2015

Pre-processing for K-means Clustering Algorithm

Mohd Ahnaf Habib Khan
Bard College

Pre-processing for K-means Clustering Algorithm

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Mohd Ahnaf Khan

Annandale-on-Hudson, New York
May, 2015

Abstract

The K-means clustering algorithm works on a data set with n data points in d dimensional space $\mathbb{R}^d$. It determines a set of K centroids in $\mathbb{R}^d$. Clustering is accomplished by assigning each point in the data set to its closest centroid. However, the K-means algorithm has a few draw backs. First, it requires the value of K, number of centroids, to be pre-specified. Second, the algorithm begins by randomly selecting the centroid locations. Third the accuracy of the output clusters in K-means is dependent on the type of clustering in the data.

In this paper we propose a distance based definition for the clusterability of a data set using the edges in a Delaunay triangulation. We propose an algorithm to pre-process K-means; the output of the algorithm contains a range for K and initial centroid information. Our results show that a pre-processed K-means requires a lower number of iterations to reach completion. Also, by using cluster evaluation techniques such as the F-measure, Purity, and Entropy, we show that the results obtained from a pre-processed K-means consistently produces more accurate clusters.

We also propose a new clustering algorithm which uses Delaunay triangulation to obtain clusters. We also show that this algorithm produces very accurate clusters in large number of data sets, even in data sets where K-means fails.

Contents

List of Figures

Dedication

To my father, mother, and brother

Acknowledgments

I would like to express my sincere gratitude to my adviser Mary Krembs for her continuous support and patience. I am extremely grateful to her for listening to me, understanding me, and foreseeing a topic that would be well suited to my interests. I would like to thank her for keeping me motivated and believing in me all through this process. Apart from my adviser, I would like to thank the other members of my board Amir Barghi and Maria Belk for their guidance during the mid way board. Their insight really helped me move forward with my project. I would like to thank them for their willingness to be on my board. I am deeply grateful to the professors in the Mathematics, Economics and Computer Science Department whose expertise and supervision helped me grow as an individual. I would not be here today without the Distinguished Scientist Scholarship, and I would like to thank Bard College for honoring me with the opportunity.

I would also like to thank my parents Sabina Rahman Khan and Habibur Rahman Khan. They have been nothing but the greatest support in my life. Their hard work and sacrifices to ensure that I receive a good education was the inspiration and driving force that made me want to perform better every day. I would also like to thank my brother Azfar Khan who has always been my role model. Last but not least, I would also like to thank Zahra Ahmed for supporting me through this project.

1
Introduction

Data mining is the process of analyzing data from different perspectives and summarizing it into useful information. Fayyad et al states that data mining is used to find correlations and patterns in large relational databases with multiple dimensions [6]. Clustering is a form of data mining where data are grouped together into disjoint clusters or sets based on some form of similarity measure[1] between these data. This is done such that data in one cluster are 'similar' and data from two different clusters are 'different'. The greater the similarity within a cluster and the greater the difference between clusters, the better or more distinct the clustering. Figure 1.0.1 shows the formation of two distinct clusters from a given set of data.

Clustering is important because of its vast applications in many different fields such as pattern recognition, image analysis, bioinformatics, machine learning, voice and text mining, web search engines, weather report analysis, market organization, retail, finance, and academics [11]. For example in supermarkets clustering can be done to determine optimal market baskets. That is the supermarket can record customer purchase information and

[1] See definition Definition 2.2.1

determine which types of items are usually bought together. They can then use this information to arrange their store accordingly in order to increase sales. Clustering algorithms can also be used to improve results from search engines. Clustering algorithms can be used to bundle together the output of a specific search query into groups and then show the best result from each group. Clustering algorithms can also be used to improve other data analysis techniques. Data analysis techniques that have $O(m^2)$ or higher complexity can become extremely slow when working with large data sets [12]. A faster clustering algorithm such as the K-means can be used instead to obtain the centroids of the data. The results of data analysis on the centroids is comparable to that of the whole data set [12].

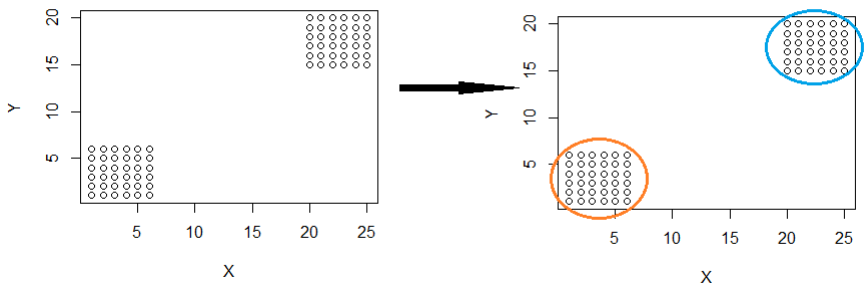

Figure 1.0.1. Clustering a set of data.

The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. Unlabeled data is the type of artifact which contains only the data and no additional information such as the classifications or grouping of the data. Clustering is the process of grouping together data objects using very limited or no information; it assigns elements of a data set into non predefined classes. This is important because labeled data are harder to build and maintain and hence rarely available. Jain, Murty and Flynn discuss the importance of clustering as an unsupervised classification of data [2]. Clustering is very

important method of analysis when inadequate information is available about the data and the analyst has to make decision about the data.

Clustering is a concept that is hard to define objectively and uniquely. That is because when working with unlabeled data, it is hard to determine what constitutes good clustering; different data analysis experiments require different conditions under which clustering must be accomplished. Because of this there are many different ways to define clustering: Hierarchical versus Partitional, Exclusive versus Overlapping versus Fuzzy, Complete versus Partial[2] [12]. Based on these different methods of defining a cluster, there are many different forms of data clustering models and algorithms. These models are based on specific features such as connectivity, centroids, distribution, or density. There is no one clustering method which is better than all others. Clustering aims to find useful groups of objects, where usefulness is defined solely by the goals of the data analysis. Therefore different sets of data or types of clusters require a different type of clustering method. The ideal method required for a specific set of data is determined either experimentally or mathematically and is unique to a specific data set.

This paper will address a specific method of clustering known as centroid based or distance based clustering. The centroid of a cluster is a point whose coordinate values are the mean of the coordinate values of all the points in the cluster. The centroid does not necessarily have to be a point in the set of data. Generally, the distance between two points is taken as a common metric to assess the similarity among the components of a set.

---

[2]Hierarchical clustering is a set of nested clusters in the form of a tree. The child nodes act as subclusters which when combined together form the super cluster in the parent node.
Partitional clustering is the separation of the data points into clusters such that the clusters do not overlap.
Exclusive clustering allows a specific data point to only be assigned to a single cluster.
Overlapping clustering allows a specific data point to be assigned to multiple clusters.
Fuzzy clustering assigned all data points to all clustering with a specific weight value ranging from 0 to 1, where 0 means that the point absolutely does not belong to the cluster and 1 means that the point absolutely belongs to the cluster.
Complete clustering assigns every point in the data to atleast one cluster.
Partial cluster does not require all points to be assigned to a cluster. That is, there may be points that do not belong to any cluster (outliers).

A commonly used similarity measure is the Euclidean metric which defines the distance between two points $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., p_n)$ a $l = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$. In centroid or distance-based clustering, the clusters are determined by assigning every point in the data to a specific centroid; the objective is to minimize the intra-cluster (interior) distance while maximizing the inter-cluster (exterior) distance. All the points assigned to a centroid represents a cluster. According to Epter, Krishnamoorthy and Zaki, centroid or distance-based models work very accurately when working with sets of data that have very definite clustering, preferably in spherically-shaped groups [10]. Centroid or distance-based clustering models are especially weak at clustering non convex clusters.

This paper will focus on improving the results of a centroid or distance based clustering algorithm, proposed first by James MacQueen in 1967 called the *K-means clustering algorithm.* K-means is a partitional clustering technique that produces K number of clusters, where K is an input parameter [12]. The algorithm was modified by Lloyd, and this version is the one that is most commonly used today [17]. However, even this version of the K-means has its issues. The accuracy and consistency of the results of K-means rely heavily on the value of K, choice of centroids, and similarity (distance) measure used [1,4,5,7,9,12,13,15]. K-means is also very sensitive to outliers.

We will propose the addition of a pre-processing algorithm to Lloyd's K-means. The pre-processing algorithm uses Delaunay triangulation to connect the points in the data set. The algorithm helps improve the results of K-means in three ways. First, by using a histogram of the Delaunay edge list, we can determine the clusterability[3] of a set of data by K-means. Second, by analyzing the change in number of clusters when edges above a given length are removed in the connected graph, we can determine the number of clusters K required for K-means. Last, by using the components of the Delaunay connected graph,

[3]Clusterability is a way to determine the strength of the clustering structure of a data set

we can generate seed centroids for the K-means. We will also propose a new distance-based clustering algorithm and compare its results with that of K-means. All the algorithms and test required for this project were developed and run using R [22].

The rest of this paper is structured as follows: Chapter 2 discusses past work and describes in detail how the K-means algorithm works, and its drawbacks. In Chapter 3 we define the key concepts used in this paper, and we discuss the framework of our algorithm. In Chapter 4 we describe methods to determine a cut-off length g which is critical to our algorithm. In Chapter 5 we compare the results obtained by Lloyd's K-means with that of our pre-processed K-means using different data sets. In Chapter 6 we propose a new distance based clustering algorithm using edges created via a Delaunay triangulation of the data set.

2
Background and K-means

2.1 Basic K-means

Wu et. al. mentions that among the vast number of clustering algorithms, K-means stands out because it is a simple algorithm, easily understood and reasonably scaleable [1]. It has varied applications in the fields of artificial intelligence, pattern recognition, economics, ecology, psychiatry, marketing and more.

The main purpose of K-means is to partition an N-dimensional data set into K sets or clusters [1,3,12]. The original K-means algorithm proposed by MacQueen takes K as input on a data set $X = \{x_1, ..., x_n\}$ [3]. The target is to select K cluster centers such that the squared distance between each point in the data and the center is minimized. The original design of the algorithm is NP-hard[1] [17].

Lloyd improves this algorithm by focusing on local optima instead [17]. His algorithm independently assigns $K \in \mathbb{R}^d$, where K is a user specified parameter, points $(w_1, ..., w_k)$ in the data set with equal probability p as the initial centroids. Each x_i is assigned to a centroid w_j where $j = 1, ..., K$ such that it minimizes the pairwise distance, that is each

[1]NP-hard: Non-deterministic polynomial-time hard

point is assigned to its closest centroid based on some similarity measure (Chapter 2.2.3). In the next step the mean of all the points connected to a centroid is calculated and the mean point now acts as the new centroid. Using the new centroids further iterations of the same process is run till the centroid positions no longer changes or a stable centroid is reached. The total number of possible clusters in the data is K^n, hence the process is guaranteed to end. For the remainder of the paper, when we discuss K-means we will be referring to Lloyds algorithm. Lloyd's algorithm is shown in greater details below

Algorithm 2.1.1: K-means(X,K)

Input : data set $X = \{x_1, ..., x_n\} \in \mathbb{R}^d$, K = number of clusters.

Output: K cluster centroid $(w_1, ..., w_k) \in \mathbb{R}^d$ with each data points in X connected to its closest centroid.

Randomly initialize K vectors $(w_1, ..., w_k)$

repeat

Assign each $x \in X$ to $\text{argmin}_{w_j}\ dis(x, w_j)$ ▷ (where dis is the similarity measure, Definition 2.2.1)[2]

for j in 1:K **do**

$X_j = \{x \in X | x \text{ assigned to cluster } j\}$

$w_j = \frac{1}{|X_j|} \sum_{x \in X_j} x$ ▷ where $|X_j|$ is the number of points assigned to cluster j

end for

until no change in $w_1, ..., w_k$

return $\{w_1, ..., w_k\}$

[2] $argmin_x f(x)$ is the value of x for which $f(x)$ attains it's minimum.

Figure 2.1.1 illustrates few steps of a K-means process, with $K = 3$, run on a Iris-Sepal data[3]. The red solid dots represent cluster centroids and the hollow dots represent data points. The color of the hollow dots represent which cluster the data points belong to, that is data points connected to the same centroid have the same color and data points connected to different centroids have different colors. Figure 2.1.1 (a) shows the initial step with three randomly selected initial centroids. Each point in the data is assigned to its closest centroid. The output of the mean of all the red, green, and black data points respectively in Figure 2.1.1(a) act as the initial centroids in Figure 2.1.1(b). In (b) the data points are also reassigned to their new closest centroid, this can be observed by the change in the color of some of the hollow circles in the figure. The process of recalculating the mean and changing the centroids is continued for a number of iterations. Figure 2.1.1 (c) and (d) shows the 16th and 17th iteration respectively. Notice that when calculating the new centroid location between (c) and (d) the centroid location does not change, which in turn means that the points connected to the centroids remains constant. This is why (c) and (d) are exactly identical looking figures. When this event occurs we say that the K-means has converged to a solution; in this case the K-means converged after 16 iterations.

We now know how K-means works, but before moving on it is important to understand how effective K-means is, when does it work, and when does it not. A large number of sources discuss the great effectiveness of K-means in producing good clustering results for many practical applications [1,2,4,5,10,12]. Figure 2.1.2 compares the output a K-means with that of the original classifications in the data. The graph on the left shows the original

[3]The Iris-Sepal data is a 2 dimensional data with 150 observation and is known to have 3 known classes: virginica, versicolor, and setosa. This data will be used as the primary example to demonstrate the processes that we will be using in this paper. It was chosen because of its small data size which can be visualized quickly and because its a raw data with defined classes which can be used as a basis of comparison.
Source: Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems. Annals of Eugenics, 7, Part II, 179188.
The data were collected by Anderson, Edgar (1935). The irises of the Gaspe Peninsula, Bulletin of the American Iris Society, 59, 25.

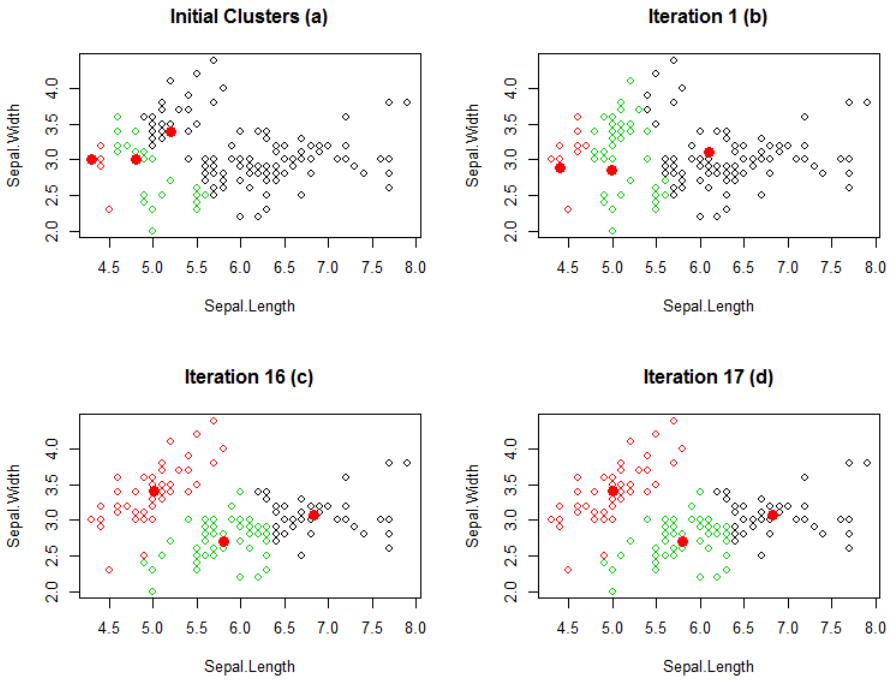

Figure 2.1.1. Different iteration steps of a K-mean algorithm on a Iris-Sepal data set

class separations in the data; this is based on the natural classes of the Iris-Sepal data. The graph on the right is the output of the K-mean algorithm run on the same data. For this purpose $K = 3$ since we know that the original data can be separated into 3 classes. Based solely on visual observation we can see that not all the points are clustered correctly, which is to be expected. What is important to note, however, is that a majority of the points are. So it is safe to say that K-means does cluster this data set reasonable well. We will provide a more quantitative measure of how well K-means performed using cluster evaluation techniques in Chapter 5.

At the end of this paper we will need to focus on validity of clusters formed by different methods and discuss which methods produce optimal results on average. Rendon, Abundez, Arizmendi and Quiroz discuss the importance of cluster validity index as way to determine the accuracy of clusters [27]. They point out that internal validation and

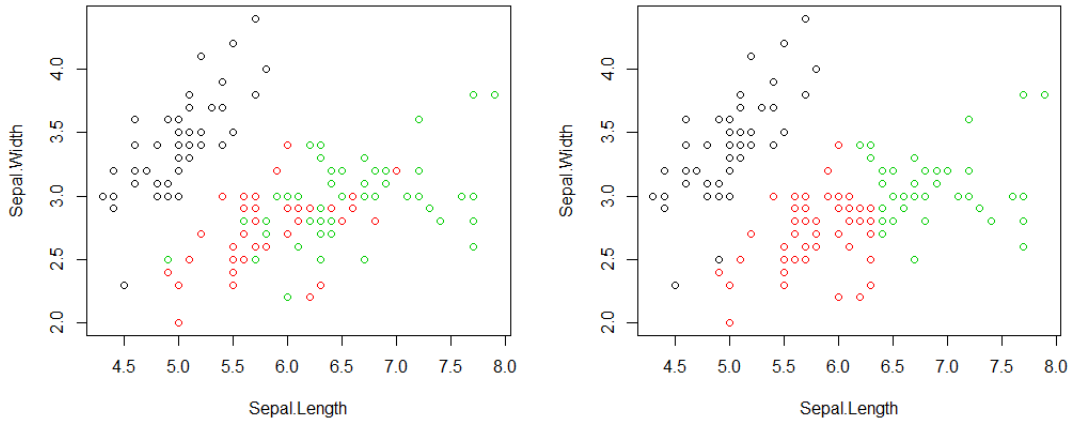

Figure 2.1.2. Comparing K-means clustering to the original classification in Iris-Sepal data set. Left Image: Clustering output of K-means. Right Image: Original Classification.

external validation are the two main methods of cluster validation. External validation is done by using previous knowledge about the data whereas internal validation is done using intrinsic information using the data alone. Their work focuses on a comparative study between different clustering indexes. Cluster validation index usually looks into the compactness and the separability of the clusters. Compactness is the measure of how strong the clustering is within a cluster and separability indicated how distinct two clusters are from each other.

