Categorical and Fuzzy Ensemble-Based Algorithms for Cluster Analysis

Bridget Nicole Manning

Categorical and Fuzzy Ensemble-Based Algorithms for Cluster Analysis

by

Bridget Nicole Manning

Bachelor of Science
College of Charleston, 2010

Master of Science
College of Charleston, 2013

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Statistics

College of Arts and Sciences

University of South Carolina

2020

Accepted by:

David B. Hitchcock, Major Professor

John Grego, Committee Member

Karl Gregory, Committee Member

John Rose, Committee Member

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

Dedication

I dedicate this dissertation to my nieces, nephews, and whatever future children I may have. I did this for you and everyone like you. As I write this dissertation, the veil covering racial tension and disparities within our nation has been removed. The media has portrayed people that look like you and me in a way that I never want you to believe. I want you to see yourselves as the beautiful queens and handsome kings that you are. Know no matter how big you may grow, that you have an even greater power inside of you. A power that will allow you to be whoever and whatever you want to be. Keep God first in all you do and go after your dreams. Don't be afraid to try or fail. I have done both many times, and I promise you the only things I regret are those I didn't try. I am an African-American woman that started as a poor black girl from rural South Carolina. I had no college fund and no idea what I was doing in life. To be honest, I still don't. I follow my heart and pray things work out. I cry and have days that I want to quit, but I keep going for you. You are just like me. The blood that runs in my veins runs in yours, and I pray that one day when you are older, you let these words guide you when I can't: If a little black girl from Dillon, South Carolina could earn a Ph.D. in Statistics, you too can do anything! I dedicate this dissertation, the culmination of 5.5 years of my life, to you. I pray you see it not as pieces of paper or a random book, but instead as a symbol of what your blood and your lineage can achieve.

I love you with all of my heart!

Acknowledgments

I would first like to thank God for giving me the ability, strength, knowledge, and provisions necessary to complete this academic journey! I would also like to thank my patient advisor, Dr. David B. Hitchcock, for all the edits, encouragement, and all-around guidance in this process. I thank my committee members, Drs. John Grego, John Rose, and Karl Gregory, for your suggestions that have improved the quality of my research, and I thank my family of friends at USC who went through the craziness with me: Zichen, Chunling, Ennan, Liu, Ethan, Brandy, Ryan, Geophrey, and Zach.

To Wilma Sims and Drs. Candace Porter, Shamarick Paradise, and Tracey Gunter, my modern-day Hidden Figures: You have no idea how much you have motivated and inspired me. I thank you for the career advice, life chats, and most importantly for showing me by example that African-American women, too, have a place in the world of statistics!

To my inner-circle, Chelsey, Eric, and Brandon: Thank you so much for going through this process with me! Thank you for always believing in me and having my back in everything life brings. I love you so much, and thank God for bringing you into my life!

Lastly and most importantly, I thank my family for your sacrifices, support, and love through this journey. I say thank you to my mother and father, Peter and Julia Caldwell; my bonus parents, Larry and Jackie McQueen, Charles McLain, and John Brown; my sisters, Scherylle, Michelle, Meagan, La'Tavia, and Gina; my brothers, Julius, Braxton, Shkari, and Keyion.

Abstract

This dissertation focuses on improving multivariate methods of cluster analysis. In Chapter 3 we discuss methods relevant to the categorical clustering of tertiary data while Chapter 4 considers the clustering of quantitative data using ensemble algorithms. Lastly, in Chapter 5, future research plans are discussed to investigate the clustering of spatial binary data.

Cluster analysis is an unsupervised methodology whose results may be influenced by the types of variables recorded on observations. When dealing with the clustering of categorical data, solutions produced may not accurately reflect the structure of the process that generated them. Increased variability within the latent structure of the data and the presence of noisy observations are two issues that may be obscured within the categories. It is also the presence of these issues that may cause clustering solutions produced in categorical cases to be less accurate. To remedy this, in Chapter 3, a method is proposed that utilizes concepts from statistics to improve the accuracy of clustering solutions produced in tertiary data objects. By pre-smoothing the dissimilarities used in traditional clustering algorithms, we show it is possible to produce clustering solutions more reflective of the latent process from which observations arose. To do this the Fienberg-Holland estimator, a shrinkage-based statistical smoother, is used along with 3 choices of smoothing. We show the method results in more accurate clusters via simulation and an application to diabetes.

Solutions produced from clustering algorithms may vary regardless of the type of variables observed. Such variations may be due to the clustering algorithm used, the initial starting point of an algorithm, or by the type of algorithm used to produce such solutions. Furthermore, it may sometimes be of interest to produce clustering solutions that allow observations to share similarities with more than one cluster. One method proposed to combat these problems and add flexibility to clustering solutions is fuzzy ensemble-based clustering. In Chapter 4 three fuzzy ensemble-based clustering algorithms are introduced for the clustering of quantitative data objects and compared to the performance of the traditional Fuzzy C-Means algorithm. The ensembles proposed in this case, however, differ from traditional ensemble-based methods of clustering in that the clustering solutions produced within the generation process have resulted from supervised classifiers and not from clustering algorithms. A simulation study and two data applications suggest that in certain settings, the proposed fuzzy ensemble-based algorithms of clustering produce more accurate clusters than the Fuzzy C-Means algorithm.

In both of the aforementioned cases, only the types of variables recorded on each object were of importance in the clustering process. In Chapter 5 the types of variables recorded and their spatial nature are both of importance. An idea is presented that combines applications to geodesics with categorical cluster analysis to deal with the spatial and categorical nature of observations. The focus in this chapter is on producing an accurate method of clustering the binary and spatial data objects found in the Global Terrorism Database.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The world is at a unique place in its history. We are able to track vital statistics, exercise regimens, and smoking habits using compact wearable technology. We are able to participate in forums with those across the country on common topics of interests from the latest evolution in sports to the latest makeup releases. We are able to communicate via telephone to those in other parts of the world and instantly post and receive feedback on almost anything. There is no denying the current state of affairs is much more advanced than anything we have ever seen in the history of the human species. Yet, these advances bring challenges. How do we process this information? How do we find patterns and meaning within this free and readily available information? How can a business like Samsung use the data it collects from smart watches to study the lifestyle habits of those who use their device? How can those in the biological life sciences find similarities in those patients fighting a common disease using only what information is collected in surveys completed in the doctors' office, and how can a clinical psychologist determine if there may be more levels of depression than what may be shown in the current DSM-5 (APA, 2013) using reader comments left on his/her personal blog for those dealing with depression? Is it possible that the technological advances of the last few years can help us find the answer?

Over the years, researchers from various fields such as business and the life sciences have used traditional methods such as cluster analysis to find meaning in their data with the hope that this additional information could then be used to serve other needs. For example, Windgassen et al. (2018) argues that cluster analysis can be used to help create complete clinical profiles for mental health patients that can later be used to develop treatment regimen; while Grabowski, Herbeck, and Poon (2018) provided a review of how the clustering of genetic sequences has been used in epidemiology to identify clusters within HIV that can then be studied to learn about transmission rates in sub-Saharan Africa. While these examples display some of the more commonly perceived uses of cluster analysis, it has also been applied in another context that may not seem quite as natural. Since the start of the 21st century, cluster analysis has also been applied in text analytics. In this context, it has been referred to as text mining.

Text mining is a term used to define machine learning methods by which informative meaning can found in works of text. These "texts" may include books, documents, or even webpages. The term can be thought of as an "umbrella" term that refers to a host of methods including that of text-based clustering (Everitt et al., 2011). Just like cluster analysis, text-based clustering has been employed in several fields. Some examples include information retrieval where search engines employ such methods to return the desired search results or in business where corporations like Amazon have utilized it to cluster comments from its customers in order to make recommendations (Huang, 2008).

This indicates that while cluster analysis is an older methodology, with one of the earliest clustering methods dating back to the the early 19th century, it still has important uses. With the ease of access to the World Wide Web and current technological methods that make data collection much more efficient and affordable, it has become a pressing issue to be able to classify it. The questions now become, "Is it possible to make these traditional methods of multivariate classification more efficient? Could classical methodologies in statistics be combined with modern ideas to help meet this end goal?"

In this dissertation, we present multivariate methods of cluster analysis that can be used to achieve this end. More specifically we present a method to create more accurate clusters in tertiary data, introduce three fuzzy ensemble-based algorithms for cluster analysis, and explore a future method in which spatial binary data may be clustered using applications to geodesics. Improvements to the traditionally used methods may impact those in fields ranging from business and hospitality to those in the social and biological life sciences fields. But perhaps the answer to these questions lie in not one field, but in the combination of many. To motivate this point, consider the history of cluster analysis.

1.1 History

From pre-computer times until the present day, the field of cluster analysis has evolved. One of the earliest methods developed began as a means of solving a problem in anthropology. In 1909, Jan Czekanowski, a Polish anthropologist motivated by the need to classify human remains, created a classification method that would precede many modern-day methods of cluster analysis (Soltysiak and Jaskulski, 1999).

To motivate Czekanowski's method, first consider the traditional framework used to denote a set of n multivariate data objects, $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n})$, each with P observed measurements. Such data can be stored in a $n \times P$ matrix like $\mathbf{X}$ as shown below:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1P} \\ \vdots & & \ddots & \\ x_{n1} & x_{n2} & \cdots & x_{nP} \end{bmatrix}$$

In Czekanowski's proposed method, the matrix $\mathbf{X}$ would be created using the bones' measurements as the entries with the stipulation that variable types be the same.

In the next step, the average pairwise dissimilarity between objects were calculated using the Manhattan distance (1.1)

$$d_{ij} = \frac{1}{P}\sum_{p=1}^{P} |x_{ip} - x_{jp}|, \tag{1.1}$$

where d_{ij} denotes the average distance between objects i and j, x_{ip} denotes the ith object's measurement on the pth variable and x_{jp} denotes the jth object's measurement on the pth variable (Soltysiak and Jaskulski, 1999). Next the average distances were stored in a square matrix such that the measurements along the diagonals were all 0 (since each object is identical to itself) and the off diagonals were positive (as it is distance). After this the matrix was rearranged such that the objects closest to each other were placed side-by-side as neighbors, and, in an effort to make cluster visualization easier, graphical objects (like circles and squares) were used to denote the distance between observations (Soltysiak and Jaskulski, 1999). For visualization, the smallest distances were shown with a completely black square and the largest distance with a completely white square. Object pairings with an average distance that fell between the smallest and largest average distances of all possible pairings were shown by a graphical image whose size was reflective of this distance (Soltysiak and Jaskulski, 1999). In other words, the more similar a pair of objects are, (i.e., the smaller distance they have) the larger the graphical image. In the final step, the matrix was rearranged once more so that the diagonals contained zero and the smallest distances were shown closest to the diagonal as can be seen in Figure 1.1.

Though Czekanowski's proposed method was limited by the fact that the variables had to be of the same type (or transformed), and that only the Manhattan metric was used, it was still considered a breakthrough method in its time as it could be used with incomplete data and was fairly simple to use (Soltysiak and Jaskulski, 1999). In fact, Soltysiak and Jaskulski (1999) argues that the most challenging aspect was reorganizing the diagram in such a manner that clusters could be clearly identified; while one of its biggest advantages was that it allowed users to visualize the potential

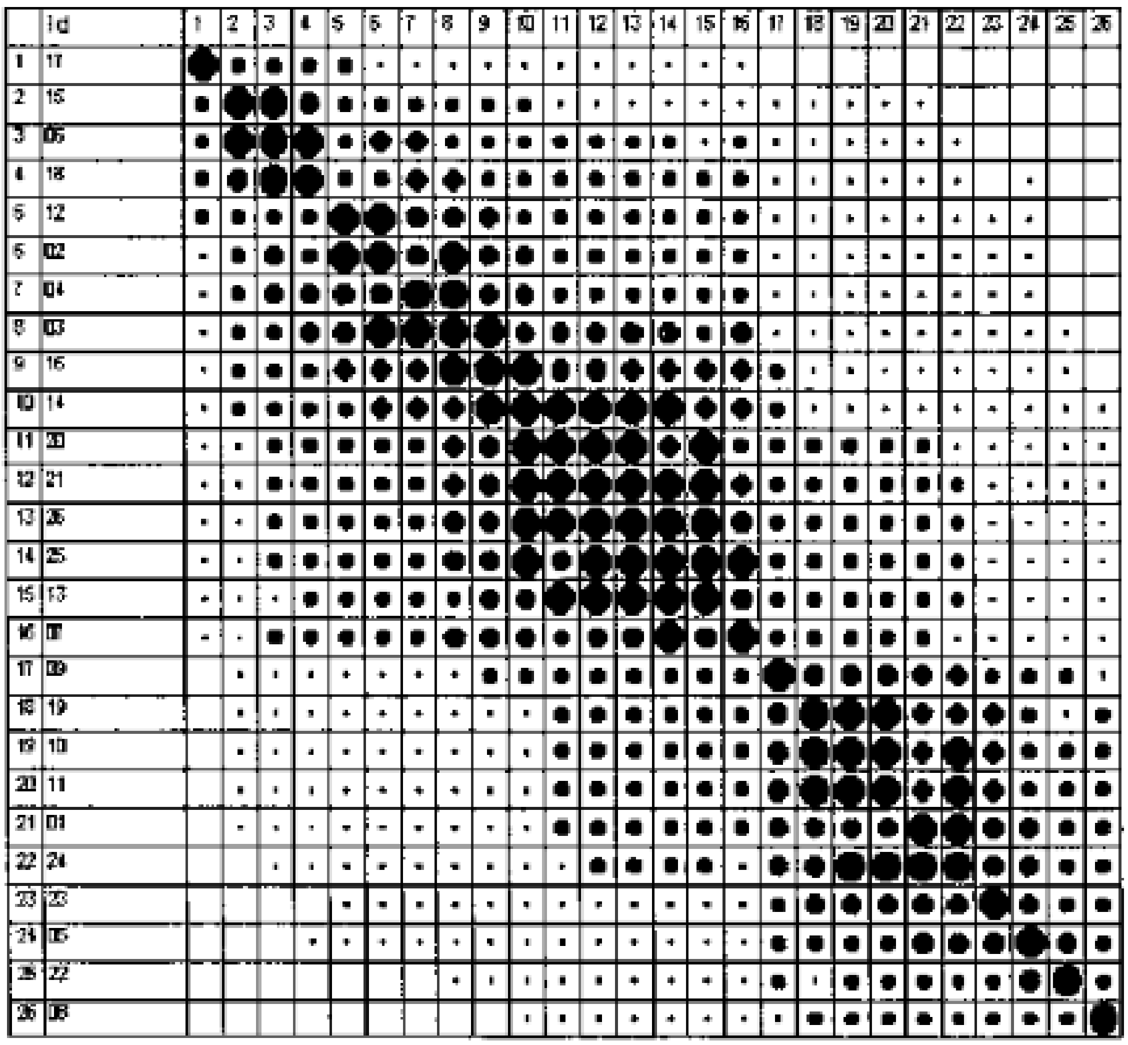
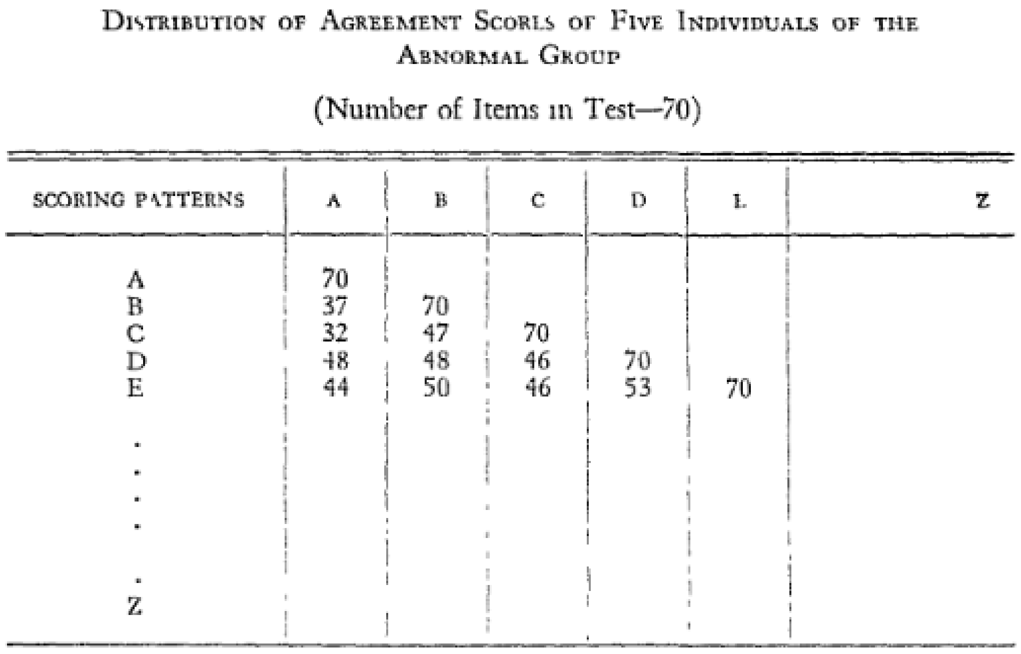


Figure 1.1 Czechanowski matrix showing distance between object pairs using dots. The relative size of the black dots indicates closeness with bigger dots indicating closer relationship and smaller dots showing greater distance between objects. (Soltysiak and Jaskulski, 1999)



Distribution of Agreement Scores of Five Individuals of the Abnormal Group

(Number of Items in Test—70)

Scoring Patterns	A	B	C	D	E	Z
A	70					
B	37	70				
C	32	47	70			
D	48	48	46	70		
E	44	50	46	53	70	
.						
.						
.						
.						
.						
Z						

Figure 1.2 Example from Zubin (1938) depicting degree of agreement. In the diagram, the body shows the number of questions on which each pair of individuals agree.

clustering structure of the observations in two dimensions—a feat that had not been possible before.

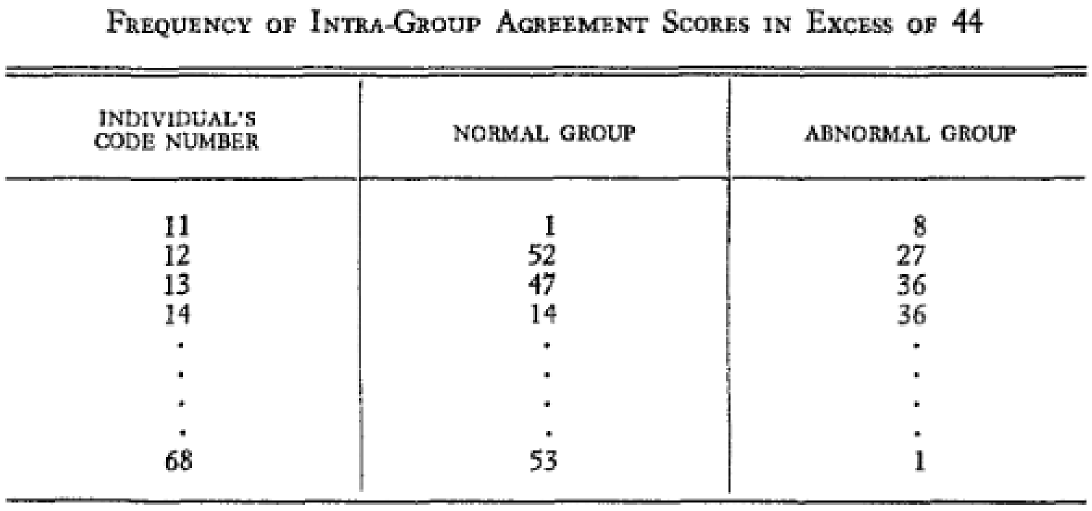
Decades later, psychologist Zubin (1938) used the survey responses of 136 people to 70 questions on a "Personal Inquiry Form" to produce groups of like-minded individuals. The participants in this survey included 68 schizophrenics and 68 controls matched on various characteristics including gender and age. Zubin's method involved three steps:

1. Determine the average degree of agreement for each pair of individuals.
2. Sort individuals based on similarity scores.
3. Determine the actual patterns of agreement.

Within the first step, to find the average degree of agreement, Zubin counted the number of items on which a pair of individuals agreed. In other words, the number of items upon which their answers were the same. He organized this in a table such that the diagonals had a value of 70 (as the diagonals represent a person's degree of agreement with themselves) and the off-diagonals denoted the degree of agreement for the pairs. For an example of this, see Figure 1.2.

At the next step, individuals within the schizophrenic group were compared to other individuals within the schizophrenic group (the same was done for the control group). For this portion, Zubin looked at the number of times each individual in a specific group agreed with his/her fellows in at least 45 items. (There is no discussion of why 45 was chosen.) To see how this was tabulated, see Figure 1.3. These individuals were then grouped into two groups—those schizophrenics who agreed with each other in at least 45 items and the controls who agreed with each other in at least 45 items. Next, the number of items of agreement was reduced to 40. These individuals were grouped together—one group for schizophrenics and a second for the control. The process continued until there were a total of 8 subgroups. The last step of Zubin's method focused on determining what the actual agreement patterns were; however, this was a qualitative interpretation of the cluster analysis rather than a part of the algorithm.

Zubin's method appears to be completely different than Czekanowski's in its implementation, but it has a similar purpose in that each researcher wanted to be able to find sub-groupings within their data. For Zubin this was in terms of like-mindedness (using categorical variables); whereas for Czekanowski it was actual measures related to human bones (using quantitative variables). Regardless, each method met its goal in finding objective methods upon which sub-groupings could be formed. These methods preceeded modern day cluster analysis.



FREQUENCY OF INTRA-GROUP AGREEMENT SCORES IN EXCESS OF 44

INDIVIDUAL'S CODE NUMBER	NORMAL GROUP	ABNORMAL GROUP
11	1	8
12	52	27
13	47	36
14	14	36
.	.	.
.	.	.
.	.	.
.	.	.
68	53	1

Figure 1.3 Example borrowed from Zubin (1938) depicting intragroup similarities. The table shows the number of fellow individuals in which a particular individuals agrees on at least 45 items.

Around this same time period and into the early 1940s, the first predecessors of the modern computer were being built. Before long computers would be available in many places. This allowed for the creation of more robust and complex methods of cluster analysis. Not only could cluster analysis be done, it could be done more quickly and efficiently than what was possible using Czekanowski's and Zubin's methods.

Since the advent of the computer, hundreds of clustering algorithms have been proposed by computer scientists and researchers in a variety of fields. Therefore, it is not surprising that in our present "big data" age, that again we rely on advances in computers. Today, in fact, many more methods of cluster analysis have been invented to deal with this special case of clustering.

Since trends in data collection show no signs of slowing and classification continues to be important, it is unlikely that cluster analysis will lose relevance anytime soon. Instead, it seems that the more pertinent issue is to be able to come up with more efficient methods. Modern methods that are motivated by not just computer scientists and statisticians, but quite possibly by practitioners in many diverse areas including anthropology and psychology.

1.2 Classification

Modern methods of classification continue to apply ideas similar to those proposed by Czekanowski and Zubin to find natural partitions that may exist in a set of observations. However, the goal in classification may not always be to create sub-groupings. Sometimes, the goal of the researcher may be to make predictions of class membership for unknown observations. Other times, the intent may be to make better predictions based on sub-groupings that can be formed utilizing some of the information in the data. These three scenarios form the three branches of classification used in modern-day statistics: supervised, unsupervised, and semi-supervised learning.

To conceptualize each scenario, first assume there are n multivariate observations in a dataset each with P variable measurements recorded upon them. Further suppose k of these observations include a label denoting class membership, and the remaining $n - k$ observations do not. With supervised learning, only the k labeled observations are used to build a model that can be used to make class predictions for the $n - k$ unlabeled observations. In unsupervised learning, all n observations are treated as if they are unlabeled and sub-groupings are formed, if possible, using the multivariate observations on each object or some proximity measure. In this scenario, the information about labels is ignored. Lastly, in semi-supervised learning, as in unsupervised, all n objects are used—the k observations with membership labels and the $n - k$ that are not labeled. However, in this setting, the information about the labeling is not ignored as in the unsupervised case. As each classification method has its advantages and disadvantages that make them useful in certain settings or inappropriate in others, each is equally important to the field of statistics.

Within supervised learning, as aforementioned, the primary goal is to make predictions about the the $n-k$ unlabeled observations based solely on the k observations that are labeled. To do this, the original dataset is segmented into two parts the training set and the testing set. The training set *supervises* the learning. The result of training is a model or output learner that can then be used to predict the output for the testing set. When the output is quantitative, the learners are usually referred to as some type of regression; whereas, when the output is qualitative, the learners are usually deemed classifiers.

Popular learners used in each setting differ based on the underlying assumptions about the data. Some examples of statistical models that may be used to predict quantitative output include the ordinary least squares regression model or the weighted least squares regression model. Alternatively, one may use a generalized linear model like Poisson regression, for example, when the output is a particular type of quantitative data. (In the Poisson case, the output is count data). On the other hand, when the predictions are qualitative, learners may be trained using logistic regression, decision trees, support vector machines, or the k-nearest neighbor classifier amongst others.

In unsupervised learning, as aforementioned, all n observations are treated as unlabeled. This method may be used for exploratory data analysis and is helpful to find natural groupings that may be present in the observations. Cluster analysis is an example of this.

In semi-supervised learning, all n observations are used to build a learner. Common methods used in semi-supervised learning included self-training, co-learning, expectation-maximization algorithms with generative models, and graph-based modelling. We begin our discussion with a more in depth introduction to cluster analysis in Chapter 2.

Chapter 2

Cluster Analysis

Cluster analysis is an unsupervised learning method. In computational fields, it may be included within the umbrella term data mining. When it involves text, it may be dubbed text mining, in psychology it may be called Q-analysis, and in biology it may be described as numerical taxonomy (Everitt et al., 2011). Regardless of the name, the goal of cluster analysis is usually to segment data observations into groups, called "clusters", such that observations within the same cluster are more similar to each other than they are to any other observations in another cluster. Though a plethora of algorithms for cluster analysis exist, they usually can be subdivided into a few different methods. Five of the more popular methods include: hierarchical, partitioning, fuzzy, model-based, and density-based methods. As no one class of clustering methodology is always best, each method has its advantages and disadvantages that may make them more applicable in one setting rather than in another. In this section, there is a discussion of each of the aforementioned methods of cluster analysis and a brief discussion of the advantages and disadvantages of each. There is also a discussion of methods by which the goodness of clustering solutions can be evaluated, and a discussion about commonly used measures to define similarity and dissimilarity under various settings.

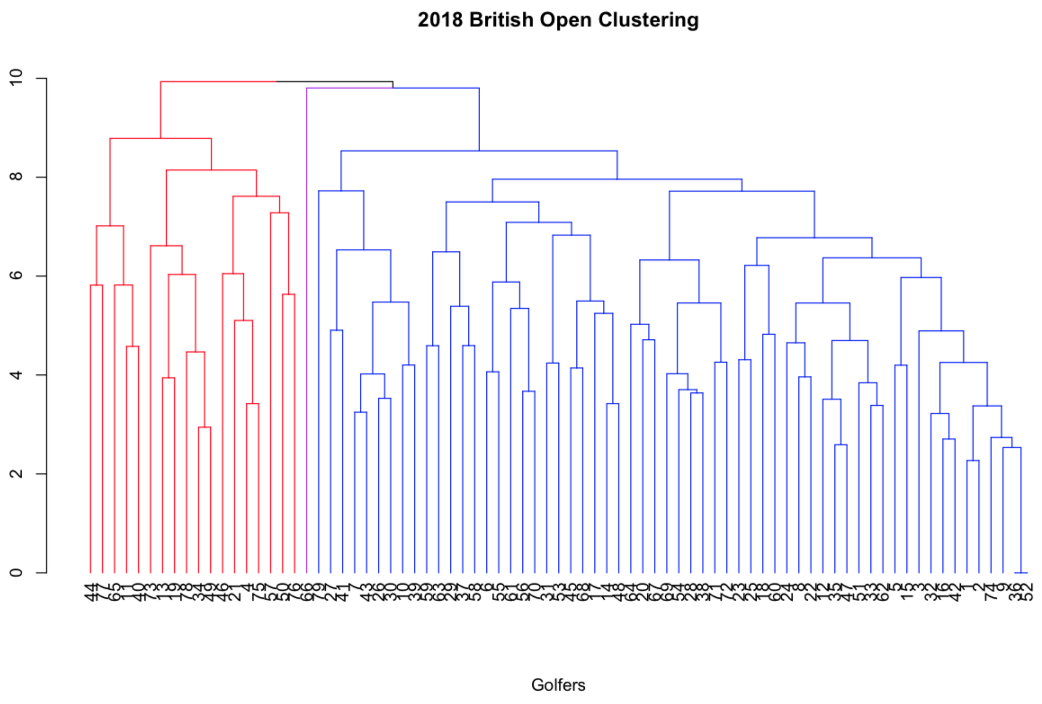


Figure 2.1 A dendrogram created from the fourth round of the 2018 British Open where golfers were clustered based on scores at each of 18 holes.

2.1 Popular Clustering Methods

2.1.1 Hierarchical Methods

Hierarchical methods of cluster analysis build a hierarchy within the observations. These methods may be agglomerative or divisive with the output of each being shown, usually, with a dendrogram (example shown in Figure 2.1). The dendrogram depicts the hierarchical structure of the data. In the case of agglomerative methods, this hierarchy is built from the bottom up; whereas, in divisive methods, this hierarchy is built from the top to the bottom. In the final stage of the clustering, the dendrogram is cut wherever necessary to obtain the desired number of clusters. For an example of this, consider Figure 2.1. The colors show where the dendrogram could be cut to produce 3 clusters within the golfers. This serves as both an advantage and a disadvantage of hierarchical methods. On the one hand, dendrograms are fairly easy to understand. However, employing a hierarchical method will always result in a hierarchy being built in the data—even when the latent structure of the data does not support this assumption. To illustrate how these dendrograms (hierarchies) are created, first consider the approach taken by an agglomerative method.

Algorithms of the agglomerative type begin with every observation being perceived as a singleton cluster. At each level, pairs of observations are merged together based on some measure of similarity or dissimilarity. This process repeats at every successive level until only one cluster exists (at the top) that contains all observations. Common methods of defining the similarity that is used to help determine what observations should be merged at each iteration include single linkage, complete linkage, average linkage, and Ward's method. For notational purposes, assume there are two clusters denoted as A and B. Furthermore assume that there are two data objects $i \in A$ and $j \in B$, then each method can be defined as shown below with d_{ij} representing the distance between object i and j:

1. Single Linkage

$$d(A, B) = \min_{i \in A, j \in B} d_{ij}$$

2. Complete Linkage

$$d(A, B) = \max_{i \in A, j \in B} d_{ij}$$

3. Average Linkage

$$d(A, B) = \frac{1}{|A|\,|B|} \sum_{i \in A} \sum_{j \in B} d_{ij}$$

4. Ward's method

$$d(A, B) = \sum_{i \in A} d_{i,\bar{A}} + \sum_{j \in B} d_{j,\bar{B}} - \sum_{u \in A \cup B} d_{u,\bar{u}}$$

Ward's method of clustering merges the two pairs of clusters that result in the least increase in the within-cluster variability as measured by the sum of squared error (SSE) on each iteration of the algorithm. In the notation above, $d_{i,\bar{A}}$ refers to the SSE between the observations in cluster A and their mean, $d_{i,\bar{B}}$ refers to the SSE between the observations in cluster B and their mean. $d_{u,\bar{u}}$ then refers to the SSE obtained by merging of clusters A and B. The goal is by minimizing the within-cluster variability, the between cluster variability is maximized.

Agglomerative methods of clustering can be implemented in several software packages. Two functions in R that can implement such hierarchical clustering include the `hclust` function of the `stats` package, or the `agnes` function located in the `cluster` package (Maechler et al., 2018). The `PROC CLUSTER` procedure can be used in `SAS`® as well.

For divisive methods, the process is reversed. Observations begin in one single cluster. At each successive level of the hierarchy, the most heterogeneous observations are removed based on a similarity or dissimilarity measure (typically the same ones that may be used in an agglomerative approach). At the end of the process, each observation is a singleton cluster. Divisive methods of clustering are not as commonly used; however, they too can be implemented in R. One way to do this is through the use of the `diana` function in the `cluster` package (Maechler et al., 2018).

2.1.2 Partitioning-Based Methods

Partitioning-based methods of cluster analysis divide a set of observations into k clusters at once. They require that the number of clusters, k, to be created be assigned a priori usually with the added stipulation that each observation can only be a member of one partition (fuzzy methods do not have this stipulation). The two most popular algorithms that implement partitioning methods of clustering are the K-Means (MacQueen, 1967) and K-Medoids (Rousseeuw and Kaufman, 1987) methods.

In the K-Means algorithm, an initial random set of k observations are chosen as starting centroids. At the next step, the distance from each observation to each centroid is computed using the Euclidean distance. Observations are then grouped together with the closest centroid. Next, the overall mean is calculated for all cluster members and this value becomes the new centroid for the cluster. (For the K-Means algorithm the centroid need not be a cluster member). The last two steps repeat until convergence occurs.

The K-Medoids algorithm is implemented in a similar manner as the K-Means algorithm with the exceptions that Euclidean distance does not have to be used in this case and the centroid no longer represents the cluster means. Instead it is called a "medoid" and is the cluster member whose distance is closest to the overall mean of its cluster members.

When comparing the two algorithms, the K-Medoids method of cluster analysis is noted as being more robust in the presence of outliers than the K-Means algorithm (Sanse and Sharma, 2015). However, the K-Means algorithm tends to be more popular. Furthermore, the K-Medoids method can take a proximity measure as an input; whereas, original observations have to be used within the K-Means algorithms in order to calculate the Euclidean distance. Both algorithms are found in various software packages. For example, the K-Means algorithm can be implemented in R using the `kmeans` function in the `stats` package or the `PROC FASTCLUS` function in SAS. The K-Medoids algorithm can be implemented using the `pam` function in the `cluster` package (Maechler et al., 2018) in R.

2.1.3 Fuzzy Methods

We now discuss the subfield of cluster analysis known as fuzzy clustering. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T$ denote a set of multivariate objects to cluster, and let $C = \{C_1, C_2, \ldots, C_c\}$ denote a set of c disjoint and non-empty partitions of $\boldsymbol{X}$ such that

$\bigcup_{j=1}^{c} C_j = \boldsymbol{X}$. Now define $u_j(\boldsymbol{x}_k) = u_{kj}$ where u_{kj} denotes the grade of membership for the kth object in the jth cluster. In the special case in which

$$u_{kj} = \begin{cases} 1 & \text{if } \boldsymbol{x}_k \in C_j \\ 0 & \text{otherwise} \end{cases}$$

$\{u_j : j = 1, 2, \ldots, c\}$ denotes a hard partition of $\boldsymbol{X}$. In the general case in which $u_{kj} \in [0, 1]$, $\{u_j : j = 1, 2, \ldots, c\}$ is considered a fuzzy partition of $\boldsymbol{X}$. Fuzzy methods of clustering remove the assumption that data objects belong to solely one cluster. Instead, fuzzy-based methods allow for data objects to share similarity with more than one cluster.

Arguably the most popular fuzzy method of clustering is the Fuzzy C-Means algorithm (Bezdek, Ehrlich, and Full, 1984). For this algorithm in particular there are the additional stipulations that $\sum_{j=1}^{c} u_{kj} = 1$ and $0 \leq \sum_{k=1}^{n} u_{kj} \leq n$. The algorithm then seeks to minimize the objective function

$$Q_v(\boldsymbol{U}, \boldsymbol{m}) = \sum_{j=1}^{c} \sum_{k=1}^{n} u_{kj}^{v} d(\boldsymbol{x}_k, \boldsymbol{m}_j)$$

where $\boldsymbol{U}$ denotes a $n \times c$ matrix that contains the membership grades u_{kj}, v denotes the fuzzifier that controls the amount of fuzziness in the clustering solution, and $d(\boldsymbol{x}_k, \boldsymbol{m}_j)$ refers to the distance between object k and the center of cluster j, $\boldsymbol{m}_j$. (Details of this algorithm are explained in more detail in Section 4.1). The `cmeans` function of the `e1071` package can be used to implement the Fuzzy C-Means algorithm in R.

2.1.4 Model-Based Methods

Model-based clustering assumes data objects have arisen from a mixture of probability distributions. Through such a method, they are able to determine the number of clusters that exist in a data set, to identify clusters of various shapes, and handle noisy observations. To motivate this method of clustering, consider a population with G subpopulations and a set of multivariate data objects $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$. In model-based clustering each of the G subpopulations is assumed to have arisen from an underlying distribution $f_k(\mathbf{x};\boldsymbol{\theta})$ for $k = 1, 2, ..., G$, where $\boldsymbol{\theta}$ is a vector of unknown parameters. In this context, the problem of clustering observations amounts to assigning objects to clusters in a manner that maximizes the complete likelihood function,

$$L(\mathbf{x};\boldsymbol{\theta}) = \Pi_{k=1}^{G} f_k(\mathbf{x};\boldsymbol{\theta}). \tag{2.1}$$

where $f_k(\mathbf{x};\boldsymbol{\theta})$ denotes the probability distribution of the kth subpopulation. One aspect commonly explored with model-based clustering is the determination of the appropriate probability model or models that should be used to define clusters in differing scenarios. Typically this is done based on some assumed features of the clusters. For example, Fraley and Raftery (2002) discuss how when the underlying distributions are believed to be a mixture of Gaussian distributions with a covariance structure defined as $\mathbf{\Sigma}_k = \sigma^2\mathbf{I}$, then the shape of the clusters will be spherical having the same size and shapes, as considered by Ward (1963). When the covariance structure can be defined as $\mathbf{\Sigma}_k = \mathbf{\Sigma}$ for all clusters, then the shape of the clusters will be ellipsoidal having the same size, shape, and orientation (Friedman and Rubin, 1967).

Banfield and Raftery (1993) proposed a method to change this so that additional features could be found from the data, other families of distributions could be fit, and noise could be implemented in the model. In particular, they proposed using the eigenvalue decomposition of the covariance matrix to determine the features (like orientation, size, and shape) of the clusters that will be the same across all clusters and those that may differ across clusters.

To discuss this, the notation from Banfield and Raftery (1993) is borrowed. Consider the eigenvalue decomposition of the covariance matrix, $\mathbf{\Sigma}_k$=$D_k\mathbf{\Lambda}_k D_k^T$, where D_k denotes the matrix of eigenvectors of $\mathbf{\Sigma}_k$ and $\mathbf{\Lambda}_k = \boldsymbol{\lambda}_k A_k$ denotes the square matrix with the eigenvalues of $\mathbf{\Sigma}_k$ on the diagonal. Banfield and Raftery (1993) argues that feature specification can be controlled by making changes to different matrices from this decomposition. In particular, D_k determines the orientation of each cluster, A_k determines the shape of each cluster, and $\boldsymbol{\lambda}_k$ determines the size of each cluster (Banfield and Raftery, 1993).

In the case of the MVN$(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ distributions the complete likelihood function, can be written as

$$L(\mathbf{x}; \boldsymbol{\theta}) = \Pi_{i=1}^{n} \Pi_{k=1}^{G} \left(2\pi \boldsymbol{\Sigma}_k\right)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k)\right\}$$

However, as aforementioned, the distributions need not be Gaussian. To this end, Banfield and Raftery (1993) propose a method to extend the model-based framework to non-Gaussian distributions. For a starting point, consider the complete data likelihood function as shown in Equation (2.1). The paper recognizes that in general, Equation (2.1) is enough for most non-Gaussian cases; however, they propose the use of a local parameterization to fit other distributions. In this case, consider $\mathbf{z_i} = D_k(\mathbf{x}_i - \boldsymbol{\mu}_k)$, where D_k are matrices and let $\mathbf{z}_i$ have the density denoted by $g_k(\mathbf{z}_i; \boldsymbol{\theta})$. From this framework then, much like Equation (2.1) was maximized, one can instead maximize this likelihood with $g_k(\mathbf{z}_i; \boldsymbol{\theta})$ in place of $f_k(\mathbf{x}; \boldsymbol{\theta})$.

