	0.51	0.50	0.51	0.55	0.54	0.47	k7	0.48	0.49	0.45	0.46	0.44	0.43
k8	0.52	0.52	0.48	0.54	0.53	0.48	k8	0.48	0.50	0.47	0.44	0.43	0.45
k9	0.51	0.50	0.48	0.52	0.53	0.45	k9	0.48	0.47	0.43	0.47	0.41	0.49
k10	0.50	0.48	0.44	0.50	0.47	0.44	k10	0.45	0.49	0.43	0.45	0.41	0.46
			SPECT							Votes			
k	CZE	JAC	SAS	SIM	RAT	RAR	k	CZE	JAC	SAS	SIM	RAT	RAR
k2	0.62	0.61	0.63	0.63	0.63	0.63	k2	0.88	0.88	0.88	0.88	0.88	0.88
k3	0.54	0.59	0.58	0.56	0.55	0.51	k3	0.80	0.76	0.75	0.78	0.78	0.78
k4	0.50	0.50	0.52	0.49	0.51	0.49	k4	0.68	0.69	0.69	0.70	0.69	0.68
k5	0.46	0.49	0.48	0.46	0.48	0.45	k5	0.64	0.64	0.67	0.63	0.61	0.63
k6	0.44	0.42	0.46	0.43	0.43	0.42	k6	0.59	0.60	0.61	0.61	0.61	0.60
k7	0.42	0.43	0.43	0.40	0.41	0.41	k7	0.57	0.56	0.58	0.57	0.58	0.56
k8	0.41	0.41	0.42	0.39	0.40	0.39	k8	0.55	0.55	0.55	0.55	0.55	0.53
k9	0.39	0.40	0.40	0.38	0.38	0.38	k9	0.53	0.52	0.50	0.54	0.53	0.53
k10	0.38	0.36	0.38	0.36	0.36	0.37	k10	0.51	0.49	0.50	0.52	0.51	0.52
			Zoo							Genes			
k	CZE	JAC	SAS	SIM	RAT	RAR	k	CZE	JAC	SAS	SIM	RAT	RAR
k2	0.67	0.67	0.67	0.67	0.67	0.66	k2	0.53	0.54	0.54	0.49	0.51	0.53
k3	0.75	0.71	0.74	0.72	0.74	0.73	k3	0.52	0.54	0.52	0.47	0.49	0.50
k4	0.71	0.75	0.79	0.72	0.77	0.74	k4	0.53	0.51	0.52	0.50	0.50	0.49
k5	0.73	0.75	0.79	0.75	0.71	0.68	k5	0.51	0.52	0.50	0.48	0.49	0.50
k6	0.73	0.78	0.75	0.76	0.78	0.74	k6	0.50	0.48	0.50	0.46	0.49	0.46
k7	0.78	0.78	0.77	0.73	0.73	0.77	k7	0.49	0.49	0.49	0.48	0.48	0.47
k8	0.75	0.75	0.77	0.71	0.73	0.73	k8	0.49	0.49	0.48	0.47	0.49	0.45
k9	0.73	0.74	0.77	0.71	0.75	0.72	k9	0.49	0.46	0.48	0.46	0.48	0.45
k10	0.71	0.72	0.74	0.72	0.72	0.74	k10	0.46	0.47	0.45	0.46	0.48	0.44

Table 5.11: The G-means scores for the binary datasets when NJW(K-means) spectral clustering algorithm is used.

Balloon Dataset

The Balloon dataset achieved the best scores when the *SIM* and *RAT* coefficients are used as the similarity coefficients. According to the F-measure scores, the *SIM* coefficient scored 0.69 and the *RAT* coefficient scored 0.68. In contrast, the *RAR* coefficient scored the minimum score. The F-measure value for the *RAR* coefficient is 0.62 when

			Bolloon								Lenses			
k	CZE	JAC	SAS	SIM	RAT	RAR	k	CZE	JAC	SAS	SIM	RAT	RAR
k2	**0.64**	**0.64**	**0.64**	0.56	0.56	**0.64**	k2	0.89	0.82	0.90	0.90	**0.91**	0.90
k3	0.56	**0.58**	0.56	0.47	0.50	**0.58**	k3	0.86	0.68	**0.89**	0.86	0.88	0.88
k4	0.56	0.55	**0.59**	0.47	0.48	0.53	k4	0.82	0.74	**0.86**	0.81	0.83	0.79
k5	**0.51**	0.45	0.47	0.41	0.42	0.50	k5	0.77	0.68	0.78	0.76	**0.79**	0.73
k6	0.44	0.42	0.47	0.38	0.38	**0.52**	k6	**0.73**	0.67	0.72	0.71	0.68	0.68
k7	0.40	0.41	0.39	0.32	0.34	**0.47**	k7	0.63	0.55	0.67	0.63	0.67	**0.69**
k8	0.34	0.32	0.39	0.29	0.30	**0.42**	k8	0.54	0.48	0.60	0.60	**0.67**	0.60
k9	0.29	0.29	0.33	0.26	0.25	**0.36**	k9	0.49	0.44	0.61	0.54	**0.62**	0.51
k10	0.26	0.28	**0.34**	0.20	0.29	0.31	k10	0.47	0.41	0.56	0.49	**0.58**	0.49
			SPECT							Votes			
k	CZE	JAC	SAS	SIM	RAT	RAR	k	CZE	JAC	SAS	SIM	RAT	RAR
k2	**0.42**	**0.42**	**0.42**	0.38	0.38	**0.42**	k2	0.34	0.34	0.34	0.34	0.34	0.34
k3	**0.42**	**0.42**	0.41	0.38	0.39	**0.42**	k3	0.32	**0.34**	0.30	0.33	0.32	0.32
k4	**0.41**	**0.41**	**0.41**	0.38	0.37	**0.41**	k4	**0.31**	0.29	0.30	0.30	0.30	0.29
k5	**0.41**	**0.41**	**0.41**	0.37	0.36	**0.41**	k5	**0.30**	0.28	0.28	0.29	**0.30**	0.29
k6	0.40	0.40	0.40	0.36	0.37	**0.41**	k6	0.29	0.25	0.25	**0.31**	0.26	0.27
k7	0.40	0.39	0.40	0.37	0.37	**0.41**	k7	0.28	0.26	0.26	**0.29**	0.27	0.25
k8	0.39	0.39	0.38	0.37	0.36	**0.40**	k8	0.25	0.25	0.25	**0.29**	0.25	0.26
k9	0.38	0.38	0.38	0.36	0.36	**0.40**	k9	0.23	0.24	0.24	**0.26**	0.24	0.24
k10	0.38	0.38	0.37	0.36	0.35	**0.39**	k10	0.24	0.24	0.23	**0.26**	0.25	0.24
			Zoo							Genes			
k	CZE	JAC	SAS	SIM	RAT	RAR	k	CZE	JAC	SAS	SIM	RAT	RAR
k2	1.02	0.98	0.98	0.99	0.99	1.06	k2	**1.61**	1.45	1.45	1.54	1.48	1.51
k3	0.72	0.77	0.71	0.75	0.72	0.76	k3	1.39	1.36	1.40	**1.44**	1.42	1.41
k4	0.64	0.59	0.51	0.63	0.53	**0.64**	k4	1.33	1.34	1.33	1.35	**1.38**	**1.38**
k5	0.51	0.46	0.47	0.47	0.53	**0.61**	k5	1.34	1.31	1.34	**1.35**	1.33	1.33
k6	0.44	0.40	0.41	0.41	0.33	**0.47**	k6	1.30	1.31	1.33	**1.34**	**1.34**	1.32
k7	0.30	0.30	0.34	0.36	0.33	**0.38**	k7	1.27	1.29	1.27	**1.32**	1.31	1.30
k8	0.29	0.34	0.29	0.36	0.31	**0.41**	k8	1.26	1.27	1.26	**1.31**	1.30	1.27
k9	0.30	0.29	0.27	0.29	0.26	**0.36**	k9	1.25	1.27	1.26	**1.29**	1.28	1.26
k10	0.25	0.27	0.24	0.28	0.26	**0.30**	k10	1.24	1.24	1.26	1.25	**1.28**	1.24

Table 5.12: The Entropy scores for the binary datasets when NJW(K-means) spectral clustering algorithm is used.

$k = 2$. The G-means and Entropy scores also indicate a similar pattern. The maximum F-measure score is 0.69 and the minimum is 0.62 when $k = 2$. The maximum G-means score is 0.70 and the minimum is 0.63, whereas the minimum Entropy (the lower the value the better) is 0.54 and the maximum is 0.64.

Lenses Dataset

The JAC similarity coefficient performed the best for the Lenses dataset. In contrast, the *RAT* similarity coefficient scored the lowest. The Entropy scores also show a similar trend where the *JAC* coefficient performed the best and the *RAT* coefficient scored the highest value (for Entropy the lower the value the better). The maximum F-measure, G-means and the minimum Entropy scores are, 0.57, 0.60, and 0.68, respectively, when $k = 3$. The minimum F-measure, G-means and the maximum Entropy scores are, 0.44, 0.46, and 0.89, respectively.

SPECT Dataset

According to the F-measure scores, the SPECT dataset performed well for the *SAS* and *RAR* coefficients. In contrast, the G-means scores suggest that the *SIM*, *RAT*, *SAS*, and *RAR* coefficients scored the highest values. When compared against the Entropy scores, it achieved the lowest scores for the *SIM* and *RAT* coefficients. Therefore, we consider the *SAS*, *SIM*, and *RAT* coefficients to be the best performers for this dataset. However, we noticed that the difference is very low between the highest and lowest value when $k = 2$. For F-measure and G-means both, this value is 0.02. The maximum F-measure score is 0.61 and the minimum score is 0.59 when $k = 2$. The maximum G-means score is 0.63 and the minimum is 0.61.

Votes Dataset

When consulting the external class information, all three evaluation measures suggest that the Votes dataset performs equally well for all six coefficients. The highest and the lowest F-measure and G-means score is 0.88 when $k = 2$. The Entropy score is also the same for all the similarity measures. When $k = 2$, the Entropy score for this dataset is 0.34.

Zoo Dataset

According to the F-measure, G-means, and Entropy scores, the *JAC* and *CZE* coefficients scored the highest for this dataset. When k is set to the external number of classes, the Entropy scores indicate that the *RAR* similarity coefficient scored the highest entropy. The maximum F-measure and G-means scores are (when $k = 7$) 0.75 and 0.78, respectively. The minimum scores are 0.63 and 0.73, accordingly.

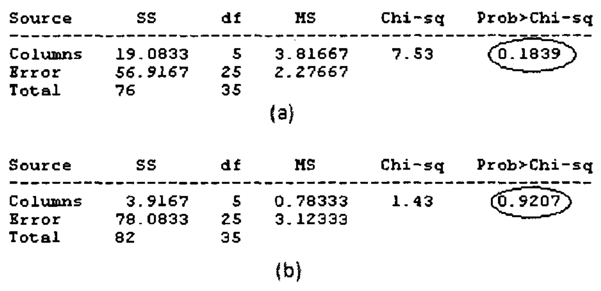
Genes Dataset

According to the F-measure and G-means scores, the Genes dataset scored the highest scores when the *CZE*, *JAC*, and *RAT* coefficients are used. For the same evaluation measures, the *RAR* coefficient scored the lowest values when compared to the rest of the similarity coefficients. Results from the G-means and Entropy measures also show similar trends. The maximum F-measure and G-means scores when $k = 8$ are 0.46 and 0.49, respectively. The minimum scores are 0.42 and 0.45, accordingly.

In the next section we analyze the results from this section and evaluate the performance of the similarity measures.

5.3 Result Evaluation

In the previous two sections, we provided the quantitative results from our experiments. The analysis showed similar results for both of the spectral clustering algorithms. The results indicated that the average difference between the highest score and the lowest score achieved for each of the datasets is low for all the evaluation measures. Recall from Chapter 4, that we applied the Friedman test to measure the statistical significance of our results. Figure 5.2 shows the results, when the Friedman test is applied on the results from the *SM(NCut)* and *NJW(K-means)* algorithm. The *p-values* are 0.1839 and 0.9207, respectively. In this case, both the *p-values* are greater than 0.05. As mentioned in Chapter 4, this indicate that the difference between the performance of the proximity measures for binary data is not statistically significant on our datasets. The



```
Source       SS      df      MS       Chi-sq   Prob>Chi-sq
--------------------------------------------------------------
Columns   19.0833     5   3.81667     7.53       0.1839
Error     56.9167    25   2.27667
Total     76         35
```

(a)

```
Source       SS      df      MS       Chi-sq   Prob>Chi-sq
--------------------------------------------------------------
Columns    3.9167     5   0.78333     1.43       0.9207
Error     78.0833    25   3.12333
Total     82         35
```

(b)

Figure 5.2: Results from the Friedman test when applied on the results from (a) the SM(NCut) algorithm and (b) the NJW(K-means) algorithm.

average difference between the F-measure scores are 0.04 and 0.06 for *SM(NCut)* and *NJW(K-means)* algorithm, respectively. The average differences for the G-means scores are 0.039 and 0.06, accordingly. This indicates that the performance of the coefficients is similar but varies slightly from one another. There is no single similarity coefficient that outperformed for all or a majority of the datasets. In contrast, the results show that the performance of the similarity coefficients varies depending on the datasets. One similarity coefficient may perform slightly better than the others for a given dataset, whereas for another dataset, a different coefficient may outperform. For example, the *JAC* coefficient scored the highest scores for the Lenses dataset and the *SIM* coefficient scored the highest for the Balloon dataset. Nevertheless, one exception is the *RAR* coefficient. The *RAR* coefficient frequently attained the lowest scores, compared to the rest of the five similarity coefficients. Among the six datasets used for the experiments, the *RAR* coefficient achieved the lowest scores in four (i.e. Balloon, Lenses, SPECT and Zoo datasets) of the datasets when the *SM(NCut)* algorithm is used for testing. We also found similar results for the *RAR* coefficient when the experiments are performed on the *NJW(K-means)* algorithm. In this case, the four datasets, i.e. Balloon, SPECT, Zoo, and Genes, also scored the lowest for the *RAR* similarity coefficient. Therefore, the results indicate that the *RAR* coefficient is the only coefficient for which the performance is lower, on average, than the other coefficients for majority of the datasets.

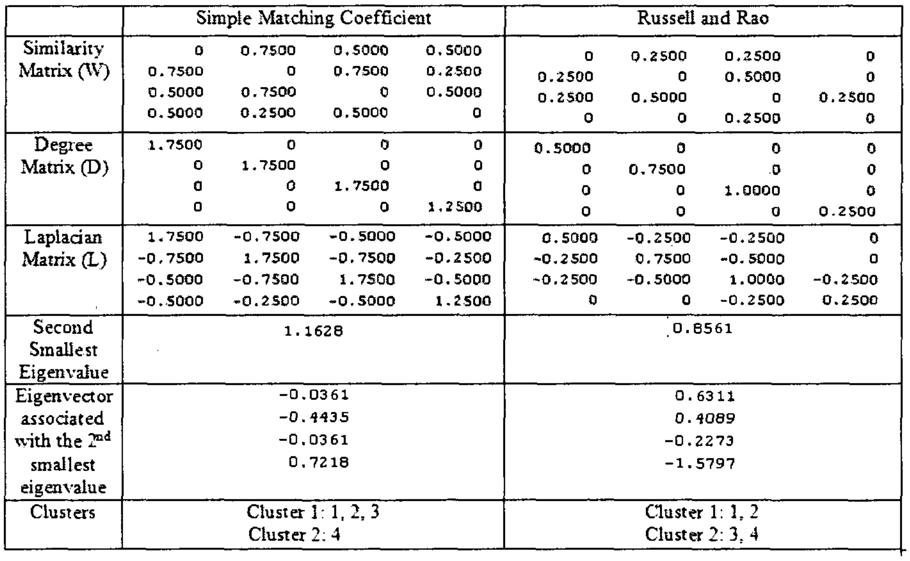
Different coefficients give different similarity values for a given pair of objects. Recall from Chapter 3 that there are four variables (a, b, c and d) that are used to define the similarity coefficients. If 1 is denoted as *present* (the positive cases) and 0 is denoted as *absent* (the negative cases) then for any two instances (x and y) in the dataset, these four variables are defined as: a = *number of occurrences when* $x_i = 1$ *and* $y_i = 1$, b = *number of occurrences when* $x_i = 0$ *and* $y_i = 1$, c = *number of occurrences when* $x_i = 1$ *and* $y_i = 0$ and d = *number of occurrences when* $x_i = 0$ *and* $y_i = 0$. The coefficients vary from one another depending on the weight they give on each of the four variables. Since, in our case the results vary over the datasets, we observe the characteristics of the datasets to find the possible causes.

In spectral clustering, the objects in a given dataset are considered the nodes in a graph and the similarity between them are the edges between the objects. To find the clusters from this data, the algorithms from this family perform a cut on this graph such that the similarity within the clusters is maximized and the similarity between the clusters is minimized. Since different coefficients give different similarity values for the same pair of objects, the similarity matrix created from one coefficient may vary from another coefficient. Therefore, depending on the values in the similarity matrix, the cut position may also change. As such, the clustering results may also be different for two different coefficients. For instance, consider a small dataset of four objects as given in Table 5.13. Figure 5.3 contains the similarity matrix, degree matrix, Laplacian matrix and the eigenvalue and eigenvector when the *SIM* coefficient and the *RAR* coefficient are used as the similarity coefficient. When the *SIM* coefficient is used, the spectral clustering algorithm returns one cluster with objects 1, 2, and 3 and a second cluster with only object 4. In contrast, the *RAR* coefficient returns two clusters where one of them contains object 1 and object 2 and the second cluster contains object 3 and object

4. Now, depending on the *true* cluster labels, both the similarity coefficients may give different values for the external evaluation scores.

ID	a1	a2	a3	a4
1	0	0	1	0
2	0	0	1	1
3	1	0	1	1
4	1	0	0	0

Table 5.13: Sample binary dataset.



	Simple Matching Coefficient	Russell and Rao
Similarity Matrix (W)	$\begin{array}{rrrr} 0 & 0.7500 & 0.5000 & 0.5000 \\ 0.7500 & 0 & 0.7500 & 0.2500 \\ 0.5000 & 0.7500 & 0 & 0.5000 \\ 0.5000 & 0.2500 & 0.5000 & 0 \end{array}$	$\begin{array}{rrrr} 0 & 0.2500 & 0.2500 & 0 \\ 0.2500 & 0 & 0.5000 & 0 \\ 0.2500 & 0.5000 & 0 & 0.2500 \\ 0 & 0 & 0.2500 & 0 \end{array}$
Degree Matrix (D)	$\begin{array}{rrrr} 1.7500 & 0 & 0 & 0 \\ 0 & 1.7500 & 0 & 0 \\ 0 & 0 & 1.7500 & 0 \\ 0 & 0 & 0 & 1.2500 \end{array}$	$\begin{array}{rrrr} 0.5000 & 0 & 0 & 0 \\ 0 & 0.7500 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0.2500 \end{array}$
Laplacian Matrix (L)	$\begin{array}{rrrr} 1.7500 & -0.7500 & -0.5000 & -0.5000 \\ -0.7500 & 1.7500 & -0.7500 & -0.2500 \\ -0.5000 & -0.7500 & 1.7500 & -0.5000 \\ -0.5000 & -0.2500 & -0.5000 & 1.2500 \end{array}$	$\begin{array}{rrrr} 0.5000 & -0.2500 & -0.2500 & 0 \\ -0.2500 & 0.7500 & -0.5000 & 0 \\ -0.2500 & -0.5000 & 1.0000 & -0.2500 \\ 0 & 0 & -0.2500 & 0.2500 \end{array}$
Second Smallest Eigenvalue	1.1628	0.8561
Eigenvector associated with the 2nd smallest eigenvalue	-0.0361 -0.4435 -0.0361 0.7218	0.6311 0.4089 -0.2273 -1.5797
Clusters	Cluster 1: 1, 2, 3 Cluster 2: 4	Cluster 1: 1, 2 Cluster 2: 3, 4

Figure 5.3: The spectral clustering algorithm applied on the sample data given in Table 5.13. The first column contain the results when the *SIM* coefficient is used as the similarity measure and the second column contain the results for the *RAR* coefficient.

Recall that our results from the previous section indicated that the *RAR* coefficient performed slightly lower than the rest of the five coefficients. We noticed that the *RAR* coefficient is the only coefficient that considers d in the denominator but not in the numerator. The numerator, in this case is a (the number of positive matches), and the denominator is $a+b+c+d$ which is same as the total number of attributes in the dataset. Therefore, this coefficient gives the proportion of positive matches calculated against the total number of attributes. Other coefficients either neglect the negative matches (i.e.

CZE, *JAC*, *SAS*) or give equal weight to both the positive and the negative matches (i.e. *SIM*, *RAT*). Combining the negative matches in the denominator, on average, results in a smaller similarity value for a dataset, unless the dataset contains more 1's than 0's. For instance, for the Votes dataset, where the number of 1's is more than the number of 0's, the *RAR* coefficient performed as well as the other coefficients. We noticed that the similarity matrix constructed from the *RAR* coefficient often contains values in a smaller range, where the similarity varies over several fixed values. If there are 3 attributes then we know that the possible values are: 0 (when $a = 0$), 1/3 (when $a = 1$), 2/3 (when $a = 2$), and 3/3 (when $a = 3$). Moreover, the frequency of scoring a higher similarity value is also very low and most of the values that dominate the similarity matrix are among the ones with smaller similarity value. This is because most of the datasets we used contain more 0's than 1's. A higher similarity value in this case means that two objects contain many positive matches. Consider the Balloon dataset, which contains 4 attributes. There are five possible values that this coefficient may take (i.e. 0, 0.25, 0.5, 0.75, and 1.0). The similarity matrix for this dataset is mostly dominated by the value 0.25 (42.90%) and 0.5 (29.06%). Therefore, the objects that may not otherwise be placed together, are placed in the same cluster by the *RAR* coefficient for having the same similarity value as the other objects in that cluster. This indicates that using the variable d only in the denominator may slightly degrade the performance, especially for the situations when the number of positive matches is low in the dataset.

As mentioned, the Balloon dataset contains four attributes. For all four attributes, the values (1 and 0) represent two different states of an attribute with equal weight. For instance, the attribute color has two values, yellow and purple, where 1 is used to represent yellow and 0 is used to represent purple. Therefore, the attributes in this dataset carry equal weight. Our results show that the *SIM* and *RAT* coefficient scored the highest for this dataset. Recall from Chapter 3 that these two coefficients give equal weight to both, the positive (1) and the negative (0) values as both contain the variable d in their equations (in both, the numerator and the denominator). There are several objects that are clustered differently by the rest of the four coefficients because of not giving equal weight to both 1 and 0. For example, two objects from this dataset are $O_1 = 1, 0, 0, 0$ and $O_2 = 1, 0, 0, 1$. According to the class labels provided with the dataset, they belong to the same *true* cluster. According to our results, when we used *SIM* as the similarity coefficient, they are placed in the same cluster. In contrast, when we used *RAR* as the coefficient, they are placed in two different clusters. The similarity between these two objects is, 0.75 for the *SIM* coefficient and 0.25 for the *RAR* coefficient.

The Lenses dataset contains six attributes. The first three attributes contain twice the number of 0's as 1's. For these three attributes, the 1's represent the positive values and carry more weight. Therefore, the significance of positive matches a is much higher than the negative matches d. The rest of the three attributes are symmetric where both the values 1's and 0's carry equal weight. The results show that the JAC coefficient performed well for this dataset and *SIM*, RAT and RAR coefficients scored the lowest. Recall from Chapter 3 that the *SIM* coefficient and RAT coefficient both give equal weight to both the variables a and d and that the JAC coefficient do not consider the negative matches. Since we have more pairs of negative matches in the first three attributes, including these pairs in the calculation of similarity affects the overall similarity. Therefore, there are several situations where two objects that are different from one another, have high similarity values and are placed in the same cluster. For instance, two objects from this datasets are $O_1 = 0,1,0,0,1,0$ and $O_2 = 0,1,0,1,0,0$. According to the external class information, they belong to two different categories. When the *SIM* coefficient is used as the similarity coefficient to cluster the objects, these two objects are placed in the same cluster. In contrast, when the JAC coefficient is used, the objects are placed in two different clusters.