2.2 Drawbacks

The original K-means algorithm is conceptually quite simple but it is still very effective that is why a 2007 survey paper found K-means to be one of the "most influential algorithms that have been widely used in the data mining community." (Wu 2) [1] This popularity is mainly due to its simplicity, adaptability and scalability [1]. However K-means does come with a few drawbacks that causes it to sometimes produce inaccurate and inconsistent results. These drawbacks include: the need to specify K at the start of

the algorithm, the choice of centroid locations, and the similarity measure used [7,9,13]. The K means algorithm is also very sensitive to outliers and tends to converge to local optimal [1]. K-means is also very poor at clustering few types of data.

2.2.1 Choosing the right K

The K-means algorithm requires an initial parameter input K. When clustering a data set the right number of K cluster to use is sometimes ambiguous. This is because a data set with n objects can be grouped together into any number of clusters between 1 and n. It is, however, very important to determine the correct value for K. This is illustrated in Figure 2.2.1, which contains four different clustering results on the Iris-Sepal data set using different values for K. We have already seen that when the correct value of K is used, the algorithm tends to perform reasonably well (Figure 2.1.2). Figure 2.2.1 shows that with different values of K as parameter, the resulting clusters produced can be vastly different. With an incorrect value of K, the algorithm tends to produce clusters that group together data points incorrectly or it separates data points, that should be grouped together, into different clusters.

A lot of work has been done to determine the correct value for K [1,10,13,14]. To determine the appropriate value for K for a given data set requires a lot of trial and error [13]. Pham, Dimov and Nguyen discuss the many different methods for deciding the value of K [13]. In general, the simplest way to determine the validity of a clustering result based on a given K is done visually. However, this method is hard to perform on multi-dimensinal data sets.

A more robust method is known as the Elbow Method [13,14]. In this method a plot of W_K versus the number of cluster K is used. Where

$$W_K = \sum_{j=1}^{K} \frac{1}{|X_j|} D_j.$$

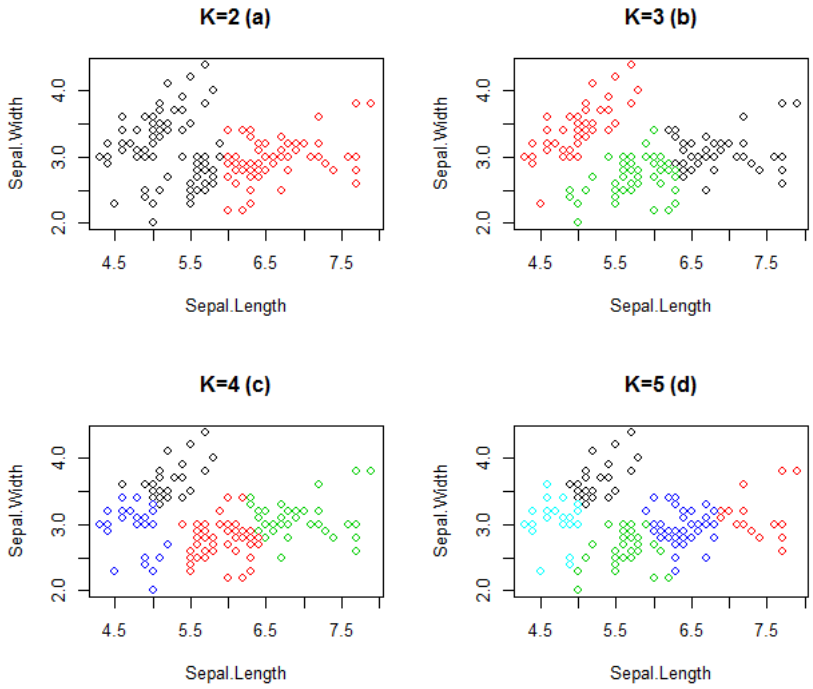

Figure 2.2.1. Output of four different K-means on Iris-Sepal data using $K = 2, 3, 4, 5$.

Such that

$$D_j = \sum_{i,i'} dis_{ii'}$$

is the sum of the pairwise distance between all the points in a cluster such that $dis_{ii'}$ is the distance between data points x_i and $x_{i'}$. We will illustrate the method used to calculate K from by using Figure 2.2.2 [4]. From the figure we observe that generally W_K decreases with increasing value of K. However, at one point the slope changes noticeably more than before. That is initially the slope of the graph is a large negative and at $K = 3$ onward the slope changes to a small negative. Based on this method the correct value for the number of clusters is 3. The Elbow method has the potential of being computationally heavily

[4]Figure 2.2.2 was not built by using actual data collected from any given data set. The figure was designed to illustrate the ideas proposed in [13,14]

especially for large data sets mainly because it requires K-means to be run multiple times with different values of K before it can suggest a suitable range of values for K.

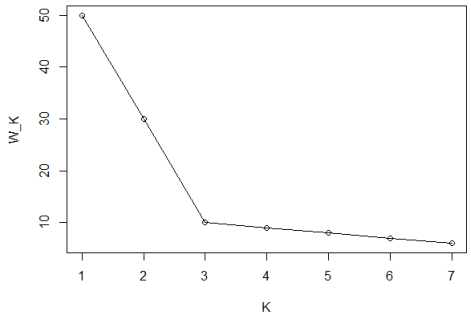

Figure 2.2.2. Diagram to illustrate the effect of change in K in the value of W_K.

2.2.2 Initial Centroids

In the basic K-means the initial centroids are chosen at random. K-means is an algorithm that tries to converge to stable centroids very quickly- with the least number of iterations possible. Due the greedy descent nature of the algorithm; it tends to force the centroids to converge onto local optima [17]. Since local optima may not necessarily produce the correct clusters we can deduce that the output of K-means is greatly affected by the initial centroid locations [17].

Figure 2.2.3 shows four different scatter plot outputs of K-means on the Iris-Sepal data set with different initial centroids (solid red dots) and $K = 3$. The different colors indicate the final stable clusters produced from the given starting centroids The four graphs all have different clustering. Figure 2.2.3 (a) and (b), are almost identical to one another even though the initial centroids locations are very different. Figure 2.2.3 (c) and (d) produces vastly different cluster outputs.

Tan, Steinbach and Kumar suggest that optimal clustering will be obtained given that two initial centroids fall anywhere in a pair of clusters, this is because the centroids will

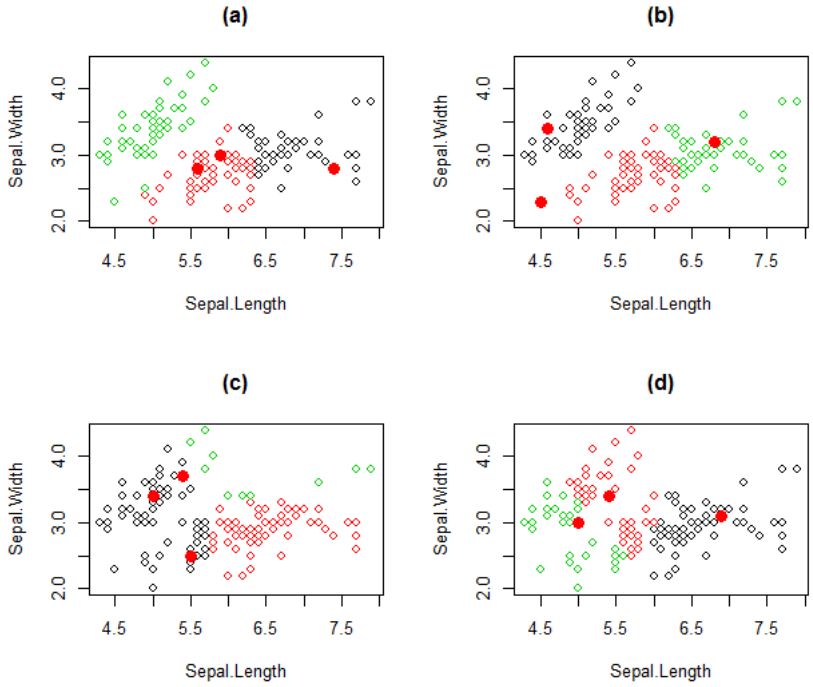

Figure 2.2.3. Output of four different K-means on Iris data with four different randomly selected starting centroids.

redistribute themselves; the results will be one centroid in each cluster [12]. This, however, may not necessarily be the case. This can be observed in figure 2.2.3 (a) and (d). The two figures have two initial centroids between a pair of clusters. The main difference between the two is that in (a) the pair of centroids start in center cluster whereas in (b) the pair start in the left cluster. This would suggest that the third centroid plays a crucial role. The figure indicates that it is easier for pair of centroids to move away from the third cluster than move closer to it. This explains why in (d) the two centroids starting at the left cluster could not move properly into the middle cluster whereas in (a) with the two centroids starting at the middle cluster had no problem moving into the left cluster. This notion is further reinforced when we look at (c) where the top centroids was forced to move up and right by the other two centroids.

Tan, Steinbach and Kumar discuss a few techniques to solve the initial centroid problem [12]. A way to solve the problem is to run K-means multiple times using different random initial centroids. Wu et. al. also suggests that the a simple modification to the K-means which involves running the algorithm multiple times using either different initial centroids or different values for k and using the best result as the solution of the K-means [1]. To determine the best result, for each set of clusters produced calculate the sum of squared errors (SSE)[5]. The set of centroids that produce the clusters with the minimum SSE is the one we should select. Tan, Steinbach and Kumar also point out that this method does not always work very well [12].

$$\text{SSE} = \sum_{j=1}^{K} \sum_{x \in X_j} dis(c_j, x)^2$$

where dis is the similarity measure between two points (defined in Section 2.2.3).

A different method includes selecting the first initial centroids as the mean of all the points in the data. After which each successive centroids are selected such that it is in the farthest point from all current initial centroids. This ensures that the user will obtain well separated initial centroids. However, with data set that contain large outliers, it will force the intial centroids to be placed on the outliers which will produce very unwanted results. Chawla and Gionis discuss the sensitivity of the K-means algorithm to outliers [8]. In their paper *A unified approach to clustering and outlier detection* they illustrate the effect of K-means output when large outliers are present. They show that by removing outliers they are able to obtain much tighter clusters.

Other modification to the K-means were made by Arthur and Vassilvitskii who worked on an algorithm called K-mean++. This modification to the K-means took the advantage of careful seeding of the cluster centroids [9]. According to Arthur and Vassilvitskii, 'it is the speed and simplicity of K-means method that makes it appealing, not its ac-

[5]SSE is a measure of the discrepancy between the data and an estimation model.

curacy'(Arthur 1). They managed to show that their algorithm 'generally outperforms k-means' (Arthur 2).

The k-means++ algorithm [9]

1. Take one center w_1, chosen uniformly at random from X. [6]
2. Take a new center w_i, choosing $x \in X$ with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$.
3. Repeat step 2 until there are k centers altogether.
4. Proceed as regular K-means.

2.2.3 Similarity Measure

K-means determines the points in a cluster by assigning points to their closest centroid based on some similarity measure. Similarity measure is the method used to determine how close or far two points are. This can be thought of as a basic distance function between two points. The most common similarity measure used in K-means is the Euclidean distance between the data points and the centroids. However, this does not have to be the only one. Su and Chow point out that different similarity measures produce different kinds of clusters and they are more successful with K-means depending on the specific type of data sets [4]. They point out that unless the user determines the appropriate similarity measure, no meaningful cluster analysis is possible. Data points in the euclidean space [7] for example can be clustered really well using Euclidean or Manhattan distance whereas cosine similarity and Jaccard measure can be employed to determine clusters in documents [12].

[6] Uniformly at random from set X means that choosing it so that each element X has the same probability of being chosen.

[7] A Euclidean space has some number of real-valued dimensions and dense points. There is a notion of average of two points. A Euclidean distance is based on the locations of points in such a space.
A Non-Euclidean distance is based on properties of points, but not their location in a space.

Definition 2.2.1. *Dis* is a similarity measure if it is a function from pairs of points x,y to reals such that:

1. $Dis(x, y) > 0$.
2. $Dis(x, y) = 0$ iff $x = y$.
3. $Dis(x, y) = Dis(y, x)$.
4. $Dis(x, y) < Dis(x, z) + d(z, y)$ [8]

Definition 2.2.2. The Manhattan distance between two points x_s and x_t with d dimensions is defined as:

$$Dis(x_s, x_t) = \sum_{i=1}^{d} |x_{si} - x_{ti}|$$

Definition 2.2.3. The Euclidean distance between two points x_s and x_t with d dimensions is defined as:

$$Dis(x_s, x_t) = \sqrt{\sum_{i=1}^{d} (x_{si} - x_{ti})^2}$$

Definition 2.2.4. "The Cosine Distance between two points is one minus the cosine of the included angle between points (treated as vectors). Given an $m - by - n$ data matrix X, which is treated as $m(1 - by - n)$ row vectors $x_1, x_1, ...x_m$, the cosine distance between the vector x_s and x_t are defined as follows" [9]:

$$\text{“}Dis(x_s, x_t) = 1 - \frac{x_s x_t'}{\sqrt{(x_s x_t')(x_t x_t')}}\text{”}$$

2.2.4 Different Types of Clusters

K-means has a few limitations when it comes to clustering data sets that are non convex [12]. It is also not very good at distinguishing clusters that greatly vary in size and density

[8] Ullman 13 [29]
[9] Definition 3.3 (Bora 2502) [15]

[12]. Figure 2.2.4 shows how a distance-based clustering algorithm like the K-means is not suitable for clustering a data set which has natural classes that form non-convex shapes. The set of black data points and the set of red data points each represent a different class. We can see that point A and C belong to the black class and the point B belongs to the red class. However the distance between point A and C is larger than that between A and B. Therefore a distance based clustering algorithm would assume that A and B are more likely to belong to the same cluster than A and C. This will of course lead to inaccurate clustering of this data set.

Figures 2.2.5 and 2.2.6 show the results of K-means on data sets with clusters of different sizes and densities respectively. In Figure 2.2.5, K-means is not able to distinguish between the two small clusters and the one large cluster properly. This is because the way K-means is designed forces it to usually obtain clusters that are of similar size. In this case we see that the when K-means tries to determine the clusters, it assigns a lot of points from the large cluster into the two smaller clusters. In Figure 2.2.6, we observe something similar. Here K-means cannot distinguish between the clusters of different density. The two more dense clusters get one centroid assigned for the pair whereas the less dense cluster contains two centroids in it.

However, it is worth noting that even with non convex clusters, or different density or sized clusters, K-means can often still produce good results as long as the clusters are distinct enough, that is the clusters are far apart. This can be easily shown by looking at data sets similar to the ones in Figure 2.2.4, 2.2.5 and 2.2.5 but with the different clusters being further apart.

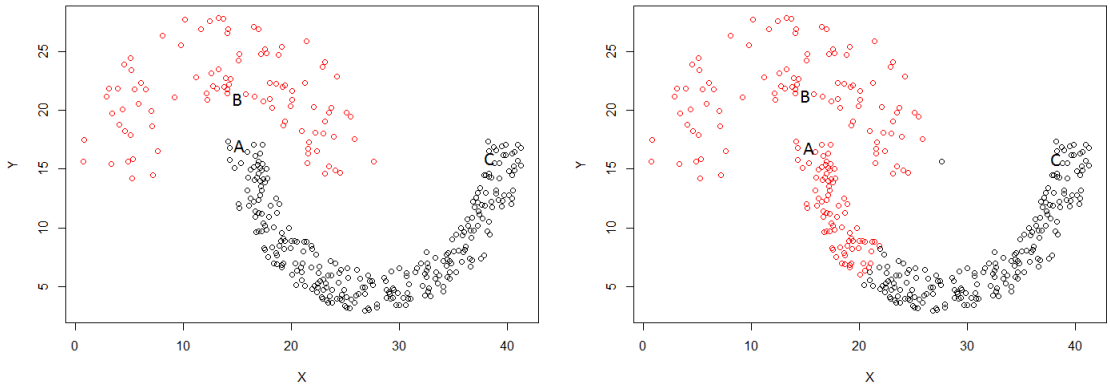

Figure 2.2.4. Non Convex Clusters. Left Image: Original classification, Right Image: K-means Clustering(K=2)

2.3 Summary

Clustering is very important in unsupervised classification of data. K-means is a centroid and distance-based clustering algorithm which produces K clusters, where K is an input parameter to the algorithm. It does so by randomly choosing initial centroids and connecting data points to a specific centroid such that it minimizes a chosen similarity measure. The centroids are then moved to the means of each cluster. This process of assigning points to centroids and changing centroid locations is continued till a point is reached where centroid locations no longer change. K-means is very useful in determining globular shaped and well separated clusters. However, the accuracy of K-means rely heavily on the value of K, choice of centroid location, and choice of similarity measure. K-means is also very sensitive to outliers.

The focus of the pre-processing algorithm proposed in this paper will show a variant way of choosing a more accurate range for the value of K and also as a means to choose the starting position of the centroid location. The initial centroid locations are made more

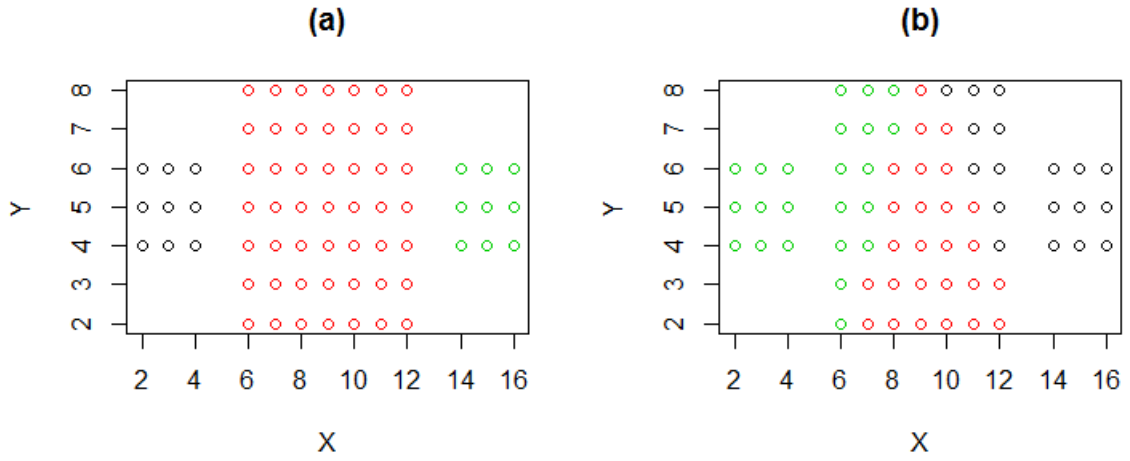

Figure 2.2.5. K-means with different cluster sizes, (a) Original Clustering, (b) K-means clustering with K=3

accurate by removing the influence of outliers in their determination. The distance measure used to perform all the processes proposed in this paper is the Euclidean distance.