Thus model-based clustering solves several issues that may plague other methods of clustering. In this method, since each cluster refers to a subpopulation within a probability model, and noise or outlying values are implemented as a component of the model, issues involving the best number of clusters to be fit are solved using criteria such as the Bayesian Information Criterion.

A popular algorithm that implements this type of clustering is the EM algorithm (Dempster, Laird, and Rubin, 1977). This algorithm tries to find the appropriate mixture distributions of the latent variables from which the data objects are assumed to have arisen. To do this, there are two steps:

1. Expectation Step: The conditional expectation for the complete log-likelihood given the observed data and current estimate for the parameters is evaluated.

2. Maximization Step: New estimates for the parameters are calculated by maximizing the expected log-likelihood found on the expectation step.

These two steps repeat until the log-likelihood no longer changes. The `Mclust` function of the `mclust` package (Scrucca et al., 2016) and `PROC MI` function in `SAS` can be used to implement model-based clustering via the EM algorithm. However, the method employed in these packages assumes the clusters have arisen from a finite mixture of Gaussian distributions.

2.1.5 Density-Based Methods

Density-based methods of cluster analysis try to find clusters based on the density in a region. Pertinent to any density-based method of cluster analysis are the concepts of *neighborhood*, *density* and *connectivity*. To illustrate these concepts, consider a data point, $\mathbf{x}$. In these methods, the ϵ-neighborhood refers to the ball that could be formed of radius $\epsilon > 0$ with $\mathbf{x}$ at the center. Density, then, often refers to the number of observations or the fraction of observations (sometimes referred to as the mass) contained within the ϵ ball centered at $\mathbf{x}$. Perhaps the most popular algorithm implementing this method of clustering is that of the “Density Based Spatial Clustering of Applications with Noise” (Ester et al., 1996) often abbreviated as DBSCAN. The different types of connectivity will be explained in terms of this particular algorithm.

In the DBSCAN algorithm, the previously mentioned observation $\mathbf{x}$, may or may not be considered a *core point*. DBSCAN requires an ϵ value and a *minpt* value to be specified a priori. The value specified in the *minpt* command refers to the minimum number of observations that must be contained within the ϵ-neighborhood of $\mathbf{x}$ for it to be considered a *core point*. Any observation within the ϵ-neighborhood of $\mathbf{x}$ then would be considered *density reachable* (one type of connectivity) from $\mathbf{x}$. To motivate an additional method of connectivity, suppose there exist two other points, $\mathbf{x}_2$ and $\mathbf{x}_3$, such that $\mathbf{x}_2$ is density reachable from $\mathbf{x}$ but $\mathbf{x}_3$ is not. If $\mathbf{x}_3$ is density reachable from $\mathbf{x}_2$, then $\mathbf{x}$ and $\mathbf{x}_3$ are said to be *density connected*. Through these concepts then, the DBSCAN can also recognize points on a cluster boundary and points that are noise. Any observation whose ϵ-neighborhood consist of less than the *minpt* specified, but is density connected to at least one other point is called a *border point*. If a point were not density connected to another point, then it would be considered a *noise* point.

Density-based methods of clustering are able to identify clustering formations that are of different shapes and sizes and often employ a method to address issues such as boundary point and noise identification; however for the DBSCAN algorithm, because it requires the ϵ and *minpt* values to be given a priori without a proven method of doing such clustering solutions may vary with different inputs (Albalate and Minker, 2011). This particular clustering algorithm can be implemented using the `dbscan` function in the `dbscan` package of R.

2.1.6 Comparison of Methods

Each of the different methods for clustering observations are useful in various settings. However, there are some cases in which one method may outperform another. In particular, partitioning-based algorithms like the K-Means, K-Medoids, and K-Modes algorithms have been found to be more computationally efficient than hierarchical methods when the data set is large (see e.g., Sanse and Sharma (2015) or Huang (1997a)). However, the K-Means algorithm is known to perform poorly in the presence of noisy or outlying data. These methods also identify clustering solutions that are locally optimum and not necessarily globally optimum (Sanse and Sharma, 2015) which may be a problem in some settings. Hierarchical methods, on the other hand tend to be less sensitive to outliers and have the added advantage that a solution with any number of clusters can be obtained by cutting the dendrogram wherever one deems necessary. These algorithms also tend to be able to handle different types of observations; however, they are not able to scale well, which is the reason they are often not used in the clustering of very large data sets (Sanse and Sharma, 2015). Hierarchical methods also are limited in that once a cluster is merged or divided, the operation can not be undone.

Model-based clustering is another method that is robust to noisy data or outlying values. It also has the advantage that the number of clusters can be found objectively unlike many of the partitioning-based methods. However, model-based clustering methods tend to be more complex in nature (Sanse and Sharma, 2015). Lastly, density-based methods of clustering, too, perform well in the presence of noisy data. However, many of these algorithms tend to break down in high-dimensional data sets. Algorithms of this type have the additional advantage, like the model-based methods, that they do not require the number of clusters to be known a priori. They also can fit clusters of arbitrary shapes (Sanse and Sharma, 2015).

2.2 Cluster Goodness Evaluation

As cluster analysis is an unsupervised method of classification, it is important to measure the quality of a clustering solution. Consequently, much study over the years has focused in ways to adequately measure cluster goodness. Cluster goodness can be thought of as a method of evaluating a clustering solution in terms of some specified criterion. Methods that may be used vary depending on whether class labels are available and the type of clustering algorithm being used. In this section, commonly used relative, external, and internal measures will be discussed.

2.2.1 Relative Cluster Evaluation

Relative methods of evaluating clustering solutions compare the results of two opposing methods. Two commonly used relative measures include the Rand (Rand,

1971) and Adjusted Rand Index (Hubert and Arabie, 1985). The formulas for each are given below, as well as additional information about their range of possible values. To motivate each method, consider Table 2.1 from McNicholas (2017), which shows the cross-tabulation of two clustering solutions from two different clustering algorithms. In the table, A denotes the number of pairs of data objects placed in the same group by both methods, B denotes the number of pairs of data objects put in different groups by method one and in the same group by method two (C denotes the reverse). D denotes the number of pairs of data objects placed in different groups by both methods. Using this table, the measures are defined as follows:

Table 2.1 Cross-Tabulation of Two Clustering Partitions

	Partition One		
Partition Two	Same Group	Different Group	Totals
Same Group	A	B	$A+B$
Different Group	C	D	$C+D$
Totals	$A+C$	$B+D$	N

1. Rand Index (RI):
$$RI = \frac{A+D}{N}$$
The RI takes values inclusively between 0 and 1 with smaller values indicating less agreement between two clustering solutions and a larger value indicating higher agreement. A value of 0 denotes no agreement and 1 denotes complete agreement between two clustering solutions.

2. Adjusted Rand Index (ARI):
$$ARI = \frac{N(A+D) - [(A+B)(A+C) + (C+D)(B+D)]}{N^2 - [(A+B)(A+C) + (C+D)(B+D)]}$$

The ARI (Hubert and Arabie, 1985) is a correction to the RI introduced to adjust for inflation resulting from chance agreement between two partitions (McNicholas, 2017). The ARI can take on negative values or positive values as large as 1 with values closer to 1 denoting higher agreement between two partitions. A value of 0 denotes partitions equivalent to matching by chance; while negative values denote worse than chance classification (see e.g. McNicholas (2017)).

2.2.2 External Cluster Evaluation

Sometimes the actual class labels may be known. This is usually the case in artificial settings such as simulation studies or in simple examples. In this case, cluster goodness is an evaluation of how a particular clustering solution compares to these known labels. Some commonly used measures in this particular setting include the entropy and purity.

1. Entropy:

$$e_j = -\sum_{l=1}^{L} \log_2 (p_{jl}) \tag{2.2}$$

Equation (2.2) denotes the entropy for a particular cluster, j. Here, l denotes a known class label with a total of L known classes, and p_{jl} denotes the probability that an object classified as l is in cluster j.

$$E(C) = -\sum_{j=1}^{K} \frac{n_j}{n} e_j \tag{2.3}$$

Equation (2.3) denotes the total entropy for a clustering solution, C. In this equation, e_j denotes the entropy of cluster j, K denotes the total number of clusters, n_j denotes the the total number of items in cluster j, and n denotes the total number of data objects.

Entropy values are bounded below by 0 and bounded above by $\log(n)$ for finite clusters. Since entropy is a measure of heterogeneity, lower values indicate better clustering solutions (Albalate and Minker, 2011).

2. Purity:

$$P_j = \max_l (p_{jl}) \tag{2.4}$$

Equation (2.4) denotes the purity for an individual cluster, j. In this equation, l and p_{jl} are defined the same as for cluster entropy.

$$P(C) = -\sum_{j=1}^{K} \frac{n_j}{n} P_j \tag{2.5}$$

Equation (2.5) denotes the purity for a particular clustering solution, C, with j, K, n_j, and n defined as before. P_j denotes the purity for cluster j.

Measures of purity are bounded between 0 and 1 with values closer to 1 indicating a better clustering solution. A purity value of 1 indicates that the clustering solution aligns perfectly with the known class labels and a value of 0 denotes no agreement.

2.2.3 Internal Cluster Evaluation

When class labels are not known and a practitioner is interested in evaluating one particular clustering solution, cluster goodness may be evaluated using an internal measure. These measures usually look to maximize the inter-cluster separation and minimize the intra-cluster separation based on some criteria. They are commonly used to address the appropriate number of clusters that should be fit from the data. Some of the more commonly used measures for this purpose include the cluster silhouette width (Rousseeuw, 1987), Gap statistic (Tibshirani, Walther, and Hastie, 2001), cophenetic correlation, and the Davies-Bouldin index (Davies and Bouldin, 1979).

1. Silhouette width:

$$sil(\mathbf{x}_i) = \frac{\bar{b}(i) - \bar{a}(i)}{\max(\bar{a}(i), \bar{b}(i))} \tag{2.6}$$

Equation (2.6) refers to the silhouette width of the ith data object, where $\bar{a}(i)$ denotes the average distance between the ith object and all other objects in its respective cluster and $\bar{b}(i)$ refers to the average distance between the ith object and all other objects in the nearest cluster.

$$sil(C_j) = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} sil(\mathbf{x}_i) \tag{2.7}$$

$$sil(K) = \frac{1}{K} \sum_{j=1}^{K} sil(C_j) \tag{2.8}$$

Equation (2.7) refers to the silhouette width of cluster C_j, and Equation (2.8) refers to the average silhouette width of a particular clustering solution with K total clusters. The value of K that maximizes the average silhouette width shown in Equation (2.8) is considered to be the optimum number of clusters.

2. Gap Statistic:

$$W_k = \sum_{j=1}^{K} \frac{1}{2n_j} \sum_{i,i' \in C_j} D(i,i') \tag{2.9}$$

Equation (2.9) measures the inter-cluster distance of a particular clustering solution with K total clusters. In this equation, C_j refers to the jth cluster, n_j denotes the number of objects in the jth cluster, and $D(i,i')$ refers to the distance between the ith and i'th object in cluster C_j.

$$Gap(K) = E(\log(W_K)) - \log(W_K) \tag{2.10}$$

Equation (2.10) gives the actual Gap statistic that is used in practice to determine the optimum number of clusters, K, to be fit within the data. Albalate and Minker (2011) mention that often the $E(\log(W_K))$ is computed using a Monte-Carlo simulation, thus the optimum value of K is denoted as

$$\min_K : Gap(K) \geq Gap(K+1) - s_{K+1}$$

where s_{K+1} is a factor of the Monte-Carlo simulated standard error of W_{Kb}.

3. Cophenetic Correlation:

$$c = \frac{\sum_{i<i'}(D(i,i') - \bar{x})(t(i,i') - t)}{\sqrt{[\sum_{i<i'}(D(i,i') - \bar{x})^2][\sum_{i<i'}(t(i,i') - \bar{t})^2]}} \tag{2.11}$$

The c in Equation (2.11) refers to the cophenetic correlation for use with hierarchical clustering solutions. In this equation, $D(i,i')$ denotes the Euclidean distance between the ith and i'th data objects and $t(i,i')$ denotes the distance at which the ith and i'th objects are joined together (height on the dendrogram). $\bar{x}$ refers to the average distance (across all observations) between the ith and i'th object, and $\bar{t}$ refers to the average of the $t(i,i')$ values. This measure is commonly used in biostatistics to measure the quality of cluster-based models created for DNA-sequences (Saraçli, Doğan, and Doğan, 2013). A higher value of c denotes a better dendrogramatic solution.

4. Davies-Bouldin (DB) Index:

$$DB(K) = \frac{1}{K}\sum_{j=1}^{K}\max_{k\neq j}\frac{D(\bar{C_j}) + D(\bar{C_k})}{D(C_j, C_k)} \tag{2.12}$$

Equation (2.12) refers to the DB index for a particular clustering solution with K total clusters. $D(\bar{C_j})$ and $D(\bar{C_k})$ refer to the average distance between all cluster members and the centroid of the jth and kth cluster, respectively; whereas $D(C_j, C_k)$ refers to the distance between the centroids of the two clusters. This index may be used to determine the optimum number of clusters to be fit, by choosing the value of K that minimizes the DB index.

2.3 (Dis) Similarity Measures for Cluster Analysis

In the introduction of this chapter, it was mentioned that clustering methods seek to group objects in a manner such that objects in the same group are more similar to each other than they are to objects in different groups. Consequently, the concept of similarity or dissimilarity is pertinent to many clustering methods. In this section, commonly used measures of similarity and dissimilarity are defined in the context of multivariate cluster analysis. To motivate these concepts, assume there are n data objects, each having P attribute measurements that can be stored in a $n \times P$ data matrix, $\mathbf{X}$ as shown below.

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1P} \\ \vdots & & \ddots & \\ x_{n1} & x_{n2} & \cdots & x_{nP} \end{bmatrix}.$$

In the matrix, $\mathbf{X}$, each row corresponds to one data object, $\mathbf{x}_i$, and each entry in the matrix, x_{ip}, gives the ith object's measurement on the pth variable. To begin the process of clustering, some algorithms will take the matrix, $\mathbf{X}$, containing the original data observation, as an input. Others allow for matrices containing measures of similarity or dissimilarity to be used. These may be called *proximity matrices*. To illustrate such a matrix, we first define the concept of *pairwise dissimilarity*.

Consider two observations from $\mathbf{X}$, $\mathbf{x}_i$ and $\mathbf{x}_j$. Pairwise dissimilarity may be defined as shown in Equation (2.13).

$$D(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p=1}^{P} d_p(x_{ip}, x_{jp}) \tag{2.13}$$

In Equation (2.13), $d_p(x_{ip}, x_{jp})$ refers to how different the ith object is from the jth object on the pth variable's measurement. These values, in turn, can be stored in a $n \times n$ dissimilarity matrix, $\mathbf{D}$ as shown below. For simpler notation, let d_{ij} denote $D(\mathbf{x}_i, \mathbf{x}_j)$.

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ \vdots & & \ddots & \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{bmatrix}.$$

$\mathbf{D}$ is considered a type of proximity matrix. Similarly, if instead of dissimilarity, similarities were stored in its place, this too would constitute a proximity matrix. As aforementioned, regardless of whether clustering is being performed directly using a matrix like **X** or with a proximity matrix like **D**, the definition of similarity and dissimilarity are key components to any method. Therefore, some commonly used methods will be defined next.

The most appropriate definition for similarity or dissimilarity will depend upon the subject area and the types of attributes being used in the analysis. It will also depend on whether the original data or some other measures are being used as input. Even then, the most one could hope for is to choose a more appropriate measure rather than best measure.

2.3.1 Quantitative Variable Measurements

The multivariate observations may be completely quantitative, completely qualitative or a mixture of both. When observation measurements are completely quantitative, similarity, rather than dissimilarity, is often used and is commonly defined in terms of distance. For each method we assume there are two multivariate objects $\mathbf{x}_i$ and $\mathbf{x}_j$ each with P attribute measurements. Some commonly used measures of similarity that may be used include:

1. Euclidean Distance
$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{p=1}^{P}(x_{ip} - x_{jp})^2}$$

2. Manhattan Distance
$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p=1}^{P} |x_{ip} - x_{jp}|$$

3. Pearson Correlation
$$\rho(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_p (x_{ip} - \bar{x}_i)(x_{jp} - \bar{x}_j)}{\sqrt{\sum_p (x_{ip} - \bar{x}_i)^2 \sum_p (x_{jp} - \bar{x}_j)^2}}$$

4. Minkowski's Distance
$$d(\mathbf{x}_i, \mathbf{x}_j) = (\sum_{p=1}^{P} |x_{ip} - x_{jp}|^k)^{\frac{1}{k}}$$

5. Cosine Similarity

$$s(\mathbf{x}_i, \mathbf{x}_j) = \cos(\theta) = \frac{\mathbf{x_i} \cdot \mathbf{x_j}}{||\mathbf{x_i}||\,||\mathbf{x_j}||}$$

2.3.2 Qualitative Variable Measurements

When variable measurements are completely qualitative, similarity may be defined in various manners since distance is not as natural in this setting. To discuss this, consider the simplest case of qualitative data—the binary case.

Table 3.1 (taken from Everitt et al. (2011)) shows two binary data objects $\mathbf{Y}_i$ and $\mathbf{Y}_j$ that, on each of the P variables, can have one of two outcomes denoted as 0 and 1. In the table, a denotes the number of variable measurements upon which both

Table 2.2 Pairwise Dissimilarity for Binary Data Objects $\mathbf{Y}_i$ and $\mathbf{Y}_j$

	$\mathbf{Y}_j$		
$\mathbf{Y}_i$	1	0	Totals
1	a	b	$a+b$
0	c	d	$c+d$
Totals	$a+c$	$b+d$	P

objects, Y_i and Y_j, have variable outcomes in category 1. b denotes the number of variable measurements upon which, object Y_j has an outcome in category 0, while Y_i has an outcome in category 1. c denotes the number of variable measurements upon which object Y_i has a value of 1, when Y_j has a value of 0. Lastly, d denotes the number of variable measurements upon which both objects have a categorical outcome of 0. The total number of variable measurements then, is $a + b + c + d = P$.

Using this table's notation, some similarity measures that can be used for the clustering of binary data from Everitt et al. (2011) include:

1. Matching Coefficient

$$s_{ij} = \frac{a + d}{P}$$

2. Jaccard coefficient

$$s_{ij} = \frac{a}{a+b+c}$$

3. Rogers and Tanimoto

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

4. Sneath and Sokal

$$s_{ij} = \frac{a}{a+2(b+c)}$$

5. Gower and Legendre

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

A more extensive list of similarity measures that can be used for the clustering (or classification) of binary data can be found in Choi, Cha, and Tappert (2009). In this article, the measures are defined based on a similar table as shown in Table 3.1; however, in this case, the binary outcomes of zero denote the absence of a particular feature.

When the variable measurements being used have more than two categories, most methods use similar methods that have been generalized to deal with more classes. Boriah, Chandola, and Kumar (2008) gives an extensive survey of some of these measures.

2.3.3 Mixed Variable Measurements

When data objects are mixed types, most measures of similarity are defined by combining similarities of both types. One example is that of K-Prototypes (Huang, 1997b). K-Prototypes applies a different similarity measure for each variable type—the quantitative variable measurements of each pair of observations are compared using Euclidean distance; whereas the qualitative parts are compared using the matching coefficient. For a clearer discussion of how the matching coefficient could be generalized, we take up the case for the tertiary case.

2.3.4 Similarity and Dissimilarity for Tertiary Data Objects

Similarity, for a pair of tertiary objects, typically refers to the proportion of variable measurements upon which a pair of objects have the same outcome. Thus dissimilarity for tertiary data observations is calculated as the proportion of mismatches among P tertiary attribute measurements for each pair of observations.

Table 2.3 Pairwise Dissimilarity for Tertiary Data Objects $\mathbf{Y}_i$ and $\mathbf{Y}_j$

	$\mathbf{Y}_j$			
$\mathbf{Y}_i$	1	2	3	Totals
1	a	b	c	$a+b+c$
2	d	e	f	$d+e+f$
3	g	h	i	$g+h+i$
Totals	$a+d+g$	$b+e+h$	$c+f+i$	P

Consider Table 2.3 which depicts the cross-tabulation of two objects, $\mathbf{Y}_i$ and $\mathbf{Y}_j$, from whence we demonstrate the notion of “pairwise dissimilarity" for the tertiary case. In this table, each attribute has an outcome of 1, 2, or 3 denoting its membership. Letters a, e, and i represent the total number of variables upon which both objects have outcomes denoted as 1, 2, and 3, respectively. The other letters denote the number of attributes on which objects $\mathbf{Y}_i$ and $\mathbf{Y}_j$ have different membership. For example, b denotes the number of variables for which object $\mathbf{Y}_i$ has a value of 1 and object $\mathbf{Y}_j$ has a value of 2; similarly, c denotes the number of variables for which object $\mathbf{Y}_i$ has a value of 1 and $\mathbf{Y}_j$ has a value of 3. Based on this 3×3 table, we define similarity as

$$s_{ij} = \frac{1}{P}(a + e + i)$$

and dissimilarity as

$$d_{ij} = 1 - s_{ij}.$$

Using these values a dissimilarity matrix $\mathbf{D}$ can be formed as before.

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ \vdots & & \ddots & \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{bmatrix}.$$

This particular method of defining dissimilarity could be thought of as a generalization of the matching coefficient method that is often used in the binary case or the overlap method that may be used in computer science. In this method, matches and mismatches are weighted equally. However, other methods of calculating dissimilarities for use with categorical attributes are available and vary depending upon the field of use. As aforementioned, Boriah, Chandola, and Kumar (2008) gives an extensive survey of measures that can be used along with a comparison of each method's ability to detect outliers.

2.4 Literature Review: Algorithms for Clustering Categorical Data

The problem of clustering categorical data does not appear to have been studied much until the late 1990s after being recognized as a severe deficit in the field of data mining. At this point in time, the K-Means algorithm (MacQueen, 1967) had been cited in many articles (and continues to be cited) for being one of the most computationally efficient algorithms for the processing of large data sets compared to other competing algorithms (Huang, 2008). However, since the K-Means algorithm assumes all attribute measurements are quantitative, it had not been able to be used for clustering of categorical data objects. To remedy this, Ralambondrainy (1995) proposed a method to handle this using the K-Means algorithm by first transforming all the categorical variables into binary data objects. Huang (1997a) suggests that this method, while it worked on smaller data sets, may not be feasible for use on large data sets or with high dimensional data. It was further argued in Huang and

Ng (1999) that since binary coding was used, that the centers for the K-Means were not really representative of the true underlying clusters of the data. Because of the first reasoning, Huang (1997a) proposed the use of the K-Modes algorithm. The K-Modes algorithms works similarly to the K-Means algorithm, but with a few key differences. In the K-Modes algorithm, an initial set of k data objects are randomly chosen as the modes of each cluster. Then the following three steps iterate:

1. The distance between each data object and each mode is calculated.
2. Data objects are assigned to the cluster within which the distance to the mode is shortest.
3. A new mode is selected for each cluster.

If the values for the mode (in Step 3) are the same as the previous mode, the algorithm stops. If not, the steps repeat until the modes no longer change.

The proposal of the K-Modes algorithm was and continues to be significant to the clustering of categorical data as it allows for there to still be convergence to a local minimum, as is done with K-Means, and maintains an equivalent computational efficiency (Huang, 1997a) meaning it is possible to still work in the clustering of large data sets of categorical data in sufficient time. One key difference between K-Means and K-Modes is that the centers for the K-Modes algorithm are the modes, and thus each center represents the modal categorical value of each attribute at each iteration. Another is that there is no reliance on the Euclidean distance in K-Modes. Instead, the K-Modes algorithm uses the simple matching coefficient to calculate distance. In this case, Equation (2.13) has

$$d_p(x_{ip}, x_{jp}) = \begin{cases} 0 & \text{for } x_{ip} = x_{jp} \\ 1 & \text{for } x_{ip} \neq x_{jp} \end{cases}.$$

In Huang and Ng (1999) a fuzzy K-Modes algorithm is proposed. This algorithm mimics that of fuzzy K-Means with the difference being it allows for categorical attribute measurements. Because this algorithm is a fuzzy method, data objects are allowed membership in more than one cluster. This is one of the major differences between fuzzy K-Modes and the original K-Modes algorithm. The original K-Modes is considered a hard partitioning method because data objects are allowed membership in only one cluster. The fuzzy method is considered a soft partitioning method as it assigns objects to all clusters with a corresponding membership confidence. Thus with the proposal of fuzzy K-Modes, Huang and Ng improved upon the original method via the use of the fuzzy dissimilarity matrix and the corresponding confidence values that allow for identification of boundary objects. Thus this algorithm provides a method that may be useful in many data mining settings where an issue is identification of boundary objects (Huang and Ng, 1999).

Another interesting approach to clustering of categorical data, proposed a few years later, is that of K-Histograms (He et al., 2005). This algorithm is another one that attempts to keep the desirable properties of the K-Means algorithm while allowing for categorical data. However, it differs from that of K-Means, and both versions of the K-Modes algorithms in that the centers of a cluster are represented by histograms. The authors showed in this paper, that on real data sets, the cluster partition accuracy beat that of K-Modes.

The original K-Modes and the fuzzy K-Modes algorithms, as well as the K-Histograms algorithm, are examples of partitioning-based methods of clustering. The Robust Clustering Algorithm for Categorical Attributes (ROCK) (Guha, Rastogi, and Shim, 2000) was possibly the first hierarchical method proposed for the clustering of categorical data. This agglomerative method differs from that of traditional hierarchical methods by using the concept of links instead of distance to merge data objects. The motivation for this method appears to be the clustering of market basket transactions. Consequently, it uses the number of links between a pair of objects, defined as the number of common neighbors between the two points, where objects are neighbors if there similarity is above a common threshold, to measure similarity. This measure is then used at each level of the hierarchy to determine observations that should be merged together.

Since the invention of the four aforementioned algorithms, a plethora of additional algorithms for the clustering of categorical data have been proposed. Several of the more recent algorithm proposals have been influenced by the aforementioned ones and encompass several diverse topics such as genetics, ensembles, and swarm intelligence.

Gan, Wu, and Yang (2009) proposed a genetic fuzzy K-Modes algorithm specifically for use with genetic categorical data. This algorithm amended the original five-step genetic algorithm by adding the fuzzy K-Modes algorithm in the final crossover step. By doing this, the author hoped the resulting clustering algorithms would be more closely resemblant of the true underlying structure of genetic data.

Ensemble methods of clustering have also been proposed in recent literature. These methods are based on the idea that no single clustering algorithm provides the best and most accurate result in all settings (Vega-Pons and Ruiz-Shulcloper, 2011). Therefore, ensemble methods try to combine several algorithms in two steps. The first step is the generation step in which several clustering algorithms may be run to get several different clustering solutions. Many of these methods use the K-Means algorithm here with different initial centers to create various clustering solutions. In the second step, the consensus step, the clustering solutions are combined into one final clustering partition. In Sarumathi, Shanthi, and Sharmila (2013) a survey of different ensemble methods for categorical data are provided. These methods include the Weighted Cluster Ensemble, K-Means Cluster Ensemble based on center matching, Extended Evidence Accumulation Clustering Ensemble, Squared Error Adjacent Matrix Cluster Ensemble, and Bayesian Cluster Ensemble Method, among many other methods that have been used in particular for categorical data clustering for data mining purposes. While each of these methods are based on the combination of algorithms, other algorithms that have been proposed look to nature for inspiration.

Masmoudi et al. (2015) is a clustering method proposed for binary data, in particular, data that result from survey responses that may have multiple modes. The algorithm uses swarm intelligence as its inspiration. Clustering methods inspired by swarm intelligence seek to replicate the manner in which ants in nature are able to group themselves with other similar ants and then act as one homogeneous unit. To do this, Masmoudi et al. (2015) use the ideas of pheromone rates and thresholds of affiliation to create clusters. Though this method is not considered an ensemble method, it uses the K-Means algorithm as a precursory step to determine the initial k centers for the CL-ant algorithm. Based on these a preliminary partition is formed. The next step is where the clustering process for the ant-based method begins. In this step a dynamic graph is formed that retains the original data clusters from the first step. In this graph, the original cluster represent nodes on a graph, and artificial ants are used to represent each data object. From here ants move from one cluster to another cluster by choosing a path that has a stronger pheromone concentration than its current cluster. At this point, if the similarity as measured between the ant and the centroid of the desired cluster is greater than or equal to the specified threshold of affiliation, the ant joins that cluster (Masmoudi et al., 2015). The CL-Ant method is a swarm-based method for categorical data with roots in biomimetics, but statistical methods of clustering categorical data have also been proposed.

Jacques and Biernacki (2018) presents a statistical-based method for the co-clustering of ordinal data. Ordinal data is categorical data in which the categories show order. Such data may result from surveys in which a customer is asked to list a preference. Co-clustering, then, attempts to define partitions within both, the original data set and within the variables measured upon each object, called features. Then co-clusters, or blocks, are obtained by crossing the resulting partitions. The goal is that the final blocks maintain all features and each observation is in one and only one block. Jacques and Biernacki (2018) does this using a Binary Ordinal Search Model probability distribution (Biernacki and Jacques, 2016) and estimates the model's two parameters, π_{kl} and μ_{kl}, using a SEM-Gibbs algorithm. In this model, π_{kl} denotes the precision parameter and μ_{kl} denotes the position parameter.

Recently, Nguyen and Kuo (2019) proposed a two step algorithm, the Partition and Merge Based Fuzzy Genetic Clustering Algorithm (PM-FGCA) with the intention of providing more accurate clusters in chromosomes. The algorithm repeats two steps until the desired number of clusters is reached. Step one consists of partitioning the dataset into the maximum number of clusters possible and step two consists of merging the two clusters whose inter-cluster distance is smallest. In this paper, the authors find the PM-FGCA outperforms other categorical methods including the K-Modes, the fuzzy K-Modes, and the genetic fuzzy K-Modes algorithms.

There are many other algorithms for the clustering of categorical data that have been proposed beyond what has been covered in this section. Many of these algorithms focus on computational efficiency and show algorithmic ways to handle noisy data. In Chapter 3 we propose a method to handle noisy data with a connection to statistics. In that chapter we present a method of clustering tertiary data objects that may be used in any existing algorithm that can take a proximity or dissimilarity matrix as an input. The focus in this method is to increase the cluster partition accuracy in the presence of noisy data with the hope being that it can be implemented in methods that are already computationally efficient to improve the accuracy of the clustering solution.

Chapter 3

Clustering Smoothed Dissimilarities in Tertiary Data

Suppose there exists a set of grade-school students upon which we measure performance in subjects like Mathematics, Reading, English, and History as being “below”, “meeting,” or “exceeding” the expectations set by a local school district. Suppose further that the interest is to determine whether there exist subgroups within the students such that students in the same group are more similar to each other than they are to students in other groups. How can such groupings be formed, if possible, when the only measurements recorded for each student is “below”, “meeting”, or “exceeding” the pre-determined standards in each subject? In this paper we introduce a method of creating such groups in such settings using cluster analysis and show this method results in the formation of more accurate cluster partitions than a standard approach under certain conditions.

Cluster analysis is a method of separating a set of images, patterns, or data objects into homogeneous groups such that objects placed in the same group share some property that makes them more similar to each other than they are to any other objects within different groups. Consequently, pertinent to many clustering algorithms is a mathematically defined measure of similarity or dissimilarity that may vary depending upon the type of variable measurements recorded on each object.

When variable measurements are completely quantitative, dissimilarity is usually based on a measure of distance. For example, we may consider the Euclidean or

Manhattan distance between observations. In this setting, a smaller distance between observations implies the observations are more similar while a larger distance implies observations are less similar. Alternatives to the Euclidean and Manhattan distance used in these settings can be found in Everitt et al. (2011), or Friedman, Hastie, and Tibshirani (2017).

On the other hand, when variable measurements are completely qualitative, the notion of distance is not as natural, and thus dissimilarity may be defined by looking at the proportion of attribute measurements upon which objects disagree. However, other methods of defining similarity and dissimilarity for qualitative data can be found in Everitt et al. (2011) or Boriah, Chandola, and Kumar (2008).

It is also possible for variable types to be mixed, including both qualitative and quantitative variables in the same dataset. In this setting, dissimilarity may be defined in terms of distance for the quantitative variables and the proportion of mismatches for the qualitative variables. Clustering algorithms like K-Prototypes (Huang, 1997b) employ this methodology.

Let us consider the case where all variables are qualitative. The simplest case of this is when each attribute measurement is binary, taking one of two outcomes. The clustering of binary data has been studied and used extensively in several fields over the last few years (see e.g., Cornell et al. (2009); Dolnicar and Leisch (2004); and Hitchcock and Chen (2008)).

In this paper, however, we wish to focus on tertiary data. We propose a dissimilarity-based method of creating clusters when the attribute measurements are tertiary: qualitative with 3 classes. For example, a variable may measure level of autism with the responses being "requiring support", "requiring substantial support", or "requiring very substantial support." Though algorithms exist within the field of cluster analysis for use on such data, many of the algorithms that have been proposed for use with qualitative attribute measurements tend to neglect two key issues that may affect the accuracy of clustering solutions produced: variability and noise.

Consider the aforementioned example of clustering a set of grade-school children based on their performance in various subjects. Based on the observed category, one has no idea how close a student who was below the pre-determined standard in Math was to meeting the standard. Similarly, one has no idea how close a student who met the pre-defined standard was to exceeding those standards in that subject. Furthermore, when working with qualitative data it is possible for some information to be obscured within the categories such as variability within the latent variables underlying the structure of the data. This can be an issue as cases of high variability may complicate the clustering task. Another potential issue is that of random noise. As is often the case with data in general, there is always possible measurement error, data input issues, or perhaps issues with subjectivity that complicate the clustering tasks. Therefore, it is important to use clustering methods that compensate for the imperfections that may plague qualitative data. To this end, in this paper we propose the use of statistical smoothing.

Statistical smoothing is a technique used commonly to help find a signal or uncover the true structure of the data that may be buried by noise. In this paper we use smoothing via shrinkage which allows us to move the observed data towards a particular model should such a model be supported by the data. This may help reduce misspecification errors that may occur when assuming a particular model (see e.g., Agresti (2012) or Simonoff (2012)) and allows us to supplement the information in a pair of observations with that of the entire data set (Hitchcock and Chen, 2008). We propose a dissimilarity-based method for the clustering of tertiary observations that uses a shrinkage-based smoother with the purpose of combating high variability and underlying noise that may exist in the data. The ideas presented are an extension of the work of Hitchcock and Chen (2008) where it was shown pre-smoothing of dissimilarities helped improve partitioning accuracy in the case of binary data that had a noisy underlying structure.

The outline of the paper is as follows: We provide some background information relevant to the method introduced in Section 2, formally define pairwise dissimilarities for a set of tertiary data objects, and introduce a clustering algorithm based on a smoothed version of the dissimilarity matrix in Section 3. In Section 4, we describe a simulation study undertaken to illustrate the effect of the proposed method of smoothing dissimilarities on the accuracy of cluster partitions. Then, in Section 5, we apply the proposed algorithm to the Pima Indian Diabetes dataset. We conclude the paper in Section 6 with a brief discussion of the methodology and its possible ramifications.

3.1 Background

3.1.1 Estimation of Multiple Cell Probabilities

Table 3.1 Cell Probabilities for a Pair of Tertiary Objects $\mathbf{Y}_k$ and $\mathbf{Y}_{k'}$

	$\mathbf{Y}_{k'}$		
$\mathbf{Y}_k$	1	2	3
1	π_{11}	π_{12}	π_{13}
2	π_{21}	π_{22}	π_{23}
3	π_{31}	π_{32}	π_{33}

When cross-classifying tertiary data objects, data on each pair of observations can be summarized within a 3×3 contingency table as shown in Table 3.1 where the entries within the table, $\{\pi_{kk'}\}, k = 1, 2, 3; k' = 1, 2, 3$ denote the true cell probabilities. For example, the probability that object $\mathbf{Y}_k$ has a value of 1 for a particular variable and object $\mathbf{Y}_{k'}$ has value of 2 for that variable is denoted by π_{12}. Since these true cell probabilities are never known, we estimate them. To do this, we look at the data as arising from a multivariate distribution.

Borrowing from categorical data analysis, Table 3.1 can be thought of as probabilities of the distribution of multinomial random variables, $\mathbf{X} = (X_1, X_2, \ldots, X_n)$, where $\boldsymbol{\pi} = (\pi_{11}, \pi_{12}, \pi_{13}, \pi_{21}, \pi_{22}, \pi_{23}, \pi_{31}, \pi_{32}, \pi_{33})$ represents the true cell probabilities, with the conditions $0 \leq \pi_{ij} \leq 1, i = 1, 2, 3, j = 1, 2, 3$, and $\sum_i \sum_j \pi_{ij} = 1$. Let $\hat{\pi}_{ij}, i = 1, 2, 3, j = 1, 2, 3$, denote an estimate for the true cell probability π_{ij}. In this setting, $\hat{\boldsymbol{\pi}} = n^{-1}\mathbf{X}$ is the unique minimum variance unbiased estimator (UMVUE) of $\boldsymbol{\pi}$ (see, e.g., Fienberg and Holland (1973)). However, when the goal is to simultaneously estimate multiple cell probabilities, (in the 3×3 case, we estimate 9, of which 8 are free parameters), Fienberg and Holland (1973) presented a shrinkage estimator that we will denote as $\boldsymbol{\pi}^*$ and showed the ordinary multivariate sample mean to be inadmissible in terms of mean squared error loss. In this paper we will use this estimator to smooth our dissimilarity matrix, $\mathbf{D}$. In the next subsection, we discuss smoothing.

3.1.2 Smoothing

Smoothing is a statistical technique that is used to aid in detecting the underlying signal or latent structure that may be hidden by noisy data. The general form of a smoothed estimator in the multinomial setting can be obtained as shown in equation (3.1) where $\hat{\pi}_{ij}$ denote the observed cell proportions and $\widetilde{\pi}_{ij}$ denote estimated cell probabilities under an assumed model (see, e.g., Hitchcock and Chen (2008); or Albert (1987)).

$$\pi^*_{ij} = (1 - \lambda)\hat{\pi}_{ij} + \lambda(\widetilde{\pi}_{ij}). \tag{3.1}$$

Equation (3.1) is a "data-dependent" form of a smoothed estimator, but the smoothed estimator does not have to be in this form. Simonoff (1995) discussed other methods of smoothing categorical data that could be used as alternatives to this approach. In equation (3.1), λ denotes the degree of smoothing. For small λ, more emphasis would be placed on the observed cell probabilities $\hat{\pi}_{ij}$ and for larger values, more emphasis on the estimated cell probabilities under the assumed model, $\tilde{\pi}_{ij}$. Though the estimate π^*_{ij} is a biased estimator of π, we use it here since it proves to be more robust than $\hat{\pi}$ under certain settings like that of sparse multinomial tables, and because in the multinomial setting, it allows us to use information from neighboring cells to garner better estimates for cell probabilities (see Simonoff (1995) or Simonoff (1998)). This method of smoothing is closely related to Stein estimation (see Efron and Morris (1977)).

3.2 Method

In this section we discuss in detail our proposed method of pre-smoothing tertiary dissimilarities as a precursory step to clustering and discuss the clustering algorithms that will be used in this paper.

3.2.1 Dissimilarities for a tertiary data set

Table 3.2 Summary of matches and mismatches for a pair of objects $\mathbf{Y}_k$ and $\mathbf{Y}_{k'}$.