In the case of SPECT and Genes, both the datasets contain more 0's than 1's. The SPECT dataset contains almost twice as many 0's (65.63%) as 1's (34.37%), whereas the Genes dataset contains almost 4 times as many 0's (80.09%) as 1's (19.91%). Our results show that the SAS coefficient performed better than the other similarity coefficients. This is a coefficient that gives more weight to the positive matches and neglects the negative matches. This indicates that the positive attributes in these datasets are given more weight than the negative attributes. Moreover, as we mentioned previously, the datasets contain more 0's than 1's. This measure also gives more weight to the unmatched pairs in the denominator. As a result, if a pair of objects contains more unmatched pairs than the positive matches, the similarity between the objects will be very low. In contrast, the similarity will be high only if there are more positive matches. The *SIM* and RAT coefficients performed slightly lower as they both give equal weight to the positive and negative matches. Therefore, when there are many pairs of negative matches in between a pair of objects these coefficients will give a very high similarity value. As such, they may be placed together in the same cluster, despite being different from one another. We also noticed that the two other coefficients (JAC and *CZE*), which are suitable for these types of datasets, also perform well for the Genes dataset. The results also indicate that, among these three coefficients, the performance is slightly better for SAS and JAC than

CZE. The difference in general is very small; less than 1%. For these three coefficients, the difference between the functions is that, while they all give equal weight to the positive matches (variable a), they give different amounts of weight to the unmatched pairs (variables b and c) in the denominator. For instance, the SAS coefficient gives double weight to the unmatched pairs, the *CZE* coefficient gives the lowest weight (half the weight) and the JAC coeffcient keeps the original. Therefore, the similarity value will be low when the number of positive values (a) is small and the number of unmatched pairs is relatively high. For instance, if $a = 1$ and $b + c = 10$ then, the similarity values are 0.0476 for the SAS coefficient, 0.0909 for the JAC coeffcient, and 0.1667 for the *CZE* coefficient. In contrast, when a is high, and $b + c$ is low, the situation will be reversed. For these two datasets, we have more 0's than 1's and both of the datasets have a large number of attributes. Therefore, this may be one of the possible reasons for the SAS coeffcient and the JAC coefficient performing slightly better than the *CZE* coefficient.

The Votes dataset is the only dataset for which all the similarity coefficients achieved almost the same scores. This dataset also differs from the rest of the datasets used in this study in the sense that the number of 1's in the dataset is higher than the number of zeros. The ratio of the number of 1's and 0's is 53.30 : 46.70 (in percentage). The dataset contains 16 attributes represented by the values, *yes* and *no*. We notice that for each of the objects in the dataset, the number of 1's on average is higher than the number of 0's. One of the possible reasons the coefficients perform exactly the same is that the inclusion of the negative matches does not have impact on the similarity. Therefore, the similarity coefficients that only consider the positive matches perform similar to the coefficients that consider both a and d.

The Zoo dataset contains 15 attributes to denote a number of features that are used to describe each of the animals in the datasets. For each of the attributes, the values denote the *presence* and *absence* of a feature in a given object in the dataset. According to the *SM(NCut)* algorithm, the *SIM* coefficient and the RAT coefficient performed the best and according to the *NJW(K-means)* algorithm, the SAS coefficient and the JAC coefficient performed well. The attributes in the dataset give equal weight to both the attribute values where 56.44% values are from *absence* (0's) and 43.56% values are from *presence* (1's). The *SIM* and RAT coefficients performed slightly better as they both give equal weight to both the attribute values. One possible reason that the SAS coefficient and the JAC coefficient also performed well for this dataset may be that the ratio of 0's and 1's is not too high. As for the *NJW(K-means)* algorithm, we notice, that the algorithm tends to split the largest *true* cluster into smaller subgroups as k increases.

This may be one possible reason that the *NJW(K-means)* algorithm provided a slightly different result.

In summary, our results indicate that in terms of the evaluation measures, all of the similarity coefficients show similar results except for the *RAR* coefficient. However, as expected the difference between the performances of each individual coefficient depends on the datasets. We noticed that when the amount of 1's is relatively higher than the 0's, the performances of the similarity coefficients are the same (i.e. the Votes dataset). We also noticed that when the datasets contain relatively more 0's than 1's (i.e. the SPECT and Genes Dataset) and when the attributes give more weight to the positive values, then the *CZE*, *JAC* and *SAS* coefficients performed well. All the three coefficients give equal weight to the positive matches (a), do not consider the negative matches (d), and only vary on the amount of weight each of them give on the unmatched pairs (b and c). The results do not show any particular pattern within the performance of these three coefficients. The only coefficient that scored the lowest by the evaluation measures for majority of the datasets most often, is the *RAR* coefficient. Therefore, this coefficient may not be a suitable choice when spectral clustering algorithms are performed on the binary datasets.

5.4 Comparing the Performance of Splitting Methods

As mentioned above, the *SM(NCut)* algorithm uses a splitting point to bipartition the eigenvector. We discussed five splitting methods, that are proposed in previous studies, in Chapter 4. In this section, we compare the results from the perspective of the splitting methods. When comparing the splitting methods, the results from Table 5.8 and Table 5.9 show that the *Split Zero* method often performed well for most of the datasets irrespective of the similarity measures used. The F-measure scores show that in 51.43% cases the *Split Zero* method achieved the highest scores. As for the G-means scores, in 54.28% cases, this method performed the best. We also noticed that in most of the cases the *Split Mean* method achieved a score very close to the one obtained from the *Split Zero* method. The difference is, on average, only 1.0 or 2.0 percent. This is followed by the *Split NCut2* method, which achieved the highest scores in 48.57% cases for the F-measure and 42.86% cases for the G-means. In 34.28% (F-measure) and 40% (G-means) cases the *Split NCut1* method achieved the highest scores. We found that there is a similarity between the values obtained from the *Split NCut1* and *Split NCut2* methods. The *Split NCut1* method usually achieved either the same value or a value very close to the score achieved by the *Split NCut2* method. However, we also noticed that in all the datasets except for the Zoo and Lenses datasets, these two splitting points tend to have partitions with very small sets of objects. Therefore, the results from these two splitting methods may be biased in some situations. Thus, even though the scores show that the *Split NCut1* and *Split NCut2* method work well, this claim may need to be checked manually by looking at the partitions. For this reason, these two splitting points may not always be a good option to consider. Among the five splitting points, the *Split Median* method scored low most of the time. In 34.28% cases for the F-measure and 42.85% cases for the G-means, the splitting method scored the lowest values. This is also confirmed by the low percentage of cases when the *Split Median* method achieved the highest scores. The splitting method did not work well in any cases for the F-measure scores and only worked well in 2.85% cases for the G-means score, which is very low compared to the other splitting methods.

Recall from Chapter 2 that the eigenvector associated with the second smallest eigenvalue is used to partition the data into two subgroups. In the ideal case, this vector should contain two discrete values and the signs of these values usually indicate the cluster assignments. We also discussed in Chapter 2 that the eigenvectors usually take continuous values, rather than the two discrete values, when applied to real problems [58]. However, we notice that even though the values are continuous, they range from positive to negative values. Thus, splitting at 0 will separate the positive values from the negative values and this situation becomes very close to the ideal case. This is also true for the split at mean method. In most of the cases, we noticed that the mean value is very close to the point zero. In contrast, the split at median method usually performs poorly since it splits the eigenvector exactly at the middle point, disregarding the values. Thus, we get two same sized partitions which may not separate the clusters correctly. The case is also similar for the splitting methods based on the normalized cut. These methods perform a search over the eigenvector values and take the cluster that minimizes the NCut value. For instance, the *Split NCut2* method performs a search over m evenly spaced values and considers the one that minimizes the NCut value as discussed in Chapter 2. In several cases, these two methods create partitions with only a number of objects. Therefore, this observation indicates that the *Split Zero* or *Split Mean* method may be a better choice as a splitting method.

5.5 Clustering Results

In this section, we analyze the results to determine the performance of the spectral clustering algorithms on each of the datasets. Table 5.14 contains the best F-measure scores for each of the datasets. For the *SM(NCut)* algorithm, this is the maximum score achieved by the similarity measure that performed the best combined with the *Split Zero* option. For the *NJW(K-means)* algorithm, this is the score when the number of clusters is set to the number of *true* clusters for the similarity measure that scored the best.

Dataset	SM(NCut)	NJW(K-means)
Balloon	0.72	0.69
Lenses	0.59	0.57
SPECT	0.62	0.61
Votes	0.88	0.88
Zoo	0.94	0.75
Genes	0.52	0.46

Table 5.14: The F-measure scores for the SM(NCut) and NJW (K-means) spectral clustering algorithm for the binary datasets.

The F-measure scores indicate that the *SM(NCut)* algorithm performs slightly better than the *NJW(K-means)* algorithm. This confirms the work performed in [66] where several different versions of the spectral clustering algorithm were compared. The approach, evaluation criteria, and the datasets considered for the work performed in [66] are different from our work. In [66], the possible reason is not clearly indicated; however, they suggest that the *NJW(K-means)* algorithm uses larger eigenvectors which may add noise to the solution.

In terms of the datasets, the results from Table 5.14 indicate that the spectral clustering algorithm, in general, performed very well for the Votes and Zoo datasets. We may verify this by comparing the similarity matrix ordered by the *true* clusters and the solution from the spectral clustering algorithms. If the dataset contains well-separated clusters, than by ordering them according to the *true* cluster labels, we will get a block diagonal similarity matrix [63]. The blocks on the main diagonal refer to each individual cluster where the similarity is high. The off diagonal boxes refer to the between cluster similarity. Figure 5.4(b) depicts the ordered similarity matrix according to the *true* cluster information. It shows that there are two well-defined clusters in the dataset. Figure 5.4(a) depicts the similarity matrix from the spectral clustering algorithm. In this figure there are also two well-defined blocks representing the underlying clusters on the main diagonal. The situation is also similar for the Zoo dataset. The Zoo dataset is an example of an imbalanced dataset, as one of the *true* clusters contains relatively more objects than the others. This indicates that the spectral clustering algorithm works well even when the dataset is imbalanced.

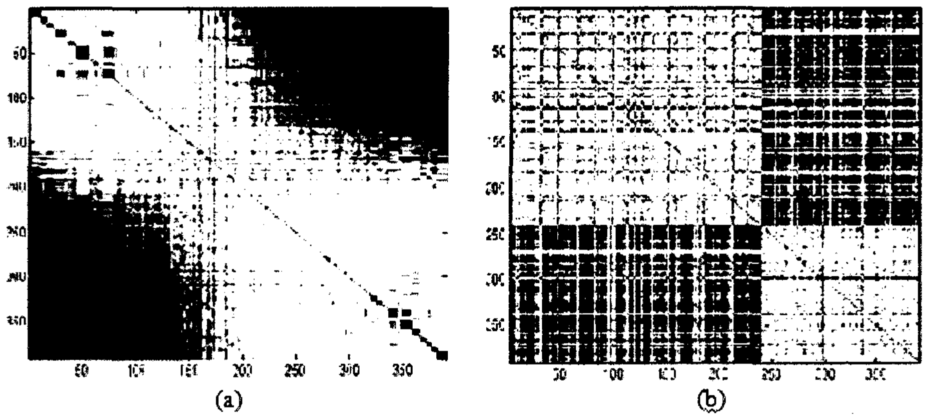


Figure 5.4: The similarity matrix for the Votes dataset. (a) The similarity matrix according to the results from the spectral clustering algorithm. (b) The similarity matrix according to the true cluster index (sorted).

For the Genes, SPECT and Lenses datasets, the performance is comparatively poorer than the other datasets. When we ordered the original similarity matrix according to the *true* cluster index, the datasets also did not show any strong clusters. Figure 5.5 (a) depicts the original similarity matrix sorted according to the *true* cluster labels. Notice that the figure does not contain any blocks on the main diagonal. The results from the spectral clustering algorithm also do not show any clear blocks with strong within cluster similarity. Therefore, the spectral clustering algorithm did not work well, as the *true* clusters according to the class labels do not correspond to the natural groupings.

The external cluster evaluation measures provide a numeric value to measure how good a clustering solution is, compared to the *true* cluster information. When a score is high, it indicates that the discovered clusters are very similar to the *true* clusters and vise versa. However, it is also necessary to inspect the results to see how the clusters are formed. For each of the datasets, we provide the solutions from the spectral clustering algorithms in the next two sections.

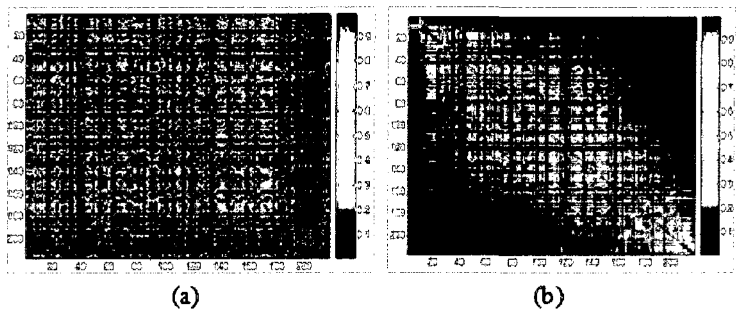


Figure 5.5: The similarity matrix for the SPECT dataset. (a) The similarity matrix according to the true cluster index (sorted). (b) The similarity matrix according to the results from the spectral clustering algorithm.

5.5.1 Clustering Results from the SM(NCut) Algorithm

Here we observe the clusters generated for each of the datasets when the *SM(NCut)* algorithm is used. Figure 5.6 depicts the hierarchical structure of the Votes dataset. The

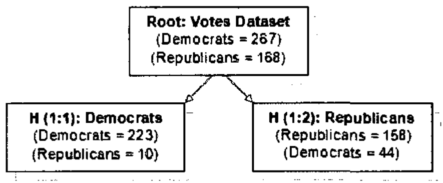


Figure 5.6: The hierarchical tree structure of the Votes dataset when SM(NCut) algorithm is applied.

tree starts from the root, which contains all the members from the dataset. Each square box represents a node in the hierarchical tree and provides the dominating *true* cluster labels and the number of members from each *true* cluster present in that particular node. Since we know that the dataset contains two *true* clusters, we only show the first level of the tree. The actual tree contains many nodes, as the algorithm continues to recursively partition the nodes until a stopping criterion is satisfied. Table 5.2 contains the *true* cluster distribution for this dataset. As the results suggest, the overall error rate for this dataset is approximately 12.41% with 54 incorrectly clustered objects out of 435, when compared to the *true* clusters. Among these 54 voters, 10 of them are the Republicans wrongly placed with the Democrats, and 44 of them are the Democrats who are wrongly clustered with the Republicans. Table 5.15 provides the original centroid along with the centroids from the partitions from the first level of the hierarchy. We notice that the centroids are almost the same except for one attribute (attribute a10), where the values differ from the original. The spectral clustering algorithms cluster the objects such that the similarity within the cluster in maximized, whereas the similarity between the clusters is minimal. Therefore, the objects that are wrongly placed according to the *true* class, may be placed in a particular cluster by the spectral clustering algorithm, as they are more similar to the other objects in that particular cluster.

Original Centroids																
a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16	True Cluster Labels
1	1	1	0	0	0	1	1	1	0	0	0	0	0	1	1	Democrat
0	1	0	1	1	1	0	0	0	1	0	1	1	1	0	1	Republican
Centroids from the first level of the hierarchy																
a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16	Dominating members
1	1	1	0	0	0	1	1	1	1	0	0	0	0	1	1	Mostly Democrat
0	1	0	1	1	1	0	0	0	0	0	1	1	1	0	1	Mostly Republicans

Table 5.15: The true cluster centroids from the original Votes dataset and the cluster centroids from the clusters generated from the SM(NCut) algorithm.

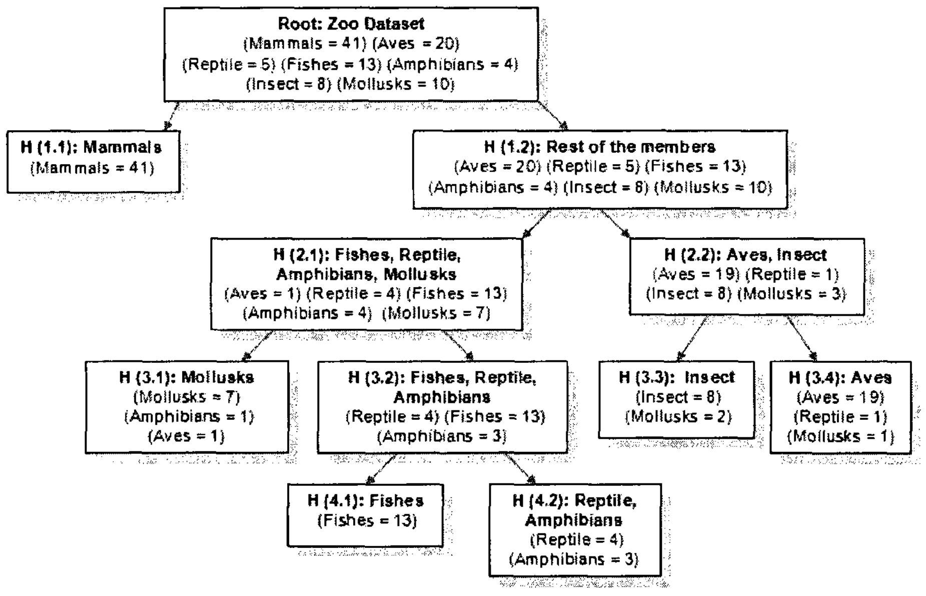

Root: Zoo Dataset
(Mammals = 41) (Aves = 20)
(Reptile = 5) (Fishes = 13) (Amphibians = 4)
(Insect = 8) (Mollusks = 10)
H (1.1): Mammals
(Mammals = 41)
H (1.2): Rest of the members
(Aves = 20) (Reptile = 5) (Fishes = 13)
(Amphibians = 4) (Insect = 8) (Mollusks = 10)
H (2.1): Fishes, Reptile, Amphibians, Mollusks
(Aves = 1) (Reptile = 4) (Fishes = 13)
(Amphibians = 4) (Mollusks = 7)
H (2.2): Aves, Insect
(Aves = 19) (Reptile = 1)
(Insect = 8) (Mollusks = 3)
H (3.1): Mollusks
(Mollusks = 7)
(Amphibians = 1)
(Aves = 1)
H (3.2): Fishes, Reptile, Amphibians
(Reptile = 4) (Fishes = 13)
(Amphibians = 3)
H (3.3): Insect
(Insect = 8)
(Mollusks = 2)
H (3.4): Aves
(Aves = 19)
(Reptile = 1)
(Mollusks = 1)
H (4.1): Fishes
(Fishes = 13)
H (4.2): Reptile, Amphibians
(Reptile = 4)
(Amphibians = 3)

Figure 5.7: The hierarchical tree structure for the Zoo dataset when SM(NCut) algorithm is applied.

The hierarchical tree of the Zoo dataset is given in Figure 5.7 and the class distribution is given in Table 5.4. The overall error rate for this dataset is 5.95% with 6 incorrectly clustered animals out of the 101 animals. Thus, almost all the animals are placed in their respective groups. We noticed that the algorithm failed to distinguish between the members from the class *Reptile* (i.e. Sea snake, Slowworm, and Tortoise) and *Amphibians* (i.e. Frog and Toad). Moreover, the six members that are incorrectly clustered are from the class *Reptile*, *Amphibians* and *Mollusks* (i.e. Starfish, Worm, Clam, and Lobster). While *Amphibians* are very similar to the class *Reptile*, on the basis that animals from both the classes usually reside in both water and ground, the *Mollusks* usually live in marine environment. This may be one of the reasons for placing some of the members from this class with the members of classes *Amphibians* and the *Reptiles*. This dataset is also an example of an imbalanced dataset with the largest class containing 41 members out of the 101 members. However, the results from manual inspection and the F-measure and G-means scores are exceptional for this dataset. Thus, the dataset is an example where the algorithm works very well for the imbalanced dataset. The spectral clustering algorithm separates the data by maximizing the within cluster similarity and minimizing the between cluster similarity. Therefore, it does not make any assumption on the size of the clusters. As such, if the data is imbalanced, but the within cluster similarity is high and between cluster similarity is very low, it should find the clusters.

For the Balloon dataset the overall error rate is 36.84%, with 7 incorrectly clustered objects out of the 19 instances. The Lenses dataset contains 24 instances, and only 10 of them are correctly clustered with an error rate of 58.33%. The error rate for the SPECT dataset is 44.63%. The dataset contains 242 instances and only 134 of them are correctly clustered. The hierarchical tree, similar to the one given for the Zoo and Votes datasets, is depicted in Figure 5.8. As we see, in both the partitions the dominating class includes the patients with an abnormal heart condition. We look into the entire tree (the picture depicts only the first level) to see whether at some point in the tree we have the patients with a normal heart condition grouped together. However, we fail to find any such node in the tree. In contrast to the Zoo dataset, the SPECT dataset is an example when the spectral clustering algorithm failed to find correct clusters for the imbalanced datasets. As depicted in Figure 5.5, the original dataset does not show well-defined structures. The situation for the Genes dataset is also similar to the SPECT dataset. Only 64.56% members from the *Nucleus* class and 49.03% from the *Cytoplasm* class are correctly clustered. In most of the cases, the clusters are shared with the members from different classes, which may be the reason for having a very low F-measure and G-means values.

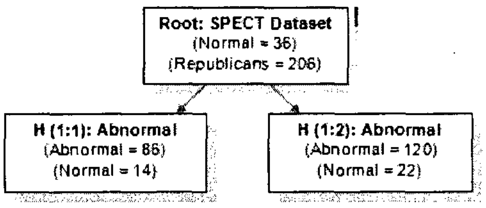


Figure 5.8: The hierarchical tree structure for the SPECT dataset when SM(NCut) algorithm is applied.

5.5.2 Clustering Results from the NJW(K-means) Algorithm

In this section, we discuss the clustering results from the *NJW(K-means)* algorithm. Table 5.14 provides the best F-measure scores for each of the datasets. The performance of the Votes dataset is the same for both the algorithms. However, the performance of the Zoo dataset is degraded for the *NJW(K-means)* algorithm. We notice that the algorithm splits the larger *true* clusters into smaller clusters, rather than separating the *true* clusters of smaller sizes into individual groups as the number of clusters increases. One possible reason for this, is that the algorithm uses the K-means algorithm to discover the clusters from the k eigenvectors and the K-means algorithm has the tendency to discover clusters of uniform sizes [79]. The Zoo dataset is an imbalanced dataset, as one of the *true* clusters contains relatively more objects than the others. Therefore, as the number of clusters increases, the algorithm splits the largest *true* cluster into several smaller groups, resulting in lower F-measure scores. The number of *true* clusters for the Zoo dataset is 7. Below we provide each of the cases when k is in between [2..7] to show how the clusters are formed as the number of k increases.

$k = 2$: *Cluster 1* includes all the members from *Class 1* which represents the animals from class *Mammals* from the animal kingdom. *Cluster 2* contains the rest of the members. The overall error rate is 39.60%.

$k = 3$: *Cluster 1* comprises the members from class *Mammals*, *Cluster 2* contains mostly members from *Class 2* which denotes the class *Aves* from the animal kingdom. *Cluster 3* contains the rest of the animals. The overall error rate is 27.72% and this is due to the members of *Cluster 3*.

$k = 4$: In this case, apart from separating the members from classes *Mammals* (*Cluster 1*) and *Aves* (*Cluster 2*), the members from class *Insect* are also accurately grouped separately in *Cluster 3*. The rest of the members are placed in *Cluster 4*. The error rate in this case is 18.81%.

$k = 5$: The class *Mammals*, which is the largest with 41 members, is split into two separate clusters (*Cluster1* and *Cluster 2*). *Cluster 3*, and *Cluster 4* mostly contain members from class *Aves* and *Insect* including a number of members from other classes. *Cluster 5* contains the members from class *Reptiles*, *Fishes*, *Amphibians* and *Mollusks*. The error rate is 37.63%.

$k = 6$: The class *Mammals* is split into three clusters this time. *Cluster 4* mostly comprises the members from class *Aves* and *Insect* and *Cluster 5* contains the members from class *Fishes*. *Cluster 6* includes the members from class *Reptiles*, *Fishes* and *Mollusks*. We notice that one common characteristic between the members from class *Aves* and the members from class *Insect* is that they are all airborne. In contrast, the common characteristics for the members from classes *Reptiles*, *Fishes* and *Mollusks* are that they are mostly aquatic and predator. This may be one of the reasons for which they are placed together.

$k = 7$: In this case, the class *Mammals* is again split into two different clusters. *Cluster 3*, *Cluster 4*, *Cluster 5*, and *Cluster 6* mostly comprise the members of classes *Aves*, *Fishes*, *Insects* and *Mollusks*, respectively. *Cluster 7* is shared by the members of classes *Reptiles* and *Amphibians*.

Thus, the results show that as the number of clusters increase, the class with the largest number of members is split into smaller clusters. In addition, the members from class *Reptiles* and class *Amphibians* almost always share a cluster. The reason for this is that the members from both the classes have some characteristics in common (e.g. they all reside in both water and ground) and when clustering is performed, the members from these classes are placed in one cluster as the objects have similar attribute values.