Unlike the K-mean++, which focuses primarily on improving the speed of K-means, the algorithm we propose will (on average) improve the accuracy of clusters, and lower the number of iterations it takes for K-means to produce stable centroids. The latter is important when working on large data sets. We also show that by running the pre-processing algorithm on small samples of large data sets we are able to provide very reasonable values for K and for initial centroid locations.

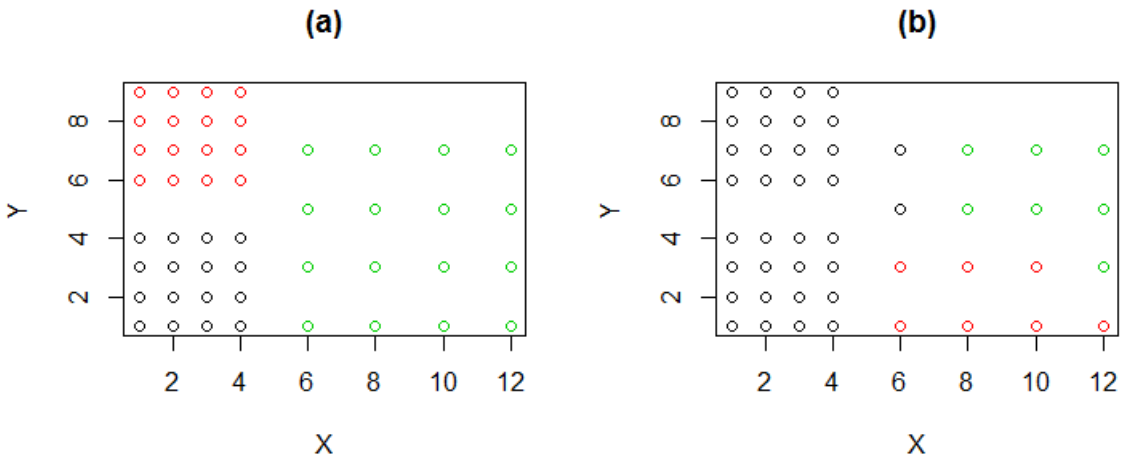

Figure 2.2.6. K-means with different cluster density, (a) Original Clustering, (b) K-means clustering with K=3

3
Developing the Model

3.1 Delaunay Triangulation

Epter, Krishnamoorthy, and Zaki used completely connected graphs[1] to pre-process the K-means in search for more optimal clustering for a given data set X [10]. However, the method they used is sensitive to noise and is computationally expensive due to the fact that a completely connected graph contains $n(n-1)/2$ number of edges, where n is the number of data points in X. In this paper we will propose the use of the edges created via a Delaunay triangulation of a data set X to be used instead. A Delaunay triangulation connects a point to all of its nearest neighbors. Delaunay triangulation allows us to quickly focus on a point and its relationship with other points near to it. The Euclidean distances captured in the edges of a Delaunay triangulation lets us weight[2] the edges so that we can determine how close (similar) or far (dissimilar) two points are. A Delaunay triangulation also connects the data set X with $3n-3-h$ edges (Definition 3.1.1); this is fewer than using a completely connected graph. The fewer number of edges will help us optimize our

[1] A complete graph is a graph in which every pair of vertices is connected by an edge
[2] In this paper we will use edge weights and edge lengths interchangeably

algorithm significantly, especially for very large data sets. For given data set X with n points, computing the Delaunay triangulation takes $O(n \log(n))$ time [3].

The theorems below are provided to help the reader recall relevant properties of a Delaunay triangulation. For a more detailed literature on Delaunay triangulation we encourage the reader to refer to *Topics in Design and Analysis of Algorithm: Delaunay Triangulation* by John Augustine [16].

Theorem 3.1.1. *Given a data set X with n number of points, which under a triangulation contains h points on the convex hull then:*

- *Number of triangular faces $m = 2n - 2 - h$*
- *Number of edges is $3n - 3 - k$* [4]

Theorem 3.1.2. *The Delaunay triangulation for a set X of points in a plane is a triangulation $DT(P)$ such that*

- *Three points form a triangle in $DT(X)$ iff the circumcircle*[5] *of the three points does not enclose any other points in X.*
- *"Two points form an edge in $DT(X)$ if there exists a circle though those two points that does not enclose any other point in X."* [6]

Theorem 3.1.3. *"The Delaunay graph is a plane graph"* [7]

[3] Theorem 13 pg 23 [16]
[4] Theorem 1[16] pg 1
[5] The circumcircle is a unique circle where the circle passes though all the points in a triangle
[6] Theorem 7 and 8[16] pg 11
[7] Theorem 6 [16] pg 7

3.2 Definitions

Definition 3.2.1. [Cluster] A cluster C of a data set X is a subset of X

Definition 3.2.2. [Clustering] A clustering Q of a data set X is a set of distinct clusters $\{C_1, C_2, ..., C_K\}$ such that $\cup_{i=1}^{k} C_i = X$ and $\cap_{i=1}^{k} C_i = \emptyset$

Definition 3.2.3. [Non Trivial Clustering] A non trivial clustering with K clusters of a data set X is such that $K \neq 1$, where all points are in one cluster, or $K \neq |X|$, where all points are their own cluster.

Definition 3.2.4. [Edge] "An edge is a four-tuple: $\{u, v, \Delta_{u,v}, t_Q\}$, where Q is a clustering, u and v are points in a data set, $\Delta_{u,v}$ is the edge weight or distance between u and v and":[8]

$$\text{“}t_Q = \begin{cases} \text{interior} & \text{if } u \text{ and } v \text{ belong to the same cluster in } Q \\ \text{exterior} & \text{otherwise} \end{cases} \text{„}$$

Definition 3.2.5. [Normal Delaunay Clusterable Data Set] A normal Delaunay clusterable data set is a data set X for which there exists a nontrivial clustering $Q = \{C_1, C_2, ..., C_K\}$ such that, if X is connected using a Delaunay triangulation, each point $x_i \in C_i$ will have edges $\{x_i, v, \Delta_{x_i,v}, \text{interior}\}$ and $\{x_i, w, \Delta_{x_i,w}, \text{exterior}\}$. Where $\Delta_{x_i,v} < \Delta_{x_i,w}$

Definition 3.2.6. [Perfect Delaunay Clusterable Data Set] A perfect Delaunay clusterable data set X is a normal Delaunay clusterable data set such that for all the edges $\{u, v, \Delta_{u,v}, \text{internal}\}$ and all edges $\{w, y, \Delta_{w,y}, \text{exterior}\}$ we have $\Delta_{u,v} < \Delta_{w,y}$.

Definition 3.2.7. [Mini Cluster] A mini cluster c in a cluster in X with clustering $Q = \{C_1, C_2, ..., C_K\}$. Such that the number of points $|c|$ is less a given threshold T and $|c| <<$

[8] Definition 3.2 pg 4 [10]

$\{|C_1|, |C_2|, ...|C_K|\}$. That is the number points in a mini cluster is significantly less than the number of points in any other cluster in the clustering of the data set.[9]

Definition 3.2.8. [Class] A class S of a data set X is a subset of X, where all $x \in S$ have the same predefined label.

Definition 3.2.9. [Classification] A classification H of a data set X is a set of distinct classes $\{S_1, S_2, ..., S_m\}$ such that $\cup_{i=1}^{m} S_i = X$ and $\cap_{i=1}^{m} S_i = \emptyset$

3.3 Framework

To obtain more accurate clustering using K-means, information is required about the data: the number of clusters K and location of the initial centroids w_j where $j = 1, ..., K$. However, such information about a data set is not always readily available. In this section we will develop a method to determine an optimal value for both K and the location of the initial centroids w_j.

Conceptually the algorithm we propose is similar to that proposed by Epter, Krishnamoorthy, and Zaki [10], where they use a complete graph to determine the number of clusters in a data set. We propose an alternative, where instead of looking at all the connected edges, we only consider the edges created via a Delaunay triangulation of the data set. By using Delaunay triangulation edges, we hope to be able to obtain greater information about the clusterability[10] and clustering in the data.

To understand this method let us first go back to our example of the Iris-Sepal data. Figure 3.3.1 shows the Iris-Sepal data connected with edges created via a Delaunay triangulation. From the figure we can observe that as predicted the triangulation tends to, for the most part, connect points that are fairly close together and few that are far apart.

[9] The threshold T is user defined and is dependent on the specific data set in question.
[10] Clusterability is the strength of the clustering structure of a given data set

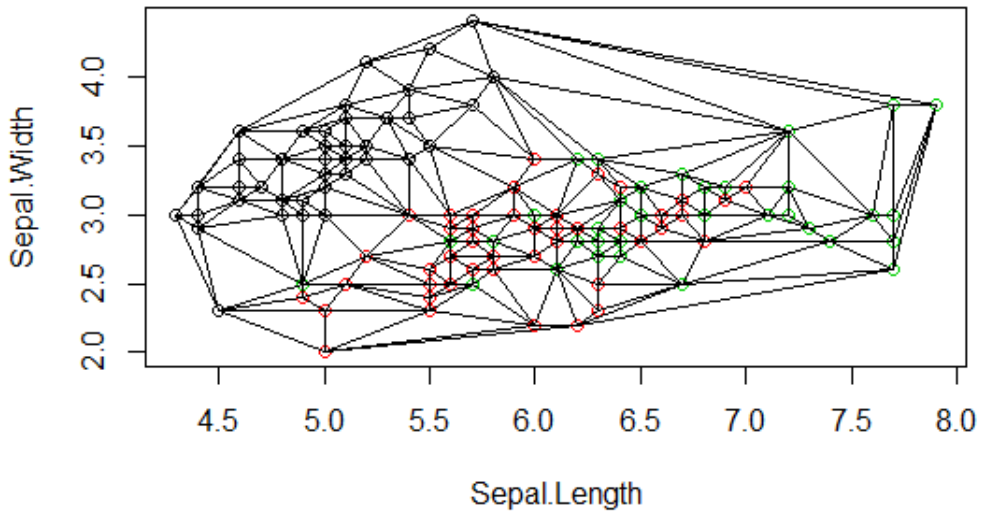

Figure 3.3.1. The Iris-Sepal data set with edges created via a Delaunay Triangulation.

From Figure 3.3.1 we can also notice that the short edges tend to connect data points that are in the same natural class and the long edges connect data points that are from different natural classes. Remember that clustering is an attempt to determine the natural classification in a data set. Therefore, this indicates that the distances between the data points in a Delaunay triangulation can be used to provide information about the clustering of the data.

Before we move to develop our algorithm using the Iris-Sepal data let us first try to develop how it would work with a more ideal data set, a data set that is perfect Delaunay clusterable. We know that if we consider any cluster $C_i \subset X$ such that X is a perfect Delaunay clusterable data set, the internal distance between any point in C_i must be less than the external distances, that is the longest internal edge in X is shorter than the shortest external edge. Therefore there exists a cut-off value g, such that any edge weight less than g is internal and any edge weight greater than or equal to g is external

[10]. Thus if we know the value of g and we remove all the edges with weights g and above we will be able to separate the data is different connected graphs. We will consider each of these separate connected graphs as a separate cluster. Figure 3.3.2 (a) shows an example of a perfect Delaunay clusterable data set. Figure 3.3.2 (b) shows what a Delaunay triangulation of the data set looks like. In this connected graph when we remove all edge lengths equal to greater than $g = 2.784833$, we obtain the result shown in (c). Figure 3.2.2 (c) shows the data set with two separate connected graphs. We will consider each of these connected graphs as a clusters.

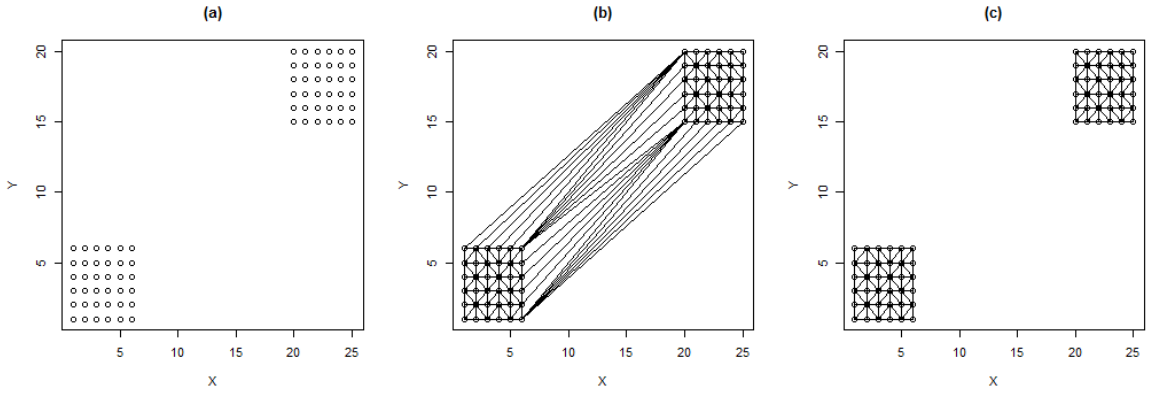

Figure 3.3.2. (a) Data set 2, (b) Data set 2 with edges connected via a Delaunay triangulation, (c) All edges equal to greater than $g = 2.784833$ removed.

Now that we have developed a foundation of the algorithm using an ideal data set, we can go back to the Iris-Sepal data in Figure 3.3.1. We will replicate the same process as we did with the perfect Delaunay clusterable data set. If we sort the edges in order of increasing weight in the Iris-Sepal data and we remove the top 30 percentile of the edges we obtain the graph shown in Figure 3.3.3. The figure shows the presence of 2 large clusters and 8 small clusters. This seems to indicate that just like the perfect Delaunay clusterable data we can get information about the clustering patterns of the Iris-Sepal data by removing the edges with largest weights. However, unlike in the perfect Delaunay clusterable data, in the Iris-Sepal data when we remove edges we also produce very small

clusters: we will refer to these as mini clusters. Mini clusters are clusters that are small compared to the overall data set or other clusters; single disconnected points are also considered to be mini clusters. Therefore, we can add this idea of mini clusters to our original definition of normal clusterable data set

Definition 3.3.1. [Normal Delaunay Clusterable Data Set (addition)] A normal Delaunay clusterable data set is a data set X for which there exists a non trivial clustering $Q* = \{C*_1, C*_2, ..., C*_K\}$ as defined in 3.3.5. In addition when mini clusters are removed from a normal Delaunay clusterable data set we obtain a perfect Delaunay clusterable data set (Definition 3.3.6).

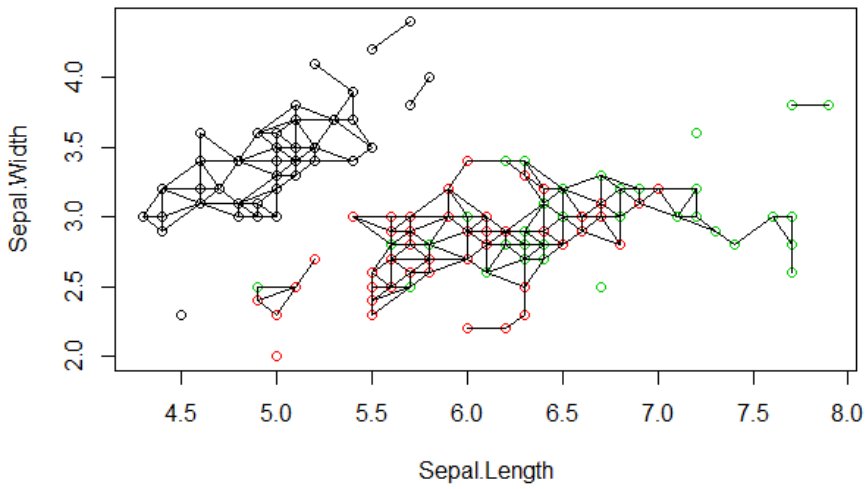

Figure 3.3.3. The Iris-Sepal data set with edges created via a Delaunay triangulation. The top 30 percentile of the edges with the largest weights were removed.

For a normal Delaunay clusterable data when we remove the edges connecting points based on their edge weight, we obtain large clusters along with a few mini clusters. Figure 3.2.4 shows the Iris-Sepal data with all edges above the 50th percentile, in terms of edge weight, removed. The figure contains three large clusters and a number of mini clusters.

The points are colored based on the natural classes in the data: Virginica, Verisicolor, and Setosa.

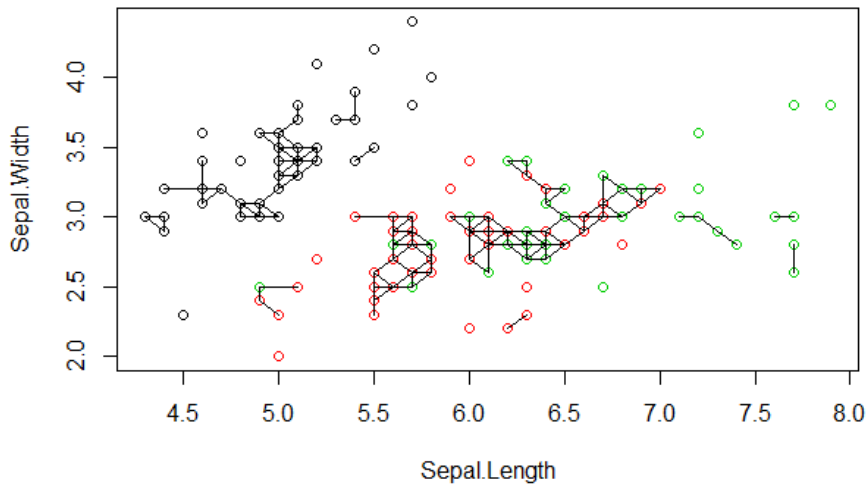

Figure 3.3.4. The Iris-Sepal data set with edges created via a Delaunay triangulation. The top 50 percentile of the edges with the largest weights were removed.

The formation of the large number of mini clusters will affect the result of the analysis. A mini cluster is formed when there exists a point $x \in C$ such that $\Delta_{x,x_j} >> \Delta_{x_i,x_j}$ for at least $|C| - T$ $x_i, x_j \in C$. Therefore we can consider these points to be outliers and remove from consideration in the rest of the process. We can do this without facing a large repercussion because our goal is to develop a method to obtain suitable value for K and the initial centroid locations. By removing the mini clusters we are able to obtain a more suitable and accurate value for K without much loss in our centroid location. This is because the centroid location will be dependent primarily on the denser region of the cluster, which is not affected when mini clusters are removed because mini clusters by definition are outside of the denser region of the cluster. Figure 3.3.5 contains the Iris-Sepal data with edges created via a Delaunay triangulation where edges and mini cluster

were removed such that the weight cut-off $g = 0.2$ and the mini cluster threshold $T = 4$. This means that all edges above 0.2 and all connected graphs of size 4 and below were removed. The connected graphs formed as a results seem to align very nicely with the natural class in the data, shown by the coloring of the points.

