Robustification of the sparse K-means clustering algorithm

by

Yumi Kondo

B.S. Economics, American University, 2009

B.A. Business Administration, Ritsumeikan University, 2009

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES
(Statistics)

The University Of British Columbia
(Vancouver)

August 2011

Abstract

Searching a dataset for the ''natural grouping / clustering'' is an important explanatory technique for understanding complex multivariate datasets. One might expect that the true underlying clusters present in a dataset differ only with respect to a small fraction of the features. Furthermore, one might afraid that the dataset might contain potential outliers.

Through simulation studies, we find that an existing sparse clustering method can be severely affected by a single outlier. In this thesis, we develop a robust clustering method that is also able to perform variable selection: we robustified sparse *K*-means (Witten and Tibshirani [28]), based on the idea of trimmed *K*-means introduced by Gordaliza [7] and Gordaliza [8]. Since high dimensional datasets often contain quite a few missing observations, we made our proposed method capable of handling datasets with missing values. The performance of the proposed robust sparse *K*-means is assessed in various simulation studies and two data analyses. The simulation studies show that robust sparse *K*-means performs better than other competing algorithms in terms of both the selection of features and the selection of a partition when datasets are contaminated. The analysis of a microarray dataset shows that robust sparse *K*-means best reflects the oestrogen receptor status of the patients among all other competing algorithms. We also adapt Clest (Duboit and Fridlyand [5]) to our robust sparse *K*-means to provide an automatic robust procedure of selecting the number of clusters. Our proposed methods are implemented in the `R` package `RSKC`.

Contents

A Technical Appendix 86
A.1 The algorithm of Hartigan and Wong 86
A.2 The difference between Silhouette and the revised silhouette . . . 87

List of Tables

List of Figures

Figure A.1 The example of negative Silhouette. 89

Acknowledgments

I would like to express my gratitude to my supervisors, Professor Matias Salibian-Barrera and Professor Ruben H. Zamar. Your patient teaching, encouragement and guidance were absolutely necessary to complete this thesis. I am very thankful that you let me think about problems throughly and taught me how to approach to them. You made Statistics very interesting to me. I also express my gratitude to Professor John Petkau to be the second reader of this thesis.

I would like to thank every single friend of mine at UBC Statistics department. The discussions with you at LSK office late night inspired me and enriched my understanding in Statistics. Without all of you, I could not have enjoyed the process of learning this much nor could I push myself for learning. I will never forget this intense two years at UBC and I am proud of completing MSc in Statistics with you.

Chapter 1

Introduction

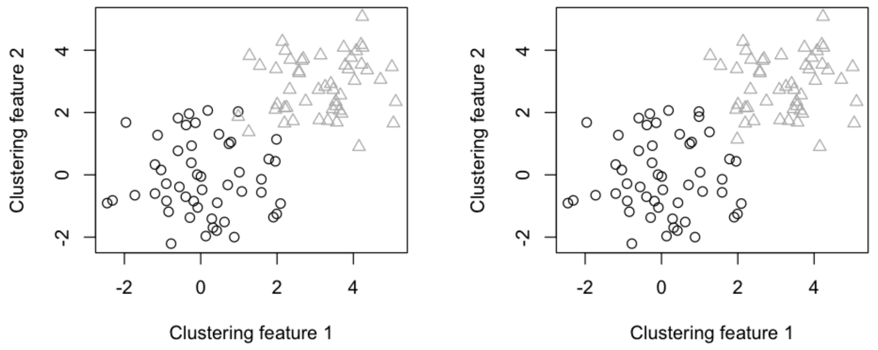
1.1 Clustering

Searching for ‘‘natural’’ groups of objects is an important exploratory technique for understanding complex data. The origins of clustering can be traced back to taxonomy where it is necessary that different people assign similar objects to the same group. Clusterings or groupings were traditionally done by taxonomists who picked important grouping variables based on their rich knowledge of species. Today, the principal function of clustering is to name, display, summarize and to elicit an explanation for the resulting partitions of the dataset (Hartigan [10]).

In Statistics and Computer Science, clustering means automatic (computer aided) *grouping of similar cases* based on some dissimilarity measure. Clustering is sometimes refereed to as ‘‘numerical taxonomy’’.

For example, a DNA microarray is a type of dataset to which clustering algorithms are applied. A microarray is a rectangular array of N rows, one for each case (e.g. a patient or a tumor) and p columns, one for each feature (e.g. genes, SNP’s). A reliable, precise and robust classification of tumors is essential for successful diagnosis of cancer. In a clinical application of microarray-based cancer diagnostics, the definition of new tumor classes can be based on the partitions produced by clustering. The clusters would then be used to build predictors for new tumor samples (Duboit and Fridlyand [5]).

Current applications of clustering algorithms often include a large number of

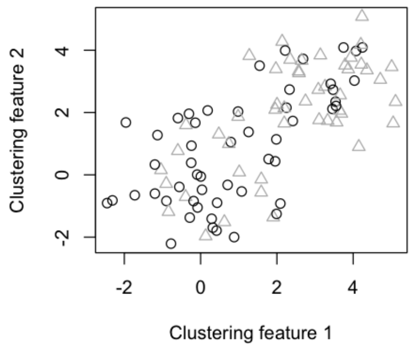


(a) True cluster labels (b) The partition from 2-means

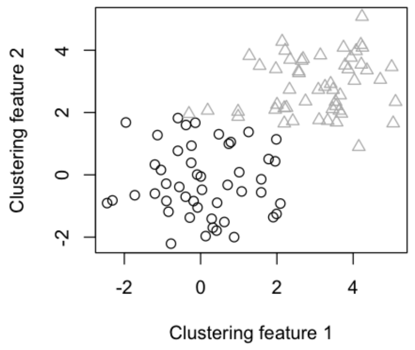
Figure 1.1: The simulated dataset consists of 2 clustering features.

features and visualizing such datasets is difficult. Typically, only a relatively small number of features are important to determine the class memberships of the cases. If thousands of potential clustering features must be considered, the traditional taxonomists' approach of hand picking important features becomes difficult and impractical. Instead, we need a method that automatically chooses the important clustering variables. Furthermore, large datasets may contain outliers, which are defined as cases that do not belong to any of the given clusters. Then we may wish to use a wise algorithm that identifies the important features and outliers together with the clusters.

As a motivating example, we generate 100 independent cases from a bivariate normal distribution. Fifty cases are from a bivariate normal distribution with mean $(0,0)^T$ and the remaining 50 cases have mean $(3,3)^T$. In this thesis, the features whose means vary over clusters are called ''*clustering features*''. Non-clustering features are called ''*noise features*''. The first panel of Figure 1.1 shows the distribution of the generated sample marked by true cluster labels. From the second panel, we see that 2-means successfully identifies the two clusters. In the first and the second panels of Figure 1.2, we show the clustering result when 1000 noise



(a) The partition from 2-means



(b) The partition from sparse 2-means

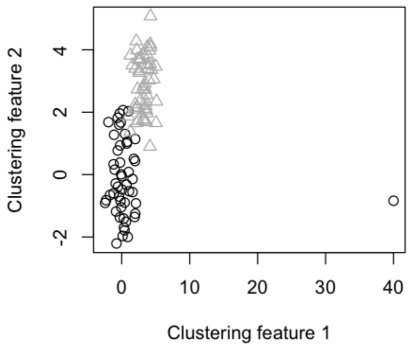
Figure 1.2: The simulated dataset consists of 2 clustering features and 1000 noise.

features are added to the original bivariate data. The noise features are all independent standard normal variables. In this new setting, 2-means fails to identify the two clusters (first panel in Figure 1.2). Sparse 2-means, however, automatically identifies the two clustering features and manages to distinguish the given clusters (second panel in Figure 1.2).

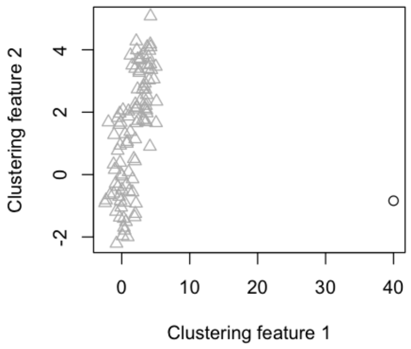
Now, we examine the effect of adding some outliers to this dataset. A single extreme value is added to the clustering feature 1. Figure 1.3 shows the partition results: sparse 2-means fails to identify the two clusters.

Sparse 2-means returns a cluster with a single outlier and a cluster with all the rest of the cases. However, our proposed robust sparse 2-means automatically finds the outlier and correctly identifies the two clusters.

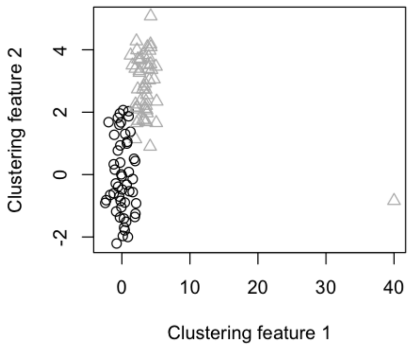
As another motivating example, 10 outliers are added to the clustering features 1 and 2. The first panel of Figure 1.4 shows the true cluster labels with respect to the marginal bivariate distribution of the two clustering features. Outliers are marked with a symbol ‘‘+’’. The second panel shows the partition result of sparse 2-means. Again, sparse 2-means returns a cluster which contain only the outliers. The third panel shows the partition result of robust 2-means. It successfully captures all the



(a) The true cluster labels

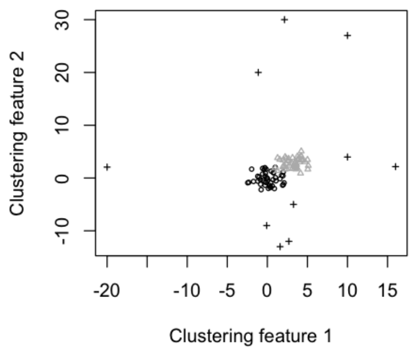


(b) The partition from sparse 2-means

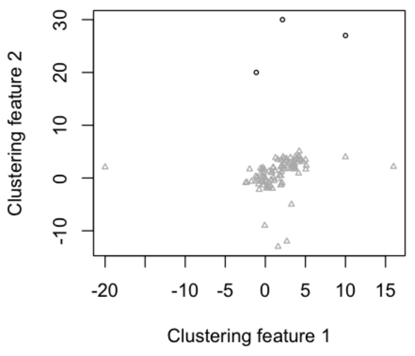


(c) The partition from robust sparse 2-means

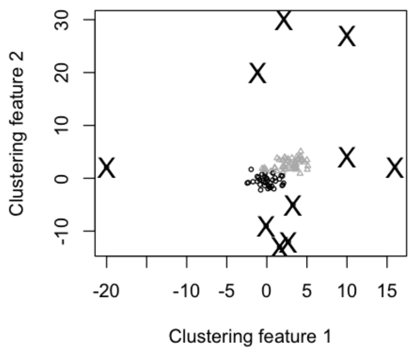
Figure 1.3: The simulated dataset is consists of 2 clustering features and 1000 noise feautres and a single outlier.



(a) The true cluster labels



(b) The partition from sparse 2-means



(c) The partition from robust sparse 2-means

Figure 1.4: The simulated dataset consists of two clustering features, 1000 noise features and more than 10 outliers.

outliers and identifies the two true clusters.

The rest of this thesis is organized as follows. In Chapter 2, we review some existing clustering methods: K-means, sparse K-means and trimmed K-means are discussed there. In Chapter 3, robust sparse K-means is proposed. The performance of the proposed algorithm is assessed using several simulated datasets.

Chapter 4 discusses methods to select the appropriate number of clusters. Chapter 5 contains the application of the robust sparse K-means method to a microarray dataset. The conclusion is in Chapter 6.

Chapter 2

Literature review of clustering methods

2.1 Introduction

In this chapter we review three clustering methods which provide the foundation for our proposed robust sparse procedure: K-means, sparse K-means and trimmed K-means. Let $\mathbf{X} \in \mathbb{R}^{N \times p}$ denote the data matrix with N cases and p features and let $\mathbf{x}_i \in \mathbb{R}^p$ denote the i^{th} row of $\mathbf{X}$ ($i = 1, \cdots, N$). Moreover, let $\mathbf{X}_j \in \mathbb{R}^N$ denote the j^{th} column of $\mathbf{X}$ ($j = 1, \cdots, p$). That is:

$$\mathbf{X} = [\mathbf{X}_1 \cdots \mathbf{X}_p] = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & & \vdots \\ x_{N1} & \cdots & x_{Np} \end{bmatrix}.$$

A cluster can be described as a group of 'similar' cases. The cases within a cluster should be more 'similar' to each other than to cases from other clusters. It is immediately clear that partitions depend on the definition of 'dissimilarity'. Seber [21] defines a function $d : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}_+$ to be a dissimilarity if $d(\mathbf{a}, \mathbf{a}) = 0$, $d(\mathbf{a}, \mathbf{b}) \geq 0$ and $d(\mathbf{a}, \mathbf{b}) = d(\mathbf{b}, \mathbf{a})$ for all $\mathbf{a}$ and $\mathbf{b} \in \mathbb{R}^p$. It measures how different $\mathbf{a}$ and $\mathbf{b}$ are. Note that all distance measures are dissimilarity measures. The most intuitive choice for a dissimilarity measure is the Euclidean distance or the squared

Euclidean distance. The Euclidean distance between two cases is the length of the line segment connecting them. The Euclidean distance between vectors $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ is defined as:

$$\begin{aligned} ||\mathbf{x}_i - \mathbf{x}_{i'}|| &= \sqrt{(\mathbf{x}_i - \mathbf{x}_{i'})^T(\mathbf{x}_i - \mathbf{x}_{i'})} \\ &= \sqrt{(x_{i1} - x_{i'1})^2 + \cdots + (x_{ip} - x_{i'p})^2} \\ &= \sqrt{\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2}. \end{aligned} \tag{2.1}$$

This distance takes into account the differences between $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ directly, based on the magnitude of the coordinates in the difference vector. The squared Euclidean distance is simply $||\mathbf{x}_i - \mathbf{x}_{i'}||^2$. The squared Euclidean distance is additive in the features. This additivity will play a central role for sparse K-means clustering (see Section 2.3). For this reason, we mostly employ the squared Euclidean distance as a dissimilarity measure in this thesis.

The squared Euclidean distance between two cases assigns equal weights to all the squared differences along features:

$$||\mathbf{x}_i - \mathbf{x}_{i'}||^2 = 1 \cdot (x_{i1} - x_{i'1})^2 + \cdots + 1 \cdot (x_{ip} - x_{i'p})^2.$$

It implicitly assumes that units of all the features are the same. That is, without appropriate normalization of a dataset, the variable with the largest scale may dominate the distance. Let $\mathbf{X}^* = [\mathbf{X}_1^* \cdots \mathbf{X}_p^*] = [x_{ij}^*] \in \mathbb{R}^{N \times p}$ denote a nonnormalized dataset. From here, let $\mathbf{X} \in \mathbb{R}^{N \times p}$ denote the normalized dataset:

$$x_{i,j} = \frac{x_{ij}^* - \bar{x}_j^*}{sd_j}, \tag{2.2}$$

where

$$\bar{x}_j^* = \frac{\sum_{i=1}^N x_{ij}^*}{N}, \text{ and } sd_j = \sqrt{\frac{\sum_{i=1}^N (x_{ij}^* - \bar{x}_j^*)^2}{N-1}}.$$

Then clearly, all the features of $\mathbf{X}$ have unit sample standard deviation. One can

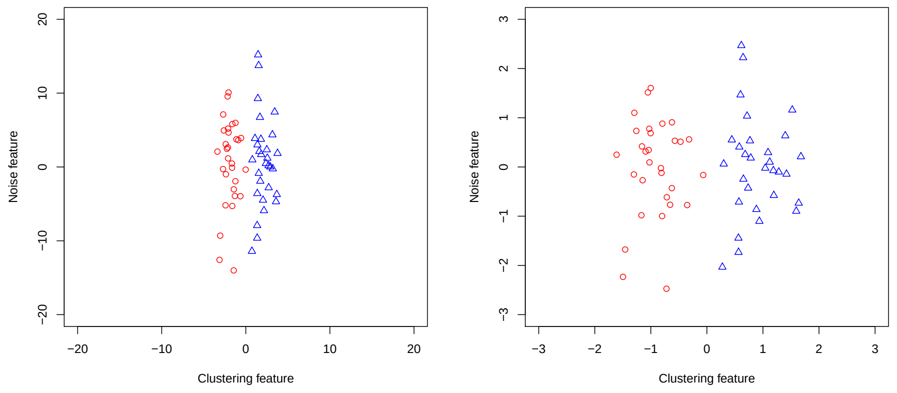


Figure 2.1: Scatter Plots with and without normalization of datasets.

see the necessity of normalization of the dataset by a simple simulated example in Figure 2.1. It shows 60 cases from two clusters, generated from two bivariate normal distributions. The two clusters are separated only by a clustering feature. The marginal variance of clustering feature is 1 while the marginal variance of the noise feature is set to 40. The left panel shows a nonnormalized raw dataset and the right panel shows its normalized dataset. Since the scale of the noise feature is $\sqrt{40}$ times larger, the two clusters are almost hidden in the left panel. However, after the normalization, one can see the two clusters more clearly.

For these reasons, normalization is critical for the squared Euclidean dissimilarity measure. Throughout this paper, we assume that the dataset $\mathbf{X}$ has already been normalized.

2.2 *K*-means

The term '*K*-means' was first used by MacQueen [16] in 1967 but the idea goes back to Steinhaus [23] in 1956.

2.2.1 The method

K-means partitions N cases into a predetermined number, K, of clusters ($1 \leq K \leq N$). The goal of K-means is to find the partition of cases which minimizes the sum of the dissimilarities between cases within each cluster.

Let C_k denote the set of the case indices in the k^{th} cluster and set $|C_k| = n_k$, where $|A|$ is the number of elements in set A. Then $\mathscr{C} = \{C_1, .., C_K\}$ is a partition of the dataset into K clusters. For any partition $\mathscr{C}$ and dissimilarity d, we can define the overall within cluster dissimilarity:

$$OWCD(\mathscr{C}) = \sum_{k=1}^{K} \sum_{i,i' \in C_k} d(\mathbf{x}_i, \mathbf{x}_{i'}).$$

An issue may arise if one looks for the partition $\mathscr{C}$ that minimizes $OWCD(\mathscr{C})$: partitions $\mathscr{C}$ with same cluster sizes, $n_1 = \cdots = n_K$, may be favored even if there is a clear cluster structure with different cluster sizes. To see this, if the $\tilde{i}^{th}$ case is clustered in $C_{\tilde{k}}$, then there are $2n_{\tilde{k}}$ distance components in the sum of $OWCD(\mathscr{C})$ associated with the $\tilde{i}^{th}$ case. This is because each distance is counted twice and the distance of itself, $d(\mathbf{x}_{\tilde{i}}, \mathbf{x}_{\tilde{i}})$, is also counted. Suppose there is another cluster $C_{k'}$ whose cluster size is considerably smaller than the cluster size of $C_{\tilde{k}}$: $n_{\tilde{k}} \gg n_{k'}$. Then even if the cases in $C_{k'}$ are more dissimilar to the $\tilde{i}^{th}$ case than the cases in $C_{\tilde{k}}$, the partition that minimizes $OWCD(\mathscr{C})$ would prefer to have the $\tilde{i}^{th}$ to be clustered in $C_{k'}$. This is because by doing so the number of distance components decreases in $OWCD(\mathscr{C})$. In order to solve this issue, we can define the within cluster dissimilarity as:

$$WCD(\mathscr{C}) = \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{i,i' \in C_k} d(\mathbf{x}_i, \mathbf{x}_{i'}).$$

The goal of K-means is to find a partition $\mathscr{C}^*$ that minimizes $WCD(\mathscr{C})$. That is:

$$\mathscr{C}^* = \underset{\mathscr{C};\ |\mathscr{C}|=K}{\operatorname{argmin}}\ WCD(\mathscr{C}). \tag{2.3}$$

When the dissimilarity measure is the squared Euclidean distance, $WCD(\mathscr{C})$ is called within cluster sum of Squares:

$$WSS(\mathscr{C}) = \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{i,i' \in C_k} ||\mathbf{x}_i - \mathbf{x}_{i'}||^2.$$

Interestingly, $WSS(\mathscr{C})$ can be rewritten in terms of the distances to the k^{th} cluster center if one defines a center of the k^{th} cluster as the average vector over the cases within the k^{th} cluster:

$$\bar{\mathbf{x}}_k = \frac{1}{n_k} \sum_{i \in C_k} \mathbf{x}_i.$$

$WSS(\mathscr{C})$ can be expressed in terms of the cluster centers as follows:

$$\begin{aligned}
WSS(\mathscr{C}) &= \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{i,i' \in C_k} ||\mathbf{x}_i - \mathbf{x}_{i'}||^2 \\
&= \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{i,i' \in C_k} ||\mathbf{x}_i - \bar{\mathbf{x}}_k + \bar{\mathbf{x}}_k - \mathbf{x}_{i'}||^2 \\
&= \sum_{k=1}^{K} \frac{1}{2n_k} \Bigg(n_k \sum_{i \in C_k} ||\mathbf{x}_i - \bar{\mathbf{x}}_k||^2 \\
&\quad + \sum_{i \in C_k} (\mathbf{x}_i - \bar{\mathbf{x}}_k)^T \sum_{i' \in C_k} (\bar{\mathbf{x}}_k - \mathbf{x}_{i'}) + n_k \sum_{i' \in C_k} ||\bar{\mathbf{x}}_k - \mathbf{x}_{i'}||^2 \Bigg) \\
&= \sum_{k=1}^{K} \sum_{i \in C_k} ||\mathbf{x}_i - \bar{\mathbf{x}}_k||^2. \qquad (2.4)
\end{aligned}$$

Then K-means with squared Euclidean distance solves the minimization problem:

$$\min_{\mathscr{C};\, |\mathscr{C}|=K} WSS(\mathscr{C}) = \min_{\mathscr{C};\, |\mathscr{C}|=K} \sum_{k=1}^{K} \sum_{i \in C_k} ||\mathbf{x}_i - \bar{\mathbf{x}}_k||^2. \qquad (2.5)$$

Use of the expression in (2.5) with the mean term facilitates the computation. This is one of the reasons why the squared Euclidean distance is a popular choice for the dissimilarity measure.