The Votes dataset contains 435 instances and among them 87.82% objects are correctly clustered when $k = 2$. Figure 5.9 shows the visual representation of the clusters of the Votes dataset when K-means is performed on the largest two eigenvectors. In this case, the centroids of the clusters (when $k = 2$) are also similar to the ones given in Figure 5.15 for the first level of the hierarchical structure, when the *SM(NCut)* algorithm is applied to the dataset. For the Balloon dataset, the percentage of incorrectly clustered instances is 36.84%, with 7 (out of 19) objects wrongly placed in different clusters. The Lenses dataset also has a high error rate of 62.56%, with 15 out of 24 objects wrongly placed in different clusters (when $k = 3$). Similarly to the Zoo dataset, in this case the largest *true* cluster is also divided into several smaller clusters. The SPECT dataset also has a high error rate of 46.69%. The evaluation scores from Table 5.10 and Table 5.11

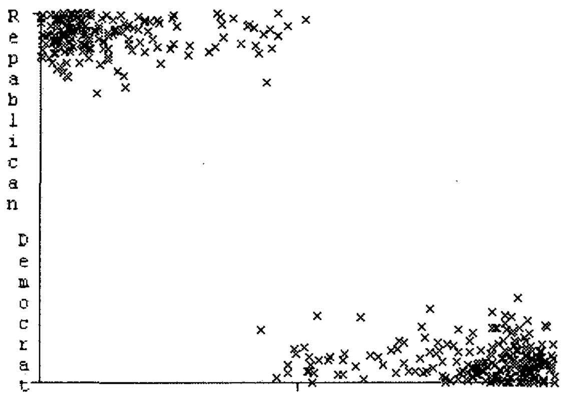


Figure 5.9: Cluster visualization for the Votes dataset when K-means is performed on the two largest eigenvectors.

show that the Genes dataset also performed poorly. Recall that the dataset determines the localization of each of the genes based on a number of functions in which they participate. When $k = 2$, we find that one of the clusters is mostly comprised of members from class *Nucleus*, which is the largest group with 539 members. The cluster identifies 345 genes from this class. The second cluster includes the rest of the objects including the remaining members from class *Nucleus*. The total error rate (when $k = 2$) is 53.70%. When $k = 3$, the error rate is 52.49%. In this case, two of the clusters contained members mostly from class *Nucleus* and class *Cytoplasm*, and the third cluster contain the rest of the members. When k is set to 8 (the number of *true* clusters), the algorithm subdivided the largest *true* cluster into several smaller clusters. The rest of the clusters contain the members from various classes.

5.6 Chapter Summary

Datasets with binary variables are composed of two states (i.e. presence-absence, yes-no, 1-0). There are a number of coefficients that compute the similarity between the binary objects. This chapter discussed the performance of six such similarity measures that are suitable for binary datasets. To compare and evaluate the performance of the coefficients, we first performed the spectral cluster analysis on six binary datasets. We then evaluated the results from cluster analysis using a number of external evaluation measures. Next, the similarity coefficients were compared using the results obtained from the external evaluation measures.

Our results indicated that the *Russell and Rao* (*RAR*) similarity coefficient performed poorer than the rest of the five similarity coefficients. For the majority of the datasets, this coefficient scored the lowest. Recall from Chapter 2 that, this coefficient is the only coefficient that considers the negative matches in the denominator but not in the numerator. Therefore, when the spectral clustering algorithm is applied on the binary datasets, this coefficient may not be a proper choice. The three coefficients (*Czekanowski* (*CZE*), *Jaccard* (*JAC*), and *Sokal and Sneath* (*SAS*)) that neglect the negative matches, give equal weight to the positive matches, and vary mostly on the weight given on the unmatched pairs, did not show any particular pattern within their performance. We also observed that the performance of the coefficients depends heavily on the datasets. Therefore, we analyzed the datasets to determine the possible reasons that caused the differences in the performance of each coefficient. Our analysis showed that the issues that affect the performance most are, the weight given on the positive and negative matches, the amount of 0's and 1's in the dataset, and the amount of unmatched pairs. We also compared the performance of the splitting methods of the *SM(NCut)* algorithm. The results showed that the *Split Zero* and *Split Mean* methods often performed well. In contrast, the *Split NCut1* and *Split NCut2* methods often produced imbalanced clusters. Therefore, based on our results, the *Split Zero* or *Split Mean* methods may be a preferred selection for the *SM(NCut)* algorithm. The results from the two spectral clustering algorithms are also comparable. Based on the evaluation scores, we noticed that the *SM(NCut)* algorithm outperformed the *NJW(K-means)* algorithm most of the time. One possible reason may be the use of the larger eigenvectors in the *NJW(K-means)* algorithm which might have added noise to the solution. However, this needs further research and more in depth experimentation to establish a strong conclusion.

In the next chapter, we perform a similar analysis for the similarity coefficients suitable for the datasets with mixed variables.

Chapter 6

Result Analysis - Mixed Data

In practical applications, datasets often combine attributes of mixed types. As mentioned in Chapter 1, the most common types are binary, numeric, and nominal attributes. Most of the traditional cluster analysis algorithms are suitable for datasets of a particular type (numeric or nominal). Nevertheless, the spectral clustering algorithm is capable of handling attributes of any type so long as a similarity matrix is constructed from the dataset. In Chapter 3, we discussed a number of proximity coefficients particularly suitable for datasets where the attributes are mixed. Using these coefficients, we construct the similarity matrices as discussed in Chapter 4 and use them as an input to two different versions of the spectral clustering algorithm. In this chapter we evaluate the results obtained from these experiments to compare the performance of the proximity measures. The chapter begins in Section 6.1 with a description of the datasets used for the experiments. Next, in Section 6.2, we provide the quantitative results from the spectral clustering algorithms obtained from the external evaluation measures. This section is followed by Section 6.3, where we perform an evaluation on the results. We then discuss the clusters formed from the spectral clustering algorithms in Section 7.4. We conclude the chapter with a summary in Section 6.5.

6.1 Mixed Datasets

In this section we describe each of the datasets used in our experiments to compare the performance of the proximity measures for mixed data. All the datasets are obtained from the UCI repository [3]. The datasets are selected to have different characteristics. First, they all vary in size. For instance, the largest dataset contains 690 (CRX dataset) instances and the smallest dataset includes 24 (Lenses dataset) instances. Second, they are based on real-world problems. For example, the Hepatitis dataset is a clinical dataset, whereas, the CRX dataset is a credit card application dataset. The datasets used in this study also contain the *true* cluster information which is used for the assessment of the results. The datasets also vary in the number and size of the *true* clusters. For instance, the Hepatitis dataset is an imbalanced dataset where the size of one of the *true* clusters is comparatively larger than the others. However, the *true* cluster information is strictly used for the cluster evaluation purpose. We provide a brief description for each of the datasets below. We also provide the *true* cluster information for each of the datasets and refer to them as either *true* cluster or *class* in the rest of the chapter.

Automobile (Auto) Dataset: The dataset comprises 205 instances and 25 attributes. There are 15 numeric attributes, 6 nominal attributes, and 4 binary attributes associated with each instance. According to the information provided with the dataset, a number of these attributes may be considered and used as the *true* cluster attribute. We use the attribute that represents the risk rating (also known as the *Symboling*) as the class attribute, which indicates the degree to which the automobile is more risky than its price. This attribute's values range from $+3$ to -2. A positive value indicates that the vehicle is risky, whereas a negative value indicates that the automobile is safe. The dataset contains 41 missing values, which is 1.14% of the entire attribute values, and these are replaced using WEKA - an open source data mining software [76], as part of the pre-processing task. WEKA replaces the missing values by modes if the attribute is nominal or binary and by means if the attribute is numeric [76]. The *true* cluster distribution for this dataset is given in Table 6.1.

True Clusters	Number of Members	True Clusters	Number of Members
1	54	0	67
2	32	-1	22
3	27	-2	3

Table 6.1: The true cluster distribution of the Automobile (Auto) dataset.

Credit Approval (CRX) Dataset: The instances in this dataset represent various attributes of credit card applications and the class labels consider whether or not the credit card application is approved by the company. For this dataset, all attribute names and values are changed to meaningless symbols to protect the confidentiality of the data [3]. There are 6 numeric attributes, 5 nominal attributes, and 4 binary attributes. There are 690 instances in total and 67 missing values in the entire dataset. The missing values are replaced by WEKA. The *true* cluster distribution for this dataset is given in Table 6.2.

True Cluster	Number of Members
+ (Approved)	307
-(Not Approved)	383

Table 6.2: The true cluster distribution of the CRX dataset.

Dermatology Dataset: The Dermatology dataset determines the type of a disease called Eryhemato-Squamous from various attributes. The attributes include the clinical as well as the histopathological features of the patients suffering from the disease. According to the dataset, the disease is categorized into six different types and the type attribute is used as the *true* cluster attribute (Table 6.3). There are 366 instances and 33 attributes (32 numeric attributes and 1 binary attribute) in the dataset. However, one of the attributes has many missing values and is therefore discarded for this task. The rest of the dataset has 8 missing values.

Class	Number of Members
1 (Psoriasis)	112
2 (Seboreic Dermatitis)	61
3 (Lichen Planus)	72
4 (Pityriasis Rosea)	49
5 (Cronic Dermatitis)	52
6 (Pityriasis Rubra Pilaris)	20

Table 6.3: The true cluster distribution of the Dermatology dataset.

Hepatitis Dataset: The Hepatitis dataset predicts whether a patient with hepatitis will survive or not by examining various clinical attributes collected from the patients affected by the disease. The dataset contains 155 examples and there are 20 attributes including the class attribute. There are 5 numeric attributes, 1 nominal attribute, and 13 binary attributes in the dataset. The dataset is an example of an imbalanced dataset, as there are only 32 examples (out of 155) that fall into the category in which the patients will most probably die from the disease. There are 162 missing values (5.2% of the total attribute values) in the dataset. The class distribution is given in Table 6.4.

Class	Number of Members
Live	123
Die	32

Table 6.4: The true cluster distribution of the Hepatitis dataset.

Post-Operative Dataset: This dataset provides information to determine the place to which the post-operative patients should be sent. The decision is made by examining various attributes, including the body temperature, blood pressure, amongst others. The options are: I - patients are transferred to the intensive care unit, S - the patients are sent home, and A - the patients are sent to the general hospital floor. There are 90 instances in the dataset and 8 attributes (5 Nominal, 2 binary, and 1 Numeric attribute) excluding the class attribute. Only 3 of the values are missing in the dataset and are replaced using WEKA. Table 6.5 contains the class distribution for this dataset.

Class	Number of Members
A	64
S	24
I	2

Table 6.5: The true cluster distribution of the Post-Operative dataset.

Soybean Dataset: The large Soybean dataset initially contained 307 instances and 19 classes. However, only 15 classes are used in most of the prior studies. The remaining four classes either involve very few examples, or contain mostly missing values, and thus, are discarded in this study also. We have a total of 290 instances and 15 classes to consider. There are 35 attributes (18 nominal and 17 binary attributes) associated with each instance. There are also 390 missing values and these are replaced using WEKA. The classes denote various diseases of the soybean plant that are diagnosed by analyzing the attribute values. The class distribution is given in Table 6.6.

Class	Number of Members	Class	Number of Members
Alternarialeaf-spot	40	Downy-mildew	10
Anthracnose	20	Frog-eye-leaf-spot	40
Bacterial-blight	10	Phyllosticta-leaf-spot	10
Bacterial-pustule	10	Phytophthora-rot	40
Brown-spot	40	Powdery-mildew	10
Brown-stem-rot	20	Purple-seed-stain	10
Charcoal-rot	10	Rhizoctonia-root-rot	10
Diaporthe-stem-canker	10		

Table 6.6: The true cluster distribution of the Soybean dataset.

A summary of the datasets as discussed above is given in Table 6.7.

Dataset	Instances	True Clusters	Attributes	Notes
Automobile	205	6	25	Imbalanced, 1.14% missing values
CRX	690	2	15	0.6% missing values
Dermatology	366	6	33	Imbalanced, 0.06% missing values
Hepatitis	155	2	19	Imbalanced, 5.2% missing values
Post Operative	90	3	8	Imbalanced, 0.37% missing values
Soybean	290	15	35	3.7% missing values

Table 6.7: Summary of the datasets with the mixed data types. Given in the table (from left) the name of the datasets, the number of instances, the number of *true* clusters, the number of attributes, and several additional information about the datasets.

6.2 Comparison of Proximity Measures

In this section we provide the quantitative results from our experiments on datasets with mixed variable type. To compare the performance of the proximity measures for datasets with mixed variables, we first apply the spectral clustering algorithms on each of the datasets. Recall from Chapter 2, that all of the experiments are performed on two different versions of spectral clustering algorithm: 1) Normalized Cut spectral clustering algorithm (SM(NCut)) and 2) Spectral algorithm with K-means (NJW(K-means)). The results from these two algorithms are assessed using the external evaluation measures as discussed in Chapter 4. It is important to note that the class labels or the *true* clusters are only used for the assessment of the results and that these attributes are not used during clustering. The performance of the proximity measures are then compared and evaluated by analyzing the scores from external evaluation measures. In Chapter 3, we discussed two existing proximity coefficients suitable for these types of datasets. The coefficients are *Gower's General Dissimilarity Coefficient (GOWER)* and *Laflin's General Coefficient (LAFLIN)*.

6.2.1 Results from SM(NCut) Spectral Clustering Algorithm

Table 6.8 provides the F-measure and G-means scores when the *SM(NCut)* algorithm is used as the cluster analysis method. Recall from Chapter 4 that the Entropy measure may not be suitable for evaluating hierarchical clustering solutions. Therefore, we do not consider this measure to evaluate the clustering results from the *SM(NCut)* algorithm. As mentioned in Chapter 4, there are five splitting methods that may be used to discover the partitions. These splitting methods are *Split Zero*, *Split Mean*, *Split Median*, *Split NCut1*, and *Split NCut2*. In Chapter 5 we observed that the performance of the *Split Zero* method and the *Split Mean* method is better than the *Split Median*, *Split NCut1*, and *Split NCut2* methods. The *Split Median* method always produces two equal sized partitions, giving emphasis to the size of the clusters, and may not be suitable when the sizes of the *true* clusters are not the same. The *Split NCut1* and *Split NCut2* methods perform a search over the eigenspace, and in several cases produce imbalanced partitions. Figure 6.2 depicts the average F-measure scores for each of the datasets. Recall from Chapter 4, that a higher F-measure and G-means value indicates a better result. Below we provide the statistics for each of the datasets individually.

Automobile Dataset

For this dataset, the *GOWER* coefficient performs better than the *LAFLIN* coefficient. The average F-measure score for the *GOWER* coefficient is 0.48 and the average G-means score is 0.51. The *LAFLIN* coefficient scored 0.46 for F-measure and 0.49 for G-means. The *Split Zero* and *Split Mean* methods both scored 0.46 for F-measure. The lowest F-measure score is 0.44, whereas the lowest G-means score is 0.48. The average F-measure scores fall between 0.46 and 0.48. The average G-means scores fall in the range $[0.49 - 0.51]$. Therefore, both the F-measure and G-means scores indicate that the

	F-measure											
	Automobile		CRX		Dermatology		Hepatitis		Post Operative		Soybean	
	Gower	Laflin	Gower	Laflin	Gower	Laflin	Gower	Laflin	Gower	Laflin	Gower	Laflin
Split Zero	0.46	0.44	0.76	0.79	0.85	0.87	0.71	0.73	0.56	0.56	0.70	0.70
Split Mean	0.46	0.46	0.76	0.79	0.84	0.82	0.73	0.75	0.56	0.56	0.70	0.70
Split Median	0.46	0.46	0.76	0.79	0.82	0.70	0.70	0.69	0.55	0.57	0.64	0.64
Split NCut1	0.49	0.44	0.78	0.72	0.83	0.70	0.74	0.73	0.67	0.67	0.64	0.56
Split NCut2	0.51	0.48	0.79	0.71	0.85	0.82	0.80	0.80	0.65	0.66	0.63	0.66
	G-means											
	Automobile		CRX		Dermatology		Hepatitis		Post Operative		Soybean	
	Gower	Laflin	Gower	Laflin	Gower	Laflin	Gower	Laflin	Gower	Laflin	Gower	Laflin
Split Zero	0.49	0.48	0.76	0.79	0.86	0.87	0.74	0.75	0.57	0.57	0.72	0.72
Split Mean	0.50	0.49	0.76	0.79	0.85	0.83	0.75	0.77	0.58	0.57	0.72	0.72
Split Median	0.48	0.49	0.76	0.79	0.83	0.73	0.73	0.72	0.56	0.58	0.66	0.66
Split NCut1	0.53	0.50	0.78	0.73	0.84	0.72	0.75	0.75	0.67	0.67	0.67	0.59
Split NCut2	0.53	0.50	0.80	0.72	0.86	0.83	0.81	0.81	0.67	0.66	0.65	0.68

Table 6.8: The F-measure and G-means scores from the SM(NCut) algorithm for the datasets with mixed variable type.

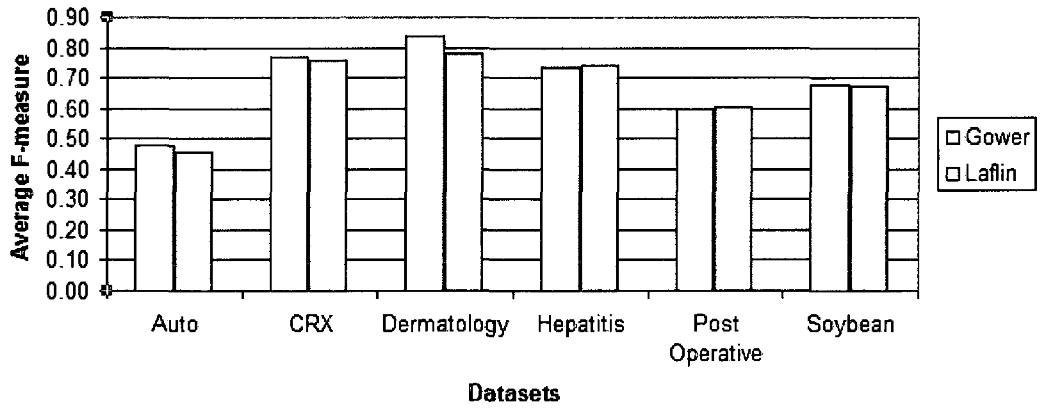


Figure 6.1: The average F-measure scores for the mixed datasets.

difference between the coefficients is very low.

CRX Dataset

The average F-measure scores indicate that the CRX dataset performs slightly better for the *GOWER* coefficient than for the *LAFLIN* coefficient. However, the *LAFLIN* coefficient performs better for the *Split Zero*, *Split Mean*, and *Split Median* splitting methods, whereas the *GOWER* coefficient performs better for the *Split NCut1* and *Split NCut2* splitting methods. As mentioned, the *Split Zero* and *Split Mean* splitting methods often provide robust results. Therefore, we consider the *LAFLIN* coefficient for this dataset. The highest score by the *LAFLIN* coefficient is 0.79 and the lowest score is 0.76 (the

GOWER coefficient) for both, F-measure and G-means. The average F-measure and G-means scores fall between the range [0.76 – 0.77] and also indicate that the difference is very low.

Dermatology Dataset

The Dermatology dataset performs well for the *GOWER* coefficient. The average F-measure and G-means scores for the *GOWER* coefficient are 0.84 and 0.85, respectively. For the *LAFLIN* coefficient, the scores are 0.78 (F-measure) and 0.80 (G-means). Therefore, the overall range for the average F-measure is [0.78 – 0.84]. The average G-means score falls in between 0.80 and 0.84.

Hepatitis Dataset

The average F-measure and G-means for this dataset are 0.74 and 0.76, respectively. The results indicate that the average scores are the same for both the coefficients. However, the F-measure scores for the *LAFLIN* coefficient when the *Split Zero* and *Split Mean* splitting methods are used are slightly higher than the scores from the *GOWER* coefficient. This indicates that the performance of the *LAFLIN* coefficient is slightly better than the *GOWER* coefficient. For instance, the *LAFLIN* coefficient scored 0.73 (G-means: 0.75) and the *GOWER* coefficient scored 0.71 (G-means: 0.74) when the *Split Zero* is used as the splitting method.

Post-Operative Dataset

For this dataset, the average F-measure and G-means scores are same for both of the coefficients. The average F-measure score is 0.60 and the average G-means score is 0.61. This also indicates that the performance of both the coefficients is the same for this dataset.

Soybean Dataset

The average F-measure and G-means scores suggest that the *GOWER* coefficient performs better for this dataset. However, we notice that for *Split Zero*, *Split Mean*, and *Split Median*, both F-measure and G-means scores are similar. The highest F-measure score is 0.70 and the highest G-means score is 0.72. The lowest F-measure score is 0.64 and the lowest G-means score is 0.66. The average F-measure score varies from 0.65 to 0.66 and the average G-means scores varies from 0.67 to 0.68, indicating that the average difference between the *GOWER* coefficient and the *LAFLIN* coefficient is very low.

From the discussion above, we observe that the *GOWER* coefficient performs slightly better than the *LAFLIN* coefficient when the average F-measure and G-means scores are considered. In four (i.e. Automobile, CRX, Dermatology, and Soybean dataset) out of six datasets, the average scores are higher for the *GOWER* coefficient than the *LAFLIN* coefficient. After considering the splitting methods, two datasets (i.e. Automobile and Dermatology dataset) score well for the *GOWER* coefficient. In contrast, the *LAFLIN* coefficient performs slightly better for the CRX and Hepatitis dataset when splitting methods are considered. The performance of both the coefficients is the same for the Post-Operative and Soybean dataset when *Split Zero* and *Split Mean* are used as the splitting points.

6.2.2 Results from NJW(K-means) Spectral Clustering Algorithm

Table 6.9 provides the evaluation scores from the *NJW(K-means)* spectral clustering algorithm. Recall from Chapter 2, that this algorithm requires the number of clusters (k) as input, in addition to the similarity matrix. Table 6.9 provides the results when k is set to the number of *true* clusters (which is available to us). The number of *true* clusters for each dataset is given in Table 6.7. Figure 6.2 illustrates the F-measure scores for each dataset. Recall from Chapter 3, that the higher the F-measure and G-means scores, the better the solution. In contrast, the lower the Entropy scores, the better. In several situations, results may vary from F-measure and G-means to Entropy. According to [78], Entropy tends to favor results with uniform cluster sizes. Therefore, in such situations, we rely more heavily on the scores from F-measure and G-means.

	Automobile		CRX		Dermatology		Hepatitis		Post Operative		Soybean	
	Gower	Laflin	Gower	Laflin	Gower	Laflin	Gower	Laflin	Gower	Laflin	Gower	Laflin
F-measure	0.47	0.45	0.76	0.80	0.84	0.82	0.71	0.74	0.52	0.47	0.57	0.53
G-means	0.48	0.46	0.76	0.80	0.86	0.84	0.74	0.76	0.53	0.49	0.61	0.56
Entropy	1.12	1.09	0.54	0.49	0.24	0.31	0.41	0.40	0.58	0.58	0.84	0.92

Table 6.9: The external evaluation scores (F-measure, G-means, and Entropy) from the NJW(K-means) algorithm for the datasets with mixed variables when k (the number of clusters) is set to the *true* cluster number.

Automobile Dataset

For this dataset, k is set to 6, the number of *true* clusters. According to the F-measure and G-means scores, the *GOWER* coefficient performs well for this dataset. The highest scores are 0.47 for F-measure and 0.48 for G-means. The lowest scores are 0.45 and 0.46, respectively. The lowest Entropy is 1.09 and the highest Entropy is 1.12. The difference between the highest and lowest score is 0.02, which indicates that the difference is very low for the coefficients.

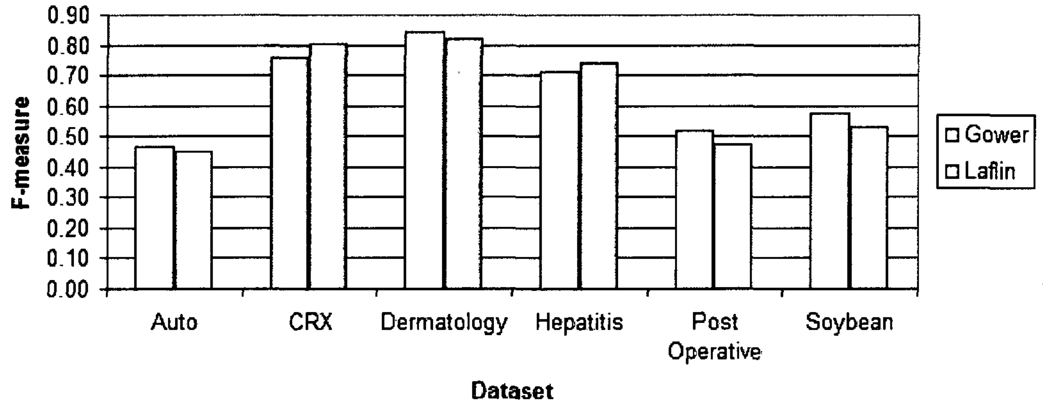


Figure 6.2: The F-measure scores for the mixed datasets for NJW(K-means) algorithm.