.

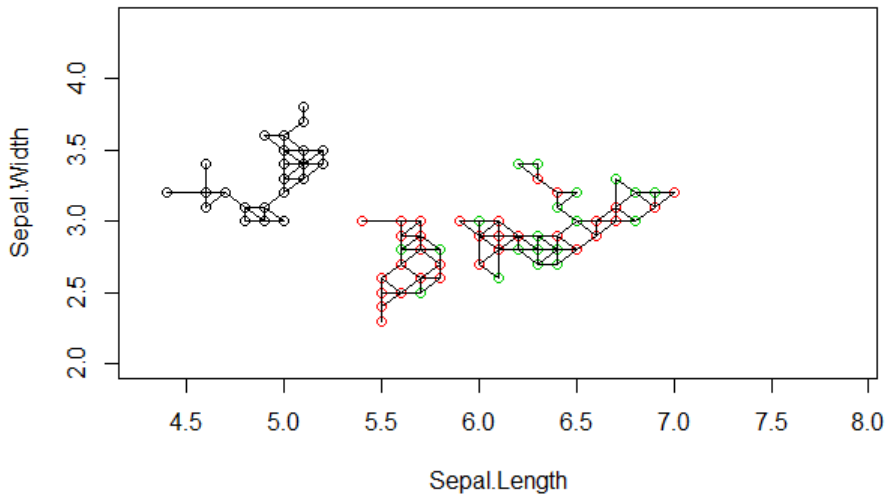

Figure 3.3.5. The Iris-Sepal data set with edges created via a Delaunay triangulation. The top 50 percentile of the edges with the largest weights were removed. The mini cluster threshold $T = 4$.

Definition 3.3.2. [Connected Graph] A graph is connected if given any pair of distinct vertices, there is a path (sequences of edges) between them

Once the correct number of edges connecting the points in a data set is removed. We can determine the points in each cluster by determining the points in a connected graphs. Connected components in a graph can be found using a breadth-first search algorithm [28]. The algorithm initiates by trying to find a path between two points. All points in the discovered in the path must belong to the same connected graph. Next we repeat the

process starting at an undiscovered point. This process is continued till all points have been discovered. Breath-first search algorithms are ideal for finding connected in a graph since they run in $O(n + m)$ time on undirected graphs with n vertices (points) and m edges [28].

By using the breath-first search algorithm on the output shown in Figure 3.3.5 we can correctly determine the value for the number of clusters $K = 3$. We can also assign each point to their respective clusters. Using the separated points we can obtain the location of the initial centroids to be used in K-means. From all points in each connected graph (cluster) C_j we can obtain a value for an initial cluster centroid w_j where $x_i \in C_j$ such that

$$w_j = \text{Min} \sum_{i=1}^{|C_j|} ||x_i - w_j||$$

or

$$w_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$$

At this point we have obtained the value of K and w_j where $j = 1, ..., K$. We can now use these to initiate a K-means algorithm and hopefully obtain better clustering results. For the remainder of the paper we will refer to the entire process explained in this chapter as the pre-processed K-means algorithm. The entire process explained in this chapter is shown in the algorithm below

Algorithm 3.3.1: Pre-processing Algorithm:

Input: Data set X, Threshold size of mini clusters T

Output: Number of clusters K and list of initial centroid locations $\{w_1, ...w_K\}$

G = a connected graph of X with edges created via a Delaunay Triangulation.

g = cut-off edge weight ▷ The cut-off edge weight is determined interactively using methods discussed in Chapter 4

$G = G$ | All edges equal to and above g removed

$G = G|$ All connected graphs have more than T data points in it.

K = number of connected graphs in $G = \{G_1, ..., G_K\}|G_i$ is a connected graph such that $|G_i| > T$

$w = \{\}$

for i in 1:K **do**

$w[i]$ = mean of all points in G_i $\triangleright$ $(w_i = w[i])$

end for

return K and w

3.4 Summary

In our algorithm we connect the data set with edges created via a Delaunay triangulation. After doing so we remove all edges of length greater than or equal to a cut-off g and all mini clusters of size equal to or less than a threshold T. This gives us K separate connected graphs where K is the number of clusters input parameter to be used when K-means is run. Also by taking the mean of each of the separate connected graph we obtain the initial centroid locations to be used in K-means.

For the algorithm to perform, the essential point left to consider is how to determine the cut-off distance g? For this we will start looking into the distribution of the edge weights $\Delta_{u,v}$ for all $u, v \in X$. This can also be thought of as the probability distribution for $P[\Delta_{u,v} < g]$ [10]. To accomplish this we will produce histograms of the edge weights, and by using the probability density information we obtain, we will try to determine an approximate for the cut-off g.

4
Determining the Cut off Weight

4.1 Initial Tests

A proposed solution to understanding when to end the process of disconnecting the Delaunay connected graph includes understanding the distribution of edge weights that the triangulation produces. To understand the data of edge weights we attempt to characterize them with known probability distribution. By doing a probability distribution fitting with the edge weight data, we hope to predict the probability and hence the frequency of the occurrence of different magnitudes of the edge weights in certain intervals. Therefore, by characterizing the edge data into a known probability distribution function we may be able to obtain more information about the data clustering. The parameters of the probability distribution function such as the mean and the variance may hold information that could be used to determine where the edge weight cut-off g could be.

Our attempts to characterize the edge weight data using known distributions does not however yield any conclusive result. We show that the edge weight data cannot be characterized by the known distributions used in this paper. This section contains our methods and results from trying to characterize the edge weight data. Readers who are mainly

interested in the actual working solutions to obtain the cut-off g can skip to the Section 4.2.

To determine which distribution the edge weight data comes from we need to first answer a few questions about the data: is the data necessarily discrete or continuous; and is the data symmetric or asymmetric;

Figure 4.1.1 shows a histogram of the edge weight data obtained from the Iris-Sepal data set. We can observe that the data distribution is right skewed, the larger weight values tend to be farther away from the mean than the smaller distance values. Based on this information alone we can start looking into known right skewed distributions such as geometric distributions, negative binomial distributions, exponential distributions, gamma distributions, log normal distribution, log logistic distribution, Gumbel distribution, and Weibull distribution [32,33,34].

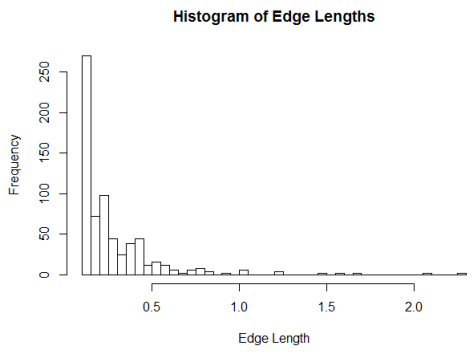

Figure 4.1.1. Histogram of Iris-Sepal data set with 30 bins

The edge weight between two points in a data can have any numeric value and can be meaningfully subdivided into finer and finer increments, depending upon the precision of the measurement system. This leads to the conclusion that the edge weights are obtained from a continuous distribution. Therefore from the list of distributions mentioned before, we focus on the ones that are continuous.

To determine if the data fit a specific probability distribution function the following tests were conducted with known distributions; histogram test, Normal Q-Q plot, and Chi-sq goodness of fit test. Multiple tests were run using edge weights collected from different data sets. The histogram of the edge weights in Figure 4.1.1 visually suggests that the edge weights are possibly obtained from a Gamma distribution or a Gumbel distribution. To further test this the quantile-quantile plot is drawn between the edge weight distribution data and the Gamma or Gumbel distribution. In a quantile-quantile plot, if the points fall roughly on the straight line, then we conclude that the data follow an approximate Gamma or Gumbel distribution.

When conducting these analyses we need to consider that raw data is almost never as well behaved as we would like it to be. Therefore fitting a statistical distribution to a data requires some compromises to be made during the process. During this analysis one key factor that must be maintained is the balance between a good distributional fit, while sustaining the underlying information provided by the original data.

Definition 4.1.1. [Gamma Distribution] "The Gamma function is defined , for $k > 0$, by

$$\Gamma(k) = \int_0^\infty x^{k-1} e^{-1} dx.$$

A continuous random variable X has the gamma distribution with parameters $k > 0$, $\theta > 0$ if its probability distribution function is given by

$$f(x) = \frac{1}{\theta^k \Gamma(k)} x^{k-1} e^{-x/\theta}, x \geq 0."^1$$

Definition 4.1.2. [Gumbel Distribution] "The Gumbel distribution is defined, for the scale parameter $\alpha > 0$ and the location parameter β. The Probability Density function

[1] (Chihara pg 382-385) [34]

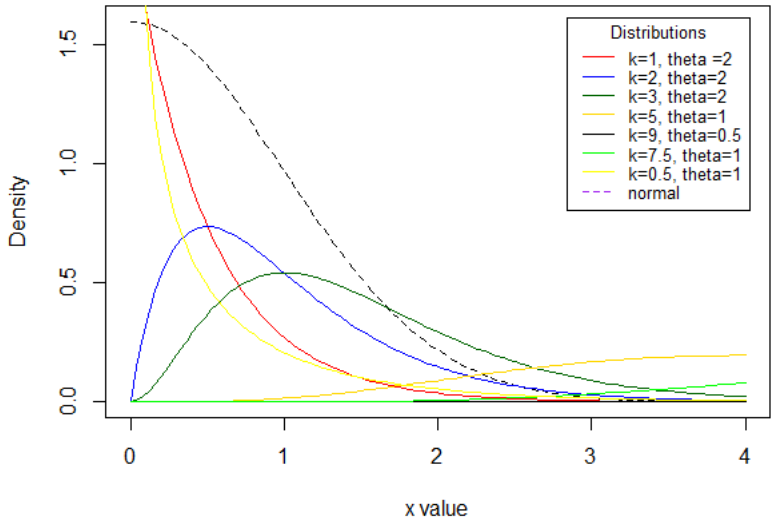

Figure 4.1.2. Gamma Distribution with different values of k and θ

for the Gumbel distribution is

$$f(x) = \frac{1}{\alpha} e^{-\frac{x-\beta}{\alpha} - e^{-\frac{x-\beta}{\alpha}}}.$$"[2]

Figure 4.1.4 and 4.1.5 shows the graphical results of our fit tests of the edge weight distribution with the Gamma and Gumbel distribution. From Figure 4.1.4, we can see that the empirical edge weight data matches the theoretical Gamma density and cumulative distribution function (CDF) very tightly. Figure 4.1.5 shows that this is also the case with Gumbel distribution. However, the quantile-quantile plot of the edge weight data does not seem to match very well with neither Gamma or Gumbel.

Therefore for more rigorous analysis on the fit we calculated the Cramer-von Mises statistic and the Anderson-Darling statistic [3] For the Gamma distribution we obtained a

[2] *e-Handbook of Statistical Methods*, NIST/SEMATECH 1.3.6.6.16 [33]

[3] Cramer-von Mises statistic is calculated using the algorithm of Csrgo and Faraway (1996) [20]
Anderson-Darling statistic is calculated using the algorithm of Marsaglia and Marsaglia (2004) [21]

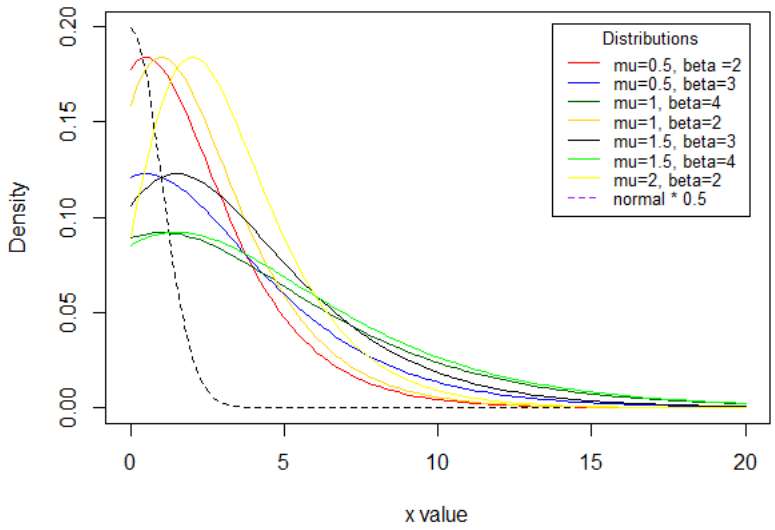

Figure 4.1.3. Gumbel Distribution with different values of α and β

Anderson-Darling statistic of 18.6579112 and a Cramer-von Mises statistic of 2.2976870. Both statistics are well outside of the range, therefore we can conclude with 95% confidence that the edge weight data does NOT fit a Gamma distribution. With the Gumbel distribution we obtained a Anderson-Darling statistic of 26.1057420 and a Cramer-von Mises statistic of 4.0950236. Again both of these are outside of the range and again we conclude with 95% confidence that the edge weight data does NOT fit the gumbel distribution.

4.2 Density

In the last section we attempted to fit a distribution to our edge lengths data and tried to see if the distribution of the list of edge weights as a whole could provide us with some information about the clustering in the data. This however, did not yield any fruitful result. In the work done by Epter, Krishnamoorthy, and Zaki, they use a method where they focus on the individual brackets of the histogram instead of the entire distribution of the edge weights [10]. In this section we will try and incorporate their idea into our model.

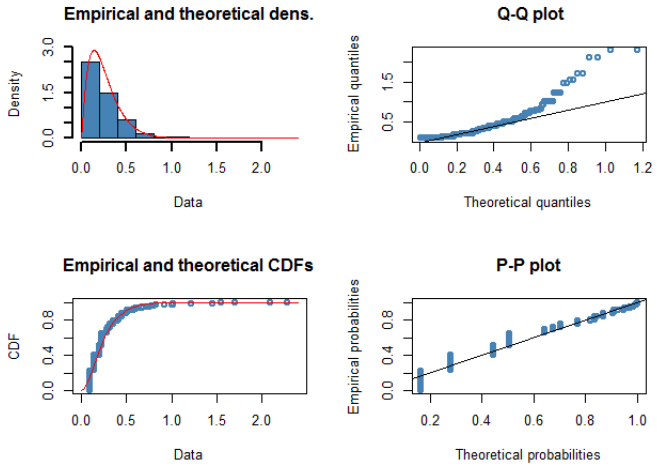

Figure 4.1.4. Exploratory analysis on fitting a Gamma distribution on the edge weight distribution for the Iris-Sepal data set

Epter, Krishnamoorthy and Zaki suggest that the distribution of edge weightss can be defined as the distribution of the probability that the distance between any pair of points is less than the cut-off g. To approximate the distribution we use a histogram[4] of our edge weight data in the hope that it will be able to provide us information about the clusterability of the data set. This is because, when looking at the histogram of a perfect Delaunay clusterable data set we should be able to spot two things, first a clearly defined initial spikes, and second an empty bucket or gap following the spikes.

In the edge weight data of a perfect Delaunay clusterable data set we expect to observe clearly defined spikes near the left (smaller) end of the histogram as well a few very small bumps at the right (larger) end of the histogram. This is because with clustered data the majority of the Delaunay edges, which generally connect a point to other points near it, will connect the points to other points in the same cluster which are smaller in magnitude, with only a few long edges connecting the points between clusters.

[4]The histogram is an approximation for the density function

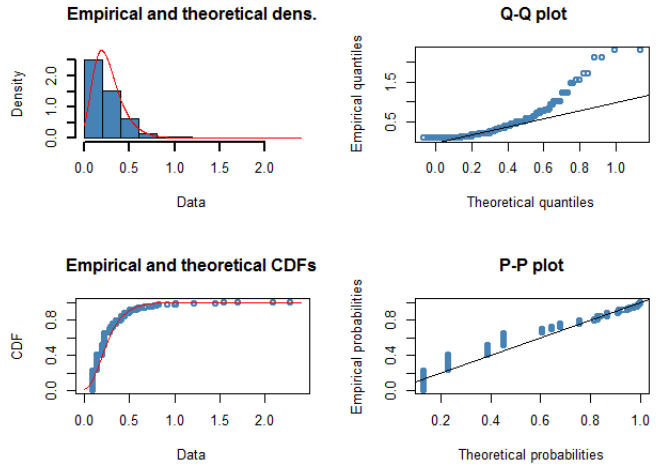

Figure 4.1.5. Exploratory analysis on fitting a Gumbel distribution on the edge weight distribution for the Iris-Sepal data set

We defined earlier that for perfect Delaunay clusterable data set the largest interior edge weight is smaller than the smallest exterior edge weight, and that the cut-off distance g lies in an empty bucket in between the largest interior edge weight and the smallest exterior edge weight. Therefore there exists a gap in the density function, that is there exists a bucket in between the largest interior weight and smallest exterior weight such that no edge weight falls in that range. Thus for perfect Delaunay clusterable data set we should be able to find an approximate value for g at a point where the first clearly defined spike reaches zero.

For improved results we will approximate the edge weight distribution using a Kernal density estimation instead of the histogram[5]. Kernal Density estimation has $O(M + N)$ complexity for observations with M Gaussians at N evaluation points [18]. The density function is determined by keeping track of the frequency of occurrence of all the edge weights. The full Kernal density estimation function is available in Algorithm 7.1.1.

[5]Kernal density estimation is done using the density.default function in R [22]

4.2.1 Results

This section contains example of data sets and their Kernal density estimations. This will help illustrate how the process of determining the clusterability of a data set can be performed by looking at the density estimations. Figure 4.2.1 contains a data set X where every point $(x, y) \in X$ such that $x, y \sim^{i.i.d}$ Unif$[0, 100]$ [6]. Figure 4.2.1 (a) shows the data set and (b) shows the Kernal density estimation (Algorithm 7.1.1) for the edge weights of the data set. Notice that the distribution of the edge lengths are almost completely in one large region between 0 and 14. This means that the data does not contain any definite spike. Also there are no actual gaps in the data for our g value [7]. Therefore we can say that this data set is not a perfect Delaunay clusterable data set.

Figure 4.2.2, 4.2.3 and 4.2.4 each contain a different type of perfect Delaunay clusterable data set. For each of these the Kernal density estimation (Algorithm 7.1.1) is performed and the results are shown in the figures below.