2.2.2 The standard computing algorithm (Lloyd's algorithm) and Hartigan and Wong's algorithm

No analytic solution is available for (2.3). One might calculate the values of the objective functions for all the partitions. However, even with K=2 one has to consider $\sum_{i=1}^{\lfloor N/2 \rfloor} \binom{N}{i}$ possible partitions. The number of partition grows rapidly with the sample size, N. Various algorithms have been proposed to achieve a *local* optimum of (2.3). The most popular ones are due to Lloyd [14], Hartigan and Wong [11], MacQueen [16] and Forgy [6]. All these algorithms adopt the squared Euclidean distance as a dissimilarity measure. MacKay [15] stated that the most commonly used standard iterative algorithm was that proposed by Lloyd [14]. The trimmed K-means algorithm, introduced in Section 2.4, robustifies this method. The `kmeans` command in the `R` package `stat` uses Hartigan and Wong's algorithm by default. This algorithm made Lloyd's algorithm (Lloyd [14]) more efficient and generally achieves better local optima than other methods. Sparse K-means, introduced later, uses Hartigan and Wong's algorithm. First, we introduce Lloyd's algorithm then we will introduce Hartigan and Wong's algorithm, which is substantially more complicated.

2.2.2.1 The standard computing algorithm (Lloyd's algorithm)

Here, we use another notation, $IC1(i)$, to describe a partition $\mathscr{C}$. $IC1(i)$ represents the index of the cluster to which the i^{th} case belongs. That is, $IC1(i) = k$ is equivalent to $i \in C_k$. Randomly choose K cases to serve as initial cluster centers and assign each case to its closest initial cluster center. Repeat 1 and 2 below until convergence.

- Step (a): Given a cluster assignment, update new cluster centers equal to the cluster means, $\bar{\mathbf{x}}_{\mathbf{k}}$ for $k = 1, \cdots, K$.

- Step (b): Given cluster centers, update a class assignment by assigning each case to its closest cluster center. The index of the cluster to which the i^{th} case belongs is denoted by $IC1(i) = \underset{1 \le k \le K}{\operatorname{argmin}} ||\mathbf{x}_i - \bar{\mathbf{x}}_k||^2 \;\; \forall i \in \{1, \cdots, N\}$.

Since each iteration minimizes the objective function, the algorithm converges to a local minimum. The solutions can differ from each other depending on the random selection of initial cluster centers. The user of the K-means algorithm should try several initial cluster centers and choose the partition that corresponds to the local minimum with the smallest objective function, $WSS(\mathscr{C})$, among all the local minimums. Also, this algorithm can return a partition with less than K clusters. This may happen when some cluster centers do not have any case that is closest to them.

2.2.2.2 Hartigan and Wong's algorithm

Hartigan and Wong's algorithm differs from Lloyd's algorithm mainly in three ways. First, they differ in the reassignment criteria. Lloyd's algorithm reassigns a case from the k_1^{th} cluster to the k_2^{th} cluster if it is closer to the k_2^{th} cluster center. Therefore, Step (b) of Lloyd's algorithm compares the distances:

$$||\mathbf{x}_i - \bar{\mathbf{x}}_{k_2}||^2 < ||\mathbf{x}_i - \bar{\mathbf{x}}_{k_1}||^2. \tag{2.6}$$

However, Hartigan and Wong's algorithm uses a different criterion. Sparks (1973) proposed that it is more effective to reassign the case if the squared Euclidean distance from the center of the k_2^{th} cluster is less than that from the center of the k_1^{th} cluster, even when the cluster centers are simultaneously repositioned (Sparks [22]). That is, when:

$$\frac{n_{k_2}}{n_{k_2}+1}||\mathbf{x}_i - \bar{\mathbf{x}}_{k_2}||^2 < \frac{n_{k_1}}{n_{k_1}-1}||\mathbf{x}_i - \bar{\mathbf{x}}_{k_1}||^2. \tag{2.7}$$

Criterion (2.6) is a sufficient condition for criterion (2.7) because $\frac{n_{k_2}}{n_{k_2}+1} < 1 < \frac{n_{k_1}}{n_{k_1}-1}$. Therefore, reassignments of cases occur less frequently with (2.6) than with (2.7).

Second, Hartigan and Wong's algorithm is more efficient. The Lloyd's algorithm is inefficient in the sense that it computes the squared Euclidean distances between *all* cases and *all* cluster centers until convergence. Consider the situation when K=4 and cluster C_1 and C_2 are clearly defined while remaining clusters, C_3 and C_4, have a large overlap. Then the cases in C_1 and C_2 would stop switching between clusters at an early stage while the cases in C_3 and C_4 would keep switching clusters. Lloyd's algorithm, however, calculates the squared Euclidean distances of *all* cases, including the cases in C_1 and C_2, to *all* cluster centers until *all* cases stop switching between clusters (convergence). Hartigan and Wong's algorithm recomputes the squared Euclidean distances between a case and cluster centers that have the potential to be its new cluster centers. Hartigan and Wong define a set, the ‘‘live set’’, that contains all the clusters whose cluster centers have the potential to change. By treating the cases in clusters that belong to the live set differently from others, Hartigan and Wong's algorithm became computationally more efficient.

Third, Hartigan and Wong's algorithm achieves a better local optimum. In Lloyd's algorithm, when each case is examined to see if it should be reassigned to a different cluster at Step (b), only the closest cluster center is used to check for the possible reallocation. However, Hartigan and Wong's algorithm examine not only the closest cluster center but also the second closest center. The details of how they examine the two cluster centers and the algorithm are described in Appendix A.1.

Hartigan and Wong [11] showed that their algorithm achieves a better local optimum than Lloyd's algorithm in general. Therefore, we will use Hartigan and Wong's algorithm to compute K-means and sparse K-means throughout this thesis.

2.2.3 Measurements of the agreement between two partitions

In order to assess the performance of different cluster algorithms we need to measure how well two given partitions $\mathscr{C}$ and $\tilde{\mathscr{C}}$ agree. For instance, in a simulation study we would like to compare the partitions produced by the different clustering algorithms with the ‘‘true partition’’ generated by the model. Given two cases and two partitions $\mathscr{C}$ and $\tilde{\mathscr{C}}$, there are four possible scenarios (Hubert [12]):

Type 1: the two cases belong to a single cluster in both $\mathscr{C}$ and $\tilde{\mathscr{C}}$,

Type 2: the two cases belong to different clusters in both $\mathscr{C}$ and $\tilde{\mathscr{C}}$,

Type 3: the two cases belong to different clusters in $\mathscr{C}$ but to the same cluster in $\tilde{\mathscr{C}}$,

Type 4: the two cases belong to the same cluster in $\mathscr{C}$ but to different clusters in $\tilde{\mathscr{C}}$.

Cases of types 1 and 2 suggest agreement of the two partitions while cases of types 3 and 4 suggest disagreement. This definition is used in the famous Rand index (Rand [19]), as well as in the classification error rate (CER) (Chipman and

	$I_{\mathscr{C}}(i,i')$	$I_{\tilde{\mathscr{C}}}(i,i')$	$\lvert I_{\mathscr{C}}(i,i') - I_{\tilde{\mathscr{C}}}(i,i')\rvert$
Type 1	1	1	0
Type 2	0	0	0
Type 3	0	1	1
Type 4	1	0	1

Table 2.1: The pair types and corresponding values of the indicator functions, $I_{\mathscr{C}}(i,i')$ and $I_{\tilde{\mathscr{C}}}(i,i')$.

Tibshirani [2]). The CER is simply the proportion of pairs of cases that are together in one partition and apart in the other partition, i.e., Type 3 or Type 4 pairs of cases. On the other hand, the Rand Index is the proportion of Type 1 or Type 2 pairs of cases. Therefore, CER = 1 - Rand index. Throughout this thesis, we will use CER as a measurement of agreement of two partitions. The calculation of CER is described in the following section.

2.2.3.1 The calculation of CER

CER was first proposed by Chipman et al. [3] in the context of decision trees. Later, Chipman and Tibshirani [2] applied this measurement for clustering. Let $I_{\mathscr{C}}$ be an indicator function, $I_{\mathscr{C}} : \{1,\cdots,N\} \times \{1,\cdots,N\} \rightarrow \{0,1\}$. Given the pair of case indices, (i,i'), the indicator function returns 1 if the i^{th} case and the i'^{th} case are paired in partition $\mathscr{C}$ and returns 0 if they are apart:

$$I_{\mathscr{C}}(i,i') = \begin{cases} 1 & \text{if } i \text{ and } i' \text{ belong to the same cluster in partition } \mathscr{C}, \\ 0 & \text{otherwise.} \end{cases}$$

Given two partitions $\mathscr{C}$ and $\tilde{\mathscr{C}}$, the four types of the pairs can be represented by the output of two indicator functions $I_{\mathscr{C}}(i,i')$ and $I_{\tilde{\mathscr{C}}}(i,i')$ as in Table 2.1. Then the absolute difference of the two indicator functions, $I_{\mathscr{C}}$ and $I_{\tilde{\mathscr{C}}}$ is zero if the pairs of cases are apart or together in both partitions while it is one if the pair is apart in one partition and together in another.

CER is the percentage of the pairs of cases that are Type 3 or Type 4:

$$CER(\mathscr{C},\mathscr{C}') = \frac{\sum_{i>i'} |I_{\mathscr{C}}(i,i') - I_{\tilde{\mathscr{C}}}(i,i')|}{\binom{N}{2}}.$$

Note that there are $\binom{N}{2}$ pairs of cases and this factor scales the range of CER to be between zero and one. $CER(\mathscr{C},\tilde{\mathscr{C}})$ equals zero if the two partitions perfectly agree and becomes one if the two partitions completely disagree (Chipman and Tibshirani [2]). For example, if $\mathscr{C}$ contains a single cluster, $|\mathscr{C}| = 1$, while $\tilde{\mathscr{C}}$ contains as many clusters as sample size, $|\tilde{\mathscr{C}}| = N$, then all the pairs of clusters are Type 4 and CER becomes 1.

2.2.4 Small simulation studies of *K*-means

Since *K*-means is easy to understand (minimizes a simple objective function) and readily available in many statistical packages, this algorithm has been a popular choice in many fields. In fact, *K*-means performs well to identify clusters when the datasets do not contain noise features. To see this, we report on a small simulation study with the following simulation model. We will refer to this model as ‘‘clean model’’. A dataset generated from the clean model contains 60 cases $\mathbf{x}_i = [x_{i1},\cdots,x_{i50}]^T$ $(i = 1,\cdots,60)$. The cases are generated from 3 clusters as $x_{ij} = \mu_i + \varepsilon_{ij}$, with $\varepsilon_{ij} \overset{i.i.d}{\sim} N(0,1)$ where:

$$\mu_i = \begin{cases} \mu & \text{if } 1 \le i \le 20, \\ 0 & \text{if } 21 \le i \le 40, \\ -\mu & \text{if } 41 \le i \le 60. \end{cases}$$

Note that all the 50 features are clustering features. We consider different values for μ in our simulations ($\mu = 1$ and 0.8). The magnitude of μ controls how close the clusters are. In this simulation model, the true partition of cases is $\mathscr{C}' = \{C_1,C_2,C_3\}$ where $C_1 = \{1,\cdots,20\}, C_2 = \{21,\cdots,40\}$ and $C_3 = \{41,\cdots,60\}$.

We generate 100 datasets from the clean model for each μ. Then *K*-means with *K*=3 is performed on each dataset. *K*-means with *K* predetermined to be 3 is called 3-means. Harigan and Wong’s algorithm is used, as implemented in the `R`

μ	CER
1.0	0.00 (0.003)
0.8	0.01 (0.016)

Table 2.2: The simulation results of 3-means on 100 datasets generated from the clean model.

package `stat` as a default method. The performance of 3-means is assessed by CER between the true partition $\mathscr{C}'$ and the partition returned by the algorithm.

The average and the standard deviation of CERs over 100 simulations are reported in Table 2.2. The CERs are nearly 0 for both μs, indicating that 3-means can recover the true cluster structure nearly perfectly.

A drawback of K-means is that it uses all p features to obtain partitions regardless of their importance for clustering. Since K-means does not automatically perform feature selection, the user must decide by himself which features are the clustering features. To illustrate this issue, we examine the performance of 3-means in a dataset generated from a ‘‘noisy model’’. The dataset generated from the noisy model consists of 60 cases, $\mathbf{x}_i = [x_{i1}, \cdots, x_{i500}]^T$ and 500 features ($i = 1, \cdots, 60$). The first 50 features are clustering features, generated as in the clean model. The last 450 features are noise features. That is for $j = 51, \cdots, 500$, x_{ij} is defined as $x_{ij} = \varepsilon_{ij}$.

μ	CER
1.0	0.02 (0.020)
0.8	0.10 (0.061)

Table 2.3: The simulation results of 3-means on 100 datasets generated from the noisy model.

We generate 100 datasets from the noisy model for each μ ($\mu = 1$ or 0.8) and perform 3-means on each dataset, using Harigan and Wong’s algorithm. The average CER values are reported in Table 2.3. The result is shown in Table 2.3. The performance of 3-means becomes poorer on datasets from the noisy model than on datasets from the clean model.

2.3 Sparse K-means

Our small simulation study above shows that the performance of K-means may deteriorate when datasets contain many noise features. To address this problem, Witten and Tibshirani [28] proposed the sparse K-means algorithm which clusters the cases using an adaptively chosen subset of the features. Sparse K-means is based on the fact that the problem of K-means can be viewed as a maximization problem. Moreover, it hinges critically on the assumption that the dissimilarity is additive in each feature. Before moving onto the discussion of sparse K-means, we introduce this different representation of the problem of K-means in Section 2.3.1. We will explain that some dissimilarities can be decomposed by each feature in Section 2.3.2.

2.3.1 K-means as a maximization problem

We will show below that minimizing the overall within cluster dissimilarity is the same as maximizing the overall between cluster dissimilarity. The overall between cluster dissimilarity is defined as:

$$OBCD(\mathscr{C}) = \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{i' \notin C_k} d(\mathbf{x}_i, \mathbf{x}_{i'}).$$

Moreover, we define the overall total cluster dissimilarity to be the sum of the dissimilarities of all the pairs of cases:

$$OTCD = \sum_{i=1}^{N} \sum_{i'=1}^{N} d(\mathbf{x}_i, \mathbf{x}_{i'}). \tag{2.8}$$

Therefore, $OTCD$ does not depend on a partition, $\mathscr{C}$.

Now we will show that $OBCD(\mathscr{C})$ is simply the difference between the overall total cluster dissimilarity and the overall within cluster dissimilarity:

$$OBCD(\mathscr{C}) = OTCD - OWCD(\mathscr{C}).$$

To see this:

$$\begin{aligned}
OTCD &= \sum_{i=1}^{N}\sum_{i'=1}^{N} d(\mathbf{x}_i,\mathbf{x}_{i'}) \\
&= \sum_{k=1}^{K}\sum_{i\in C_k}\sum_{i'=1}^{N} d(\mathbf{x}_i,\mathbf{x}_{i'}) \\
&= \sum_{k=1}^{K}\sum_{i\in C_k}\left(\sum_{i'\notin C_k} d(\mathbf{x}_i,\mathbf{x}_{i'}) + \sum_{i'\in C_k} d(\mathbf{x}_i,\mathbf{x}_{i'})\right) \\
&= \sum_{k=1}^{K}\sum_{i\in C_k}\sum_{i'\notin C_k} d(\mathbf{x}_i,\mathbf{x}_{i'}) + \sum_{k=1}^{K}\sum_{i\in C_k}\sum_{i'\in C_k} d(\mathbf{x}_i,\mathbf{x}_{i'}) \\
&= OWCD(\mathscr{C}) + OBCD(\mathscr{C}). \qquad (2.9)
\end{aligned}$$

As in (2.8), the total cluster dissimilarity can be defined as:

$$TCD = \frac{1}{2N}\sum_{i=1}^{N}\sum_{i'=1}^{N} d(\mathbf{x}_i,\mathbf{x}_{i'}).$$

Then it is natural from (2.9) to define the between cluster dissimilarity:

$$\begin{aligned}
BCD(\mathscr{C}) &= TCD - WCD(\mathscr{C}) \\
&= \frac{1}{2N}\sum_{i=1}^{N}\sum_{i'=1}^{N} d(\mathbf{x}_i,\mathbf{x}_{i'}) - \sum_{k=1}^{K}\frac{1}{2n_k}\sum_{i,i'\in C_k} d(\mathbf{x}_i,\mathbf{x}_{i'}).
\end{aligned}$$

With the definition of $WCD(\mathscr{C})$, $BCD(\mathscr{C})$ and TCD, now we can show that the minimization problem of K-means (2.3) can be rewritten as a maximization problem:

$$\begin{aligned}
\mathscr{C}^* &= \underset{\mathscr{C};\,|\mathscr{C}|=K}{\text{argmin}}\; WCD(\mathscr{C}) \\
&= \underset{\mathscr{C};\,|\mathscr{C}|=K}{\text{argmin}}\; \{TCD - BCD(\mathscr{C})\} \\
&= \underset{\mathscr{C};\,|\mathscr{C}|=K}{\text{argmin}}\; \{-BCD(\mathscr{C})\} \quad \text{(since } TCD \text{ does not depend on } \mathscr{C}\text{)} \\
&= \underset{\mathscr{C};\,|\mathscr{C}|=K}{\text{argmax}}\; BCD(\mathscr{C}). \qquad (2.10)
\end{aligned}$$

For the squared Euclidean dissimilarity measure, the total cluster dissimilarity is called the total cluster sum of squares (TSS) and the between cluster dissimilarity is called the between cluster sum of square (BSS). Then we have:

$$\begin{aligned} TSS &= \frac{1}{2N}\sum_{i=1}^{N}\sum_{i'=1}^{N}||\mathbf{x}_i - \mathbf{x_{i'}}||^2, \\ BSS(\mathscr{C}) &= TSS - WSS(\mathscr{C}). \end{aligned} \tag{2.11}$$

As $WSS(\mathscr{C}), TSS(\mathscr{C})$ can also be expressed in terms of the distances to the cluster centers:

$$\begin{aligned} TSS &= \frac{1}{2N}\sum_{i=1}^{N}\sum_{i'=1}^{N}||\mathbf{x}_i - \bar{\mathbf{x}} + \bar{\mathbf{x}} - \mathbf{x_{i'}}||^2 \\ &= \frac{1}{2N}\left(N\sum_{i=1}^{N}||\mathbf{x}_i - \bar{\mathbf{x}}||^2 + \sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})^T\sum_{i'=1}^{N}(\bar{\mathbf{x}} - \mathbf{x_{i'}}) + N\sum_{i'=1}^{N}||\bar{\mathbf{x}} - \mathbf{x_{i'}}||^2\right) \\ &= \sum_{i=1}^{N}||\mathbf{x}_i - \bar{\mathbf{x}}||^2. \end{aligned} \tag{2.12}$$

Therefore, $BSS(\mathscr{C})$ can be also expressed in terms of the distances to the cluster centers:

$$BSS(\mathscr{C}) = \sum_{i=1}^{N}||\mathbf{x}_i - \bar{\mathbf{x}}||^2 - \sum_{k=1}^{K}\sum_{i\in C_k}||\mathbf{x}_i - \bar{\mathbf{x}}_k||^2.$$

Then the minimization problem of K-means with the squared Euclidean dissimilarity measure (2.5) is equivalent to the maximization problem:

$$\mathscr{C}^* = \underset{\mathscr{C}}{\operatorname{argmax}}\, BSS(\mathscr{C}). \tag{2.13}$$

This maximization approach to the K-means problem is used in the sparse K-means algorithm.

2.3.2 The additivity of dissimilarities in each feature

Sparse K-means hinges critically on the assumption that the dissimilarity measures can be represented as a sum of terms which depend on a single feature each. In other words, sparse K-means assumes that $BCD(\mathscr{C})$ can be decomposed as follows:

$$\begin{aligned}
WCD(\mathscr{C}) &= \sum_{j=1}^{p} WCD_j(\mathscr{C}),\\
TCD &= \sum_{j=1}^{P} TCD_j,\\
BCD(\mathscr{C}) &= \sum_{j=1}^{p} BCD_j(\mathscr{C}).
\end{aligned}$$

where $WCD_j(\mathscr{C}), TCD_j$ and $BCD_j(\mathscr{C})$ depend only on the j^{th} column of $\mathbf{X} \in \mathbb{R}^{N \times p}$. This assumption is satisfied if the dissimilarity measure is additive in the features:

$$d(\mathbf{x}_i, \mathbf{x}_{i'}) = \sum_{j=1}^{p} d_j(\mathbf{x}_i, \mathbf{x}_{i'}).$$

Note that $d_j(\mathbf{x}_i, \mathbf{x}_{i'})$ is the dissimilarity between $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ along the j^{th} feature. Our derivations below show that this assumption holds for the squared Euclidean distances:

$$\begin{aligned}
WSS(\mathscr{C}) &= \sum_{k=1}^{K} \sum_{i \in C_k} ||\mathbf{x}_i - \bar{\mathbf{x}}_k||^2 \text{ (From (2.4))}\\
&= \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{k,j})^2 \text{ where } \bar{x}_{k,j} = \frac{1}{n_k} \sum_{i \in C_k} x_{ij}\\
&= \sum_{j=1}^{p} \left\{ \sum_{k=1}^{K} \sum_{i \in C_k} (x_{ij} - \bar{x}_{k,j})^2 \right\}\\
&= \sum_{j=1}^{p} WSS_j(\mathscr{C})
\end{aligned}$$

$$
\begin{aligned}
TSS &= \sum_{i=1}^{N} ||\mathbf{x}_i - \bar{\mathbf{x}}||^2 \text{ (From (2.12))} \\
&= \sum_{i=1}^{N}\sum_{j=1}^{p} (x_{ij} - \bar{x}_j)^2 \text{ where } \bar{x}_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \\
&= \sum_{j=1}^{p} \left\{ \sum_{i=1}^{N} (x_{ij} - \bar{x}_j)^2 \right\} \\
&= \sum_{j=1}^{p} TSS_j.
\end{aligned}
$$

Finally,

$$
\begin{aligned}
BSS(\mathscr{C}) &= WSS - TSS \text{ (From (2.11))} \\
&= \sum_{j=1}^{p} \left\{ TSS_j - WSS_j(\mathscr{C}) \right\} \\
&= \sum_{j=1}^{p} BSS_j(\mathscr{C}).
\end{aligned}
$$

Notice that $BSS_j(\mathscr{C})$, can be interpreted as the contribution of the j^{th} feature to the separation between the given K clusters. We can expect clustering features to have large values of $BSS_j(\mathscr{C})$ while noise features should have small values. In Section 2.3.3, we will introduce the sparse K-means with the squared Euclidean dissimilarity measure.