CRX Dataset

The evaluation scores indicate that, for this dataset, the *LAFLIN* coefficient performs better than the *GOWER* coefficient (when $k = 2$). The highest F-measure and G-means score is 0.80 and the lowest score is 0.76, for both the evaluation measures. The average difference between the highest and lowest scores is 0.04, which, is again, very low. The lowest Entropy is 0.49 and the highest Entropy is 0.54.

Dermatology Dataset

The *GOWER* coefficient performs well for the Dermatology dataset when k is set to 6, which is the number of *true* clusters. For this dataset the highest F-measure score is 0.84, whereas the lowest score is 0.82. This again shows that the difference is very low between the scores. The highest G-means score as scored by the *GOWER* coefficient is 0.86 and the lowest score as scored by the *LAFLIN* coefficient is 0.84. The Entropy value for the *GOWER* coefficient is 0.24 and 0.31 for the *LAFLIN* coefficient.

Hepatitis Dataset

For Hepatitis dataset, the *LAFLIN* coefficient performs better than the *GOWER* coefficient when $k = 2$. The F-measure scores are 0.74 for the *LAFLIN* coefficient and 0.71 for the *GOWER* coefficient. The G-means value for the *LAFLIN* coefficient is 0.76 and 0.74 for the *GOWER* coefficient. The difference between the highest and lowest F-measure score is 0.03 and 0.02 for G-means. Low differences indicate that the results from the two coefficients are not very different from one another.

Post-Operative Dataset

The Post-Operative dataset scores the highest F-measure and G-means scores for the *GOWER* coefficient. The F-measure scores are 0.52 and 0.47 for the GOWER coefficient and the *LAFLIN* coefficient, respectively. The G-means scores are 0.53 and 0.49, accordingly. Both of the coefficients achieve the same Entropy score (0.58). The number of *true* clusters for this dataset is 3 and this value is used as the number of clusters.

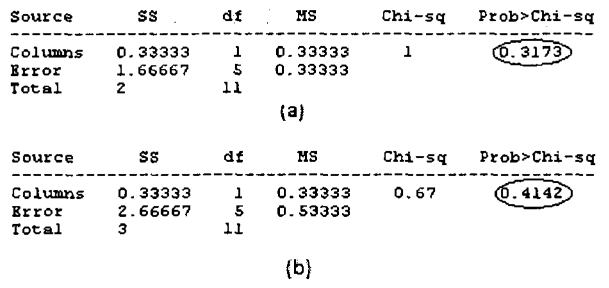
Soybean Dataset

The *GOWER* coefficient achieves the highest F-measure and G-means scores and the lowest Entropy score for the Soybean dataset when $k = 15$. The highest and lowest F-measure scores are 0.57 for the *GOWER* coefficient and 0.53 for the *LAFLIN* coefficient, respectively. The G-means score for the *GOWER* coefficient is 0.61 and 0.84 for the Entropy. The G-means and Entropy scores for the *LAFLIN* coefficient are 0.56 and 0.92, respectively.

The above discussion indicates that the *GOWER* coefficient scored the highest F-measure and G-means scores in four out of six datasets. The datasets are Automobile, Dermatology, Post-Operative, and Soybean. The *LAFLIN* coefficient scored the highest scores for the CRX and Hepatitis dataset. We also observe that, for all the datasets the difference between the scores is very low. The average difference between the highest and the lowest F-measure and G-means scores is 0.03 for both of the evaluation scores. In the next section, we perform an evaluation on the results obtained from this section.

6.3 Result Evaluation

In the previous section, we discussed the quantitative results from the *SM(NCut)* and *NJW(K-means)* spectral clustering algorithms. The results from the external evaluation scores from these two algorithms show similar trends. For both the algorithms, the *GOWER* coefficient frequently achieved the highest scores. The datasets that performed well for the *GOWER* coefficient are Automobile, Dermatology, Post-Operative, and Soybean, whereas CRX and Hepatitis performed well for the *LAFLIN* coefficient. The results also indicate that the difference between the performances of the two coefficients is very low. On average the difference is 0.02 for the *SM(NCut)* algorithm and 0.03 for the *NJW(K-means)* algorithm. Figure 6.3 contains the results from the Friedman test. The *p-value* for the results from the *SM(NCut)* algorithm is 0.3173. The *p-value* is 0.4142 for the results from the *NJW(K-means)* algorithm. Recall from Chapter 4 that a *p-value* greater than 0.05 indicates that the difference between the performance of the proximity measures is not statistically significant on our datasets. We analyze the coefficients to determine the relationship between them. The function for the *GOWER*



```
Source      SS        df    MS         Chi-sq    Prob>Chi-sq
--------------------------------------------------------------
Columns    0.33333     1   0.33333     1          0.3173
Error      1.66667     5   0.33333
Total      2          11
                      (a)

Source      SS        df    MS         Chi-sq    Prob>Chi-sq
--------------------------------------------------------------
Columns    0.33333     1   0.33333     0.67       0.4142
Error      2.66667     5   0.53333
Total      3          11

                      (b)
```

Figure 6.3: Results from the Friedman test when applied on the results from (a) the SM(NCut) algorithm and (b) the NJW(K-means) algorithm.

coefficient as given in Equation 3.18 is:

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}} \tag{6.1}$$

Where $\delta_{ij}^{(f)}$ is an indicator variable associated with each of the variables present in the dataset and $d_{ij}^{(f)}$ is the distance or dissimilarity calculated for each variable for objects i and j. According to the definition given in Chapter 3, $\delta_{ij} = 0$, for asymmetric binary variables and for all the other types $\delta_{ij} = 1$. In our datasets, all of the attributes are numeric, nominal, or symmetric binary. Therefore, δ in the denominator of Equation 6.1 represents the total number of variables in the dataset. Also, recall from Chapter 3, that the *GOWER* coefficient is a dissimilarity measure, where the dissimilarity between the two objects, i and j, falls in between 0 and 1. Equation 6.1 is converted into a similarity measure using the Equation 3.1. Therefore, the equation for the *GOWER similarity coefficient* is:

$$s(i,j) = 1 - \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}} \tag{6.2}$$

Let $N = \sum_{f=1}^{p} \delta_{ij}^{(f)}$, be the total number of attributes and for each attribute $\delta_{ij} = 1$, then the Equation 6.2 becomes:

$$s(i,j) = 1 - \frac{\sum_{f=1}^{p} 1 * d_{ij}^{(f)}}{N} = \frac{N - \sum_{f=1}^{p} d_{ij}^{(f)}}{N} \tag{6.3}$$

The function for the *LAFLIN* similarity coefficient is given in Equation 6.4. In this case s_i is the total similarity value of attribute type i, and N_i is the total number of variables of attribute type i. In our datasets, the attribute types are numeric (N_1 and

s_1), nominal (N_2 and s_2), and symmetric binary (N_3 and s_3).

$$s(i,j) = \frac{N_1.s_1 + N_2.s_2 + N_3.s_3}{N_1 + N_2 + N_3} \tag{6.4}$$

Equation 6.4 may be rewritten as:

$$\begin{aligned} s(i,j) &= \frac{N_1 + N_2 + N_3 - N_1 - N_2 - N_3 + N_1.s_1 + N_2.s_2 + N_3.s_3}{N_1 + N_2 + N_3} \qquad (6.5)\\ &= \frac{(N_1 + N_2 + N_3) - (N_1 - N_1.s_1 + N_2 - N_2.s_2 + N_3 - N_3.s_3)}{N_1 + N_2 + N_3} \qquad (6.6)\end{aligned}$$

The denominator in Equation 6.6 is the total number of attributes in the dataset, which we previously denoted as N. Then, $N = N_1 + N_2 + N_3$ and Equation 6.6 becomes,

$$\begin{aligned} s(i,j) &= \frac{N - (N_1 - N_1.s_1 + N_2 - N_2.s_2 + N_3 - N_3.s_3)}{N} \qquad (6.7)\\ &= \frac{N - (N_1(1 - s_1) + N_2(1 - s_2) + N_3(1 - s_3))}{N} \qquad (6.8)\end{aligned}$$

At this point, Equation 6.3 (*GOWER* similarity coefficient) and Equation 6.8 (*LAFLIN* similarity coefficient), have similar patterns. Both of the equations have the same denominator. They differ only in terms of the numerator, which is $\sum_{f=1}^{p} d_{ij}^{(f)}$ for the *GOWER* similarity coefficient and $N_1(1-s_1) + N_2(1-s_2) + N_3(1-s_3)$ for the *LAFLIN* similarity coefficient. As mentioned above, d_{ij} is the distance or dissimilarity between the two objects i and j, whereas $(1 - s_i)$ is also a dissimilarity measure (s_i is the similarity).

Recall from Chapter 3, that the distance between the nominal or symmetric binary attributes for the *GOWER* coefficient is calculated as: if $x_{if} = x_{jf}$ then $d_{ij}^{(f)} = 0$; otherwise $d_{ij}^{(f)} = 1$. This implies that the distance for all the nominal or binary attributes is the total number of unmatched pairs within the two objects. For the *LAFLIN* coefficient, the dissimilarity for nominal or binary attributes are calculated as $d(i,j) = \frac{p-m}{p}$. Here, p is the total number of variables of type nominal or binary, and m is the number of variables for which i and j have the same value. Notice that $(p - m)$ is the total number of unmatched pairs for binary or nominal attributes. This implies that the term $N_2(1 - s_2) + N_3(1 - s_3)$ in $N_1(1 - s_1) + N_2(1 - s_2) + N_3(1 - s_3)$ is the same as the total number of unmatched pairs. Here, $N_2 = p_{nominal}$ and $N_3 = p_{binary}$, the total number of nominal and binary attributes, respectively, and $(1 - s_2)$ and $(1 - s_3)$ are the total dissimilarity for the nominal and binary attributes, respectively. Thus, $N_2(1 - s_2) = N_2 * \frac{p_{nominal} - m_{nominal}}{p_{nominal}}$ and $N_3(1 - s_3) = N_3 * \frac{p_{binary} - m_{binary}}{p_{binary}}$. Therefore, the binary and nominal attributes are handled similarly by both of the similarity coefficients.

This also implies that the only difference in the equations occurs due to the functions selected for the numeric attributes which are handled differently by the two coefficients. This is one of the reasons that the difference between the performances of both of the coefficients is very low. For the numeric attributes, the *GOWER* similarity coefficient considers the equation given by Equation 6.9. The *LAFLIN* similarity coefficient uses the Euclidean distance given in Equation 3.22.

$$d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{max_h x_{hf} - min_h x_{hf}} \tag{6.9}$$

In the previous section, we observed that when the *SM(NCut)* algorithm is applied on the Post-Operative and Soybean dataset, the evaluation values were similar for both of the coefficients. We noticed that for both of the datasets, the number of nominal and binary attributes were very high. The Post-Operative dataset only contained one numeric variable, whereas the Soybean dataset has none. Therefore, both of the coefficients scored similar scores, as they both handle the nominal and binary attributes in the same manner.

The results also show that, on average, the *GOWER* coefficient performs better than the *LAFLIN* coefficient. In particular, the Automobile and Dermatology datasets, for which the *GOWER* coefficient scored the highest, contain more numeric variables than the nominal or binary variables. As discussed above, both of the coefficients in our experiments vary based on the functions used for calculating the distance between the numeric objects. We use the two numeric functions with the spectral clustering algorithms and apply them on the Iris dataset from the UCI repository [3] to evaluate their performances. We selected this dataset because the *true* clusters are well-separated and the dataset contains only four attributes which simplifies the calculations. Therefore, the differences between the distance measures may be more apparent from the Iris dataset. Figure 6.4 illustrates the clusters obtained from the *true* clusters (left), the clusters obtained from the numeric function of the *GOWER* coefficient (middle), and the clusters obtained from the numeric function of the *LAFLIN* coefficient (right). We notice that both of the measures correctly cluster the objects from *true* cluster 1, however, the difference between them is clear in *true* cluster 2 and 3. Notice that these two *true* clusters have objects that overlap near the boundary of the clusters. The objects located at the boundary usually have attribute values slightly different from the other members of their own *true* clusters. While, the numeric function for the *GOWER* coefficient correctly distinguishes several objects near the boundary, the *LAFLIN* coefficient, which used the Euclidean distance to compute the distance between the objects, placed the objects which are located near the boundary, in two different clusters. We notice that the clusters formed from this measure have a shape similar to a sphere. This may be the reason for this measure performing slightly differently than the function of the *GOWER* coefficient. However, the performance of the *LAFLIN* coefficient may be improved by using a different distance measure for the numeric variables.

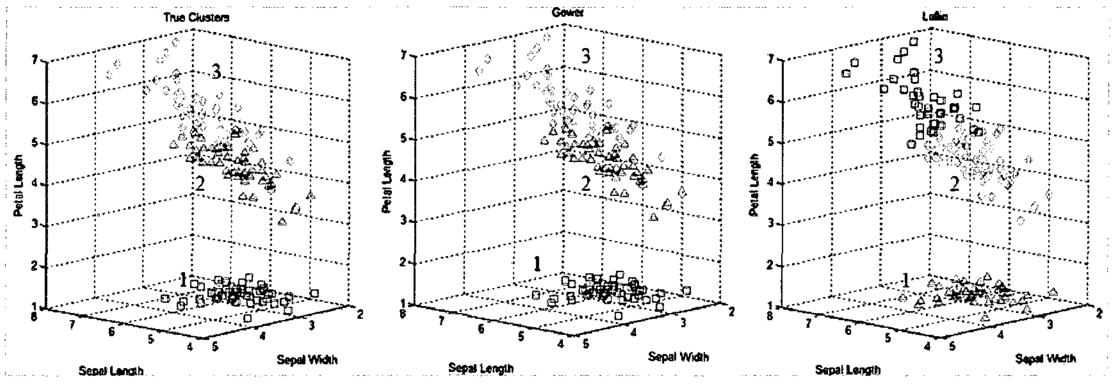


Figure 6.4: Comparison of numeric functions on Iris dataset. (From left) the clusters obtained from the *true* clusters, the clusters obtained from the numeric function of the *GOWER* coefficient, and the clusters obtained from the numeric function of the *LAFLIN* coefficient.

In summary, the discussion from this chapter indicates that, under certain conditions, the *GOWER* similarity coefficient and the *LAFLIN* coefficient may perform similarly. The constrains are as follows: 1) the dataset does not include asymmetric binary variables, and 2) the distance and similarity measures for each of the variables are the same. In our experiments, the two coefficients vary only for the numeric variables. Therefore, the difference between the performances is very similar, as noted in the previous section. According to the evaluation measures, the *GOWER* coefficient, on average, scored higher values for the majority of the datasets. Nevertheless, the performance of the *LAFLIN* coefficient may be improved by incorporating a different function for the numeric variables.

6.4 Clustering Results

In Table 6.10, we provide the best F-measure scores for each of the datasets used in this study. Recall from Chapter 3, that a higher F-measure score indicates that the result from cluster analysis is very similar to the *true* clusters. Here, the results from the cluster analysis discover the natural clusters from the datasets based on the distance or similarity of the objects in the datasets. Therefore, a low F-measure score indicates that the *true* clusters do not correspond well to the natural clusters. For instance, two objects with different sizes may belong to the same *true* cluster, whereas the natural grouping may place them into two different clusters. The F-measure scores from Table

Dataset	SM(Ncut)	NJW(K-means)
Automobile	0.46	0.47
CRX	0.79	0.80
Dermatology	0.85	0.84
Hepatitis	0.73	0.74
Post-Operative	0.56	0.52
Soybean	0.70	0.57

Table 6.10: The F-measure scores from the SM(NCut) and NJW(K-mean) algorithm for the mixed data.

6.10 suggest that the CRX and Dermatology datasets scored 0.80 and 0.85, respectively. The error rate is 19.86 for the CRX dataset, and when compared to the *true* cluster information, we notice that there are 137 members that are placed differently. Among the 137 members, 57 of them are from the *Approved* class, which are placed with the members from the *Not Approved* class. In contrast, there are 80 members from the *Not Approved* class, which are placed with the *Approved* class. In Figure 6.5, we illustrate the clusters from the CRX dataset. The left most figure in Figure 6.5 depicts the clusters according to the *true* cluster labels; the clusters when the *LAFLIN* coefficient is used, are depicted in the middle figure; the clusters from the *GOWER* coefficient are depicted in figure at the right corner. While, the *LAFLIN* coefficient and the *GOWER* coefficient both find similar clusters when compared to the *true* clusters, their sizes differ from that of the *true* clusters, which indicates that there are members that are wrongly placed into the other cluster. The Dermatology dataset correctly clustered 86% of members. We notice that one of the clusters from spectral clustering algorithm is shared by the members from the *true* cluster *2 (Sebpreic Dermatitis)* and *4 (Pityriasis Rosea).*

The Automobile and Post-Operative datasets scored the lowest F-measure scores. This implies that the *true* clusters may not correspond to the natural groupings. The clusters for these two datasets are mostly shared by the members from more than one *true* cluster. For the Hepatitis dataset, one cluster contains 62.60% of members from *true* cluster *Live*. The rest of the members from this cluster are placed with the members

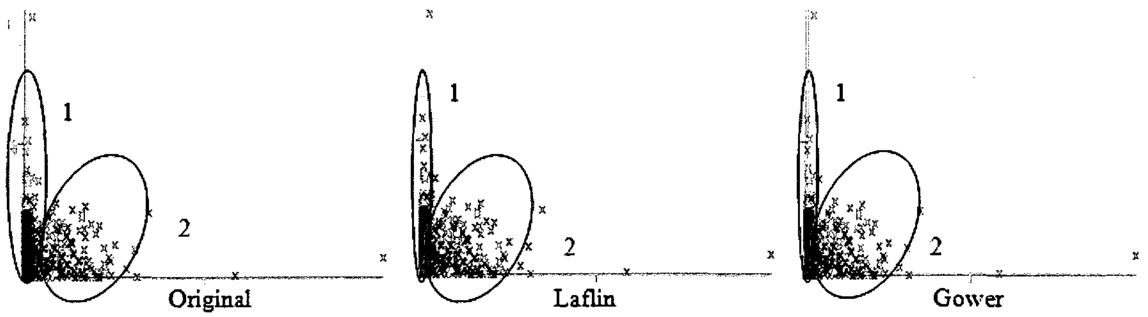


Figure 6.5: Cluster visualization for the CRX dataset. (From left) the clusters obtained from the *true* clusters, the clusters obtained from the *LAFLIN* coefficient, and the clusters obtained from the *GOWER* coefficient.

from *true* cluster *Die*. In Figure 6.6, we illustrate the clusters obtained from the spectral clustering method and the clusters from the *true* cluster. We notice that in the *true* cluster assignments (left), the members from class *Die*, are sparsely located on the left corner, whereas the members from class *Live*, are placed in the denser area marked with the *true* cluster label. In contrast, the clusters obtained from the spectral clustering algorithm (right), show that the members from class *Die* and class *Live* overlap, as the similarity between the objects from these two *true* clusters are high.

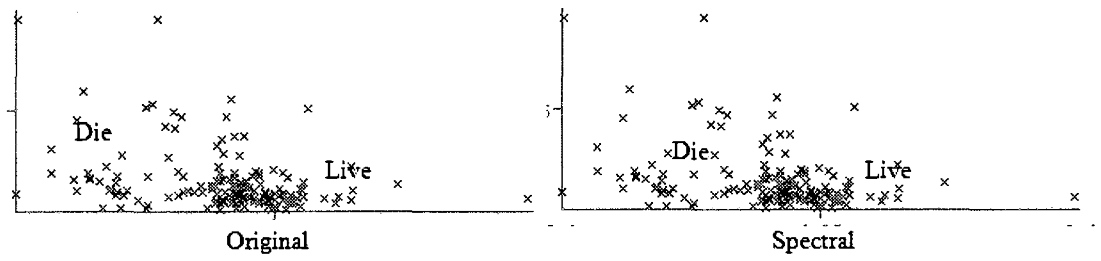


Figure 6.6: Cluster visualization for the Hepatitis dataset. (From left) the clusters obtained from the *true* clusters and the clusters obtained from the *Spectral* algorithm.

6.5 Chapter Summary

In this chapter, we performed spectral cluster analysis on datasets with mixed variable types. Our results from the *SM(NCut)* and *NJW(K-means)* spectral clustering algorithm indicate that the *GOWER* coefficient achieved higher evaluation scores for a majority of the datasets. However, the average difference between the scores is very low for most of the datasets. We also found that under certain conditions the *GOWER* similarity coefficient and the *LAFLIN* similarity coefficient may perform similarly. The constraints are: 1) the dataset does not contain asymmetric binary variables and 2) the distance measure for each of the attributes for both the coefficients is the same. We also notice that, in our experiments, these two coefficients vary only by the distance measures for the numeric variables. Therefore, the difference in the performance of the coefficients occurs due to the distances measured for the numeric variable types. In the next section, we discuss the results from the datasets with the numeric attribute type. The results from the next chapter may also be used to select a suitable distance measure for the numeric variables for the *LAFLIN* coefficient; this is a future research direction which needs to be further explored.

Chapter 7

Result Analysis - Numeric Data

In the previous two chapters we explored the performance of the proximity measures for the binary and mixed data. In this chapter, we compare and evaluate the performance of the distance measures for the datasets with numeric variables. As previously discussed in Chapter 3, we consider eight distance measures in this study. We perform spectral cluster analysis on each of these measures against six numeric datasets. The performance is then measured by comparing the scores of the external cluster evaluation measures. This chapter begins in Section 7.1 with the description of the datasets we used for our experiments. Section 7.2 provides the quantitative result analysis of the scores obtained from the external cluster evaluation measures. Next, in Section 7.3, we provide the evaluation of our results from Section 7.2. We also present a discussion on the clusters obtained from the spectral clustering algorithms in Section 7.4 and we conclude the chapter with a brief summary in Section 7.5.

7.1 Numeric Datasets

We use six datasets to compare the performance of the distance measures. As in earlier chapters, the datasets used in this study also vary in size, the number of *true* clusters, and the type (balanced or imbalanced). They also belong to various application domains. None of the datasets have missing values. For each of the datasets, the *true* cluster labels, or *class labels* are known to us. We use the class information for the purpose of cluster evaluation only. For the rest of this study we will refer to them as the *class* or *true* cluster. A brief description of each of the datasets is given below:

Body Measurement Dataset: The body measurement dataset consists of the body girth measurements and the skeletal diameter measurements of 507 individuals residing in California, US [33]. There are 24 attributes that describe each individual in the dataset. Nine of the attributes are skeletal or diameter measurements, and twelve features represent the body girth or circumference measurements. In addition to these attributes, there are four more features associated with each subject including age, weight, height, and gender. The *class* or *true* cluster attribute is the gender attribute. For the experiments, we keep the last four attributes separate and use only the body girth and skeletal measurements to find the clusters. However, we later use these attributes to analyze the results.

True Clusters	Number of Members
Male	247
Female	260

Table 7.1: The true cluster distribution of the Body Measurement dataset.

Ecoli Dataset: The Ecoli dataset [3], provides information in order to determine cellular localization sites of the proteins. There are 336 instances and 7 numeric attributes to represent each protein. The class is the localization site where a protein may reside. There are eight such locations in this dataset. However, four classes (*OM*, *OML*, *IMS*, and *IML*) are merged together as each of these classes contains only a number of examples. This dataset is an example of an imbalanced dataset (Table 7.2).

True Clusters	Number of Members
Cytoplasm (CP)	143
Inner membrane without signal sequence (IM)	77
Perisplasm (PP)	52
Inner membrane, uncleavable signal sequence (IMU)	35
Others (OM, OML, IMS, IML)	29

Table 7.2: The true cluster distribution of the Ecoli dataset.

Glass Dataset: This dataset contains attributes that define various types of glass. The dataset is obtained from the UCI repository [3]. There are 214 instances present in the dataset, with 9 attributes, including the class or *true* cluster attribute. Since the attribute values have different units, we standardize the dataset to ensure that the distance measures that are not scale invariant work accurately. There are seven classes in the dataset; however, class number 4 does not have any examples in the dataset. The dataset is also an example of an imbalanced dataset. The objects in the dataset form a hierarchical structure as illustrated in Figure 7.11.