Figure 4.2.2 contains two uniform globular clusters of the same size and density[8]. Figure 4.2.2 (a) shows the data set with the points in the two classes colored differently. Figure 4.2.2 (b) shows the Kernal density estimation result. The figure indicates that there is sharp spike between 1 and 1.414214 and a small bumps between 16.643317 and 23.600847. Therefore the cut-off edge weight g lies in the gap such that $1.414214 < g < 16.643317$. We can select any value for g in that range, and the result of our analysis would be the same. We consider $g = 2$ for the remainder of the analysis, this is shown as the red vertical line in (b). When all edges with weight equal to or greater than $g = 2$ are removed from a Delaunay triangulation of this data set, we obtain the separate connected graphs shown in Figure 4.2.2 (c). Figure 4.2.2 (c) contains two connected graphs, clusters in this case.

[6] B.7 pg 381 [34]

[7] There does seem to be a small gap followed by a very small density of points around 16-20. However, this value is too small and is therefore not being considered.

[8] Essentially the type of cluster K-means is best at detecting.

Notice that these clusters are identical to the original classes in the data set shown in Figure 4.2.2 (a).

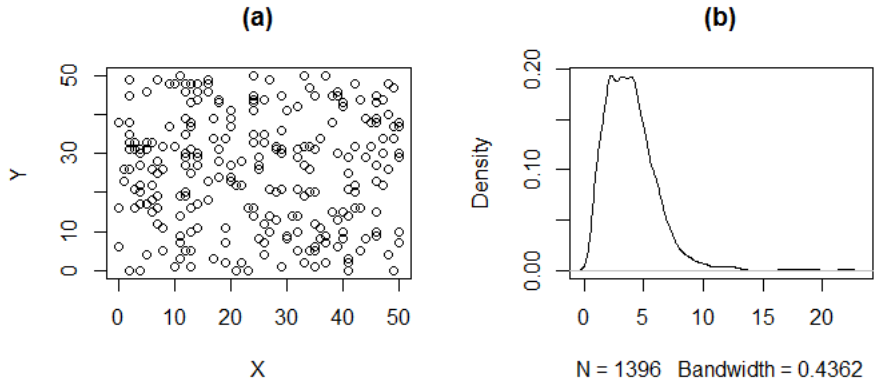

Figure 4.2.1. (a) Data set with coordinates whose values were assigned randomly, (b) Kernal Density Estimation of data set.

Figure 4.2.3 contains three uniform globular clusters of the different sizes but same density. Figure 4.2.3 (b) indicates that there are sharp spikes between 1 and 1.414214 and a small bumps between 2 and 4.472136. Therefore the cut-off edge weight g lies in the gap such that $1.414214 < g < 2$. Again we can select any value for g in that range, and the result of our analysis would be the same. We consider $g = 1.7$ for the remainder of the analysis, this is shown as the red vertical line in (b). When all edges with weights equal to or greater than $g = 1.7$ are removed from a Delaunay triangulation of this data set we obtain the separate connected graphs shown in Figure 4.2.3 (c). Figure 4.2.3 (c) contains three connected graphs, clusters in this case. And these clusters are identical to the original classes in the data shown in Figure 4.2.3 (a).

Figure 4.2.4 contains two uniform non convex clusters of the same sizes and density. Figure 4.2.4 (b) indicates that there are sharp spikes between 1 and 1.414214 and a small bumps between 2 and 9.848858. Therefore the cut-off edge weight g lies in the gap such that $1.414214 < g < 2$. Again we can select any value for g in that range, and the result

of our analysis would remain the same. We consider $g = 1.7$ for the remainder of the analysis, this is shown as the red vertical line in (b). When all edges with weights equal to or greater than $g = 1.7$ are removed from a Delaunay triangulation of this data set we obtain the separate connected graphs shown in Figure 4.2.4 (c). Figure 4.2.4 (c) contains two connected graphs, clusters in this case. And these clusters are identical to the original classes in the data shown in Figure 4.2.4 (a).

We saw that after determining an approximate for g we can separate the data set into different clusters correctly. Therefore we can obtain a very accurate value for K. We can also obtain very good initial centroid location value by taking the mean of all the points in each of the connected graphs or clusters receptively. Using the correct K and initial centroid locations we should now be able to obtain better results using K-means than before.

Inputting these results into a K-means may still not produce the best clustering overall. Recall, in Chapter 2.2.4 we showed that K-means is poor at clustering data sets that contain clusters that are non-convex or clusters that are of different sizes. This was demonstrated when we tried to cluster data sets similar to those in Figure 4.2.3 and 4.2.4 using K-means and failed (Figure 2.2.5 and Figure 2.2.4). This means that for these data sets even after obtaining the information from the pre-processing algorithm, K-means will still not be able to produce adequate clusters. However, we have shown by the correct removal of edge length that we were able to accurately separate the cluster and obtain connected graphs that represent the original classes in the data even for data sets that contain non-convex clusters or clusters that are of different sizes. This idea will be explored further in Chapter 6, where we will use it to create a new clustering algorithm.

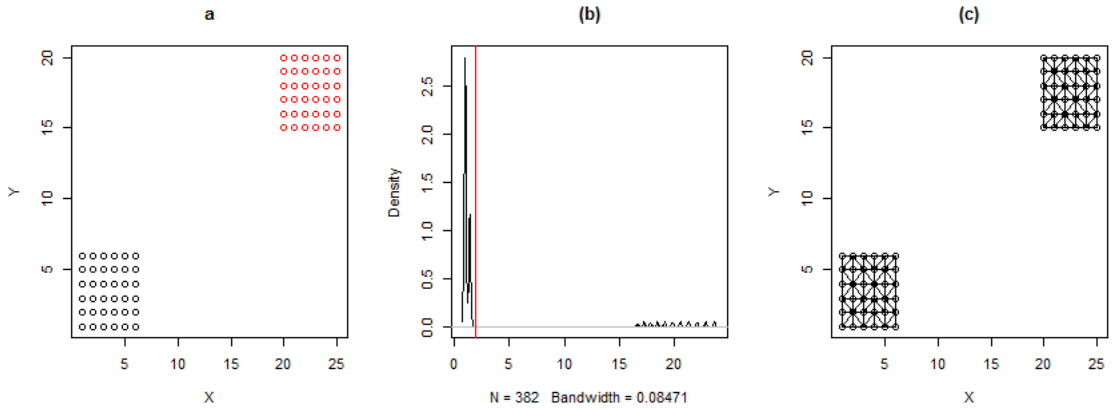

Figure 4.2.2. (a) Natural Classifications of Data set 2, (b) Kernal Density Estimation of Data set 2 with vertical red line at $g = 2$, (c) Remaining graph after all edges with weight greater than or equal to $g = 2$ removed.

4.3 Cluster Weight

We saw in the previous section that using Density estimation analysis we can deduce the clustering structure of a perfect Delaunay clusterable data set. However, this method does not apply to the large number of data sets that do not fall under the constraints of our definition of the perfect Delaunay clusterable data set. Data sets such as the Iris-Sepal data, which we assume to be a normal Delaunay clusterable data set, gives us the following result shown in Figure 4.3.1. The Kernal density estimation in Figure 4.3.1 (b) is similar to that in Figure 4.2.1 (b). There are no defined spikes before any of the gaps in the density function, this means that we cannot correctly estimate the cut-off weight g using this method. We can check the accuracy of g by still going through the steps of our analysis. The vertical red line in Figure 4.3.1 (b) shows the estimate of g we would get based on the density model. Figure 4.3.1 (c) shows the resultant connected graph obtained if all edges with edge weights greater than or equal to $g = 1.1$ are removed. The resultant connected graph is very different from the original classification in Iris-Sepal data (Figure 4.3.1 (a)).

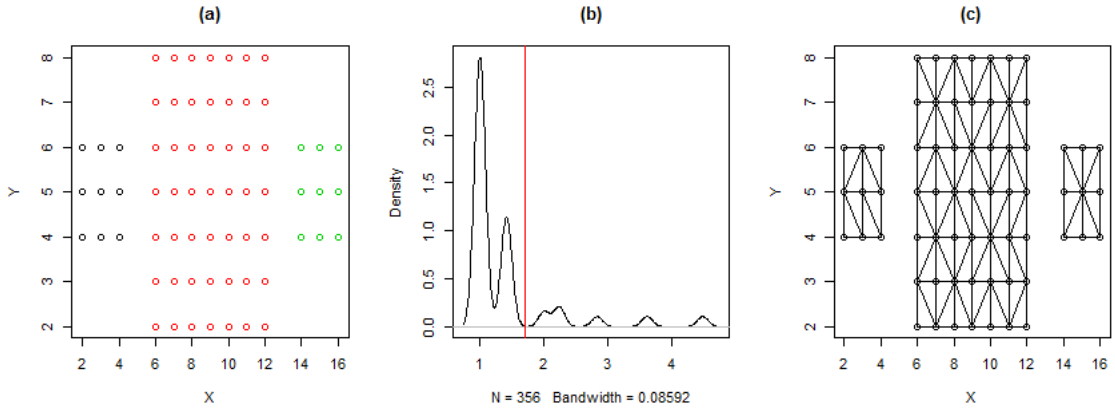

Figure 4.2.3. (a) Natural Classifications of Data set 4, (b) Kernal Density Estimation of Data set 4 with vertical red line at $g = 1.7$, (c) Remaining graph after all edges with weight greater than or equal to $g = 1.7$ removed.

Therefore the density model does not allow us to properly pre-process normal Delaunay clusterable data sets.

In this section we will develop a new model to determine the cut-off weight g. This model is developed with the hope that it will be able to determine the correct g for normal Delaunay clusterable data sets. For this model we will use a modified version of the intra cluster dispersion W_K from chapter 2.2.1 [13,14].

In this method a plot of W_K versus the value of g for a given data set X is used. Where

$$W_K = \sum_{j=1}^{K} \frac{1}{|C_j|} D_j.$$

Such that D_j is the sum of all the edge weights in the connected graph or cluster X_j such that $j = 1, ..., K$ and $|C_j|$ is the number data points in the connected graph or cluster C_j.

In the plot of W_K versus g we will observe the change in W_K as we decrease the value of g. To plot our graph we start by assigning the value of $g = \text{Max}(\Delta_{u,v})$, where $\Delta_{u,v}$ is the edge weight of the edges between all the points $u, v \in X$ (see Definition 3.2.4). In each of n iteration we decrease the value of g by div and calculate the new W_K, where

$$div = \frac{\text{Max}(\Delta_{u,v}) - \text{Min}(\Delta_{u,v})}{n+1}.$$

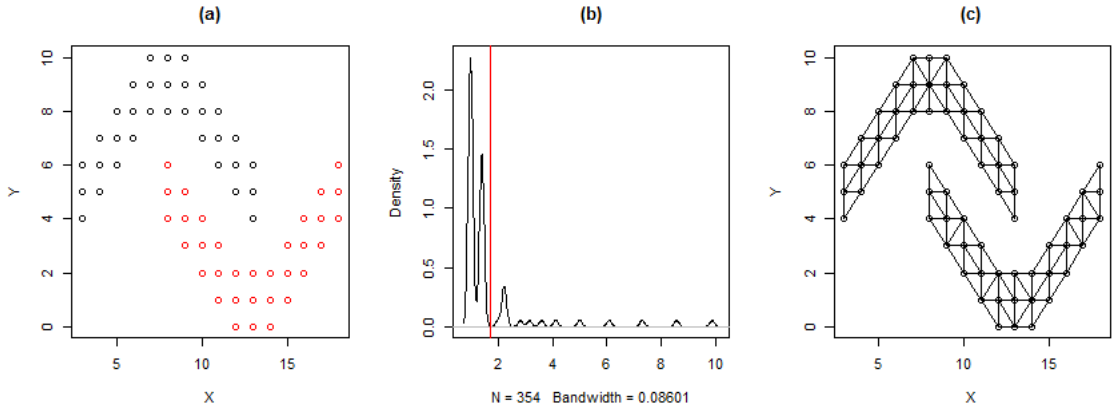

Figure 4.2.4. Natural Classifications of Data set 5, (b) Kernal Density Estimation of Data set 5 with vertical red line at $g = 1.7$, (c) Remaining graph after all edges with weight greater than or equal to $g = 1.7$ removed.

The cut-off g extracted from the model is the one that causes the value of W_K to increase the most.

The idea behind our model is that for a data sets where all edges $\{u, v, \Delta_{u,v}, \text{interior}\}$ and all edges $\{w, z, \Delta_{w,z}, \text{exterior}\}$, $\Delta_{u,v} \ll \Delta_{w,z}$, W_K increases when the removal of edges breaks down a cluster to two or more smaller clusters. Also W_K increases the most when the new clusters formed from the removal of an edge contain equal number of data points. That is, when a cluster C breaks into two clusters C_1 and C_2, the change in W_K is proportional to $|C_1|$ - $|C_2|$, where $|C_1|$ and $|C_2|$ are the number of points in clusters C_1 and C_2 respectively. The smaller $|C_1|$ - $|C_2|$ is, the greater the increase in W_K.

Theorem 4.3.1. *If the removal of an edge e causes a connected graph C to separate into two connected graphs C_1 and C_2, where $|C_1| - |C_2| = p$, W_K increases as $|p|$ decreases. So W_K increases the most when $|C_1| = |C_2|$. Assuming:*

- *The edge weights are a continuous random variable.*
- *For all edges $\{u, v, \Delta_{u,v}, interior\}$ and all edges $\{w, z, \Delta_{w,z}, exterior\}$, $\Delta_{u,v} \ll \Delta_{w,z}$.*
- *Data points in a cluster are uniformly distributed.*

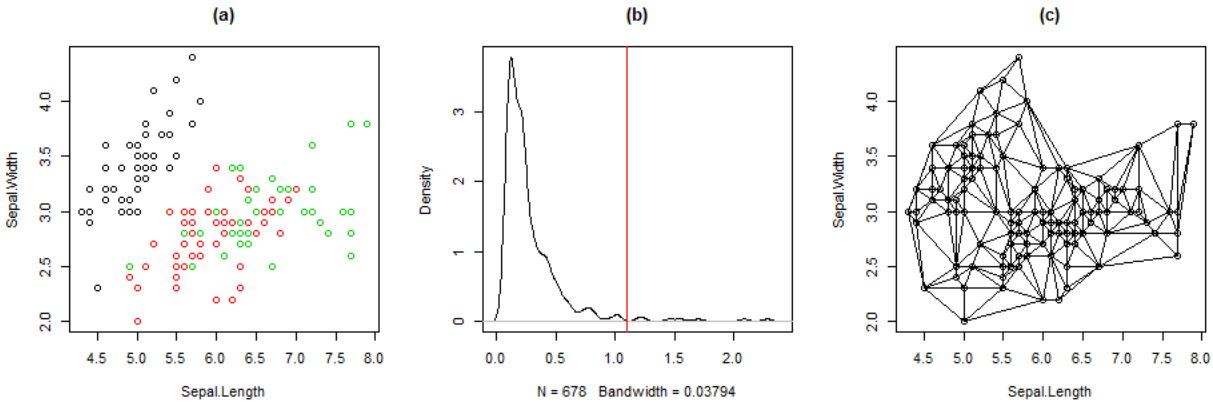

Figure 4.3.1. (a) Natural Classifications of Data set 1 (Iris-Sepal), (b) Kernal Density Estimation of Data set 4 with vertical red line at $g = 1.1$, (c) Remaining graph after all edges with weight greater than or equal to $g = 1.1$ removed.

- *The intra-cluster edge weights are all approximately equal.*
- *The value of e is constant.*

Proof. Let E_j be the sum of edge lengths in a cluster j, N_j be the number of points in cluster j, K be an integer where K is the total number of clusters, and W_K is the total cluster weight such that

$$W_K = \sum_{j=1}^{K} \frac{1}{N_j} E_j.$$

We assume that the edge lengths are a continuous random variable, therefore the probability of any two edge lengths $\Delta_{u,v}$ and $\Delta_{w,z}$ equaling each other is $P[\Delta_{u,v} = \Delta_{w,z}] = 0$. Hence it is always possible to decrease the value of the cut-off g such that only one cluster C_X breaks, and it breaks into exactly two clusters C_a and C_b.

Let us also assume that after the cluster breaks, the data points in each cluster are uniformly distributed such that all edges $\{w, z, \Delta_{w,z}, \text{interior}\}$ where $w, z \in C_a$ and $\{u, v, \Delta_{u,v}, \text{interior}\}$ where $u, v \in C_b$, have lengths such that $\Delta_{w,z} \approx \Delta_{u,v}$. Let us also assume that two clusters with same number of points N_j have the same edge sum E_j.

Therefore when one cluster C_X breaks down into two clusters C_a and C_b , W_K changes from $\frac{1}{N_X}E_X$ to $\frac{1}{N_a}E_a + \frac{1}{N_b}E_b$. Then $N_X = N_a + N_b$, and since the new clusters are formed by the removal of an edge $E_X > E_a + E_b$, such that $E_X - e = E_a + E_b$.

Also let c and d be integers where $d < c$ such that $N_a = \frac{c-d}{c}N_X$ and $N_b = \frac{d}{c}N_X$ where $\frac{c}{N_X} \leq d \leq \frac{1}{2}c$, this is because if $d = \frac{c}{N_X}$ then $N_b = 1$ and if $d = \frac{1}{2}c$ then $N_b = \frac{N_x}{2}$. Therefore we can rewrite W_K for the two clusters as

$$\begin{aligned} W_K &= \frac{1}{N_a}E_a + \frac{1}{N_b}E_b \\ &= \frac{\frac{d}{c}N_X E_a + \frac{c-d}{c}N_X E_b}{(\frac{c-d}{c}N_x)(\frac{d}{c}N_X)} \\ &= \frac{c(dE_a + (c-d)E_b)}{(c-d)N_x d} \\ &= \frac{c(cE_b + d(E_a - E_b))}{(c-d)N_X d} \end{aligned}$$

Case 1: Let $d = \frac{c}{N_X}$ then $N_b = 1$, then $N_a = N_X - 1$. When $N_b = 1$, then $\hat{E}_b = 0$ this is because a single point cannot have any connected edges. Therefore

$$\begin{aligned} \hat{W_K} &= \frac{1}{N_a}\hat{E_a} + \frac{1}{N_b}\hat{E_b} \\ &= \frac{\hat{E_a}}{N_X - 1} \end{aligned}$$

Case 2: Let $d = \frac{1}{2}c$ then $N_b = \frac{N_X}{2}$ and $N_a = N_X - N_b = \frac{N_X}{2}$. Then

$$\begin{aligned} \tilde{W_K} &= \frac{1}{N_a}\tilde{E_a} + \frac{1}{N_b}\tilde{E_b} \\ &= \frac{(N_b)(\tilde{E_a}) + (N_a)(\tilde{E_b})}{(N_a)(N_b)} \\ &= \frac{(\frac{N_X}{2})(\tilde{E_a}) + (\frac{N_X}{2})(\tilde{E_b})}{(\frac{N_X}{2})(\frac{N_X}{2})} \\ &= \frac{2(\tilde{E_a} + \tilde{E_b})}{N_X} \end{aligned}$$

If the clustering in case 1 and case 2 were both formed by the removal of an edge of length e then $\hat{E_a} = \tilde{E_a} + \tilde{E_b}$. Then $\tilde{W_K} > \hat{W_K}$ when

$$\frac{2(\tilde{E_a} + \tilde{E_b})}{N_X} > \frac{\hat{E_a}}{N_X - 1}$$

$$\frac{2\hat{E_a}}{N_X} > \frac{\hat{E_a}}{N_X - 1}$$

$$2\hat{E_a}(N_X - 1) > \hat{E_a}N_X$$

$$2(N_X - 1) > N_X$$

$$N_X > 2$$

Therefore for all cluster with more than 2 data points $\tilde{W_K} > \hat{W_K}$. We have taken care of the two extremes of d. Now let us observe what happens to W_K as d goes from $\frac{c}{N_X}$ to $\frac{1}{2}c$ where

$$W_K = \frac{c(cE_b + d(E_a - E_b))}{(c-d)N_X d}$$

$$\frac{\delta W_K}{\delta d} = \frac{c((E_a - E_b)d^2 - E_b c(c - 2d))}{N_X d^2 (c-d)^2}$$

As d increases from $\frac{c}{N_X}$ to $\frac{1}{2}c$ then d^2 is increasing and $c - 2d$ is decreasing till becoming zero at $d = \frac{c}{2}$ also $d^2(c-d)^2$ is decreasing since $c > d$. Therefore W_K is an increasing function with respect to d as d goes from $\frac{c}{N_X}$ to $\frac{1}{2}c$.