	$\mathbf{Y}_{k'}$			
$\mathbf{Y}_k$	1	2	3	Totals
1	a	b	c	$a+b+c$
2	d	e	f	$d+e+f$
3	g	h	i	$g+h+i$
Totals	$a+d+g$	$b+e+h$	$c+f+i$	P

Dissimilarities for tertiary data observations may be calculated as the proportion of mismatches among P tertiary attribute measurements for each pair of observations. Consider Table 3.2 which depicts the cross-tabulation of two multivariate objects, $\mathbf{Y}_k$

and $\mathbf{Y}_{k'}$, from whence we demonstrate the notion of "pairwise dissimilarity." In this table, each attribute has an outcome of 1, 2, or 3 denoting its membership. Letters a, e, and i represent the total number of variables for which both objects have an outcome classified as 1, 2, and 3, respectively. The other letters denote the number of attributes upon which objects $\mathbf{Y}_k$ and $\mathbf{Y}_{k'}$ have different membership. For example, b denotes the number of variables for which object $\mathbf{Y}_k$ has a value of 1 and object $\mathbf{Y}_{k'}$ has a value of 2; similarly, c denotes the number of variables for which object $\mathbf{Y}_k$ has a value of 1 and $\mathbf{Y}_{k'}$ has a value of 3. Based on this 3×3 table, we define similarity for the kth and k'th object as

$$S_{kk'} = \frac{1}{P}(a + e + i)$$

and dissimilarity as

$$D_{kk'} = 1 - S_{kk'}.$$

This particular method of defining dissimilarity may be referred to as the matching coefficient method when used in the binary case or the overlap method in the terminology of computer science with matches and mismatches weighted equally. There are a multitude of other methods besides the matching coefficient that can be used with categorical attributes (see, e.g., Everitt et al. (2011) or Boriah, Chandola, and Kumar (2008)); however, regardless of the method used to define the pairwise dissimilarities, the smoothing method presented in Sections 3.2.2 and 3.2.3 should still be applicable.

In some traditional methods, once the pairwise dissimilarities for each pair of observations have been calculated, they are then used as the elements of a $n \times n$ dissimilarity matrix $\mathbf{D}$ that is then used as the input to a clustering algorithm of choice.

We propose, instead, pre-smoothing the dissimilarity matrix before implementing the clustering algorithm. In the next subsection, we discuss possible choices for models to be used in the smoothing step along with rationale underlying each model.

3.2.2 Choice for Model-Based Estimators

In this section we discuss possible choices for the model used to estimate $\boldsymbol{\pi}$. For notation, we use i and j to denote the ith row and jth column of the 3×3 table formed for a particular pair of objects k and k'. However, this procedure would be repeated for each pair of objects.

Previously (in Section 2) we discussed how data in a 3×3 contingency table like Table 3.2 could be thought of as data arising from a multinomial distribution. In this context, we view the problem as that of estimating the cell probabilities of the multinomial distribution. Recall, for each pair of tertiary objects, there are two cell probabilities of interest—the true cell probabilities and the estimated cell probabilities under an assumed model. Let

$$\boldsymbol{\pi} = (\pi_{11}, \pi_{12}, \pi_{13}, \pi_{21}, \pi_{22}, \pi_{23}, \pi_{31}, \pi_{32}, \pi_{33})$$

denote the set of true probabilities for each cell in the 3×3 table shown in Table 3.1, and let

$$\widetilde{\boldsymbol{\pi}} = (\widetilde{\pi}_{11}, \widetilde{\pi}_{12}, \widetilde{\pi}_{13}, \widetilde{\pi}_{21}, \widetilde{\pi}_{22}, \widetilde{\pi}_{23}, \widetilde{\pi}_{31}, \widetilde{\pi}_{32}, \widetilde{\pi}_{33})$$

denote an estimate of the probabilities of observations falling in each cell of the 3×3 table under the assumed model.

If the researcher has no prior information about the relationship between a pair of observations, then a non-informative model may be used. In this case, one possibility is an equal-probability model. Under this model, $\widetilde{\pi}_{ij} = \frac{1}{9}, i = 1, 2, 3, j = 1, 2, 3$. This suggests observations are just as likely to fall into any cell in the 3×3 contingency table. If, instead, the researcher has the belief that a pair of observations are independent in their tertiary variable measurements, then an independence model may be appropriate. In this setting, the rows and columns of Table 1 are independent; therefore, $\widetilde{\pi}_{ij} = \hat{\pi}_{i+}\hat{\pi}_{+j}, i = 1, 2, 3, j = 1, 2, 3$. Lastly, if the researcher feels that a pair of observations are more likely to match in a particular category and less likely to match on another, then a model of dependence could be chosen to reflect those prior beliefs. For example,

$$\widetilde{\pi}_{ij} = \begin{cases} \frac{1}{10}, & \text{for } i = j \\ \frac{1}{60} & \text{for } i \neq j \end{cases}$$

would mean the pair of observations are more likely to have the same tertiary attribute membership for a particular attribute than to have any other combination of membership classes. This method of choosing a model can be thought of similarly to the Bayesian framework of choosing a prior distribution that reflects the researchers' prior belief about the structure of the data. Just as it is important to choose an appropriate prior in Bayesian inference, it is also important to choose a smoothing model carefully, being mindful that an appropriate model should be supported by the data.

Once the appropriate smoothing model is chosen, then shrinkage-based smoothing is implemented.

3.2.3 Shrinkage-type Smoother for the 3×3 table

In this section we specifically discuss our proposed method of smoothing the dissimilarity matrix using the Fienberg-Holland estimator (Fienberg and Holland, 1973). We begin with a discussion of the estimator and then show how each of the cells in the 3×3 table of pairwise dissimilarities can be smoothed.

Fienberg-Holland Estimator

The Fienberg-Holland estimator (Fienberg and Holland, 1973) has been shown to be a better estimator of $\boldsymbol{\pi}$ than $\hat{\boldsymbol{\pi}}$ in terms of minimizing the total mean squared error loss, and it can be used to reflect prior information about the data structure. This is significant because when we perform clustering using the original observed dissimilarities, this is akin to using $\hat{\boldsymbol{\pi}}$ to estimate the cell probabilities without using knowledge about the latent structure of the data. Consider the motivation behind the method.

Consider placing a Dirichlet prior with mean vector $\boldsymbol{\gamma}$ on $\boldsymbol{\pi}$. The posterior mean, then, is given by

$$(1-\lambda)\hat{\pi}_{ij} + \lambda(\gamma_{ij}) \tag{3.2}$$

with

$$\lambda = \frac{\kappa}{P+\kappa},$$

where P denotes the number of attributes recorded on each observation. Fienberg and Holland (1973) denote the value that minimizes the expected squared error loss between $\boldsymbol{\pi}$ and the estimate given in equation (3.2) by κ. Thus,

$$\kappa = \frac{1-\sum \pi_{ij}^2}{\sum(\gamma_{ij}-\pi_{ij})^2}.$$

Then a pseudo-Bayes estimator can be written as shown in equation (3.3)

$$\frac{P}{P+\kappa}(\hat{\pi}_{ij}) + \frac{\kappa}{P+\kappa}(\gamma_{ij}). \tag{3.3}$$

Since $\boldsymbol{\gamma}$ represents prior estimates of the mean vector, model-based estimates for $\boldsymbol{\pi}$ can be used as is traditionally done in an empirical Bayesian approach (see Fienberg and Holland (1973)). Lastly, $\boldsymbol{\pi}$ and κ must also be estimated as their true values are not known. Therefore, their maximum likelihood estimators are used, and equation (3.3) can be rewritten with

$$\hat{\kappa} = \frac{1-\sum \hat{\pi}_{ij}^2}{\sum(\tilde{\pi}_{ij}-\hat{\pi}_{ij})^2}.$$

In the case of tertiary data, the Fienberg-Holland estimate of κ can be written specifically as

$$\hat{\kappa} = \frac{1-(\hat{\pi}_{11}+\hat{\pi}_{12}+\cdots+\hat{\pi}_{33})}{(\tilde{\pi}_{11}-\hat{\pi}_{11})^2+(\tilde{\pi}_{12}-\hat{\pi}_{12})^2+\cdots+(\tilde{\pi}_{33}-\hat{\pi}_{33})^2}.$$

Therefore, the Fienberg-Holland estimate for π_{ij} is given by

$$\pi_{ij}^* = \frac{P}{P+\hat{\kappa}}(\hat{\pi}_{ij}) + \frac{\hat{\kappa}}{P+\hat{\kappa}}(\tilde{\pi}_{ij}). \tag{3.4}$$

Equation (3.4) can be used to smooth each of the cells of the table for a particular pair of observations.

Smoothing the 3×3 Table

To smooth the 3×3 table, we use the James-Stein-type estimator shown in equation (3.4) to obtain the final smoothed pairwise dissimilarities. To do this we multiply the expression given in equation (3.4) by P as shown in equation (3.5).

$$\{i,j\}^{(smooth)} = \tilde{\pi}_{ij}^* P = [\frac{P}{P+\hat{\kappa}}(\hat{\pi}_{ij}) + \frac{\hat{\kappa}}{P+\hat{\kappa}}(\tilde{\pi}_{ij})]P \tag{3.5}$$

The multiplication by P allows us to change from the expected proportion of attributes to fall in each cell (same as the cell estimate as given in equation (3.4)) to the expected number of attribute outcomes falling in each cell. We use these to obtain our inputs for the smoothed dissimilarity matrix.

For cell $\{i,j\}, i = 1,2,3, j = 1,2,3$, we use equation (3.5) to obtain smoothed cell counts that correspond to each cell location as shown in Table 3.1. Then

$$S_{kk'}^{smooth} = \frac{1}{P}(a^{smooth} + e^{smooth} + i^{smooth})$$

$$D_{kk'}^{smooth} = 1 - S_{kk'}^{smooth}$$

Once these smoothed dissimilarities are formed for each pair of observations, they are input into a $n \times n$ smoothed dissimilarity matrix, $\mathbf{D}_{smooth}$, that is used as the input for the clustering algorithm of choice.

3.2.4 Clustering Algorithms Used

In this paper, we perform cluster analysis using two different algorithms: Average Linkage and K-Medoids. The different algorithms are used here to glean in which settings smoothing via shrinkage may be useful.

The Average Linkage algorithm (Sokal and Michener, 1958) is an agglomerative hierarchical method of clustering. This means it starts with each individual observation being a singleton cluster and on each iteration merges pairs of observations or clusters together based on a similarity measure until there is only one cluster containing all observations at the highest level of the hierarchy. Since the output of hierarchical clustering can be shown with a dendrogram that can be cut to obtain the desired number of clusters, using such an algorithm eliminates the need to know the number of clusters a priori (see e.g., Friedman and Rubin (1967) or Albalate and Minker (2011)). The Average Linkage algorithm, in particular, is used here as it merges observations based on the average pairwise distance between cluster members. Using this helps to avoid issues that can occur when using single linkage or complete linkage algorithms, which look at the shortest and longest distance, respectively, between the members of different clusters (see e.g., Albalate and Minker (2011)).

The K-Medoids algorithm (Kaufman and Rousseeuw, 1987) is a partitioning-based method of clustering, which partitions observations into groups based on distance to a central value (medoid) with the assumption that an observation can only belong to only one cluster. The medoid is defined as the ”most representative” cluster member (see Kaufman and Rousseeuw (1987) for details). This algorithm is used because it can take a dissimilarity matrix as an input, does not require Euclidean distance to be used as a dissimilarity measure, and has been shown to be a more robust method of clustering than K-Means clustering (see e.g., Friedman and Rubin (1967) or Albalate and Minker (2011)).

Each algorithm is implemented using R (R Core Team, 2019). The Average Linkage algorithm is implemented using the `hclust` function in the `stats` package, while the K-Medoids algorithm is implemented using the `pam` (Rousseeuw and Kaufman, 1987) function in the `cluster` package.

3.3 Simulations

In this section we discuss a simulation study undertaken to assess the performance of the proposed method of pre-smoothing tertiary dissimilarities using the Fienberg-Holland estimator. We examine its performance when using the Average Linkage and K-Medoids algorithms and assess the method's effect on the accuracy of clustering partitions produced. To better assess the method, we perform simulations in two settings: one in which we assume the data has arisen from a latent mixture of multinomial distributions and in the other we assume it has arisen from a mixture of continuous distributions. For each we measure this "accuracy" of the clustering solution in terms of the Adjusted Rand Index (ARI) as proposed by Hubert and Arabie (1985). We begin with a brief introduction to this measure.

Table 3.3 Cross-Tabulation of Two Partitions

	Partition One	
Partition Two	Same Group	Different Group
Same Group	A	B
Different Group	C	D

3.3.1 Adjusted Rand Index

Consider Table 3.3 (see e.g., McNicholas (2017) for this type of table) which shows the cross-tabulation of the results of two different clustering methods. In this table, the columns refer to the partition created from one method and the rows denote the partition created by another method. Here, A denotes the number of pairs of objects that both partitioning methods put in the same groups. B denotes the number of pairs of objects put in the same group by the first method but placed in different groups by the second method. C denotes the reverse, the number of pairs of objects put in the same group by the second method but in different groups by the first method, and D denotes the number of pairs of objects that both partitioning methods put in different groups. These values, then, can be used to assess cluster partition accuracy when one of the partitions is assumed to represent the ground-truth partition.

Using the notation of McNicholas (2017), the ARI can be computed as:

$$ARI = \frac{N(A+D) - [(A+B)(A+C) + (C+D)(B+D)]}{N^2 - [(A+B)(A+C) + (C+D)(B+D)]}$$

where N denotes the total number of possible pairings.

As the ARI is a correction to the Rand Index (Rand, 1971) it can take values as large as 1, with higher values denoting more agreement between two clustering solutions and values closer to 0 denoting chance agreement. For our study, one of the partitions will denote the true clustering structure and the other, the proposed clustering partition produced by either the Average Linkage or K-Medoids algorithm. Thus, we will compare these two partitions using the values found in Table 3.3 to assess accuracy. A method which results in higher ARI values is then considered to be more reflective of the true latent structure of the data and hence a better method of clustering the observations in the context of our simulation study.

3.3.2 Multinomial Simulation Setup

In this section we discuss the method used to perform our multinomial simulations beginning with a discussion of the data generation and concluding with a discussion of the clustering scenarios and separation settings used to assess cluster goodness.

Data Generation

In this simulation, we assume there are n data objects, each with P tertiary features recorded on them, that have arisen from C clusters. We further assume that each of these P measurements is independent of the other $P-1$ measurements (mutually independent). In this case, the tertiary observation, Y_{lkp}, refers to the specific categorical outcome of the kth object in cluster l on feature p where $l = 1, 2, \ldots, C, k = 1, 2, \ldots, n_l, \sum_{l=1}^{C} n_l = n, p = 1, 2, \ldots, P$. Thus $Y_{lkp} \in \{1, 2, 3\}$. Lastly, the probability of an attribute measurement being in either category remains constant for each of the P measurements. Therefore, in this setting, $\mathbf{Y}_{lk} \sim Multi(P, \boldsymbol{\tau}_l)$ with $\boldsymbol{\tau}_l = (a_l, b_l, c_l), 0 \leq a_l \leq 1, 0 \leq b_l \leq 1, 0 \leq c_l \leq 1$. $\boldsymbol{\tau}_l$ denotes the parameter vector for the lth cluster, and $a_l, b_l,$ and c_l refer to the probability of obtaining an attribute measurement that falls in category 1, 2, or 3, respectively, for data objects in the lth cluster.

Parameter Settings

For the simulations, we generate 5000 data sets each with a total of $n = 600$ objects each with $P = 10$ tertiary features. We assume these objects have arisen from $C = 3$ clusters. In the first case, we assume an equal number of observations from each cluster ($n_1 = n_2 = n_3 = 200$). In the second case, we assume there are a varying number of observations from each cluster ($n_1 = 100, n_2 = 200, n_3 = 300$).

Table 3.4 Parameter Settings for Multinomial Simulations

	Separation Setting				
Parameter Vector	I	II	III	IV	V
$\boldsymbol{\tau}_1$	(0.40,0.30,0.30)	(0.50,0.25,0.25)	(0.60,0.20,0.20)	(0.70,0.15,0.15)	(0.80,0.10,0.10)
$\boldsymbol{\tau}_2$	(0.30,0.40,0.30)	(0.25,0.50,0.25)	(0.20,0.60,0.20)	(0.15,0.70,0.15)	(0.10,0.80,0.10)
$\boldsymbol{\tau}_3$	(0.30,0.30,0.40)	(0.25,0.25,0.50)	(0.20,0.20,0.60)	(0.15,0.15,0.70)	(0.10,0.10,0.80)

For each scenario, we generate the kth tertiary object in cluster $1, 2,$ and 3, respectively, as follows: $\mathbf{Y}_{1k} \sim Multi(10, \boldsymbol{\tau}_1)$, $\mathbf{Y}_{2k} \sim Multi(10, \boldsymbol{\tau}_2)$, and $\mathbf{Y}_{3k} \sim Multi(10, \boldsymbol{\tau}_3)$. The value of $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2,$ and $\boldsymbol{\tau}_3$ are specified as shown in Table 3.4.

The cluster separation settings denoted I, II, III, IV, and V, represent the distance between cluster centers. As we increase the settings, the distance between cluster centers increases. Therefore, Setting I denotes the smallest distance between centers and setting V denotes the largest. Table 3.4 shows the parameter vector $\boldsymbol{\tau}_l$ used for clusters $l = 1, 2, 3$ for each separation setting in our simulations. To better motivate the purpose of these settings, consider separation setting V. Under this last setting, observing an object with several attributes with measures of "1" would suggest it is more likely that the object arose from sub-population 1 as opposed to either of the other two sub-populations. On the other hand, if this same outcome was observed in cluster separation setting I, it would be tougher to determine from which cluster it had arisen. Thus, the overlap between clusters decreases, in general, as the separation settings increase. This indicates that the clustering problem gets easier as we increase from setting I to V.

After generating the tertiary data objects, each object is stored in a $n \times P$ data matrix as shown below where object m's measurements are stored in row m.

$$\boldsymbol{Y} = \begin{bmatrix} Y_{11} & Y_{12} & \dots & Y_{1P} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{m1} & Y_{m2} & \dots & Y_{mP} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{n1} & Y_{n2} & \dots & Y_{nP} \end{bmatrix}.$$

Once the tertiary objects are created, we perform clustering using the Average Linkage and K-Medoids algorithms with the observed dissimilarities and the dissimilarities smoothed under three smoothing models: independence, equal probability, and high probability of match. For the independence model, we set our smoothed cell estimates to $\widetilde{\pi}_{ij} = \hat{\pi}_{i+}\hat{\pi}_{+j}, i = 1,2,3, j = 1,2,3$. For the equal-probability model, we set $\widetilde{\pi}_{ij} = \frac{1}{9}, i = 1,2,3, j = 1,2,3$. For the high probability of match model, we set $\widetilde{\pi}_{ij} = \begin{cases} \frac{1}{10}, & \text{for } i = j \\ \frac{1}{60} & \text{for } i \neq j \end{cases}$. We then compare the outcomes of each method using the average ARI and show the results in the next section.

Results

In this simulation, the Average Linkage algorithm produced the highest average ARI values overall. This suggests that a hierarchical method of clustering may be a better method to use in this setting rather than a partitioning-based method like K-Medoids. Furthermore, in most cases smoothing under the assumption of independence produced more accurate cluster partitions, as measured by the average ARI, compared to not smoothing, and always performed better than any other smoothing method considered. This suggests clustering via smoothing with an independence assumption may be the better method to use when working with tertiary data arising from a multinomial setting.

Table 3.5 Average ARI Values for the Average Linkage Clustering of Multinomial Simulated Tertiary Data Assuming 200 Observations in Each Cluster.

Average Linkage Algorithm				
I	II	III	IV	V
0.678 o	0.870 o	0.983 o	0.993 o	0.998 o
0.688 s/i	0.873 s/i	0.987 s/i	0.993 s/i	0.998 s/i
0.002 s/E	0.072 s/E	0.516 s/E	0.857 s/E	0.978 s/E
0.008 s/H	0.187 s/H	0.560 s/H	0.856 s/H	0.977 s/H

In Table 3.5, the resulting cluster accuracy obtained using the Average Linkage algorithm is shown when we assume there are an equal number of observations from each subpopulation. Note: Each value is the average (across 5000 data sets) ARI for the clustering produced from an Average Linkage algorithm based on (top within each cell) the observed dissimilarities (o); (second within cell) the smoothed dissimilarities based on the independence model (s/i); (third within cell) the smoothed dissimilarities based on the equal-probability model (s/E); (last within cell) the smoothed dissimilarities based on the high probability of match model (s/H). Within this table, the average ARI value tends to increase from cluster separation setting I to cluster separation setting V. This is to be expected as the distance between the cluster centers is increasing. Furthermore, we see that the highest average ARI values are obtained in all cases by first pre-smoothing the dissimilarities towards a model of independence except in cluster separation settings IV and V. In these two settings, the clustering solutions produced from both pre-smoothing the dissimilarities towards a model of independence and for using the observed (non-smoothed) dissimilarities result in the same cluster partition accuracy of 0.993 and 0.998, respectively. It is also worth noting that the clustering solutions obtained after pre-smoothing the dissimilarities towards a model of equal probability or high probability of match result in a lower average ARI, suggesting the solutions obtained in these cases are not as reflective of the true latent structure of the data as are the other two methods (observed and pre-smoothed toward independence). In the most challenging cases (separation settings I and II) the average ARI values are near 0. An average ARI value close to 0 is comparable to forming clusters by chance. These results can be seen graphically in the left plot of Figure 3.1.

Table 3.6 provides the clustering accuracy obtained when using the K-Medoids algorithm and assuming an equal number of observations from each sub-population. In this case, the majority of the average ARI values are lower in most separation settings than in Table 3.5. However, we see smoothing towards a model of equal probability with the K-Medoids algorithm results in higher clustering accuracy in each cluster separation setting than in Table 3.5. The same is true for pre-smoothing the dissimilarities towards a model of high probability of match in cluster separation settings

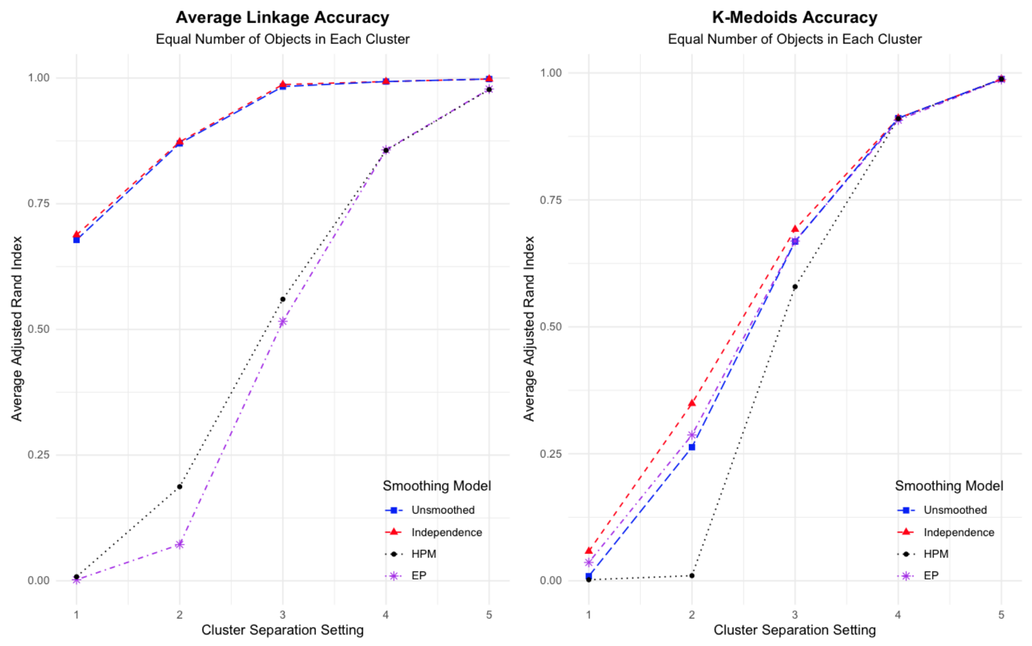


Figure 3.1 Average ARI value assuming 200 observations from each cluster. The left plot corresponds to the Average Linkage clustering results and the right corresponds to the K-medoids clustering results.

III through V. Despite the aforementioned differences, the same general trends observed in Table 3.5 are replicated in Table 3.6. In particular, the average ARI value still increases from cluster separation setting I to cluster separation setting V (this means the lowest accuracy is observed when there is less separation between cluster centers), and we still observe the highest average ARI value when the dissimilarities are first smoothed towards a model of independence in each of the first three cluster separation settings. However, in separation settings IV and V (as before), the average ARI values are the same for the clustering solutions produced using the observed dissimilarities. The average ARI values obtained from the K-Medoids algorithm in this setting are shown graphically in the right plot of Figure 3.1.

Table 3.7 shows the accuracy of the clustering solutions produced using the Average Linkage algorithm when there are an unequal number of observations from each sub-population. In this case, the average ARI values for the clustering solu-

Table 3.6 Average ARI Values for the K-Medoids Clustering of Multinomial Simulated Tertiary Data Assuming 200 Observations in Each Cluster.

K-Medoids Algorithm				
I	II	III	IV	V
0.009 o 0.058 s/i 0.036 s/E 0.002 s/H	0.263 o 0.349 s/i 0.287 s/E 0.010 s/H	0.668 o 0.692 s/i 0.669 s/E 0.579 s/H	0.911 o 0.911 s/i 0.907 s/E 0.910 s/H	0.988 o 0.988 s/i 0.987 s/E 0.988 s/H

Table 3.7 Average ARI Values for the Average Linkage Clustering of Multinomial Simulated Tertiary Data Assuming an Unequal Number of Observations in Each Cluster.

Average Linkage Algorithm				
I	II	III	IV	V
0.541 o 0.551 s/i 0.011 s/E 0.014 s/H	0.579 o 0.877 s/i 0.084 s/E 0.235 s/H	0.985 o 0.987 s/i 0.551 s/E 0.601 s/H	0.990 o 0.990 s/i 0.868 s/E 0.872 s/H	0.997 o 0.997 s/i 0.980 s/E 0.979 s/H

tions produced using the unsmoothed dissimilarities and for those resulting from pre-smoothing the dissimilarities towards an independence model are all lower than the corresponding values found in Table 3.5. The opposite, however, is true when the dissimilarities are pre-smoothed towards a model of equal probability or high probability of match. Under these two assumptions the resulting clustering accuracy is higher than their corresponding values in Table 3.5. However, the latter two pre-smoothing methods (equal probability and high probability of match), still do not produce clustering solutions with as high accuracy as those produced using either the observed dissimilarities or those pre-smoothed towards an independence model. This can be seen visually in the left plot of Figure 3.2.

In Table 3.8, the resulting clustering accuracy is shown when observations are clustered using the K-Medoids algorithm assuming there are an unequal number of

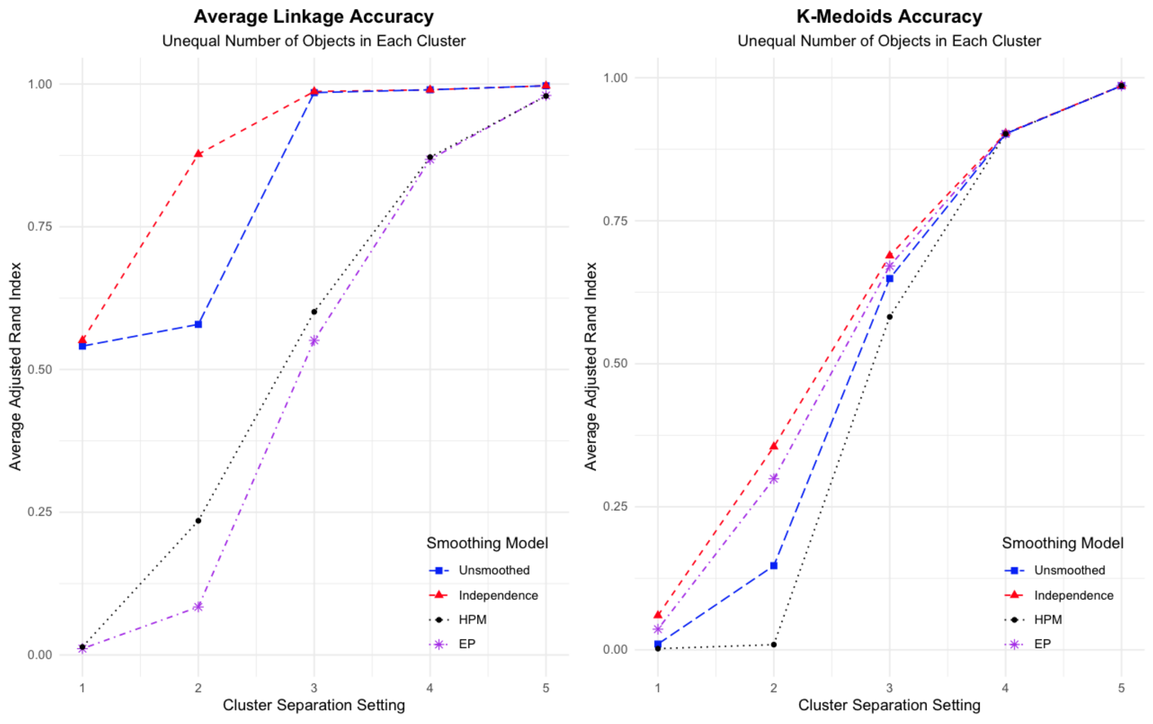


Figure 3.2 Average ARI value assuming 100 observations from cluster one, 200 from cluster two, and 300 from cluster three. The left plot corresponds to the Average Linkage clustering results and the right corresponds to the K-medoids clustering results.

Table 3.8 Average ARI Values for the K-Medoids Clustering of Multinomial Simulated Tertiary Data Assuming an Unequal Number of Observations in Each Cluster.

K-Medoids Algorithm				
I	II	III	IV	V
0.010 o 0.060 s/i 0.036 s/E 0.002 s/H	0.147 o 0.355 s/i 0.299 s/E 0.009 s/H	0.649 o 0.689 s/i 0.671 s/E 0.582 s/H	0.902 o 0.903 s/i 0.902 s/E 0.902 s/H	0.986 o 0.986 s/i 0.986 s/E 0.986 s/H

observations from each subpopulation. In this table, many of the accuracy measures differ in various ways from their corresponding values in Table 3.6. For example, under cluster separation settings III through V, the clustering solutions resulting from the use of the observed dissimilarities obtain lower accuracy values with average

ARI values of 0.649, 0.902, 0.986 from Table 3.8, for cluster separation settings III, IV, and V, respectively, and 0.668, 0.911, and 0.988 in Table 3.6. Similar results are shown for when the dissimilarities are pre-smoothed towards a high probability of match model. On the other hand, pre-smoothing the dissimilarities towards a model of independence, in this case, results in higher accuracy measures than observed in Table 3.6 with the exception of cluster separation setting IV, in which the average ARI is marginally lower than before. Lastly, for dissimilarities pre-smoothed towards a model of equal probability, the results are more diverse. In some cases (separation settings II, III, and V) the accuracy increases slightly from Table 3.6, but in others it decreases from the aforementioned table. On the other hand, for the clustering solutions produced from pre-smoothing the dissimilarities towards a high probability of match model results stay fairly the same in all cluster separation settings. Each method's performance can be seen graphically in the right plot of Figure 3.2.

Overall the simulation results suggest that if smoothing will be performed, it is better to smooth the dissimilarities towards a model of independence when the data is believed to have arisen from a multinomial setting. In each setting explored, the accuracy obtained regardless of algorithm used from using the pre-smoothed dissimilarties in the aforementioned manner were as high or higher than that resulting from the observed dissimilarities. This finding agrees with Hitchcock and Chen (2008) where it was found pre-smoothing may not be necessary in cases of larger separation between cluster centers, but may result in better performance when this is not the case. The results also suggests that the Average Linkage algorithm may be the better algorithm to use under such settings in general as the accuracy obtained from this algorithm is typically higher than what is seen with the K-Medoids accuracy. Thus smoothing (while influential for the Average Linkage algorithm) may be even more beneficial if a practitioner will be using the K-Medoids algorithm.

3.3.3 Normal Simulation Setup

Within this simulation study we assess the proposed method's performance while assuming the tertiary objects have arisen from an underlying continuous process. We examine this performance after placing various structural settings on the data. In this section, we explain these settings specifically along with their rationality. We begin our discussion with data generation.

Data Generation

For each data set we assume there are n data objects with P tertiary observations recorded on each of them. We further assume these objects have arisen from C subpopulations (clusters) and that the tertiary values have resulted from the discretization of P latent Gaussian distributed features. To simulate this, we generate each feature of the kth object in cluster l such that $\boldsymbol{Y}^*_{lk} \sim N_P(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}), k = 1, 2, ...n_l, l = 1, 2, ...C, C \leq n$, where $\boldsymbol{\mu}_l$ may be different for each subpopulation.

For feature generation, we consider two settings. In setting I, we assume the P features are mutually independent and set $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_P$. In setting II, we assume the P latent features are positively (and equally) correlated with $\{\sigma^2_{pp'}\} = \begin{cases} \sigma^2 & \text{for } p = p' \\ \frac{\sigma^2}{4} & \text{for } p \neq p' \end{cases}$. To generate the tertiary observations for object m (we use m here to denote a general object and not an object specific to a particular cluster) we define a cutoff for each of the P latent variables as shown below

$$Y_{mp} = \begin{cases} 0, & \text{if } Y^*_{mp} \leq -\xi_p \\ 1, & \text{if } -\xi_p < Y^*_{mp} \leq \xi_p \\ 2, & \text{if } Y^*_{mp} > \xi_p \end{cases},$$

where ξ_p denotes the specific cutoff for the p-th latent variable. After doing this for each of the n objects, the final tertiary dataset can be represented in a $n \times P$ matrix, $\boldsymbol{Y}$, where object m's features are represented within row m:

$$\boldsymbol{Y} = \begin{bmatrix} Y_{11} & Y_{12} & \dots & Y_{1P} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{m1} & Y_{m2} & \dots & Y_{mP} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{n1} & Y_{n2} & \dots & Y_{nP} \end{bmatrix}$$

For the simulation study presented here we generate 5000 datasets. Each dataset consists of $n = 400$ objects with $P = 10$ tertiary features. We assume these objects have arisen from $C = 3$ subpopulations with $n_1 = 200, n_2 = 100$, and $n_3 = 100$. Their P features are simulated as follows: We generate $\boldsymbol{Y}^*_{1k} \sim N_P(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$ for $k = 1, 2, ..., n_1$, $\boldsymbol{Y}^*_{2k} \sim N_P(\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$ for $k = 1, 2, ..., n_2$, and $\boldsymbol{Y}^*_{3k} \sim N_P(\boldsymbol{\mu}_3, \boldsymbol{\Sigma})$ for $k = 1, 2, ..., n_3$. For our simulations, the mean vectors were randomly generated by generating $\boldsymbol{\mu}_1 \sim N_P(\boldsymbol{\delta}, \boldsymbol{I})$ for each object from cluster one, $\boldsymbol{\mu}_2 \sim N_P(\boldsymbol{0}, \boldsymbol{I})$ for objects from cluster two, and $\boldsymbol{\mu}_3 \sim N_P(-\boldsymbol{\delta}, \boldsymbol{I})$ for each object from cluster three. Next, each of these $P = 10$ variables is converted to a tertiary variable based on the following cutpoints

$$Y_{mp} = \begin{cases} 1, & \text{if } Y^*_{mp} \leq -0.43 \\ 2, & \text{if } -0.43 < Y^*_{mp} \leq 0.43 \\ 3, & \text{if } Y^*_{mp} > 0.43 \end{cases}$$

where the 0.43 refers to the value below which $\frac{2}{3}$ of all observations would fall on a standard normal distribution (Note $\frac{2}{3} = Pr(X \leq 0.43)$ where $X \sim N(0, 1)$). This process is repeated for all n observations with the resulting tertiary values being stored in row m of the aforementioned data matrix, $\mathbf{Y}$.

Parameter Settings

The method proposed in this paper is hypothesized to be most effective in cases in which the data objects have arisen from a noisy underlying structure that may be exhibited by a vast amount of variation within clusters themselves or from a large amount of overlap occurring between clusters. To examine this, we vary the values of δ and σ for our simulations. δ represents the distance between the cluster centers and σ represents the within-cluster variability. We look at the performance of the proposed method for $\delta \in \{0.5, 2, 3.5\}$ and $\sigma \in \{1, 5, 10\}$. Larger values of δ denote more distance between cluster centers which should lead to more separation between clusters (all other factors held constant) and hence an easier clustering problem. On the other hand, larger values of σ represent more intra-cluster dispersion. Therefore, holding other things constant, this would signify a harder cluster problem. Lastly, we assess the performance using the observed dissimilarities (traditional method) versus 3 smoothing models: independence, equal-probability, and high probability of match. For the independence model, we set our smoothed cell estimates to $\widetilde{\pi}_{ij} = \hat{\pi}_{i+}\hat{\pi}_{+j}, i = 1, 2, 3, j = 1, 2, 3$. For the equal-probability model, we set $\widetilde{\pi}_{ij} = \frac{1}{9}, i = 1, 2, 3, j = 1, 2, 3$. Finally, for the high probability of match model, we set $\widetilde{\pi}_{ij} = \begin{cases} \frac{1}{10}, & \text{for } i = j \\ \frac{1}{60} & \text{for } i \neq j \end{cases}$.

Thus in our simulation, we examine the proposed method of pre-smoothing dissimilarities and study its effects on cluster accuracy when the $P = 10$ latent features are considered to be mutually independent and pairwise correlated. We vary the within-cluster variation and between cluster separation to simulate a noisy underlying structure and judge our method's performance when using the Average Linkage and K-Medoids algorithms. The results are discussed in the next section.

Tables 3.9-3.12 show the average ARI values obtained from the cluster solutions formed using the Average Linkage and K-Medoids algorithms. Tables 3.9 and 3.10 give the accuracy when the features are assumed to be mutually independent and Tables 3.11 and 3.12 give the accuracy when the features are assumed to be equally and positively correlated as specified previously. Overall, the highest accuracy was obtained in most cases by clustering with the Average Linkage algorithm. The general trend (though violated in some places) shows an increase in accuracy as the distance between cluster centers increase, and a decrease in accuracy as the intra-cluster variability increases.

Table 3.9 Average ARI Values for the Average Linkage Clustering of Normally Simulated Tertiary Data Assuming Mutually Independent Features.

Average Linkage Method			
δ	$\sigma = 1$	$\sigma = 5$	$\sigma = 10$
0.5	0.9532(0.001) o	0.4826(0.001) o	0.3776(0.003) o
	0.9526(0.001) s/i	0.4809(0.001) s/i	0.4030(0.003) s/i
	0.8482(0.001) s/E	0.0849(0.001) s/E	0.0093(0.000) s/E
	0.7852(0.001) s/H	0.0476(0.001) s/H	0.0096(0.000) s/H
2	0.9711(0.001) o	0.9039(0.002) o	0.5917(0.002) o
	0.9779(0.001) s/i	0.9688(0.001) s/i	0.5561(0.002) s/i
	0.9548(0.001) s/E	0.7079(0.001) s/E	0.0505(0.001) s/E
	0.9244(0.001) s/H	0.7081(0.001) s/H	0.0277(0.000) s/H
3.5	0.9779(0.001) o	0.9737(0.001) o	0.5738(0.002) o
	0.9883(0.000) s/i	0.9810(0.001) s/i	0.5240(0.002) s/i
	0.9772(0.001) s/E	0.8873(0.001) s/E	0.5878(0.002) s/E
	0.9633(0.001) s/H	0.8879(0.001) s/H	0.5066(0.003) s/H

For the Average Linkage algorithm with mutually independent features, we see the highest accuracy is obtained mostly when the observed dissimilarities are presmoothed towards an independence model. However, there are a few cases in which the accuracy obtained using the observed dissimilarities is marginally higher than

Table 3.10 Average ARI Values for the K-Medoids Clustering of Normally Simulated Tertiary Data Assuming Mutually Independent Features.