2.3.3 The sparse K-means clustering algorithm

Now we are ready to introduce sparse K-means. More precisely, sparse K-means is defined as the solution to the problem:

$$
\max_{\mathscr{C},\mathbf{w};\, |\mathscr{C}|=K} \sum_{j=1}^{p} w_j BSS_j(\mathscr{C}) \;\; s.t. \;\; ||\mathbf{w}||_1 \le l_1, ||\mathbf{w}||^2 \le 1, w_j \ge 0 \;\; \forall j, \qquad (2.14)
$$

where $\mathbf{w} = [w_1, \cdots, w_p]^T$. Note that $||\mathbf{a}||_1$ represents the L_1 norm of the vector $\mathbf{a}$. The idea is that the algorithm returns sparse weights such that only clustering features receive nonzero weights. The L_1 penalty on $\mathbf{w}$ results in sparsity for small values of the tuning parameter l_1. The L_1 tuning parameter value should be selected between 1 and $\sqrt{p}$ to have sparse weights. The L_2 penalty also serves an important role, since without it, at most one element of $\mathbf{w}$ would be nonzero in general. This point is discussed in Section 2.3.4.

Witten and Tibshirani [28] proposed an algorithm to solve (2.14): the algorithm simply maximizes (2.14) over $\mathscr{C}$ and $\mathbf{w}$ iteratively.

Initially choose equal weights for all features, i.e. let $\mathbf{w} = 1/p\mathbf{1}$ where $\mathbf{1} = [1, \cdots, 1]^T \in \mathbb{R}^p$. Repeat steps (a) and (b) below until convergence.

- Step (a): For a given $\mathbf{w}$, maximize (2.14) with respect to $\mathscr{C}$. That is, perform K-means on the transformed dataset, $\mathbf{Y}$:

  $$\underset{\mathscr{C};\,|\mathscr{C}|=K}{\text{argmin}}\ WSS_{\mathbf{Y}}(\mathscr{C}). \tag{2.15}$$

  where $WSS_{\mathbf{Y}}(\mathscr{C})$ represents the within cluster sum of squares calculated on the dataset $\mathbf{Y} \in \mathbb{R}^{N \times p}$. The columns of $\mathbf{Y}$ are the columns of $\mathbf{X}$ multiplied by the corresponding weights. The transformation of the dataset into $\mathbf{Y}$ is explained in Section 2.3.3.1.

- Step (b): For a given $\mathscr{C}$, maximize (2.14) with respect to $\mathbf{w}$:

  $$\max_{\mathbf{w}} \mathbf{w}^T\mathbf{D} \quad s.t.\ ||\mathbf{w}||^2 \leq 1, ||\mathbf{w}||_1 \leq l_1, w_j \geq 0\ \forall j, \tag{2.16}$$

  where $\mathbf{D} = [BSS_1(\mathscr{C}), \cdots, BSS_p(\mathscr{C})]^T$. The analytic solution for this optimization problem, given by Witten and Tibshirani [28], is $\mathbf{w}(\triangle*)$ where:

  $$\mathbf{w}(\triangle) = \frac{(\mathbf{D} - \triangle\mathbf{1})_+}{||(\mathbf{D} - \triangle\mathbf{1})_+||}, \tag{2.17}$$

and

$$\triangle^* = \begin{cases} 0 & \text{if } ||\mathbf{w}(0)||_1 \leq l_1, \\ \text{root of } \ ||\mathbf{w}(\triangle)||_1 - l_1 = 0 & \text{otherwise} \end{cases}.$$

Note that $a_+ = a$ if $a \geq 0$ and $a_+ = 0$ if $a < 0$.

2.3.3.1 The transformation of the dataset X into Y

Sparse K-means aims at finding clusters and weights that simultaneously maximize the sum of the dissimilarity measures across clusters. If weights were fixed, then (2.14) becomes the objective function of K-means with weighted within cluster sum of squares:

$$\underset{\mathscr{C};\,|\mathscr{C}|=K}{\text{argmax}} \sum_{j=1}^{p} w_j BSS_j(\mathscr{C}) = \underset{\mathscr{C};\,|\mathscr{C}|=K}{\text{argmin}} \sum_{j=1}^{p} w_j WSS_j(\mathscr{C}). \tag{2.18}$$

One can see this problem as a K-means problem on a transformed dataset $\mathbf{Y} \in \mathbb{R}^{N\times p}$. We transform the dataset, $\mathbf{X}$, into $\mathbf{Y}$ by scaling the j^{th} feature of the dataset $\mathbf{X}_j$ by $\sqrt{w_j}$. Specifically, let:

$$\begin{aligned} \mathbf{Y} = & \begin{bmatrix} y_{11} & \cdots & y_{1p} \\ \vdots & & \vdots \\ y_{N1} & \cdots & y_{Np} \end{bmatrix} \\ = & \begin{bmatrix} \sqrt{w_1}x_{11} & \cdots & \sqrt{w_p}x_{1p} \\ \vdots & & \vdots \\ \sqrt{w_1}x_{N1} & \cdots & \sqrt{w_p}x_{Np} \end{bmatrix} \\ = & [\sqrt{w_1}\mathbf{X}_1 \cdots \sqrt{w_p}\mathbf{X}_p]. \end{aligned}$$

The within cluster sum of squares calculated on the transformed dataset $\mathbf{Y}$ can be expressed as:

$$WSS_{\mathbf{Y}}(\mathscr{C}) = \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{i,i' \in C_k} ||\mathbf{y}_i - \mathbf{y}_{i'}||^2, \tag{2.19}$$

and is equivalent to the objective function on the left hand side in (2.18). To see this:

$$\begin{aligned}
WSS_{\mathbf{Y}}(\mathscr{C}) &= \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (y_{ij} - y_{i'j})^2 \\
&= \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (\sqrt{w_j} x_{ij} - \sqrt{w_j} x_{i'j})^2 \\
&= \sum_{k=1}^{K} \frac{1}{2n_k} \sum_{i,i' \in C_k} \sum_{j=1}^{p} w_j (x_{ij} - x_{i'j})^2 \\
&= \sum_{j=1}^{p} w_j \sum_{k=1}^{K} \sum_{i,i' \in C_k} \frac{1}{2n_k} (x_{ij} - x_{i'j})^2 \\
&= \sum_{j=1}^{p} w_j WSS_j(\mathscr{C}).
\end{aligned}$$

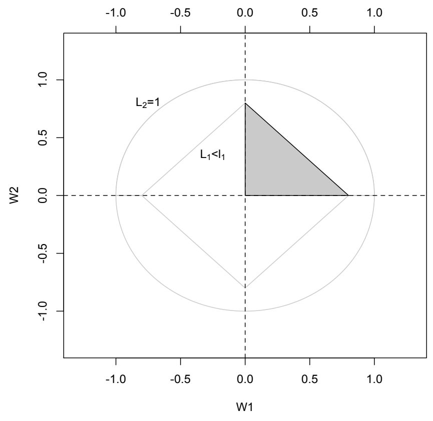
Therefore, for fixed weights, the problem of (2.14) becomes the problem of K-means on the transformed dataset $\mathbf{Y}$:

$$\underset{\mathscr{C};\, |\mathscr{C}|=K}{\operatorname{argmax}} \sum_{j=1}^{p} w_j BSS_j(\mathscr{C}) = \underset{\mathscr{C};\, |\mathscr{C}|=K}{\operatorname{argmin}} \; WSS_{\mathbf{Y}}(\mathscr{C}).$$

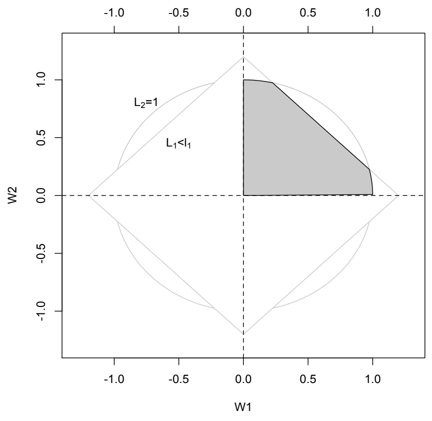
The `R` package `sparcl` contains the function `KMeanSparseCluster`, which performs sparse K-means. Although we can apply any standard implementation of K-means on the transformed dataset $\mathbf{Y}$ at Step (a), `KMeanSparseCluster` uses Hartigan and Wong's algorithm to run K-means on a transformed dataset $\mathbf{Y}$ at Step (a).

2.3.4 The L_1 tuning parameter values and the sparsity of features

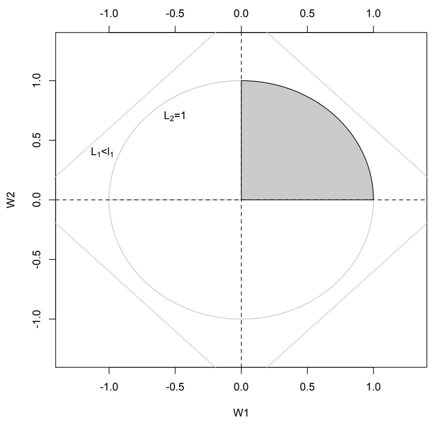
Sparse K-means requires users to prespecify the value of the tuning parameter l_1. The lasso-type penalty on $\mathbf{w}$ results in sparsity when the value of the L_1 tuning parameter value is sufficiently small. In Step (b), the objective function is maximized over the domain of the vector $\mathbf{w}$. Figure 2.2 shows the admissible region for $\mathbf{w}$ for different values of the upper bound l_1 for p=2. The L_2 constraint is the area inside of the unit circle and the L_1 constraint is the area inside of the diamond with y intercepts l_1 and $-l_1$. Since $w_j \geq 0$ for all j, the shaded region in Figure 2.2 is the



(a) $l_1 \leq 1$



(b) $1 \leq l_1 \leq \sqrt{2}$



(c) $\sqrt{2} < l_1$

Figure 2.2: Regions of L_1 and L_2 constraints in $\mathbb{R}^2$.

admissible region of $\mathbf{w}$.

The L_1 tuning parameter value should be selected between 1 and $\sqrt{p}$. When $l_1 \leq 1$, the diamond is inside of the unit circle and the L_2 constraint become redundant in the presence of the L_1 constraint. Therefore we can think of the problem of Step (b) (2.16) without the L_2 constraint (e.g. see the panel (a) of Figure 2.2). As a result, the feature with the largest $BSS_j(\mathscr{C})$ receives all the weight: $\mathbf{w} = l_1 \mathbf{e}_{j_0}$ where $BSS_{j_0}(\mathscr{C}) \geq BSS_j(\mathscr{C})$ for all $j = 1, \cdots, p$ and $\mathbf{e}_k$ is a standard vector in $\mathbb{R}^p$ To see this, without the L_2 constraint, the optimization problem at Step (b) becomes:

$$\max \ \mathbf{w}^T \mathbf{D} \, s.t. \, ||\mathbf{w}||_1 = w_1 + \cdots + w_p \leq l_1.$$

Without loss of generality, assume that $BSS_1(\mathscr{C}) \geq \cdots \geq BSS_p(\mathscr{C})$, then:

$$\begin{aligned}
\mathbf{w}^T \mathbf{D} &= w_1 BSS_1(\mathscr{C}) + \cdots + w_p BSS_p(\mathscr{C}) \\
&\leq \ w_1 BSS_1(\mathscr{C}) + \cdots + w_p BSS_1(\mathscr{C}) \\
&\leq \ (w_1 + \cdots + w_p) BSS_1(\mathscr{C}) \\
&= l_1 BSS_1(\mathscr{C}) \\
&= l_1 [1 \, 0 \cdots 0] \mathbf{D} \\
&= l_1 \mathbf{e}_1^T \mathbf{D}.
\end{aligned}$$

Therefore, when $l_1 \leq 1$, Step (b) of the sparse K-means algorithm returns the weights $\mathbf{w} = l_1 \mathbf{e}_1$. That is, 100% of the weight will be on the feature with the largest $BSS_j(\mathscr{C})$.

If $l_1 \geq \sqrt{p}$, then the L_1 constraint becomes redundant in the presence of the L_2 constraint (the panel (c) in Figure 2.2). As a result, all the features receive nonzero weights. Furthermore,the j^{th} feature receives the weight proportional to $BSS_j(\mathscr{C})$. To see this, without the L_1 constraint, the optimization problem at Step (b) becomes:

$$\max \ \mathbf{w}^T \mathbf{D} \, s.t. \, ||\mathbf{w}||^2 = w_1^2 + \cdots + w_p^2 \leq 1.$$

Since $\mathbf{w}^T \mathbf{D}$ is an increasing function of w_j for all j, the optimum is attained at the boundary. Let $h(\mathbf{w}) = \mathbf{w}^T \mathbf{D} - \lambda(||\mathbf{w}|| - 1)$ where λ is a Lagrange multiplier. Set the gradient vector to be zero, then:

$$\begin{aligned}
\bigtriangledown h &= [BSS_1(\mathscr{C}) - 2\lambda w_1 \cdots BSS_p(\mathscr{C}) - 2\lambda w_p]^T = \mathbf{0}. \\
\Rightarrow w_j &= \frac{BSS_j(\mathscr{C})}{2\lambda} \ \forall j.
\end{aligned}$$

Since the L_2 constraint is 1:

$$\begin{aligned}||\mathbf{w}||_2^2 &= w_1^2 + \cdots + w_p^2\\ &= \left(\frac{BSS_1(\mathscr{C})}{2\lambda}\right)^2 + \cdots + \left(\frac{BSS_p(\mathscr{C})}{2\lambda}\right)^2\\ &= 1.\\ \Rightarrow \lambda &= \pm\frac{1}{2}\sqrt{\sum_{j=1}^{p} BSS_j(\mathscr{C})^2}.\end{aligned}$$

Since w_j and $BSS_j(\mathscr{C})$ are all nonnegative, it follows that:

$$w_j = \frac{BSS_j(\mathscr{C})}{\sqrt{\sum_{j=1}^{p} BSS_j(\mathscr{C})^2}}.$$

Therefore, the j^{th} feature weight is proportional to the dissimilarity along the j^{th} feature, $BSS_j(\mathscr{C})$. One thing to note is that even with a large L_1 tuning parameter value sparse K-means does not act the same as K-means. Although with $l_1 \geq \sqrt{p}$ sparse K-means returns nonzero weights for all the features, the weight on each feature is not constant as in K-means but proportional to the $BSS_j(\mathscr{C})$ of the feature.

Depending on the degree of the desired sparsity in feature weights, the user must choose $l_1 \in [1, \sqrt{p}]$. Further details on the lasso-type penalty is given in Tibshirani [24]. See Witten and Tibshirani [28] for the details for the automatic procedure for selecting the L_1 tuning parameter value, called the permutation approach.

2.3.5 A small simulation study of sparse *K*-means with datasets generated from the noisy model

We perform sparse 3-means on the datasets generated from the noisy model (see Section 2.2.4). The L_1 tuning parameter values are chosen by the permutation approach. The reported values in Table 2.4 are the averages (and standard deviations) over 100 simulations. In comparison with the partitions from 3-means in Table 2.3, the partitions from sparse 3-means are more desirable: CER reduces to 0 and 0.03 from the 0.02 and 1.00, returned by 3-means. On average, about 80% of

		% weights on	
μ	CER	Clustering features	l_1
1.0	0.00 (0.007)	84.24 (3.54)	8.01 (0.63)
0.8	0.03 (0.029)	79.38 (6.87)	8.15 (0.87)

Table 2.4: The simulation results of sparse 3-means on 100 datasets generated from the noisy model.

the weights are on the clustering features for both values of μ and the CERs stay low. In other words, sparse K-means returns a desirable selection of features which results in desirable partitions.

2.3.6 Simulation studies of sparse K-means with contaminated datasets

The simulation study in Table 2.4 showed that when datasets are not contaminated, sparse 3-means was able to choose the clustering features and produce better partitions than 3-means. Now, we examine the performance of sparse 3-means with contaminated data in various settings. In models 1, 2, and 3, a single case of the dataset generated from the noisy model is replaced by an outlier.

- **Model 1:** The value of a single noise feature for a single case, $x_{1,500}$, is replaced by the value, *out*.
- **Model 2:** The value of a single clustering feature for a single case, $x_{1,1}$, is replaced by the value, *out*.
- **Model 3:** A single case, $\mathbf{x}_1$, is replaced by one generated from a multivariate normal distribution with $\mu = (5,\cdots,5)^T \in \mathbb{R}^{500}$ and an identity variance covariance matrix.

We examine different magnitudes of *out* in models 1 and 2: 15, 25 and 500. We generated 100 contaminated datasets from each simulation setting: i.e., all the possible combination of Model and μ (and *out*, for models 1 and 2).

				% of Weights on		
	μ	*out*	CER	Clustering features	Contaminated feature	l_1
Model 1	1.0	15	0.00 (0.007)	77.73 (7.72)	0.28 (0.19)	9.25 (1.45)
		25	0.34 (0.125)	42.14 (28.77)	50.43 (25.53)	2.45 (1.11)
		500	0.50 (0.012)	0.08 (0.02)	99.56 (0.03)	1.20 (0.00)
	0.8	15	0.07 (0.126)	61.65 (18.1)	5.26 (18.97)	9.52 (2.75)
		25	0.40 (0.114)	26.38 (21.8)	51.49 (23.51)	2.35 (0.89)
		500	0.50 (0.012)	0.06 (0.01)	99.57 (0.02)	1.20 (0.00)
Model 2	1.0	15	0.03 (0.099)	87.38 (11.35)	9.42 (20.36)	7.35 (2.46)
		25	0.27 (0.042)	71.21 (4.65)	21.81 (2.32)	5.07 (0.77)
		500	0.41 (0.034)	99.64 (0.01)	99.34 (0.08)	1.20 (0.00)
	0.8	15	0.20 (0.175)	89.15 (16.36)	36.07 (36.9)	5.01 (3.75)
		25	0.35 (0.053)	63.08 (5.82)	30.34 (4.58)	3.63 (0.61)
		500	0.43 (0.032)	99.64 (0.02)	99.47 (0.05)	1.20 (0.00)
Model 3	1.0	–	0.25 (0.021)	20.33 (6.52)	–	19.88 (1.73)
	0.8	–	0.26 (0.020)	16.94 (0.54)	–	20.12 (0.00)

Table 2.5: The simulation results of sparse 3-means with the L_1 tuning parameter values chosen by the permutation approach.

Table 2.5 shows the average and standard deviations of CERs (the 4^{th} column), percentages of weights on clustering features (the 5^{th} column), percentages of weights on contaminated features (the 6^{th} column) and l_1 tuning parameter values over the 100 generated datasets for each simulation setting. We calculated CERs without the contaminated cases. Note that we do not report the percentage of weights on contaminated features for the simulation result with Model 3, because all the 500 features of Model 3 are contaminated.

In Model 1 and Model 2, a small increase in the outlier value from 15 to 25, causes CER to jump from about 0 to about 0.3 for μ=1. Sparse 3-means fails to produce the desired partition in the presence of a single outlier in a noise feature. This failure is probably due to the algorithm failing to select the appropriate features. The 5^{th} and the 6^{th} columns of Table 2.5 show the average percentage of weights on the clustering features and the contaminated noise features. It shows that for Model 1 the weights shift from the clustering features to the contaminated noise feature as *out* increases for both values of μ. For Model 2, the contaminated clustering feature takes up all the weight as *out* increases. CERs of Model 3 are over 0.25 for both μ.

In this simulation study, we applied the permutation approach (Witten and Tibshirani [28]) to choose the L_1 tuning parameter value. It becomes almost 1 as *out* increases from 15 to 500 for models 1 and 2. It is a reasonable explanation that the single outlier affects the permutation approach of the L_1 tuning parameter selection and the weights become too sparse. As a result, the weights shift to the single contaminated feature as *out* increases. In Model 3, the permutation approach is also affected by the outlier: L_1 tuning parameter value becomes nearly 20 and weights become nonsparse. From these simulation results, we find that the permutation approach of the L_1 tuning parameter value selection is very vulnerable to outliers.

2.3.7 Simulation studies of sparse 3-means with contamintaed datasets and fixed L_1 tuning parameter values

One might argue that the poor performance of sparse 3-means in models 1, 2 and 3 are due to the too small / too large L_1 tuning parameter values, which were selected by the permutation approach. The user of sparse K-means has his / her own preference on how many features to be used for partitioning: say, one might want to use 20% of the features for clustering. Then this 20% is the preference of sparsity for this user. In this sense, the sparsity is interpretable and can be predetermined by the user of the algorithm. The same models as above are then examined with fixed L_1 tuning parameters: 7.959 for $\mu = 1$ and 8.055 for $\mu = 0.8$. These values were obtained as follows: for each μ, we generated 30 datasets from the noisy model. For each generated dataset, we performed the permutation approach to select a L_1 tuning parameter value. Then we use the averages of the estimated L_1 tuning parameter values for each μ.