True Clusters	Number of Members
Building Windows Float Processed (1)	70
Building Windows Non Float Processed (2)	76
Vehicle Windows Float Processed (3)	17
Vehicle Windows Non Float Processed (4)	0
Containers (5)	13
Tableware (6)	9
Headlamps (7)	29

Table 7.3: The true cluster distribution of the Glass dataset.

Iris Dataset: The Iris dataset is one of the most commonly used and popular datasets from the UCI repository [3]. The dataset consists of 150 instances and there are three *true* clusters or classes. There are only four attributes to describe the characteristics of each of the Iris flowers and these include the sepal and petal sizes (length and width) of each flower. The class labels denote the type of the Iris plant (Table 7.4).

True Clusters	Number of Members
Iris Setosa	50
Iris Versicolour	50
Iris Virginica	50

Table 7.4: The true cluster distribution of the Iris dataset.

SPECT Dataset: The SPECT dataset consists of 44 numeric features extracted from the Single Proton Emission Computed Tomography (SPECT) images [3]. The dataset contains 267 patients and each of them belongs to either of the two classes

True Clusters	Number of Members
Abnormal	267
Normal	55

Table 7.5: The true cluster distribution of the SPECT dataset.

(i.e. normal or abnormal heart condition). This dataset is also an example of an imbalanced dataset, and the *true* cluster distribution is given in Table 7.5.

Wine Dataset: The wine dataset contains the chemical properties of wine grown in Italy [3]. The chemical properties are used to define three different types of wine. There are 13 different features that represent various chemical properties. The wine dataset contains 178 instances. The attributes are standardized to ensure that the distance measures that are scale-dependent work properly. The *true* cluster distribution for this dataset is given in Table 7.6.

True Clusters	Number of Members
1	59
2	71
3	48

Table 7.6: The true cluster distribution of the Wine dataset.

Dataset	Number of Instances	Number of True Clusters	Number of Attributes	Notes
Body Measurement	507	2	21	Balanced
Iris	150	3	4	Balanced
Wine	178	3	13	Balanced
Glass	214	6	8	Imbalanced
Ecoli	336	5	7	Imbalanced
SPECT	267	2	44	Imbalanced

Table 7.7: Summary of numeric datasets. Given in the table (from left) the name of the datasets, the number of instances, the number of *true* clusters, the number of attributes, and the type of the dataset.

Recall from Chapter 4, that we performed spectral cluster analysis with a ten-fold cross validation method on these datasets. The results obtained from the spectral cluster analysis are then evaluated using the external cluster evaluation measures to assess the results. We use the results from the cluster evaluation measure to compare the performance of each of the distance measures. In the next section, we discuss the quantitative results of our experiments.

7.2 Comparison of Distance Measures

Recall from Chapter 4, that the construction of the similarity matrix for a given numeric dataset is more complex than the datasets with binary or mixed attributes. Firstly, most of the existing measures used to compare two numeric objects are distance measures, whereas the spectral clustering algorithms consider the similarity between the objects and take the similarity matrix as an input. Therefore, we need to convert the distance measure into a similarity measure so that they are ready as an input for the algorithms. Secondly, the conversion of distance measures to similarity measures using Equation 3.2, involves searching for a user specified parameter *Sigma*, which, as we discussed in Chapter 4, makes the task more complex. Recall from Chapter 3, that the distance measures for the numeric data that are considered in this study are *Angular Distance (COS)*, *Pearson Correlation Distance (COR)*, *Canberra Distance (CAN)*, *Euclidean Distance (EUC)*, *Minkowski distance (MIN)*, *Manhattan Distance (MAN)*, *Chebyshev Distance (CHEB)*, and *Mahalanobis Distance (MAH)*. In the rest of the chapter we refer to each of these distance measures with their respective abbreviations. Also recall from Chapter 2, that we performed the experiments on two different versions of spectral clustering algorithms: 1) The normalized cut spectral clustering algorithm (*SM(NCut)*) and 2) The spectral clustering algorithm with K-means (*NJW(K-means)*). In the next section we provide the results from the *SM(NCut)* algorithm.

7.2.1 Results from SM(NCut) Spectral Clustering Algorithm

Table 7.8 provides the F-measure and G-means scores obtained for each of the distance measures when tested on the *SM(NCut)* algorithm. Again, as mentioned in Chapter 4, the Entropy measure has several limitations when it is used to evaluate the solutions from the hierarchical clustering algorithms. Thus, to avoid further complications, we do not consider this measure to evaluate the solutions from the *SM(NCut)* algorithm. Recall from Chapter 2, that the algorithm finds the partitions by splitting the eigenvector of the Laplacian matrix (a matrix constructed from the similarity matrix). In Chapter 4, we presented the five splitting points that are used in the experiments: *Split Zero*, *Split Mean*, *Split Median*, *Split NCut1* and *Split NCut2*. From Chapter 5, the results indicate that the *Split Zero* and *Split Mean* splitting methods outperformed among the five splitting points. In contrast, the *Split NCut1* and *Split NCut2* splitting methods often produce imbalanced clusters. We observed that the results from Table 7.8 also show similar results and therefore, we do not repeat the discussion in this chapter. Below we discuss the F-measure and G-means scores from Table 7.8 for each of the datasets individually. The average F-measure scores for each of the datasets are illustrated in Figure 7.1. For both of the evaluation measures (F-measure and G-means), a high value indicates a better result and vise versa.

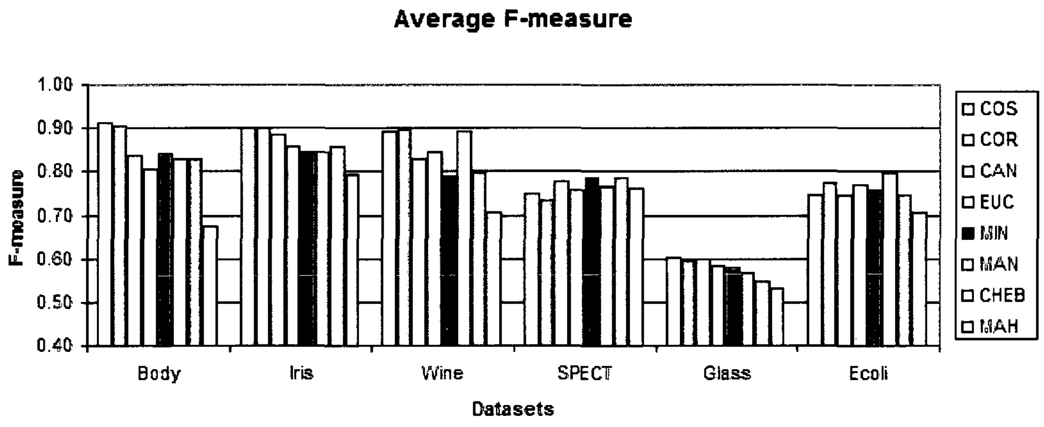


Figure 7.1: The average F-measure scores for the numeric datasets from SM(NCut) algorithm.

Body Dataset

According to the F-measure and G-means scores in Table 7.8, the *COS* distance and the *COR* distance measures performed the best for the Body dataset. The highest F-measure and G-means score is 0.97 and was achieved by the *COS* distance measure when *Split Zero* is used as the splitting point. For the same splitting point the *COR* distance measure scored 0.96. The lowest score was achieved by the *MAH* distance measure as observed from both the F-measure and G-means scores. The average F-measure scores for the *COS* distance and the *COR* distance measures are 0.91 (G-means: 0.92) and

	F-measure Scores									G-means Scores							
				Body									Body				
	COS	COR	CAN	*EUC*	*MIN*	*MAN*	*CHEB*	*MAH*		COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
Split Zero	0.97	0.96	0.90	0.87	0.87	0.87	0.85	0.68	Split Zero	0.97	0.96	0.90	0.87	0.87	0.87	0.85	0.71
Split Mean	0.97	0.97	0.90	0.82	0.84	0.87	0.85	0.68	Split Mean	0.97	0.97	0.90	0.83	0.84	0.87	0.85	0.71
Split Median	0.96	0.96	0.90	0.87	0.86	0.87	0.85	0.64	Split Median	0.96	0.96	0.90	0.87	0.87	0.87	0.85	0.64
Split NCut1	0.85	0.80	0.76	0.72	0.71	0.78	0.81	0.68	Split NCut1	0.86	0.82	0.78	0.74	0.74	0.80	0.82	0.71
Split NCut2	0.83	0.83	0.72	0.74	0.71	0.76	0.78	0.69	Split NCut2	0.84	0.84	0.75	0.77	0.88	0.71	0.79	0.71
				Iris									Iris				
	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH		COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
Split Zero	0.97	0.96	0.94	0.90	0.88	0.86	0.88	0.79	Split Zero	0.97	0.96	0.94	0.90	0.88	0.87	0.88	0.80
Split Mean	0.97	0.96	0.94	0.89	0.88	0.86	0.89	0.78	Split Mean	0.97	0.96	0.94	0.90	0.89	0.87	0.89	0.80
Split Median	0.75	0.76	0.77	0.75	0.75	0.75	0.76	0.72	Split Median	0.77	0.77	0.79	0.77	0.77	0.77	0.77	0.74
Split NCut1	0.90	0.91	0.87	0.86	0.85	0.88	0.88	0.86	Split NCut1	0.90	0.92	0.88	0.87	0.86	0.88	0.88	0.86
Split NCut2	0.91	0.91	0.89	0.88	0.87	0.88	0.88	0.83	Split NCut2	0.91	0.91	0.90	0.88	0.88	0.89	0.88	0.84
				Wine									Wine				
	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH		COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
Split Zero	0.91	0.95	0.84	0.81	0.84	0.90	0.87	0.79	Split Zero	0.92	0.95	0.86	0.83	0.86	0.91	0.87	0.80
Split Mean	0.88	0.90	0.85	0.89	0.85	0.89	0.89	0.64	Split Mean	0.89	0.91	0.86	0.90	0.86	0.90	0.89	0.67
Split Median	0.82	0.76	0.83	0.81	0.82	0.81	0.75	0.72	Split Median	0.83	0.77	0.84	0.82	0.83	0.82	0.77	0.73
Split NCut1	0.91	0.91	0.72	0.78	0.75	0.91	0.71	0.55	Split NCut1	0.91	0.91	0.76	0.79	0.78	0.91	0.73	0.56
Split NCut2	0.94	0.96	0.92	0.93	0.68	0.96	0.76	0.83	Split NCut2	0.94	0.96	0.92	0.93	0.72	0.96	0.79	0.83
				SPECT									SPECT				
	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH		COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
Split Zero	0.80	0.77	0.81	0.80	0.80	0.81	0.80	0.80	Split Zero	0.81	0.80	0.82	0.82	0.81	0.81	0.80	0.82
Split Mean	0.79	0.77	0.82	0.81	0.79	0.81	0.81	0.80	Split Mean	0.81	0.80	0.83	0.82	0.82	0.82	0.82	0.81
Split Median	0.59	0.57	0.70	0.59	0.72	0.61	0.72	0.61	Split Median	0.61	0.59	0.73	0.61	0.74	0.62	0.74	0.64
Split NCut1	0.78	0.78	0.77	0.78	0.81	0.79	0.79	0.79	Split NCut1	0.80	0.80	0.80	0.80	0.82	0.81	0.80	0.81
Split NCut2	0.81	0.77	0.79	0.81	0.82	0.81	0.81	0.80	Split NCut2	0.82	0.80	0.81	0.82	0.83	0.82	0.82	0.81
				Glass									Glass				
	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH		COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
Split Zero	0.62	0.61	0.61	0.61	0.60	0.61	0.59	0.60	Split Zero	0.65	0.64	0.65	0.66	0.64	0.64	0.64	0.62
Split Mean	0.64	0.63	0.64	0.61	0.59	0.61	0.57	0.57	Split Mean	0.66	0.65	0.67	0.64	0.62	0.64	0.61	0.60
Split Median	0.57	0.62	0.62	0.55	0.55	0.44	0.51	0.46	Split Median	0.58	0.63	0.62	0.56	0.56	0.47	0.52	0.47
Split NCut1	0.58	0.50	0.53	0.55	0.58	0.57	0.54	0.50	Split NCut1	0.61	0.54	0.57	0.61	0.62	0.60	0.59	0.56
Split NCut2	0.61	0.60	0.60	0.59	0.57	0.60	0.52	0.52	Split NCut2	0.64	0.64	0.65	0.63	0.61	0.64	0.57	0.56
				Ecoli									Ecoli				
	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH		COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
Split Zero	0.81	0.79	0.83	0.82	0.83	0.85	0.82	0.81	Split Zero	0.82	0.80	0.83	0.82	0.83	0.86	0.83	0.81
Split Mean	0.81	0.80	0.80	0.86	0.85	0.86	0.85	0.82	Split Mean	0.81	0.81	0.81	0.86	0.86	0.86	0.85	0.82
Split Median	0.60	0.75	0.62	0.62	0.64	0.71	0.52	0.56	Split Median	0.63	0.76	0.64	0.64	0.66	0.72	0.55	0.58
Split NCut1	0.76	0.74	0.73	0.71	0.70	0.78	0.74	0.68	Split NCut1	0.77	0.75	0.74	0.73	0.71	0.80	0.75	0.69
Split NCut2	0.75	0.80	0.75	0.84	0.76	0.78	0.80	0.66	Split NCut2	0.76	0.80	0.77	0.84	0.77	0.80	0.81	0.69

Table 7.8: The F-measure and G-means scores for the numeric datasets when tested on the SM(NCut) algorithm.

0.90 (G-means: 0.91), respectively. The average score for the *MAH* distance is 0.67 (G-means: 0.68), which is the lowest among all the distance measures. For the rest of the five distance measures, the average scores are: 0.84 (G-means: 0.85) for the *CAN* distance, 0.80 (G-means: 0.81) for the *MIN* distance, 0.83 (G-means: 0.84) for the *MAN* distance, 0.83 (G-means: 0.83) for the *CHEB* distance, and 0.80 (G-means: 0.82) for the *EUC* distance. The *EUC* distance, which is probably the most widely used distance measure in cluster analysis, scored low for this dataset. We noticed that the distance measures may be grouped into three groups based on the average F-measure scores: 1) the *COS* distance and *COR* distance measure in one group where the scores fall in the range [0.90 – 0.91], 2) the *CAN* distance, *EUC* distance, *MIN* distance, *MAN* distance, and *CHEB* distance measure in the second group where the range is [0.80 – 0.84], and 3) the *MAH* distance measure which scores the lowest (0.67). The average difference between the highest and the lowest F-measure score is 0.24 and the range in which the scores vary is [0.67 – 0.91]. The G-means scores also showed similar results.

Iris Dataset

The *COS* distance and *COR* distance measure scored the highest for this dataset. The highest score is 0.97 for both F-measure and G-means. The lowest score was achieved by the *MAH* distance. The lowest average F-measure score is 0.79. The performance of the rest of the five distance measures is moderately lower than the best score. The average scores for these measures are: 0.88 (G-means: 0.85) for the *CAN* distance, 0.86 (G-means: 0.82) for the *EUC* distance, 0.86 (G-means: 0.83) for the *CHEB* distance, 0.85 (G-means: 0.84) for the *MIN* distance, and 0.85 (G-means: 0.84) for the *MAN* distance. In this case, the distance measures may also be grouped into three categories according to their F-measure scores. The first group contains the *COS* distance and the *COR* distance measure, which scored the highest with a value of 0.90 by both of the evaluation measures; the second group contains the *CAN* distance, *EUC* distance, *MIN* distance, *MAN* distance, and *CHEB* distance measures, where the values fall into the range [0.85 – 0.88]; and the third group includes the *MAH* distance which scores 0.79. The average highest and lowest values falls into the range [0.79 – 0.90] and the average difference is 0.11. The G-means scores also show similar patterns.

Wine Dataset

The highest score is achieved by the *COR* distance measure. Both the F-measure and G-means scored 0.95 when *Split Zero* is used as the splitting method. The second largest scores are achieved by the *COS* distance and the *MAN* distance measure in which both scored 0.89 by the F-measure. The *MAH* distance measure scored the lowest. For the *MAH* distance, the F-measure score is 0.71 and the G-means score is 0.72. In this case, if we grouped the distance measures according to their average F-measure scores, the first group would consist of the *COR* distance, *COS* distance, and *MAN* distance measure as the scores are similar and vary over the range [0.89 – 0.90]. The next group would include the *EUC* distance, *MIN* distance, *CAN* distance, and *CHEB* distance measures where the scores range in [0.79 – 0.84]. The *MAH* distance, which scored the lowest (0.69), would be placed in a separate group. The average difference between the highest and the lowest score is 0.21 ([0.69 – 0.90]).

SPECT Dataset

According to the evaluation measures, the SPECT dataset performed well when the *CAN* distance and the *MIN* distance are used as the distance measures. However, we notice that the *MIN* distance measure scored the highest when the *Split Median*, *Split NCut1*, and *Split NCut2* are used as the splitting methods, whereas the *CAN* distance performed best when the *Split Zero* and *Split Mean* are used as the splitting method. From our discussion in Chapter 5 and 6, we observed that the *Split Zero* and *Split Mean* methods outperformed other methods. Therefore, we consider the results from these two splitting methods to be more robust and select the *CAN* distance for this dataset. The average difference between the highest and lowest scores for this dataset is 0.05, which also implies that the performance of all of the distance measures is very close to one another. The average scores vary over the range [0.73 – 0.79].

Glass Dataset

The glass dataset scored the highest F-measure and G-means scores when the *COS* distance is used as the distance measure. The average highest scores achieved by this distance measure are 0.60 (for F-measure) and 0.63 (for G-means). According to the F-measure scores, the *CHEB* distance and the *MAH* distance measure scored the lowest, and the G-means scores showed that the *MAH* distance achieved the lowest score. Similar to the SPECT dataset, the difference between the highest and the lowest scores is low (0.07) and the average F-measure scores for the distance measures range between [0.53 – 0.60].

Ecoli Dataset

The Ecoli dataset performed well when the *MAN* distance is used as the distance measure. The highest F-measure score achieved by the *MAN* distance is 0.85 (G-means: 0.86). On average the highest score for the F-measure is 0.80 and for the G-means is 0.81. The minimum scores are achieved by the *MAH* distance (0.70). The average difference between the highest and the lowest F-measure scores are 0.09 and the scores for each of the distance measures fall within the range [0.70 – 0.80]. The average scores for the rest of the distance measures are: 0.77 for the *EUC* distance, 0.76 for the *MIN* distance, and 0.75 for rest of the three distance measures (*CHEB* distance, *CAN* distance, and *COS* distance).

In general, our results show that the *COR* distance and the *COS* distance measure often scored higher than the rest of the distance measures. These two distance measures performed well in four out of the six datasets. The datasets are Body, Iris, Wine, and Glass. We also notice, that most of the time, these two coefficients achieved similar values for both of the evaluation measures. The overall average difference is 0.02, irrespective of the dataset or the splitting method used. In contrast, the *MAH* distance measure performed poorly in four out of the six datasets. The datasets for which this distance measure scored the lowest are Body, Iris, Wine, and Glass. The *MAN* distance performed well for the Ecoli dataset. We also notice, that the performance of *EUC* distance, *MIN* distance, *MAN* distance, *CAN* distance, and *CHEB* distance is very similar, and that they often scored moderately, in comparison to the highest and the lowest scores. As observed from the results, the *EUC* distance measure, which is often used in the spectral cluster analysis algorithms, may not always be a suitable choice. We observe that, if the *COS* distance or the *COR* distance measure is used instead of the *EUC* distance, then on average the performance improved by 6.12% and 7.14%, respectively, for our datasets.

7.2.2 Results from NJW(K-means) Algorithm

Here we provide the results from the *NJW(K-means)* spectral clustering algorithm. Recall from Chapter 4, that to compare the performance of the distance measures for numeric datasets, we must first construct the similarity matrix using each of the distance measures for a given dataset. We then apply the *NJW(K-means)* algorithm on each of the similarity matrices and assess the results using the external evaluations measures (i.e. F-measure, G-means, and Entropy). The performance of each of the distance measures is then compared by analyzing the results obtained from the external evaluation measures. Therefore, in this section we analyze the results from F-measure, G-means, and Entropy, as given in Table 7.9, Table 7.10, and Table 7.11, respectively. Recall from Chapter 4, that the higher the F-measure and G-means values, the better the clustering result, and for Entropy a lower value indicates a good result. The datasets used in this study contain the *true* cluster information. Therefore, we use the number of *true* clusters as the number of clusters k for the K-means algorithm in order to compare the distance measures. However, we performed our experiments on a range of values for k ([2..10]) when the *NJW(K-means)* algorithm is used. The reason is two-fold. First, we observe that by considering a range of values for k, the results provided us with additional information about the behaviour of the distance measures. For instance, with the increase of the value of k, the F-measure scores usually decrease. However, for the *MAH* distance measure, we find that the F-measure scores are exactly the same for all values of k for several datasets (i.e. Body, Glass, and SPECT). This provided us with an indication that the MAH distance measure may perform differently than other measures. In the next section, we discuss this in detail when we evaluate the results. Second, by using a range of values for k, we observe the formation of the clusters when k is set to a value other than the number of *true* clusters. For example, consider the Glass dataset, which possess a hierarchical structure. The original number of *true* clusters is 5. However, since we used a range of values for k, starting from 2, we may see how each of the clusters is formed as k increases. Below we provide the evaluation scores for each of the datasets.

Body Dataset

The Body dataset performed best when the COS distance and the COR distance measure are used. The highest score achieved by F-measure and G-means is 0.88. The Entropy also scored the lowest for these two distance measures. The lowest Entropy is 0.36 for both of the distance measures. The lowest F-measure and G-means scores, and the highest Entropy score, are achieved by the MAH distance. The lowest scores are 0.64 and 0.67, respectively, and the highest Entropy as scored by the MAH distance, is 0.69. The F-measure and G-means scores usually decrease as k increases. However, while these scores gradually decreased for all of the distance measures, we notice that it remained constant for the MAH distance. Therefore, this indicates that the MAH distance may have worked differently than the rest of the distance measures. We will analyze this further in the next section. The F-measure scores for the rest of the five distance measures are (from highest to lowest): 0.83 for the CAN distance, 0.80 for the MAN distance, 0.79 for the EUC distance, 0.78 for the $CHEB$ distance, and 0.66 for the MIN distance. The results from G-means and Entropy also show the same pattern. In this case, the performance of the EUC distance is also lower than the other distance measures with the exclusion of the MIN distance and the MAH distance measures. The F-measure scores for all the distance measures range in between 0.64 and 0.88 when $k = 2$.