Thus we can conclude that when a cluster C breaks into two clusters C_1 and C_2, W_K will increase more the closer the clusters are in terms of number of points.

□

This is an acceptable outcome because the clusters in K-means are usually of similar size. Therefore the output of our pre-processing algorithm should also be focusing on clusters that are of similar size. Also when the removal of an edge produces one really big cluster and one really small clusters, we consider the small clusters to be mini clusters and

therefore an outliers to the main clustering of the data. So essentially we do not consider that as the formation of a new cluster at all, but rather the removal of outliers from a cluster.

Figure 4.3.2 shows the implementation of this method on the Iris-Sepal data set. Figure 4.3.2 (b) shows the 150^{th} to 200^{th} iteration of the process, where the total number of iterations is equal to 200. The vertical red line indicates that the 192^{nd} iteration is where W_K increases the most. The two blue horizontal lines show the change in W_K from 2.0592834 to 4.3783762. Figure 4.3.2 (c) shows what what that iteration means in terms of number of clusters. From (c) we can see that the change in W_K occurred when the number of clusters in the data set increased from 2 to 3. At the 192^{nd} iteration we have $g = 0.1976277$. From this value of g we get the resultant separate connected graphs shown in Figure 4.3.2 (d). The result indicates that the algorithm was able to correctly determine the number of clusters as $K = 3$ for K-means. Also the clustering of the connected graphs in (d) seem to very accurately represent the denser regions of the three natural classes of the Iris-Sepal data set as shown in (a).

Figure 4.3.3 shows another implementation of this method using the Lsun data set (Data set 6). Figure 4.3.3 (b) shows the 31^{st} to 45^{th} iteration of the process, where the total number of iterations is equal to 50. The vertical green line and red line indicated at the 43^{rd} and 45^{th} iteration respectively represent the location where W_K increases the most. The value of the cut-off edge weight for these iterations are $g = 0.70677042$ or $g = 0.55193255$ respectively. In cases where the largest increase in W_K is similar for two different values of g we will generally choose the smaller value of g. This is because the smaller value of g will be the one that separates a connected graph into greater number of connected graphs. That is the smaller value of g will generally give us the larger value for K. When in doubt between choosing the right value for K, we prefer the larger value of K over the smaller. This is because if the K we obtain is less than the number of natural

classes in the data, we will end up combining two distinct classes and treating them as one cluster. Whereas, if K is larger than the number of natural classes in the data, we will end up breaking a natural class into two clusters. This outcome is still favorable because each of smaller clusters is still pure in the sense that it contains only points from one natural class [12]. Therefore we take the cut-off $g = 0.55193255$ at the 45^{th} iteration. We see from Figure 4.3.3 (c) that this gives us 3 cluster. Using $g = 0.55193255$ we obtain the separate connected graphs shown in Figure 4.3.3 (d). The result indicates that the algorithm was able to correctly determine the number of clusters as $K = 3$ for K-means. Also the clustering of the connected graphs in (d) seem to exactly represent the three natural classes of the LSun data set as shown in (a).

Figure 4.3.4 and 4.3.5 illustrates how this model can work for other data sets as well. Notice again that by using this method we are able to very accurately separate even non-convex clusters and clusters of different sizes. We will build on this idea in Chapter 6

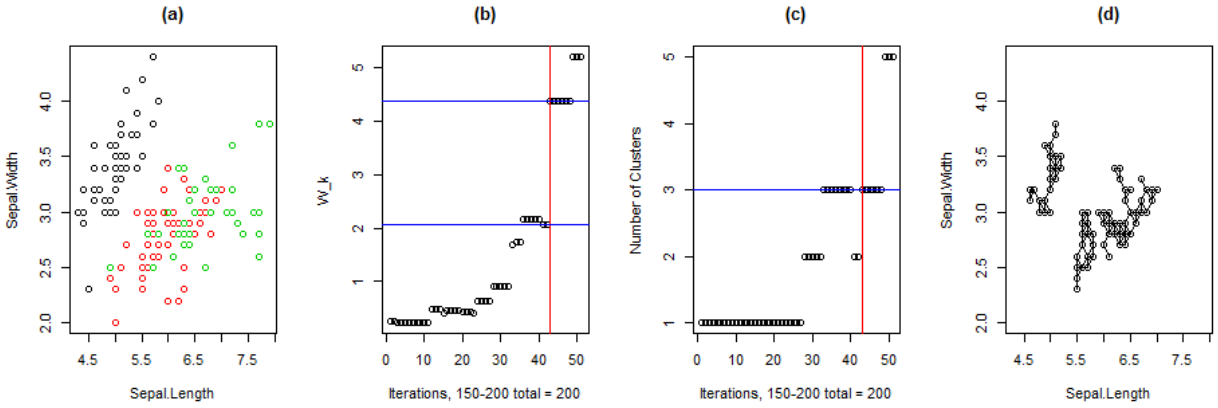

Figure 4.3.2. (a)Natural Classifications of Data set 1 (Iris-Sepal), (b) W_k for each iteration with red vertical line at 192^{nd} iteration, (c) Number of cluster at each iteration, (d) Remaining graph after all edges g for the 192^{nd} iteration is used.

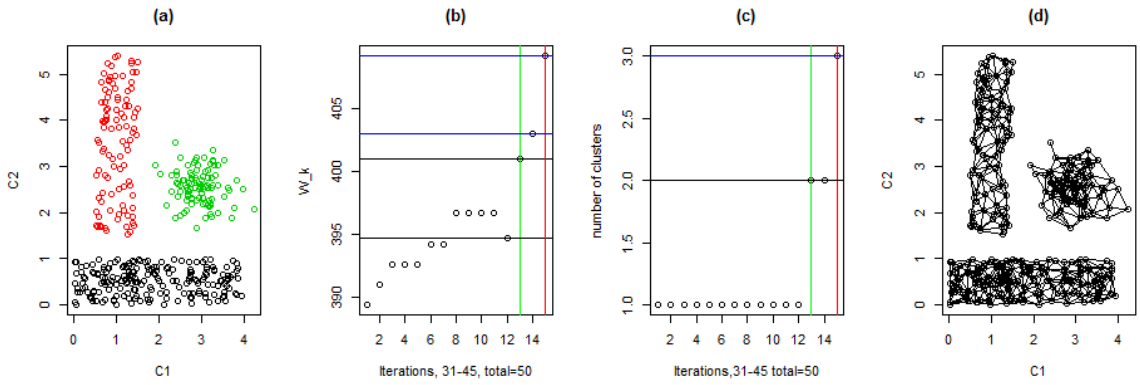

Figure 4.3.3. (a)Natural Classifications of Data set 6, (b) W_k for each iteration with red vertical line at 45^{rd} iteration, (c) Number of cluster at each iteration, (d) Remaining graph after all edges g for the 45^{rd} iteration is used.

4.4 Summary

The focus of this chapter was to determine a method to obtain a suitable value for the cut-off distance g. We showed a density estimation method by which we can interactively determine the value of g and a W_K method by which g is obtained solely via the algorithm.

In the first section we attempted to use known distributions to explain the distribution of edge lengths. This however, did not yield any applicable results. In the second section we moved on to estimate the distribution of the edge lengths using Kernal density estimation. Using the density estimation, we showed that for perfect Delaunay clusterable data we are able to approximate g. We showed that g exists in the gap between the initial (small end) large spikes in the data (interior edges) and the small spikes near the end of the data (exterior edges).

We also showed that the density estimation cannot be used to approximate g for normal Delaunay clusterable data sets. In the third section we proposed alternative method that can be used for normal Delaunay clusterable data sets. This method calculates the cluster weights W_K for a given data set in the form of a Delaunay connected graph. We notice that the value of W_K changes when edges are removed, that is when the value of g is

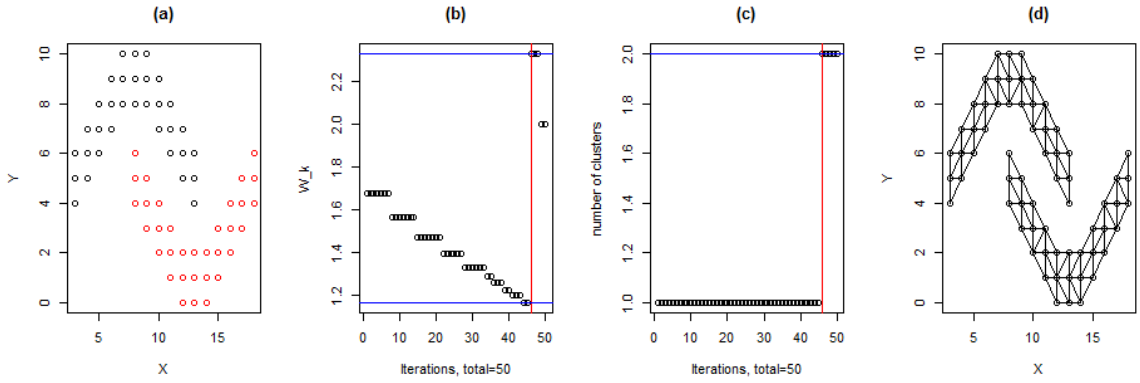

Figure 4.3.4. (a)Natural Classifications of Data set 5, (b) W_k for each iteration with red vertical line at 46^{th} iteration, (c) Number of cluster at each iteration, (d) Remaining graph after all edges g for the 46^{th} iteration is used.

changes. We also show that W_K increases most when the removal of an edge causes the a connected graph C in the data set to split into two separate connected graphs C_1 and C_2 such that $|C_1| - |C_2| = 0$. That is, when a connected graph splits into two separate connected graphs, the less the difference in the number of points between each connected graph the greater the increase in W_K (Theorem 4.3.1). This is an ideal solution because in this model, we are mainly interested in clusters of similar size. Therefore in this model g is approximated by equaling it to the edge weight that cases W_K to increase the greatest. This entire process is fully automated and requires no human interaction.

By obtaining a suitable method for approximating g we are now able to fully pre-process the K-means based on the model explained in chapter 3. In the next section we will see how the output of the pre-processed K-means performs against that of a regular K-means. For this we will use rigorous external evaluation techniques such as the Purity test, F-test and Entropy.

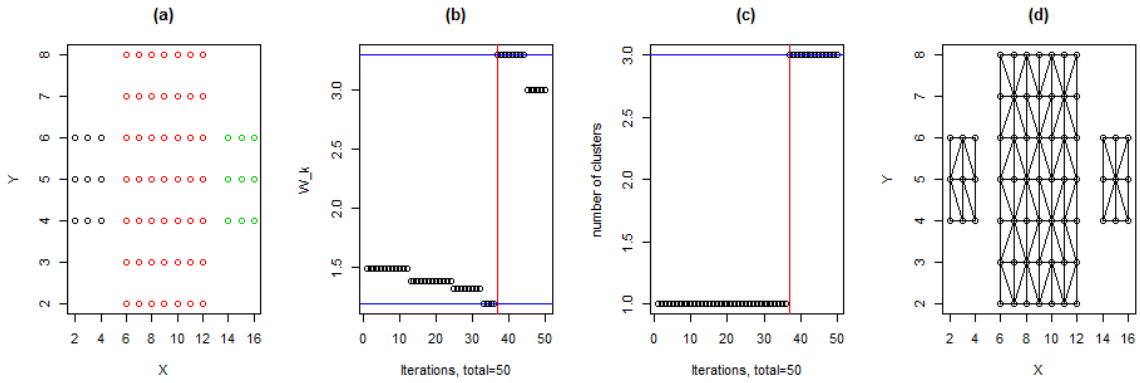

Figure 4.3.5. (a)Natural Classifications of Data set 4, (b) W_k for each iteration with red vertical line at 37^{th} iteration, (c) Number of cluster at each iteration, (d) Remaining graph after all edges g for the 37^{th} iteration is used.

5
Comparative Analysis

In chapters 3 and 4 we discussed in details the model of our pre-processing algorithm. The pre-processing algorithm takes a data set X as input and its output is the number of clusters K and the initial centroid locations $\{w_1, w_2, ..., w_K\}$. A pre-processed K-means is one which takes the output of the pre-processing algorithm as its input. The goal behind developing the pre-processed K-means was based on the hope that the pre-processed K-means would produce better clustering than the regular K-means defined in Algorithm 2.1 as well as being able to reach stable centroids in fewer iterations.

In this section we will evaluate the accuracy of the clusters produced by the pre-processed K-means and the regular K-means. The analysis will be done by using different validation indexes. There are two main types of validation methods: internal and external [27]. Internal evaluation involves using information already in the data set to determine the validity of the clusters [27]. This type of analysis requires no priori information about the data. External evaluation is the comparison between clusters formed by the clustering algorithm and the original class grouping that is already represented in the data - but not used when the algorithm is run [27]. That is, these types of evaluation methods

measure how close the clustering is to a predetermined benchmark class. External evaluation requires priori information about the data. The first part of this section will go over the algorithms and methodology that will be used for the purpose of the analysis. The second part will analyze the results obtained when using the K-means and pre-processed K-means on example data sets using the different validation methods.

5.1 Indexes and Methodology

In this paper we will primarily use external evaluations to analyze the accuracy of clusters. External evaluations compares the clustering produced by a clustering algorithm to a reference, in this case the actual natural classes in the original data sets. Generally this is difficult to do in real life, because when working with an arbitrary unlabeled data set external evaluation is useless. This is when analysts use internal evaluations as a compromise to approximate the accuracy of the clustering. However, since this is a research paper we have the luxury to test our algorithms with data sets we already know information about. This makes external evaluation the ideal technique to use. Still, it is to be made clear that when we ran the K-means and the pre-processed K-means we assumed that additional class information about the data sets were not known, and that the algorithms were run only on the raw data.

In this paper we will use three different types of external evaluation techniques to determine the accuracy of our clustering. The reason for using varied methods goes back to a problem discussed in chapter 2: it is hard to define what is a good cluster. The three methods used in this paper all address the accuracy of clusters in different ways each giving different weights to different requirements of a good cluster. The external evaluation techniques we will be using in this paper are the cluster Purity, F-measure, and Entropy. Detailed description on each evaluation technique is given below.

- Cluster Purity

$$\text{Purity}(S, C) = \frac{1}{N} \sum_{m} \text{Max}_n |c_n \cap s_m|$$

where N is the total number of data points, $S = \{s_1, ...s_m\}$ is the set of classes and $C = \{c_1, ...c_n\}$ is the set of clusters [27]. Each class s is assigned to a cluster w in which it most frequently appears. Essentially, in the purity method, we count the number of points a given cluster has from each class. We then take the number of the times the most frequent class in each cluster appear. Purity is calculated by summing total of this, and then dividing it by the total number of data points in the data set. That is, the results indicates the percentage of data points have been correctly clustered. The value Purity test goes from 0 and 1 with larger values indicating a higher clustering quality.

- F-measure

$$F(i, j) = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Recall}(i, j) = \frac{n_{ij}}{n_i}$$
$$\text{Precision}(i, j) = \frac{n_{ij}}{n_j}$$

where n_{ij} is the number of data points in the cluster j that are originally from class i, n_i is the number of data points in class i, and n_j is the number of data points cluster j [27]. In simple terms, precision is the fraction of retrieved instances that are relevant and recall is the fraction of the relevant instances retrieved. The F-measure provides a harmonic mean of both the precision and recall metric[1].

[1]Harmonic mean: $\frac{1}{F} = \frac{1}{2} \times \frac{1}{\text{Recall}} + \frac{1}{2} \times \frac{1}{\text{Precision}}$

Macro-averaged F-measure for a data set X with m number of classes is

$$F = \frac{1}{m}\sum_{i=1}^{m} \text{Max}_{j=1,...k}\{F(i,j)\}$$

The value F-measure goes from 0 and 1 with larger values indicating a higher clustering quality.

- Entropy

  This is the measure of how the various classes of our data set are distributed within each cluster [27]. Given a particular cluster of c_j of size n_j, the entropy of the cluster is defined as

$$P_{i,j} = |c_j \cap s_i|$$

$$E(S_j) = -\sum_{c_j \cap s_i \neq \emptyset} \frac{P_{i,j}}{n_j} \log \frac{P_{i,j}}{n_j}$$

  where $S = \{s_1, ...s_m\}$ is the set of classes and $C = \{c_1, ...c_n\}$ is the set of clusters. The entropy of the entire data set is defined to be

$$Entropy = \sum_{j=1}^{k} \frac{n_j}{N} E(S_j).$$

  where N is the total number of data points. Therefore the Entropy of the entire data set is the sum of individual cluster entropies weighted according to the cluster size. The smaller the value of entropy, the better the clustering solution. In the entropy measure, when all clusters consist of points from a single class, the entropy is 0. As more and more variations appear in a cluster, the value of the entropy increases.

Other than testing the accuracy of the clusters produced by the K-means and the preprocessed K-means, we will also test how many iterations it takes for each process to

reach stable centroids. The lower the number of iterations it takes an algorithm to reach stable centroids the faster the algorithm is. This is very useful for large data sets since recalculating the centroid locations in each iteration is computationally heavy.