In both models 1 and 2 (Table 2.6), the weight of the single contaminated feature becomes dominant as *out* increases. The partition becomes undesirable as *out* increases from 15 to 500. These results are consistent with the results of models 1 and 2 with l_1 chosen by the permutation method. In Model 3, CERs from the fixed L_1 method are as high as those from the non-fixed L_1 method. This shows that the clustering is still affected by the contaminated case.

				% of Weights on	
	μ	*out*	CER	Clustering features	The contaminated feature
Model 1	1.0	15	0.00 (0.008)	84.01 (0.79)	0.33 (0.24)
		25	0.25 (0.023)	51.57 (1.63)	19.61 (0.71)
		500	0.50 (0.018)	0.08 (0.03)	99.56 (0.04)
	0.8	15	0.05 (0.072)	76.88 (8.89)	1.24 (2.59)
		25	0.29 (0.064)	38.54 (8.38)	24.92 (3.85)
		500	0.51 (0.011)	0.06 (0.01)	99.58 (0.02)
Model 2	1.0	15	0.00 (0.008)	83.56 (0.89)	3.04 (0.67)
		25	0.28 (0.022)	69.92 (1.52)	22.09 (1.27)
		500	0.41 (0.039)	99.64 (0.02)	99.33 (0.09)
	0.8	15	0.04 (0.063)	77.63 (5.20)	3.95 (2.06)
		25	0.34 (0.052)	61.32 (3.43)	28.21 (2.81)
		500	0.42 (0.035)	99.63 (0.02)	99.46 (0.05)
Model 3	1	–	0.25 (0.019)	65.96 (3.33)	–
	0.8	–	0.26 (0.019)	49.27 (4.16)	–

Table 2.6: The simulation results of sparse 3-means with fixed L_1 tuning parameter values.

From these simulation studies, we conclude that sparse K-means itself is severely affected by a single outlier. In fact, sparse K-means performed poorly with less than 2% (1 out of 60) of contamination in the dataset.

2.4 Trimmed K-means

Our robustification of sparse K-means (discussed in Chapter 3) uses the idea of ‘‘trimmed K-means’’ introduced by Gordaliza [7] and Gordaliza [8]. Trimmed K-means uses squared Euclidean distances as a similarity measure. Once cluster centers are defined, cases are assigned to their closest cluster centers in squared Euclidean distance. In this sense, clusters are determined by cluster centers. The idea of the trimmed K-means is that the outliers should be discarded or trimmed in the calculation of cluster centers. If O is the set of outliers to be trimmed, then the robust cluster center is:

$$\mathbf{tm}(O)_k = \frac{1}{|C_k \setminus O|} \sum_{i \in C_k \setminus O} \mathbf{x}_i \in \mathbb{R}^p.$$

where, $A \setminus B$ is a set of elements that are contained in a set A and not in a set B. Therefore $C_k \setminus O$ represents the set of cases that are in the k^{th} cluster and are not outliers. Note that if $O = \emptyset$, then $\mathbf{tm}(O)_k = \bar{\mathbf{x}}_k$. The robust within cluster sum of squares is:

$$WSS(\mathscr{C}, O) = \sum_{k=1}^{K} \sum_{i \in C_k \setminus O} ||\mathbf{x}_i - \mathbf{tm}(O)_k||^2.$$

Note that $WSS(\mathscr{C}, \emptyset) = WSS(\mathscr{C})$. Since we do not know which cases are outliers in practice (if we knew, we could simply discard the cases from the dataset), trimmed K-means trims the cases which have the largest $\lfloor \alpha N \rfloor$ squared Euclidean distances to their cluster centers, where $\lfloor a \rfloor$ is the largest integer smaller than a. Here, α is a predetermined trimming proportion of a dataset. Therefore, the trimmed K-means clustering is the solution, $\mathscr{C}$, to the problem:

$$\underset{\mathscr{C};\, |\mathscr{C}|=K}{\text{argmin}}\ WSS(\mathscr{C}, O).$$

where:

$$\begin{aligned} O =& \{\text{The cases with the } \lfloor \alpha N \rfloor \text{ largest squared Euclidean distances} \\ & \text{to their cluster centers}\}. \end{aligned} \tag{2.20}$$

The asymptotic and robustness properties of trimmed K-means are studied by Cuesta-Albertos et al. [4].

2.4.1 Trimmed K-means computing algorithm

The trimmed K-means algorithm essentially robustifies Lloyd's algorithm. As in the notation of the K-means algorithms, let $IC1(i)$ denote the cluster to which the i^{th} case belongs. Denote $\mathbf{tm}(O)_{IC1(i)}$ as the k^{th} cluster center to which the i^{th} case belongs.

Randomly choose K cases as initial cluster centers. Set $O = \emptyset$. Repeat 1, 2 and 3 until convergence.

- Step (a): Given cluster centers, update the class assignment by assigning each case to the closest cluster center:

For all $i = 1, \cdots, N$

$$IC1(i) = \underset{1 \le k \le K}{\operatorname{argmin}} ||\mathbf{x}_i - \mathbf{tm}(O)_k||^2.$$

- Step (b): Select the $\lfloor \alpha N \rfloor$ cases with the largest squared Euclidean distances to their cluster centers. Let O be specified as in (2.20).

- Step (c): Given the cluster assignments $IC1(i)$ ($i = 1, \cdots, N$), and the set of trimmed cases O, update the cluster centers, $\mathbf{tm}(O)_k$, $k = 1, \cdots, K$.

Since the cases with extreme values are trimmed when the updated cluster centers are evaluated, the extreme values do not affect the defined K clusters. The algorithm is implemented in the `R` package `trimcluster`.

In this algorithm, there are two ways in which some of the C_k's could become empty sets. First, in the standard K-means algorithm, it could happen that there is no case that has the closest distance to some of the K cluster centers at Step 1 of algorithm. Second, all the cases of some clusters could be trimmed in Step (b). If one of the C_k's becomes an empty set, `trimcluster` chooses the most outlying case relative to its cluster center to form a cluster of a single element.

Chapter 3

Robust sparse K-means clustering

3.1 The method

From simulation studies of sparse K-means, we found that outliers may affect two steps of the sparse K-means algorithm: the selection of a partition and the selection of feature weights. Combining ideas from sparse K-means and trimmed K-means, we propose our robust sparse K-means.

Robust sparse K-means trims cases in both squared Euclidean and weighted squared Euclidean distances. By replacing Step (a) of the sparse K-means algorithm, which performs standard K-means on a transformed dataset $\mathbf{Y}$, by trimmed K-means, cases are trimmed in weighted distances. We denote the set of trimmed cases in the weighted squared Euclidean distances as O_W. This step is intended to eliminate the effect of outliers from the selection of a partition.

Before executing Step (b), we introduce a second robustifing Step: Step (a-2). We trim cases in the nonweighted squared Euclidean distances, intended to eliminate outliers ‘‘surviving’’ Step (a) because of small weights. We notice that Step (a) of robust sparse K-means would not be enough to eliminate the effect of outliers. In fact, features including outlying observations tend to be assigned high weights by the classical sparse K-means algorithm. Step (a) would fail to downweight cases that have outlying observations on noise features if such features are assigned a negligible weight. Therefore, the second robustifying Step (a-2) is necessary to identify outliers in noise features and prevent them from getting a high weight in Step (b). We denote the cases trimmed in squared Euclidean distance in Step (a-2) by O_E. However cases in O_E are not excluded in the calculation of cluster centers in Step (a). In Step (b) of sparse K-means, the cases in both O_E and O_W are eliminated so that the effects of outliers are eliminated from the selection of weights. Robust sparse K-means considers only the squared Euclidean distance as a dissimilarity measure.

We define robust sparse K-means clustering as the solution set $(\mathscr{C}, \mathbf{w})$ to the problem:

$$\max_{\mathscr{C}, \mathbf{w};\ |\mathscr{C}|=K} \sum_{j=1}^{p} w_j BSS_j(\mathscr{C}, O_{EW}) \ \ s.t.\ ||\mathbf{w}||_1 \leq l_1, ||\mathbf{w}||^2 \leq 1, w_j \geq 0\ \forall j, \tag{3.1}$$

where O_{EW} is the set of potential outliers being trimmed: $O_{EW} = O_W \cup O_E$. The L_1 tuning parameter value l_1 and the number of clusters K must be predetermined. The trimming proportion value, α, is the proportion of cases trimmed in the squared Euclidean distances and in the weighted squared Euclidean distances. This trimmed proportion must also be predetermined. Note that $BSS_j(\mathscr{C}, O_{EW})$ is a between cluster sum of squares along the j^{th} feature, calculated without the cases in O_{EW}.

We propose an iterative algorithm for this optimization problem:

3.1.1 Robust Sparse *K*-means algorithm

Let $\mathbf{w} = [1/\sqrt{p}, \cdots, 1/\sqrt{p}]^T$, $O_E = \emptyset$ and $O_W = \emptyset$ and repeat steps (a), (a-2) and (b) below until the stopping rule is satisfied.

- Step (a): For a given $\mathbf{w}$, perform trimmed K-means on the transformed dataset $\mathbf{Y}$:

  $$\underset{\mathscr{C}}{\text{argmin}}\, WSS_{\mathbf{Y}}(\mathscr{C}, O_W).$$

  Step (a) returns the partition, $\mathscr{C}$, and the set of trimmed cases denoted as O_W.

- Step(a-2): For a fixed $\mathbf{w}$ and $\mathscr{C}$, trim cases in the squared Euclidean distance:

  $$O_E = \{\text{the cases with the } \lfloor \alpha N \rfloor \text{ largest squared Euclidean distances} \\ \text{to their cluster centers}\}.$$

  Given the partition $\mathscr{C}$ and the trimmed cases O_W from Step (a), the cluster centers, $\mathbf{tm}(O_W)_k$,based on the non-weighted data matrix, $\mathbf{X}$, are calculated without the trimmed cases O_W. The cases with the $\lfloor \alpha N \rfloor$ largest squared Euclidean distances to their cluster centers $\mathbf{tm}(O_W)_k$ are contained in the set O_E. Let $O_{EW} = O_W \cup O_E$.

- Step(b): For fixed $\mathscr{C}$, O_{EW}, maximize (3.1) with respect to $\mathbf{w}$. That is:

  $$\max_{\mathbf{w}} \mathbf{w}^T \mathbf{D}(O_{EW}) \;\; s.t. \;\; ||\mathbf{w}||^2 \leq 1, ||\mathbf{w}||_1 \leq l_1, w_j \geq 0 \;\; \forall j,$$

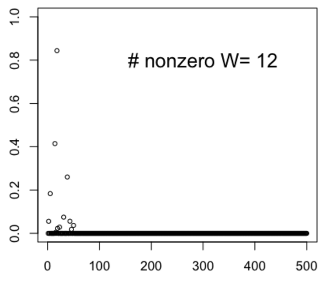
  where $\mathbf{D}(O_{EW}) = (BSS_1(\mathscr{C}, O_{EW}), \cdots, BSS_p(\mathscr{C}, O_{EW}))^T$. The analytic solution for this optimization problem, given by Witten and Tibshirani [28], is $\mathbf{w}(\triangle *)$ where:

  $$\mathbf{w}(\triangle) = \frac{(\mathbf{D}(O_{EW}) - \triangle \mathbf{1})_+}{||(\mathbf{D}(O_{EW}) - \triangle \mathbf{1})_+||}, \tag{3.2}$$

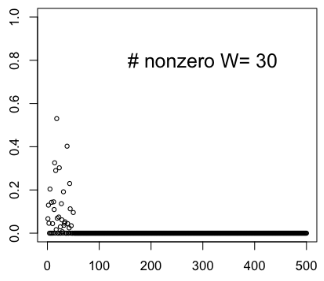
  and

  $$\triangle^* = \begin{cases} 0 & \text{if } ||\mathbf{w}(0)||_1 \leq l_1, \\ \text{root of } \; ||\mathbf{w}(\triangle)||_1 - l_1 = 0 & \text{otherwise} \end{cases} .$$

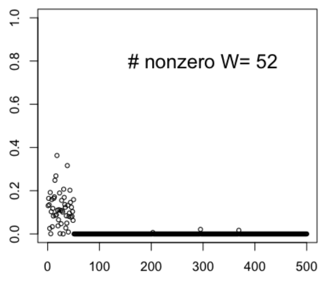
As a price for robustification, the algorithm no longer monotonely increases the objective function, $\sum_{j=1}^{p} w_j BSS_j(\mathscr{C}, O_{EW})$, from the q^{th} iteration of Step (b) to the $q+1^{st}$ iteration of Step (b). This is because the set of trimmed cases at the q^{th} iteration, O_{EW}^{q} is not necessarily the same as the set of trimmed cases at the $q+1^{st}$ iteration, O_{EW}^{q+1}. However, in the high-dimentional datasets that we have examined, the first iterations do increase the objective function evaluated at Step (b). For this reason, we use the following stopping rule: When the objective function at the $q+1^{st}$ Step (b) stops increasing, the algorithm terminates and the partition from the $q+1^{st}$ Step (a) and feature weights from the q^{th} Step (b) are returned as a solution.



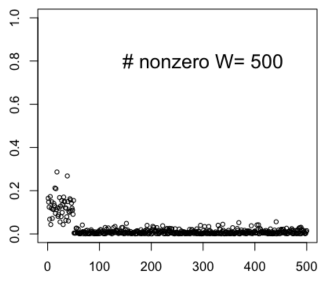
(a) l_1=2



(b) l_1=4



(c) l_1=6



(d) l_1=22

Figure 3.1: The distributions of weights from robust sparse K-means on a dataset simulated from Model 2. The x-axis shows the feature indices and y-axis shows the weights.

3.1.2 The choice of l_1

In robust sparse K-means, the L_1 tuning parameter is one of the arguments to be predetermined. Since sparsity is interpretable and the sparsity preference varies according to the purpose for clustering, we will assume that the sparsity parameter is chosen by the user of robust sparse K-means. The user should choose a L_1 tuning parameter value in $[1, \sqrt{p}]$.

In order to examine how the spread of weights from robust sparse K-means changes with respect to L_1 tuning parameter values, we generate a dataset from Model 2 with μ=0.8 and *out*=25 and robust sparse 3-means is performed with different L_1 tuning parameter values. Figure 3.1 shows the weights from robust sparse 3-means with four different L_1 tuning parameter values between 1 and 22 ($\approx \sqrt{500}$). Since the first 50 features in this simulation dataset are clustering features and the rest are noise features, all the L_1 tuning parameter values return larger weights on the first 50 features than on the remaining 450 features. As l_1 increases, the number of nonzero weights increases and when $l_1 \approx \sqrt{500}$ all the features receive nonzero weights.

Section 2.3.4 showed that the feature weights become proportional to the corresponding dissimilarity measures across clusters when $l_1 \geq \sqrt{p}$ and only the feature with the largest dissimilarity across clusters receives a nonzero weight when $l_1 \leq 1$ in sparse K-means. These relationships hold for robust sparse K-means except that the dissimilarities across clusters are now calculated without the set of potential outliers. When $0 < l_1 < \sqrt{p}$, the features become sparse: features with 'large' dissimilarities across clusters receive weights while the rest of the features receive zero weights in sparse K-means and robust sparse K-means. We can see this relationship between the feature weights and the corresponding $BSS_j(\mathscr{C}, O_{EW})$ for the generated dataset in Figure 3.2. The same simulation results as Figure 3.1 are used. The users of robust sparse K-means (and also sparse K-means) should know that the weights are not assigned either zero or one. Even among the nonzero weighted features, the contribution for separating clusters, i.e., the magnitude of $BSS_j(\mathscr{C}, O_{EW})$ varies: the interpretation is that the features with larger weights are contributing more to the separation of clusters.

Robust sparse K-means is implemented in the `R` package `RSKC`. The package will be made publicly available at `http://cran.r-project.org/`.

3.2 Simulation studies of robust sparse 3-means with contaminated datasets

The performance of robust sparse 3-means is examined on datasets generated from models 1, 2 and 3. The L_1 tuning parameter values are fixed to 7.959 for $\mu = 1.0$ and 8.055 for $\mu = 0.8$. We fixed the percentage of cases to be trimmed, α, to

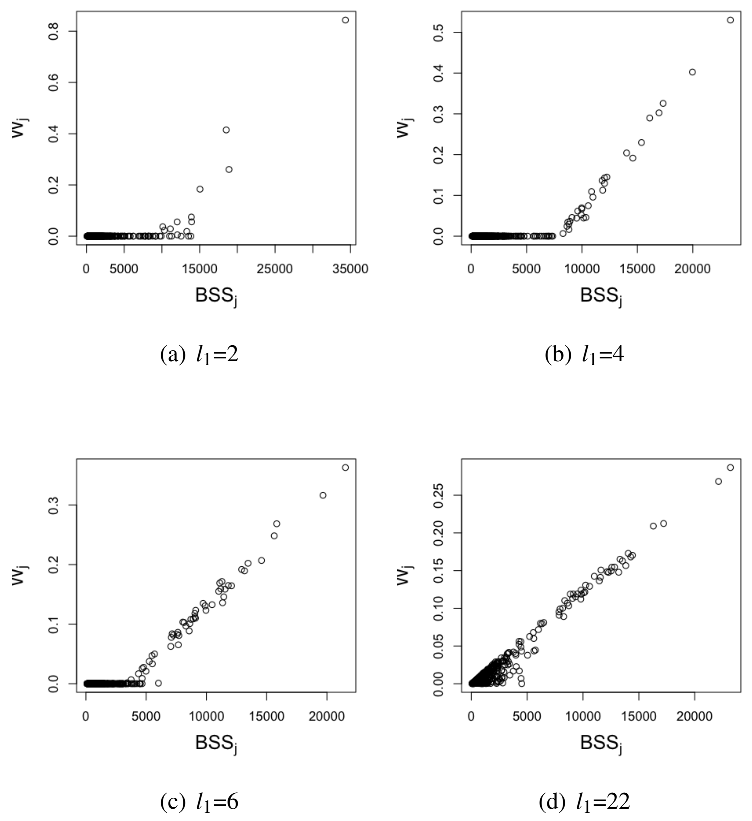


Figure 3.2: The weights from robust sparse *K*-means on a dataset generated from Model 2 v.s. the between clster sum of squares of the j^{th} feature.

$1/60$ to choose one case as an outlier in squared Euclidean and weighted squared Euclidean distances. We generate 100 datasets for each combination of Model and μ (and *out* for models 1 and 2), and perform robust sparse *K*-means for each dataset.

Table 3.1 shows the results. We calculated CERs without the contaminated cases as in the simulation summaries of sparse *K*-means with the contaminated models.

For Model 1, when $\mu = 1.0$, the CERs remain almost 0 for all the generated

	μ	*out*	CER	% of Weights on Clustering features	Contaminated features
Model 1	1.0	15	0.00 (0.008)	83.80 (0.81)	0.035 (0.07)
		25	0.00 (0.007)	83.76 (0.77)	0.029 (0.06)
		500	0.00 (0.009)	83.90 (0.96)	0.038 (0.07)
	0.8	15	0.06 (0.084)	78.18 (4.81)	0.249 (1.49)
		25	0.03 (0.060)	79.23 (1.64)	0.037 (0.09)
		500	0.06 (0.108)	76.64 (13.6)	3.010 (17)
Model 2	1.0	15	0.00 (0.008)	83.92 (0.87)	1.680 (0.51)
		25	0.00 (0.008)	83.91 (0.81)	1.640 (0.52)
		500	0.00 (0.008)	83.80 (0.81)	1.658 (0.56)
	0.8	15	0.04 (0.067)	79.08 (2.72)	1.701 (1.13)
		25	0.04 (0.077)	78.98 (2.38)	1.881 (2.38)
		500	0.05 (0.079)	78.90 (1.91)	1.559 (0.67)
Model 3	1.0	–	0.00 (0.008)	83.77 (0.86)	–
	0.8	–	0.05 (0.070)	78.87 (3.94)	–

Table 3.1: The results of models 1 to 3 with robust sparse 3-means and fixed l_1 constraint.

μ	*out*	The number of times the outlier is captured in O_W	in O_E
1.0	15	8	99
	25	13	100
	500	44	100
0.8	15	12	99
	25	15	100
	50	30	100

Table 3.2: The number of times the single outlier is captured in O_W and O_E over 100 datasets generated from Model 1.

datasets and over 83% of the weights are on the clustering features for all the values of outliers. For this reason, we can say that the value of the outlier does not affect the performance of robust sparse 3-means when the clusters are reasonably separated. Even when the true 3 clusters become closer to each other (μ=0.8), the algorithm still assigns over 76% of the weights on clustering features and the CER stays as low as 0.06.

Table 3.2 shows the number of times that robust sparse 3-means trims the single contaminated case in the squared Euclidean distances or the weighted squared

Euclidean distances. In Model 1, the outlier is in one of the noise features, so it is trimmed only in the squared Euclidean distance most of the times. As long as the contaminated case is captured in either O_E or O_W, the contaminated case does not affect the selection of features: the potential outliers do not affect Step (b) of robust sparse K-means algorithm.

In Model 2, one of the clustering features contains an outlier value. Again, when $\mu = 1.0$, the CERs stay almost zero for all the generated datasets and for all the values of out. When μ=0.8, the CERs stay as low as 0.05 for all out. In fact, both O_W and O_E capture the contaminated case for all the generated datasets (the results are not shown). The contaminated feature receives 1.7 % of weights unlike Model 1 where it receives almost no weight. This is because the contaminated feature in Model 2 is a clustering feature. Since 50 out of 500 features are clustering features, it is ideal to assign the nonzero weights to all the 50 clustering features. Then each clustering feature should receive $100\%/50 = 2\%$ weight on average. Therefore, the 1.7% weight on this contaminated noise feature is reasonable.

Model 3 has a single case whose features are all contaminated: both clustering features and noise features. The CERs are as low as 0.00 ($\mu = 1.0$) and 0.06 ($\mu = 0.8$), and about 80% of the weight is on the 50 clustering features. Again, O_W and O_E successfully capture the outlier case in all the generated datasets (the results are not shown).