Iris Dataset

According to the F-measure and G-means scores, the Iris dataset performed best for the COR distance measure. The highest scores when $k = 3$ are 0.82 (F-measure) and 0.83 (G-means). The lowest Entropy for the COR distance is 0.52. We notice that the scores from the COS distance is also very similar to the highest scores. For this distance measure, the F-measure and G-means both scored 0.81 and the Entropy scored 0.51. The distance measure that performed the most poorly for this dataset is the MAH

									F-measure								
					Body										Iris		
k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH	k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
k2	0.88	0.88	0.83	0.79	0.79	0.80	0.78	0.64	k2	0.70	0.70	0.70	0.70	0.70	0.69	0.70	0.70
k3	0.83	0.85	0.75	0.76	0.76	0.68	0.73	0.64	k3	0.81	0.82	0.79	0.78	0.80	0.78	0.79	0.69
k4	0.69	0.68	0.61	0.65	0.66	0.62	0.66	0.64	k4	0.80	0.73	0.72	0.72	0.72	0.73	0.73	0.65
k5	0.61	0.59	0.54	0.58	0.58	0.55	0.59	0.64	k5	0.74	0.68	0.73	0.68	0.68	0.67	0.67	0.63
k6	0.51	0.55	0.50	0.54	0.52	0.52	0.52	0.64	k6	0.63	0.66	0.68	0.64	0.66	0.66	0.63	0.60
k7	0.47	0.48	0.49	0.50	0.50	0.49	0.50	0.64	k7	0.60	0.60	0.66	0.61	0.62	0.63	0.61	0.55
k8	0.44	0.45	0.45	0.45	0.45	0.46	0.44	0.64	k8	0.57	0.57	0.62	0.59	0.58	0.59	0.59	0.50
k9	0.41	0.39	0.42	0.43	0.43	0.42	0.42	0.64	k9	0.54	0.54	0.57	0.56	0.55	0.57	0.56	0.48
k10	0.39	0.38	0.40	0.41	0.40	0.40	0.39	0.64	k10	0.51	0.51	0.55	0.54	0.52	0.54	0.50	0.45
					Wine										Glass		
k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH	k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
k2	0.72	0.71	0.66	0.67	0.66	0.73	0.71	0.51	k2	0.41	0.53	0.53	0.41	0.41	0.41	0.41	0.41
k3	0.94	0.96	0.97	0.85	0.97	0.86	0.83	0.50	k3	0.58	0.50	0.55	0.41	0.41	0.41	0.41	0.42
k4	0.94	0.95	0.84	0.92	0.87	0.87	0.92	0.49	k4	0.56	0.56	0.54	0.49	0.53	0.56	0.55	0.42
k5	0.85	0.88	0.78	0.89	0.86	0.88	0.83	0.49	k5	0.50	0.54	0.53	0.55	0.55	0.57	0.57	0.42
k6	0.76	0.84	0.69	0.81	0.80	0.91	0.83	0.58	k6	0.54	0.54	0.53	0.55	0.56	0.58	0.54	0.50
k7	0.79	0.81	0.65	0.82	0.68	0.86	0.81	0.55	k7	0.46	0.56	0.53	0.51	0.56	0.58	0.55	0.48
k8	0.75	0.78	0.63	0.80	0.68	0.79	0.80	0.61	k8	0.50	0.54	0.53	0.56	0.56	0.55	0.53	0.46
k9	0.71	0.78	0.59	0.73	0.57	0.87	0.71	0.65	k9	0.56	0.51	0.55	0.59	0.58	0.57	0.56	0.49
k10	0.70	0.72	0.56	0.71	0.56	0.82	0.67	0.59	k10	0.56	0.48	0.54	0.54	0.56	0.52	0.57	0.46
					SPECT										Ecoli		
k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH	k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
k2	0.77	0.77	0.77	0.77	0.77	0.77	0.76	0.77	k2	0.44	0.41	0.41	0.44	0.44	0.44	0.44	0.44
k3	0.77	0.77	0.77	0.77	0.77	0.77	0.75	0.77	k3	0.44	0.60	0.41	0.64	0.44	0.44	0.44	0.44
k4	0.76	0.76	0.74	0.75	0.76	0.76	0.74	0.77	k4	0.44	0.73	0.41	0.74	0.63	0.64	0.43	0.45
k5	0.76	0.64	0.74	0.75	0.74	0.74	0.74	0.77	k5	0.65	0.72	0.63	0.74	0.73	0.75	0.65	0.48
k6	0.75	0.63	0.74	0.74	0.75	0.74	0.72	0.76	k6	0.44	0.74	0.53	0.76	0.73	0.75	0.74	0.47
k7	0.67	0.62	0.71	0.74	0.74	0.74	0.70	0.76	k7	0.44	0.68	0.61	0.72	0.62	0.75	0.74	0.45
k8	0.35	0.52	0.70	0.73	0.74	0.74	0.71	0.76	k8	0.43	0.71	0.59	0.69	0.73	0.76	0.74	0.46
k9	0.34	0.61	0.70	0.74	0.72	0.74	0.63	0.76	k9	0.65	0.59	0.57	0.73	0.65	0.75	0.76	0.59
k10	0.30	0.60	0.68	0.72	0.73	0.73	0.61	0.76	k10	0.71	0.60	0.60	0.68	0.75	0.75	0.68	0.55

Table 7.9: The F-measure values for the numeric datasets when experimented on the NJW(K-means) spectral clustering algorithm.

distance. The F-measure and G-means score is 0.69 and the Entropy value is 0.71. The remaining five distance measures scored similarly; the F-measure scores fall within the range [0.78 – 0.80]. Both the *EUC* distance and the *MAN* distance scored 0.78, while the *CHEB* distance and the *CAN* distance scored 0.79 and the *MIN* distance scored 0.80 according to the F-measure scores. For this dataset, when $k = 3$, all of the F-measure scores fall within the range [0.69 – 0.82].

G-means																	
				Body										Iris			
k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH	k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
k2	0.88	0.88	0.83	0.79	0.79	0.80	0.78	0.67	k2	0.72	0.72	0.72	0.72	0.72	0.71	0.72	0.72
k3	0.83	0.85	0.75	0.76	0.76	0.69	0.74	0.67	k3	0.81	0.83	0.79	0.79	0.80	0.79	0.79	0.69
k4	0.71	0.71	0.64	0.68	0.67	0.64	0.68	0.67	k4	0.81	0.75	0.74	0.73	0.73	0.75	0.74	0.67
k5	0.65	0.63	0.59	0.62	0.62	0.60	0.63	0.67	k5	0.76	0.71	0.74	0.70	0.70	0.69	0.70	0.65
k6	0.58	0.61	0.57	0.59	0.57	0.57	0.58	0.67	k6	0.67	0.69	0.71	0.67	0.69	0.70	0.67	0.64
k7	0.55	0.55	0.56	0.57	0.56	0.56	0.56	0.67	k7	0.65	0.65	0.69	0.66	0.66	0.67	0.65	0.59
k8	0.52	0.53	0.53	0.53	0.53	0.54	0.52	0.66	k8	0.63	0.63	0.66	0.64	0.63	0.64	0.63	0.55
k9	0.50	0.49	0.51	0.52	0.51	0.51	0.51	0.66	k9	0.59	0.60	0.63	0.62	0.61	0.63	0.62	0.55
k10	0.48	0.47	0.49	0.50	0.50	0.50	0.49	0.66	k10	0.58	0.57	0.60	0.60	0.59	0.60	0.58	0.51
				Wine										Glass			
k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH	k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
k2	0.74	0.74	0.68	0.69	0.67	0.75	0.73	0.58	k2	0.51	0.59	0.60	0.51	0.51	0.51	0.51	0.51
k3	0.94	0.96	0.97	0.86	0.97	0.87	0.84	0.57	k3	0.62	0.54	0.61	0.50	0.50	0.50	0.50	0.51
k4	0.94	0.95	0.86	0.92	0.88	0.88	0.92	0.55	k4	0.58	0.57	0.57	0.55	0.59	0.61	0.60	0.50
k5	0.87	0.89	0.80	0.90	0.87	0.89	0.84	0.54	k5	0.52	0.55	0.56	0.58	0.61	0.61	0.62	0.50
k6	0.80	0.85	0.72	0.83	0.82	0.92	0.84	0.62	k6	0.56	0.55	0.56	0.58	0.58	0.62	0.56	0.55
k7	0.82	0.83	0.69	0.84	0.72	0.88	0.83	0.59	k7	0.48	0.57	0.56	0.54	0.59	0.62	0.58	0.53
k8	0.78	0.81	0.68	0.83	0.72	0.82	0.82	0.63	k8	0.52	0.55	0.55	0.60	0.57	0.58	0.54	0.51
k9	0.75	0.81	0.65	0.76	0.63	0.88	0.74	0.69	k9	0.57	0.52	0.56	0.63	0.60	0.58	0.57	0.53
k10	0.74	0.76	0.61	0.75	0.62	0.83	0.71	0.64	k10	0.58	0.49	0.56	0.57	0.58	0.53	0.59	0.50
				SPECT										Ecoli			
k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH	k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
k2	0.80	0.80	0.80	0.80	0.80	0.80	0.79	0.80	k2	0.52	0.51	0.51	0.53	0.53	0.53	0.53	0.52
k3	0.80	0.80	0.80	0.80	0.80	0.80	0.77	0.80	k3	0.52	0.63	0.51	0.68	0.52	0.53	0.52	0.52
k4	0.79	0.79	0.77	0.78	0.78	0.78	0.77	0.80	k4	0.52	0.75	0.50	0.76	0.67	0.68	0.52	0.50
k5	0.79	0.64	0.77	0.77	0.76	0.77	0.77	0.79	k5	0.69	0.74	0.68	0.75	0.75	0.77	0.68	0.52
k6	0.78	0.64	0.76	0.76	0.77	0.77	0.75	0.79	k6	0.52	0.76	0.58	0.77	0.74	0.77	0.75	0.52
k7	0.69	0.64	0.74	0.76	0.76	0.77	0.72	0.79	k7	0.52	0.70	0.63	0.73	0.67	0.77	0.76	0.50
k8	0.41	0.56	0.72	0.76	0.76	0.76	0.74	0.79	k8	0.51	0.72	0.63	0.70	0.74	0.77	0.75	0.50
k9	0.38	0.63	0.73	0.76	0.75	0.77	0.64	0.79	k9	0.68	0.63	0.61	0.74	0.67	0.77	0.77	0.61
k10	0.36	0.63	0.70	0.74	0.76	0.75	0.63	0.78	k10	0.72	0.64	0.64	0.71	0.77	0.77	0.71	0.58

Table 7.10: The G-means values for the numeric datasets when experimented on the NJW(K-means) spectral clustering algorithm.

Wine Dataset

The highest F-measure and G-means scores for this dataset are archived by the *CAN* distance and the *MIN* distance measures. Both the F-measure and the G-means scores are 0.97 for both of the distance measures. The *COS* distance and the *COR* distance measure also scored a value close to the highest F-measure and G-means score. For F-measure and G-means both, it scored 0.94 by the *COS* distance and 0.96 by the *COR* distance. The lowest Entropy score is 0.11 for the *MIN* distance and 0.13 for the *CAN* distance. In this case, the *MAH* distance measure also scored the lowest F-measure

								Entropy									
				Body										Iris			
k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH	k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
k2	0.36	0.36	0.46	0.51	0.51	0.50	0.52	0.69	k2	0.76	0.76	0.76	0.77	0.77	0.79	0.78	0.76
k3	0.34	0.35	0.46	0.49	0.49	0.51	0.51	0.69	k3	0.51	0.52	0.58	0.60	0.58	0.60	0.59	0.71
k4	0.50	0.52	0.69	0.34	0.48	0.46	0.49	0.46	k4	0.42	0.51	0.56	0.54	0.53	0.52	0.53	0.66
k5	0.30	0.30	0.39	0.36	0.39	0.38	0.36	0.69	k5	0.40	0.47	0.44	0.51	0.50	0.49	0.49	0.63
k6	0.28	0.28	0.36	0.32	0.34	0.31	0.34	0.68	k6	0.47	0.42	0.40	0.46	0.41	0.40	0.45	0.60
k7	0.27	0.27	0.31	0.30	0.31	0.31	0.33	0.68	k7	0.42	0.39	0.37	0.39	0.39	0.35	0.43	0.57
k8	0.27	0.26	0.28	0.29	0.30	0.30	0.32	0.68	k8	0.38	0.40	0.37	0.34	0.38	0.32	0.39	0.58
k9	0.26	0.26	0.28	0.29	0.29	0.28	0.32	0.68	k9	0.39	0.36	0.36	0.33	0.39	0.32	0.36	0.52
k10	0.25	0.26	0.26	0.27	0.29	0.27	0.31	0.68	k10	0.34	0.34	0.37	0.32	0.36	0.29	0.36	0.54
				Wine										Glass			
k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH	k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
k2	0.61	0.62	0.66	.66	0.66	0.60	0.65	1.08	k2	1.48	1.10	1.20	1.48	1.48	1.48	1.48	1.48
k3	0.19	0.15	0.13	0.38	0.11	0.34	0.42	1.05	k3	0.97	1.08	1.17	1.49	1.48	1.49	1.47	1.47
k4	0.12	0.14	0.19	0.15	0.13	0.23	0.16	1.05	k4	0.95	1.04	1.07	1.33	1.23	1.10	1.14	1.44
k5	0.18	0.13	0.18	0.08	0.10	0.18	0.25	1.02	k5	1.00	0.90	1.06	1.07	1.13	1.08	1.09	1.42
k6	0.18	0.16	0.21	0.06	0.13	0.13	0.15	0.84	k6	0.87	0.86	1.04	1.01	1.00	1.00	1.00	1.26
k7	0.13	0.15	0.19	0.10	0.22	0.07	0.14	0.85	k7	0.99	0.82	1.02	1.02	0.99	0.94	0.99	1.23
k8	0.14	0.13	0.18	0.12	0.18	0.03	0.16	0.47	k8	0.96	0.84	0.99	0.97	0.86	0.95	0.92	1.28
k9	0.20	0.16	0.19	0.12	0.19	0.03	0.15	0.23	k9	0.81	0.85	0.93	0.91	0.85	0.86	0.88	1.10
k10	0.16	0.11	0.15	0.06	0.19	0.03	0.10	0.21	k10	0.77	0.80	0.83	0.94	0.83	0.88	0.89	1.19
				SPECT										Ecoli			
k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH	k	COS	COR	CAN	EUC	MIN	MAN	CHEB	MAH
k2	0.51	0.51	0.51	0.51	0.51	0.51	0.50	0.51	k2	1.39	1.43	1.43	1.39	1.38	1.38	1.38	1.39
k3	0.51	0.51	0.51	0.51	0.51	0.51	0.50	0.51	k3	1.38	1.04	1.43	0.92	1.38	1.38	1.38	1.38
k4	0.50	0.51	0.49	0.50	0.50	0.50	0.49	0.51	k4	1.38	0.74	1.42	0.68	0.91	0.91	1.37	1.35
k5	0.50	0.49	0.49	0.49	0.49	0.49	0.49	0.51	k5	0.87	0.74	0.78	0.60	0.68	0.66	0.86	1.28
k6	0.50	0.47	0.49	0.49	0.49	0.49	0.48	0.50	k6	1.35	0.63	0.93	0.58	0.63	0.60	0.63	1.28
k7	0.46	0.44	0.48	0.49	0.49	0.49	0.47	0.50	k7	1.34	0.71	0.89	0.59	0.89	0.61	0.63	1.24
k8	0.42	0.44	0.47	0.49	0.49	0.49	0.48	0.50	k8	1.35	0.59	0.70	0.55	0.62	0.57	0.59	1.25
k9	0.43	0.44	0.47	0.49	0.48	0.49	0.43	0.50	k9	0.68	0.63	0.67	0.53	0.55	0.57	0.55	0.93
k10	0.43	0.44	0.46	0.48	0.49	0.48	0.42	0.50	k10	0.60	0.55	0.63	0.44	0.52	0.49	0.50	0.86

Table 7.11: The Entropy values for the numeric datasets when experimented on the NJW(K-means) spectral clustering algorithm.

and G-means scores and the highest Entropy score. The scores are 0.50, 0.57, and 1.05, respectively. The *MAN* distance, *EUC* distance, and *CHEB* distance scored moderate scores where the F-measure score ranges between 0.83 and 0.86. The overall F-measure score ranges between 0.50 and 0.97 when $k = 3$.

SPECT Dataset

For the SPECT dataset, all of the distance measures scored similarly. This also includes the *MAH* distance measure, which performed low in the previous three datasets. The F-measure and G-means scores for this dataset are 0.77 and 0.81, respectively. The

CHEB distance performed slightly lower that the rest of the five distance measures but the difference is negligible (0.01). All three evaluation measures showed a similar trend, which we will discuss further in the next section when we evaluate our results.

Glass Dataset

According to the F-measure and G-means scores, the *MAN* distance measure achieved the highest score. The values are 0.58 and 0.62, respectively. However, the Entropy scores suggest that the *COS* distance and the *COR* distance performed the best. The lowest Entropy in this case is 0.86 and 0.87, respectively. For clarification, we manually looked into the clusters and found that the clusters are more robust than the clusters formed by the *MAN* distance. We also consulted the results from the *SM(NCut)* algorithm and observed that the *COS* distance performed slightly better than the *MAN* distance. Therefore, we consider the *COS* distance, *COR* distance, and *MAN* distance measures for this dataset. In this case, the *MAH* distance measure also scored the lowest. The F-measure and G-means scores are 0.50 and 0.55, respectively, and the lowest Entropy is 1.26. The F-measure scores for all of the distance measures fall in the range [0.50 – 0.58] when $k = 6$. Therefore, the rest of the distance measures fall within the range [0.53 – 0.56] excluding the distance measures with the highest and the lowest values. This indicates that the performance of these distance measures is very similar.

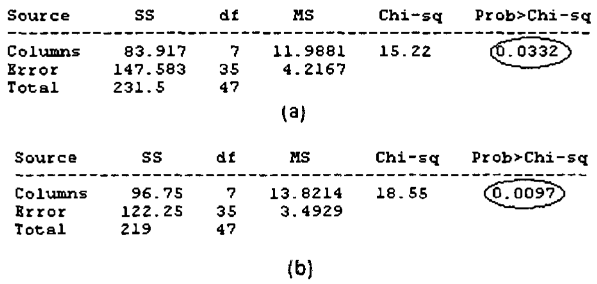
Ecoli Dataset

The F-measure and G-means scores show that the *MAN* distance performed best for the Ecoli dataset. The highest scores are 0.75 and 0.77, respectively. The *MAH* distance measure again scored the lowest. The values are 0.48 for F-measure and 0.52 for G-means. We also notice that the *COR* distance, *EUC* distance and *MIN* distance scored well for this dataset. The F-measure scores for these three distance measures are 0.72, 0.74, and 0.73, respectively. The F-measure scores for all the distance measures fall in the range [0.48 – 0.75] when $k = 5$.

The discussion in this section showed results similar to those of experiments performed on the *SM(NCut)* algorithm. In both cases, the results indicate that the *MAH* distance measure often scored the lowest scores over a range of datasets. The results also indicate that the performance of the *MAH* distance measure is poor for all of the six datasets. Moreover, the *EUC* distance measure never scored the highest score in any of the datasets. In two datasets (i.e. Body and Iris), the *COR* distance and the *COS* distance performed well, and for the Wine and Glass datasets, the scores were very close to the highest scores achieved for these two datasets. The *MAN* distance performed well for the Ecoli and Glass datasets.

7.3 Result Evaluation

In the previous section, we performed a quantitative analysis on the results from the cluster evaluation measures to compare the performance of the distance measures. In this section, we analyze further to evaluate the performance of the distance measures and address the issues related to the possible reasons that may have caused the differences in the performance. The results from the Friedman Test that we used to measure the statistical significance, is depicted in Figure 7.2. The *p-values* are 0.0332 and 0.0097, when the proximity measures are combined with the *SM(NCut)* and *NJW(K-means)* algorithm, respectively. In both cases, the *p-values* are less than 0.05. According to our discussion in Chapter 4, from a *p-value* less than 0.05, we may conclude that there is a significant difference between the performance of the proximity measures on our datasets. Our re-



```
Source        SS       df      MS       Chi-sq   Prob>Chi-sq
-------------------------------------------------------------
Columns      83.917     7    11.9881    15.22      0.0332
Error       147.583    35     4.2167
Total       231.5      47
                             (a)

Source        SS       df      MS       Chi-sq   Prob>Chi-sq
-------------------------------------------------------------
Columns      96.75      7    13.8214    18.55      0.0097
Error       122.25     35     3.4929
Total       219        47
                             (b)
```

Figure 7.2: Results from the Friedman test when applied on the results from (a) the SM(NCut) algorithm and (b) the NJW(K-means) algorithm.

sults showed that the *MAH* distance often performed poorly when compared to the rest of the distance measures according to the cluster evaluation measures. We noticed that, when the *MAH* distance is used, the spectral clustering algorithms produced imbalanced clusters. Here the clusters are imbalanced when one partition contains relatively fewer objects than the other cluster. We also noticed that the objects that are placed in the smaller cluster are the objects that have the lowest *degree*. Recall from Chapter 2 that the *degree* is the total similarity value from one object to rest of the objects in a dataset. In spectral clustering, the objects are considered as nodes in the graph, and a partition separates objects where the total within cluster similarity is high and the between cluster similarity is very low. Therefore, when the *degree* is low for an object, in comparison to the rest of the objects, it indicates that the object is less similar than most of the objects in the dataset. Now, the equation of the *MAH* distance defines an ellipsoid in n-dimensional space [45], [31]. The distance considers the variance (how spread out the values are from the mean) of each attribute as well as the covariance (how much two variables change together) of the attributes in the datasets. It gives less weight to the dimensions with high variance and more weight to the dimensions with small variance. The covariance between the attributes allows the ellipsoid to rotate its axes and increase and decrease its size [31]. Therefore, the distance measure is very sensitive to the extreme points [21]. Figure 7.3 illustrates a scenario showing the *MAH* distance between the objects. In the figure, the distance between object 1 and 2 will be less than the distance between object 1 and 3, according to the *MAH* distance. This is because, object 2 lies very close to the main axes along with the other objects, whereas the object 3 lies further away from the main axes. Therefore, in such situations, the *MAH* distance will be large. For numeric data, the similarity will be very low when the distance is very large. The function in Equation 4.3, which is used to convert a distance value into a similarity value, will give a value close to zero when the distance is very large. Therefore, the *degree* from this object to the rest of the objects becomes very low and the spectral methods separate these objects from the rest of the objects. This is one of the possible reasons for the *MAH* distance performing poorly. It either discovers imbalanced clusters or places similar objects wrongly into two different clusters. However, one possible way to improve the performance of the *MAH* distance measure might be by changing the value of σ to a larger value. In this way, according to our discussion in Chapter 4, the similarity will not have a very small value.

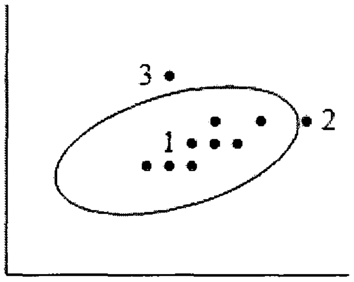


Figure 7.3: A scenario depicting the Mahalanobis (MAH) distance between three points.

Our results also indicate that the *COR* distance and the *COS* distance performed best for four out of the six datasets. Both of the distance measures calculate the *relative* distance from a fixed point (mean or zero, respectively). Therefore, two objects with a similar pattern will be more similar even if their sizes are different. The *EUC* distance, *MAN* distance, *MIN* distance, *CAN* distance, and *CHEB* distance measure, however, calculate the *absolute* distance (i.e. straight line distance from one point to another). For instance, the Body dataset partitions the objects into two main clusters, one with larger body dimensions and another cluster with smaller body dimensions. According to the *true* cluster information, the larger body dimensions denote the *Male* population and the smaller body dimensions denote the *Female* population. When compared to the *true* clusters, we observed that several individuals whose body dimensions are comparatively lower than the average body dimensions of the *Male* population, are placed with the individuals from the *Female* population by the distance measures that calculate the absolute distance. Conversely, *Female* individuals with larger body dimensions than the average body dimensions of the *Female* population are placed with the individuals from *Male* population. Therefore, these individuals that fall very close to the boundary of the two *true* clusters, are placed differently by the distance measures that calculate the absolute distance (i.e. *EUC* distance, *MAN* distance, and *MIN* distance) than the distance measures that consider the relative distance (i.e. *COR* distance and *COS* distance). In such cases, the *COS* distance and the *COR* distance correctly identify these individuals. In Figure 7.4, we plot the first two attributes of the Body dataset when the *EUC* distance is used as the distance measure. The object marked with a smaller circle is an example of a *Male* individual with smaller body dimensions. When the *EUC* distance is used as the distance measure in the spectral clustering algorithm, this object is placed with the *Female* population. The objects marked with the larger circle illustrate the reversed situation, where a *Female* individual with larger body size is placed with the individuals from Male population. In both of the situations, the *COR* distance and the *COS* distance placed the objects with their own groups.

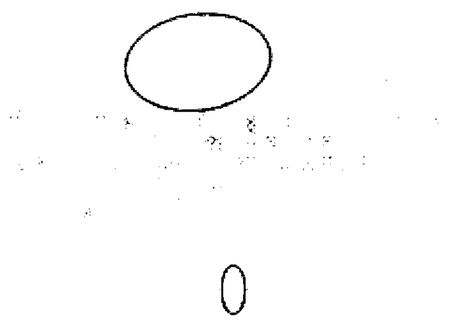


Figure 7.4: Example of cluster assignments of the Body dataset. The circles are used to point to the several individual members that are placed differently.

In Figure 7.5, we provide the clusters from the Ecoli dataset when the *MAN* distance (Left) and the *COS* distance (Middle) are used as the distance measure. The farthest right figure (with the title *Original*) depicts the *true* clusters. The objects, according to the *true* clusters, overlap between the clusters in a number of situations (e.g. the objects marked with circle 2 and 4, or the objects marked with circle 3 and 5). This indicates that there are several objects in the dataset that may be very similar but are placed in two different *true* clusters. When the spectral clustering algorithms are applied to this dataset, both the *COS* distance and the *MAN* distance divide the *true* cluster marked with circle 1 (in Figure7.5) into two different clusters. However, the clusters produced by the *COS* distance contain members from *true* cluster 1 and 3, whereas the clusters produced by the *MAN* distance contain the members from *true* cluster 1. The figure indicates that the shape of the clusters produced by the *COS* distance are more elongated toward the origin, which is the reason why some of the members from *true* cluster 3 are included in the cluster.