K-Medoids Method			
δ	$\sigma = 1$	$\sigma = 5$	$\sigma = 10$
0.5	0.5115(0.002) o	0.1515(0.001) o	0.0128(0.000) o
	0.4380(0.000) s/i	0.0546(0.000) s/i	0.0255(0.000) s/i
	0.5499(0.003) s/E	0.0852(0.001) s/E	0.0135(0.001) s/E
	0.5955(0.003) s/H	0.1644(0.001) s/H	0.0132(0.000) s/H
2	0.9049(0.000) o	0.6834(0.001) o	0.1084(0.000) o
	0.4770(0.001) s/i	0.5446(0.001) s/i	0.0284(0.000) s/i
	0.8656(0.000) s/E	0.5975(0.001) s/E	0.0734(0.000) s/E
	0.9168(0.000) s/H	0.6422(0.002) s/H	0.1384(0.001) s/H
3.5	0.9418(0.000) o	0.8599(0.001) o	0.3907(0.001) o
	0.7878(0.000) s/i	0.8638(0.000) s/i	0.1895(0.001) s/i
	0.8840(0.000) s/E	0.8648(0.000) s/E	0.2504(0.001) s/E
	0.9253(0.000) s/H	0.7701(0.001) s/H	0.4893(0.002) s/H

Table 3.11 Average ARI Values for the Average Linkage Clustering of Normally Simulated Tertiary Data Assuming Dependency within the Features.

Average Linkage Method			
δ	$\sigma = 1$	$\sigma = 5$	$\sigma = 10$
0.5	0.9228(0.002) o	0.5126(0.002) o	0.2047(0.003) o
	0.9168(0.001) s/i	0.4820(0.001) s/i	0.2440(0.004) s/i
	0.7950(0.001) s/E	0.0733(0.000) s/E	0.0088(0.000) s/E
	0.7297(0.001) s/H	0.0500(0.000) s/H	0.0106(0.000) s/H
2	0.8982(0.002) o	0.7416(0.002) o	0.4967(0.003) o
	0.9052(0.002) s/i	0.8558(0.002) s/i	0.4931(0.003) s/i
	0.8546(0.001) s/E	0.5058(0.001) s/E	0.0292(0.000) s/E
	0.8006(0.001) s/H	0.5282(0.001) s/H	0.0275(0.000) s/H
3.5	0.8822(0.002) o	0.8846(0.002) o	0.6029(0.002) o
	0.9138(0.001) s/i	0.9083(0.002) s/i	0.5612(0.002) s/i
	0.8843(0.001) s/E	0.7786(0.001) s/E	0.2818(0.002) s/E
	0.8604(0.001) s/H	0.7789(0.001) s/H	0.1981(0.002) s/H

what is obtain by pre-smoothing in the aforementioned manner. We see this happen, for example, when $\delta = 0.5$ and $\sigma = 1$. This may suggest that when there is not a lot of separation between cluster centers, that pre-smoothing may not be as influential as it is when the distance is a bit larger. However, the increase in accuracy in these

Table 3.12 Average ARI Values for the K-Medoids Clustering of Normally Simulated Tertiary Data Assuming Dependency within the Features.

K-Medoids Method			
δ	$\sigma = 1$	$\sigma = 5$	$\sigma = 10$
0.5	0.4906(0.002) o	0.1374(0.001) o	0.0104(0.000) o
	0.4012(0.000) s/i	0.0595(0.000) s/i	0.0253(0.000) s/i
	0.4955(0.002) s/E	0.0880(0.001) s/E	0.0108(0.000) s/E
	0.5305(0.003) s/H	0.1575(0.001) s/H	0.0112(0.000) s/H
2	0.8661(0.000) o	0.5263(0.001) o	0.0982(0.001) o
	0.4446(0.000) s/i	0.3449(0.001) s/i	0.0150(0.000) s/i
	0.8373(0.001) s/E	0.4121(0.001) s/E	0.0518(0.000) s/E
	0.8831(0.000) s/H	0.5013(0.002) s/H	0.1570(0.001) s/H
3.5	0.8922(0.000) o	0.7851(0.000) o	0.2968(0.001) o
	0.7669(0.000) s/i	0.7168(0.001) s/i	0.1394(0.001) s/i
	0.8576(0.000) s/E	0.7736(0.001) s/E	0.1772(0.001) s/E
	0.8855(0.000) s/H	0.7790(0.001) s/H	0.3452(0.001) s/H

cases is only at most by 0.0013 or .13%—that may be a by-chance variation. These results are shown graphically in Figure 3.3.

For the K-Medoids algorithm and mutually independent features, the results are completely different than what we observed with the Average Linkage algorithm. In Table 3.10, the highest accuracy is obtained in most cases when the clustering solutions are formed by pre-smoothing the observed dissimilarities towards a high probability of match model. However, in some places these results contradict what we would expect to happen. For example, for $\delta = 2$ and $\sigma = 1$, the average ARI is 0.4770 when the clusters are formed by pre-smoothing the observed dissimilarities towards an independence model. However, at $\sigma = 5$, for the same value of δ and smoothing model, the average accuracy increases to 0.5446. Intuitively, we would expect the average ARI to decrease as σ increases. Instead, within the K-Medoids algorithm, we see the reverse. Upon further inspection, we also note for the independence model with $\delta = 2$ and $\delta = 3$, the trend shows an increase in accuracy for $1 \leq \sigma \leq 5$ as seen graphically in Figure 3.4. A possible explanation for this may be due to the

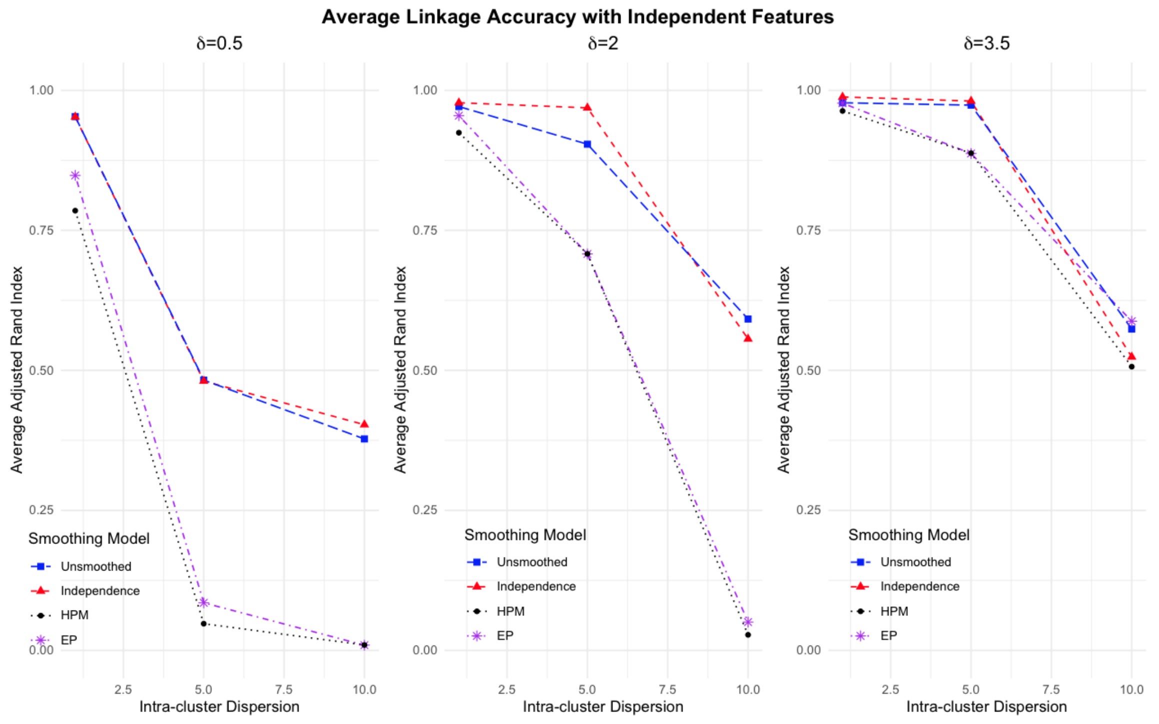


Figure 3.3 The average ARI statistic from 5000 clustering formed using the Average Linkage algorithm while assuming mutually independent features. From left to right, the graphs show the change in average ARI (or accuracy) as the intra-cluster dispersion increases for between-cluster center distances of 0.5, 2, and 3.5 units.

relationship between the two parameters. As mentioned, δ represents the distance between cluster centers and σ represents the intra-cluster dispersion. What may be occurring in these cases is that the distance between cluster centers may be large enough that even with a small bit of intra-cluster dispersion, it is not a challenging clustering problem. Whereas, when $\sigma > 5$, the intra-cluster dispersion overpowers the distance between cluster centers and makes the problem more challenging. In other words, in the latter case, the clusters exhibit more overlap than what is seen in the cases when $1 \leq \sigma \leq 5$. This could potentially explain why an increase would occur. Furthermore, it may occur based on the difference in the way the two algorithms (K-Medoids and Average Linkage) create partitions (see e.g., Section 3.2).

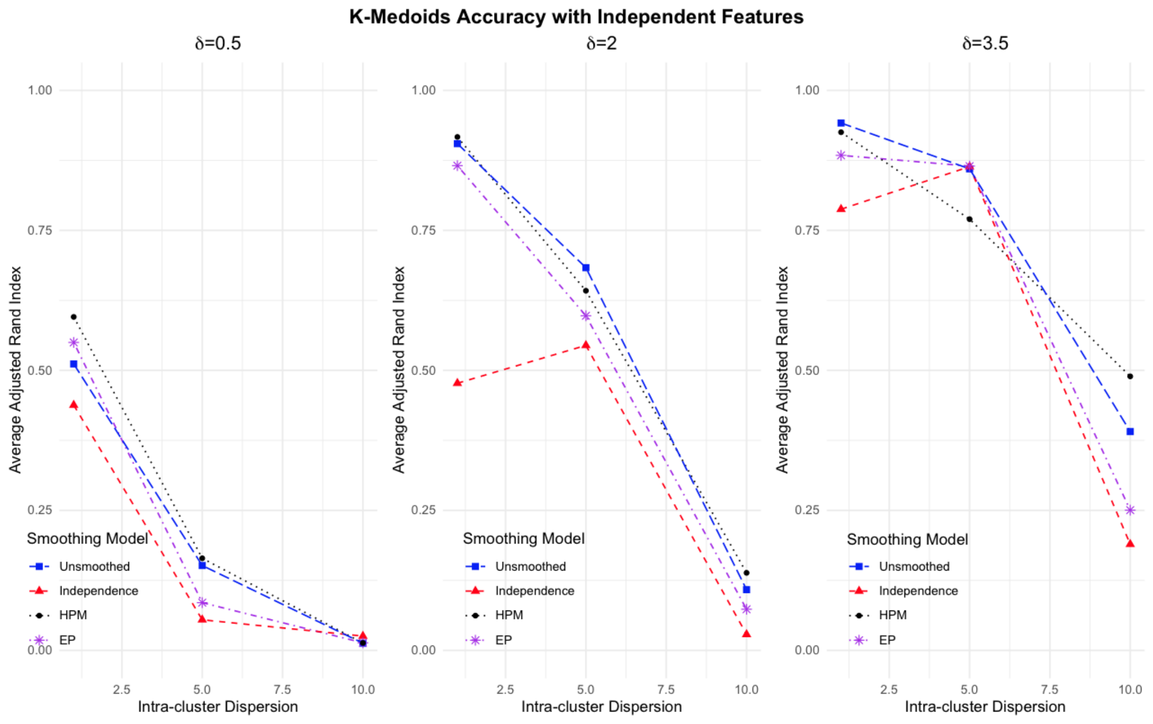


Figure 3.4 The average ARI statistic from 5000 clustering formed using the K-Medoids algorithm while assuming mutually independent features. From left to right, the graphs show the change in average ARI (or accuracy) as the intra-cluster dispersion increases for between-cluster center distances of 0.5, 2, and 3.5 units.

When we examine the results for the clustering partitions created using the Average Linkage algorithm with the assumption that the features are correlated, the results mirror Table 3.9 overall; however, the accuracy is a bit lower in some cases. For example, when the features are assumed to be independent with $\delta = 0.5$, and $\sigma = 10$, the highest accuracy is obtained by smoothing the dissimilarities towards a model of independence. The same is true here; however, the mean ARI is 0.2440 versus the previous 0.4030. Overall, the same findings remain; the highest accuracy is obtained in most cases by smoothing the dissimilarities towards a model of independence with the accuracy increasing as the separation between cluster centers increase and the intra-cluster dispersion decreases. This can be seen in Figure 3.5.

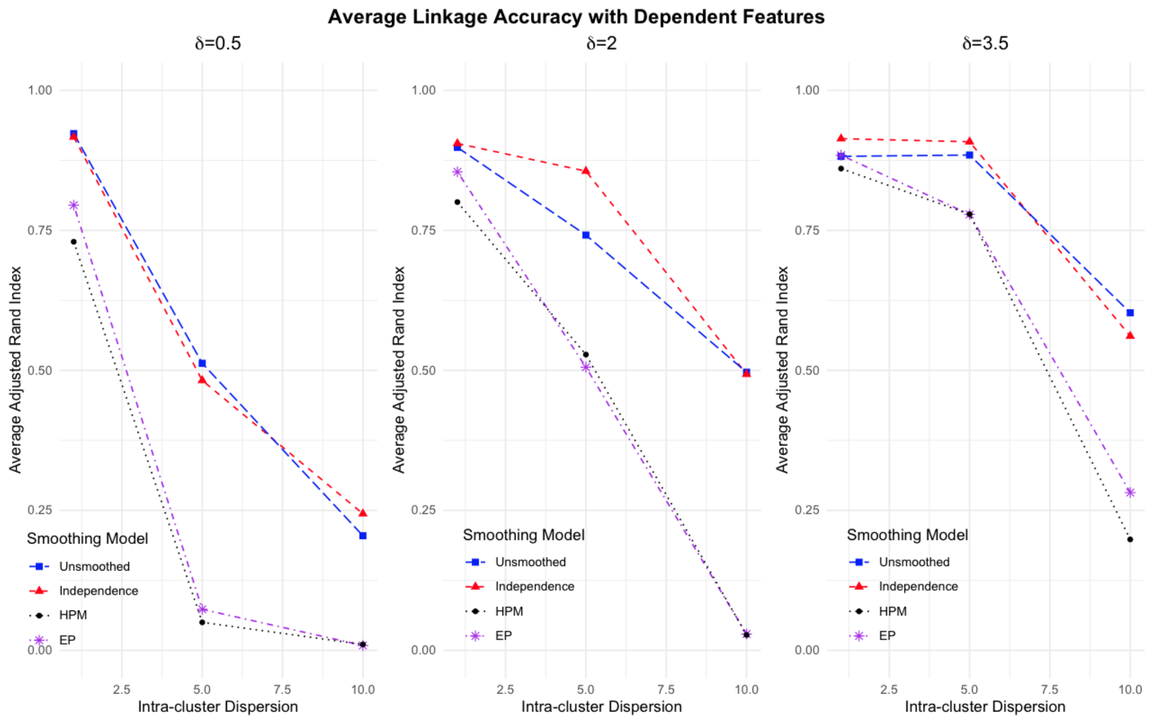


Figure 3.5 The average ARI statistic from 5000 clustering formed using the Average Linkage algorithm while assuming features are positively correlated. From left to right, the graphs show the change in average ARI (or accuracy) as the intra-cluster dispersion increases for between-cluster center distances of 0.5, 2, and 3.5 units.

On the other hand, when assuming the features are correlated and using the K-Medoids algorithm, the results are much more conclusive compared to what was observed in Table 3.10. The findings in this case support the hypothesis that pre-smoothing the dissimilarities improves the accuracy of the clustering formed especially in potentially noisier cases. This can be seen by thoroughly examining the first section of Table 3.12 with $\delta = 0.5$ or by looking at the leftmost plot of Figure 3.6. However, the overall accuracy is much lower than in the previous setting as evidenced by the lower mean ARIs in Table 3.12 compared to Table 3.10. Yet, we still see the accuracy is highest in almost all cases when pre-smoothing the dissimilarities towards a high probability of match model.

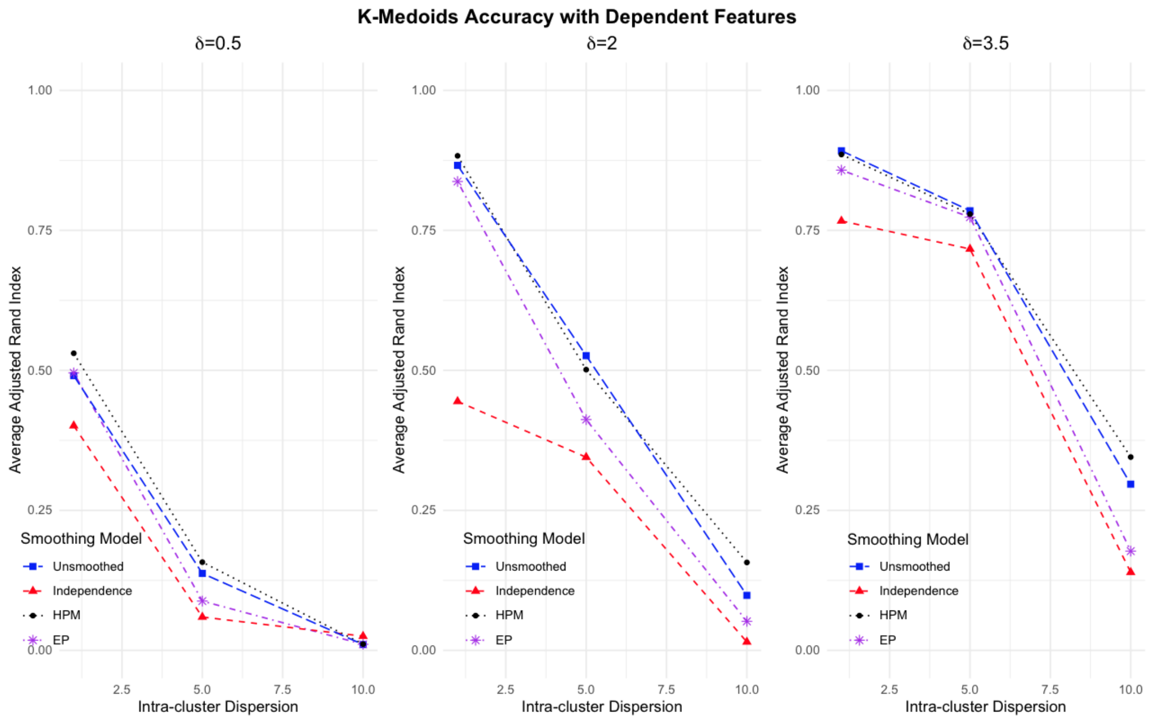


Figure 3.6 The average ARI statistic from 5000 clustering formed using the K-Medoids algorithm while assuming features are positively correlated. From left to right, the graphs show the change in average ARI (or accuracy) as the intra-cluster dispersion increases for between-cluster center distances of 0.5, 2, and 3.5 units.

Overall the simulation results do support the hypothesis that pre-smoothing is influential in some noisier settings as well as in settings where clusters are clearly defined. This seems most evident and useful when using the K-Medoids algorithm in particular, regardless of whether the features are mutually independent or positively correlated. However, the results also show the importance of choosing a smoothing model carefully. For example, in the Average Linkage cases, when using a smoothing model of equal probability or high probability of match, the accuracy suffers significantly. However, in the K-Medoids case, the high probability of match typically gives the best performance, with smoothing towards an independence model often giving a worse performance in terms of clustering accuracy. We can take these results and advise that if pre-smoothing of the dissimilarities will be done, than it should be done with a model of independence when using the Average Linkage algorithm or with the high probability of match model when using the K-Medoids algorithm. A similar conclusion was reached with Hitchcock and Chen (2008).

In Section 3.4, we will compare the performance of clustering pre-smoothed dissimilarities with that of clustering the unsmoothed dissimilarities on a real dataset to assess the proposed method's applicability to such data.

3.4 An Application To Diabetes

We here apply the pre-smoothing method proposed in Section 4.2 of this paper to the Pima Indian Diabetes data, obtained from the UCI Repository for Machine Learning (Dua and Graff, 2019). The original dataset consists of $n = 768$ Pima women with recordings for $P = 8$ variables: Number of pregnancies (Preg), plasma glucose (Glucose), blood pressure (BP), tricep skinfold thickness (Tricep), serum insulin level (Insulin), body mass index (BMI), diabetes pedigree function (Ped), and age (Age) at time of study are recorded. The original dataset also includes a ninth variable denoting diabetes status. We omit the diabetes status variable from the cluster analysis, as we are treating this as an unsupervised problem and will compare our results against the diabetes status grouping. We cluster the data using $C = 2, 3$, and 4 clusters. The value of C that results in the highest average silhouette width (Rousseeuw, 1987) will be used to identify the best number of clusters. Once the optimum C has been identified, clustering results obtained using the Average Linkage and K-Medoids algorithm will be discussed. These results will be shown in Section 3.4.2.

3.4.1 Data Preprocessing and Variable Transformations

The original Pima Indian Diabetes dataset included several observations of 0 for variable measurements, such as plasma glucose level, where such a value is nonsensical. Therefore, such observations were removed from the dataset. The final dataset used in this section thus consists of $n = 391$ observations with $P = 8$ variables.

Since the 8 variables recorded for each subject are either discrete or continuous variables, at the next stage of preprocessing, each variable was converted to a tertiary variable. Measurement values of each variable that fall in the first category are denoted as 0. Measurement values of each variable that fall in the second category are denoted as 1. Measurement values for each variable that fall in the last category are denoted as 2.

The variables *Preg*, *Tricep*, and *Age* were transformed into the categories given based on careful examination of each variables' distribution. The remaining variables were transformed into their respective categories based on practical cutoffs described in readily available literature. For example, research suggests a glucose tolerance test outcome below 140 mg/dL is normal, while a measure between 140 mg/dL and 199 mg/dL is considered pre-diabetic, and a level above 199 mg/dL is considered diabetic (Mayo Clinic, 2019). Similarly, for diastolic blood pressure, research suggests a reading below 80 beats/min is considered normal, whereas a diastolic blood pressure between 80 beats/min and 89 beats/min denotes stage I high blood pressure. The last category combines stage II high blood pressure and hypertensive crisis values (American Heart Association, 2017). The remaining 3 variables were treated similarly. The resulting ordinal categories are shown in Table 3.13.

3.4.2 Results

The average silhouette widths were highest when $C = 2$. Therefore, the results presented in this section assume there are two sub-populations in the Pima Indian Diabetes Dataset.

The original dataset included the diabetes test result for each subject, which we treat as a type of standard against which to compare our clustering results. Note that this may not be a perfect gold standard as a representation of the "true" clustering structure, but it does provide some sort of standard partition to which we can compare our clustering results. It is common practice to use principal component plots to help visualize clusters in lower dimensions (see e.g., Everitt et al. (2011)). We first visually compare our clustering solutions using such plots.

Figures 3.7 and 3.8 show the Average Linkage and K-Medoids clustering, respectively, of the subjects using the unsmoothed dissimilarities and the three models for pre-smoothing the dissimilarities. Note, in Figure 3.7, the Average Linkage algorithm seemingly places negative diabetic test results in cluster 2 when using either the unsmoothed dissimilarities or the dissimilarities pre-smoothed towards an inde-

Table 3.13 This table shows the variable cutoffs used to change the data into tertiary categories. The final categories are ordinal.

	Category		
Variable	1	2	3
Preg	{0,1}	{2,3,4}	over 4
Glucose	under 140	[140,199]	over 199
BP	under 80	[80,89]	90 or more
Tricep	≤ 24	(24,33]	over 33
Insulin	under 16	[16,166]	over 166
BMI	under 24.9	[25,29.9]	30 or more
Ped	under 0.299	(0.299,0.527]	over 0.527
Age	≤ 26	(26,36]	over 36

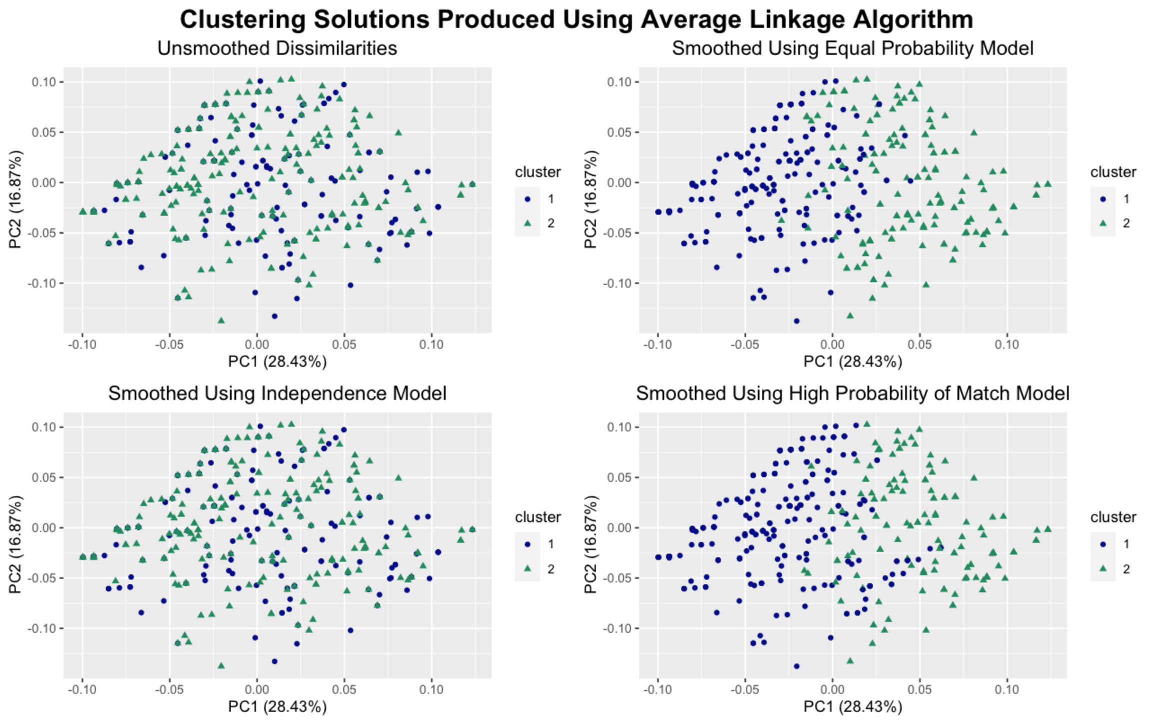


Figure 3.7 The plots above show the clustering of the Pima Indian subjects produced using the unsmoothed dissimilarities (top-left), equal probability pre-smoothed dissimilarities (top-right), independence pre-smoothed dissimiliarities (bottom-left), and high probability of match pre-smoothed dissimilarities (bottom-right) within the Average Linkage clustering algorithm.

pendence model. For the other two smoothing models (equal probability and high probability of match) a negative result appears to correspond to cluster 1 (see Figure 3.9 for reference). The K-Medoids algorithm, on the other hand, seemingly places all negative diabetes test results in cluster 1 regardless of the type of dissimilarities used. (Note that since the numerical labeling of the clusters in the output is arbitrary, it is irrelevant whether a cluster is labeled 1 or 2; what matters is how the individuals are partitioned into the two clusters.)

In Figure 3.7 the dissimilarities pre-smoothed towards a model of equal probability or high probability of match appear to result in a clustering structure with more inter-cluster separation and less overlap than that obtained using the unsmoothed

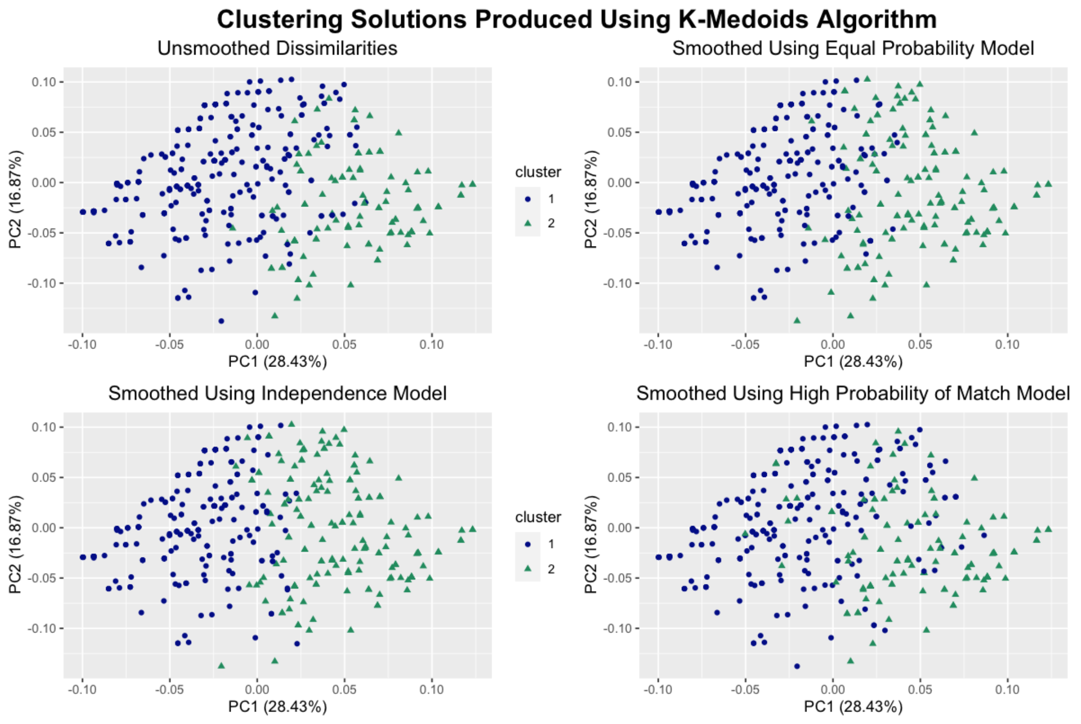


Figure 3.8 The plots above show the clustering of the Pima Indian subjects produced using the unsmoothed dissimilarities (top-left), equal probability pre-smoothed dissimilarities (top-right), independence pre-smoothed dissimiliarities (bottom-left), and high probability of match pre-smoothed dissimilarities (bottom-right) within the K-Medoids clustering algorithm.

dissimilarities or those pre-smoothed towards an independence model. In Figure 3.8, the results are not as definite. Here, the use of any of the options for the dissimilarities (except those pre-smoothed towards a high probability of match model) results in a similar amount of overlap between each cluster. To assess the clustering solutions objectively the ARI is used.

Table 3.14 shows the classification accuracy for each algorithm, as measured by the ARI for the clustering solutions from both smoothing and not smoothing the dissimilarities. The highest accuracy, as indicated by the highest ARI for each algorithm, is denoted in bold for each algorithm. Table 3.14 suggests with each algorithm that the clustering accuracy is highest when the dissimilarities are pre-smoothed. When

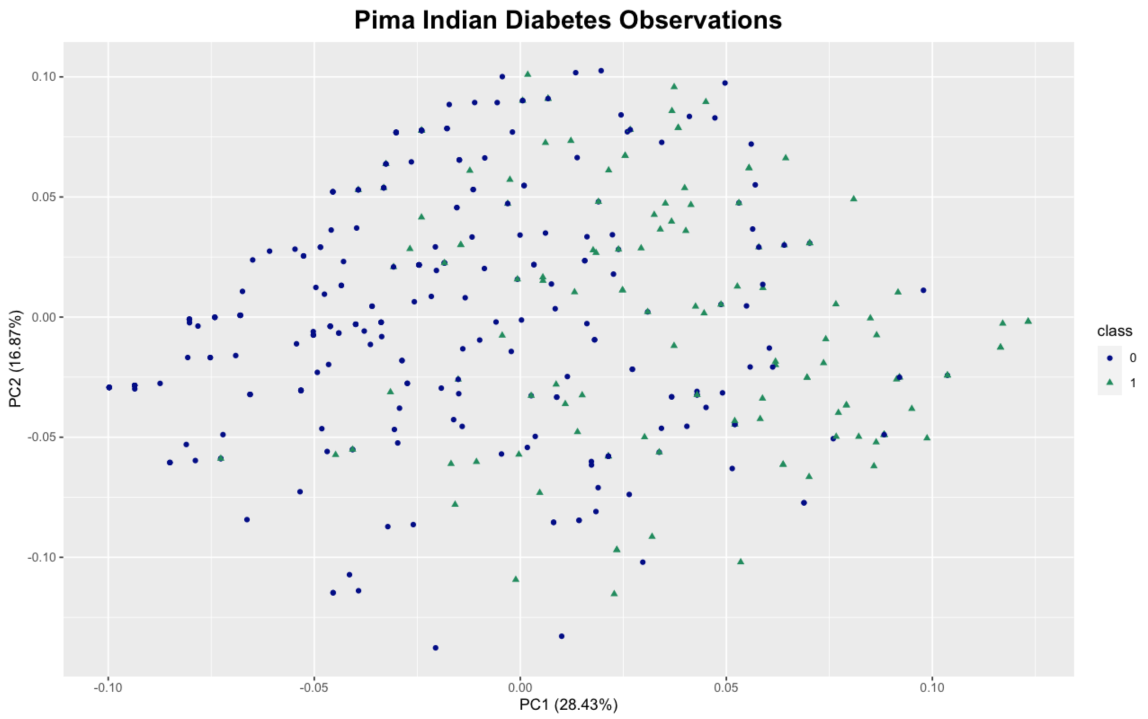


Figure 3.9 The image above shows the Pima observations based on diabetic test results. Observations shown in blue (class=0) represent a negative diabetic test result; whereas observations in green (class=1) represent a positive diabetic test result.

the Average Linkage algorithm is used, the cluster accuracy is the highest when pre-smoothing the dissimilarities towards an equal probability model. The second-best performance was achieved by pre-smoothing towards a high probability of match model. For the K-medoids algorithm, the cluster accuracy is highest for smoothing towards a model of independence, followed by pre-smoothing towards an equal probability model. In both algorithms the highest ARI seems noticeably higher for the leading smoothing method than for the clustering using the unsmoothed dissimilarities. This suggests that in each case, clustering the pre-smoothed dissimilarities (via equal probability or independence) may better reflect the true underlying structure of the data than does the clustering of the unsmoothed dissimilarities.

Table 3.14 ARI values for the clustering of Pima Indian Women using K-medoids and Average Linkage Algorithms using the unsmoothed dissimilarities and three different smoothing methods.

Cluster Goodness		
Smoothing Method	Average Linkage	K-medoids
Unsmooth	0.637	0.640
Equal Probability	**0.725**	0.703
Independence	0.637	**0.736**
High Probability of Match	0.688	0.638

Table 3.15 Confusion Matrix Formed Based on Average Linkage Clustering of Unsmoothed Dissimilarities

Unsmoothed		
Test Result	1	2
Positive	64	66
Negative	96	165

To obtain a better picture of the actual differences in clustering results obtained, Tables 3.15–3.18 are provided. Since the ARI was highest for pre-smoothing dissimilarities towards an equal probability model for the Average Linkage algorithm, Tables 3.15 and 3.16 show the cross-tabulation of outputs of each method compared to the actual diabetic test results obtained from the subjects. Based on this, there are 19 subjects who tested negative for diabetes that are placed in different clusters between the smoothing and non-smoothing methods (about 5% of the observations). Similarly, there are 30 subjects who tested positive for diabetes that are placed in different clusters between the smoothing and non-smoothing method (about 8% of the observations).

For the K-Medoids algorithm, cluster accuracy was highest for the dissimilarities smoothed towards a model of independence. Table 3.17 gives the cross-tabulation of the actual diabetes test results for each subject compared to the cluster result given for the unsmoothed dissimilarities with the K-Medoids algorithm. Table 3.18 shows

Table 3.16 Confusion Matrix Formed Based on Average Linkage Clustering of Dissimilarities Pre-Smoothed under a Model of Independence

Equal Probability		
Test Result	1	2
Negative	184	77
Positive	36	94

Table 3.17 Confusion Matrix Formed Based on K-Medoids Clustering of Unsmoothed Dissimilarities

Unsmoothed		
Test Result	1	2
Negative	218	43
Positive	60	70

Table 3.18 Confusion Matrix Formed Based on K-Medoids Clustering of Dissimilarities Pre-Smoothed under a Model of Independence

Independence		
Test Result	1	2
Negative	187	74
Positive	34	96

the analogous result obtained from usage of the dissimilarities smoothed towards an equal probability model. The two methods result in a difference for 31 and 26 subjects who tested negative and positive, respectively—a difference in clustering output for about 15% of the subjects.

3.5 Discussion

In this paper we proposed a dissimilarity-based method for the clustering of tertiary observations. The proposed method utilizes statistical smoothing to help combat a potentially noisy underlying structure to help aid in recovering the true latent structure from whence observations have arisen. An introduction to the theory behind this method was given in Section 3.1; while instructions for its implementation were given in Section 3.2. In Section 3.3, a simulation study was conducted to assess the method's ability to accurately partition observations into clusters under various settings. Lastly, in Section 3.4, the proposed method was applied to the Pima Indian Diabetes data set.

The results from the simulation study suggest that when the tertiary observations are believed to have arisen from a multinomial setting, more accurate clusters are formed in most cases by using pre-smoothed dissimilarities. Within the Average Linkage algorithm, it appears to be best to pre-smooth the dissimilarities towards a model of independence; whereas, in the K-Medoids algorithm, the high probability of match model appears to be most effective. The main findings suggest pre-smoothing is most influential, in this setting, when there is more overlap between clusters. In the cases when there is much more distance between cluster centers, the accuracy obtained using the pre-smoothed dissimilarities is comparable to using the observed dissimilarities.

When it is assumed the tertiary data have arisen from a latent Gaussian process, the same general findings are found: More accurate cluster partitions are created in general from the Average Linkage algorithm. Furthermore, they suggest that if pre-smoothing will be implemented, when using the K-Medoids algorithm, pre-smoothing towards a model of high probability of match may be best as it results in higher accuracy in most cases; whereas, if using the Average Linkage algorithm, pre-smoothing should be done towards a model of independence. In the Gaussian setting, we also observe that in some cases (more variability within clusters and less separation between centers) the performance of pre-smoothing was marginally below that of using the observed dissimilarities, but was better than that of using the observed dissimilarities as the clusters became more defined.

Another conclusion drawn from the simulation study is that the pre-smoothing appears to be more influential for the K-medoids algorithm rather than the Average Linkage algorithm. As aforementioned, this is most prevalent in the noisier and more variable settings; however, in the more well-separated settings, the accuracy obtained by pre-smoothing and not pre-smoothing are comparable. This may suggest that pre-smoothing may be a good idea to implement regardless of the believed distance between the cluster centers or within cluster variability in many cases, if a good smoothing model that is supported by the data can be applied.

As for the diabetes application, results here suggest cluster partitions produced more accurately reflected the underlying structure of the data and were more comparable to the blood diabetes test results when the pre-smoothed dissimilarities were used rather than when the traditional (non-smoothed) dissimilarities are used.

Overall, the hypothesis that pre-smoothing the observed dissimilarities may result in the formation of clusters that more accurately reflect the true underlying structure seems to be supported in many cases within the multinomial and Gaussian settings. A natural next step would be to explore other methods by which smoothing could be performed and methods to generalize to the K categorical case. Some suggestions in how to do this may consist of putting a Bayesian prior on the smoothing parameter or even exploring other estimators of $\boldsymbol{\pi}$ that could be used in place of the Fienberg-Holland estimator. It is also worth noting that the increase in accuracy resulting from pre-smoothing the dissimilarities within the K-Medoids algorithm suggests pre-smoothing may be more influential when using partitioning-based methods of clustering rather than a hierarchical algorithm. This is promising as it has been noted in many papers that such partitioning-based methods tend to be more computationally efficient than other methods of clustering (see, e.g., Huang (2008)). Consequently, a generalized method could have the ability to impact a variety of fields and applications. Some such tasks may include those of clustering large datasets based on the Likert scale, the clustering of microarray data in genomics, or even images and documents in information retrieval.

Chapter 4
Fuzzy Ensemble-Based Algorithms for Cluster Analysis

The last century has seen a rapid change in the methods through which humans make decisions. This is likely in part due to technological advances that have made it easy to collect data. As a result, we now live in a world in which data controls and directs much of the decision-making processes, not just for individuals, but also for governmental policy makers in education and for marketing corporations around the world (see e.g., Smith (2019), Mandinach (2012), or Elgendy and Elragal (2016)). With such an increase in data and an increased importance placed on it, there is also an increased need to make sense of it. One method used to do this is data mining.