3.3 Simulation studies to compare the performance of K-means, sparse K-means, trimmed K-means and robust sparse K-means

The simulation results of models 1 to 3 show that robust sparse 3-means clustering improves the selection of a partition and the selection of features for these types of models. In addition to models 1 to 3, the performance of the robust sparse 3-means clustering is examined with contaminated datasets with more than one outlier. Datasets generated from the noisy model are contaminated as follows:

- **Model 4:** Two cases from each cluster are contaminated in a single noise feature by adding noise with a large variance. In total 6 cases (10% of cases are contaminated) and 6 noise features are contaminated. Specifically, $x_{ij} \sim$

$N(0,15^2)$ for (i,j)=(1,51), (2,52), (21,53), (22,54), (41,55), (42,56).

- **Model 5:** Two cases from each cluster are contaminated in a single clustering feature each by adding noise with a large variance. In total 6 cases (10% of cases are contaminated) and 6 clustering features are contaminated. Specifically, $x_{ij} \sim N(0,15^2)$ for (i,j)=(3,1), (4,2), (23,3), (24,4), (43,5), (44,6).

- **Model 6:** Outlier values are generated from both models 4 and 5. In total 12 cases (20% of cases are contaminated), 6 clustering features and 6 noise features are contaminated.

- **Model 7:** Two cases from each cluster are replaced by ones generated from a multivariate normal distribution with large variance. Specifically, $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_{21}$, $\mathbf{x}_{22}$, $\mathbf{x}_{41}$ and $\mathbf{x}_{42}$ are generated from $\mathbf{N}_{500}(\mathbf{0}, 5^2\mathbf{I})$. In total 6 cases (10% of cases are contaminated) and all the features are contaminated.

- **Model 8:** Twenty-five clustering features of the 1^{st} case are replaced with 25 clustering features of the 60^{th} case.

In Model 8, outlier values are 'hidden', that is, the outlier values are not extreme from the univariate point of view. The projection of all the cases onto a single feature would not show the contaminated cases as outliers. It is, however, an outlier if one takes into account the direction of the extreme value created by more than one feature.

The datasets are generated 100 times from all the possible combinations of Model and μ. Simulation results of the robust sparse 3-means on the datasets are in Table 3.3. The reported values are the averages and standard deviations of CERs (the 3^{rd} column), percentages of weights on clustering features (the 4^{th} column) and percentages of weights over contaminated features (the 5^{th} column) over the 100 generated datasets for each simulation setting. Note that we do not report the average percentage of weights on contaminated features of models 7 and 8 because all the features are contaminated in Model 7 and half of the clustering features (i.e.

			% weights on	
	μ	CER	Clustering features	Contaminated features
Model 4	1.0	0.01 (0.012)	83.07 (1.02)	0.257 (0.21)
	0.8	0.05 (0.066)	77.48 (3.42)	0.341 (0.32)
Model 5	1.0	0.01 (0.013)	83.26 (0.97)	9.92 (1.35)
	0.8	0.08 (0.094)	77.66 (2.34)	9.053 (1.57)
Model 6	1.0	0.01 (0.024)	82.01 (1.18)	10.148 (1.34)
	0.8	0.12 (0.120)	74.26 (9.70)	9.311 (2.38)
Model 7	1.0	0.01 (0.015)	83.35 (0.88)	–
	0.8	0.11 (0.105)	72.51 (16.86)	–
Model 8	1.0	0.01 (0.001)	83.25 (0.97)	–
	0.8	0.07 (0.086)	77.87 (2.13)	–

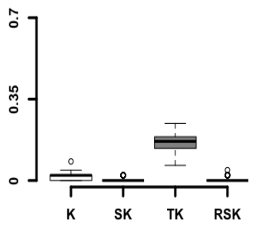
Table 3.3: The results from robust sparse 3-means with the fixed L_1 value on datasets generated from models 4 to 8.

25 features) are contaminated in Model 8. The proportion of weights on clustering features are all greater than 70% and CERs are almost zero for all the models with $\mu = 1.0$.

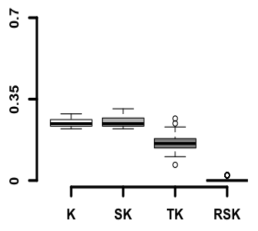
Now we compare the performances of the 3-means, sparse 3-means, trimed 3-means and robust sparse 3-means for all models. Again, L_1 tuning parameter values are fixed to 7.959 for $\mu = 1.0$ and 8.055 for $\mu = 0.8$ for the sparse methods. The trimming proportion α is set to the true proportion of contaminated cases for the robust methods. The results are shown in figures 3.3 and 3.4 for $\mu = 1.0$ and figures 3.5 and 3.6 for $\mu = 0.8$.

Figures 3.3 and 3.5 show the boxplots of CERs for each method and for each simulation model with $\mu = 1.0$ and 0.8 respectively. It is clear that robust sparse 3-means returns the best CERs among all the clustering methods for all models.

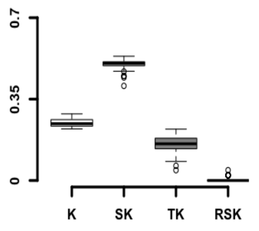
In our simulation models, all the clustering features contribute equally for separating the three clusters, because all the clustering features are generated from the same mixture model (see the description of the clean model in Section 2.2.4). Therefore, it would be ideal if all the clustering features receive equal nonzero weights while noise features receive no weight. Then each clustering feature should receive about 2% (=100 $\times$1/50) of the weight. We measure how evenly the weights are spread over the clustering features by the median proportion of the weights on



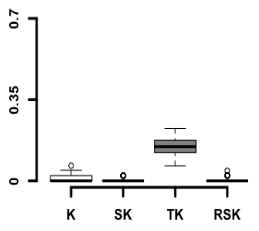
(a) Model 1 *out*=15



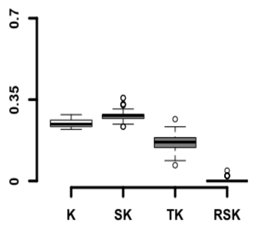
(b) Model 1 *out*=25



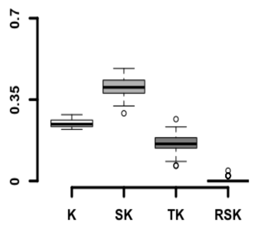
(c) Model 1 *out*=500



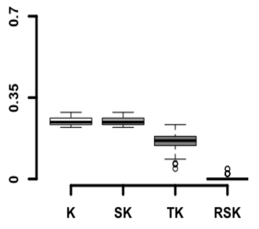
(d) Model 2 *out*=15



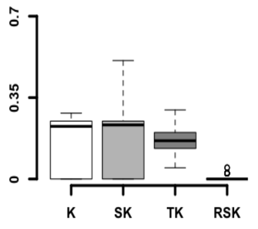
(e) Model 2 *out*=25



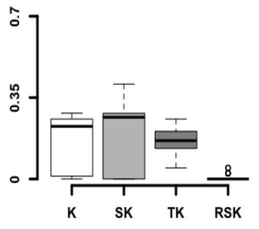
(f) Model 2 *out*=500



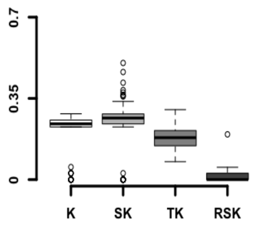
(g) Model 3



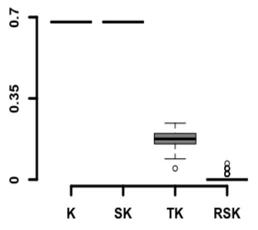
(h) Model 4



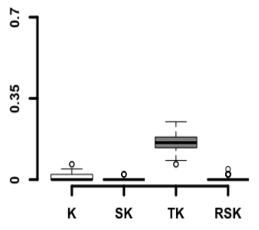
(i) Model 5



(j) Model 6

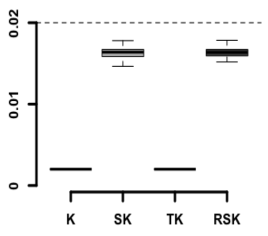


(k) Model 7

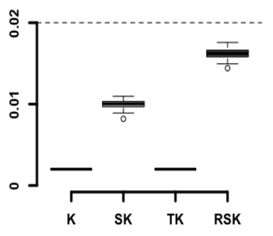


(l) Model 8

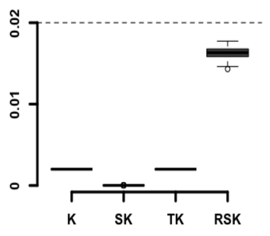
Figure 3.3: The simulation results with $\mu = 1.0$. The y-axis shows the CERs. K = K-means, SK = Sparse K-means, TK = Trimmed K-means, and RSK = Robust Sparse K-means.



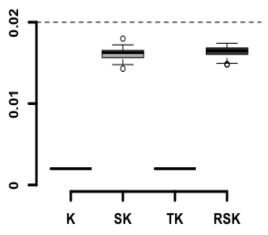
(a) Model 1 *out*=15



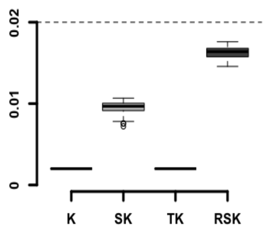
(b) Model 1 *out*=25



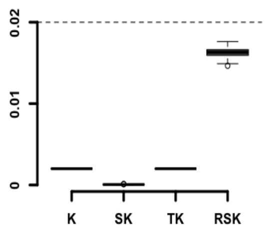
(c) Model 1 *out*=500



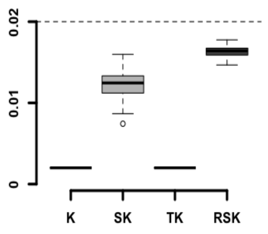
(d) Model 2 *out*=15



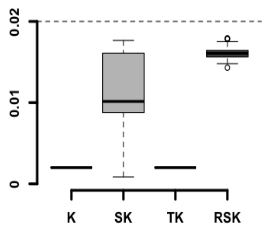
(e) Model 2 *out*=25



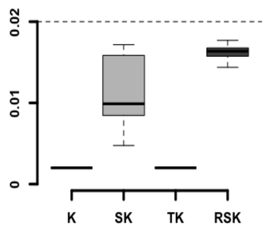
(f) Model 2 *out*=500



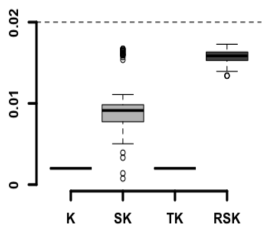
(g) Model 3



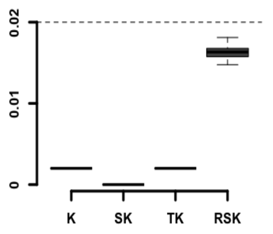
(h) Model 4



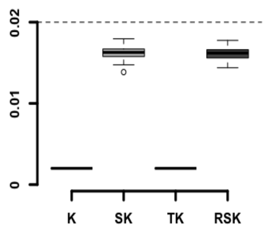
(i) Model 5



(j) Model 6

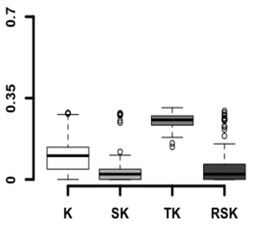


(k) Model 7

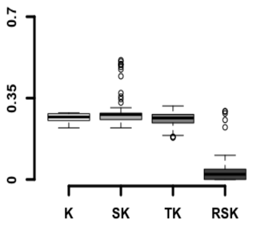


(l) Model 8

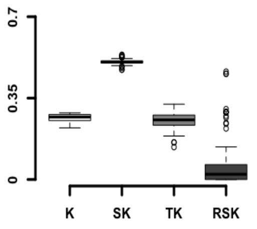
Figure 3.4: The simulation results with $\mu = 1.0$. The y-axis shows the median proportions of weights on the clustering features.



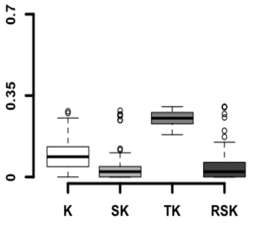
(a) Model 1 *out*=15



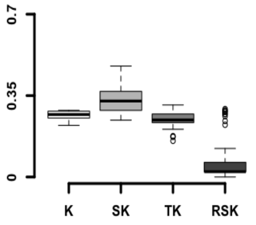
(b) Model 1 *out*=25



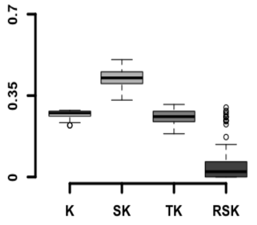
(c) Model 1 *out*=500



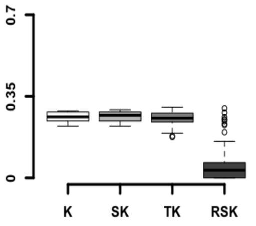
(d) Model 2 *out*=15



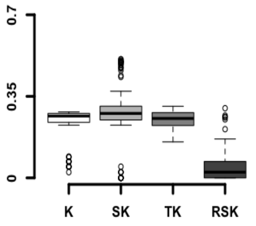
(e) Model 2 *out*=25



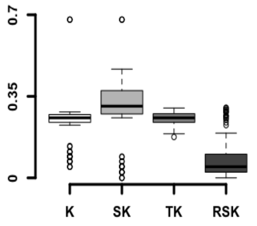
(f) Model 2 *out*=500



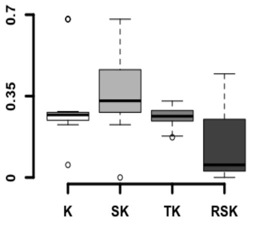
(g) Model 3



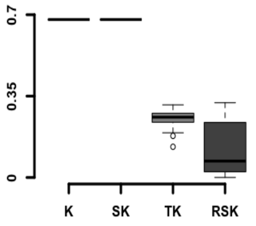
(h) Model 4



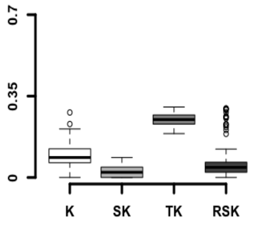
(i) Model 5



(j) Model 6

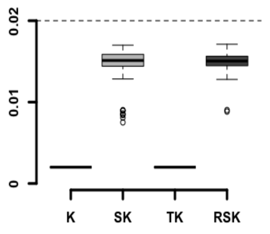


(k) Model 7

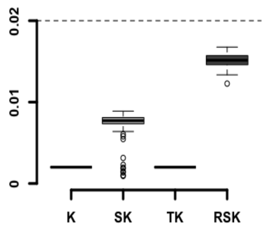


(l) Model 8

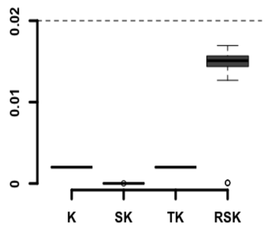
Figure 3.5: The simulation results with $\mu = 0.8$. The *y*-axis shows CERs.



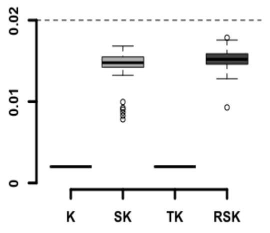
(a) Model 1 *out*=15



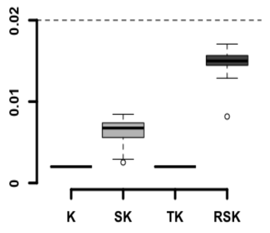
(b) Model 1 *out*=25



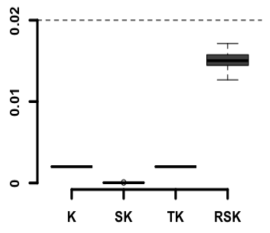
(c) Model 1 *out*=500



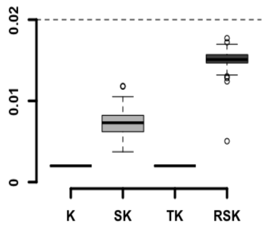
(d) Model 2 *out*=15



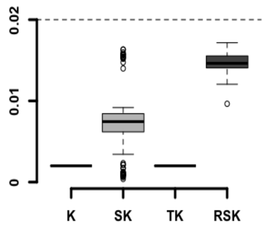
(e) Model 2 *out*=25



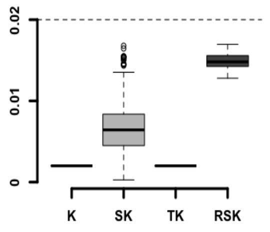
(f) Model 2 *out*=500



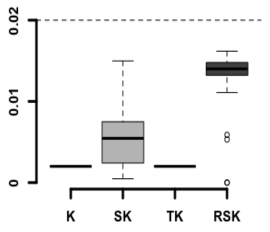
(g) Model 3



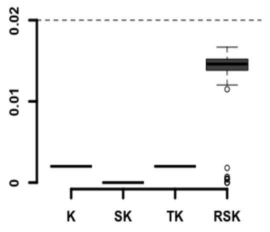
(h) Model 4



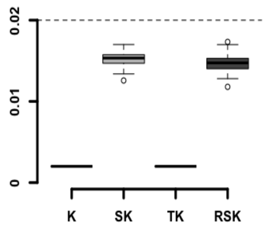
(i) Model 5



(j) Model 6



(k) Model 7



(l) Model 8

Figure 3.6: The simulation results with μ=0.8. The *y*-axis shows the median proportions of weights on the clustering features.

the clustering features. Figures 3.4 and 3.6 show the median proportion of the weight on the 50 clustering features for each method and each simulation Model for μ=1.0 and 0.8 respectively. The dashed line shows 0.02; methods with desirable selection of weights should have their boxplots around the 0.02 line. Since trimmed 3-means and 3-means do not have the feature selection property, the median proportion of the weight on the clustering features are 0.002 (=1/500) for all models. From both figures 3.4 and 3.6, we can see that the boxplots from robust sparse 3-means stay the closest to the 0.02 line among the four clusterings for all models.

We find that the relative performance of the clustering methods are almost the same between the simulation models with μ=1.0 and with $\mu = 0.8$ except that the boxplots of CERs and median proportion of the weight from $\mu = 0.8$ have larger variation than those from $\mu = 1.0$. That is, clustering methods that perform relatively well in a model with $\mu = 1.0$ in terms of CER / median proportion of the weight on clustering features also perform relatively well in the same model with $\mu = 0.8$. Now we list the overall summaries for both μ=1.0 and 0.8 for each Model.

- Model 1 and Model 2
  - The boxplots of CERs from the four algorithms are almost the same when the outlier value *out* is 15 except for trimmed 3-means. Its CERs are worse than other algorithms. Since trimmed 3-means robustifies 3-means, one might think that it should perform at least as well as 3-means. We think that this relatively poor performance of trimmed 3-means is due to its algorithm: trimmed K-means in `R` essentially robustifies Lloyd's algorithm while K-means of `R` uses Hartigan and Wong's algorithm of Hartigan and Wong [11]. The CERs from trimmed 3-means are worse than those from 3-means probably because Hartigan and Wong's algorithm achieves better local optimum than Lloyd's algorithm. The boxplots of the median proportion of the weight from the sparse methods stay high.
  - As a value of *out* increases, the boxplots of CERs of sparse 3-means shift to the highest among all. Interestingly, sparse 3-means performs

even worse than 3-means. This is because the single contaminated noise feature receives almost all the weight (see Table 2.6). On the other hand, the boxplots of CERs and the median proportion of the weight from the robust methods do not change according to the value of *out*.

- Model 3
  - The boxplots show that CERs from robust sparse 3-means are zero for almost all the generated datasets. The CERs from trimmed 3-means tend to be lower than ones from sparse 3-means although sparse 3-means assigns more weights to clustering features than trimmed 3-means.
- Model 4 and Model 5
  - The boxplots of CERs from all methods show large spread except that from robust sparse 3-means. The CERs from robust sparse 3-means are zero for almost all the generated datasets. The large spreads in the boxplots of CERs (and boxplots of median proportion of the weight from sparse 3-means) are probably because the outlier values of models 4 and 5 are generated from a distribution with large variation, $N(0, 15^2)$.
- Model 6
  - The boxplots of the median proportion of the weight from robust sparse 3-means stay the highest among all. The boxplots of CERs from robust sparse 3-means stay the lowest among all.
- Model 7
  - CERs from nonrobust methods are 0.7 for all the generated datasets. This is because the nonrobust methods produce a cluster with a single contaminated case.
  - Sparse 3-means assigns less weight to clustering features than the non-sparse methods.

  - CERs from robust sparse 3-means are almost 0 for almost all the generated datasets.

- Model 8

  - The results for Model 8 are very similar to the ones for Model 1 with $out = 15$.

3.4 Robust sparse K-means with missing values

In large datasets, we often encounter the problem of missing values. Microarray gene expression datasets (which will be analyzed in Chapter 5), in particular, often contain missing values that occur for diverse reasons, including insufficient resolution of gene expressions, image corruption or simply due to dust or scratches on the slide (Troyanskaya et al. [26]). Since cases may come from different clusters, one might not want to simply replace missing values with the feature means. One might argue that we can simply drop all cases with missing values. In high dimensional datasets, the sample size, N, is usually far smaller than the dimension p. Therefore, it is not unusual that all cases have at least one missing feature. Then we cannot simply drop cases with missing values. Moreover, dropping cases potentially makes the sample biased: for example, it could happen that all cases from a cluster are dropped.