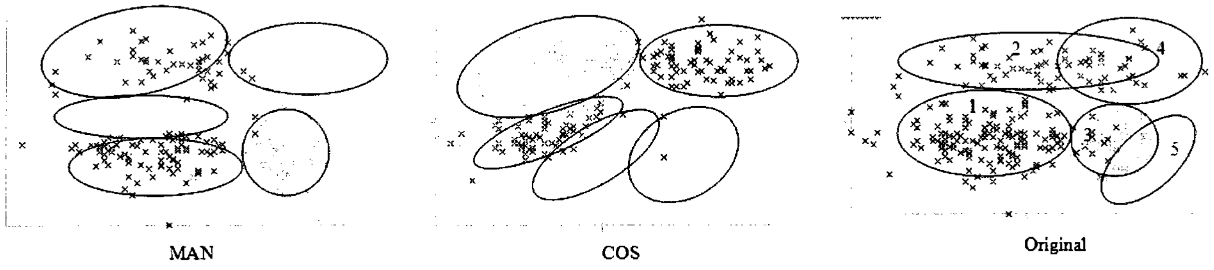


Figure 7.5: Example of cluster assignments of the Ecoli dataset. (Left) The clusters obtained by using the *MAN* distance measure, (Middle) the clusters obtained by using the *COS* distance, and (Right) the original *true* clusters.

7.4 Clustering Results

In Table 7.12, we provide the maximum F-measure scores obtained from the two spectral clustering algorithms. For each dataset the maximum F-measure score is the value obtained by the distance measure that performed the best for that particular dataset. For the first three datasets, the F-measure score is high, which indicates that the clusters discovered from the spectral clustering algorithms are very similar to the *true* clusters. In Figure 7.6, we provide the similarity matrices for the Body dataset. The similarity

Dataset	SM(Ncut)	NJW(K-means)
Body	0.97	0.88
Iris	0.97	0.82
Wine	0.95	0.97
SPECT	0.81	0.77
Glass	0.61	0.58
Ecoli	0.85	0.75

Table 7.12: The F-measure scores for the SM(NCut) and NJW(K-means) spectral clustering algorithm for the numeric datasets.

matrix on the left, in the figure, is ordered according to the *true* cluster index. The similarity matrix on the right is from the spectral clustering algorithm where objects are ordered according to the clusters index. According to [63], if the clusters are well separated, then the similarity matrix should be block diagonal, where the blocks on the main diagonal refer to each individual cluster, and where the similarity is the maximum. The Iris and Wine dataset also give similar structures.

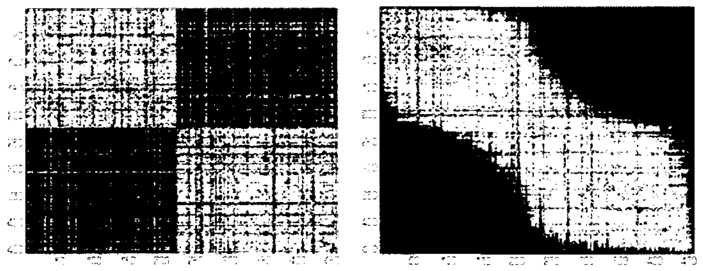


Figure 7.6: Similarity matrices for the Body dataset. (Left) Similarity matrix when objects are ordered according to their *true* cluster index. (Right) Similarity matrix when objects are assigned according to their cluster index obtained from spectral clustering algorithm.

The SPECT, Glass, and Ecoli datasets are imbalanced datasets. One of the reasons for these three datasets not performing as well as the previous three datasets is that the *true* cluster labels may not correspond to the natural clusters in the datasets. The spectral cluster algorithms discover natural clusters after considering the similarity between the objects. We provide an example to illustrate the situation. In Figure 7.7, we provide the clusters (true clusters) according to the *true* cluster labels for the Ecoli dataset (left) and the clusters obtained from the spectral clustering algorithm (right). The figure shows that for the *true* clusters, several objects from *Class 2* and *Class 4* are very similar to one another but have different class labels. This is also true from members of *Class 3* and *Class 5*. When the spectral clustering algorithm is applied on this dataset, it discovers clusters similar to the one (right) in Figure 7.7. The clusters are based on the natural grouping of the objects. Therefore, unlike the *true* clusters, which separate the objects from *Class 3* and *Class 5*, the spectral clustering algorithm groups them together, as the objects are very similar. In such situations, the evaluation measures give lower scores, as the natural groups do not necessarily correspond to the *true* clusters.

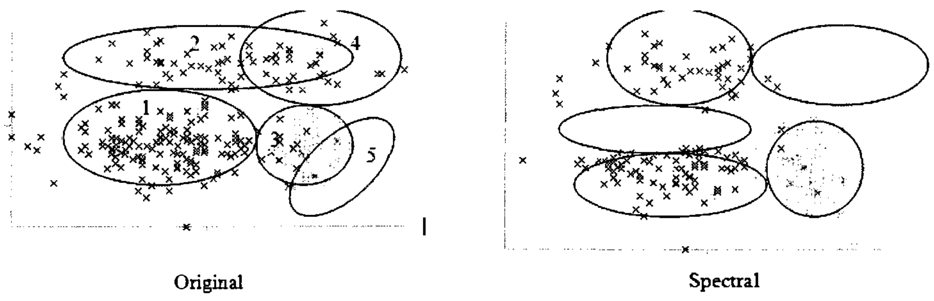


Figure 7.7: Cluster visualization for the Ecoli dataset. (Left) The clusters according to the *true* cluster labels for the Ecoli dataset. (Right) The clusters obtained from the spectral clustering algorithm.

The SPECT and Glass dataset also show similar behavior, where the objects from the *true* clusters are scattered across the space. To verify that the natural clusters may not correspond to the *true* clusters, we applied the K-means algorithm on the SPECT dataset. Figure 7.8 illustrates the clustering results from the spectral clustering algorithm (left figure) and the K-means algorithm (middle figure). In addition to the clustering result, the figure also depicts the *true* clusters (right figure). The figure shows that when the clustering algorithm is applied to the SPECT dataset, the clusters are generated based on the distance between the objects. The two clusters are represented with two differ colors (red and blue) and marked with circles. In contrast, the *true* clusters show that the objects from both of the *true* clusters are scattered across the space and there is no clear partition or natural groupings visible from the right most figure.

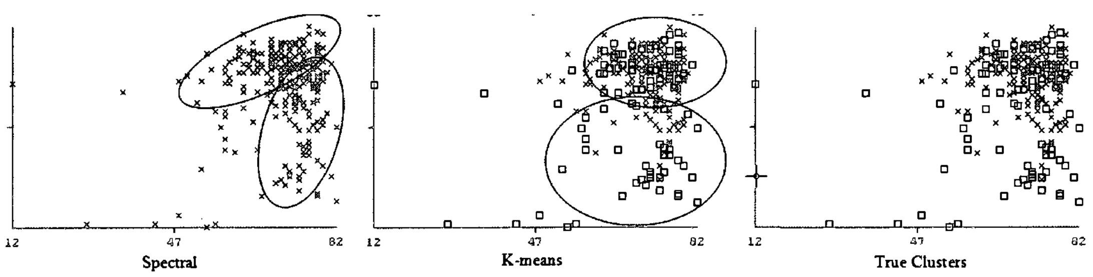


Figure 7.8: Cluster visualization for the SPECT dataset. (Left) The clustering result from the spectral clustering algorithm, (middle) the clustering result from the K-means algorithm, and (right) the *true* clusters according to the cluster labels for the SPECT dataset.

7.4.1 Clustering Results from SM(NCut) Algorithm

Here we provide the clusters discovered from the *SM(NCut)* algorithm. For each dataset, we provide the clusters obtained from the distance measure that performed the best.

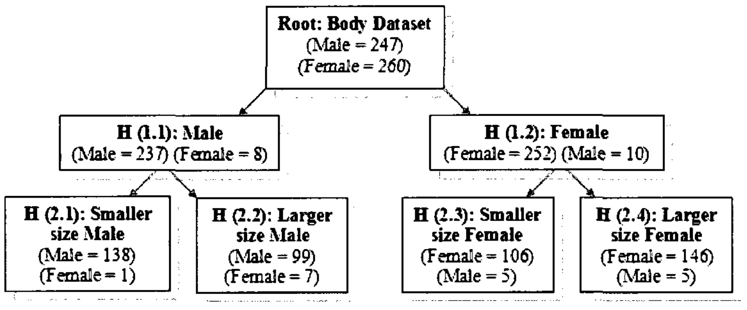

Root: Body Dataset (Male = 247) (Female = 260)
H (1.1): Male (Male = 237) (Female = 8)
H (1.2): Female (Female = 252) (Male = 10)
H (2.1): Smaller size Male (Male = 138) (Female = 1)
H (2.2): Larger size Male (Male = 99) (Female = 7)
H (2.3): Smaller size Female (Female = 106) (Male = 5)
H (2.4): Larger size Female (Female = 146) (Male = 5)

Figure 7.9: The hierarchical tree structure of the Body dataset when SM(NCut) algorithm is applied. H(1.1) and H(1.2) is the first level of the tree and H(2.1), H(2.2), H(2.3), and H(2.4) are the nodes from the second level.

Figure 7.9 depicts the hierarchical structure for the body dataset. The hierarchical structure is a symbolic way of showing the results where each square box represents a node in the tree. The boxes include the level of the tree as well as the dominating members according to the *true* cluster information. The overall error rate for this dataset is very low (3.55%) and only 18 out of 507 instances are incorrectly clustered. The class distribution for this dataset is given in Table 7.1. We see from the tree, that 8 of the female members are incorrectly clustered with the male members, and 10 of the male members are wrongly placed with the female members. The first level of the tree almost completely separates the female and male members. When comparing the members from Level 1.1, which mostly represents the male members, and Level 1.2, which mostly represents the female members, we notice that the members in Level 1.1 exceed the members in Level 1.2 in all features except the *Thigh Girth* measurement. Interestingly, the second level separates the groups into two different sizes based on the body girth measurements and skeletal diameter measurements. The first group contains the members which have comparatively smaller body structure based on these attribute values. In contrast, the members in the second group have a relatively larger body structure. We also notice that the average age and weight of the members in this group are significantly higher than the members with a relatively smaller body structure. This is shown in Table 7.13 where we provided the centroids for each of the nodes (numbered according to their position in the tree) in the hierarchical tree given in Figure 7.9. In the figure, **H(1.1):Male** and **H(1.2):Female**, represent the first level of the hierarchy where the original dataset is divided into two partitions. The second level of the hierarchy is represent by **H(2.1):Male**, **H(2.2):Male**, **H(2.3):Female**, and **H(2.4):Female**. In this case, **H(2.1):Male** and **H(2.2):Male** are the partitions from **H(1.1):Male**, whereas **H(2.3):Female**, and **H(2.4):Female** are the partitions from **H(1.2):Female**. Since the dataset contains 24 attributes and cannot be fitted into one row, we split the attributes into three segments. The segments are *Skeletal Measurements*, *Girth Measurements*, and *Other Measurements*.

Figure 7.10 depicts the hierarchical tree for the Iris dataset. The overall error rate is 2.67%. Since the original dataset has three classes (Table 7.4), we only show the first two levels in the figure. The original tree contains more nodes, as the algorithm stops only when one of the stopping criteria is satisfied. In the first level of the tree, the spectral

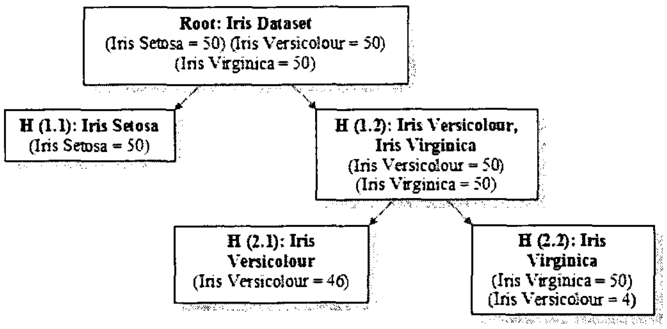


Figure 7.10: The hierarchical tree structure of the Iris dataset when SM(NCut) algorithm is applied.

	Skeletal Measurements									Other Measurements		
Hierarchy level	Biacromial Diameter	Pelvic Breadth	Bitrochanteric Diameter	Chest Depth	Chest Diameter	Elbow Diameter	Wrist Diameter	Knee Diameter	Ankle Diameter	Age	Weigh	Height
H(1.1):Male	41.18	28.00	32.45	20.74	29.94	14.41	11.22	19.54	14.68	31.96	77.84	177.44
H(2.1):Male	41.42	27.54	31.93	20.03	29.52	14.32	11.18	19.52	14.56	27.82	74.49	177.55
H(2.2): Male	40.86	28.63	33.16	21.70	30.51	14.54	11.27	19.58	14.85	37.58	82.38	177.29
H(1.2):Female	36.66	27.70	31.57	17.77	26.12	12.44	9.91	18.16	13.10	28.48	61.10	165.69
H(2.3):Female	36.70	26.93	30.96	17.21	25.83	12.29	9.82	17.89	12.91	25.70	57.71	165.40
H(2.4):Female	36.62	28.72	32.38	18.54	26.52	12.65	10.04	18.52	13.36	32.22	65.66	166.09
	Girth Measurements											
	Shoulder	Chest	Waist	Navel	Hip	Thigh	Bicap	Forearm	Knee	Calf Max	Ankle Min	Wrist Min
H(1.1):Male	116.51	100.93	84.40	87.42	97.43	56.33	34.41	28.22	37.11	37.14	23.10	17.18
H(2.1): Male	115.82	99.19	79.35	82.65	94.95	55.51	34.19	28.19	36.58	36.66	22.80	17.15
H(2.2): Male	117.43	103.29	91.24	93.90	100.78	57.43	34.71	28.25	37.82	37.77	23.51	17.21
H(1.2):Female	100.45	86.08	69.99	83.95	95.92	57.31	28.16	23.83	35.41	35.05	21.31	15.11
H(2.3):Female	99.51	84.06	66.66	78.06	92.75	55.52	27.29	23.47	34.84	34.72	21.03	14.97
H(2.4):Female	101.72	88.81	74.47	91.88	100.19	59.71	29.33	24.33	36.18	35.50	21.69	15.30

Table 7.13: The cluster centroids for the Body dataset when SM(NCut) algorithm is used.

clustering algorithm separates *Iris Setosa* from the two other types. This is because the sepal and petal sizes of the *Iris Setosa* are clearly different from *Iris Versicolour* and *Iris Virginica.* Its petal width and length is strictly smaller than the other two types. The objects that are wrongly placed according to the *true* cluster are from the class *Iris Versicolour.* They are incorrectly clustered with the members of class *Iris Virginica.*

For the wine dataset, the number of incorrectly clustered objects is 20, among 178 objects. The members from class *1* and class 3 are correctly placed. However, we noticed that the members of class *2* overlap between the clusters. In the hierarchical tree, the first two levels separate the three classes, with the exception that 17 members from class *2* are placed with the members of class *1*, and 3 of them are clustered with the members of class *3.* The overall error rate is 11.24%. This is likely because the chemical attributes of the misplaced objects from class *2* are more similar to the attribute values of class *1* for the five attributes (Magnesium, Total phenols Flavanoids, Nonflavanoid phenols, Proanthocyanins and OD280/OD315 of diluted wines).

According to the information provided with the Glass dataset, the objects in this dataset follow a hierarchical structure as given in Figure 7.11. The figure shows that in the first level of the hierarchy, there are two groups: one that contains the members from classes *1, 2, 3* and *4*, which represent the *Window Glasses*, and the second group which includes members from classes *5, 6* and *7*, which represent the *Non-Window Glasses.*

```
                      |-- 163 Window glass (building windows and vehicle windows) (1,2,3,4)
                      |     -- 87 float processed (1,3)
                      |        -- 70 building windows (1)
                      |        -- 17 vehicle windows (3)
                      |     -- 76 non-float processed (2,4)
Glass (1,2,3,4,5,6,7) |        -- 76 building windows (2)
                      |        -- 0 vehicle windows (4)
                      |-- 51 Non-window glass (5,6,7)
                      |     -- 13 containers (5)
                      |     -- 9 tableware (6)
                      |     -- 29 headlamps (7)
```

Figure 7.11: The original hierarchical structure for the Glass dataset.

In the second level, the second group stays intact, but the first group may again be subdivided into two groups with members from classes *1* and *3* (representing the family of *Float Processed Glasses*) placed together and classes *2* and 4 (*Non-Float Processed Glasses*) clustered together. As mentioned above, for this dataset the *true* clusters may not correspond to the natural groupings. Therefore, two similar objects may be placed in two different *true* clusters and this may cause a high error rate. Most of the elements from classes *1, 2* and *3* (*Window Glasses*) are placed together with exception of several members from classes *1* and *2* that are placed with the members from classes *5, 6* and *7* (*Non-Window Glasses*) in a separate cluster. The overall error rate at this level is 9.34%. In the second level the members from class *7* are placed into a separate cluster, whereas the members from classes *5* and *6* are grouped together. The partitions that subdivide the members of the *Window Glasses* are not satisfying. In most cases, both of the partitions contain members from classes *1*, *2*, and *3*, not necessarily grouping according to the original structure. Thus, the overall error rate is high for the entire tree.

The SPECT and Ecoli datasets also show similar behavior. For the SPECT dataset, both of the partitions contain members from both of the *true* clusters. The error rate at the first level of the tree is 38.58%. The first partition of the Ecoli dataset contains members from *Class 2(IM)* (69/77 members) and *Class 4(IMU)* (34/35 members). Figure 7.7 shows that in the original dataset, the objects from these two *true* clusters are located very close to one another. Therefore, several nodes include objects from both of the *true* clusters. The second partition contains the members from *Class 1(CP)* (142/143 members), *Class 3(PP)* (47/52 members), and *Class 5(Others)* (24/29 members). The next level for this partition separates the objects from *Class 1 (CP)*, *Class 3 (PP)* and *Class 5 (Others)*. The members from *Class 3 (PP)* and *Class 5 (Others)* are placed into the same cluster, as the objects from these two groups are very similar to one another (Figure 7.7).

7.4.2 Clustering Results from NJW(K-means) Algorithm

In this part, we present the clusters discovered from the *NJW(K-means)* algorithm. Similar to the previous section, we provide clusters obtained from the distance measure that performed best for a particular dataset.

There are two *true* clusters in the Body dataset. Therefore, k is set to 2. We also provide the results from $k = 3$ and $k = 4$ to see how the clusters form with an increase in k.

$k = 2$: Recall that this dataset contains the body measurements of 507 individuals. When $k = 2$, the clusters separate the *Male* individuals and the *Female* individuals and assign them to two different clusters. The algorithm correctly clusters 217 *Males* out of 247 *Male* individuals. Thus, there are 30 *Male* individuals that are wrongly placed in the cluster, which contains mainly the *Female* individuals. Out of 260 *Female* members, 28 members are wrongly clustered and thus placed in the cluster with *Male* members. A further analysis on the wrongly clustered objects shows that the *Male* members with relatively smaller body measurements are placed in the wrong cluster with the *Female* members. The average body measurements for these *Male* members are significantly below the average body size of the *Male* population. Moreover, the average body size for these members is also very close to the average size of the *Female* population. This also holds true for the *Female* members that are wrongly clustered with the *Male* members. For this wrongly placed fraction of *Female* population, the average body measurements are much higher than the average body size of the *Female* population, but are very close to the average body measurements of the *Male* members. Thus, even though some of the members are incorrectly clustered according to the class labels, according to the body measurements, the clusters separate a group with higher body size from those with the lower body size. The overall error rate when class labels are considered is 11.44%.

$k = 3$: In this case, one of the clusters is comprised mostly of the members from class *Male*. The other two clusters share the members from class *Female*. The two clusters that include members from class *Female* differ in that one is made up of *Female* members with below average *Female* body sizes and the other contains the remaining *Female* members.

$k = 4$: In this case, two of the clusters are comprised mostly of the members from class

Female and the rest of the two clusters are made up of the members from the class *Male.* When we analyzed the clusters, we found that one of the two clusters (which contains *Female* members) contains mainly the *Female* members that are below the average body size (smaller size) with regards to the entire *Female* population. The other cluster contains the *Female* members with larger body sizes. The average for this group falls above the average body measurements of the *Female* population. The situation is similar for the rest of the two clusters, which contain members from the *Male* population. One of the clusters is for larger sized *Males* (above average) and the other is for smaller sized *Males* (below average).

The Iris dataset separates the three different types of Iris plants based on the length and width of the sepal and petal.

$k = 2$: In this case, one cluster mostly contains members from class *Iris Setosa* and the second cluster includes members from class *Iris Versicolour* and *Iris Virginica.* The reason for such groupings may be that the members from *Iris Setosa* have comparatively smaller sepal length and smaller petal length and width than the other two classes.

$k = 3$: The three clusters in this case comprise members from the three different clusters, almost correctly separating the objects belonging to the same class. There are 150 objects in the dataset, with each class having exactly 50 members. The first cluster contains 45 members from class *Iris Setosa*, where the rest of the members are placed with class *Iris Virginica.* The second cluster contains 42 members from class *Iris Versicolour* and rest of the members from this class are placed either with class *Iris Setosa* or *Iris Virginica.* The third cluster comprises 43 members from class *Iris Virginica*, and in this case, the rest of the members are also placed in either of the two other clusters. The overall error rate for this solution is 13.33% because of the 20 members from the three classes that are misplaced and put into different clusters other than their own. These members have slightly different sepal and petal sizes than the rest of the objects in their group, and are therefore placed with the objects that are more similar to them.

$k = 4$: *Cluster 1* and *Cluster 2* include members from class *Iris Versicolour* and class *Iris Virginica* respectively. The other two clusters share the members from class *Iris Setosa.* The clusters differ on the attribute values and thus the members from class *Iris Setosa* that fall below the average are placed together in one cluster and the ones that are above the average are grouped together.

The Wine dataset has three *true* clusters. The clusters formed by this algorithm are:

$k = 2$: When $k = 2$, *Cluster1* is dominated by the members from a type of wine denoted as *Class 1* (The name is not provided with the dataset). The second cluster contains the members from *Class 2* and *Class 3*. We notice that *Class 1* has, on average, a comparatively lower value for all the chemical attributes than the average attribute values of *Class 2* and *Class 3*.

$k = 3$: The three clusters achieved in this case, almost correctly represent the original *true* cluster distribution. Out of 178 objects, only 7 are misplaced, and the overall error rate is 3.93%. Thus, *Cluster 1*, *Cluster 2* and *Cluster 3* represent *Class 1*, *Class 2* and *Class 3* respectively. The objects that are placed in the wrong cluster according to the *true* cluster labels are from *Class 2*, and are placed differently as their chemical properties are slightly different than the other objects in their *true* cluster.

The Glass dataset has six classes and the clusters follow a hierarchical structure as given in Figure 7.11. Below, we present each of the clustering solutions from $k = 2..6$. Recall that this dataset identifies the type of glass based on the information from various chemical substances. As mentioned above, for this dataset the *true* clusters do not correspond very well to the natural clusters. Therefore, there are objects that are very similar but belong to two different *true* clusters, and the natural clusters often contain members from several *true* clusters.

$k = 2$: *Cluster 1* mostly contains the members from classes *1, 2* and *3*. These members belong to the family of *Window Glasses*. *Non Window Glasses*, which are denoted by the symbols *5, 6* and *7*, are placed together in the second cluster. We also notice that in this cluster there are several members from the *Window Glasses* that are incorrectly clustered with the members of *Non Window Glasses*. The main difference between these two clusters is, that the *Window Glasses* (1, 2, and 3) have a relatively low value for most of the chemical substances. In contrast, the *Non Window Glasses* have high values for most of the chemical substances. Thus, the members from class *Window Glasses* that are wrongly placed with the *Non Window Glasses* have comparatively higher values for the chemical substances than the members from their own group, hence explaining why they are clustered in such way.

$k = 3$: In this case, one of the clusters is mostly comprised of the members from class *7*, which denotes the glass of type *Headlamps*, and falls into the category *Non Window Glasses*. The other two clusters are formed similarly to the previous solution. One cluster contains members from classes *1, 2* and *3* and the other contains members from classes *5, 6* and *7* along with some of the members from the former group.

$k = 4$: In contrast to the previous solution, the clustering solution in this case, separates some of the members of classes *1* and *2*, which denote the glass of type *Building Window Float Process* and *Building Window Non-Float Process*, respectively, into two separate clusters. The third cluster includes glasses of type *Non Window Glasses* and the forth cluster contains the remaining members of classes *1, 2* and *3*, which represent the *Window Glasses*.