Data mining may be used to refer to any of numerous multivariate analysis techniques that seek to discover meaning within data that may be stored in large databases (Everitt et al., 2011). Some examples include techniques like classification using decision trees, cluster analysis, association rules, or regression-based methods (Borole, 2020). There have been many approaches proposed for each of the statistical methods; however, in practice none of these approaches always preforms best. In fact the astute statistician or data scientist has a repertoire of tools that he or she may employ depending upon the task. A technique that has emerged over the past few years is that of ensemble learning.

Ensemble learners are composite algorithms that are composed of more than one algorithm. Such methods may be used for supervised tasks in which ensemble learners composed of simpler "base" models are used to create a stronger prediction model based off the strengths of each base model (Friedman, Hastie, and Tibshirani, 2017). Ensemble methods may also be used in unsupervised tasks. In fact, another area that has seen an increase in the usage of ensemble approaches is that of cluster analysis (Sarumathi, Shanthi, and Sharmila, 2013).

Cluster analysis is an unsupervised learning method in which a set of objects, patterns, documents, or images are partitioned into homogeneous groups based on a measure of similarity. Within a particular clustering solution, objects placed in the same group possess some property that make them more similar to objects within their own group than to the other objects in other groups. It is referred to as an "unsupervised" method because there are no training data that guide the clustering process. Instead the multivariate observations recorded on each data object are used to cluster the objects. Currently there are many clustering algorithms that have been proposed in literature, which each employ a particular methodology to create clusters. Some popular algorithms include partitioning-based methods like K-Means (MacQueen, 1967), K-Modes (Huang, 1997a), and K-Medoids (Rousseeuw and Kaufman, 1987), fuzzy methods like Fuzzy C-Means (Bezdek, Ehrlich, and Full, 1984), Fuzzy K- Modes (Huang and Ng, 1999), and Fuzzy K-Medoids (AL-Akhras, 2010), density-based methods like DBSCAN (Ester et al., 1996), and hierarchical methods like ROCK (Guha, Rastogi, and Shim, 2000), BIRCH (Zhang, Ramakrishnan, and Livny, 1996), and CURE (Guha, Rastogi, and Shim, 1998). Unfortunately, none of these algorithms is always best as they each may have properties desired in some settings, but not in others. For example, partitioning-based methods of cluster analysis are traditionally known to be computationally efficient (see e.g., Huang (2008) or Huang (1997a)). However, when the desire is to see the hierarchy that may exist in data, a hierarchical method may be preferred. Even within the same method of clustering, one algorithm may be preferred over another. For example, it is common knowledge that the K-Medoids algorithm is more robust to outliers than the K-Means algorithm though they are both partitioning-based methods (Albalate and Minker, 2011). Another problem with cluster analysis is that using different initialization points within the same algorithm may result in different clustering solutions for the same data set. This can even happen when employing two different clustering algorithms (perhaps K-Means versus Average Linkage) on the same data set. Therefore, it should not be surprising that combining different clustering algorithms would be beneficial. To this end, ensemble-based methods of cluster analysis have been proposed (see e.g., Sarumathi, Shanthi, and Sharmila (2013) or Vega-Pons and Ruiz-Shulcloper (2011)).

Ensemble-based methods of cluster analysis are typically composed of several clustering algorithms and consist of two steps—a generation and a consensus step. Traditionally within the generation step, several different clustering partitions are created either from several different clustering algorithms or from initializing several starting points within the same clustering algorithm. In the consensus step, these possibly diverse partitions are combined into a final partition using a consensus function (Vega-Pons and Ruiz-Shulcloper, 2011). The ultimate goal for ensemble-based methods of clustering is to garner a clustering solution that is "better" than what could be obtained by any single iteration of a clustering algorithm in the generation step. The term "better" is subjective and may refer to properties such as robustness, novelty, consistency, or stability (Vega-Pons and Ruiz-Shulcloper, 2011). Recently, several ensemble-based clustering algorithms have been proposed. Such examples include the Ensemble-Based Fuzzy with Particle Swarm Optimization Based Weighted Clustering (EFPSO-WC) algorithm (Thangamani and Ibrahim, 2018) which combined three types of fuzzy clustering algorithms to aid in the clustering of microarray gene expression data and an ensemble for the clustering of text proposed by Mateen et al. (2018) that combines the K-Means, K-Medoids, Gustafson-Kessel, Fuzzy

C-Means and a hierarchical clustering algorithm. Additionally, a survey of ensemble-based methods that have been proposed specifically for the clustering of categorical data can be found in Sarumathi, Shanthi, and Sharmila (2013), while a more general survey of ensemble clustering methods can be found in Vega-Pons and Ruiz-Shulcloper (2011).

In traditional ensemble-based methods of clustering, as aforementioned, the generation step typically consists of creating clustering solutions **solely** from clustering algorithms. As such, none of the previously mentioned clustering ensembles used supervised learners within the generation step to aid in the clustering process. In this paper, however, we propose 3 ensembles for cluster analysis that utilize a fuzzy clustering algorithm *and* a supervised learner within the generation step. When building an ensemble for clustering, Vega-Pons and Ruiz-Shulcloper (2011) suggest using a clustering algorithm within the generation step that allows for more information to be learned about the data. Consequently, we propose the use of the Fuzzy C-Means procedure as the first step in the generation, because as a soft partitioning method, objects belong to each cluster with a certain strength of membership rather than belonging only to one cluster as is traditional in hard clustering methods. As such, we are able to tell which cluster the objects are most similar to, without excluding the possibility that the object may share similarity with another cluster. Furthermore, since cluster analysis is often employed to impose structure on the data when much is not known about the latent underlying structure (see e.g., Simonoff (2012)), we propose using nonparametric learners within the generation step that utilize information from the Fuzzy C-Means algorithm. We use nonparametric methods because if model assumptions for parametric methods are violated, results can be invalid. A similar methodology was employed successfully in opinion mining by Wang et al. (2018). In that paper an ensemble-based method of clustering was proposed in which a slightly amended Fuzzy C-Means algorithm was used in conjunction with support vector machines to create a final clustering solution. In our method, however, we create our clustering partitions in a different manner—one that takes advantage of the membership matrix obtained from the Fuzzy C-Means algorithm in a statistical manner and then uses bagged supervised learners to generate different clustering solutions. These solutions are then combined into one final clustering solution and used to create a new fuzzy membership matrix.

Additionally, this paper explores the performance of the ensembles when the data is assumed to have arisen from various multivariate distributions. In this sense, it provides insight into the performance of such ensembles when additional information is known about the underlying structure of the data.

The outline of the paper is as follows: In Section 4.1, the algorithms within the ensembles are discussed. In Section 4.2 we discuss the ensembles proposed and their implementation. In Section 4.3 a simulation study is conducted to assess the properties of the proposed ensembles as compared to Fuzzy C-Means, and in Section 4.4 our ensembles are applied to two real data sets. We conclude with a discussion of the main paper findings in Section 4.5.

4.1 Background

In this section, the algorithms employed in the ensembles proposed in this paper are discussed. We consider some advantages and disadvantages of each algorithm and note the reasoning for their use in the ensemble. We begin with the first algorithm employed, Fuzzy C-Means.

4.1.1 Fuzzy C-Means

The Fuzzy C-Means algorithm is considered a soft partitioning method. This means objects are not partitioned into only one cluster as is done in a hard clustering method, but instead are a part of each cluster with a certain grade of membership.

To motivate the algorithm, consider first the difference between the two types of partitioning methods.

Let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T$ denote a set of multivariate objects to cluster, and let $C = \{C_1, C_2, \ldots, C_c\}$ denote a set of c disjoint and non-empty partitions of $\boldsymbol{X}$ such that $\bigcup_{j=1}^{c} C_j = \boldsymbol{X}$. Now define $u_j(\boldsymbol{x}_k) = u_{kj}$ where u_{kj} denotes the grade of membership for the kth object in the jth cluster. In the special case in which

$$u_{kj} = \begin{cases} 1 & \text{if } \boldsymbol{x}_k \in C_j \\ 0 & \text{otherwise} \end{cases}$$

$\{u_j : j = 1, 2, \ldots, c\}$ denotes a hard partition of $\boldsymbol{X}$. In the more general case in which $u_{kj} \in [0, 1]$, $\{u_j : j = 1, 2, \ldots, c\}$ is considered a fuzzy partition of $\boldsymbol{X}$. For the Fuzzy C-Means algorithm in particular there are also the additional stipulations that $\sum_{j=1}^{c} u_{kj} = 1$ and $0 \leq \sum_{k=1}^{n} u_{kj} \leq n$.

The algorithm seeks to minimize the objective function

$$Q_v(\boldsymbol{U}, \boldsymbol{m}) = \sum_{j=1}^{c} \sum_{k=1}^{n} u_{kj}^{v} d(\boldsymbol{x}_k, \boldsymbol{m}_j)$$

where $\boldsymbol{U}$ denotes a $n \times c$ matrix that contains the membership grades u_{kj}, v denotes the fuzzifier that controls the amount of fuzziness in the clustering solution, and $d(\boldsymbol{x}_k, \boldsymbol{m}_j)$ refers to the distance between object k and the center of cluster j, $\boldsymbol{m}_j$. While various measures of distance can be used, for the version of the algorithm employed in our paper, this distance refers to the Euclidean distance. Thus we seek to minimize the objective function shown in Equation (4.1). (Note, x_{kp} refers to the pth feature of object k and m_{jp} refers to the pth feature of the centroid of cluster j).

$$Q_v(\boldsymbol{U}, \boldsymbol{m}) = \sum_{j=1}^{c} \sum_{k=1}^{n} u_{jk}^{v} \sqrt{\sum_{p=1}^{P} (x_{kp} - m_{jp})^2} \tag{4.1}$$

Before the Fuzzy C-Means algorithm can commence, values of c and v must be fixed, as well as an ϵ value that measures the accuracy of the output and sets the threshold value at which the algorithm ends. For most applications, v is between 1.5 and 3.0

with $v = 2$ commonly being considered the standard value for the fuzzifier; consequently, $v = 2$ is used within our paper (see e.g., Everitt et al. (2011)). The steps involved in the algorithm are as follows:

1. Input an initial membership matrix, $\boldsymbol{U}^{(0)}$, that contains initial estimates or guesses for u_{kj}. These estimated elements, within each step, are denoted as $\hat{u}_{kj}$.

2. On the s^{th} step, compute an estimate for the center of cluster j, $\hat{\boldsymbol{m}}_j$, as shown in Equation (4.2) for $s = 1, 2, \ldots, S$ where S denotes the maximum number of iterations.

$$\hat{\boldsymbol{m}}_j = \frac{\sum_{k=1}^{n} \hat{u}_{kj}^{v} \boldsymbol{x}_k}{\sum_{k=1}^{n} \hat{u}_{kj}^{v}} \tag{4.2}$$

3. Update $\boldsymbol{U}^{(s)}$ using Equation (4.3).

$$\hat{u}_{kj} = \frac{1}{\sum_{i=1}^{c} \left(\frac{d(\boldsymbol{x}_k, \boldsymbol{m}_j)}{d(\boldsymbol{x}_k, \boldsymbol{m}_i)} \right)^{\frac{2}{v-1}}} \tag{4.3}$$

4. Compare $\boldsymbol{U}^{(s+1)}$ to $\boldsymbol{U}^{(s)}$.

If $d(\boldsymbol{U}^{(s+1)}, \boldsymbol{U}^{(s)}) < \epsilon$ in step 4, then the algorithm ends. If not, steps 2-4 repeat until the constraint is met.

Since its proposal, the Fuzzy C-Means algorithm has been used in many applications. One of the reasons for this may be due to the fact that as a soft clustering method, it provides a natural method through which to measure an object's similarity to every cluster via the fuzzy membership functions. Consequently, one is able to determine if there may be a secondary cluster in which a particular observation shares similarity that is comparable to its similarity to the "best" cluster. Everitt et al. (2011) notes the previously mentioned property as one not often observable when using other clustering algorithms. Another advantageous aspect of Fuzzy C-Means clustering is its ability to identify clusters of different shapes (Bezdek, Ehrlich, and Full, 1984). While the shapes that can be identified will vary based on the norm that is used, when the Euclidean norm is used, as is done in this paper, hyperspherical clusters are identified. In other cases, more hyperellipsoidal clusters may be identified through the utilization of different norms (see Bezdek, Ehrlich, and Full (1984) for more information). Another factor worth noting for the version of the algorithm described here is that the clustering solutions produced provide locally optimum fuzzy clusterings of the original data set when $\mathbf{x}_k \neq \mathbf{m}_j$ $\forall k$ and $\forall j$ (Bezdek, Ehrlich, and Full, 1984). However, as the solutions in this case are locally optimum, they may not necessarily represent the globally optimum solution, which in some cases may be a potential problem.

We use the Fuzzy C-Means algorithm here because Everitt et al. (2011) suggest that when fuzzy membership grades are scaled to be between 0 and 1 there is the ability to interpret the fuzzy grades as probabilities. Since this is true for the Fuzzy C-Means algorithm, the membership grades then provide a natural method through which we can define pseudo-classes to the observations and treat the clustering process as a pseudo-supervised problem. It is in this manner that we use the algorithm in this paper.

4.1.2 k-Nearest Neighbors

The k-nearest neighbor algorithm is a supervised learning method that can be used to classify observations into classes based on the classification of their k closest (called neighbors) points. To classify a particular observation, $\mathbf{x}_i$, the k-nearest neighbor algorithm employs the following steps:

1. Calculate the distance $d(\mathbf{x}_i, \mathbf{x}_{i'})$ between the ith and i'th observations for all $i' \neq i$.
2. Record the classes of the closest k observations as measured by d.
3. Assign the modal class to $\mathbf{x}_i$. (If there is a tie, the decision is randomized).

The $d(\mathbf{x}_i, \mathbf{x}_{i'})$ may be measured using any metric suitable for the data type, but for the purposes of this paper, it refers to Euclidean distance between observation $\mathbf{x}_i$ and $\mathbf{x}_{i'}$.

The algorithm is used here because it has been successfully used in a gamut of classification tasks including that of classifying satellite imagery and handwritten digits, to name a few and can be used to classify observations in cases involving irregular boundaries (Friedman, Hastie, and Tibshirani, 2017). These results have been observed despite the fact that the variability and bias of estimates provided by the nearest neighbor algorithm depend heavily on the value of k (Friedman, Hastie, and Tibshirani, 2017). (Methods like cross-validation can help minimize this problem (see Friedman, Hastie, and Tibshirani (2017) for additional information). Another reason it is used in this paper is because it is known, to be a stable classifier (Friedman, Hastie, and Tibshirani (2017)). With this being the case, it is of interest to determine whether bagging a stable classifier, while not necessary for true classification tasks, has an observable effect on accuracy when used within a clustering task.

4.1.3 Decision Trees

Decision trees are supervised learners that can be used for classification or regression tasks that present solutions in terms of decisions. To do this, trees seek to partition the feature space Ω into disjoint and rectangular regions, R_m, such that $\bigcup_{m=1}^{M} R_m = \Omega$. A function then is fit in each region that is used for classification purposes. The decision tree used in this paper is a CART model that works by recursively making binary splits in the feature space to make classifications.

At the highest point of the tree, the initial split occurs at a *root* node, j, based on the feature upon which an impurity measure is maximized. For our paper we use the Gini index. Using the notation of Friedman, Hastie, and Tibshirani (2017), this can be written as

$$Q_m(t) = \Sigma_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

with m defined as before, $k = 1, 2 \ldots, K$ denoting the class, and $\hat{p}_{mk}$ denoting the proportion of observations from the kth class within the mth node computed as shown in equation (4.4).

$$\hat{p}_{mk} = \frac{1}{N_m} \Sigma_{x_i \in R_m} I(y_i = k) \tag{4.4}$$

In (4.4), N_m refers to the total number of observations from the training set in node m.

The initial split in a tree leads to two partitions within the observations. The recursiveness comes in because the process repeats, with each successive split occurring at a new node represented by the feature that maximizes the Gini index. The process repeats until no more splits can be performed, at which point a terminal node is reached. To classify an observation $\mathbf{x}_i$ at the mth node, the algorithm assigns

$$k(\mathbf{x}_i) = \max_k \hat{p}_{mk}.$$

Several measures may be used to measure the impurity at each node. Other popular methods that can be used in lieu of the Gini index can be found in Friedman, Hastie, and Tibshirani (2017).

Trees are often used in practice because of their ease of interpretability even though as weak and unstable learners, their performance may vary widely based on the training set used to build them (see e.g. Friedman, Hastie, and Tibshirani (2017) or Dietterich (2000)). We use decision trees in an ensemble here because it has been shown that ensemble learners built with decision trees often obtain better predictive performance than when only one tree is used (see e.g Dietterich (2000)). In fact, in the manner in which it is used in this paper (as a bagged classifier) the method has led to good performance in a multitude of learning task (see e.g., Friedman, Hastie, and Tibshirani (2017) or Wang et al. (2018)). Trees are also used here because it is of research interest to explore clustering ensembles built using weaker learners and those built with stronger learners to measure their impact on the clustering accuracy obtained when used in a non-traditional method, as part of an ensemble clustering algorithm.

4.1.4 Support Vector Machines

Support vector machines (SVMs) are supervised learners that too can be used for regression or classification; however, in our paper, we use it as a classifier. SVMs, as described here are the "standard" form of the SVM classifier according to Friedman, Hastie, and Tibshirani (2017) in which there is more overlap between classes rather than clear separation. The version we use in this paper creates a nonlinear boundary in the feature space after first fitting a linear boundary within a transformed version of the feature space. To explain the method we use, we first start with the more simplistic version in which linear boundaries are fit between a pair of classes in the original feature space. We define the procedure in this section using a slightly amended version of the notation of Friedman, Hastie, and Tibshirani (2017).

Denote the observations in the training set as $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$ with $y_i \in \{-1, 1\}$ for $i = 1, 2, \ldots, n$. The SVM classifier seeks to find the optimal hyperplane, $f(x) = \mathbf{x}^T\beta + \beta_0$ that separates the two classes. Let M denote the margin which measures the distance between observations from the different classes closest to the separating hyperplane and set $M = \frac{1}{||\beta||}$. Then the algorithm solves the following optimization problem:

$$\min \|\beta\| \text{ subject to } \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \ \forall i \\ \xi_i \geq 0, \ \Sigma\xi_i \leq K \end{cases} \tag{4.5}$$

In equation (4.5), ξ_i refers to the ith *slack* variable and denotes the proportional amount by which $\hat{f}(\mathbf{x}_i)$ is on the wrong side of its margin, in comparison to the true group label for y_i, while K provides an upper bound on the total number of training observations that are misclassified. Given the resulting values of $\hat{\beta}_0$ and $\hat{\beta}$ (estimates for the true parameters β_0 and β) the decision function is given by $\hat{G}(\mathbf{x}) = sign[\mathbf{x}^T\hat{\beta} + \hat{\beta}_0]$. As aforementioned, this version creates a linear boundary within the feature space.

For the version of the SVM employed within this paper, we use a radial kernel to fit a nonlinear boundary between a pair of classes in the original feature space. In this version, V basis functions, which we will denote as $h_v(\mathbf{x}), v = 1, 2, \ldots, V$ are used to fit the classifier. For this case, equation (4.6) represents a nonlinear function that replaces the hyperplane used in the linearly separable case with $h(\mathbf{x}_i)$ defined as in equation (4.7).

$$f(x) = h(\mathbf{x})^T\beta + \beta_0 \tag{4.6}$$

$$h(\mathbf{x}_i) = (h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \ldots, h_v(\mathbf{x}_i)) \tag{4.7}$$

Friedman, Hastie, and Tibshirani (2017) shows that equation (4.6) can be written as in (4.8):

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle h(\mathbf{x}), h(\mathbf{x}_i)\rangle + \beta_0 \tag{4.8}$$

with the inner product in equation (4.8) being replaced with a kernel function, $K(\mathbf{x}, \mathbf{x}^{'}) = \langle h(\mathbf{x}), h(\mathbf{x}^{'})\rangle$. They note that this allows a manner through which to sidestep the identification of the basis functions $h_v(\mathbf{x})$. For our paper, we employ the radial kernel shown in equation (4.9) where γ is a tuning parameter that can be identified using cross-validation; however, other popular kernel choices can be found in Friedman, Hastie, and Tibshirani (2017).

$$K(\mathbf{x}, \mathbf{x}^{'}) = \exp(-\gamma||x - x||^2) \tag{4.9}$$

SVMs are considered in this paper as they are stable classifiers and are known to have high accuracy when used for classification tasks. The version used in this paper, as mentioned, is expected to perform well in cases when there is overlap between classes and allows for a certain amount of observations to be misclassified to deal with this issue (Friedman, Hastie, and Tibshirani, 2017). Because we want to examine the performance of the ensembles both when the clusters are well defined and not so well defined, this particular property may prove to be invaluable. A downside to the SVM is that it can be sensitive to the tuning parameters that are used to train the model. However, using cross-validation can deal with this issue. We should also note that the algorithm described in this section was described in the two-class case. For the method employed in this paper, there are more than two classes. This does not pose a problem as the SVM in the multiclass proceeds by fitting several two-class problems and choosing the best classification based on these (see e.g., Friedman, Hastie, and Tibshirani (2017)).

4.2 Method

In this section we discuss the methodology employed to create the ensembles proposed in this paper using the algorithms of Section 4.1. The methodology used here differs from the traditional way in which clustering ensembles are built; consequently, we begin with a sketch of the algorithm in Section 4.2.1. We then proceed to explain it in the traditional framework of an ensemble based clustering method, with additional justification for the steps provided in Sections 4.2.2 and 4.2.3.

4.2.1 The Ensemble Sketch

1. Divide the dataset of size n into a training set, T, of size m and test set, S, of size r using randomization.

2. Perform fuzzy-based clustering on observations within T.

3. Create B new training sets (denoted T_b, where $b = 1, 2, \dots, B$) by randomly sampling m observations from T with replacement.

4. Use the cluster membership grades obtained from step 2 to randomly generate pseudo-classes for observations within training set T_b for each b by treating the membership grades as probabilities.

5. Train the selected supervised learner on the observations within T_b using the pseudo-classes.

6. Apply the trained learner to the observations within S to obtain clustering solutions, S_b, for $b = 1, 2, \dots, B$.

7. Aggregate the B clustering solutions into final hard and soft clustering solutions.

4.2.2 The Generation Step

The generation step for each of the proposed ensembles is composed of Steps 1-6 of Section 4.2.1. The 3 ensembles considered in this paper only differ in step 5 in which the different classifiers of Section 4.1 are used. We develop our clustering ensemble after seeing ensembles of supervised learners perform in a successful manner (see e.g., Vega-Pons and Ruiz-Shulcloper (2011) or Sarumathi, Shanthi, and Sharmila (2013)). We mirror this process as closely as possible to determine whether the same results can be observed when the ensemble method is used for clustering instead; consequently, step 1 is done to mimic the training and test sets that would be used within the generation step of a supervised ensemble. Step 2 provides the mechanism through which we can place the clustering problem within the framework of an ensemble classification model since, as mentioned, the membership grades can be interpreted as probabilities. Thus they provide a natural manner through which to randomly assign class labels. In steps 3-6 we use the membership grades obtained from fuzzy clustering to label the observations within the training set and employ bagging with the traditional classifications obtained corresponding to clustering solutions. (Note, our labels are artificial labels as opposed to supervised classification tasks in which the labels are truly observed).

Bagging is used, because it is an ensemble learning method that has been shown to perform well in prediction tasks and is one of the ways in which supervised prediction ensembles can be created (see e.g., Dietterich (2000) and Friedman, Hastie, and Tibshirani (2017)). As this process often results in better performance by the ensemble than what is obtained by any one base learner when used for classification, we use it in the clustering ensembles to determine if the same can be done in clustering. More specifically, we use bagging here to determine whether the proposed ensembles create more accurate clustering solutions than what would be created if we were to use the Fuzzy C-Means algorithm alone.

4.2.3 The Consensus Step

Step 7 of the aforementioned algorithm represents the consensus step. In this step we use a relabel and voting method to obtain the final hard and soft clustering solutions. To explain the relabelling, we begin with the completion of step 6 in Section 4.2.1 in which all clustering solutions, $S_1, S_2, \ldots, S_B$ have been created. In

Table 4.1 Possible Relabeling of 3 Clusters

Cluster		
1	**2**	**3**
1	2	3
1	3	2
2	1	3
2	3	1
3	2	1
3	1	2

the relabeling process, we assume S_1 represents the true cluster labelling. Then for the remaining $B-1$ clustering partitions, we relabel them based on the labelling that maximizes the Adjusted Rand Index (Hubert and Arabie, 1985) between the 1st and bth partition. To do the relabelling, we consider all possible relabels of solutions. For example, if there are 3 clusters in a dataset, the different clusters can be denoted as 1, 2, and 3. In this case, there are 6 possible relabelling schematics possible as shown in Table 4.1. (In the general case with J clusters, there are $J!$ possible relabelings). S_b's original clustering solution is relabelled using all 6 possible labelings and the final solution, S_b^*, is denoted in the labeling that maximizes the ARI between S_1 and S_b. To demonstrate this, consider Table 4.2 which shows the partition created from S_1 and S_b where $b \neq 1$. In the table, borrowed from McNicholas (2017), A denotes the total number of pair of objects placed in the same groups by S_1 and S_b. D denotes the total number of pair of objects placed in different groups by S_1 and S_b. B denotes the total number of pair of objects placed in different groups in S_1, but the same group by S_b and C refers to the total number of pair of objects placed in the same group by S_1 and in different groups by S_b.

Using Table 4.2 and the notation of McNicholas (2017) the ARI between S_1 and S_b can be computed as

$$ARI = \frac{N(A+D) - [(A+B)(A+C)+(C+D)(B+D)]}{N^2 - [(A+B)(A+C)+(C+D)(B+D)]}$$

where N denotes the total number of pairs possible from the r objects in the test set. As the ARI is an adjustment for possible inflation due to chance agreement between two clustering partitions that can occur with the Rand Index (Rand, 1971), we use it here. In this measure, a value of 0 denotes no agreement between the two partitions while a value of 1 denotes perfect agreement between the two partitions. The better partition is that having the highest ARI. Once this process is done for all $B-1$ clustering solutions, S_1 and the resulting S_b^*, for $b = 2, 3, \ldots, B$ are then used to obtain the fuzzy and hard solutions resulting from the ensemble. Let p_{ij} denote the proportion of the B relabeled clustering solutions in which object i is in the jth cluster. The fuzzy solutions then can be represented in a $r \times J$ matrix as

$$\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1J} \\ \vdots & \vdots & \vdots & \vdots \\ p_{i1} & p_{i2} & \cdots & p_{iJ} \\ \vdots & \vdots & \vdots & \vdots \\ p_{r1} & p_{r2} & \cdots & p_{rJ} \end{bmatrix}.$$

where the membership grades for object i are stored in the ith row. The hard solution for object i is found as

$$c(\boldsymbol{x}_i) = \max_j p_{ij}$$

where $c(\boldsymbol{x}_i) \in \{1, 2, \ldots, J\}$.

4.2.4 Algorithms Used

To build the ensembles R (R Core Team, 2019) was used. To implement the k-nearest neighbor algorithm, the `knn` function in the `class` package (Venables and Ripley, 2002) was used. To implement the CART classification tree, the `rpart` function was used from the `rpart` package (Therneau and Atkinson, 2019). Lastly, for the support vector machine and the Fuzzy C-Means algorithms, the `svm` and the `cmeans` functions of the `e1071` package (Meyer et al., 2019), respectively, were used.

Table 4.2 Cross-Tabulation of Two Partitions

	$\boldsymbol{S}_1$	
$\boldsymbol{S}_b$	Same Group	Different Group
Same Group	A	B
Different Group	C	D

4.3 Simulation Study

In this section we discuss a simulation study undertaken to assess the performance of the ensemble algorithms proposed in this paper. We consider the performance of each ensemble as compared to the Fuzzy C-Means algorithm when the data has been generated from a mixture of Normal distributions, Lognormal distributions, and Student t's distributions. We look in particular at the accuracy of the ensembles proposed as measured by the average ARI when examining the hard solutions and at the average absolute loss when comparing the soft clustering solutions. The average absolute loss is first discussed.

4.3.1 Average Absolute Loss

The absolute loss is a way to gauge the effectiveness of each soft solution. To compute this, we first create a $r \times J$ truth matrix, $\boldsymbol{M}$, such that:

$$\{m_{ij}\} = \begin{cases} 1 & \text{for } i \in j \\ 0 & \text{otherwise} \end{cases},$$

where i refers to the ith observation in the pseudo-testing set of size r and j refers to the jth cluster with $j = 1, 2, \ldots, J$ total clusters in the dataset. We next create a $r \times J$ fuzzy membership matrix, $\boldsymbol{F}$ such that

$$\{f_{ij}\} = \begin{cases} \pi_{ij} & \text{for } i \in j \\ 0 & \text{otherwise} \end{cases},$$

where π_{ij} denotes the membership probability for observation i in cluster j. Then, the absolute loss is found as:

$$L = \sum_i \sum_j |m_{ij} - f_{ij}|$$

A better soft clustering solution is one that has a lower absolute loss. To gauge the soft accuracy for the simulations presented in this section, we take the average absolute loss for each clustering ensemble under each clustering scenario.

4.3.2 Normal Simulation Setup

When performing the normal simulations, we assume the n data objects to be clustered have arisen from J clusters. We assume each object has P attributes recorded on each of them with each object's features having arisen from a latent Gaussian normal process. To simulate this, we generate each object in the jth cluster such that $\boldsymbol{Y}_i \sim N_P(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}), i = 1, 2, \ldots, n_j, j = 1, 2, ...J, \sum_{j=1}^{J} n_j = n,$ and $J \leq n$.

For feature generation, we consider two settings. In setting I, we assume the P features are mutually independent and set $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_P$. In setting II, we assume the P latent features are positively (and equally) correlated with

$$\{\sigma^2_{pp'}\} = \begin{cases} \sigma^2 & \text{for } p = p' \\ \frac{\sigma^2}{10} & \text{for } p \neq p' \end{cases} .$$

Once the features have been generated in each setting, the resulting measurements for the ith object are stored in the ith row of the data matrix, $\boldsymbol{Y}$,

$$\boldsymbol{Y} = \begin{bmatrix} Y_{11} & Y_{12} & \ldots & Y_{1P} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{i1} & Y_{i2} & \ldots & Y_{iP} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{n1} & Y_{n2} & \ldots & Y_{nP} \end{bmatrix} .$$

We divide $\boldsymbol{Y}$ into two sets, a testing set, $\boldsymbol{T}$, and a training set, $\boldsymbol{S}$. To do this, we randomly sample 40% of the observations from $\boldsymbol{Y}$ (with replacement) to use as the training set. The remaining observations are used as the testing set. Once these two sets are created the ensemble procedure as discussed in Section 4.2.1 is performed.

For the simulation study presented here we generate 1000 datasets. Each dataset consists of $n = 300$ objects with $P = 10$ features. We assume these objects have arisen from $J = 3$ subpopulations with $n_1 = 100$, $n_2 = 100$, and $n_3 = 100$. Their P features are simulated as follows: We generate $\boldsymbol{Y}_i \sim N_P(-\boldsymbol{\delta}, \boldsymbol{\Sigma})$ for $i = 1, 2, ..., n_1$,

$\boldsymbol{Y}_i \sim N_P(\mathbf{0}, \boldsymbol{\Sigma})$ for $i = 1, 2, ..., n_2$, and $\boldsymbol{Y}_i \sim N_P(\boldsymbol{\delta}, \boldsymbol{\Sigma})$ for $i = 1, 2, ..., n_3$ for the ith observation in cluster 1, 2, and 3, respectively.

Parameter Settings

One goal of this paper was to determine if using an ensemble-based method of clustering would produce more accurate clustering solutions than the Fuzzy C-Means algorithm; as well as to determine in what settings the different ensembles perform most effectively. To do this, we vary the structure of the data generated and used in the clustering process. We simulate this by varying the δ and σ parameters used to generate the data. In our simulation, δ represents the distance between the cluster centers and σ represents the within-cluster variability. Holding σ constant and increasing δ constitutes an easier clustering problem. On the other hand, holding δ constant and increasing σ represents a harder clustering problem. We look in particular at the performance of the proposed ensembles for $\delta \in \{0.5, 2, 3.5, 5\}$ and $\sigma \in \{1, 3, 5, 10\}$.

Once each dataset has been generated, $\boldsymbol{T}$ and $\boldsymbol{S}$ are randomly created. The procedure as indicated in Section 4.2.1 is then implemented with $B = 200$ clustering solutions created. For the particular learners, KNN, decision tree, and SVM, the tuning parameter values were obtained from using cross-validation after 1 iteration of the Fuzzy C-Means labelling procedure. The obtained values were then held constant for all of the normal simulations. The resulting tuning parameter values were as follows: For the KNN, the 9 nearest neighbors were used for classification. For the decision tree, a cost pruning value of 0.10 was used and for the support vector machine, $\gamma = 0.01$ and a cost of 1 were used. This method of choosing parameters served as a guiding method to keep comparisons fair. Perhaps future research can be done on the best methods to choose tuning parameters for ensemble based methods of clustering that utilize supervised learners.

We gauge the accuracy obtained by each of the clustering methods. When considering the hard solutions, we use the average ARI obtained over the 1000 datasets to measure accuracy, and when considering the soft clustering solutions, we utilize the average absolute loss. A higher average ARI indicates a better hard solution while a lower average absolute loss indicates a better soft clustering solution.

Results

Tables 4.3 and 4.4 show the accuracy obtained by each of the ensemble methods proposed as well as that by the Fuzzy C-Means algorithm when the features observed on each object are mutually independent. In most cases, the highest accuracy, as measured by the average ARI, is obtained when clusters are produced with the Fuzzy C-Means algorithm. This is most noticeable in the cases where the clusters are less defined as evidenced by more overlap between the clusters and more variability within the clusters themselves. However, when there is more separation and more clearly defined clusters, the proposed ensembles appear to perform better. In several cases (for e.g., when $\delta = 3.5$ and $\sigma = 1$) the accuracy obtained by the ensembles based on the KNN and the SVM match that of the Fuzzy C-Means algorithm. In others, for example when $\delta = 3.5$ and $\sigma = 3$ or $\sigma = 5$, the performance of the Fuzzy C-Means algorithms is only marginally better than the performance of the KNN and SVM ensembles. Among the ensembles themselves, the highest accuracy tends to be given by the the ensemble with SVM as the base learner followed by the ensemble that uses the KNN in most cases. This suggests that in the case of data that has arisen from a latent Gaussian process with mutually independent features, it is better to use the Fuzzy C-Means algorithm in general if interest is in the hard solutions; however, if there is evidence that the clusters are clearly defined, then the ensemble based method that utilizes the SVM or KNN are also viable options. The ensemble based on the decision tree, in this setting, yields the worst performance in every case. So it would be better in this case to not use such an ensemble. These results can be seen graphically in Figure 4.1.

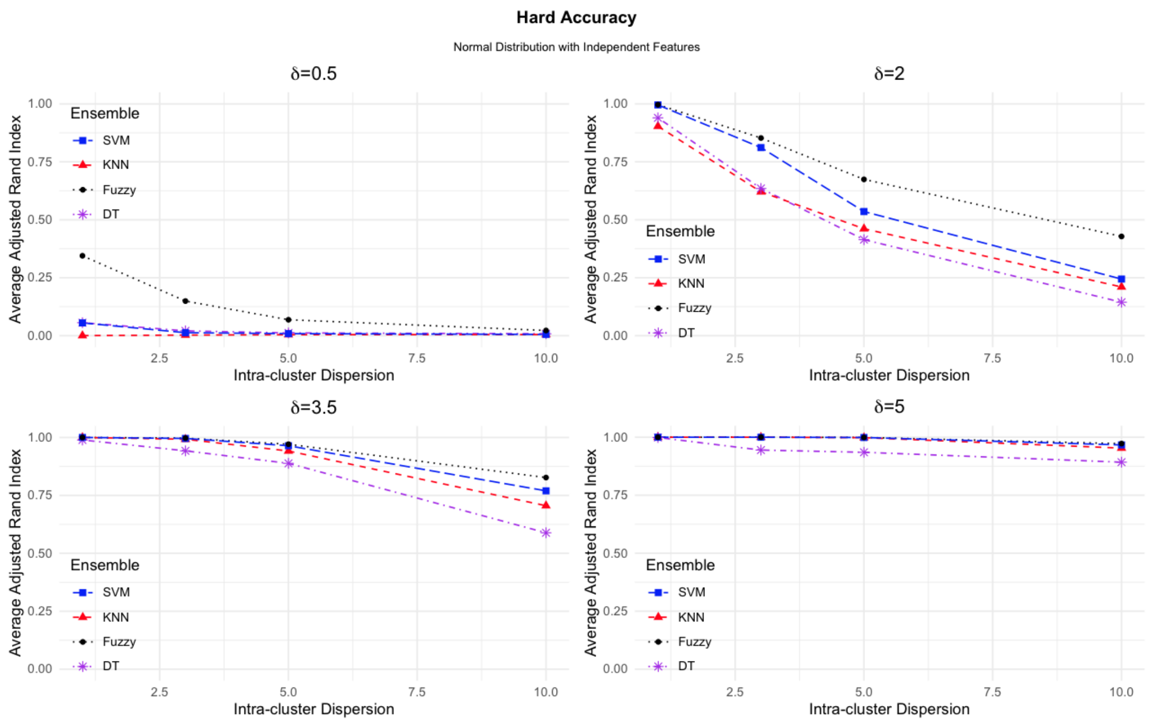


Figure 4.1 Hard accuracy of the clustering solutions produced by each clustering method when the data objects have arisen from a latent Gaussian process with independent features.

When examining Table 4.4, no clear patterns can be discerned. The fuzzy results merely suggest that each of the clustering methods (Fuzzy C-Means and each proposed ensemble) seems to be consistent in its accuracy. This is evidenced by the small change in the average absolute mean loss obtained by each method irrespective of the values of δ and σ. However, it is also worth noting that in this case the mean absolute loss is lowest for the SVM ensemble. In fact, in this table, there are instances in which each proposed ensemble gives lower mean loss than the Fuzzy C-Means algorithm. Together Tables 4.3 and 4.4 suggest that there is a clear better clustering method (Fuzzy C-Means) when one is interested in only the hard clustering solution.

However, if one is interested in the fuzzy clustering, any of the methods may be used regardless of the variability within the clustering process.

Tables 4.5 and 4.6 show the accuracy obtained by each clustering method when the features are assumed to be positively and pairwise correlated. Compared to Table 4.3, the overall average accuracy measured over the 1000 datasets is lower in nearly every case except for when $\delta = 0.5$ for the proposed ensembles. When $\delta = 0.5$, each of the proposed ensembles has a higher average ARI than in the previously mentioned table. However, the highest accuracy values from the proposed ensembles are still lower than those from the Fuzzy C-Means algorithm, in this case with an average ARI of 0.1309 versus 0.1431 and 0.0494 versus 0.0581, for the SVM ensemble versus the Fuzzy C-Means algorithm when $\sigma = 1$ and $\sigma = 5$, respectively. Overall Table 4.5 shows the same general findings and suggests that the proposed ensemble that works best in the cases where the clusters are more clearly defined is the SVM ensemble, whose accuracy matches that of the Fuzzy C-Means algorithm when $\delta = 5$ and $\sigma = 1$. These results suggest that if an ensemble-based method will be used, it is better to use the SVM method; however, the best hard accuracy is obtained by the Fuzzy C-Means algorithm when the data is believed to have arisen from a latent Gaussian process with dependent features.