In this section, we introduce an adjustment to robust sparse K-means for missing values. Instead of using the squared Euclidean distance as a dissimilarity measurement of two cases, the algorithm uses an adjusted squared Euclidean distance. The idea of the missing value adjustment is as follows. If some cases in the k^{th} cluster contain missing values along the j^{th} feature, then it is natural to calculate the k^{th} cluster center along the j^{th} feature without such cases. Denote MC_j the set of case indices such that the corresponding cases do not have missing values in the j^{th} feature. The missing value-adjusted k^{th} cluster center along the j^{th} feature is:

$$\bar{x}_{k,j} = \frac{1}{|C_k \cap MC_j|} \sum_{i' \in C_k \cap MC_j} x_{i'j}.$$

Note that $C_k \cap MC_j$ contain the case indices for the cases that belong to the k^{th}

cluster for which the j^{th} feature is not missing. If there was no weight, the distance between a case with missing values, $\mathbf{x}_i \in \mathbb{R}^p$ and the k^{th} cluster center, $\bar{\mathbf{x}}_k$, should be scaled by the ratio of the number of features p to the number of non-missing features in the vector $\mathbf{x}_i - \bar{\mathbf{x}}_k$ contains. Let $MF(\mathbf{x})$ denote the set of feature indices for non-missing features in the vector $\mathbf{x}$. The adjusted squared Euclidean distance, d_{AE}, between $\mathbf{x}_i$ and $\bar{\mathbf{x}}_k$ is:

$$d_{AE}(\mathbf{x}_i, \bar{\mathbf{x}}_k) = \frac{p}{|MF(\mathbf{x}_i - \bar{\mathbf{x}}_k)|} \sum_{j \in MF(\mathbf{x}_i - \bar{\mathbf{x}}_k)} (x_{ij} - \bar{x}_{k,j})^2. \tag{3.3}$$

Applying this idea, the weighted squared Euclidean distance between a case vector and the k^{th} cluster center should be adjusted in terms of the weights. The adjusted weighted Euclidean distance d_{AW} is defined as:

$$d_{AW}(\mathbf{x}_i, \bar{\mathbf{x}}_k) = S_{i,k} \sum_{j \in MF(\mathbf{x}_i - \bar{\mathbf{x}}_k)} w_j (x_{ij} - \bar{x}_{k,j})^2, \tag{3.4}$$

where

$$S_{i,k} = \frac{\sum_{j=1}^{p} w_j}{\sum_{j \in MF(\mathbf{x}_i - \bar{\mathbf{x}}_k)} w_j}. \tag{3.5}$$

The robust sparse K-means is adjusted for missing values by replacing all the squared Euclidean distance measures by (3.3) and all the weighted squared Euclidean distance measures by (3.5). Note that this missing value adjusted algorithm does not work when a L_1 tuning parameter value is small. If a L_1 tuning parameter value is small, only a few features receive nonzero weights. Then it could happen that all the nonzero weighted features of some cases are missing. In this case, the algorithm cannot adjust the weighted distance for such cases. This adjustment, however, works fine with moderately large L_1 tuning parameter values.

Chapter 4

Selection of K

4.1 Brief literature review

Nonhierarchical procedures usually require users to specify the number of clusters before any clustering can be accomplished. There are data-dependent automatic procedures to select K where the user does not have to subjectively specify it. In this section, we discuss two procedures that can be adopted with robust sparse K-means: the gap statistic (Tibshirani et al. [25]) and Clest (Duboit and Fridlyand [5]). Since the simulation study of Duboit and Fridlyand [5] showed that Clest performs better than the gap statistic, we adapt Clest to robust sparse K-means. We will still introduce the gap statistic because some techniques of the gap statistic are applied in Clest (Section 4.3.2). We later show the results of a simulation study assessing the performance of Clest with robust sparse 3-means (Section 4.4).

4.2 The gap statistic

Tibshirani et al. [25] proposed the gap statistic method. The gap statistic method formalized a heuristic approach that experimenters had been using. In order to select the appropriate number of clusters, experimenters often considered the within cluster sum of squares, $WSS(\mathscr{C})$, as an error measure of clustering. This $WSS(\mathscr{C})$ is a decreasing function of K: clearly, for $K = 1$, $WSS(\mathscr{C}) = TSS$ while for $K = N$, $WSS(\mathscr{C}) = 0$. If the true number of clusters is K^*, then the rate of decrease of

$WSS(\mathscr{C})$ should decrease dramatically for $K > K^*$, because we are essentially adding unnecessary cluster centers in the middle of a cluster. The location of such an elbow decline in the plot of $WSS(\mathscr{C})$ against K indicates the appropriate number of clusters and experimenters have chosen such K as estimate of the number of clusters.

A problem arises in this heuristic procedure: it assumes that there is a cluster structure ($K > 1$). The idea of gap statistic is to standardize $WSS(\mathscr{C})$ by comparing it to its expectation under an appropriate reference distribution ($K = 1$) of the dataset (See Section 4.2.1 for more detail about the reference distribution). The gap statistic wishes to screen all the evidence of $K > 1$ against $K = 1$ and selects the K which has the strongest evidence against the null hypothesis $K = 1$ after taking the sampling distribution into account. In order to estimate the expectation of $WSS(\mathscr{C})$ under an appropriate reference distribution ($K = 1$) of the dataset, Tibshirani et al. [25] proposed that we should generate B^0 Monte Carlo samples from the reference distribution. Performing clustering with all the possible numbers of clusters, K, on each Monte Carlo sampling leads to B^0 within cluster sum of squares of reference datasets for each K. Tibshirani et al. [25] use their mean as an estimate of the expectation of $WSS(\mathscr{C})$ under the reference distribution and their standard deviation to take the sampling distribution into account.

4.2.1 Two methods for generating the reference distributions of the gap statistic

If all observed cases were from a cluster, then they are from a reference distribution. Then an appropriate reference distribution should form a cluster and reflect the sampling distribution of the observed cases. Tibshirani et al. [25] proposed two methods to find such a reference distribution (see Section 4.2.1). We will explain the two methods in detail because they are also used in Clest.

1. Simple reference distribution: generate each reference feature uniformly and independently over the range of the observed values for that feature.

2. PCA reference distribution: generate the reference features from a uniform distribution over a box aligned with the principal components of the data.

Given the normalized data matrix $\mathbf{X} \in \mathbb{R}^{N \times p}$, the variance covariance matrix is:

$$\mathbf{S_X} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}.$$

Then $\mathbf{S_X}$ is orthogonally diagonalized as:

$$\mathbf{S_X} = \mathbf{PDP}^T.$$

where:

- $\mathbf{D}$ is a diagonal matrix with the eigenvalues, $\lambda_1, .., \lambda_p$ of $\mathbf{S_X}$ on the diagonal entries, arranged such that $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$.
- $\mathbf{P}$ is an orthogonal matrix whose columns are the corresponding unit eigenvectors $\mathbf{u_1}, .., \mathbf{u_p} \in \mathbb{R}^P$ of $\mathbf{S_X}$.

We transform $\mathbf{X}$ via $\mathbf{X}^* = \mathbf{XP}$. Then the transformed data is no longer correlated. To see this:

$$\begin{aligned}\mathbf{S_{X^*}} &= \frac{1}{N-1}\mathbf{X}^{*\prime}\mathbf{X}^* \\ &= \mathbf{P}'\mathbf{S_X}\mathbf{P} \\ &= \mathbf{P}'\mathbf{PDP}'\mathbf{P} \\ &= \mathbf{D}.\end{aligned}$$

Then we draw uniform features $\mathbf{Z}^*$ over the range of the columns of $\mathbf{X}^*$ as in method 1 with $\mathbf{X}$. Finally we back-transform via $\mathbf{Z} = \mathbf{Z}^*\mathbf{P}'$ to give reference data set $\mathbf{Z}$. By applying PCA, the reference distribution takes into account the shape of the data distribution, or the angle created by sampled features.

Before the gap statistic was proposed, Milligan and Cooper [18] conducted an extensive Monte Carlo evaluation of 30 criteria for selecting the number of clusters. Tibshirani et al. [25] carried out a comparative Monte Carlo study of the gap statistic and some of the 30 criteria which were examined by Milligan and Cooper [18]. Their simulation studies showed that the gap statistic performs better than the 30 criteria.

4.3 Clest

4.3.1 The idea

Duboit and Fridlyand [5] proposed a prediction-based resampling method for estimating the number of clusters in a dataset. Clest uses an idea of supervised learning in the context of clustering, which is an unsupervised learning method. In unsupervised learning, partitions are unknown and need to be discovered from a dataset. In supervised learning, partitions are predefined and the task is to build a classification rule. Then the user of a supervised learning method applies the classification rule for predicting the cluster labels on future dataset. In unsupervised learning, once clustering procedures identify the new clusters and cluster labels are assigned to the cases, the next step is often to build a classifier for predicting the cluster labels of future cases. Duboit and Fridlyand [5] argue that *the classifier built from the clustering algorithm with the true number of clusters should have the most stable predictability*. If we had a future dataset, the stability in the prediction via the classifier can be assessed by comparing the predicted partition and the partition from the clustering algorithm on the future dataset. Clest uses a resampling method to generate ''future'' datasets from the original dataset.

Their simulation study compares the performance of Clest, the gap statistic method and other criteria for selecting the number of clusters. Duboit and Fridlyand [5] concluded that for their simulation model, Clest was the most robust and accurate. For this reason and because Clest is applicable to any clustering algorithm, we apply Clest to robust sparse K-means.

4.3.2 The idea of the clest algorithm

Given a clustering algorithm, Clest returns the estimated number of clusters $\hat{K}$ that produces the most ''stable'' predictability in the clustering procedure. Since it is impossible to obtain future datasets, in order to assess predictability, Dudoit and Fridlyand used a resampling technique. The idea of Clest algorithm with arbitrary clustering algorithm is explained as follows.

First, randomly partition the dataset $\mathbf{X}$ into a learning set $\mathbf{L} \in \mathbb{R}^{N_L \times p}$, containing N_L cases and a testing set $\mathbf{T} \in \mathbb{R}^{N_T \times p}$, containing N_T cases. Given the learning set

L and a number of clusters K', a clustering algorithm partitions the cases into K' clusters. It returns a classifier as well as a partition of cases in **L**. In K-means, for example, the classifier can be cluster centers: one can classify future cases by assigning them to their closest cluster centers.

We wish to assess the prediction power of the classifiers. Given the testing set, **T**, the classifier is used to partition the cases of **T** into K' clusters. The clustering algorithm can also partition the cases of **T** into K' clusters. Then we have two partitions of the training set **T**: one from the classifier and another from the clustering algorithm. If K' is the true number of clusters, the prediction accuracy of the ''classifiers'' should be good, i.e., the two partitions of the **T** should agree well. Given a measure of the agreement of these two partitions, we can assess how well they agree. This measure can be, for example, CER.

If we repeat the above two procedures for all the potential number of clusters, say $k = 2, \cdots, M$, one can obtain M-1 observed measures of agreement of the resulting pairs of partitions. Then one might think that one can choose the number of clusters corresponding to the best observed measure of agreement of partition. However, this strategy ignores the fact that there could be no cluster structure. When a clustering algorithm is applied to a dataset, a partition of the dataset is returned whether or not the dataset has the true cluster structure. Therefore, we should standardize the observed measure of agreement by comparing it to its expected value under a null hypothesis: K=1. In order to estimate the expected value of the observed measure of agreement under K=1, Clest generates B^0 Monte Carlo reference datasets from the reference distribution as in the gap statistic. Then repeat the procedure above to obtain the B^0 reference measures of agreement of the two partitions under K=1. The median of the B^0 reference measures is used as the estimate. Then the percentage of the reference measures of agreement that are better than the observed measures of agreement can serve a role of p-values: the estimate of the probability of observing observed measure or more extreme measure under the null hypothesis $K = 1$. Now, we have two measurements of agreement of partitions: one from the observed dataset and another from the reference dataset. The standardized measurement of agreement is calculated by comparing the observed measurement of agreement to the reference measurement of agreement for each $k = 2, \cdots, M$. Finally Clest chooses the number of clusters that returns the strongest ‘‘significant’’ evidence against the hypothesis $H_0 : K = 1$.

4.3.3 The algorithm of Clest with robust sparse K-means

We introduce the algorithm adapting Clest to robust sparse K-means. We use CER to assess the agreement of two partitions. Since there should be more cases to determine classifiers in the learning set than to assess the predictability of classifiers in the testing set, $2/3$ of cases are used in the learning set. The clest algorithm needs the following parameters to be predetermined:

> M: the maximum number of clusters that you suspect;
>
> B: the number of times that an observed dataset $\mathbf{X}$ is randomly partitioned into a learning set and a training set;
>
> B^0: the number of times that the reference distribution is generated;
>
> β: the ‘‘significance level’’ (this ‘‘significance level’’ will be explained along with the algorithm later.);

In addition, for the robust sparse K-means algorithm, the following parameters must be predetermined:

> α: the proportion of cases to be trimmed;
>
> l_1: the L_1 tuning parameter value, determining the sparsity of weights.

Given the above parameter values, the clest algorithm proceeds as follows:
Repeat Step (a) and Step (b) for each $k' \in \{2, ..., \text{M}\}$.

- Step (a):
  - Repeat Step (a-1) and Step (a-2) B times ($b = 1, \cdots, B$).
    - Step (a-1):
      - Randomly partition a dataset, $\mathbf{X}$, into two sets, $\mathbf{L}^b$ and $\mathbf{T}^b$, where $\mathbf{L}^b$ has $2/3$ of the cases.
      - Apply robust sparse k'-means to the learning set $\mathbf{L}^b$; return k' cluster centers and p weights.

    * Step (a-2):
        - Apply the classifiers to the training set $\mathbf{T}^b$; return cluster labels $\mathscr{C}_{c,\mathbf{T}^b}$.

          That is, given weights and cluster centers from Step (a-1), the weighted distances between each case and every cluster center are calculated. Then the cases in T^b are assigned to the cluster whose cluster center is the closest in weighted squared Euclidean distance.
        - Apply robust sparse k'-means to the training set $\mathbf{T}^b$; return cluster labels $\mathscr{C}_{r,\mathbf{T}^b}$.
        - Compute $\text{CER}_{k',b}$=$\text{CER}(\mathscr{C}_{c,\mathbf{T}^b},\mathscr{C}_{r,\mathbf{T}^b})$.
  - Obtain $\text{CER}_{k',1}$,...,$\text{CER}_{k',B}$. Let $\text{CER}_{k'}$=median$\{\text{CER}_{k',1},...,\text{CER}_{k',B}\}$.

- Step (b):
  - Generate B^0 samples under the reference distribution.
  - Repeat Step (a) for each sample with B=1 and compute:
    * $\text{CER}^0_{k'} = \text{median}\{\text{CER}^{*1}_{k'} \dots \text{CER}^{*B^0}_{k'}\}$.

      $\text{CER}^0_{k'}$ is the estimate of expected value of CER under the null hypothesis that all the cases are from the same cluster, k'=1.
    * $p_{k'} = |\{\text{CER}^{*i}_{k'}; \text{CER}^{*i}_{k'} < \text{CER}_{k'}\}|/B^0$.

      This $p_{k'}$ is a p-value; it is the probability of observing CER more extreme (small) than the observed CER under the null hypothesis.
    * Return d_k standardized measurement of agreement:

$$d_k = \text{CER}_k - \text{CER}^0_k.$$

After performing Step (a) and Step (b) for all the $k' \in \{1, \cdots, M\}$, finally we estimate of the number of clusters:

$$\hat{k} = \begin{cases} \underset{k\in\{k;p_k\leq\beta\}}{\text{argmin}} \; d_k & \text{if } \{k; p_k \leq \beta\} \neq \emptyset \\ 1 & \text{otherwise} \end{cases}$$

	Model							
	1 (*out*=500)	2 (*out*=500)	3	4	5	6	7	8
α	1/20	1/20	1/20	0.1	0.1	0.2	0.1	1/20

Table 4.1: The trimming proportion α for the simulation studies of Clest.

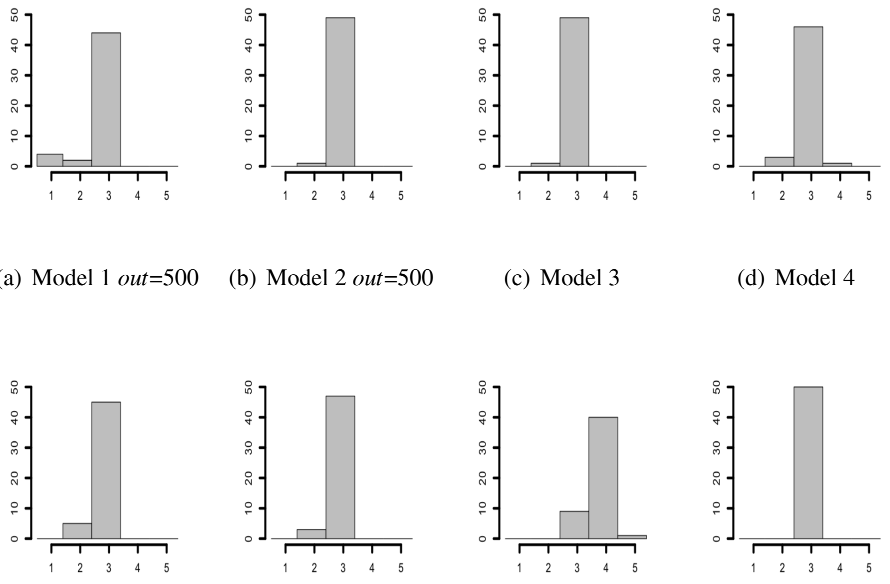
In the algorithm described above, there is a minor modification from the original clest algorithm. In the original clest algorithm, each reference dataset is partitioned into a learning set and a testing set B times. However, we set B to be 1 for the reference dataset to ease the computational cost. We found that even if we partition the reference dataset more than one time and compute the reference CERs of training sets, the CERs on the training sets were very similar, because the reference dataset forms a uniform cloud. The user still has to predetermine the number of times B that the observed dataset $\mathbf{X}$ is partitioned into a learning set and a testing set.

4.4 Simulation studies with Clest

Now we examine the performance of Clest with robust sparse K-means in the 8 simulation models (see sections 2.3.6 and 3.3). In this simulation study, we find that it is very challenging to find the true number of clusters when the clusters have considerable overlap ($\mu = 1$). Therefore, we examine the performance of the algorithm on datasets generated from models with $\mu = 1$ and 2. We implemented the code for `Clest` in the `R` package `RSKC` and we use it for the simulation study.

Table 4.1 shows the values of the trimming proportion, α, that were used in each simulation Model. The true percentage of contaminated cases are used for α values for models 4, 5, 6 and 7. For models 1, 2, 3 and 8, α levels are set so that at least one case is trimmed in the testing set. The L_1 tuning parameter value is set to 7.862 for models with $\mu = 2$ and for models with $\mu = 1$. These values are obtained as in Section 2.3.7. Clest parameters are set to (B, B^0, β, M)=(10, 20, 0.05, 5). We use PCA reference distribution to generate the reference dataset.

Figure 4.1 shows the histogram of the estimates of the number of clusters over 50 datasets generated from each Model with $\mu = 2$. The modes of the histogram are



(a) Model 1 *out*=500 (b) Model 2 *out*=500 (c) Model 3 (d) Model 4

(e) Model 5 (f) Model 6 (g) Model 7 (h) Model 8

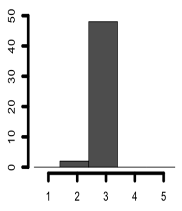
Figure 4.1: The results of Clest on datasets generated from 8 models with $\mu = 2$. The histogram of the estimated number of clusters from Clest over 50 generated datasets for each model.

at $\hat{K} = 3$ for all the models except Model 7. In particular, Clest does an excellent job for Models 1 and 2: Clest provides $\hat{K}$=3 for all 50 generated datasets of each Model. This indicates that the estimate of Clest is reliable for these types of datasets. In Model 7, Clest returns $\hat{K} = 4$ most frequently. This could be because the α value is too small. Since Model 7 contains 6 contaminated cases, if all the 6 cases are randomly selected in the testing set, it has over 30% (=6/20) contamination. Then $\alpha = 0.1$ of robust sparse K-means is too small to capture all the contaminated cases.

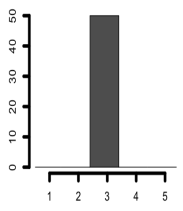
We performed Clest with larger, or more ‘‘conservative’’ α levels on datasets generated from the 8 models with $\mu = 2$. The αs are set so that robust sparse 3-

	Model							
	1 (*out*=500)	2 (*out*=500)	3	4	5	6	7	8
α	1/20	1/20	1/20	6/20	6/20	12/20	6/20	2/20

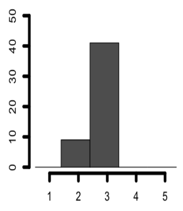
Table 4.2: The conservative trimming proportion values for the simulation studies of Clest.