It is important to note that the output of the pre-processing algorithm provides us with the correct value for K. Therefore the pre-processed K-means initiates using assumptions about both K and the initial centroid locations. However, regular K-means is not able to correctly determine the value of K. Normally K is determined for K-means by running the algorithm a large number of times with different K until the most desired clusters are produced. For the sake of simplicity we will use the correct value for the number of clusters K in K-means. Hence in the tests we are only addressing the effectiveness of the initial centroid locations provided by the pre-processing algorithm. This is all we need to test since we have already shown in Chapter 4 that by using the pre-processing algorithm, we are able to correctly determine the value of K.

5.2 Results

The results of the K-means and the pre-processed K-means were analyzed using 6 different data sets. The regular K-means was allowed to run 10 times for each data set, and the clustering result for each run was recorded. The results include the number of iteration it took each of the K-means to reach completion (stable centroids) and also the strength of clustering using the 3 different evaluation techniques.

We only include one result for the pre-processed K-means. That is because unlike the regular K-means the results of the pre-processed K-means is consistent. The reason for its consistency is because the initial centroid locations are fixed. The results below also show the number of iterations it takes the pre-processed K-means to reach completion and the accuracy of the clustering produced by the pre-processed K-means using the 3 different evaluation techniques.

Dataset	**1**	**2**	**3**	**4**	**5**	**6**
K-means 1	11	2	7	9	5	19
K-means 2	4	2	4	4	5	13
K-means 3	9	2	6	7	2	18
K-means 4	11	1	3	6	5	13
K-means 5	7	2	2	2	6	7
K-means 6	8	2	2	7	4	16
K-means7	8	2	2	6	3	15
K-means 8	7	1	4	4	6	9
K-means 9	13	2	2	5	4	18
K-means 10	9	2	2	3	5	11
Average K-means	8.7	1.8	3.4	5.3	4.5	13.9
Pre-processed K	7	1	1	1	2	14

Table 5.2.1. Number of iteration to reach stable centroids for 10 different K-means runs and 1 Pre-processed K-mean run using 6 different data sets.

Table 5.2.1 contains the results of the number of iterations it takes the pre-processed K-means, and the 10 different regular K-means - with randomly selected initial centroid locations- to reach completion. The results show that on average the pre-processed K-means outperforms the regular K-means for data sets 1 to 5. For data set 6, the pre-processed K-means and regular K-means perform almost identically on average, with the pre-processed K-means under-performing by just 0.1 number of iterations. The results also indicate that the pre-processed K-means significantly outperforms the worst case scenario for regular K-means. This is even true for data set 6, where the pre-processed K-means completes in 14 iterations where as the worst case for regular K-means is 19 iterations.

Table 5.2.2 shows the result of F-measure, Purity test and Entropy test of the clustering produced by the pre-processed K-means and by 6 random runs of the regular K-means. The results indicate that the pre-processed algorithm always produced either more or just as accurate clustering than the best result of the regular K-means. For data sets 1, 4, and 6 the pre-processed K-means produced clusters that are significantly more accurate than the best result from K-means. For data sets 2 and 5, both processes produced identical clustering, hence their clustering accuracy is equal. For data set 3, the pre-processed K-

means produced clusters that are significantly more accurate than 5 of the K-means runs and clusters that are equally accurate to 1 of the K-means runs.

	Data set 1	**Data set 2**	**Data set 3**	**Data set 4**	**Data set 5**	**Data set 6**
F-measure						
P K-means	**0.818091**	**1**	**1**	**0.881026**	**0.848485**	**0.779906**
K-means 1	0.747346	**1**	0.5	0.640833	**0.848485**	**1.159346**
K-means 2	0.510786	**1**	0.5	0.562555	**0.848485**	**1.159346**
K-means 3	0.747346	**1**	0.5	0.798344	**0.848485**	**1.159346**
K-means 4	0.510786	**1**	**1**	0.562555	**0.848485**	**1.159346**
K-means 5	0.510786	**1**	0.5	0.562555	**0.848485**	**1.159346**
K-means 6	0.747346	**1**	0.5	0.798344	**0.848485**	**1.159346**
Purity						
P K-means	**0.786325**	**1**	**1**	**0.895522**	**0.848485**	**0.7675**
K-means 1	0.760684	**1**	0.5	0.656716	**0.848485**	0.76
K-means 2	0.521368	**1**	0.5	0.567164	**0.848485**	0.76
K-means 3	0.760684	**1**	0.5	0.791045	**0.848485**	0.76
K-means 4	0.521368	**1**	**1**	0.567164	**0.848485**	0.76
K-means 5	0.521368	**1**	0.5	0.567164	**0.848485**	0.76
K-means 6	0.760684	**1**	0.5	0.791045	**0.848485**	0.76
Entropy						
P K-means	**0.113941**	**0**	**0**	**0.161357**	**0.425328**	**0.469492**
K-means 1	0.552395	**0**	0.462098	0.415988	**0.425328**	0.477794
K-means 2	0.787285	**0**	0.462098	0.466748	**0.425328**	0.477794
K-means 3	0.552395	**0**	0.462098	0.22977	**0.425328**	0.477794
K-means 4	0.787285	**0**	**0**	0.466748	**0.425328**	0.477794
K-means 5	0.787285	**0**	0.462098	0.466748	**0.425328**	0.477794
K-means 6	0.552395	**0**	0.462098	0.22977	**0.425328**	0.477794

Table 5.2.2. Accuracy of the clustering produced by 6 K-means runs and 1 pre-processed K-means run on 6 different data set measured using external evaluation techniques F-measure, Purity, and Entropy

5.3 Summary

In this section we compared the clustering results of K-means and pre-processed K-means using external evaluation techniques such as Purity, F-measure, and Entropy. We conducted the comparison using 6 very different data sets. Each data set was clustered 10

times using the K-means and 1 time with the pre-processed K-means. The results of the analysis are available in Table 5.2.1 and 5.2.2.

For these analyses we assumed we know the correct number of clusters K when running the regular K-means, which is information we would not be aware of. Unlike the regular K-means, the pre-processed K-means not only produces initial cluster centroid locations, it also gives us the number of K clusters required. This means that in these tests we gave the regular K-means an unfair advantage, which it normally does not have, and still the results show that the pre-processed algorithm produced clusters that are more accurate and require fewer number of iterations to reach completion.

6
Additional Improvements

In the last three chapters we developed our pre-processing algorithm and did a comparative analysis between the pre-processed K-means and the regular K-means. We showed that on average our pre-processed version completes the clustering process in a fewer number of iterations. We also showed that the pre-processed version almost always either performs as well as or outperforms the best case scenario clustering of the regular K-means in terms of accuracy of the clustering.

However, the question still exists as to whether we can still improve this process further or not. There are two ways we can improve this algorithm; first by reducing the number of iterations and second by improving the accuracy of the clustering. For this we propose two variant methods to the algorithm presented in Chapter 3-5. The first method reduces the number of iterations in the K-means, meaning we can reach completion significantly faster. The second method tweaks the pre-processing algorithm so that it behaves like a clustering algorithm on its own accord.

6.1 Single iteration K-means

In Chapter 5 we saw that the pre-processed K-means algorithm produced very accurate results. While working with a large number of data sets, we observed that the pre-processed K-means reached stable centroids after a single iteration for many data sets. This indicates that the initial centroid locations used in the pre-processed K-means is very strong. It also indicates that we may be able to obtain fairly accurate clustering from just one iteration from any data set. To test this, we run the pre-processed K-means with just one iteration and check the accuracy of the clusters produced using the validation test from Chapter 5. The results obtained are shown below in Table 6.1.1.

	Data set 1	**Data set 2**	**Data set 3**	**Data set 4**	**Data set 5**	**Data set 6**
F-measure	0.7854431	1	1	0.881026	0.8787879	0.9528739
P K-means	0.7863248	1	1	0.895522	0.8787879	0.955
K-means 1	0.3971288	0	0	0.161357	0.3693331	0.176247

Table 6.1.1. Accuracy of the clustering produced by the pre-processed K-means restricted to 1 iteration on 6 different data set measured using external evaluation techniques F-measure, Purity, and Entropy

The results for Data set 2, 3 and 4 are identical to that of the pre-processed K-means used in Chapter 5. This is to be expected because if we recall, the pre-processed K-means reached stable centroids with these data sets after 1 iteration. So for these, nothing actually changed when using this new restricted version.

The results for Data set 1 show that the clusters formed by this new restricted version are less accurate than the pre-processed K-means used in Chapter 5 (Table 5.2.2) . However, the cluster formed by this restricted version is still seen to be more accurate when compared to regular K-means.

The results from Data set 4 and 5 shows that the cluster formed by this new restricted version is significantly more accurate than the pre-processed K-means used in Chapter 5. Recall that Data set 4 and 5 contain the type of clusters that K-means is weak at

clustering: clusters of different shape and size. This tends to indicate that when the pre-processed K-means is run on these data sets for a large number of iterations, the clustering accuracy actually decreases.

From the results obtained by this method we hypothesize that if the classification of a data set is such that K-means would not produce good results with it, then by running a single iteration pre-processed K-means, we obtain better clustering. That is if the data set contains classes that are of different shapes, sizes and densities, then when K-means is allowed to run for longer it forces the centroid locations obtained from the pre-processed algorithm to deviate significantly and hence produce weaker clustering.

Also the results suggest that even with the clusters that K-means is good at clustering, the single iteration pre-processed K-means produces fairly accurate clusters. Therefore if the goal of the cluster analysis is to obtain fairly accurate clusters or obtain clusters that are of different shapes and sizes, then this method may be more useful than the regular pre-processed K-means.

6.2 Delaunay Clustering Algorithm

In Chapter 5 we saw that the pre-processed K-means algorithm produced more accurate clustering than the regular K-means. However, we still need to ask, is this the most accurate clustering we can get using the clustering algorithm? The answer to that question is illustrated in Figure 6.2.1, 6.2.2 and 6.2.3. For all three figures, (a) shows the regular K-mean clustering, (b) shows the pre-processed K-mean clustering, and (c) shows the separate Delaunay connected graph produced by our pre-processing algorithm. The natural classification of these data sets are available in Figure 8.2.5, 8.2.3, and 8.2.6 respectively.

In both Figure 6.2.1 and 6.2.3 the clustering produced by the pre-processed K-means algorithm is almost identical to the K-means and considerably different from the original class separations. In Figure 6.2.2 the pre-processed K-means clustering produced signifi-

cantly more accurate clustering than the regular K-means when compared to the original class separation, but the result is still not perfectly accurate to the natural classification. Part (c) of Figure 6.2.1, 6.2.2 and 6.2.3 shows the separate Delaunay connected graphs. These graphs are obtained by using the pre-processing method discussed in Chapters 3 and 4 (Figure 4.3.4, 4.3.5, and 4.3.3). Notice that each of the connected graphs in these figures are perfectly accurate clusters in relation to the original class separation of their respective data sets.

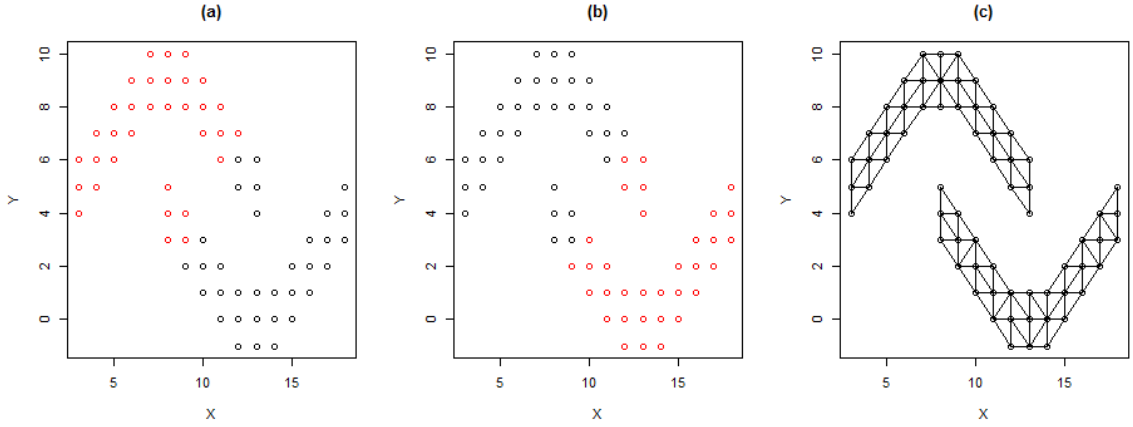

Figure 6.2.1. (a) Data set 5 clustered using K-means, (b) Data set 5 clustered using pre-processed K-means, (c) Data set 5 clustered using Delaunay Clustering algorithm.

Based on these results we propose a new distance-based clustering algorithm. The algorithm is similar to our pre-processing algorithm discussed in Chapters 3 and 4. The algorithm will use Delaunay triangulation on a data set X. It will then remove all edges with edge weights that are equal to or greater than a cut-off g. The removal of the edges will create separate connected graphs. Each connected graph will act as a separate cluster. The full algorithm is presented below.

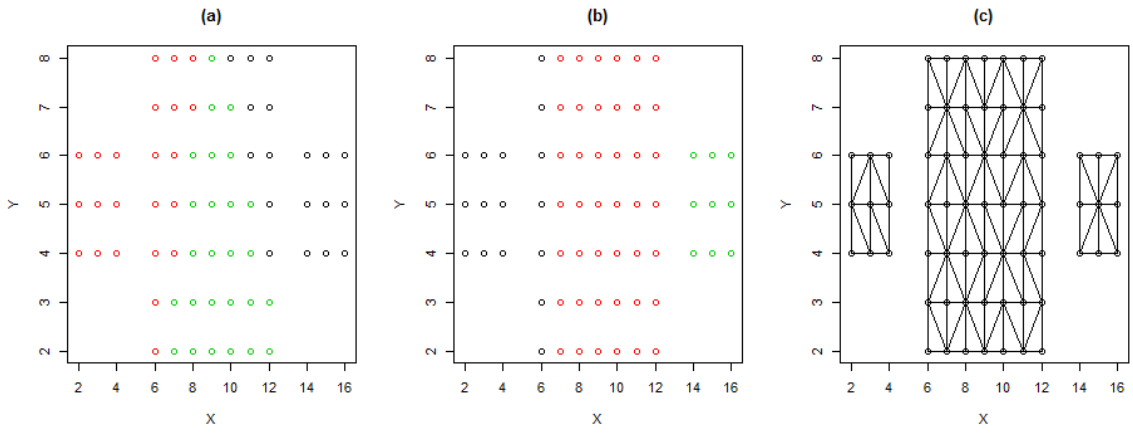

Figure 6.2.2. (a) Data set 4 clustered using K-means, (b) Data set 4 clustered using pre-processed K-means, (c) Data set 4 clustered using Delaunay Clustering algorithm.

Algorithm 6.2.1: Delaunay Clustering Algorithm:

Input: Data set X, size of mini clusters T

Output: List of lists $Q = \{C_1, ...C_K\}$, where each C_j is a list of all points in the connected graph G_j.

$G = X$ with edges created via a Delaunay Triangulation.

$L = \Delta_{u,v}$ ▷ (where L is a list and $\Delta_{u,v}$ is the edge weight between all connected points $u, v \in G$)

Plot Kernal Density estimation of L

if cut-off length g is observable in density plot **then** ▷ (Method for determining cut-off length g from density plot is provided in Chapter 4.2)

cutOff $= g$

else

cutOff = the cut-off g which causes max difference in W_K ▷ (Method for determining cut-off length g from W_K is provided in Chapter 4.3)

end if

$G = G$ | all edges equal to and above g removed

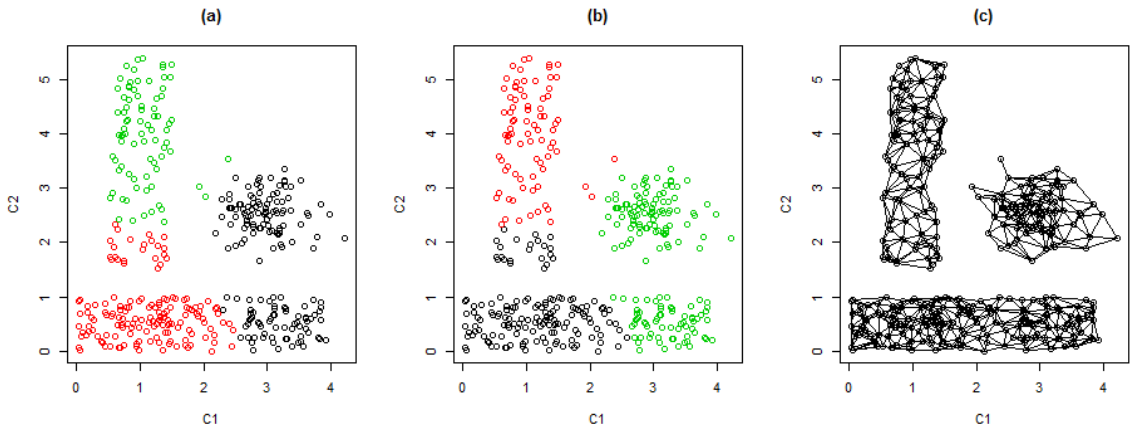

Figure 6.2.3. (a) Data set 6 clustered using K-means, (b) Data set 6 clustered using pre-processed K-means, (c) Data set 6 clustered using Delaunay Clustering algorithm.

Connect all connected graphs with T data points or less to their respective closest connected graphs with more than T data points.

K = number of connected graphs in $G = \{G_1, ..., G_K\} | G_i$ is a connected graph such that $|G_i| > T$

$C = \{\}$

for i in 1:K **do**

$C[i]$ = All points in G_i ▷ $(C_i = C[i])$

end for

return C

Essentially in this algorithm we remove all edges with weights equal to or greater than g, similar to the way we did it for our pre-processing algorithm. Next we take all the mini clusters produced and connect them to their closest cluster. The components are now separated into K connected graphs; these are our K clusters. The accuracy of the clusters produced by the Delaunay Clustering Algorithm was evaluated using the same methods used in Chapter 5. The results are shown in Table 6.2.1. These results can be compared to the results obtained for the K-means and for the pre-processed K-means in Table 5.2.2.

Based on our analysis we see that for Data sets 2 to 6 the accuracy of the clustering for the Delaunay Clustering Algorithm is significantly better. Therefore by using this method we are able to get very accurate clustering for many different types of data sets.