Table 4.3 Hard Accuracy for Normally Distributed Data with Independent Features

Accuracy of Hard Clustering Solutions				
δ	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 10$
	0.0001(0.0000) KNN	0.0024(0.0003) KNN	0.0041(0.0003) KNN	0.0048(0.0002) KNN
0.5	0.0559(0.0016) D.T	0.0206(0.0007) D.T	0.0117(0.0004) D.T	0.0081(0.0003) D.T
	0.0550(0.0024) SVM	0.0126(0.0007) SVM	0.0086(0.0005) SVM	0.0053(0.0003) SVM
	0.3446(0.0012) Fuzzy	0.1492(0.0015) Fuzzy	0.0686(0.0013) Fuzzy	0.0223(0.0007) Fuzzy
	0.9034(0.0029) KNN	0.6205(0.0032) KNN	0.4612(0.0021) KNN	0.2100(0.0041) KNN
2	0.9397(0.0018) D.T	0.6351(0.0037) D.T	0.4142(0.0022) D.T	0.1442(0.0032) D.T
	0.9955(0.0003) SVM	0.8114(0.0025) SVM	0.5354(0.0030) SVM	0.2441(0.0046) SVM
	0.9965(0.0002) Fuzzy	0.8527(0.0013) Fuzzy	0.6742(0.0017) Fuzzy	0.4280(0.0013) Fuzzy
	1.0000(0.0000) KNN	0.9923(0.0003) KNN	0.9416(0.0011) KNN	0.7055(0.0028) KNN
3.5	0.9891(0.0006) D.T	0.9423(0.0018) D.T	0.8883(0.0019) D.T	0.5887(0.0035) D.T
	1.0000(0.0000) SVM	0.9958(0.0003) SVM	0.9642(0.0007) SVM	0.7697(0.0029) SVM
	1.0000(0.0000) Fuzzy	0.9970(0.0002) Fuzzy	0.9708(0.0007) Fuzzy	0.8270(0.0015) Fuzzy
	1.0000(0.0000) KNN	1.0000(0.0000) KNN	0.9981(0.0001) KNN	0.9531(0.0009) KNN
5	0.9989(0.0001) D.T	0.9446(0.0019) D.T	0.9351(0.0022) D.T	0.8925(0.0019) D.T
	1.0000(0.0000) SVM	1.0000(0.0000) SVM	0.9988(0.0001) SVM	0.9668(0.0007) SVM
	1.0000(0.0012) Fuzzy	1.0000(0.0000) Fuzzy	0.9990(0.0001) Fuzzy	0.9724(0.0006) Fuzzy

Table 4.4 Soft Accuracy for Normally Distributed Data with Independent Features

Accuracy of Soft Clustering Solutions				
δ	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 10$
0.5	270.948(0.2507) KNN	268.4766(0.2058) KNN	268.1215(0.1889) KNN	267.8040(0.1743) KNN
	267.7189(0.2840) D.T	267.9782(0.2111) D.T	268.0893(0.1904) D.T	267.8851(0.1760) D.T
	268.4669(0.3944) SVM	268.2381(0.2347) SVM	268.1932(0.2154) SVM	267.9770(0.1931) SVM
	268.0003(0.1480) Fuzzy	268.0641(0.1508) Fuzzy	268.1306(0.1498) Fuzzy	267.8907(0.1458) Fuzzy
2	270.2662(2.6785) KNN	268.5003(2.7062) KNN	268.8103(2.0337) KNN	268.6427(0.8850) KNN
	271.7535(2.9374) D.T	267.1564(1.7660) D.T	267.9283(1.2208) D.T	268.0789(0.4672) D.T
	272.2513(4.0979) SVM	267.6436(3.0406) SVM	267.1379(2.2680) SVM	268.7998(1.0097) SVM
	269.2180(2.7890) Fuzzy	267.3035(1.6889) Fuzzy	269.2985(1.1693) Fuzzy	267.9921(0.3664) Fuzzy
3.5	267.456(4.2168) KNN	259.6945(4.3199) KNN	261.7489(3.9249) KNN	264.7711(2.8523) KNN
	267.3880(3.8022) D.T	262.1644(3.1632) D.T	264.0424(2.6109) D.T	267.1636(1.7335) D.T
	267.6301(4.2369) SVM	260.0747(4.4003) SVM	262.1026(4.0796) SVM	265.0330(3.0089) SVM
	271.1741(3.6140) Fuzzy	268.7908(2.7810) Fuzzy	267.0282(2.3744) Fuzzy	264.8604(1.6674) Fuzzy
5	262.6846(4.2677) KNN	269.7865(4.1821) KNN	259.8997(4.3385) KNN	266.4836(3.8489) KNN
	262.9342(4.1013) D.T	269.8081(3.4997) D.T	261.9558(3.2729) D.T	268.4169(2.5568) D.T
	262.6980(4.2689) SVM	269.8336(4.1907) SVM	259.9042(4.3700) SVM	266.6566(3.9501) SVM
	267.2087(3.8548) Fuzzy	266.0577(3.4517) Fuzzy	265.8562(3.0330) Fuzzy	265.6818(2.4383) Fuzzy

Table 4.5 Hard Accuracy for Normally Distributed Data with Mutually Dependent Features

Accuracy of Hard Clustering Solutions				
δ	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 10$
0.5	0.0518(0.0015) KNN	0.0335(0.0008) KNN	0.0234(0.0006) KNN	0.0147(0.0004) KNN
	0.1167(0.0012) D.T	0.0420(0.0007) D.T	0.02446(0.0006) D.T	0.0127(0.0004) D.T
	0.1309(0.0012) SVM	0.0494(0.0007) SVM	0.0301(0.0006) SVM	0.0158(0.0004) SVM
	0.1431(0.0011) Fuzzy	0.0581(0.0007) Fuzzy	0.0363(0.0007) Fuzzy	0.0186(0.0005) Fuzzy
2	0.5859(0.0029) KNN	0.3778(0.0020) KNN	0.2810(0.0018) KNN	0.1794(0.0014) KNN
	0.7407(0.0018) D.T	0.4036(0.0017) D.T	0.2862(0.0016) D.T	0.1732(0.0014) D.T
	0.7802(0.0016) SVM	0.4302(0.0017) SVM	0.3087(0.0016) SVM	0.1912(0.0013) SVM
	0.7965(0.0015) Fuzzy	0.4436(0.0017) Fuzzy	0.3209(0.0014) Fuzzy	0.2034(0.0012) Fuzzy
3.5	0.9835(0.0005) KNN	0.7530(0.0019) KNN	0.5917(0.0020) KNN	0.3955(0.0017) KNN
	0.9743(0.0006) D.T	0.7460(0.0018) D.T	0.5893(0.0019) D.T	0.3818(0.0018) D.T
	0.9886(0.0004) SVM	0.7871(0.0016) SVM	0.6198(0.0018) SVM	0.4104(0.0018) SVM
	0.9898(0.0004) Fuzzy	0.8023(0.0015) Fuzzy	0.6371(0.0017) Fuzzy	0.4241(0.0017) Fuzzy
5	0.9998(0.0001) KNN	0.9489(0.0009) KNN	0.8372(0.0015) KNN	0.6097(0.0019) KNN
	0.9946(0.0003) D.T	0.9195(0.0012) D.T	0.8018(0.0018) D.T	0.5933(0.0021) D.T
	0.9998(0.0001) SVM	0.9545(0.0008) SVM	0.8501(0.0014) SVM	0.6261(0.0019) SVM
	0.9998(0.0000) Fuzzy	0.9596(0.0007) Fuzzy	0.8603(0.0013) Fuzzy	0.6441(0.0018) Fuzzy

Table 4.6 Soft Accuracy for Normally Distributed Data with Mutually Dependent Features

Accuracy of Soft Clustering Solutions				
δ	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 10$
	272.2713(0.7479) KNN	267.8064(0.6004) KNN	267.7257(0.4940) KNN	268.0212(0.3998) KNN
0.5	269.0558(0.7488) D.T	267.7318(0.4182) D.T	268.0885(0.3283) D.T	268.090(0.2643) D.T
	270.7736(1.2222) SVM	267.5925(0.7092) SVM	268.0429(0.5440) SVM	268.1742(0.4307) SVM
	268.5164(0.6648) Fuzzy	268.0813(0.3873) Fuzzy	267.9271(0.2956) Fuzzy	268.1760(0.2436) Fuzzy
	267.8107(3.1615) KNN	269.3790(2.3886) KNN	265.5262(1.9788) KNN	263.7016(1.4905) KNN
2	268.2220(2.9073) D.T	269.9730(1.9317) D.T	265.4328(1.4989) D.T	264.2901(1.4905) D.T
	268.7015(3.6686) SVM	270.3842(2.6178) SVM	264.8701(2.1266) SVM	263.6091(1.5714) SVM
	265.2620(2.6952) Fuzzy	270.5767(1.6407) Fuzzy	267.8717(1.3196) Fuzzy	267.6501(0.9112) Fuzzy
	272.4486(4.2059) KNN	265.9061(3.5030) KNN	267.2806(3.1393) KNN	271.3598(2.3776) KNN
3.5	272.2788(3.8204) D.T	266.9722(2.8925) D.T	268.4242(2.5079) D.T	270.7004(1.8342) D.T
	272.7951(4.2404) SVM	266.5332(3.6284) SVM	268.1039(3.2614) SVM	271.6697(2.4835) SVM
	269.6799(3.5243) Fuzzy	264.2019(2.8204) Fuzzy	266.1830(2.2039) Fuzzy	269.7485(1.6194) Fuzzy
	270.7203(4.1840) KNN	275.4992(3.9709) KNN	269.0979(3.7064) KNN	269.9895(3.0953) KNN
5	270.4191(4.0164) D.T	274.7169(3.4089) D.T	268.9617(3.0350) D.T	269.1268(2.4573) D.T
	270.7246(4.1855) SVM	275.6832(3.9985) SVM	269.2045(3.7633) SVM	270.5216(3.1763) SVM
	262.5122(3.9666) Fuzzy	266.2430(3.3985) Fuzzy	262.8911(2.9123) Fuzzy	267.4213(2.2336) Fuzzy

The results of Table 4.5 are shown graphically in Figure 4.2. In the top two plots, we can see a notable difference in the accuracy obtained in the cases where the distance between clusters is smaller. However, the bottom two plots indicate the proposed ensembles obtained accuracy values that are closer to that obtained by the Fuzzy C-Means algorithm.

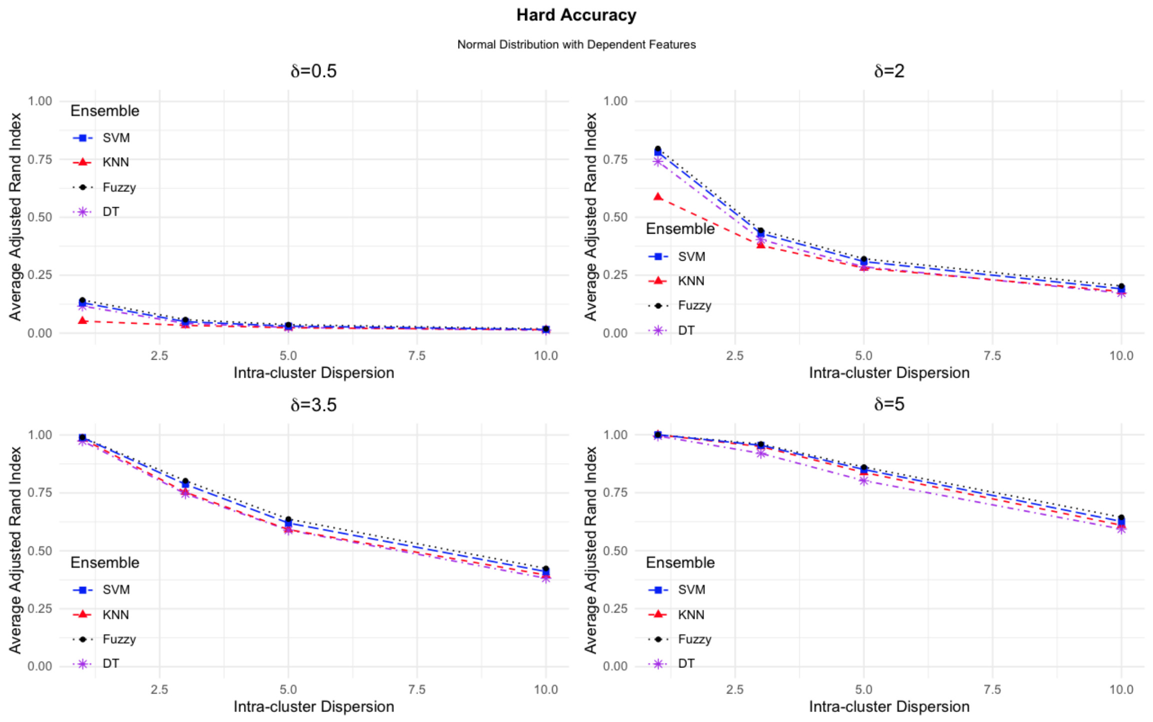


Figure 4.2 Hard accuracy of the clustering solutions produced by each clustering method when the data objects have arisen from a latent Gaussian process with dependent features.

Like Table 4.4, Table 4.6 does not have a clear pattern nor does it differ much from the loss accrued when the features are independent. This seemingly suggests that the fuzzy clustering solutions obtained by all the proposed ensemble clustering methods and the Fuzzy C-Means algorithm are approximately equal in terms of their fuzzy solutions. They do tend to be more volatile than the hard clustering solution;

however, in this case, as before, either of the clustering methods are seemingly viable options regardless of the values of δ and σ.

The findings of the normal simulation study give a few general guidelines about which clustering method should be used in different cases: If one is only interested in the hard clustering solutions, the simulations suggest the Fuzzy C-Means algorithm is best in the case of less defined clusters. If the clusters are clearly defined, regardless of feature independence or dependency, the Fuzzy C-Means algorithm or the SVM ensemble-based method may be best. However, if one is interested in only the fuzzy solution, either of the clustering methods will suffice. Lastly if one is interested in both the hard and fuzzy solutions, the Fuzzy C-Means algorithm appears to be the better option.

4.3.3 Lognormal Simulation Setup

For the lognormal simulations, we assume there are n data objects to be clustered that have arisen from a latent lognormal process. We assume we have observed P features on each object, and that the objects have arisen from J clusters. To generate the features in this case, we assume $\boldsymbol{Y}_i^* \sim \text{Lognormal}_P(\mathbf{0}, \boldsymbol{\Sigma})$, where $\mathbf{0}$ is a column vector of zeroes and $\boldsymbol{\Sigma}$ denotes a $P \times P$ covariance matrix. For the lognormal simulations, we consider the case in which the features are mutually independent as well as, when the features are pairwise positively correlated. When the features are mutually independent, $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_P$, where $\boldsymbol{I}_P$ is the identity matrix multiplied by a vector of ones. Alternatively, in the case of pairwise positively correlated features,

$$\sigma_{pp'}^2 = \begin{cases} \sigma^2 & \text{for } p = p' \\ \frac{\sigma^2}{10} & \text{for } p \neq p' \end{cases} .$$

Parameter Settings

As one of the primary interests of the paper is to judge the proposed ensembles' performance in various settings, we use parameters δ and σ to simulate the variability that may be present between within the clustering task. Specifically, δ represents the variability between clusters, while σ represents the variability within the clusters themselves.

For the lognormal simulation presented here, we generate 1000 datasets each with $n = 300$ objects arising from $J = 3$ clusters. For the clusters themselves, we set $n_1 = 100$, $n_2 = 100$, and $n_3 = 100$. We further assume each object has $P = 10$ features that have been generated from a lognormal process. To do this, we generate our multivariate objects as follows: we generate $\boldsymbol{Y}_i = \boldsymbol{Y}_i^* - \boldsymbol{\delta_i}$ for observations in cluster 1, $\boldsymbol{Y}_i = \boldsymbol{Y}_i^*$ for observations in cluster 2, and $\boldsymbol{Y}_i = \boldsymbol{Y}_i^* + \boldsymbol{\delta_i}$ for observations in cluster 3. To vary the variability in the clustering process, we consider $\delta \in \{0.5, 2, 3.5, 5\}$ and $\sigma \in \{1, 3, 5, 10\}$. In each of the datasets, once the multivariate observations have been generated, the procedure outlined in Section 4.2.1 is followed.

For the ensembles proposed, $B = 200$ clustering solutions were obtained in each parameter setting. The tuning parameters were found using cross validation after 1 iteration of the Fuzzy C-Means labelling procedure. This resulted in the following values for the tuning parameters in each ensemble: For the KNN ensemble, 10 neighbors were used for classification. A cost pruning value of 0.01 was used for the decision tree ensemble. Lastly $\gamma = 0.001$ with a cost of 10 was used to fit the SVM ensemble. These values were held constant over all the lognormal simulations. The results are presented in Section 4.3.3.

Results

Tables 4.7 and 4.8 show the clustering accuracy obtained by each clustering method when the data has arisen from a lognormal process with independent features.

The overall findings suggest for the hard clustering solutions, the best accuracy is obtained from either the Fuzzy C-Means algorithm or the decision tree ensemble-based method of clustering. In particular, Table 4.7 suggests when the clusters are more defined, the decision tree ensemble obtains better accuracy than the Fuzzy C-Means algorithm. For example, when $\delta = 2$ and $\sigma = 1$, the decision tree ensemble obtains an average ARI of 0.8519 while Fuzzy C-Means obtains an average ARI of 0.8154. When σ increases to 3 for the same value of δ, the accuracy obtained by the decision tree ensemble is 0.0910 versus 0.0774 for the Fuzzy C-Means. Similarly, as δ increases for the same σ values, the same relationship between the two clustering methods is observed. However, when σ increases to 5 for $\delta = 2$ and $\delta = 3$, the Fuzzy C-Means methods does better. This may occur due to the relationship between the two parameters. It may be the case that for $\sigma \geq 5$ when δ is equal to 2 or 3.5, that the within cluster variability overpowers the inter-cluster separation making the cluster themselves less defined. After this point, however, the Fuzzy C-Means algorithm performs the best. This can perhaps be seen more clearly in Figure 4.3 where, for $\delta \in \{2, 3.5, 5\}$, each plot shows a notably higher accuracy being obtained by the decision tree ensemble for $0.5 \leq \sigma$. However, when $\sigma \geq 5$, the average ARI values show a leveling trend in which the accuracy obtained by the proposed ensemble becomes marginally different and lower than the average ARI values obtained by the Fuzzy C-Means algorithm.

Table 4.7 Hard Accuracy for Lognormally Distributed Data with Independent Features

Accuracy of Hard Clustering Solutions				
δ	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 10$
0.5	0.0025(0.0003) KNN	0.0018(0.0001) KNN	0.0006(0.0000) KNN	0.0001(0.0000) KNN
	0.0797(0.0023) D.T	0.0130(0.0009) D.T	0.0014(0.0001) D.T	0.0003(0.0000) D.T
	0.0004(0.0001) SVM	0.0007(0.0000) SVM	0.0009(0.0000) SVM	0.0009(0.0000) SVM
	0.1471(0.0013) Fuzzy	0.0059(0.0007) Fuzzy	0.0031(0.0007) Fuzzy	0.0012(0.0000) Fuzzy
2	0.6412(0.0004) KNN	0.0153(0.0012) KNN	0.0006(0.0000) KNN	0.0001(0.0000) KNN
	0.8519(0.0062) D.T	0.0910(0.0033) D.T	0.0020(0.0002) D.T	0.0004(0.0000) D.T
	0.5654(0.0040) SVM	0.0010(0.0001) SVM	0.0009(0.0000) SVM	0.0001(0.0001) SVM
	0.8154(0.0015) Fuzzy	0.0774(0.0011) Fuzzy	0.0042(0.0002) Fuzzy	0.0012(0.0000) Fuzzy
3.5	0.9161(0.0015) KNN	0.3003(0.0035) KNN	0.0011(0.0002) KNN	0.0001(0.0000) KNN
	0.9828(0.0014) D.T	0.4376(0.0044) D.T	0.0050(0.0006) D.T	0.0004(0.0000) D.T
	0.9292(0.0016) SVM	0.0783(0.0037) SVM	0.0009(0.0000) SVM	0.0009(0.0000) SVM
	0.9401(0.0011) Fuzzy	0.2959(0.0012) Fuzzy	0.0089(0.0004) Fuzzy	0.0012(0.0000) Fuzzy
5	0.9759(0.0007) KNN	0.3809(0.0019) KNN	0.0081(0.0011) KNN	0.0001(0.0000) KNN
	0.9960(0.0006) D.T	0.4889(0.0027) D.T	0.0330(0.0027) D.T	0.0004(0.0000) D.T
	0.9784(0.0008) SVM	0.3200(0.0041) SVM	0.0010(0.0000) SVM	0.0010(0.0000) SVM
	0.9807(0.0008) Fuzzy	0.3236(0.0011) Fuzzy	0.0277(0.0010) Fuzzy	0.0013(0.0000) Fuzzy

Table 4.8 Soft Accuracy for Lognormally Distributed Data with Independent Features

Accuracy of Soft Clustering Solutions				
δ	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 10$
0.5	267.6551(0.2334) KNN	267.9251(0.2723) KNN	268.0603(0.2861) KNN	268.3057(0.2930) KNN
	268.4420(0.3323) D.T	268.3003(0.2543) D.T	268.0419(0.2689) D.T	268.2579(0.2912) D.T
	267.8206(0.2745) SVM	267.7875(0.2797) SVM	267.9869(0.2932) SVM	268.2636(0.2989) SVM
	267.9554(0.1465) Fuzzy	267.7497(0.2014) Fuzzy	268.1178(0.2539) Fuzzy	268.0119(0.2870) Fuzzy
2	263.9760(3.1063) KNN	267.8792(0.4506) KNN	267.5476(0.2890) KNN	267.4671(0.2741) KNN
	264.0729(2.4209) D.T	267.6645(0.4940) D.T	267.9099(0.2858) D.T	267.4301(0.2773) D.T
	262.8362(2.9903) SVM	267.4868(0.3047) SVM	267.5218(0.2928) SVM	267.6312(0.2789) SVM
	265.1799(1.9787) Fuzzy	267.2005(0.3532) Fuzzy	267.3044(0.0000) Fuzzy	267.6761(0.2800) Fuzzy
3.5	268.9826(4.0521) KNN	270.5687(1.9563) KNN	268.2041(0.2936) KNN	267.9547(0.2892) KNN
	269.7226(3.4842) D.T	268.8664(1.3311) D.T	268.2396(0.3258) D.T	267.9356(0.2868) D.T
	269.7736(4.0462) SVM	269.0753(1.1020) SVM	268.2776(0.2958) SVM	267.9775(0.2954) SVM
	269.4922(2.9089) Fuzzy	268.4562(1.0241) Fuzzy	267.7183(0.2822) Fuzzy	267.6762(0.2940) Fuzzy
5	266.4116(4.2227) KNN	266.2120(2.4856) KNN	268.5802(0.4253) KNN	267.798(0.2969) KNN
	266.4389(3.8596) D.T	265.7546(1.9098) D.T	268.4211(0.4587) D.T	267.7768(0.2948) D.T
	266.5263(4.2269) SVM	266.2009(2.1515) SVM	268.3874(0.2906) SVM	267.7768(0.3023) SVM
	268.1034(3.3664) Fuzzy	267.7777(1.3560) Fuzzy	267.9581(0.3396) Fuzzy	267.6743(0.2926) Fuzzy

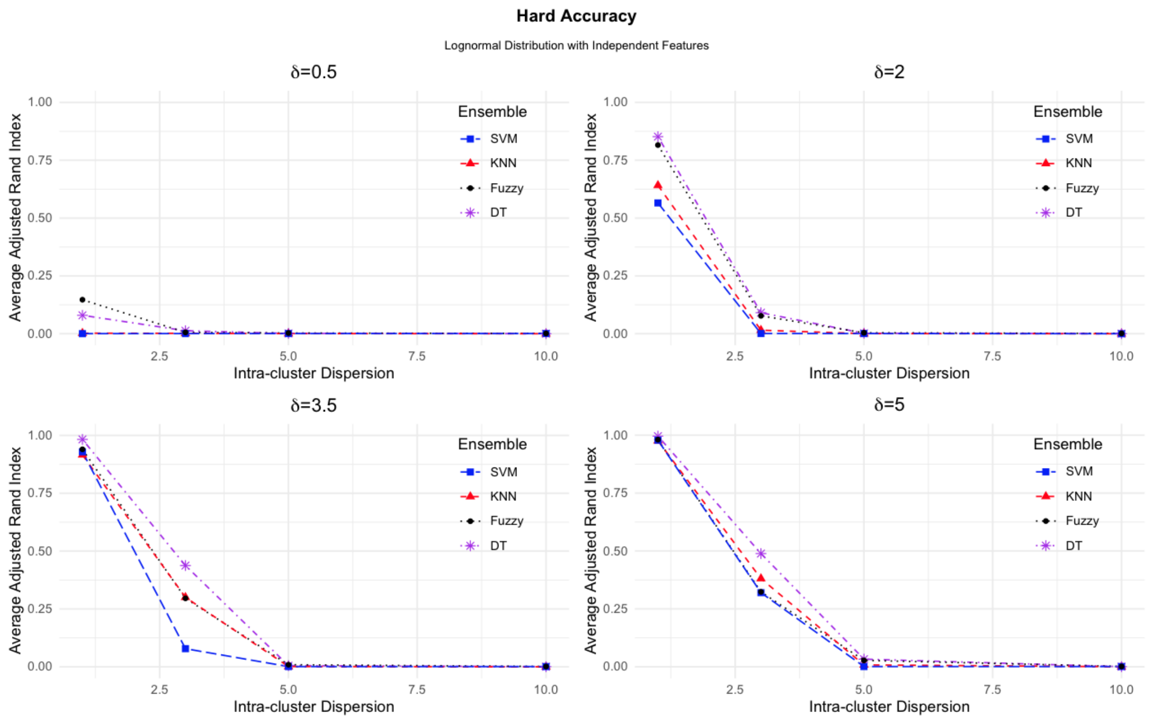


Figure 4.3 Hard accuracy of the clustering solutions produced by each clustering method when the data objects have arisen from a latent Lognormal process with independent features.

Similarly to Tables 4.4 and 4.6, Table 4.8 appears to suggest that the accuracy obtained by the soft clustering solutions are pretty similar regardless of the method used to create the clusters. A closer examination of Table 4.8 shows that irrespective of changes in δ and σ, the average absolute losses for the proposed ensembles and the Fuzzy C-Means algorithm, appear to remain consistent and marginally close to each other in each setting.

Tables 4.9 and 4.10 show the accuracy obtained when the data objects were simulated from a lognormal process with dependent features. Overall, compared to Table 4.7, the average accuracy is lower for all settings compared to the previously mentioned table. However, the same general trends remain present in this dependency case. We see in particular, that for a δ value of 2 or 3.5 when σ is 1 or 3, the accuracy obtained by the clustering solutions tends to be highest for the decision tree-based ensemble with an average ARI of 0.5667 (D.T) versus 0.4750 (Fuzzy) and 0.0517 (D.T) versus 0.0374 (Fuzzy) when $\delta = 2$ and $\sigma = 1$ and $\sigma = 3$, respectively. It is also worth mentioning that in the case of the most separation between cluster centers ($\delta = 5$) the KNN ensemble method also either matches or outperforms the Fuzzy C-Means algorithm for σ values of 1, 3, and 5. These results are shown graphically in Figure 4.4. It shows the same visual findings, from the independence case: that in cases of more defined clusters, the decision tree ensemble is better, whereas in the case of less defined clusters, the Fuzzy C-Means method appears to be better.

Alternatively, when looking at the soft clustering solutions produced by each clustering method, the results are not so definitive. In fact, a careful examination of Table 4.10 suggests similar findings to Table 4.8 in which either of the clustering methods (any proposed ensemble or the Fuzzy C-Means) appear to be viable options.

Together Tables 4.7-4.10 can be used to guide the decision of which clustering method to use when observations have arisen from a latent lognormal process. In particular, they suggest that if a person is interested in only the hard clustering solutions from clustering, then the decision tree ensemble should be used when the clusters are clearly defined regardless of whether features are independent or positively correlated. However, if the clusters are not clearly separated, but instead exhibit greater intra-cluster variability or less inter-cluster variability, then the Fuzzy C-Means algorithm should be used. If one is only interested in the soft solutions produced, then, as previously mentioned, any of the proposed ensembles or the Fuzzy C-Means algorithm is a feasible option. Finally, in the case of interest in both the hard and soft clustering solution, either the Fuzzy C-Means or the decision tree ensemble are viable options; however, attention should be shown to how well clusters are defined to make a final decision. Overall, in terms of the proposed ensembles, the findings suggest the best hard accuracy is given by the decision tree ensemble followed by the KNN ensemble in the lognormal case. Because the SVM ensemble did not perform as well as the other proposed ensembles in many settings, we would not recommend such a method of clustering in this case.

The lognormal simulation results are particularly enlightening when we consider one of the fields that Bezdek, Ehrlich, and Full (1984) cites as benefiting from soft clustering is geology. Interestingly, Limpert, Stahel, and Abbt (2001) provides some examples in which the lognormal distribution naturally arises—one of these is in geology. In particular, Limpert, Stahel, and Abbt (2001) mentions certain elements' concentrations may be described by a lognormal process. They also mention other fields, like medicine, in which the lognormal distribution can be used to describe the latency period in diseases like chicken pox. What the simulations suggest is that in these cases when clusters are clearly defined there may be value in using an ensemble such as the decision tree to cluster observations rather than the Fuzzy C-Means algorithm.

Table 4.9 Hard Accuracy for Lognormally Distributed Data with Dependent Features

Accuracy of Hard Clustering Solutions				
δ	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 10$
0.5	0.0320(0.0012) KNN	0.0014(0.0001) KNN	0.0005(0.0000) KNN	0.0001(0.0000) KNN
	0.0757(0.0013) D.T	0.0027(0.0001) D.T	0.0011(0.0000) D.T	0.0003(0.0000) D.T
	0.0057(0.0003) SVM	0.0011(0.0000) SVM	0.0009(0.0000) SVM	0.0010(0.0000) SVM
	0.0596(0.0008) Fuzzy	0.0039(0.0001) Fuzzy	0.0021(0.0001) Fuzzy	0.0010(0.0000) Fuzzy
2	0.4249(0.0033) KNN	0.0158(0.0014) KNN	0.0005(0.0000) KNN	0.0001(0.0000) KNN
	0.5667(0.0063) D.T	0.0527(0.0027) D.T	0.0013(0.0001) D.T	0.0003(0.0000) D.T
	0.4517(0.0037) SVM	0.0016(0.0001) SVM	0.0010(0.0000) SVM	0.0008(0.0001) SVM
	0.4750(0.0040) Fuzzy	0.0374(0.0010) Fuzzy	0.0026(0.0001) Fuzzy	0.0009(0.0000) Fuzzy
3.5	0.7606(0.0019) KNN	0.2260(0.0044) KNN	0.0010(0.0000) KNN	0.0001(0.0001) KNN
	0.9061(0.0018) D.T	0.3523(0.0055) D.T	0.0037(0.0005) D.T	0.0004(0.0000) D.T
	0.7876(0.0019) SVM	0.1029(0.0041) SVM	0.0010(0.0000) SVM	0.0008(0.0000) SVM
	0.8118(0.0014) Fuzzy	0.2664(0.0027) Fuzzy	0.0033(0.0001) Fuzzy	0.0010(0.0000) Fuzzy
5	0.9044(0.0011) KNN	0.3480(0.0025) KNN	0.0083(0.0013) KNN	0.0001(0.0000) KNN
	0.9601(0.0011) D.T	0.4470(0.0029) D.T	0.0223(0.0025) D.T	0.0004(0.0000) D.T
	0.8908(0.0014) SVM	0.2856(0.0043) SVM	0.0014(0.0028) SVM	0.0009(0.0000) SVM
	0.9044(0.0011) Fuzzy	0.3259(0.0016) Fuzzy	0.0070(0.0005) Fuzzy	0.0010(0.0000) Fuzzy

Table 4.10 Soft Accuracy for Lognormally Distributed Data with Dependent Features

Accuracy of Soft Clustering Solutions				
δ	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	$\sigma = 10$
0.5	275.0548(0.5255) KNN	268.3680(0.2903) KNN	267.8451(0.2865) KNN	267.6278(0.2830) KNN
	268.3654(0.7254) D.T	268.3043(0.2993) D.T	267.7884(0.2808) D.T	267.6539(0.2830) D.T
	268.3124(0.4512) SVM	268.2369(0.2925) SVM	267.8013(0.2907) SVM	267.7625(0.2900) SVM
	267.5993(0.4870) Fuzzy	268.1349(0.2638) Fuzzy	267.8493(0.2752) Fuzzy	267.4281(0.2781) Fuzzy
2	273.9079(2.4673) KNN	270.0237(0.5736) KNN	267.6039(0.3030) KNN	267.7922(0.2774) KNN
	268.0776(2.4517) D.T	268.8184(0.7346) D.T	267.7647(0.3197) D.T	267.8680(0.2773) D.T
	267.0087(2.7862) SVM	267.9307(0.3206) SVM	267.4927(0.3020) SVM	267.7362(0.2863) SVM
	268.6401(1.9382) Fuzzy	267.6608(0.4860) Fuzzy	268.2749(0.0000) Fuzzy	267.9354(0.2790) Fuzzy
3.5	269.5869(3.6856) KNN	268.5489(1.9807) KNN	267.8913(0.3035) KNN	268.1416(0.2864) KNN
	270.9175(3.4612) D.T	267.1997(1.7663) D.T	268.0017(0.3615) D.T	268.1433(0.2882) D.T
	271.3533(3.7155) SVM	266.4832(1.2468) SVM	267.7900(0.2856) SVM	268.1216(0.2900) SVM
	265.6199(3.0618) Fuzzy	267.6835(1.4011) Fuzzy	268.0481(0.2802) Fuzzy	267.9571(0.2922) Fuzzy
5	273.7498(4.0060) KNN	267.9921(2.6102) KNN	268.2870(0.4518) KNN	267.9704(0.2939) KNN
	274.0338(3.8134) D.T	267.5807(2.3215) D.T	268.5354(0.5246) D.T	267.8825(0.2958) D.T
	274.2680(4.0094) SVM	267.5354(2.2385) SVM	268.0964(0.2969) SVM	267.9569(0.2987) SVM
	263.0274(3.5565) Fuzzy	268.5479(1.8313) Fuzzy	267.5058(0.3303) Fuzzy	268.0177(0.2957) Fuzzy

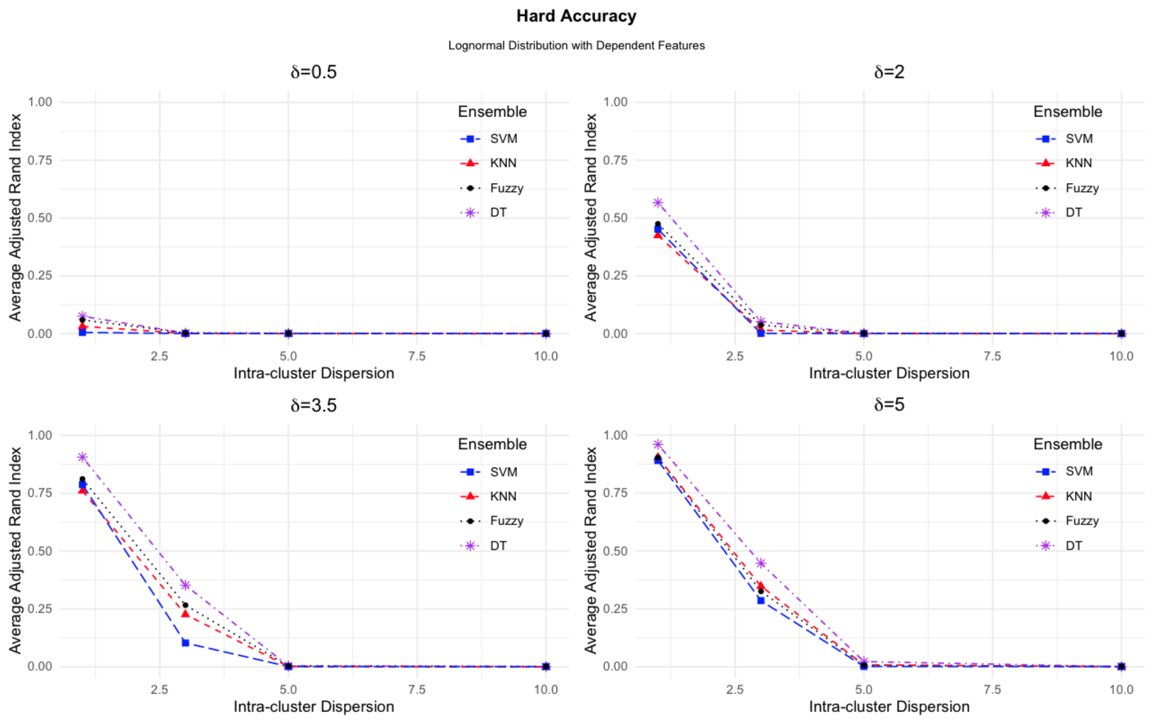


Figure 4.4 Hard accuracy of the clustering solutions produced by each clustering method when the data objects have arisen from a latent Lognormal process with dependent features.

4.3.4 Student's t Simulation Setup

For the simulations on data following the multivariate Student's t distribution done in this section, we assume there are n data objects to be clustered upon which we have observed P latent features arising from a continuous process. We also assume these observations have arisen from J clusters and consider the cases where the features are mutually independent and dependent. To generate the data, suppose $\boldsymbol{Y}_i^* \sim t_{P,\nu}(\mathbf{0}, \boldsymbol{\Sigma}^*)$, where ν refers to the degrees of freedom for the t distribution, $\mathbf{0}$ denotes a vector of zeroes, and $\boldsymbol{\Sigma}^*$ is a $P \times P$ matrix. We generate the ith object in cluster J as $\boldsymbol{Y}_i = \boldsymbol{Y}_i^* - \boldsymbol{\delta}$ for $i = 1, 2, \ldots, n_1$, $\boldsymbol{Y}_i = \boldsymbol{Y}_i^*$ for $i = 1, 2, \ldots, n_2$, and $\boldsymbol{Y}_i = \boldsymbol{Y}_i^* + \boldsymbol{\delta}$ for $i = 1, 2, \ldots, n_3$ for clusters $j = 1, 2, 3$, respectively. Once the data are generated, they are stored in a $n \times P$ matrix $\boldsymbol{Y}$ where the ith object's observations are stored in row i.

Parameter Setting

For the simulations in this section, we assume there are $n = 300$ objects each with $P = 10$ features that have arisen from $J = 3$ clusters. We set $n_1 = 100$, $n_2 = 100$, and $n_3 = 100$. In the independent feature case, we set $\boldsymbol{\Sigma}^*$ to be the identity matrix. In the dependent case, $\boldsymbol{\Sigma}^*$ is a $P \times P$ matrix with 1's along the diagonal and $\frac{1}{10}$ on the off-diagonals. The resulting covariance matrix for each case is given as:

$$\boldsymbol{\Sigma} = \frac{\nu}{\nu - 2}\boldsymbol{\Sigma}^*.$$

We introduce variation within the clustering process using parameters δ and ν where δ represents the distance between cluster centers and ν represents the degrees of freedom used for the specified t distribution, but is used here to measure the within-cluster variability. (Note, the within-cluster variability is given specifically by $\frac{\nu}{\nu-2}$). For the simulations done in this paper, we consider $\delta \in \{0.5, 2.0, 3.5, 5.0\}$ and $\nu \in \{3, 4, 5, 6\}$. As the value of δ or ν increases (while holding the other constant), we expect the clustering process to become easier.

We generate the features as described above and store the ith object's measurements in the ith row of matrix $\boldsymbol{Y}$. Next, 40% of the observations are randomly selected from $\boldsymbol{Y}$ (with replacement) and used as the training set. The remaining observations are then used as the test set. At this point, the algorithm outlined in Section 4.2.1 is employed. For the results presented in this section, $B = 200$ clustering solutions were produced in the generation step. The tuning parameters used for each algorithm (obtained after 1 iteration of the Fuzzy C-Means labelling procedure) are as follows: For the kNN, 4 neighbors were used. For the decision tree, a cost pruning value of 0.0001 was used. Lastly, for the SVM algorithm, $\gamma = 0.0001$ and a cost of 1000 is used. These values were held constant for all the simulations used in this section. The results are shown in Section 4.3.4.