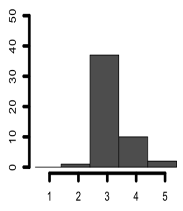
(a) Model 1 *out*=500



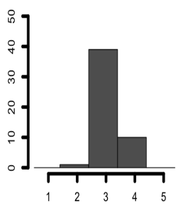
(b) Model 2 *out*=500



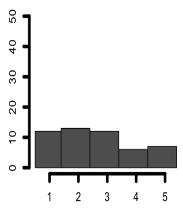
(c) Model 3



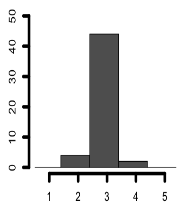
(d) Model 4



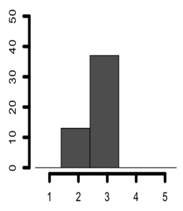
(e) Model 5



(f) Model 6



(g) Model 7



(h) Model 8

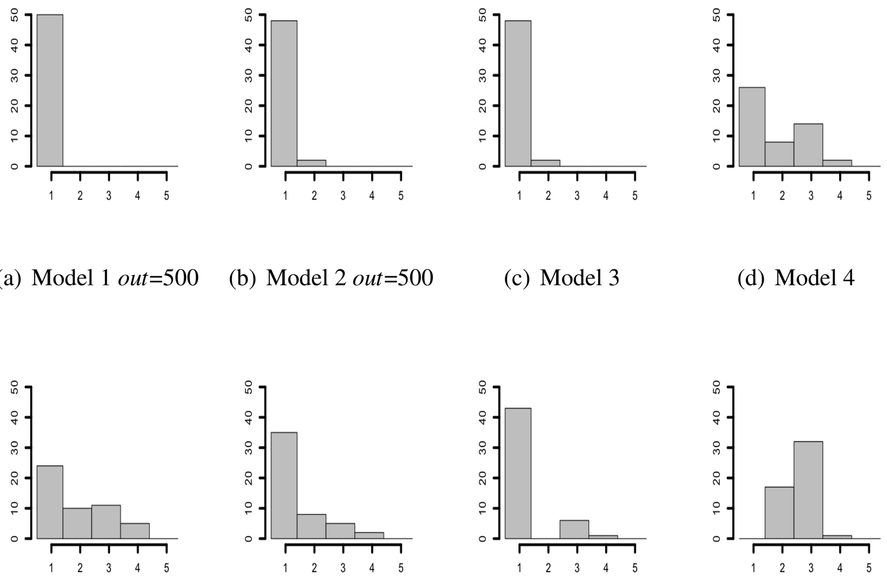
Figure 4.2: The results of Clest on datasets generated from 8 models with $\mu = 2$. The histogram of the estimated number of clusters from Clest over 50 generated datasets for each model. We use the conservative trimming proportion values.

means can capture all the outliers when they are all in the testing set. Table 4.2 shows the conservative α level for each Model. The results are in Figure 4.2. The results for models 1 to 3 are almost the same as ones from Figure 4.1, which makes sense because the α levels are the same for these Models. The performances of Clest in Models 4, 5, 7, and 8 improve in that Clest returns $\hat{K} = 3$ in more generated datasets. However, the performance of Clest in Model 6 become worse. Since Model 6 contains 12 contaminated cases, the α level must be set to at least 60% (=12/(N/3)) to capture all the contaminated cases in case all of them are in the testing set. Since only 40% of the cases are used to define the cluster centers in the testing set, Clest performs poorly. From these results, it is not recommended to use Clest algorithm if one suspects that more than 50% of the cases in a testing set could be contaminated. That is, one should not use Clest to select the number of clusters if one suspects that more than 17% (=1/6 = 50% of the 1/ 3 of the total number of the cases) of the original datasets are contaminated.

Figure 4.3 shows the simulation results of the models with μ=1 when the trimming proportion, α levels are set as in Table 4.1. In this case, the clusters overlap more and are harder to distinguish. The histograms of the estimated number of clusters for all models except Model 8 have modes at $\hat{K} = 1$. This means that none of the observed CER of $K = 2, 3$, 4 or 5 is significantly smaller than the reference CER for the models. When clusters overlap more, Clest tends to underestimate the true number of clusters.

If we know that there is a cluster structure, i.e., $K > 1$, then we should rather choose the $\hat{K}$, which minimizes the test statistics, d_k. That is, the significance level β should be 1.00. Figure 4.4 shows the simulation results of Clest with significance levels, β=1.00 and α as in Table 4.1 on the datasets generated from the models with $\mu = 1$. The performance of Clest with $\beta = 1.00$ improves substantially over the performance of Clest when $\beta = 0.05$ on them: now the modes of all models are at $\hat{K} = 3$. This indicates that if we know that and there is a cluster structure and the clusters are substantially overlapping, then we should rather use significance level β= 1.

The simulations showed that Clest with robust sparse K-means could identify the number of clusters when the clusters are reasonably well separated. Since in the clest algorithm robust sparse K-means is performed on subsets of dataset (i.e.,



(a) Model 1 *out*=500 (b) Model 2 *out*=500 (c) Model 3 (d) Model 4

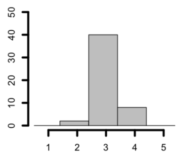
(e) Model 5 (f) Model 6 (g) Model 7 (h) Model 8

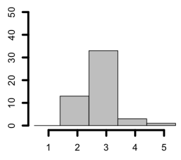
Figure 4.3: The results of Clest on datasets generated from 8 models with $\mu = 1$. The histogram of the estimated number of clusters from Clest over 50 generated datasets for each model.

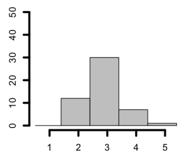
learning set or testing set), the trimming proportion, α should be selected close to the potential proportion of outliers in the training subset of dataset. Due to the random partitioning of observed datasets or reference datasets into the learning and testing sets, the proportion of outliers in the subset of the dataset could be larger than in the observed dataset. Therefore, we recommend that the user of Clest choose the conservative values of α levels. It was necessary to know that there were more than two clusters if clusters were not well separated.

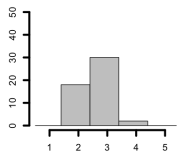
Lastly, Clest with robust sparse K-means is not suitable in the following situations:

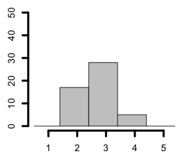
- The user does not know if a dataset has a cluster structure and if it does, the

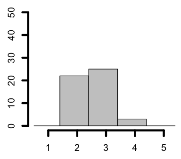

(a) Model 1 *out*=500

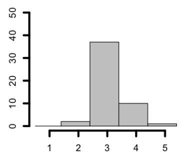

(b) Model 2 *out*=500


(c) Model 3


(d) Model 4


(e) Model 5


(f) Model 6


(g) Model 7

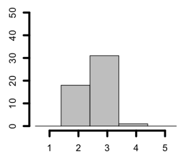

(h) Model 8

Figure 4.4: The results of Clest on datasets generated from 8 models with $\mu = 1$. The significance level, β is set to 1. The histogram of the estimated number of clusters from Clest over 50 generated datasets for each model.

clusters might overlap considerably.

- The dataset contains quite a few outliers. A rule of the thumb is that if one suspects that the proportion of outliers in the dataset is greater than 17% then one should not use Clest with robust sparse K-means.

Chapter 5

Data analysis

5.1 Before moving to the data analysis

5.1.1 The dissimilarity measure based on Pearson's correlation

We will examine a microarray dataset in this chapter. Until now, our discussions are mostly based on the squared Euclidean dissimilarity measure. In microarray datasets, however, the dissimilarity measure most commonly used is based on Pearson's correlation across features (Hardin et al. [9]). Let $\bar{x}_i$ denote the average over the features of the i^{th} case:

$$\bar{x}_i = \frac{1}{p}\sum_{j=1}^{p} x_{ij}.$$

Pearson's correlation coefficient across features is defined as:

$$r(\mathbf{x}_i, \mathbf{x}_{i'}) = \frac{(\mathbf{x}_i - \bar{x}_i\mathbf{1})^T(\mathbf{x}_{i'} - \bar{x}_{i'}\mathbf{1})}{||\mathbf{x_i} - \bar{x}_i\mathbf{1}||||\mathbf{x_{i'}} - \bar{x}_{i'}\mathbf{1}||}.$$

Pearson's correlation can be converted to dissimilarities, $d^r(\mathbf{x}_i, \mathbf{x}_{i'})$, for instance, by setting:

$$d^r(\mathbf{x}_i, \mathbf{x}_{i'}) = 1 - r(\mathbf{x}_i, \mathbf{x}_{i'}) \in [0, 2]. \tag{5.1}$$

With this formula, cases with a high positive correlation receive a dissimilarity coefficient close to zero, whereas cases with a strongly negative correlation will be considered very dissimilar. This however produces the odd result that for a completely uncorrelated pair we have $d^r(\mathbf{x}_i, \mathbf{x}_{i'}) = 1$. For this reason, some prefer to use $d^{r*} = 1 - |r(\mathbf{x}_i, \mathbf{x}_{i'})|$, in which case also variables with a strongly negative correlation will be assigned a small dissimilarity. Lance and Williams [13] compared these formulas by means of real data, and concluded that (5.1) was unequivocally the best. Therefore, we will use (5.1) as Pearson's correlation dissimilarity measure. One of the reasons for its popularity is that the correlation is scale invariant. It captures the similarity in 'shape' of two cases.

Now we illustrate the difference between the squared Euclidean dissimilarity measure and Pearson's correlation dissimilarity measure with a simple example. Consider the two cases:

$$\mathbf{x}_1^* = [1,2,3]^T \text{ and } \mathbf{x}_2^* = [4,5,6]^T.$$

In order to calculate their squared Euclidean dissimilarity measure, we first normalize $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$:

$$\mathbf{x}_1 = [-0.7, -0.7, -0.7]^T \text{ and } \mathbf{x}_2 = [0.7, 0.7, 0.7]^T.$$

Then their dissimilarity is $||\mathbf{x}_1 - \mathbf{x}_2||^2 = 2.42 \neq 0$. On the other hand, since $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$ have the same shape, their Pearson's correlation dissimilarity is $d^r(\mathbf{x}_1^*, \mathbf{x}_2^*) = 0$.

Maitra and Ramler [17] introduced the relationship between Pearson's correlation dissimilarity measure and the squared Euclidean distance dissimilarity measure between standardized profiles. Pearson's correlation dissimilarity measure is incorporated in a clustering algorithm as follows. The raw dataset $\mathbf{X}^*$ is transformed into $\tilde{\mathbf{X}} = [\tilde{x}_{ij}] \in \mathbb{R}^{N \times p}$ as:

$$\begin{aligned} \tilde{x}_{ij} &= \frac{x_{ij}^* - \bar{x}_i^*}{\sqrt{\frac{\sum_j (x_{ij}^* - \bar{x}_i^*)^2}{p}}} \\ &= \sqrt{p}\frac{x_{ij}^* - \bar{x}_i^*}{||\mathbf{x}_i^* - \bar{x}_i^* \mathbf{1}||}. \end{aligned} \tag{5.2}$$

Notice that in the normalization (2.2), the columns of the data matrix are centered then scaled while in the transformation (5.2), the rows of the data matrix are centered then scaled. Then the squared Euclidean distance between two transformed vectors, $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_{i'}$, becomes proportional to Pearson's correlation dissimilarity between the two original vectors, $\mathbf{x}_i^*$ and $\mathbf{x}_{i'}^*$.

$$\begin{aligned}
||\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{i'}||^2 &= p\,||\frac{\mathbf{x}_i^* - \bar{x}_i^*\mathbf{1}}{||\mathbf{x}_i^* - \bar{x}_i^*\mathbf{1}||} - \frac{\mathbf{x}_{i'}^* - \bar{x}_{i'}^*\mathbf{1}}{||\mathbf{x}_{i'}^* - \bar{x}_{i'}^*\mathbf{1}||}||^2 \\
&= p\left(\frac{||\mathbf{x}_i^* - \bar{x}_i^*\mathbf{1}||^2}{||\mathbf{x}_i^* - \bar{x}_i^*\mathbf{1}||^2} - 2\frac{(\mathbf{x}_{i'}^* - \bar{x}_{i'}^*\mathbf{1})'(\mathbf{x}_{i'}^* - \bar{x}_{i'}^*\mathbf{1})}{||\mathbf{x}_{i'}^* - \bar{x}_{i'}^*\mathbf{1}||||\mathbf{x}_i^* - \bar{x}_i^*\mathbf{1}||} + \frac{||\mathbf{x}_{i'}^* - \bar{x}_{i'}^*\mathbf{1}||^2}{||\mathbf{x}_{i'}^* - \bar{x}_{i'}^*\mathbf{1}||^2}\right) \\
&= 2p(1 - r(\mathbf{x}_i^*, \mathbf{x}_{i'}^*)) \\
&= 2p\,d(\mathbf{x}_i^*, \mathbf{x}_{i'}^*).
\end{aligned}$$

After the raw dataset is transformed as in (5.2), we can perform clustering algorithms on the transformed dataset. This transformation can be coded simply in `R` as `t(scale(t(data)))`.

5.1.2 The revised silhouette

In the data analysis of this chapter, we will assess how well each case is clustered by various clustering methods. Intuitively, if a case is placed ''deep'' in its cluster then it should be considered to be well clustered and if a case is placed at the edge between its cluster and its second closest cluster then it should not be considered so well clustered. Therefore, we use the following simple measure to assess how well cases are clustered. We call this measure the revised silhouette, because it is very similar to the silhouette proposed by Rousseeuw [20]. We explain the difference between the two measures and the reason why we choose to use the revised silhouette in Appendix A.2.

The revised silhouette compares the distance from a case to its cluster centers. Let $\tilde{\mathbf{X}}$ denote the transformed dataset as in (5.2). We define $a'(i)$ to be the squared Euclidean distance between the i^{th} case and its cluster center. Denote the cluster to which the i^{th} case belongs by $IC1(i)$. Then the closest cluster center vector from the i^{th} case is $\bar{\bar{x}}_{IC1(i),j}$ and:

$$a'(i) = \sum_{j=1}^{p} (\tilde{x}_{ij} - \bar{\bar{x}}_{IC1(i),j})^2.$$

We also define $b'(i)$ to be the squared Euclidean distance between the i^{th} case and its second closest cluster center. Denote the cluster such that its cluster center is the second closest from the i^{th} case by $IC2(i)$. Then the second closest cluster center vector from the i^{th} case is $\bar{\bar{x}}_{IC2(i),j}$ and:

$$b'(i) = \sum_{j=1}^{p} (\tilde{x}_{ij} - \bar{\bar{x}}_{IC2(i),j})^2.$$

We will use the following ''revised silhouette width'' to assess how well the i^{th} case is clustered:

$$rev.silhouette(i) = \frac{b'(i) - a'(i)}{b'(i)}.$$

The revised silhouette width takes values between 0 and 1. One can see that $rev.silhouette(i)$ =1 when $a'(i) = 0$, meaning that the case is placed ''deep'' in its cluster and perfectly well clustered. This occurs only if the i^{th} case itself forms the cluster center. If the i^{th} case is placed at the middle of the two clusters, then $b'(i) = a'(i)$, and $rev.silhouette(i)$ is 0. Note that to assess partitions from the sparse methods, the *weighted* squared Euclidean distances are used to calculate $b'(i)$ and $a'(i)$ by simply weighting the j^{th} feature component of the squared Euclidean distance by the corresponding weight w_j.

5.2 The dataset of van't Veer et al. [27]

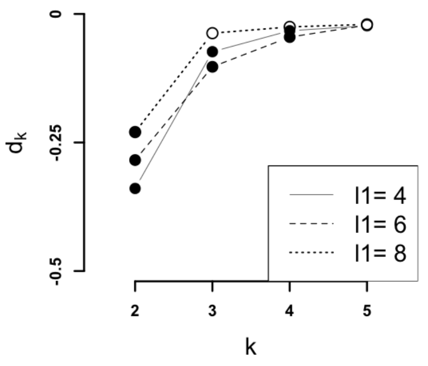
In this section we analyze a microarray dataset. The data consists of 4751 gene expressions for 78 primary breast tumors, and were previously analyzed by van't Veer et al. [27]. The dataset is available at:
`http://www.nature.com/nature/journal/v415/n6871/suppinfo/415530a.html`.

The authors applied a supervised classification method and found a subset of 70 genes that identified the two groups according to whether the patient developed distant metastasis within 5 years or not. However, the threshold 'within 5 years' appears to be somewhat arbitrary. Experimentalists are also interested in the oestrogen receptor (ER) status of patients because the ER status is essential clinical information for the treatment of metastatic disease. There is a clinical consensus that if the ER status of a patient is positive then treatment tends to work better (Barnes et al. [1]). In this section we will try to identify features that can cluster patients according to the partition based on oestrogen receptor (ER) status. We represent the ER status as the true biological partition $\mathscr{C}^{ER}$.

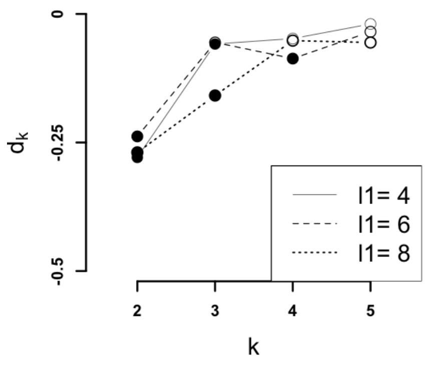
Throughout our analysis, we use a transformed dataset so that the clustering algorithm based on the squared Euclidean distance uses Pearson's correlation dissimilarity measure. The outline of our analysis is as following. First we select the number of clusters, K, via Clest with robust sparse K-means and sparse K-means. Since we have no information about potential outliers nor the number of clustering features, we run Clest with various sets of (α, l_1) values. Then given the estimate of the number of clusters, K', we run the 4 clustering algorithms: K'-means, trimmed K'-means, sparse K'-means, and robust sparse K'-means. Again, various α and l_1 values are examined. We compare CERs between partitions from algorithms and the partition according to the ER status $\mathscr{C}^{ER}$. In order to facilitate the comparison between the robust sparse K'-means and sparse K'-means, we pick one partition result for each method. We select a partition which is formed by the smallest number of features and reflects the ER status well from each sparse method. We use a call rate plot and the revised silhouette as the main tools for the comparison. Finally we provide a conclusion.

5.2.1 Selection of K

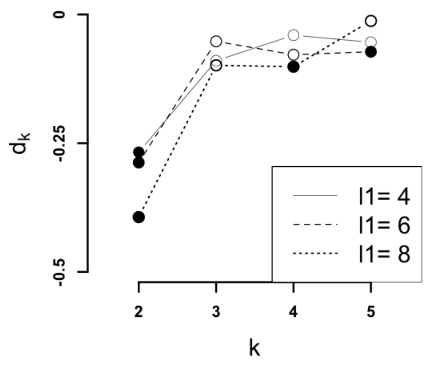
We run Clest with robust sparse K-means and sparse K-means on the dataset to choose the number of clusters. We have no information about potential outliers nor the number of clustering features. Therefore, we run Clest with all possible combinations of $\alpha = (0, 1/78, 5/78, 10/78)$ and $l_1 = (4, 6, 8)$. Clest has four parameters: B the number of times observed data is partitioned into learning set and testing set, B^0 the number of times reference datasets are generated, β the significance level and the maximum number M of clusters that are present. They are set to (B, B^0, β, *M*)= (10, 20, 0.05, 5). Since there are no missing values in the dataset, we employ the PCA reference distribution. For this analysis we use the function `Clest` in package `RSKC`.



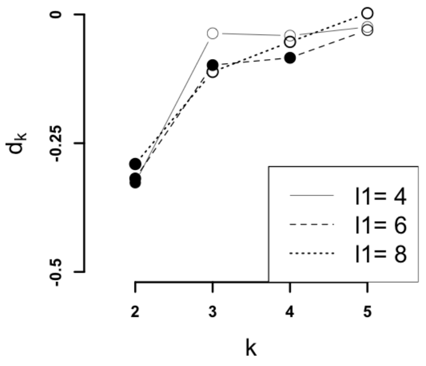
(a) α=0/78



(b) α=1/78



(c) α=5/78



(d) α=10/78

Figure 5.1: Solid circles represent significant p-values. Clest chooses K=2 as a solution for all pairs of α and l_1 values.

The circles in Figure 5.1 show the standardized measure of agreement, d_k given α, l_1 and k. We indicate the d_k values that correspond to significant p-values with solid circles and insignificant p-values with open circles. One run of Clest returns one line in the plot. It is clear that $K = 2$ is chosen for all the combinations of l_1 and α. From this result, we fix K to be 2 and perform all the clustering algorithms.

5.2.2 The comparison of performances

We now perform 2-means, sparse 2-means, trimmed 2-means and our proposed robust sparse 2-means and compare the resulting partition with the partition according to the ER status of the cases via CER. We used the function `kmeans` (in package `stat`) and `KMeansSparseCluster` in package `sparcl` to perform 2-means and sparse 2-means respectively. We use our function `RSKC` in the package `RSKC` to perform trimmed 2-means and robust sparse 2-means. Since there is no information about the clustering features nor outliers, we examine the following L_1 tuning parameter values: $l_1 = 1.05, 1.5, 2, 4, 6, 8, 12, 20, 30, 50,$ and 70 for sparse methods and the following trimming proportion: $\alpha = 1/78, 5/78,$ and $10/78$ for the robust methods. That is, we performed 2-means, trimmed 2-means, sparse 2-means and robust sparse 2-means 1, 3, 11, and 33 (=11$\times$ 3) times, respectively.

l_1	CER	# non zero weights
1.05	0.21	3
1.50	0.14	5
2	0.12	8
4	0.12	36
6	0.10 *	74
8	0.10 *	127
12	0.10 *	378
20	0.10 *	1397
30	0.10 *	4751
50	0.10 *	4751
70	0.10 *	4751

Table 5.1: The result from sparse 2-means. The * indicates partitions that agree with each other.

The 2-means algorithm returns a CER of 0.12. Trimmed 2-means with $\alpha = 1/78, 5/78$ and $10/78$ returns CERs of 0.12, 0.10 and 0.12, respectively. As for

	CER			# nonzero weight		
$l_1 \backslash \alpha$	1/78	5/78	10/78	1/78	5/78	10/78
1.05	0.23	0.19	0.19	2	2	2
1.5	0.14	0.14	0.14	4	3	4
2	0.12	0.12	0.14	7	6	6
4	0.12	0.12	0.12	31	29	28
6	0.10*	0.10*	0.10*	72	72	66
8	0.10*	0.10*	0.10*	130	131	126
12	0.10*	0.10*	0.10*	379	366	365
20	0.10*	0.10*	0.10*	1397	1386	1333
30	0.10*	0.10*	0.10*	4301	4081	3906
50	0.10*	0.10*	0.10*	4751	4751	4751
70	0.10*	0.10*	0.10*	4751	4751	4751

Table 5.2: The result from robust sparse 2-means. The * indicates partitions that agree with each other.

the sparse methods, sparse 2-means and robust sparse 2-means both return CERs of 0.10 for reasonably large L_1 tuning parameter values: $l_1 \geq 6$ (Table 5.1 and Table 5.2). On the other hand, the sparse methods return poor CERs when the L_1 tuning parameter values are too small. This is probably because feature weights are too sparse (For example, when the L_1 tuning parameter value is 4, there are only 36 features with non-zero weights for sparse 2-means). However, for all reasonably large L_1 tuning parameter values, the sparse methods consistently return as good or better partition compared to the nonsparse methods.