$k = 5, 6$: In both of these cases, the clusters are similar to the previous case (when $k = 4$). The additional clusters mainly subdivide the members from *true* cluster 1, which is the largest *true* cluster in the dataset.

The SPECT dataset, which is an imbalanced dataset, is also an example of a situation where spectral clustering fails to find the clusters according to the *true* clusters. While natural clusters are based on the similarity between the objects, the *true* clusters do not show such natural groupings. Therefore, the clusters obtained from this dataset often contain members from both of the *true* clusters.

The Ecoli dataset has 5 *true* clusters. Below are brief descriptions on each of the solutions. In this case, the *true* clusters also do not exactly correspond to the natural clusters. Therefore, members from the same *true* cluster may reside in two different cluster. Below, we provide the clusters that contain members from the dominating *true* clusters.

$k = 2$: In this case, the algorithm divides the objects into two groups: one which contains the Ecoli proteins mostly residing at the area called the *Inner Membrane without Signal Sequence (IM)* and the *Inner Membrane, Uncleavable Signal Sequence (IMU)*. The second group contains Ecoli members that are located in the *Cytoplasm (CP)*, *Perisplasm (PP)* as well as the locations listed with *Others*.

$k = 3$: When $k = 3$, the Ecoli that are located at the *Cytoplasm (CP)* are isolated in a separate cluster from the rest of the members. The other two clusters contain the members that are located in *Inner Membrane without Signal Sequence (IM)* and *Inner Membrane, Uncleavable Signal Sequence (IMU)*, respectively. The third cluster contains the members from class *Perisplasm (PP)* and *Others.*

$k = 4$: The solution is very similar to the previous one except that some of the members of class *Others* are placed in a separate cluster.

$k = 5$: Four of the clusters are very similar to those achieved, when the number of clusters was set to 4. The fifth cluster is comprised of some of the members from the largest class where Ecoli proteins are located at the *Cytoplasm (CP).*

7.5 Chapter Summary

In this chapter, we presented the performance of the eight distance measures for numeric datasets. The experiments are performed on two different versions of the spectral clustering algorithm and tested against six datasets. The results indicated that the *Pearson Correlation* distance (*COR*) and the *Angular distance* (*COS*), frequently performed better than the rest of the distance measures. Both of these distance measures calculate the *relative* distance between the objects. In contrast, the *Mahalanobis* distance measure scored relatively lower than the rest of the distance measures. We noticed that this distance measure is very sensitive to the extreme points. However, the performance may be improved by selecting the value for σ carefully. On the other hand, the results from *Euclidean* distance, *Manhattan* distance, *Minkowski* distance, *Chebyshev* distance, and *Canberra* distance measure obtained very similar scores. For these measures, the scores are moderate, in that they scored lower than the best ones but better than the *Mahalanobis* distance. In contrast to the *Angular* distance and the *Pearson Correlation* distance measure, these distance measures calculate the *absolute distance* between the objects. Our results suggest that the *Angular* distance and the *Pearson Correlation* distance may be a suitable choice when spectral clustering is applied on the numeric datasets. In addition, according to our results, the *Mahalanobis* distance measure may not be a suitable choice.

The next chapter provides a summary of this thesis. We also discuss several directions in which this study may be furthered.

Chapter 8

Conclusion

In this chapter we provide a summary of our work as presented in this thesis. We discuss the results and contributions of our work. We also suggest several directions in which this research may be pursued further.

8.1 Discussion

Cluster analysis is an exploratory data analysis technique, where the data is divided into groups or clusters, based on the similarity between the objects. Recently, a graph-based cluster analysis method, called the spectral clustering algorithm, has become popular in the research community. The spectral clustering algorithms originated from the area of *graph partitioning*. The algorithm takes the similarity matrix, constructed from the pair-wise similarity between the objects, as input. Next, the Laplacian matrix is constructed from the similarity matrix and the algorithm manipulates the eigenvector(s) and eigenvalue(s) of this Laplacian matrix to find the clusters. One of the advantages of using a spectral clustering algorithm is that the spectral algorithm is not sensitive to any particular data type. As such, datasets with numeric, categorical, binary, or mixed data types may work equally well with the spectral clustering algorithm. One needs only to convert the dataset into a similarity matrix. This may be accomplished by using existing proximity measures or by introducing new measures. As with any cluster analysis methods, the selection of the proximity measure is very crucial to the spectral algorithms. While, the *Euclidean distance* is the most popular choice for numeric datasets, it may not be the most suitable distance measure [19]. Therefore, in this thesis, we performed a comparative and exploratory study on several proximity measures, to compare their performance on the spectral clustering algorithm.

We considered three different data types. Apart from numeric data, our work included datasets with binary and mixed variables. For each of the data types a number of existing proximity measures were considered. For all our experiments, we closely followed the four fundamental steps of cluster analysis, as discussed in Chapter 2. In our experiments, we first constructed the similarity matrices for a given dataset using each of the proximity measures. The spectral algorithm was then applied to each of the similarity matrices. The result from the spectral clustering algorithm was assessed with the help of the external cluster evaluation measures. In order to compare the performance of the proximity measures, we compared the scores obtained from the cluster evaluation measures. We then analyzed the results to address the probable causes that might have affected the performance of the distance measures. The contributions from our experiments are stated below:

Performance of proximity measures for binary data: In this study, we analyzed the performance of six similarity coefficients for binary data. These included: 1) *Sokal and Sneath*, 2) *Jaccard*, 3) *Simple Matching Coefficient*, 4) *Rogers and Tanimoto*, 5) *Russell and Rao*, and 6) *Czekanowski*. We found that the performance of the *Russell and Rao* similarity coefficient is often poor when compared to the rest of the coefficients. We also noticed, as may be expected, that the performance of each of the coefficients depends on the datasets used in the experiments. The issues that most affected the performance of the similarity measures were, the weights given to the positive and negative matches, the ratio of zeros and ones in the dataset, and in a number of cases, the weights given to the number of unmatched pairs.

Performance of proximity measures for mixed data: We evaluated the performance of two coefficients for mixed data: 1) the *Gower similarity coefficient* and 2) the *Laflin similarity coefficient.* Our results showed that, on average, the *Gower* coefficient performed better than the *Laflin* coefficient. However, the difference between the scores is very low. We also found that under certain constraints the *Gower similarity coefficient* and *Laflin similarity coefficients* performed similarly. We observed that, for our experiments, the performance of the *Gower* coefficient and the *Laflin* coefficient vary mostly due to the distance measures for the numeric attributes.

Performance of proximity measures for numeric data: In our study, we analyzed the performance of eight proximity measures for numeric data. The distance measures that are considered in this study are: 1) *Pearson Correlation Coefficient*, 2) *Angular Distance*, 3) *Euclidean Distance*, 4) *Manhattan Distance*, 5) *Minkowski Distance*, 6) *Chebyshev Distance*, 7) *Canberra Distance*, and 8) *Mahalanobis Distance*. Our results suggest that the *Mahalanobis distance* might not be a suitable choice for spectral clustering algorithms. We observed, that when combined with spectral algorithm, the clusters formed by this distance measure often produced imbalanced clusters. In this case, the objects in smaller clusters contained the objects with lower *degrees*. We also noticed that the rest of the seven distance measures were not as sensitive as the *Mahalanobis distance* measure in terms of the extreme points. Our results from the evaluation measures also indicated that the *Pearson Correlation Coefficient* and the *Angular Distance* measure, most frequently performed well. We also noticed that the *Euclidean distance* often scored moderately in our experiments. In a majority of the cases in our study, the difference between the performances occurred due to the shape of the clusters formed by each distance measure.

Comparison of the splitting methods for SM(NCut) algorithm: Our results showed that the selection of the splitting methods for the *SM(NCut)* algorithm, also affect the performance of the *SM(NCut)* spectral algorithm. We compared five splitting methods in this study. They are: *Split Zero*, *Split Mean*, *Split Median*, *Split NCut1*, and *Split NCut2*. The results from our study showed that the performances of the splitting methods are comparative. The *Split Zero* and *Split Mean* are among the splitting methods that most often outperformed, in terms of the evaluation measures. Moreover, in terms of the composition of clusters, these two splitting methods often produced robust clusters. We also found that the clusters produced from the *Split NCut1* and *Split NCut2* splitting method might produce imbalanced clusters, where one cluster contains a relatively smaller number of objects than the other cluster.

8.2 Future Work

The work presented in this thesis may be furthered in several other directions. Firstly, the experiments may be extended to consider clustering algorithms other than the spectral clustering methods used in this study. It would be interesting to explore the performance of various proximity measures on different clustering algorithms and to evaluate how these measures vary for each of the algorithms. More precisely, it would be interesting to see whether the conclusions drawn from our study persist for other clustering algorithms in a similar manner. Recall from Chapter 2 that the clustering solutions may also be evaluated by applying the internal evaluation measures. Therefore, in future, a similar study using the internal cluster evaluation measures may also be performed to explore the performance of the proximity measures.

Another direction for future work is to consider more diverse selections of datasets. For example, in this study we considered the datasets from the UCI repository. In the future, steps could be taken to investigate the performance of these proximity measures on synthesis datasets, datasets with noise and outliers, or large datasets with high dimensionality. However, large datasets with high dimensionality have many additional issues to consider. For high dimensional datasets, these proximity measures may not perform as per our expectation. As mentioned in Chapter 2, with high dimensions, the data may become sparse and the distance computed from these measures may not capture the difference properly. In such cases, a different set of proximity measures may be required to deal with the problem of high dimensionality.

This research may be significantly extended for linked datasets. A linked dataset considers the attribute's information and also includes the information about the relationships in between the objects in the datasets. In recent years, a new area called *Link Mining* [25], which concentrates particularly on linked data, has become popular in the research community. Examples of linked datasets include bibliographic data, web data, social networking data, and epidemiological data. However, there are several issues that may make the task more complex. For instance, linked datasets are usually large and the similarity matrix constructed from these datasets will therefore also be very large. As such, the calculation of the eigenvalues and eigenvectors of a very large matrix will likely also be time and space consuming. These issues need special attention when linked datasets are concerned.

Appendix A

Acronyms and Mathematical Terms

A.1 Acronyms

CAN CANberra distance for numeric data.

CHEB CHEByshev distance for numeric data.

COR pearson CORrelation distance for numeric data.

COS angular distance or COSine distance for numeric data.

CZE CZEkanowski similarity coefficient for binary data.

EUC EUClidean distance for numeric data.

GOWER GOWER (dis)similarity coefficient for mixed data.

JAC JACcard similarity coefficient for binary data.

LAFLIN LAFLIN's similarity coefficient for mixed data.

MAH MAHalanobis distance for numeric data.

MAN MANhattan distance for numeric data.

MIN MINkowski distance for numeric data.

NJW(K-means) Ng, Jordan, and Weiss's spectral algorithm with K-means.

SAS Sokal And Sneath similarity coefficient for binary data.

SIM SIMple matching coefficient for binary data.

SM(NCut) Shi and Malik's Normalized Cut spectral algorithm.

RAR Russell And Rao similarity coefficient for binary data.

RAT Rogers And Tanimoto similarity coefficient for binary data.

A.2 Mathematical Terms

Eigenvectors and Eigenvalues Let A be a $n \times n$ square matrix. Then, λ is an eigenvalue of A if there exists a non-zero vector x such that, $Ax = \lambda x$. Here, x is called an eigenvector of A corresponding to eigenvalue λ. [9]

Covariance Matrix The matrix of *covariances* between elements of a vector.

Covariance In statistics, covariance [72] is the measure of how much two random variables change together. If two variables tend to vary together (that is, when one of them is above its expected value (mean), then the other variable tends to be above its expected value too), then the covariance between the two variables will be positive. On the other hand, if one of them is above its expected value and the other variable tends to be below its expected value, then the covariance between the two variables will be negative.

Bibliography

[1] AGRAWAL, R., GEHRKE, J., GUNOPULOS, D., AND RAGHAVAN, P. Automatic subspace clustering of high dimensional data for data mining applications. *ACM SIGMOD Record 27*, 2 (1998), 94 – 105.

[2] AIELLO, M., ANDREOZZI, F., CATANZARITI, E., ISGRO, F., AND SANTORO, M. Fast convergence for spectral clustering. In *ICIAP '07: Proceedings of the 14th International Conference on Image Analysis and Processing* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 641 – 646.

[3] ASUNCION, A., AND NEWMAN, D. UCI Machine Learning Repository, 2007.

[4] BACH, F. R., AND JORDAN, M. I. Learning spectral clustering. In *Neural Information Processing Systems* (2003), MIT Press.

[5] BACH, F. R., AND JORDAN, M. I. Learning spectral clustering, with application to speech separation. *The Journal of Machine Learning Research 7* (2006), 1963 – 2001.

[6] BEIL, F., ESTER, M., AND XU, X. Frequent term-based text clustering. In *KDD '02: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2002), ACM, pp. 436 – 442.

[7] BERKHIN, P. Survey of clustering data mining techniques. Technical Report, Accrue Software, 2002.

[8] BREW, C., AND IM WALDE, S. S. Spectral clustering for German verbs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (July 2002), pp. 117 – 124.

[9] Carter, T. A., Tapia, R. A., and Papakonstantinou, A. Linear algebra, An introduction to linear algebra for pre-calculus students. `http://ceee.rice.edu/Books/LA/index.html`.

[10] Cheng, D., Kannan, R., Vempala, S., and Wang, G. On a recursive spectral algorithm for clustering from pairwise similarities. Technical Report MIT-LCS-TR-906, Massachusetts Institute of Technology, Cambridge, US, 2003.

[11] Chung, F. R. K. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92).* American Mathematical Society, February 1997.

[12] Costa, I. G., de Carvalho, F. A. T., and de Souto, M. C. P. Comparative study on proximity indices for cluster analysis of gene expression time series. *Journal of Intelligent and Fuzzy Systems: Applications in Engineering and Technology 13*, 2-4 (2002), 133 - 142.

[13] Cunningham, P. Dimension reduction. Technical Report UCD-CSI-2007-7, University College Dublin, 2007.

[14] Demsar, J. On the Appropriateness of Statistical Tests in Machine Learning. *The 3rd workshop on Evaluation Methods for Machine Learning, In conjunction with ICML 2008*.

[15] Donath, W. E., and Hoffman, A. J. Lower bounds for the partitioning of graphs. *IBM J. Research and Development 17*, 5 (September 1973), 420 - 425.

[16] Dunham, M. H. *Data Mining: Introductory and Advanced Topics.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[17] Espinola, R. P., and Ebecken, N. F. F. On extending F-measure and G-means metrics to multi-class problems. In *DATA MINING VI - Data Mining, Text Mining and Their Business Applications* (2005), C. A. B. A. Zanasi and N. F. F. Ebecken, Eds., WIT press, pp. 25 - 34.

[18] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)* (1996), pp. 226 - 231.

[19] Everitt, B. S. *Cluster Analysis*, 2nd ed. Edward Arnold and Halsted Press, 1980.

[20] FIEDLER, M. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal 23*, 98 (1973), 298 - 305.

[21] FILZMOSER, P., GARRETT, R., AND REIMANN, C. Multivariate outlier detection in exploration geochemistry. *Computers and Geosciences 31*, 5 (2005), 579–587.

[22] FISCHER, I., AND POLAND, J. New methods for spectral clustering. Technical Report IDSIA-12-04, IDSIA, 2004.

[23] FISCHER, I., AND POLAND, J. Amplifying the block matrix structure for spectral clustering. Technical Report IDSIA-03-05, IDSIA, 2005.

[24] FODOR, I. K. A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, Lawrence Livermore National Laboratory, Center for Applied Scientific Computing, 2002.

[25] GETOOR, L. Link Mining: A new data mining challenge. *SIGKDD Explorations 5*, 1 (2003), 84 - 89.

[26] GHOSH, J. Scalable Clustering. In *The Handbook of Data Mining, Chapter 10*, N. Ye, Ed. Lawrence Erlbaum Associates, 2003, pp. 247 - 277.

[27] GOLUB, G. H., AND LOAN, C. F. V. *Matrix Computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[28] GUHA, S., RASTOGI, R., AND SHIM, K. CURE: An efficient clustering algorithm for large databases. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 1998), ACM, pp. 73 - 84.

[29] GUHA, S., RASTOGI, R., AND SHIM, K. ROCK: A robust clustering algorithm for categorical attributes. In *ICDE '99: Proceedings of the 15th International Conference on Data Engineering* (Washington, DC, USA, 1999), IEEE Computer Society, p. 512.

[30] GUO, H., AND VIKTOR, H. Learning from imbalanced data sets with boosting and data generation: the DataBoost-IM approach. *ACM SIGKDD Explorations Newsletter 6*, 1 (2004), 30 - 39.

[31] HAMPRECHT, F., SCHNÖRR, C., AND JÄHNE, B. *Pattern Recognition 29th DAGM Symposium, Heidelberg, Germany, September 12-14, 2007: Proceedings.* Springer, 2007.

[32] HAN, J., AND KAMBER, M. *Data Mining: Concepts and Techniques, 2nd Ed.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[33] HEINZ, G., PETERSON, L. J., JOHNSON, R. W., AND KERK, C. J. Exploring relationships in body dimensions. *Journal of Statistics Education 11*, 2 (2003).

[34] HINNEBURG, A., AND KEIM, D. A. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)* (1998), pp. 58 – 65.

[35] JAIN, A. K., MURTY, M. N., AND FLYNN, P. J. Data clustering: A review. *ACM Computing Surveys 31*, 3 (1999), 264 – 323. `www.citeseer.ist.psu.edu/jain99data.html`.

[36] JAPKOWICZ, N., AND SHAH, M. *Performance Evaluation for Classification A Machine Learning and Data Mining Perspective (in progress): Chapter 6: Statistical Significance Testing.*

[37] KANNAN, R., VEMPALA, S., AND VETA, A. On clusterings: Good, Bad and Spectral. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science* (Washington, DC, USA, 2000), IEEE Computer Society.

[38] KARYPIS, G., HAN, E.-H. S., AND KUMAR, V. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer 32*, 8 (1999), 68 – 75.

[39] KAUFMAN, L., AND ROUSSEEUW, P. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley-Interscience, 2005.

[40] KIM, Y., STREET, W. N., AND MENCZER, F. Feature selection in data mining. 80 – 105.

[41] KUBAT, M., HOLTE, R. C., AND MATWIN, S. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning 30*, 2 - 3 (1998), 195 – 215.

[42] KURUCZ, M., BENCZUR, A., CSALOGANY, K., AND LUKACS, L. Spectral clustering in telephone call graphs. In *WebKDD/SNA-KDD '07: Proceedings of the 9th*

WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis (New York, NY, USA, 2007), ACM, pp. 82 - 91.

[43] LAFLIN, S. Laflin's general coefficient. Website, 1998. `http://www.cs.bham.ac.uk/~slb/courses/Taxonomy/Taxonomy02.html`.

[44] LAROSE, D. T. *Discovering Knowledge in Data: An Introduction to Data Mining.* Wiley-Interscience, 2004.

[45] LEE, S., AND VERRI, A. *Pattern Recognition With Support Vector Machines: First International Workshop, Svm 2002, Niagara Falls, Canada, August 10, 2002: Proceedings.* Springer, 2002.

[46] LUXBURG, U. A tutorial on spectral clustering. *Statistics and Computing 17*, 4 (2007), 395 - 416.

[47] MEILA, M., AND SHI, J. Learning segmentation by random walks. In *Neural Information Processing Systems* (2000), pp. 873 - 879.

[48] MEILA, M., AND SHI, J. A random walks view of spectral segmentation. In *International Conference on Artificial Intelligence and Statistics (AISTAT)* (2001).

[49] NEVILLE, J., ADLER, M., AND JENSEN, D. Spectral clustering with links and attributes. Technical Report 04-42, University of Massachusetts Amherst, 2004.

[50] NG, A. Y., JORDAN, M. I., AND WEISS, Y. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems* (2001), T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14.

[51] NG, R. T., AND HAN, J. CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering 14*, 5 (2002), 1003 - 1016.

[52] PACCANARO, A., CASBON, J. A., AND SAQI, M. A. Spectral clustering of protein sequences. *Nucleic Acids Research 34*, 5 (2006), 1571–1580.

[53] PAGE, D. KDD cup 2001 - Genes dataset. Website. `http://pages.cs.wisc.edu/~dpage/kddcup2001/`.

[54] PEDRYCZ, W. *Knowledge-Based Clustering: From Data to Information Granules.* Wiley-Interscience, 2005.

[55] ROMESBURG, C. *Cluster Analysis for Researchers.* Lulu. Com, 2004.

[56] SHAKHNAROVICH, G. Introduction to machine learning, Lecture 25. Website, 2006. `http://www.cs.brown.edu/courses/csci1950-f/lectures/lecture25.pdf`.

[57] SHEIKHOLESLAMI, G., CHATTERJEE, S., AND ZHANG, A. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *VLDB '98: Proceedings of the 24th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 1998), Morgan Kaufmann Publishers Inc., pp. 428 - 439.

[58] SHI, J., AND MALIK, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8 (2000), 888 - 905.

[59] SOKAL, R., AND SNEATH, P. *Principles of Numerical Taxonomy.* WH Freeman, 1963.

[60] STATSOFT, I. Electronic Statistics Textbook. World Wide Web electronic publication, 2007. `http://www.statsoft.com/textbook/stathome.html`.

[61] STEINBACH, M., KARYPIS, G., AND KUMAR, V. A comparison of document clustering techniques. KDD Workshop on Text Mining, 2000.

[62] STRANG, G. *Linear Algebra and Its Applications*, 4th ed. Thomson Brooks / Cole, February 2006.

[63] TAN, P., STEINBACH, M., AND KUMAR, V. Introduction to Data Mining, 2005.

[64] TEKNOMO, K. Similarity Measurement. Website, 2007. `http://people.revoledu.com/kardi/tutorial/Similarity/`.

[65] UNIVERSITY, P. WordNet 3.0. Website, 2006. `http://wordnet.princeton.edu/perl/webwn?s=clustering`.

[66] VERMA, D., AND MEILA, M. Comparison of spectral clustering methods. Technical Report 03-05-01, University of Washington, Computer Science and Engineering, 2003.

[67] WANG, C., JUN LI, W., DING, L., TIAN, J., AND CHEN, S. Image segmentation using spectral clustering. In *ICTAI '05: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 677 - 678.

[68] WANG, W., YANG, J., AND MUNTZ, R. R. STING: A statistical information grid approach to spatial data mining. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases* (San Francisco, CA, USA, 1997), Morgan Kaufmann Publishers Inc., pp. 186 - 195.

[69] WEBB, A. R. *Statistical Pattern Recognition, 2nd Edition.* John Wiley & Sons, October 2002.

[70] WHITE, S., AND SMYTH, P. A spectral clustering approach to finding communities in graph. *SIAM Data Mining Conference* (2005).

[71] WIKIPEDIA. Class (biology) — wikipedia, the free encyclopedia, 2008. [Online; accessed 28-November-2008].

[72] WIKIPEDIA. Covariance — wikipedia, the free encyclopedia, 2008. [Online; accessed 28-November-2008].

[73] WIKIPEDIA. Exponential function — wikipedia, the free encyclopedia, 2008. [Online; accessed 28-November-2008].

[74] WIKIPEDIA. Mahalanobis distance — wikipedia, the free encyclopedia, 2008. [Online; accessed 28-November-2008].

[75] WIKIPEDIA. P-value — wikipedia, the free encyclopedia, 2009. [Online; accessed 28-January-2009].

[76] WITTEN, I. H., AND FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques*, 2 ed. Morgan Kaufmann, 2005.

[77] WU, Z., AND LEAHY, R. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 11 (1993), 1101 - 1113.

[78] XIONG, H., STEINBACH, M., TAN, P., AND KUMAR, V. HICAP: Hierarchical Clustering with Pattern Preservation. In *Proceedings of the 4th SIAM International Conference on Data Mining* (2004), pp. 279 - 290.

[79] XIONG, H., WU, J., AND CHEN, J. K-means clustering versus validation measures: a data distribution perspective. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2006), ACM New York, NY, USA, pp. 779 - 784.

[80] Xu, R., and Wunsch II, D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks 16*, 3 (May 2005), 645 – 678.

[81] Yin, X., Han, J., and Yu, P. S. Cross-relational clustering with user's guidance. In *KDD '05: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (New York, NY, USA, 2005), ACM, pp. 344 – 353.

[82] Yu, L., and Liu, H. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings, 20th International Conference on Machine Learning (ICML-2003)* (2003), vol. 2, pp. 856–863.

[83] Zhang, T., Ramakrishnan, R., and Livny, M. BIRCH: An efficient data clustering method for very large databases. *ACM SIGMOD Record 25*, 2 (1996), 103 – 114.