Results

Tables 4.11 and 4.12 show the clustering accuracy obtained by each clustering method when the data has arisen from a t distribution with independent features. Overall, the best hard accuracy was obtained in this setting by the Fuzzy C-Means algorithm; however, of the proposed ensemble methods, the best hard accuracy was obtained using the SVM ensemble. In Table 4.11 we notice specifically when $\delta = 0.5$ that the performance of the Fuzzy C-Means algorithm is substantially higher than that of the proposed algorithms with an average ARI in each case near 0.3000; whereas, for the ensemble algorithms, the highest ARI is obtained by the SVM algorithm when $\nu = 6$ at 0.0522. When the clusters become more defined, e.g., when $\delta = 3.5$ or $\delta = 5$, we see the performance of the proposed algorithms get closer to that of the Fuzzy C-Means algorithm, with the SVM and KNN ensemble algorithms only performing marginally worse than Fuzzy C-Means algorithm in these cases. For example when $\delta = 5$ and $\nu = 5$, the average ARI values are 0.9988, 0.9986, and 0.9989, for the ensemble based on the KNN, SVM, and the Fuzzy C-Means algorithm, respectively. Overall these simulations suggest that when there is more overlap between clusters and the features are independent, the Fuzzy C-Means algorithm may be the better method to use. Whereas when there is more separation between clusters (lower right-hand side of Table 4.11), the accuracy of the clusters formed using the KNN and SVM ensembles are only marginally below that of the Fuzzy C-Means algorithms. These results can be seen visually in Figure 4.5

Table 4.11 Hard Accuracy for Student's t-Distributed Data with Independent Features

Accuracy of Hard Clustering Solutions				
δ	$\nu = 3$	$\nu = 4$	$\nu = 5$	$\nu = 6$
0.5	0.0006(0.0003) KNN	0.0004(0.0001) KNN	0.0002(0.0001) KNN	0.0002(0.0001) KNN
	0.0424(0.0013) D.T	0.0445(0.0014) D.T	0.0448(0.0014) D.T	0.0489(0.0015) D.T
	0.0362(0.0017) SVM	0.0426(0.0018) SVM	0.0537(0.0020) SVM	0.0522(0.0018) SVM
	0.2693(0.0013) Fuzzy	0.2883(0.0013) Fuzzy	0.2996(0.0013) Fuzzy	0.3046(0.0012) Fuzzy
2	0.7202(0.0037) KNN	0.7236(0.0039) KNN	0.7207(0.0040) KNN	0.7152(0.0040) KNN
	0.8065(0.0021) D.T	0.8475(0.0018) D.T	0.8681(0.0018) D.T	0.8857(0.0017) D.T
	0.8703(0.0018) SVM	0.9088(0.0012) SVM	0.9322(0.0011) SVM	0.9458(0.0009) SVM
	0.8984(0.0013) Fuzzy	0.9310(0.0009) Fuzzy	0.9504(0.0009) Fuzzy	0.9605(0.0008) Fuzzy
3.5	0.9689(0.0015) KNN	0.9871(0.0006) KNN	0.9930(0.0003) KNN	0.9961(0.0002) KNN
	0.9081(0.0019) D.T	0.9336(0.0013) D.T	0.9454(0.0012) D.T	0.9550(0.0012) D.T
	0.9675(0.0016) SVM	0.9870(0.0006) SVM	0.9931(0.0003) SVM	0.9965(0.0002) SVM
	0.9762(0.0008) Fuzzy	0.9894(0.0004) Fuzzy	0.9946(0.0003) Fuzzy	0.9973(0.0002) Fuzzy
5	0.9893(0.0008) KNN	0.9969(0.0002) KNN	0.9988(0.0002) KNN	0.9995(0.0001) KNN
	0.9633(0.0010) D.T	0.9811(0.0006) D.T	0.9875(0.0005) D.T	0.9916(0.0004) D.T
	0.9889(0.0008) SVM	0.9966(0.0002) SVM	0.9986(0.0001) SVM	0.9994(0.0001) SVM
	0.9903(0.0007) Fuzzy	0.9971(0.0002) Fuzzy	0.9989(0.0001) Fuzzy	0.9996(0.0001) Fuzzy

Table 4.12 Soft Accuracy for t-Distributed Data with Independent Features

Accuracy of Soft Clustering Solutions				
δ	$\nu = 3$	$\nu = 4$	$\nu = 5$	$\nu = 6$
0.5	270.1162(0.2408) KNN	270.4716(0.2363) KNN	270.6972(0.2480) KNN	270.6678(0.2391) KNN
	267.7484(0.2635) D.T	267.9312(0.2591) D.T	267.7405(0.2772) D.T	268.1239(0.2728) D.T
	267.9494(0.3836) SVM	267.9133(0.3789) SVM	267.7564(0.4144) SVM	268.0741(0.3889) SVM
	267.5910(0.1480) Fuzzy	267.9640(0.1416) Fuzzy	267.8599(0.1592) Fuzzy	268.1039(0.1401) Fuzzy
2	261.3194(3.3090) KNN	266.5249(3.1473) KNN	271.4824(3.1434) KNN	267.2241(3.1650) KNN
	261.8182(2.7670) D.T	265.4955(2.7195) D.T	270.4135(2.7650) D.T	266.6034(2.8163) D.T
	259.7904(3.8122) SVM	265.5945(3.7122) SVM	271.6628(3.7899) SVM	266.7188(3.8640) SVM
	263.1706(2.4872) Fuzzy	267.2594(2.4445) Fuzzy	270.5907(2.4353) Fuzzy	266.2824(2.5799) Fuzzy
3.5	262.9585(4.1158) KNN	264.4337(4.0783) KNN	271.3773(3.9023) KNN	270.6418(3.8637) KNN
	263.7311(3.5323) D.T	265.3380(3.5771) D.T	271.3899(3.4953) D.T	270.8368(3.5005) D.T
	263.0842(4.2735) SVM	264.9276(4.2722) SVM	272.2220(4.1211) SVM	271.5132(4.0982) SVM
	268.2577(3.2338) Fuzzy	264.0256(1.4011) Fuzzy	270.9344(3.3491) Fuzzy	266.8103(3.4433) Fuzzy
5	265.6139(4.1677) KNN	275.7006(3.9578) KNN	265.0200(4.2411) KNN	272.1589(4.2236) KNN
	266.0299(3.7996) D.T	274.9938(3.6868) D.T	265.4488(4.1120) D.T	272.0647(4.0095) D.T
	265.7223(4.2187) SVM	274.9938(3.6868) SVM	265.1907(4.3115) SVM	272.6009(4.3018) SVM
	269.3869(3.6184) Fuzzy	272.3676(3.6752) Fuzzy	269.9758(3.7887) Fuzzy	274.4975(3.6996) Fuzzy

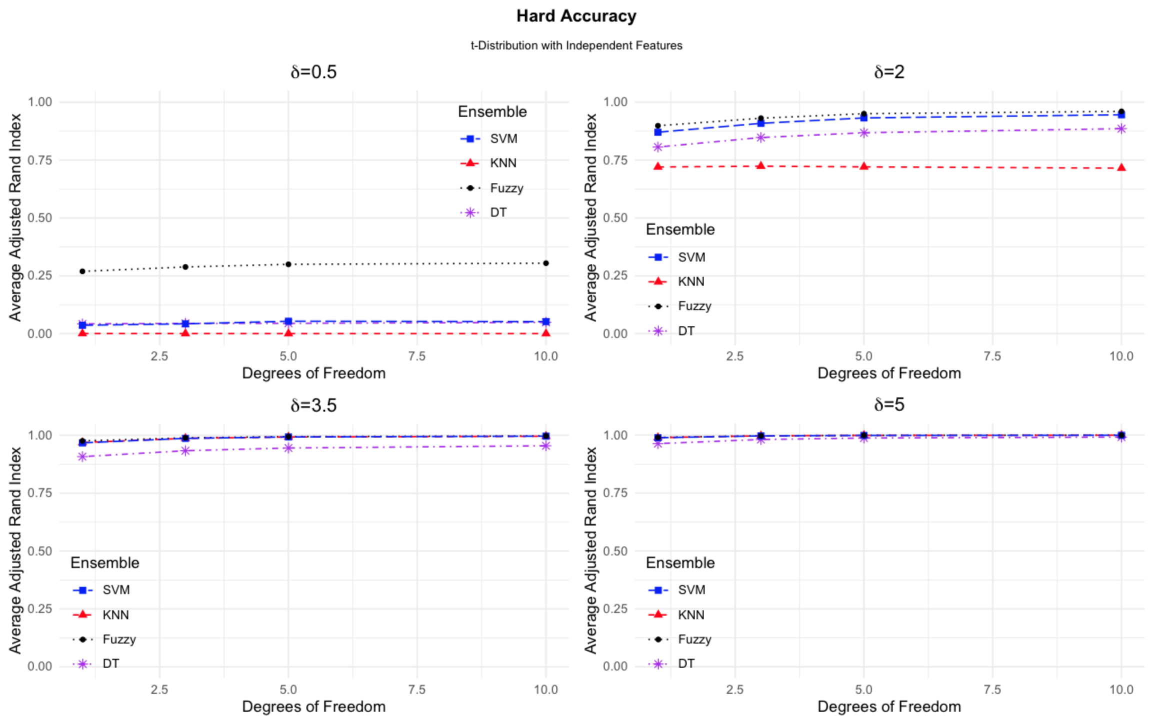


Figure 4.5 Hard accuracy of the clustering solutions produced by each clustering method when the data objects have arisen from a t-distribution with independence in the features.

Tables 4.13 and 4.14 show the clustering accuracy obtained when assuming the data objects have arisen from a t distributions with dependent features. Overall the average ARI values are lower than seen previously in Table 4.11. The soft clustering accuracies, however, remain relatively the same as that seen in Table 4.12. A closer examination of Table 4.13 suggests the highest accuracy is obtained in most settings still with the Fuzzy C-Means algorithm; however, as opposed to the independence case, the performance of the SVM ensemble method is much closer to the performance of the Fuzzy C-Means algorithm in most settings than before. It is also worth noting that when $\delta = 0.5$ in the dependent case, the performance obtained by the proposed ensembles are higher than what was observed in Table 4.11 for the same δ value even though they still are notably worse-performing than the Fuzzy C-Means algorithm. The results in the hard accuracy can be seen graphically in Figure 4.6.

When looking at the soft clustering accuracy, there is not as clear of a pattern in the results. As we observed in the normal and lognormal cases, these indicate that the clustering accuracy remains pretty consistent amongst all the algorithms regardless of the separation between clusters and the within cluster dispersion.

Overall, the accuracy findings in this setting mimic those in the normal case. This should not be surprising as both distributions exhibit a symmetric shape. Combining these results with those obtained from the lognormal setting, it appears the ensemble based methods of clustering have the potential to be more influential in cases where the observations have arisen from a highly skewed distribution. Our simulation results suggest in particular that for observations arising from a highly skewed distribution, it may possible for an ensemble based method of clustering to create more accurate clusters than the Fuzzy C-Means algorithm. Consequently, future research endeavors should look at ways to make these ensemble-based methods more efficient, and any users should consider strongly both the shape of the distribution from which clusters have arisen and whether they are expected to be well-defined before implementing either method.

Table 4.13 Hard Accuracy for Student's t-Distributed Data with Dependent Features

Accuracy of Hard Clustering Solutions				
δ	$\nu = 3$	$\nu = 4$	$\nu = 5$	$\nu = 6$
0.5	0.0344(0.0012) KNN	0.0328(0.0011) KNN	0.0341(0.0011) KNN	0.0322(0.0011) KNN
	0.0868(0.0011) D.T	0.0924(0.0011) D.T	0.0961(0.0011) D.T	0.0996(0.0011) D.T
	0.0845(0.0013) SVM	0.0925(0.0012) SVM	0.1002(0.0013) SVM	0.1046(0.0012) SVM
	0.1135(0.0010) Fuzzy	0.1182(0.0010) Fuzzy	0.1242(0.0010) Fuzzy	0.1277(0.0010) Fuzzy
2	0.4607(0.0025) KNN	0.4746(0.0018) KNN	0.4777(0.0027) KNN	0.4858(0.0028) KNN
	0.5902(0.0020) D.T	0.6294(0.0018) D.T	0.6510(0.0019) D.T	0.6670(0.0019) D.T
	0.6067(0.0020) SVM	0.6493(0.0018) SVM	0.6704(0.0018) SVM	0.6892(0.0018) SVM
	0.6368(0.0017) Fuzzy	0.6733(0.0017) Fuzzy	0.6957(0.0017) Fuzzy	0.7137(0.0017) Fuzzy
3.5	0.8512(0.0015) KNN	0.8851(0.0014) KNN	0.9061(0.0012) KNN	0.9150(0.0012) KNN
	0.8365(0.0015) D.T	0.8750(0.0014) D.T	0.9223(0.0010) D.T	0.9087(0.0011) D.T
	0.8620(0.0015) SVM	0.8992(0.0012) SVM	0.9223(0.0010) SVM	0.9344(0.0010) SVM
	0.8709(0.0014) Fuzzy	0.9051(0.0011) Fuzzy	0.9279(0.0010) Fuzzy	0.9389(0.0009) Fuzzy
5	0.9413(0.0011) KNN	0.9668(0.0007) KNN	0.9771(0.0006) KNN	0.9851(0.0001) KNN
	0.9286(0.0012) D.T	0.9557(0.0081) D.T	0.9663(0.0007) D.T	0.9740(0.0006) D.T
	0.9423(0.0011) SVM	0.9676(0.0007) SVM	0.9782(0.0006) SVM	0.9856(0.0005) SVM
	0.9452(0.0010) Fuzzy	0.9695(0.0006) Fuzzy	0.9790(0.0006) Fuzzy	0.9866(0.0005) Fuzzy

Table 4.14 Soft Accuracy for t-Distributed Data with Dependent Features

Accuracy of Soft Clustering Solutions				
δ	$\nu = 3$	$\nu = 4$	$\nu = 5$	$\nu = 6$
0.5	270.0434(0.5906) KNN	270.8531(0.5933) KNN	270.5916(0.5897) KNN	269.4788(0.6588) KNN
	267.2866(0.6284) D.T	268.0222(0.6454) D.T	268.4668(0.6659) D.T	268.1954(0.6871) D.T
	266.6877(0.8676) SVM	268.2597(0.9230) SVM	267.7139(0.9651) SVM	268.1532(0.9826) SVM
	269.2722(0.6413) Fuzzy	267.9887(0.6852) Fuzzy	267.8249(0.6465) Fuzzy	269.4788(0.6588) Fuzzy
2	266.0053(2.5979) KNN	267.2211(2.6225) KNN	267.0818(2.6519) KNN	264.3354(2.6410) KNN
	266.0506(2.5515) D.T	267.2211(2.6255) D.T	266.8566(2.7248) D.T	263.8852(2.7178) D.T
	265.8258(3.0820) SVM	267.4185(3.1728) SVM	266.2695(3.2938) SVM	263.1466(3.2876) SVM
	265.2705(2.3756) Fuzzy	270.3195(2.3938) Fuzzy	270.9315(2.3795) Fuzzy	268.4948(2.4734) Fuzzy
3.5	259.8139(3.8967) KNN	264.9391(3.7490) KNN	260.6363(3.9082) KNN	266.2785(3.7859) KNN
	261.3488(3.5761) D.T	265.8829(3.4817) D.T	262.0293(3.6729) D.T	266.7903(3.5771) D.T
	260.2613(4.0993) SVM	265.1280(3.9701) SVM	261.0914(4.1765) SVM	266.8156(4.0644) SVM
	264.1869(3.2439) Fuzzy	270.4869(3.1752) Fuzzy	266.8554(3.3846) Fuzzy	267.5221(3.4313) Fuzzy
5	259.6949(4.2280) KNN	271.0366(3.9091) KNN	267.4828(4.1188) KNN	266.1241(4.0835) KNN
	260.5524(3.9492) D.T	270.8284(3.6957) D.T	267.5334(3.9286) D.T	266.3379(3.9127) D.T
	259.7936(4.2922) SVM	271.2930(3.9834) SVM	267.5940(4.2032) SVM	266.2053(4.1722) SVM
	271.0047(3.4870) Fuzzy	266.5686(3.7836) Fuzzy	269.1636(3.7793) Fuzzy	266.7475(3.6607) Fuzzy

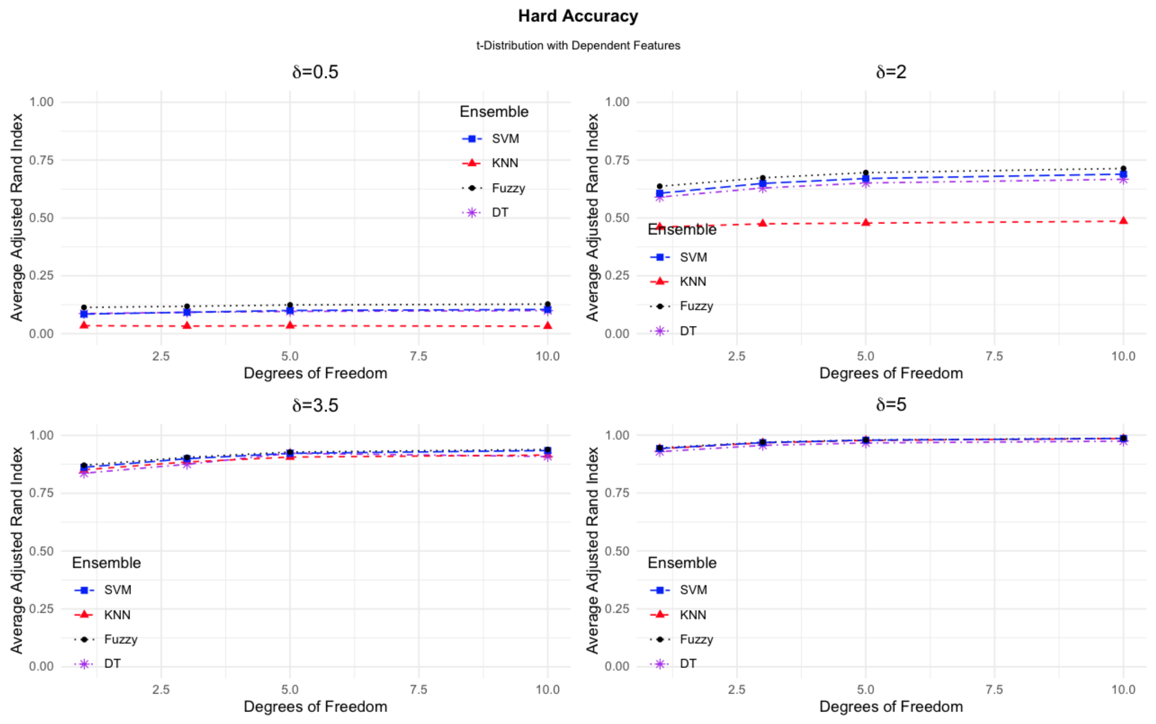


Figure 4.6 Hard accuracy of the clustering solutions produced by each clustering method when the data objects have arisen from a t-distribution with dependency in the features.

4.4 Data Application

In this section, we apply each of the proposed ensembles of Section 4.2.1 to a wine dataset and a forensic glass identification dataset obtained from the UCI Machine Learning Repository (Dua and Graff, 2019). The two datasets are used to access the applicability of the proposed algorithms on a dataset with clearly defined clusters (wine) and a more noisy dataset (glass) as can be seen in Figures 4.7 and 4.8, respectively. Since the simulation results of Section 4.3 suggested our proposed methods of clustering perform better than Fuzzy C-Means clustering in particular cases in which the data has arisen from a skewed distribution, we believe the methods are more applicable to data arising from an underlying skewed distribution. Figure 4.9 shows

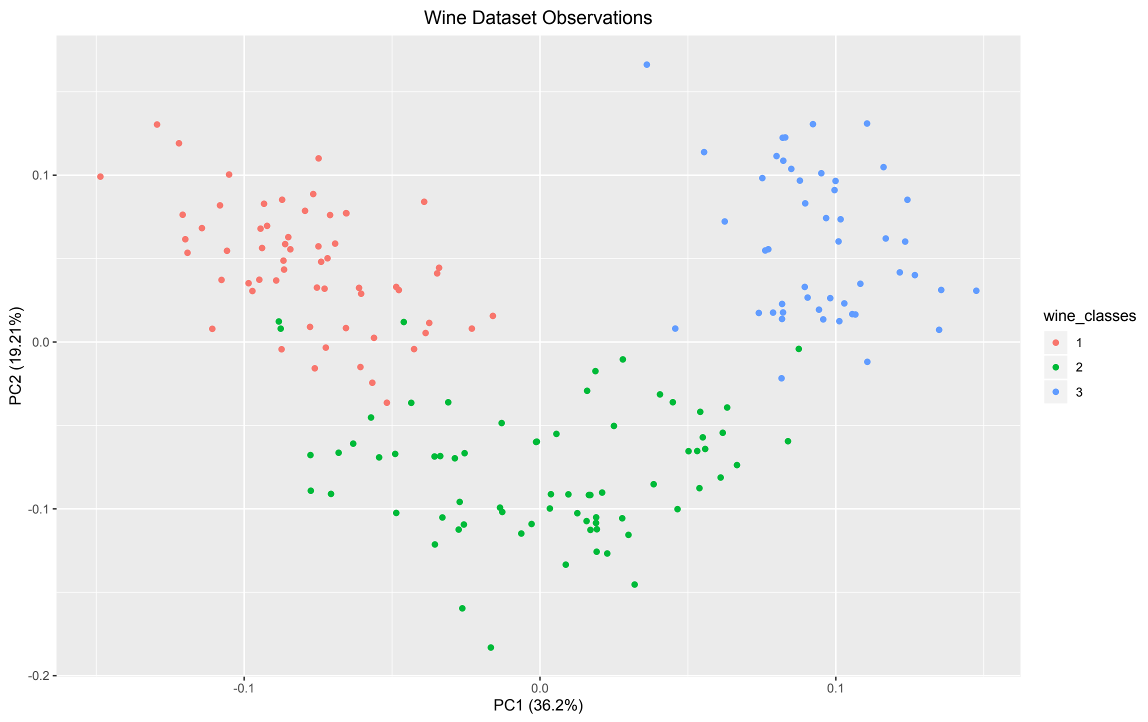


Figure 4.7 The graph above shows the wine observations plotted in the space of the first two principal components. The different colors represent the true classification for each observation in the dataset.

Chi-Square plots for the wine and glass datasets. The graphs indicate the distributions both depart from normality and instead suggest the datasets are both skewed in at least some of the dimensions. Because of this clear departure, we believe the datasets are suitable for use with our proposed alogrithms. In the following sections, we begin with a description of each dataset and conclude with a discussion of the clustering results.

4.4.1 Wine Dataset

The wine dataset consists of $n = 178$ wines each with $P = 13$ attributes from 3 locations in Italy. The attributes record the following in each wine: alcohol, malic acid, ash, alkalinity, magnesium, phenols, flavanoids, nonflavanoids, proanthocyanins,

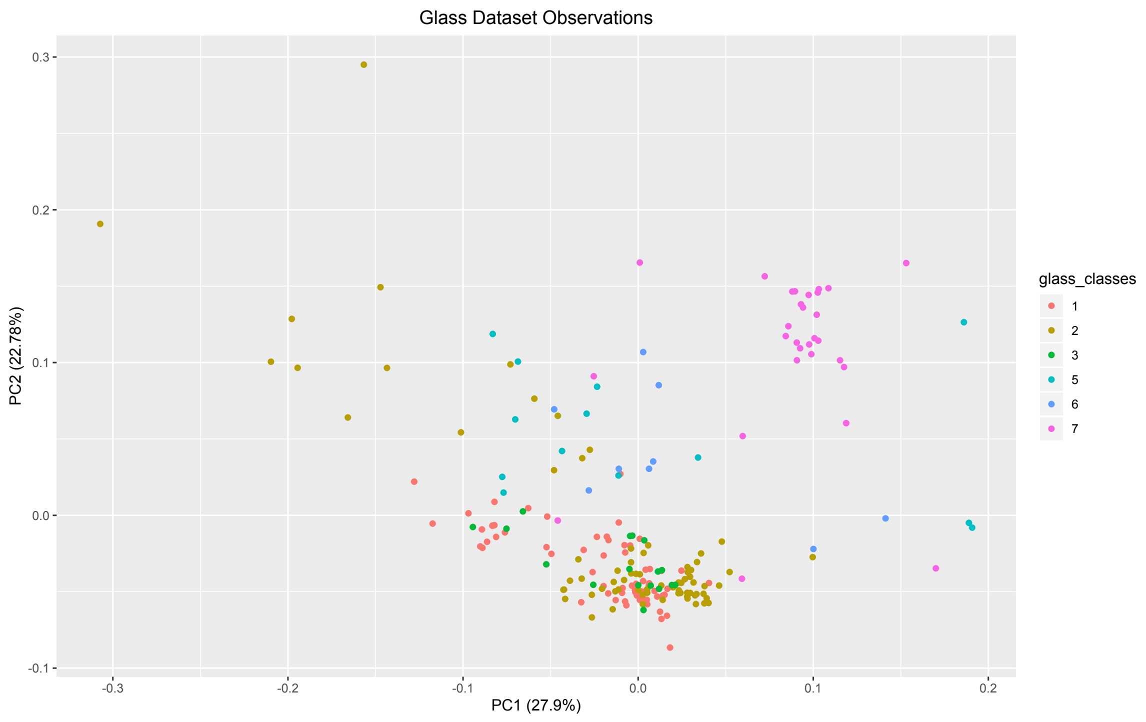


Figure 4.8 The graph above shows the glass observations plotted in the space of the first two principal components. The different colors represent the true classification for each observation in the dataset.

color, hue, dilution, and proline. Because each attribute is measured in different scales, the dataset was first scaled. (Note: Scaling here refers to standardization of each variable done by transforming each variable to have a mean of zero and unit variance). Next, the tuning parameters for the ensembles were found. For the ensembles, cross-validation was used to obtain the tuning parameters after 1 iteration of the Fuzzy C-Means labelling process. The resulting parameter values include 6 neighbors for the k-nearest neighbor ensemble, a cost pruning value of 0.01 for the decision tree ensemble, and a $\gamma = 0.1$ with a cost of 1 for the SVM ensemble. Next, 40% of observations were randomly selected with replacement from the original dataset and the procedure in Section 4.2.1 was applied with $B = 300$ clustering solutions. The results are discussed in the next section.

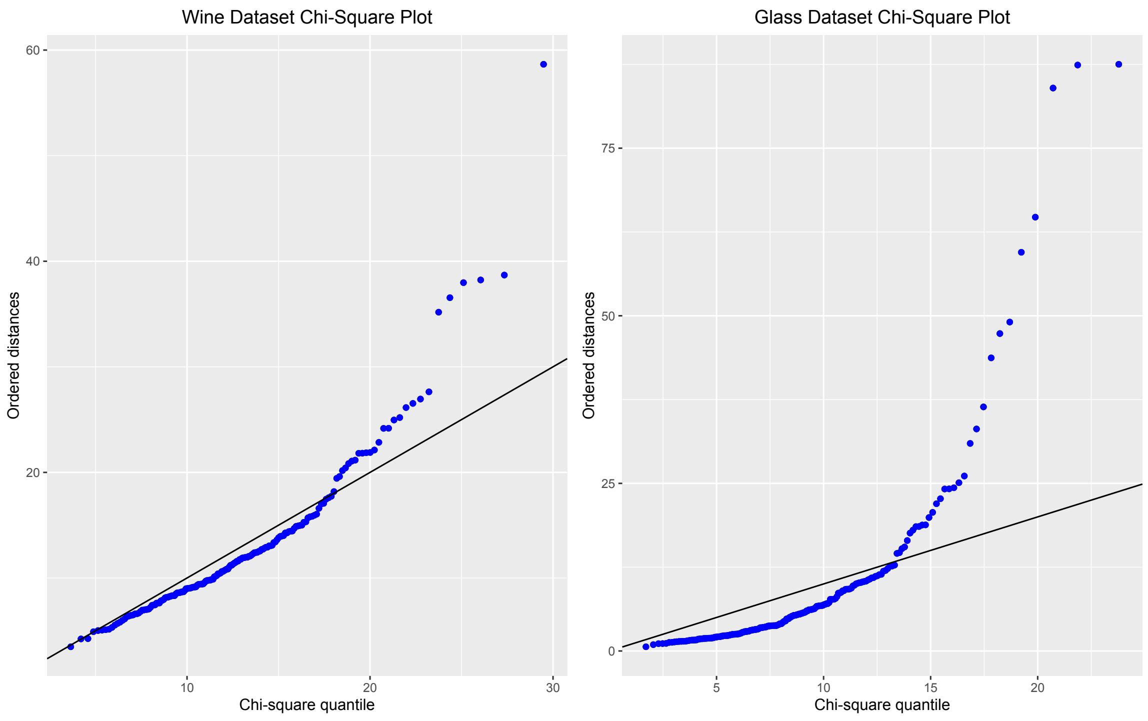


Figure 4.9 The Chi-Square plots above provide a method through which to check for normality of the wine (left) and glass (right) data. Both plots indicate a departure from normality and suggest at least some of the dimensions may be skewed.

Table 4.15 Confusion Matrix Formed from k-Nearest Neighbor-Based Ensemble Clustering of Wine Dataset

	KNN		
Truth	1	2	3
1	3	27	11
2	8	40	0
3	30	0	0

Table 4.16 Confusion Matrix Formed from Decision Tree-Based Ensemble Clustering of Wine Dataset

Decision Tree			
Truth	1	2	3
1	0	0	41
2	1	46	1
3	30	0	0

Table 4.17 Confusion Matrix Formed from Support Vector Machine-Based Ensemble Clustering of Wine Dataset

Support Vector Machine			
Truth	1	2	3
1	0	0	41
2	0	47	1
3	30	0	0

Table 4.18 Confusion Matrix Formed from Fuzzy C-Means Clustering of Wine Dataset

Fuzzy C-Means			
Truth	1	2	3
1	41	0	0
2	1	2	45
3	0	30	0

Results

Figure 4.7 suggests $J = 3$ clusters and each of the clustering algorithms also produce 3 clusters. Tables 4.15-4.18 show the cross-tabulations of the true classifications and the clustering results obtained by each of the clustering algorithms (also called the confusion matrices). Table 4.19 shows the hard and soft accuracy that corresponds to each of these algorithms. Based on these results, the most accurate hard solutions are obtained by the SVM-based ensemble algorithm with an ARI of 0.9720.

The decision tree ensemble method and the Fuzzy C-Means algorithms, too, obtain a high accuracy with ARIs of 0.9478 and 0.9241, respectively. When looking at the soft clustering accuracy, the highest accuracy is obtained by the k-nearest neighbor ensemble at around 148. The next best soft accuracy is given by the Fuzzy C-Means algorithm.

To better visualize the clusters, each of the resulting solutions are plotted in the space of the first two principal components in Figure 4.10. Each of these plots also contain frames to allow for quicker cluster identification and to help visualize the differences between each method. Note that cluster 1 in the proposed ensemble algorithms appears to correspond to cluster 2 in the Fuzzy C-Means algorithm. This is not an issue of concern as clustering labels are completely arbitrary and do not represent true labels. What is worth noting is that in the middle-left of each plot in Figure 4.10, the ensemble-based methods all show some degree of overlap between two clusters while the Fuzzy C-Means algorithm shows clear separation between the clusters. Revisiting Figure 4.7 we note there is some overlap in the groups when the principal components plot is coded based on the true class labels. More specifically, the principal components plots displaying the clusterings from the SVM and decision tree-based algorithms suggest that there may be some overlap between two of the three clusters, while the Fuzzy C-Means results suggest a lack of overlap. This may suggest an erroneous finding in the Fuzzy C-Means algorithm.

Table 4.19 Accuracy of Wine Clustering Solutions

	Accuracy	
Algorithm	Hard	Soft
KNN	0.3584	147.9067
DT	0.9478	163.0200
SVM	0.9720	163.8533
Fuzzy	0.9241	150.6017

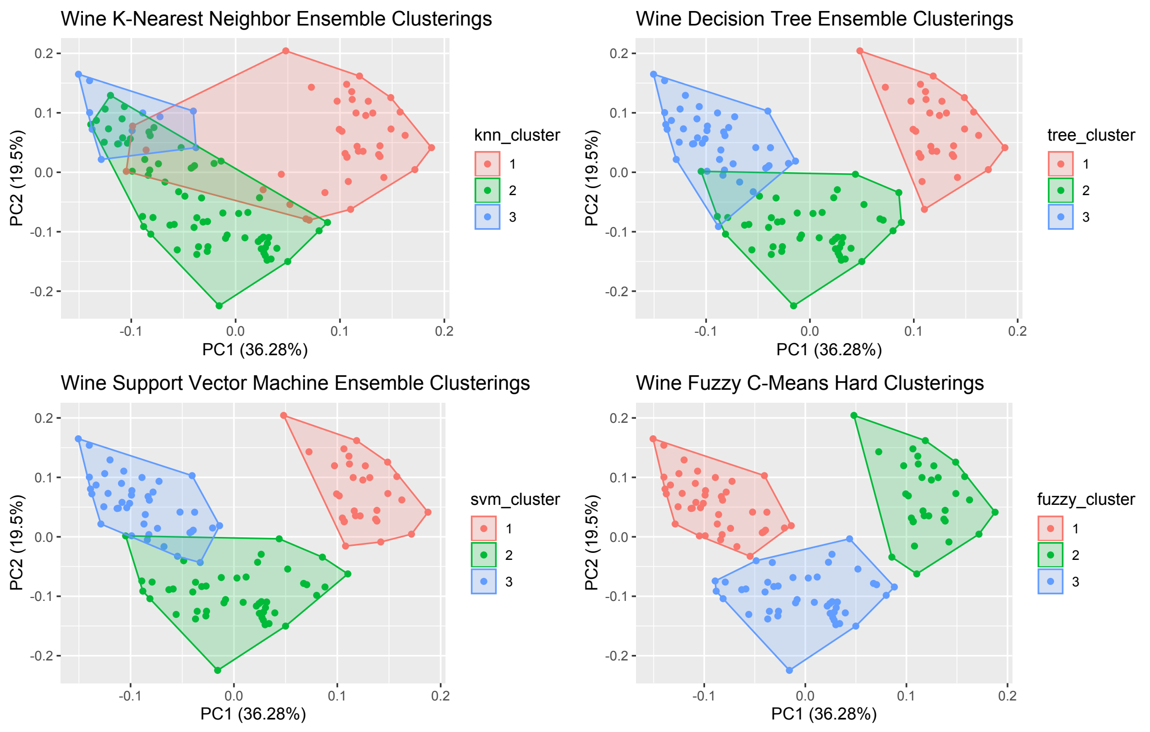


Figure 4.10 Wine clusters plotted in the first two principal components created using the (left to right and top to bottom) k-Nearest Neighbor ensemble, decision tree ensemble, support vector machine ensemble, and the Fuzzy C-Means algorithm.

4.4.2 Glass Identification Dataset

The glass identification dataset consists of $n = 214$ glass samples with $P = 9$ attributes from 7 classes of glass (Note: Only 6 classes are actually represented in the dataset). The classes of glass found in the dataset are: float-processed building window glass, non-float-processed building window glass, float-processed vehicle glass, container glass, tableware glass, and headlamp glass. The attributes recorded upon them include: refractive index and the weight percentage in oxides of sodium, magnesium, aluminum, silicon, potassium, calcium, barium, and iron. We use these attributes to cluster the glass and begin with finding the tuning values for each of the proposed ensemble algorithms after one iteration of the Fuzzy C-Means labelling procedure. The resulting tuning values include 8 neighbors for the k-nearest neighbor ensemble, a cost pruning value of 0.01 for the decision tree ensemble, a $\gamma = 0.1$ and a cost of 1 for the support vector machine ensemble. Next, 40% of the observations were chosen with replacement from the original glass dataset and used as a training set. Then the procedure as outlined in Section 4.2.1 was employed with $B = 300$ clustering solutions. The results obtained from each clustering algorithm are next discussed.

Table 4.20 Confusion Matrix Formed from k-Nearest Neighbor-Based Ensemble Clustering of Glass Dataset

	KNN	
Truth	**1**	**2**
1	0	47
2	5	42
3	0	12
5	4	5
6	2	3
7	13	8

Results

We note a major difference in the clustering results of the glass dataset—each clustering algorithm results in a different number of clusters. This is not surprising since Figure 4.8 displays overlap between observations in different classes, at least in the first two principal components. The k-nearest neighbor ensemble resulted in 2 clusters. The decision tree-based ensemble resulted in 4 clusters. The SVM-based algorithm resulted in 5 clusters, and the Fuzzy C-Means algorithm resulted in 6. When looking at the true nature of the dataset, what appears to be happening is that different classes of glass compose a single cluster. For example, in the k-nearest-neighbor-based method of clustering, the majority of the building glass (both float and non-float processed) and the float-processed vehicle glass are placed in the second cluster (see Table 4.20 where Truth=$1, 2,$ and 3 denotes float-processed building glass, non-float-processed building glass, and vehicle glass, respectively); however, this represents three classes in the original dataset. When we examine the decision tree-based ensemble results, we note more segmentation between each class of glass. For example, the majority of the headlamp glass (Truth=7) is represented in cluster 6 from the decision tree-based algorithm, while float-processed window glass (Truth=1) has been split among clusters 2, 5, and 6 (see Table 4.21). Similar results can be found from the SVM-based ensemble and the Fuzzy C-Means algorithm.

Tables 4.20-4.23 show the confusion matrices obtained from each clustering algorithm. (We note the absence of a 4 label under the truth column. This is due to the absence of non-float-processed vehicle glass within the original dataset). Despite the differences in clustering results, we see in Table 4.24 that the accuracy of the hard clustering solutions do not differ much between each method. We note in particular here that the ARI for the decision tree-based ensemble, SVM ensemble, and the Fuzzy C-Means algorithm all are around 0.25. A lower ARI value is not unanticipated as this dataset exhibits more variability compared to that of the wine dataset previously presented. However, the soft clustering solutions have more variation in accuracy, with the SVM-based algorithm producing the lowest absolute loss. Figure

Table 4.21 Confusion Matrix Formed from Decision Tree-Based Ensemble Clustering of Glass Dataset

Decision Tree				
Truth	**1**	**2**	5	**6**
1	0	30	17	0
2	1	39	3	4
3	0	9	3	0
5	0	3	1	5
6	0	0	2	3
7	0	1	1	19

4.11 and Table 4.24 raise an interesting question regarding the number of clusters present in this dataset. The algorithms suggest that there are anywhere between 2 and 6 clusters in the data (and such distinct solutions are similar in accuracy). Perhaps methods like these could be used in a way not previously explored, to help determine an initial number of clusters within a dataset when a method producing a hard solution is employed.