	Data set 2	**Data set 3**	**Data set 4**	**Data set 5**	**Data set 6**
F-measure	1	1	1	1	1
Purity	1	1	1	1	1
Entropy	0	0	0	0	0

Table 6.2.1. Accuracy of the clustering produced by Delaunay Clustering algorithm on 6 different data set measured using external evaluation techniques F-measure, Purity, and Entropy

7
Conclusion and Future Work

In this paper we introduced a pre-processing algorithm to the K-means, loosely based upon the ideas presented in *Clusterability Detection and Cluster Initialization* [10]. K-means is a centroid and distance-based clustering algorithm which produces K clusters, where K is an input parameter in the algorithm. K-means initiates by randomly choosing initial centroid locations. We discussed the popularity and effectiveness of K-means as a clustering algorithm. However, the accuracy of K-means is highly dependent on the value of K, choice of centroids, and type of clustering. The purpose of the pre-processing algorithm is to help determine a suitable value for K using solely information provided in the data set. Also, the pre-processing algorithm provides a more suitable way to choose the initial centroid location in K-means.

The algorithm takes a data set and connects points with edges created via a Delaunay triangulation. We then show that by correctly removing all edges of weight equal to or greater than g and all mini clusters, we are able to obtain K separate connected graphs. These connected graphs act as clusters in the data set. Therefore an approximation for input parameter for K-means is the number of separate connected graphs obtained by our

pre-processing algorithm. Also, the mean of all the points in each connected graph is used as the initial centroid location for K-means.

In this paper we also demonstrated two different methods that can be used to semi-interactively determine a suitable value for the cut-off g. By using the cut-off g obtained through our method, we showed that we were able to correctly pre-process the K-means for a large number of data sets. Also, by using different cluster validation indexes, we showed that the clustering produced by a pre-processed K-means is better than a regular K-means, both in terms of the accuracy of the clustering and in terms of the number of iterations it took to reach stable centroids.

One key shortcoming of this algorithm would be the lack of a fully automated process for determining the cut-off g. Another shortcoming is the need to approximate the user specified threshold T for the size of the mini cluster. Approximating a suitable value for T is not a very difficult task but an automated process that can do this for the user would prove to be very useful and is something that we would like to address for future work.

Also, in this paper we mainly focused on two dimensional data sets. However, we believe that our method can easily be extended to larger dimensions. This is because the Delaunay triangulation works with higher dimension data sets [31]. We believe, that we will be able to use our method on high dimensional data set to produce suitable results in smaller subspaces [10,30].

We believe that by running the pre-processing algorithm on samples of very large data sets, we will be able to obtain reasonably accurate information about K and the initial centroid locations [10,12]. This information can then be fed into the K-means and used to determine the clusters in the large data sets. We have already started working on exploring the results of the pre-processing algorithm run on a sample of very large data sets and have obtained very accurate clustering for those data sets. However, more work still needs to be done on this topic before we can make any conclusive arguments.

We proposed two different ways this pre-processed K-means algorithm can modified. First, we showed that the pre-processing algorithm can be used on a single iteration K-means and still produce reasonably accurate clusters. Second, we showed that the pre-processing algorithm can be modified and used on its own: Delaunay Clustering Algorithm. We also showed that the clustering results obtained by this method produced clusters that are significantly more accurate than that of both the K-means and the pre-processed K-means.

We believe that both the pre-processsed K-means and the Delaunay Clustering Algorithm will be very useful for distance based clustering of a given data set. We wish to continue working on these algorithms. Our work will be focused on automating the process further and analyzing the results obtained by these algorithms on more data sets, preferably with different sizes and number of dimensions.

Bibliography

[1] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, Philip S Yu, Zhi-Hua Zhou, Michael Steinbach, David J Hand, and Dan Steinberg, *Top 10 algorithm in data mining*, Springer **14:1-37** (2007).

[2] A.K. Jain, M.N. Murty, and P.J. Flynn, *Data Clustering: A review*, ACM Computing Surveys **31** (1999).

[3] J MacQueen, *Some Methods For Classification and Analysis of Multivariate Observation*, Western Management Science Institute **047-041** (1967).

[4] Mu-Chun Su and Chien-Hsing Chou, *A Modified Version of the K-means Algorithm with a Distance Based Cluster Symmetry*, IEEE Transactions on pattern analysis and machine learning **23** (2001), ¡Starting Page–Ending Page¿.

[5] Alsabti Khaled, Sanjay Ranka, and Vineet Singh, *An Efficient K-Means Clustering Algorithm*, Information Technology Lab(ITL).

[6] Usama Fayyad, Gregory Oiatetsky-Shapiro, and Padhraic Smyth, *From Data Mining to Knowledge Discovery in Database*, AI Magazine Volume **17** (1996).

[7] Raed T. Aldahdooh and Wesam Ashour, *DIMK-means "Distance-based Initialization Method for K-means Clustering Algorithm"*, I.J. Intelligent Systems and Applications **02** (2013), 41–51.

[8] Sanjay Chawla and Aristides Gionis, *k-means–: A unified approach to clustering and outlier detection.*

[9] David Arthur and Sergei Vassilvitskii, *k-means++: The Advantages of Careful Seeding.*

[10] Scott Epter, Mukkai Krishnamoorthy, and Mohammed J Zaki, *Clusterability Detection and Cluster Initialization*, ResearchGate.

[11] L.V. Bijuraj, *Clustering and its Applications*, National Conference on New Horizons in IT (2013), 169-172.

[12] Pang-Ning Tan, Steinbach Michael, and Kumar Vipin, *Introduction to Data Mining*, Pearson, 2006.

[13] D T Pham, S S Dimov, and Nguyen C D, *Selection of K in K-means clustering*, Journal of Mechanical Engineering Science **219** (2005), 103-119.

[14] Tibshirani Robert, Guenther Walther, and Hastie Trevor, *Estimating the number of clusters in a data set via the gap statistic*, Royal Statistical Society **63** (2001), 411-423.

[15] Dibya Joyti Bora and Anil Kumar Gupta, *Effect of Different Distance Measures on the Performance of K-Means Algorithm: An Experimental study in Matlab*, International Journal of Computer Science and Information Technologies **5** (2014), 2501-2506.

[16] John Augustine, *Delaunay Triangulation*, 2012. CS6100: Topics in Design and Analysis of Algorithm.

[17] Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy, *The Effectiveness of Lloyd-Type Methods for the k-Means Problem*, Foundations of Computer Science **5** (2006), 165-176.

[18] Ahmed Elgammal, Ramani Duraiswami, and Larry S. Davis, *Efficient Kernal Density Estimation Using the Fast Gauss Transform with Applications to Segmentation and Tracking*, Pattern Analysis and Machine Intelligence, IEEE **25** (2003), 1499-1504.

[19] Rui Castro, *Lecturers 2 and 3- Goodness-of-Fit(GoF) Tests*, 2013. Technische Universiteit Eindhoven, Applied Statistics, Lecture 23.

[20] S. Csrgo and J.J. Faraway, *The exact and asymptotic distributions of Cramr-von Mises statistics.*, Journal of the Royal Statistical Society **Series B 58** (1996), 221234.

[21] G. Marsaglia, J. Marsaglia, and Larry S. Davis, *Evaluating the Anderson-Darling Distribution.*, Journal of Statistical Software **9 (2)** (2004), 1-5.

[22] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.

[23] Gabor Csardi and Tamas Nepusz, *The igraph software package for complex network research*, InterJournal **Complex Systems** (2006), 1695.

[24] Kai Habel and Raoul Grasman and Andreas Stahel and Andreas Stahel and David C. Sterratt, *geometry: Mesh generation and surface tesselation*, 2014. R package version 0.3-5.

[25] Evgenia Dimitriadou, *cclust: Convex Clustering Methods and Clustering Indexes*, 2014. R package version 0.6-19.

[26] Gregory R. Warnes and Ben Bolker and Lodewijk Bonebakker and Robert Gentleman and Wolfgang Huber Andy Liaw and Thomas Lumley and Martin Maechler and Arni Magnusson and Steffen Moeller and Marc Schwartz and Bill Venables, *gplots: Various R Programming Tools for Plotting Data*, 2015. R package version 2.16.0.

[27] Erendira Rendon, Itzel Abundez, Alejandra Arizmendi, and Elvia Quiroz, *Internal versus External cluster validation indexes*, International Journal of Computers and Communications **5** (2011), 27-34.

[28] Steven S. Skiena, *The Algorithm Design Manual*, 2nd ed., Springer, 2008.

[29] Jeffrey D. Ullman, *Clustering Algorithms*, 2005. Stanford University, CS 345.

[30] Charu C. Aggarwal and Yu Philio S., *Finding generalized projected clusters in high dimensional spaces.*, ACM SIGMOND International Conference on Management of Data (2000).

[31] Long Chen and Jin-Chao Xu, *Optimal Delaunay Triangulations*, Journal of Computational Mathematics **22-2** (2004), 299-308.

[32] *e-Handbook of Statistical Methods*, NIST/SEMATECH, 2013. *http* : *//www.itl.nist.gov/div*898/*handbook/eda/section*3/*histogr*6.*htm*, April 2015.

[33] *e-Handbook of Statistical Methods*, NIST/SEMATECH, 2013. *http* : *//www.itl.nist.gov/div*898/*handbook/eda/section*3/*eda*366*g.htm*, April 2015.

[34] Laura Chihara and Tim Hesterberg, *Mathematical Statistics with Resampling and R*, Wiley, 2011.

8
Appendix

8.1 Algorithms

Algorithm 8.1.1: Kernal Density function, R density.default [22]

```
‘function (x, bw = "nrd0", adjust = 1, kernel = c("gaussian",
    "epanechnikov", "rectangular", "triangular", "biweight",
    "cosine", "optcosine"), weights = NULL, window = kernel,
    width, give.Rkern = FALSE, n = 512, from, to, cut = 3, na.rm = FALSE,
    ...)
{
    if (!missing(...))
        warning("non-matched further arguments are disregarded")
    if (!missing(window) && missing(kernel))
        kernel <- window
    kernel <- match.arg(kernel)
    if (give.Rkern)
        return(switch(kernel, gaussian = 1/(2 * sqrt(pi)), rectangular = sqrt(3)/6,
            triangular = sqrt(6)/9, epanechnikov = 3/(5 * sqrt(5)),
            biweight = 5 * sqrt(7)/49, cosine = 3/4 * sqrt(1/3 -
                2/pi^2), optcosine = sqrt(1 - 8/pi^2) * pi^2/16))
    if (!is.numeric(x))
        stop("argument ’x’ must be numeric")
    name <- deparse(substitute(x))
    x <- as.vector(x)
    x.na <- is.na(x)
    if (any(x.na)) {
        if (na.rm)
            x <- x[!x.na]
        else stop("’x’ contains missing values")
    }
```

```
N <- nx <- as.integer(length(x))
if (is.na(N))
    stop("invalid value of length(x)")
x.finite <- is.finite(x)
if (any(!x.finite)) {
    x <- x[x.finite]
    nx <- length(x)
}
if (is.null(weights)) {
    weights <- rep.int(1/nx, nx)
    totMass <- nx/N
}
else {
    if (length(weights) != N)
        stop("'x' and 'weights' have unequal length")
    if (!all(is.finite(weights)))
        stop("'weights' must all be finite")
    if (any(weights < 0))
        stop("'weights' must not be negative")
    wsum <- sum(weights)
    if (any(!x.finite)) {
        weights <- weights[x.finite]
        totMass <- sum(weights)/wsum
    }
    else totMass <- 1
    if (!isTRUE(all.equal(1, wsum)))
        warning("sum(weights) != 1  -- will not get true density")
}
n.user <- n
n <- max(n, 512)
if (n > 512)
    n <- 2^ceiling(log2(n))
if (missing(bw) && !missing(width)) {
    if (is.numeric(width)) {
        fac <- switch(kernel, gaussian = 4, rectangular = 2 *
            sqrt(3), triangular = 2 * sqrt(6), epanechnikov = 2 *
            sqrt(5), biweight = 2 * sqrt(7), cosine = 2/sqrt(1/3 -
            2/pi^2), optcosine = 2/sqrt(1 - 8/pi^2))
        bw <- width/fac
    }
    if (is.character(width))
        bw <- width
}
if (is.character(bw)) {
    if (nx < 2)
        stop("need at least 2 points to select a bandwidth automatically")
    bw <- switch(tolower(bw), nrd0 = bw.nrd0(x), nrd = bw.nrd(x),
        ucv = bw.ucv(x), bcv = bw.bcv(x), sj = , `sj-ste` = bw.SJ(x,
            method = "ste"), `sj-dpi` = bw.SJ(x, method = "dpi"),
        stop("unknown bandwidth rule"))
}
```

```
    if (!is.finite(bw))
        stop("non-finite 'bw'")
    bw <- adjust * bw
    if (bw <= 0)
        stop("'bw' is not positive.")
    if (missing(from))
        from <- min(x) - cut * bw
    if (missing(to))
        to <- max(x) + cut * bw
    if (!is.finite(from))
        stop("non-finite 'from'")
    if (!is.finite(to))
        stop("non-finite 'to'")
    lo <- from - 4 * bw
    up <- to + 4 * bw
    y <- .Call(C_BinDist, x, weights, lo, up, n) * totMass
    kords <- seq.int(0, 2 * (up - lo), length.out = 2L * n)
    kords[(n + 2):(2 * n)] <- -kords[n:2]
    kords <- switch(kernel, gaussian = dnorm(kords, sd = bw),
        rectangular = {
            a <- bw * sqrt(3)
            ifelse(abs(kords) < a, 0.5/a, 0)
        }, triangular = {
            a <- bw * sqrt(6)
            ax <- abs(kords)
            ifelse(ax < a, (1 - ax/a)/a, 0)
        }, epanechnikov = {
            a <- bw * sqrt(5)
            ax <- abs(kords)
            ifelse(ax < a, 3/4 * (1 - (ax/a)^2)/a, 0)
        }, biweight = {
            a <- bw * sqrt(7)
            ax <- abs(kords)
            ifelse(ax < a, 15/16 * (1 - (ax/a)^2)^2/a, 0)
        }, cosine = {
            a <- bw/sqrt(1/3 - 2/pi^2)
            ifelse(abs(kords) < a, (1 + cos(pi * kords/a))/(2 *
                a), 0)
        }, optcosine = {
            a <- bw/sqrt(1 - 8/pi^2)
            ifelse(abs(kords) < a, pi/4 * cos(pi * kords/(2 *
                a))/a, 0)
        })
    kords <- fft(fft(y) * Conj(fft(kords)), inverse = TRUE)
    kords <- pmax.int(0, Re(kords)[1L:n]/length(y))
    xords <- seq.int(lo, up, length.out = n)
    x <- seq.int(from, to, length.out = n.user)
    structure(list(x = x, y = approx(xords, kords, x)$y, bw = bw,
        n = N, call = match.call(), data.name = name, has.na = FALSE),
        class = "density")
}'
```

8.2 Data sets

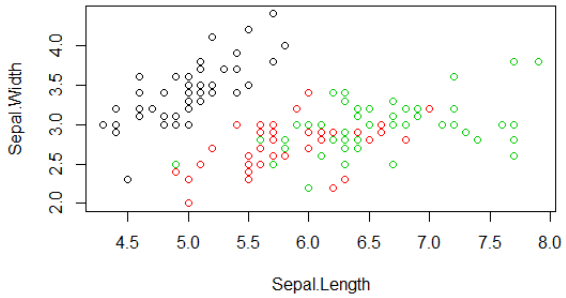

Figure 8.2.1. Data Set 1

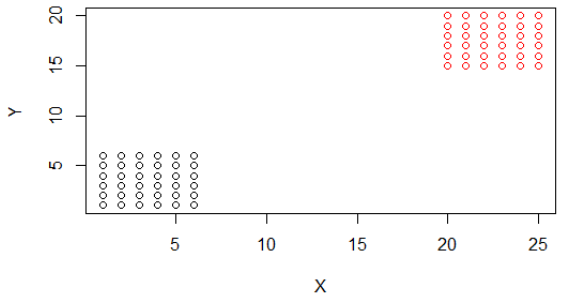

Figure 8.2.2. Data Set 2

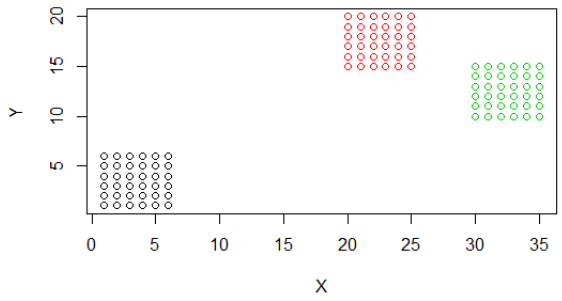

Figure 8.2.3. Data Set 3

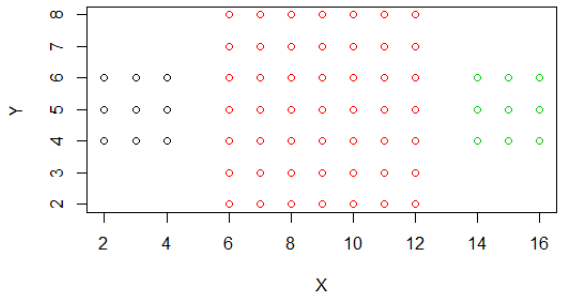

Figure 8.2.4. Data Set 4

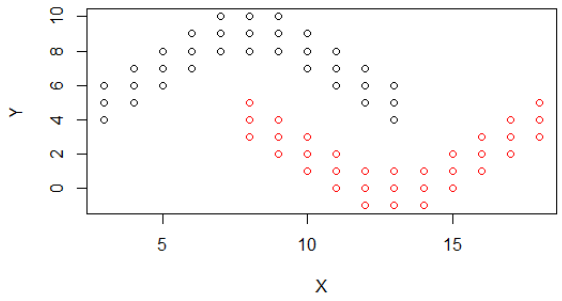

Figure 8.2.5. Data Set 5

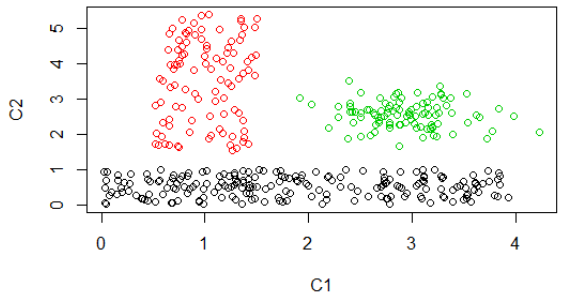

Figure 8.2.6. Data Set 6

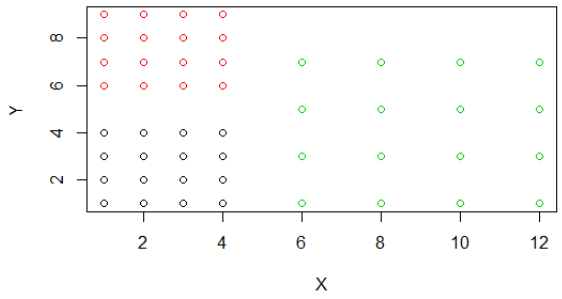

Figure 8.2.7. Data Set 7