Table 4.22 Confusion Matrix Formed from Support Vector Machine-Based Ensemble Clustering of Glass Dataset

Support Vector Machine					
Truth	**1**	**2**	**4**	**5**	**6**
1	0	33	0	14	0
2	5	41	0	1	0
3	0	11	0	1	0
5	6	2	1	0	0
6	1	2	0	2	0
7	0	1	1	1	18

Table 4.23 Confusion Matrix Formed from Fuzzy C-Means Clustering of Glass Dataset

Fuzzy C-Means						
Truth	**1**	**2**	**3**	**4**	**5**	**6**
1	0	0	17	8	22	0
2	0	3	2	25	15	2
3	0	0	1	9	2	0
5	1	7	0	1	0	0
6	1	2	2	0	0	0
7	18	1	1	1	0	0

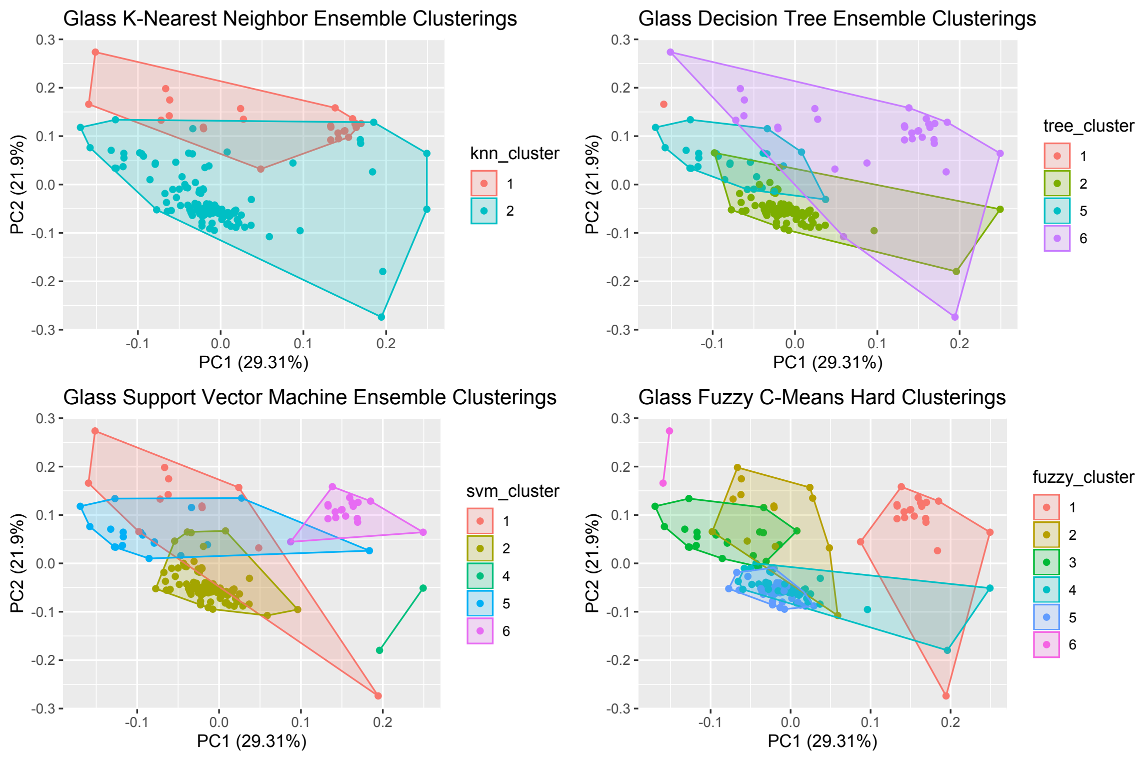


Figure 4.11 Glass clusters plotted in the first two principal components created using the (left to right and top to bottom) k-Nearest Neighbor ensemble, decision tree ensemble, support vector machine ensemble, and the Fuzzy C-Means algorithm.

Table 4.24 Accuracy of Glass Clustering Solutions

	Accuracy	
Algorithm	**Hard**	**Soft**
KNN	0.2444	195.5867
DT	0.2554	179.3067
SVM	0.2528	166.6333
Fuzzy	0.2513	265.5141

4.5 Discussion

The main goal of this paper was to introduce 3 new fuzzy ensemble algorithms for clustering that utilized the information from the Fuzzy C-Means membership matrix in a statistical manner and combined this information with knowledge of ensemble supervised learners. A secondary goal included learning about which supervised learners showed the most promise in this pursuit and to determine in what settings each learner-based ensemble algorithm was most applicable. In Section 4.1 we discussed the background of each learner used in the paper. In Section 4.2 we provided the outline for each ensemble, the rationale behind their development, and explained how they fit into the framework of an ensemble clustering algorithm. In Section 4.3 a detailed simulation study was conducted to assess the properties of each proposed ensemble and to learn in what scenarios their performances were best. Lastly, in Section 4.4, the ensembles were applied to two different datasets to assess each ensemble's performance on well-defined and less-defined clustering problems.

Simulation results suggest the Fuzzy C-Means algorithm is a better method of clustering when the data are believed to follow a Gaussian or t distribution regardless of intra-cluster dispersion, inter-cluster variability, or presence or lack of dependence among attributes. Furthemore, the simulation results also suggest that if an ensemble method is to be used in the normal or t case, that it should be one based upon the support vector machine learner. However, in cases in which the data has arisen from a latent lognormal process, the ensemble based on the decision tree learner gives better accuracy than all of the other clustering algorithms irrespective of feature dependency or independence when clusters are clearly defined. When the clusters contain substantial overlap, the Fuzzy C-Means algorithm appears to be a better choice.

When we consider the wine data application, an application with clearly defined clusters, we see the ensembles based on the SVM and the decision trees produce more accurate clusters than the Fuzzy C-Means algorithm. However, when we consider the glass data application, we see the results from the different clustering methods differ. Each method of clustering yields a different number of clusters in the final dataset while obtaining similar accuracy. This may suggest that ensemble-based methods of clustering may be useful in helping to determine the best number of clusters that exists within a dataset when there is a high amount of overlap between clusters if a hard solution must be made. However, the methodology presented in this paper is a novel approach and one that can be improved upon with future research.

As mentioned in the introduction to this paper, proposed clustering ensemble algorithms should exhibit properties such as robustness, novelty, consistency, and stability (Vega-Pons and Ruiz-Shulcloper, 2011). Based on this, future research should be done to ensure proposed algorithms like those presented in this paper meet such criteria. Specific suggestions for future research include objective methods to choose tuning parameters for the supervised learners used in each algorithm; methods for choosing the number of clustering solutions to be generated in the generation step; objective assessment of the proper size necessary for the pseudo-training set; methods to check convergence of clustering algorithms; and methods for assessing stability of clustering results. Future research on ensemble-based methods of clustering that utilize supervised learners is valuable since many applications may be affected by their use. Such applications include fuzzy clustering in geology or medicine, as well as, big data analytics where ensemble methods have already proven effective.

Chapter 5

Plans for Future

One goal of this dissertation was to explore more efficient methods of cluster analysis that can be used in the clustering of multivariate categorical data. To this end, in the future I hope to explore methods of clustering spatial binary data with an end goal of creating a clustering algorithm that produce accurate clustering solutions in this special case. To do this I suggest the utilization of supervised methods of classification—similar to what was done in Chapter 4.

5.1 Introduction

The idea for the spatial binary clustering project was motivated by a previous study by Hiers et al. (2009) in which there was a need to cluster point-intercept data collected from controlled forestry burns in the Southeastern region of the United States. In Hiers et al. (2009), wildlife fuel cells were studied to examine the relationship between fuel and fire behavior at ”fine” (very small) scales. The authors noted that much of the information pertaining to this relationship was lost in the controlled burns; therefore improvement could come through understanding the variation in those burned areas. The hope was that past burns could be used to predict the effects that may be observed in future burns—a major concern for forestry. Within this previous study, cluster analysis was performed to group the fuel cell plots based on various binary features; however, the spatial components of the data were ignored. It is possible that, had the spatial information been used, the final clustering produced could have been different. To remedy such issues, it is pertinent to consider clustering methods that can make use of different aspects of the objects being clustered–including their spatial components.

5.2 Clustering Binary Spatial Data

The clustering of spatial data has been studied extensively in the last few years due to increased technological advances that allow for spatial information to be easily collected for example, through the use of satellite imaging or Global Positioning System (GPS) tracking software. However, as mentioned in Section 2.4, the clustering of categorical data has only been studied extensively since the early 1990s. A problem that has seen less investigation is that of clustering binary spatial data. One of the more popular algorithms for the clustering of spatial data appears to be that of the CLARANS (Ng and Han, 2002) algorithm. This algorithm, while a partition-based method, solves the problem of clustering by searching through a graph. In this context, the algorithm begins like K-Medoids by randomly selecting k objects as medoids. Then the graph is constructed with each node on the graph being represented as a set of objects with each neighboring node differing by only one object. The algorithm iterates by randomly choosing new neighboring nodes. If it represents a better medoid, the neighboring node becomes the medoid. Otherwise, the current node is considered a local minimum (see Ng and Han (2002) for more information). This particular algorithm is one that has been generated with the intent of being applicable in the clustering of large databases of spatial data. In this section we present an idea for an algorithm that may be used for the clustering of spatial binary data that uses a simpler idea of distance relevant to the nature of the spatial component.

Binary spatial data can be thought of as data that has arisen from various spatial regions with each variable denoting the absence or presence of some notable feature. For example, a spatial data set may contain a feature that records whether mountain ranges or oceans are a part of a region. The spatial component is based on the fact that every location can be represented by a latitude and longitude measurement. The goal of clustering then, would be to cluster the geographical regions in such a manner that those in the same cluster share more features with each other than they do to other geographical regions in other clusters. To do this, we propose reformulation of the problem into one of clustering mixed variable data. In this context, the spatial component can be thought of as a quantitative variable and the binary variables as qualitative. For the point-intercept problem, we propose using a distance metric more appropriate for the fine-scale nature of the data for the quantitative parts and a dissimilarity measure like those given in Chapter 2 for the binary parts. Another option in that case is to use down-weighting of more distant points if something like Euclidean distance will be used. When viewed in this context, the clustering of such spatial binary data may not be a new idea.

The forestry-burn data, however, motivated an additional problem for investigation—that of clustering binary spatial data when the spatial observations correspond to various locations across the Earth. Such a framework can be utilized in clustering problems associated with the Global Terrorism Database (LaFree and Dugan, 2007). This database records global terrorism attacks that have occurred since 1970 and records location (in latitude and longitude) as well as additional properties relevant to the attack such as country, means, and whether an attack was ongoing. Within this framework of spatial data, a natural idea for clustering of such binary spatial data is to use the Vincenty formula (which is typically used for geodesic length) as a way to calculate the distance between the locations for each pair of data objects and to use a dissimilarity measure (based on a similarity measure shown in Chapter 2) to calculate distance between the binary attributes. Combining these two methods in an appropriate manner may allow for a way to cluster such binary spatial data. In the context of the Global Terrorism Database datafile, this amounts to clustering the terrorism attacks (restricted to the United States) based on some of the binary measurements present. In this dataset the variables considered include: Extended incident (yes or no), criterion 1 (yes or no), criterion 2 (yes or no), criterion 3 (yes or no), doubt terrorism proper (yes or no), part of multiple incident (yes or not), success (yes or no), suicide attack (yes or no), claim of responsibility (yes or no), competing claim (yes or no), casualties (yes or no). One should note that each of the criterion (1-3) refer to the type of terrorism attack. The last variable (casualties) is one that we measure based on whether a casualty is reported. Note this classification could also be formed based on a certain threshold casualty level. My future research considers clustering in this framework. The proposed method relies heavily on Vincenty's inverse formula for geodesics, which will be discussed next.

5.2.1 Vincenty's Formula for Geodesics

Geodesics can be defined as a "natural straight line defined as the line of minimum curvature for the surface of the Earth" (Karney, pg.1, 2011). Such lines are of interest as the "shortest path between any two points on Earth is always a geodesic" (Karney, pg.1, 2011). Vincenty (1975) presents a solution to two common problems in geodesics. For this project, the second solution, that of finding the length of the geodesic between two points on an ellipsoid, is most relevant. To discuss the algorithm, the notation from Vincenty (1975) and Karney (2011) are borrowed (and mixed), as well as an image from the latter to aid with visualization. For notation, assume there are two points, denoted A and B, for which interest is in determining the length of the geodesic, denoted AB, connecting them.

Notation:

- a, b major and minor semi-axes
- $f = \frac{a-b}{a}$, flattening parameter
- ϕ, geodetic latitude

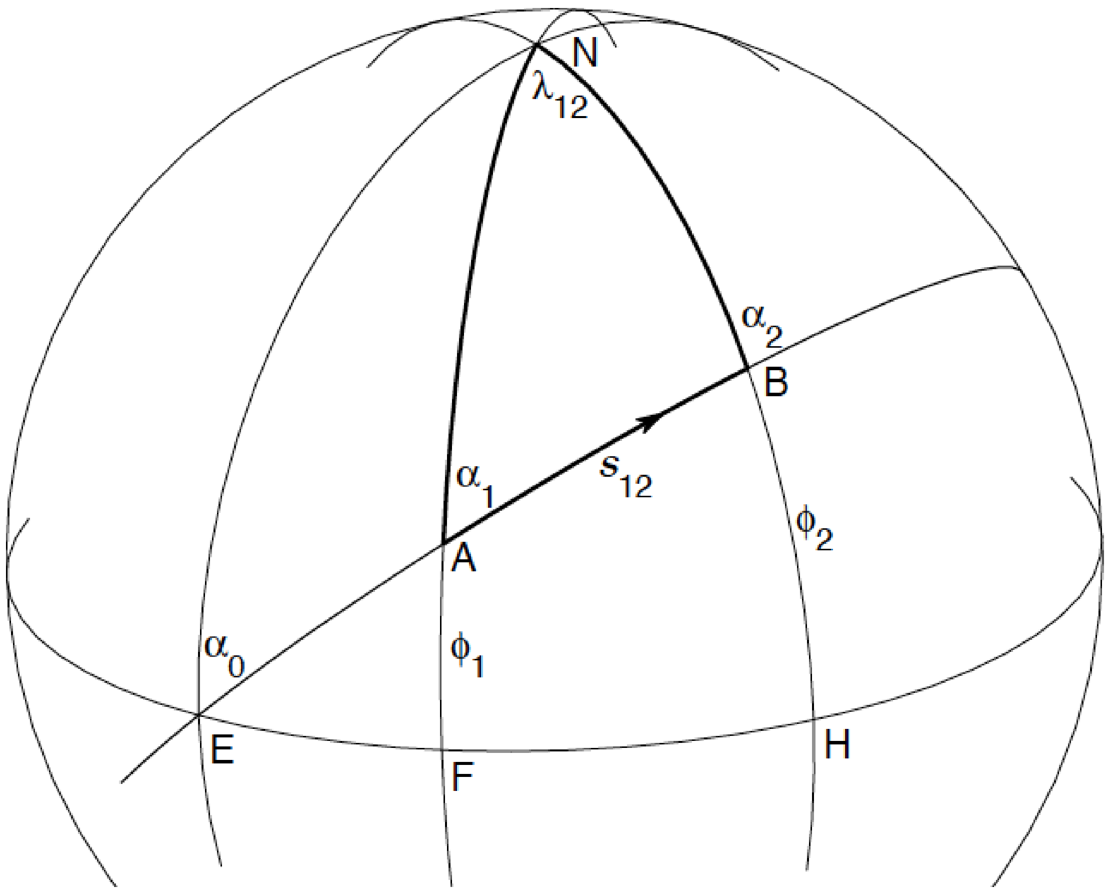


Figure 5.1 Depiction of an ellipsoid (from Karney (pg. 1, 2011)) used here to clarify notation used for (Vincenty, 1975) inverse solution. In this figure ϕ_1 and ϕ_2 denote the latitude of points A and B, respectively. α_1 and α_2 denote the azimuths of the geodesics at points A and B, respectively. α_0 denotes the azimuth of the geodesic at the equator (here E is a point on the equator that lies on an extension of the geodesic AB). s_{12} denotes the length of geodesic AB. N denotes the North Pole, and points E, F, and H are just points on the equator EFH. λ_{12} denotes the longitude of B relative to A. (Karney, 2011)

- L, difference in longitude between A and B, positive east.
- s, length of the geodesic
- α_1, α_2 azimuths (bearings) of the geodesic at point A and point B, respectively, measured clockwise from the north
- α_0, azimuth of the geodesic at the equator
- $u^2 = (\cos \frac{\alpha_0(a^2-b^2)}{b^2})^2$
- U, reduced latitude defined as $\tan U = (1-f)\tan\phi$
- λ, difference in longitude on an auxiliary sphere.
- σ, angular distance AB on the sphere
- σ_1, angular distance on sphere from the equator to A
- σ_m, angular distance on the sphere from the equator to the midpoint of the line

Using this notation, Vincenty (1975) presents a method that can be used to give an approximate distance between the two points. This algorithm is given below.

$$\lambda = L \text{ (first approximation)} \tag{5.1}$$

$$(\sin\sigma)^2 = (\cos U_2 \sin\lambda)^2 + (\cos U_1 \sin U_2 - \sin U_1 \cos U_2 \cos\lambda)^2 \tag{5.2}$$

$$\cos\sigma = \sin U_1 \sin U_2 + \cos U_1 \cos U_2 \cos\lambda \tag{5.3}$$

$$\tan\sigma = \frac{\sin\sigma}{\cos\sigma} \tag{5.4}$$

$$\sin\alpha_0 = \cos U_1 \cos U_2 \frac{\sin\lambda}{\sin\sigma} \tag{5.5}$$

$$\cos 2\sigma_m = \cos\sigma - 2\sin U_1 \frac{\sin U_2}{(\cos\alpha_0)^2} \tag{5.6}$$

The calculation of the geodesic length begins using the difference in the longitude of points A and B as an initial starting approximation for λ as shown in Equation (5.1).

Next, Equations (5.2)—(5.6) are iterated with λ being updated each round according to Equations (5.7) and (5.8) until λ remains relatively constant.

$$C = \frac{f}{16}(\cos\alpha_0)^2\{4 + f(4 - 3(\cos\alpha_0)^2\} \tag{5.7}$$

$$\lambda = L + (1 - C)f\sin\alpha_0\{\sigma + C\sin\sigma[\cos 2\sigma_m + C\cos\sigma(-1 + 2(\cos 2\sigma_m)^2)]\} \tag{5.8}$$

$$s_{12} = bX(\sigma - \Delta\sigma) \tag{5.9}$$

$$X = 1 + \frac{u^2}{16384}\{4096 + u^2[-768 + u^2(320 - 175u^2)]\} \tag{5.10}$$

$$Y = \frac{u^2}{1024}256 + u^2[-128 + u^2(74 - 47u^2)] \tag{5.11}$$

$$\begin{aligned}\Delta\sigma = Y\sin\sigma\Bigg\{ \cos 2\sigma_m + \frac{1}{4}Y\Big[\cos\sigma(-1 + 2(\cos 2\sigma_m)^2 \\ -\frac{1}{6}Y\cos 2\sigma_m(-3 + 4(\sin\sigma)^2(-3 + 4(\cos 2\sigma_m)^2)\Big]\Bigg\}\end{aligned} \tag{5.12}$$

At this point the geodesic length between points A and B can be calculated as given in Equation (5.9) with X, Y, and $\Delta\sigma$ defined as in Equations (5.10), (5.11), and (5.12) respectively.

Vincenty's inverse formula gives the geodesic distance between two points on an ellipsoid; however, of interest to this project is distance when two points are on Earth. In this setting, Earth represents a specific ellipsoid. To this end, the focus is on the World Geodetic System (National Imagery and Mapping Agency, 1997) with the latest version denoted as WGS84. This particular model of the Earth provides a 3-dimensional coordinate system for geospatial data (National Imagery and Mapping Agency, 1997). It is this specific model of the Earth (WGS 84) that modern technology like GPS receivers reference. With this being the case, it is considered for this project. To use the WGS84 model with Vincenty's inverse formula we define the following parameters as shown below, borrowed from National Imagery and Mapping Agency (1997):

- $a = 6378137.0$ meters

- $b = 6356752.3142$ meters
- $\frac{1}{f} = 298.257223563$

The gdist function in the Imap package of R (R Core Team, 2019) employs Vincenty's inverse formula with reference to the WGS84. This function may be used to calculate the distance between the ith and jth observations within the analysis. One possible ensemble method of doing this could be given as shown below:

1. Select a random sample of observations to be used as a training set.
2. Cluster these observations (ignoring the spatial variables) using the Jaccard similarity measure.
3. Assign the clustering solutions obtained as labels for each observation in the training set.
4. Use a k-nearest neighbor classifier trained using Vincenty's distance to assign a cluster membership to the remaining observations.
5. Output the final results as a clustering solution.

In the aforementioned algorithm, the binary parts of the observations are being considered within the clustering, and the spatial parts are being considered within the k-nearest neighbor step. Therefore, each aspect, which is equally important, is influencing the final clustering result. If this method works, then it may be possible to weigh each component differently so that the spatial or binary components can be given more importance. After reviewing results from Chapter 4, it may also be helpful to consider an ensemble approach.

To reiterate, for the purposes of clustering the original fine-scale data set, the aforementioned method may not be appropriate as Vincenty's distance is more appropriate for calculating the distance between objects many degrees of latitude and/or longitude apart. For the point-intercept data the locations are close together, thus the Vincenty's distance should replaced with a more appropriate method for dealing with small-scale distances. Vincenty's method is mentioned as it appears to be a possible approach to solve a widely-applicable problem needing further investigation and it makes use of GPS. Promising results in this project have the ability to impact many other geospatial applications. This will be the initial focus for my future research; however, other measures of distance could also be used depending on the type of spatial data included.

Chapter 6

Conclusion

Technological innovations of the 21st century have made it fairly easy to collect a wealth of data from various sources resulting in a need to then process such data. Cluster analysis is one of the ways conclusions can be made from such data and in fact, has helped researchers and practitioners in business, psychology, anthropology, information retrieval, and many other fields to solve real-life problems and find meaning in massive datasets. New methods that seek to improve pre-existing cluster analysis algorithms are pertinent as any improvements to the state of the art of cluster analysis have the potential to impact a multitude of fields outside of statistics. With this in mind, this dissertation sought to introduce improved multivariate methods of cluster analysis for just this purpose.

In Chapter 3 the focus was on the clustering of tertiary data. In this chapter, the outcomes of the simulation studies and the Pima Indian Diabetes data application suggested the accuracy of cluster solutions could be improved by the use of the Fienberg-Holland estimator. By pre-smoothing dissimilarities using this shrinkage-type estimator, cluster solutions produced were shown to be more reflective of the true latent structure of the data in noisy settings as well as those in which the clusters are not well-defined when measured by the Adjusted Rand Index. In the cases where the data was not noisy nor the clusters well-defined, pre-smoothing of the dissimilarities was not necessary.

In Chapter 4 novel ensemble-based methods of cluster analysis were introduced and investigated. In this chapter, we discovered several promising findings when considering the combination of supervised and unsupervised methods of classification for fuzzy clustering. In particular, our results suggest using decision trees, support vector machines, or the k-nearest neighbor algorithms within a clustering ensemble yields more accurate soft and hard clustering solutions in some settings as compared to the Fuzzy C-Means algorithm. These conclusions depend upon the type of distribution from whence data objects have arisen as well as the variability within the latent structure of the data.

When the data had arisen from a symmetric distribution like a Gaussian or t-based distribution, the clustering algorithms based on the support vector machine or decision tree showed the most promising results of all the proposed algorithms, especially in the cases where the latent structure of the clusters exhibit high variability. In the cases where there was less variability in the latent structure, the improvement decreased. In both cases, however, the performance by the best proposed fuzzy ensemble algorithm still was not better than that produced by the Fuzzy C-Means algorithm.

On the other hand, when the variables measured on the data objects had arisen from a highly-skewed distribution, like the Lognormal distribution, the performance by the decision tree-based fuzzy ensemble algorithm produced the best results of all the proposed fuzzy ensemble algorithms. Furthermore, in the cases where the latent structure of the data showed high variability, the decision-tree based method produced the most accurate hard clustering partitions as measured by the average Adjusted Rand Index via simulations, even outperforming the Fuzzy C-Means algorithm. When the clusters were more defined, the differences were not as drastic, though the decision tree-based methods still performed well.

When considering the wine and glass data applications, the proposed fuzzy ensemble-based algorithms also showed promise. In the case where the clusters were well-defined (wine dataset), the most accurate hard clusters were produced through the use of the support vector machine-based ensemble followed by the decision tree-based method. When considering the soft clustering solutions, the most accurate solutions were produced by the k-nearest neighbor-based ensemble followed by the Fuzzy C-Means algorithm.

When considering the case where the clusters were not well-defined, (the glass dataset), the results were not as straightforward. In this case the hard clustering accuracy was marginally higher for the decision tree-based and support vector machine-based fuzzy ensembles. Both were trailed closely by the Fuzzy C-Means algorithm. When considering the soft clustering solutions, the support vector machine-based algorithm, followed by the decision-tree based method, produced the best accuracy. In this case, however, the accuracy of these methods were notably better than the Fuzzy C-Means algorithm. The general findings of this chapter suggest it is best to first consider whether the interest lies in a soft or hard solution, as well as the believed latent structure of the data before concluding which ensemble is the better method of choice.

The ongoing project of Chapter 5 also has the potential to improve the field of cluster analysis through its novelty. This chapter in particular seeks to produce a new algorithm for the clustering of binary spatial data through an approach similar to that used in Chapter 4 with motivations arising from the Global Terrorism Database. It sketches a tentative outline to use supervised learners with geodesic application in order to cluster the locations of terrorist attacks. If this approach proves fruitful, to my knowledge, at the time of this dissertation it would be one of the simplest methods proposed to deal with the clustering of such binary spatial data and one that uses a seemingly unrelated field of knowledge (geodesics).

Many of the methodologies introduced in this dissertation have used ideas from other fields with the goal of producing updated algorithms that are also more efficient. This should not be surprising considering the history of cluster analysis (see e.g., Chapter 1) in which much innovation has been sparked by researchers outside of the field of cluster analysis. This dissertation does similarly. More specifically, the goal in Chapter 3 was to improve the accuracy of clustering solutions produced for tertiary data objects using statistical smoothing, while in Chapter 4 the focus was on creating fuzzy ensemble algorithms that could produce more accurate clustering solutions than the well-known Fuzzy C-Means algorithm (Bezdek, 2013) through the use of machine learning methodologies. Finally, in Chapter 5, an idea was produced with a goal to create stable and accurate clustering solutions in binary spatial data objects using applications of geodesics. Each of the methods showed promise in certain settings and provide evidence in support of combining traditional methodology from various fields with that of cluster analysis. It is my hope that this dissertation has effectively showcased the need for such work and sparked interests in future collaborative efforts in the field of cluster analysis.

Bibliography

Agresti, Alan. 2012. *Categorical Data Analysis.* Hoboken, New Jersey: John Wiley & Sons, Inc.

AL-Akhras, Mousa. 2010. "An Efficient Fuzzy K-Medoids Method". *World Applied Sciences Journal* 10:574–583.

Albalate, Amparo and Minker, Wolfgang. 2011. *Semi-Supervised and Unsupervised Machine Learning: Novel Strategies.* Hoboken, New Jersey: John Wiley & Sons, Inc.

Albert, James H. 1987. "Empirical Bayes Estimation in Contingency Tables". *Communications in Statistics-Theory and Methods* 16 (8): 2459–2485. doi:10.1080/03610928708829518.

American Heart Association. 2017. *Understanding Blood Pressure Readings.* http://www.heart.org/en/health-topics/high-blood-pressure/understanding-blood-pressure-readings.

American Psychiatric Association. 2013. *Diagnostic and Statistical Manual of Mental Disorders.* Fifth Edition. Arlington, Virginia: American Psychiatric Association.

Banfield, Jeffrey D and Raftery, Adrian E. 1993. "Model-Based Gaussian and Non-Gaussian Clustering". *Biometrics* 49 (3): 803–821.

Bezdek, James C. 2013. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Boston, MA: Springer. doi:10.1007/978-1-4757-0450-1.

Bezdek, James C, Ehrlich, Robert, and Full, William. 1984. "FCM: The Fuzzy C-Means Clustering Algorithm". *Computers & Geosciences* 10 (2-3): 191–203.

Biernacki, Christophe and Jacques, Julien. 2016. “Model-Based Clustering of Multivariate Ordinal Data Relying on a Stochastic Binary Search Algorithm”. *Statistics and Computing* 26 (5): 929–943.

Boriah, Shyam, Chandola, Varun, and Kumar, Vipin. 2008. “Similarity Measures for Categorical Data: A Comparative Evaluation”. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, 243–254. SIAM.

Borole, Rachana Tushar. 2020. “A Survey on Data Mining Techniques”. *Advance and Innovative Research*: 209.

Choi, Seung-Seok, Cha, Sung-Hyuk, and Tappert, Charles C. 2009. “Correlation Analysis of Binary Similarity and Distance Measures on Different Binary Database Types”. In *Proceedings of the International Conference on Artificial Intelligence and Pattern Recognition*. Orlando, Florida, USA.

Cornell, John E et al. 2009. “Multimorbidity Clusters: Clustering Binary Data from Multimorbidity Clusters: Clustering Binary Data from a Large Administrative Medical Database”. *Applied Multivariate Research* 12 (3): 163–182. doi:10.1080/03610928708829518.

Davies, David L and Bouldin, Donald W. 1979. “A Cluster Separation Measure”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1 (2): 224–227.

Dempster, Arthur P, Laird, Nan M, and Rubin, Donald B. 1977. “Maximum Likelihood from Incomplete Data Via the EM Algorithm”. *Journal of the Royal Statistical Society: Series B (Methodological)* 39 (1): 1–22.

Dietterich, Thomas G. 2000. “Ensemble Methods in Machine Learning”. In *International Workshop on Multiple Classifier Systems*, 1–15. Springer.

Dolnicar, Sara and Leisch, Friedrich. 2004. "Segmenting Markets by Bagged Clustering". *Australasian Marketing Journal (AMJ)* 12 (1): 51–65. doi:`10.1016/S1441-3582(04)70088-9`.

Dua, Dheeru and Graff, Casey. 2019. *UCI Machine Learning Repository.* University of California, Irvine, School of Information and Computer Sciences. `http://archive.ics.uci.edu/ml`.

Efron, Bradley and Morris, Carl. 1977. "Stein's Paradox in Statistics". *Scientific American* 236 (5): 119–127. doi:`10.1038/scientificamerican0577-119`.

Elgendy, Nada and Elragal, Ahmed. 2016. "Big Data Analytics in Support of the Decision Making Process". *Procedia Computer Science* 100:1071–1084.

Ester, Martin et al. 1996. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 96:226–231. 34.

Everitt, Brian S et al. 2011. *Cluster Analysis.* West Sussex, United Kingdom: John Wiley & Sons.

Fienberg, Stephen E and Holland, Paul W. 1973. "Simultaneous Estimation of Multinomial Cell Probabilities". *Journal of the American Statistical Association* 68 (343): 683–691. doi:`10.1080/01621459.1973.10481405`.

Fraley, Chris and Raftery, Adrian E. 2002. "Model-Based Clustering, Discriminant Analysis, and Density Estimation". *Journal of the American Statistical Association* 97 (458): 611–631.

Friedman, Herman P and Rubin, Jerrold. 1967. "On Some Invariant Criteria for Grouping Data". *Journal of the American Statistical Association* 62 (320): 1159–1178.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. 2017. *The Elements of Statistical Learning: Data Mining, Inference,and Prediction.* New York, New York: Springer Series in Statistics. doi:10.1007/b94608.

Gan, Guojun, Wu, Jianhong, and Yang, Zijiang. 2009. "A Genetic Fuzzy K-Modes Algorithm for Clustering Categorical Data". *Expert Systems with Applications* 36 (2): 1615–1620.

Grabowski, Mary Kate, Herbeck, Joshua T, and Poon, Art FY. 2018. "Genetic Cluster Analysis for HIV Prevention". *Current HIV/AIDS Reports* 15 (2): 182–189.

Guha, Sudipto, Rastogi, Rajeev, and Shim, Kyuseok. 1998. "CURE: An Efficient Clustering Algorithm for Large Databases". In *ACM Sigmod Record*, 27:73–84. 2. ACM.

— . 2000. "ROCK: A Robust Clustering Algorithm for Categorical Attributes". *Information Systems* 25 (5): 345–366.

He, Zengyou et al. 2005. "K-Histograms: An Efficient Clustering Algorithm for Categorical Dataset". *arXiv preprint cs/0509033*.

Hiers, J Kevin et al. 2009. "The Wildland Fuel Cell Concept: An Approach to Characterize Fine-Scale Variation in Fuels and Fire in Frequently Burned Longleaf Pine Forests". *International Journal of Wildland Fire* 18 (3): 315–325.

Hitchcock, David B and Chen, Zhimin. 2008. "Smoothing Dissimilarities to Cluster Binary Data". *Computational Statistics and Data Analysis* 52 (10): 4699–4711. doi:10.1016/j.csda.2008.03.012.

Huang, Anna. 2008. "Similarity Measures for Text Document Clustering." In *Proceedings of the Sixth New Zealand Computer Science Research Student Conference(NZCSRSC2008), Christchurch, New Zealand*, 4:9–56.

Huang, Zhexue. 1997a. "A Fast Clustering Algorithm to Cluster very Large Categorical Data Sets in Data Mining." *DMKD* 3 (8): 34–39.

— . 1997b. “Clustering Large Data Sets with Mixed Numeric and Categorical Values”. In *Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 21–34. Singapore.

Huang, Zhexue and Ng, Michael K. 1999. “A Fuzzy K-Modes Algorithm for Clustering Categorical Data.” *IEEE Transactions on Fuzzy Systems* 7 (4): 446–452.

Hubert, Lawrence and Arabie, Phipps. 1985. “Comparing Partitions”. *Journal of Classification* 2 (1): 193–218. doi:`10.1007/bf01908075`.

Jacques, Julien and Biernacki, Christophe. 2018. “Model-Based Co-Clustering for Ordinal Data”. *Computational Statistics & Data Analysis* 123:101–115.

Karney, Charles FF. 2011. “Geodesics on an Ellipsoid of Revolution”. *arXiv preprint arXiv:1102.1215*.

Kaufman, Leonard and Rousseeuw, Peter J. 1987. “Clustering by Means of Medoids”. *Statistical Data Analysis based on the L1 Norm*: 405–416.

LaFree, Gary and Dugan, Laura. 2007. “Introducing the Global Terrorism Database”. DOI:10.1080/09546550701246817, *Terrorism and Political Violence* 19 (2): 181–204.

Limpert, Eckhard, Stahel, Werner A, and Abbt, Markus. 2001. “Log-normal distributions across the sciences: keys and clues: on the charms of statistics, and how mechanical models resembling gambling machines offer a link to a handy way to characterize log-normal distributions, which can provide deeper insight into variability and probability—normal or log-normal: that is the question”. *BioScience* 51 (5): 341–352.

MacQueen, James. 1967. “Some Methods for Classification and Analysis of Multivariate Observations.” In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297. 14. Oakland, CA, USA.

Maechler, Martin et al. 2018. *cluster: Cluster Analysis Basics and Extensions.* R package version 2.0.7-1.

Mandinach, Ellen B. 2012. “A Perfect Time for Data Use: Using Data-Driven Decision Making to Inform Practice”. *Educational Psychologist* 47 (2): 71–85.

Masmoudi, Nesrine et al. 2015. “How to Use Ants for Data Stream Clustering”. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, 656–663. IEEE.

Mateen, Muhammad et al. 2018. “Text Clustering using Ensemble Clustering Technique”. *International Journal Of Advanced Computer Science and Applications* 9 (9): 185–190.

Mayo Clinic. 2019. *Glucose Tolerance Test.* `http://www.mayoclinic.org/tests-procedures/glucose-tolerance-test/about/pac-20394296`.

McNicholas, Paul D. 2017. *Mixture Model-Based Classification.* Boca Raton, Florida: Taylor & Francis Group.

Meyer, David et al. 2019. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien.* R package version 1.7-2. `https://CRAN.R-project.org/package=e1071`.

National Imagery and Mapping Agency. 1997. *Department of Defense World Geodetic System 1984: Its Definition and Relationship with Local Geodetic Systems.* National Imagery / Mapping Agency.

Ng, Raymond T and Han, Jiawei. 2002. “CLARANS: A Method for Clustering Objects for Spatial Data Mining”. *IEEE Transactions on Knowledge & Data Engineering*, no. 5: 1003–1016.

Nguyen, Thi Phuong Quyen and Kuo, RJ. 2019. “Partition-and-Merge Based Fuzzy Genetic Clustering Algorithm for Categorical Data”. *Applied Soft Computing* 75:254–264.

R Core Team. 2019. *R: A Language and Environment for Statistical Computing.* `https://www.R-project.org`. Vienna, Austria: R Foundation for Statistical Computing.

Ralambondrainy, Henri. 1995. “A Conceptual Version of the K-Means Algorithm”. *Pattern Recognition Letters* 16 (11): 1147–1157.

Rand, William M. 1971. “Objective Criteria for the Evaluation of Clustering Methods”. *Journal of the American Statistical Association* 66 (336): 846–850. doi:`10.1080/01621459.1971.10482356`.

Rousseeuw, Leonard and Kaufman, Peter J. 1987. *Clustering by Means of Medoids.* Ed. by In: Dodge Y and editor.

Rousseeuw, Peter J. 1987. “Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis”. *Journal of Computational and Applied Mathematics* 20:53–65. doi:`10.1016/0377-0427(87)90125-7`.

Sanse, Keshav and Sharma, Meena. 2015. “Clustering Methods for Big Data Analysis”. *International Journal of Advanced Research in Computer Engineering & Technology* 4 (3).

Saraçli, Sinan, Doğan, Nurhan, and Doğan, İsmet. 2013. “Comparison of Hierarchical Cluster Analysis Methods by Cophenetic Correlation”. *Journal of Inequalities and Applications* 2013 (1): 203.

Sarumathi, S, Shanthi, N, and Sharmila, M. 2013. “A Comparative Analysis of Different Categorical Data Clustering Ensemble Methods in Data Mining”. *International Journal of Computer Applications* 81 (4).

Scrucca, Luca et al. 2016. “mclust 5: Clustering, Classification and Density Estimation using Gaussian Finite Mixture Models”. *The R Journal* 8 (1): 205–233.

Simonoff, Jeffrey S. 1995. “Smoothing Categorical Data”. *Journal of Statistical Planning and Inference* 47 (1-2): 41–69. doi:`10.1016/0378-3758(94)00121-b`.

— . 1998. *Smoothing Methods in Statistics*. New York, New York: Springer-Verlag New York, Inc. doi:10.1007/978-1-4612-4026-6.

— . 2012. *Smoothing Methods in Statistics*. New York, New York: Springer-Verlag, New York, Inc.

Smith, Andrew. 2019. *Consumer Behaviour and Analytics: Data Driven Decision Making*. Routledge.

Sokal, R R and Michener, C D. 1958. “A Statistical Method for Evaluating Systematic Relationships”. *University of Kansas Science Bulletin* 38:1409–1438.

Soltysiak, Arkadiusz and Jaskulski, Piotr. 1999. “Czekanowski’s Diagram: A Method of Multidimensional Clustering”. *BAR International Series* 757:175–184.

Thangamani, M and Ibrahim, S Jafar Ali. 2018. “Ensemble Based Fuzzy with Particle Swarm Optimization Based Weighted Clustering (Efpso-Wc) and Gene Ontology for Microarray Gene Expression”. In *Proceedings of the 2018 International Conference on Digital Medicine and Image Processing*, 48–55.

Therneau, Terry and Atkinson, Beth. 2019. *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-15. https://CRAN.R-project.org/package=rpart.

Tibshirani, Robert, Walther, Guenther, and Hastie, Trevor. 2001. “Estimating the Number of Clusters in a Data Set via the Gap Statistic”. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63 (2): 411–423.

Vega-Pons, Sandro and Ruiz-Shulcloper, José. 2011. “A Survey of Clustering Ensemble Algorithms”. *International Journal of Pattern Recognition and Artificial Intelligence* 25 (03): 337–372.

Venables, W. N. and Ripley, B. D. 2002. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer. http://www.stats.ox.ac.uk/pub/MASS4.

Vincenty, Thaddeus. 1975. “Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations”. *Survey Review* 23 (176): 88–93.

Wang, Gang et al. 2018. “FCE-SVM: A New Cluster Based Ensemble Method for Opinion Mining from Social Media”. *Information Systems and e-Business Management* 16 (4): 721–742.

Ward, Joe H. 1963. “Hierarchical Grouping to Optimize an Objective Function”. *Journal of the American Statistical Association* 58 (301): 236–244.

Windgassen, S et al. 2018. “The Importance of Cluster Analysis for Enhancing Clinical Practice: An Example from Irritable Bowel Syndrome.” *Journal of Mental Health* (Abingdon, England) 27 (2): 94–96.

Zhang, Tian, Ramakrishnan, Raghu, and Livny, Miron. 1996. “BIRCH: An Efficient Data Clustering Method for Very Large Databases”. In *ACM Sigmod Record*, 25:103–114. 2. ACM.

Zubin, Joseph. 1938. “A Technique for Measuring Like-Mindedness”. *The Journal of Abnormal and Social Psychology* 33 (4): 508–516.