Interestingly all the partitions from robust sparse 2-means (regardless of the α levels) and sparse 2-means are exactly the same for $l_1 \geq 6$. We denote these partitions by ‘‘*’’ in Table 5.1 and Table 5.2. When the L_1 tuning parameter is set to 6, sparse 2-means and robust sparse 2-means return partitions using about 70 features. The partitions resulting from larger values of the L_1 tuning parameter use substantially more features. If the performance of the partition is the same, then the experimenters would want to use fewer features, because 70 features are much easier to interpret than 4751 features. For this reason, we will compare the performance of sparse 2-means and robust sparse 2-means with $l_1 = 6$.

We wonder if there are any outliers in the dataset. We cannot simply treat the trimmed cases as outliers, because robust sparse K-means trims a predetermined number of cases, regardless of the existence of outliers. Moreover, in this way, it would be impossible to find outliers via sparse K-means simply because it does not trim cases. For these reasons, we use the following rule based on ‘‘cut off’’ values to find the potential outliers. The cut off value is calculated as follows. Given a partition result, the median $\tilde{m}$ and MAD $\tilde{\tau}$ of all the weighted squared Euclidean distances to their cluster centers are calculated. Then the cut off value is defined as:

$$\text{cut off} = \tilde{m} + threshold\ \ \tilde{\tau},$$

where we choose $threshold$ values between 1 and 5. The cases that are greater than this cut off value are considered as potential outliers.

CERs are again calculated, excluding the potential outliers. Note that the first $\lfloor \alpha N \rfloor$ cases that are excluded from the CER calculation between the robust sparse 2-means partition and the $\mathscr{C}^{ER}$ are the trimmed cases. This is because the $\lfloor \alpha N \rfloor$ trimmed cases have the top $\lfloor \alpha N \rfloor$ largest weighted squared Euclidean distances to their cluster centers.

Figure 5.2 is the call rate plot for four algorithms: sparse 2-means with l_1=6 and robust sparse 2-means with (l_1,α)=(6,1/78), (6,3/78), (6,10/78). One can see that for large value of the $threshold$, the CERs from the four algorithms are almost the same. This makes sense because the partitions of all four algorithms agree. As the $threshold$ value decreases, robust sparse 2-means with moderately large α, ($\alpha = 5/78$ and $10/78$), returns smaller CERs than sparse 2-means. This means that some cases excluded from the CER calculation of the robust sparse 2-means are, indeed, misclassified cases. In fact, robust sparse 2-means with $\alpha = 5/78$ and $10/78$ trim 1 and 2 misclassified cases, respectively. Therefore the misclassified cases end up placed far from the cluster centers and found as potential outliers. However, sparse 2-means fails to find the misclassified cases as potential outliers.

Robust sparse 2-means trimmed some of the cases that are misclassified by other algorithms. Now, we wonder how well the misclassified cases and trimmed cases are clustered in the partitions from the sparse methods. Figure 5.3 shows the revised silhouette widths of all cases. The black lines are the revised silhouette

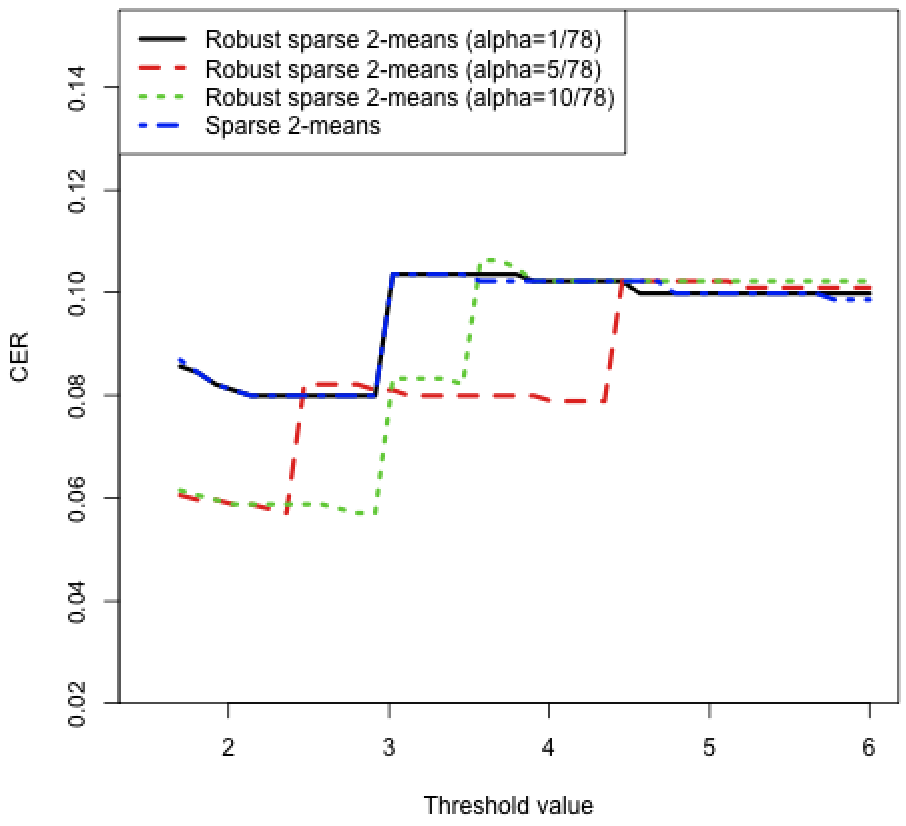


Figure 5.2: The call rate plot of the 4 algorithms. The results of the sparse 2-means and robust sparse 2-means with $\alpha = 1/78, 5/78$ and 10/78. The results of sparse methods with l_1=6 are shown.

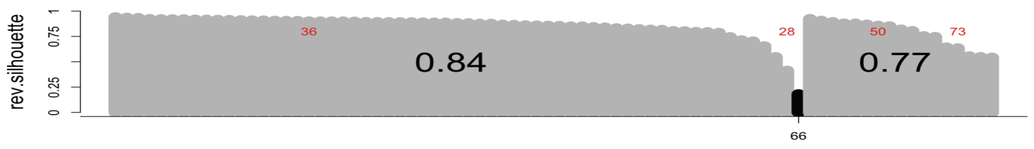
widths of the trimmed cases. The red digits show the case indices of misclassified cases. Robust sparse 2-means with $\alpha = 5/78$ and $\alpha = 10/78$ capture 1 (the 28^{th} case) and 2 (the cases 28^{th} and the 73^{th}) misclassified cases, respectively. Interestingly, the revised silhouette widths of most trimmed cases are almost zero, which means that they lie at the border of the two clusters. In this example, robust sparse 2-means trims the cases whose class memberships are ''ambiguous'' and defines the cluster centers without such cases. The misclassified cases are excluded in the definition of the cluster centers and placed ''far'' from their cluster centers and identified as potential outliers. As a result, the partition from the robust sparse 2-means without the potential outliers becomes more desirable than the partition from the sparse 2-means without the potential outliers. Figures 5.4 and 5.5 show the weighted distances between the cases, including trimmed cases, and their cluster centers from the result with $l_1 = 6$.

Lastly, figures 5.6 and 5.7 show the spread of the feature weights over the 4751 features from robust sparse 2-means and sparse 2-means for selected l_1 values. The x-axis shows the feature indices. The distributions are very similar among the different algorithms for each l_1 level.

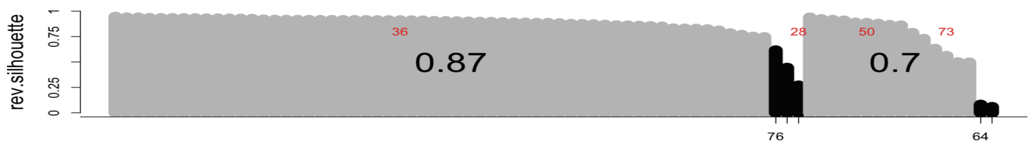
In summary, we have two conclusions:

- The sparse methods perform better than the nonsparse methods to identify the ER status of each case. The sparse methods find small subsets of about 70 features which create the two clusters of ER status. The performances of the sparse methods with about 70 subset features are as good as the performances of the sparse methods with the full set of features.

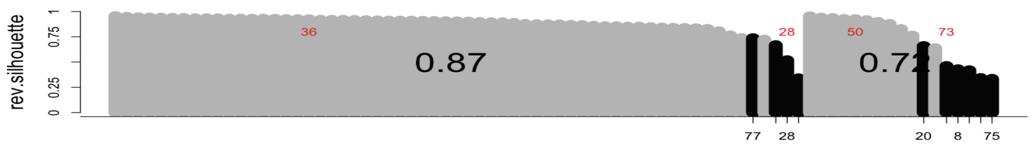
- Robust sparse 2-means trims some of the cases that are misclassified by sparse 2-means. The misclassified cases are excluded from the calculation of cluster centers. As a result, the misclassified cases are placed far from their cluster centers and the call rate plot is able to identify them as potential outliers. Robust sparse 2-means found a partition more similar to the partition based on ER status by excluding potential outliers than sparse 2-means did.



(a) α=1/78

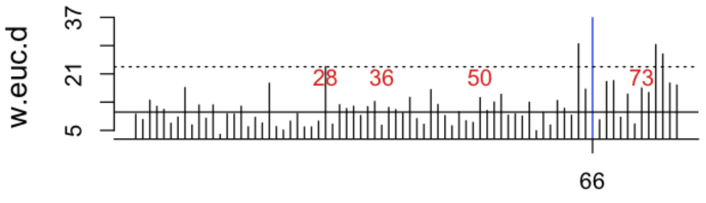


(b) α=5/78

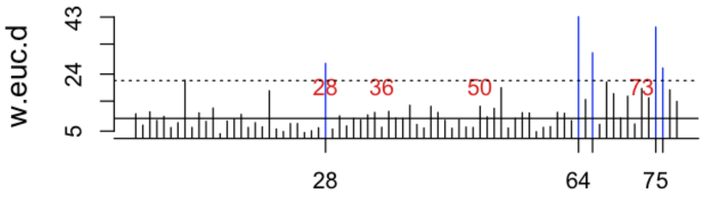


(c) α=10/78

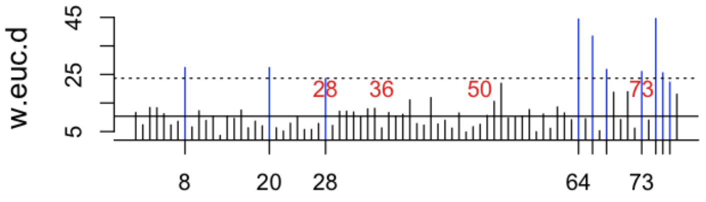
Figure 5.3: The silhouettes of the partition returned by the robust sparse 2-means clustering. The L_1 tuning parameter value is 6.



(a) α=1/78



(b) α=5/78



(c) α=10/78

Figure 5.4: The weighted squared Euclidean distances between all cases and their cluster centers from the result of robust sparse 2-means with $l_1 = 6$.

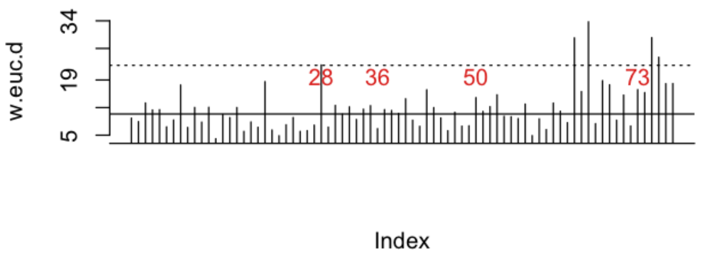
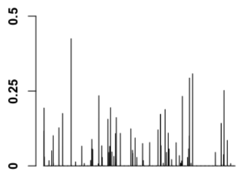
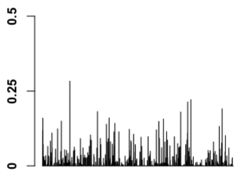


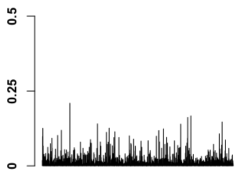
Figure 5.5: The weighted squared Euclidean distances between the cases and their cluster centers for the result of sparse 2-means with $l_1 = 6$.



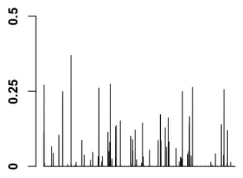
(a) l_1=6



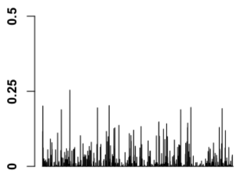
(b) l_1=12



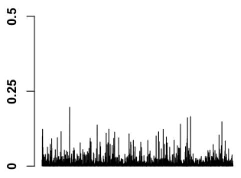
(c) l_1=70



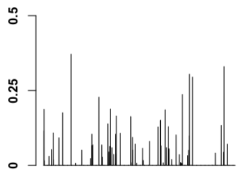
(d) l_1=6



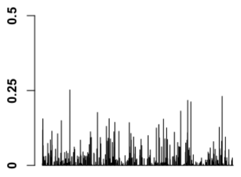
(e) l_1=12



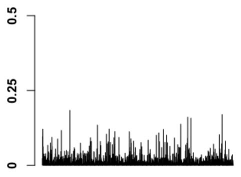
(f) l_1=70



(g) l_1=6

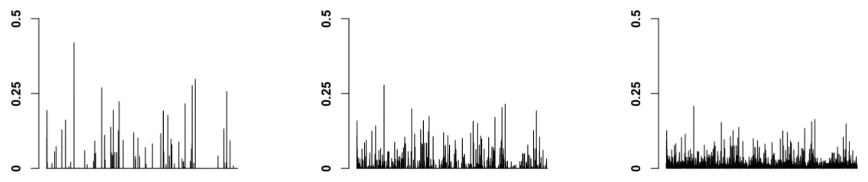


(h) l_1=12



(i) l_1=70

Figure 5.6: The nonzero weights from robust sparse 2-means for selected l_1 values. The three rows show the results of robust sparse 2-means with α=1/78, 5/78 and 10/78, respectively.



(a) l_1=6 (b) l_1=12 (c) l_1=70

Figure 5.7: The nonzero weights from sparse 2-means for selected l_1 values.

Chapter 6

Conclusion

We developed a robust clustering method that is also able to perform variable selections. Our proposed method can handle datasets with missing values. We adapted Clest to our robust sparse K-means method to provide an automatic way of selecting the number of clusters. We implemented our proposed method in the `R` package `RSKC`.

Our simulation studies show that our proposed robust sparse K-means clustering was robust against various kinds of contaminated datasets. They also show that Clest with robust sparse K-means could identify the number of clusters when the clusters were reasonably well separated. It was necessary to know that there were more than two clusters if clusters were not well separated and the dataset is contaminated. The application of robust sparse K-means on the microarray datasets of van't Veer et al. [27] showed that our proposed method could identify the ER status of 78 breast tumors better than other methods. The cases which were misclassified by other algorithms were trimmed by robust sparse 2-means. Therefore, the misclassified cases ended up far from their cluster centers defined by robust sparse K-means and we could find them as potential outliers.

As future research, an automatic robust method of selecting the L_1 tuning parameter values is of interest. Since both current robust clustering `R` packages, `trimcluster` and `RSKC`, apply Lloyd's algorithm, one might implement them with Hartigan and Wong's algorithm, which is computationally more intense but tends to achieve a better local optima.

Bibliography

[1] D. Barnes, R. Millis, L. Beex, S. Thorpe, and R. Leake. Increased use of immunohistochemistry for oestrogen receptor measurement in mammary carcinoma: the need for quality assurance. *European Journal of Cancer*, 34, 1998.

[2] H. Chipman and R. Tibshirani. Hybrid hierarchical clustering with applications to microarray data. *Biostatistics*, 7(2):286–301, 2005.

[3] H. A. Chipman, E. I. George, and R. E. Mcculloch. Making sense of a forest of trees. In *Proceedings of the 30th Symposium on the Interface*, pages 84–92, 1998.

[4] J. A. Cuesta-Albertos, A. Gordaliza, and C. Matran. Trimmed k-means: An attempt to robustify quantizers. *The Annals of Statistics*, 25(2):553–576, 1997.

[5] S. Duboit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7), 2002.

[6] E. W. Forgy. Cluster analysis of multivariate data: efficiency vs interpretability of classifications. *Biometrics*, 21:768769, 1965.

[7] A. Gordaliza. Best approximations to random variables based on trimming procedures. *Journal of Approximation Theory*, 64, 1991a.

[8] A. Gordaliza. On the breakdown point of multivariate location estimators based on trimming procedures. *Statistics & Probability Letters*, 11, 1991b.

[9] J. Hardin, A. Mitani, L. Hicks, and B. VanKoten. A robust measure of correlation between two genes on a microarray. *BMC Bioinformatics*, 8:220, 2007.

[10] J. A. Hartigan. *Clustering Algorithm*. John Wiley and Sons, Inc., 1975.

[11] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[12] L. Hubert. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[13] G. N. Lance and W. T. Williams. Inver: A program for the computation of distance-measures between attributes of mixed types. *Australian Computer Journal*, 11(1):27–28, 1979.

[14] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on information theory*, 28(2):129–136, 1982.

[15] D. MacKay. *Chapter 20. An Example Inference Task: Clustering*. Cambridge University Press, 2003.

[16] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, eds. L.M. LeCam and J. Neyman*, 1967.

[17] R. Maitra and I. P. Ramler. A k-mean-directions algorithm for fast clustering of data on the sphere. *Journal of Computational and Graphical Statistics*, 19:377–396, 2010.

[18] G. Milligan and M. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 2(50):159–179, 1985.

[19] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

[20] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Computational and Applied Mathematics*, 20, 1987.

[21] G. A. F. Seber. *Multivariate Observations*. Wiley, 2004.

[22] D. Sparks. Algorithm AS 58. Euclidean cluster analysis. *Euclidean cluster analysis*, 22:126–130, 1973.

[23] H. Steinhaus. Sur la division des corps matriels en parties. *Bull. Acad. Polon. Sci*, 4(12):801–804, 1956.

[24] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society.*, 58(11), 1996.

[25] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of Royal Statistical Society, Series B*, 63, 2000.

[26] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2000.

[27] L. J. van't Veer, H. Dal, M. J. van de Vijver, Y. D. He, A. A. M. Hart, M. Mao, H. L. Peterse, K. van der Kooy, M. J. Marton, A. T. Witteveen, G. J. Schreiber, R. M. Kerkhoven, C. Roberts, P. S. Linsley, R. Bernards, and S. H. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415, 2002.

[28] D. M. Witten and R. Tibshirani. A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490): 713–726, 2010.

Appendix A

Technical Appendix

A.1 The algorithm of Hartigan and Wong

This Appendix explains the technical details of Hartigan and Wong's algorithm. To simplify the notation, define a function:

$$R2(i,k) = \frac{n_k}{n_k+1}||\mathbf{x}_i - \bar{\mathbf{x}}_k||^2.$$

Then $R2(i,k_2)$ is the left hand side of the inequality in (2.7). Let $LIVE$ denote a live set.

Randomly choose K cases to serve as initial cluster centers. Perform the first and the second step of Lloyd's algorithm. Let all clusters belong to the live set: $C_1, \cdots, C_K \in LIVE$. Repeat 1,2 and 3 until the live set becomes empty.

1. This is the optimal-transfer (OPTRA) stage. In this stage, calculations and reallocations are made with regard to all cases relative to clusters in the live set. For all the cases, $i = 1, \cdots, N$, perform:

   (a) If $C_{IC1(i)} \in LIVE$

      i. Denote $k_1 = IC1(i)$. Compare the criterion (2.7) where k_2 is the cluster which returns the smallest $R2(i,k)$ among all clusters except the k_1^{th} cluster. If (2.7) holds, then allocate the i^{th} case into C_{k_2} and set $IC1(i) = k_2$ and $IC2(i) = k_1$. Otherwise, keep the i^{th} case in C_{k_1} and set $IC1(i) = k_1$ and $IC2(i) = k_2$.

ii. Given a cluster assignment $IC1(i)$ for all i from Step i, cluster centers $\bar{\mathbf{x}}_k$'s are updated. The two clusters that are involved in the transfer of cases, i's, at this step are now in the live set.

(b) if $C_{IC1(i)} \notin LIVE$
This step is the same as Step 1(a), except that the minimum $R2(i,k)$ is computed only over clusters in the live set.

2. Stop if the live set is empty.

3. This is the quick transfer (QTRAN) stage. This stage only checks for the memberships of recently reallocated clusters and updates the clusters.

   i. Let $k_1 = IC1(i)$ and $k_2 = IC2(i)$ and compare the criterion (2.7). If the criterion is satisfied then switch $IC1(i)$ and $IC2(i)$. The clusters that are involved in the transfer of cases at this step are now in the live set.

   ii. Given a cluster assignment $IC1(i)$ for all i from Step i, cluster centers $\bar{\mathbf{x}}_k$'s are updated. The two clusters that are involved in the transfer of cases at this step are now in the live set.

4. If no reallocation took place in the quick transfer stage, then go to the optimal-transfer stage, Otherwise go to the quick transfer stage.

A.2 The difference between Silhouette and the revised silhouette

In our data analyses, we used the revised silhouette to assess if a case is well clustered. In fact, this measure is very similar to the famous measure, Silhouette, proposed by Rousseeuw [20]. In this section, we explain the difference between Silhouette and the revised silhouette and explain why we decided to use the revised silhouette.

Silhouette is based on a measure, called the silhouette width, which assesses how well a case is assigned. The author defined a ‘‘neighbor’’ of a case to be the cluster to which the case could have been assigned if the first choice cluster had not been available. The author argued that if the i^{th} case is well clustered, the average dissimilarity between the i^{th} case and the others in the same cluster, denoted as $a(i)$, should be considerably smaller than the average dissimilarity between the i^{th} case and the cases in its neighbor, denoted as $b(i)$. The silhouette width of the i^{th} case, $silhouette(i)$, was defined to be the standardized difference of $b(i)$ and $a(i)$.

Although similar, this silhouette width introduced by Rousseeuw [20] is different from our proposed revised silhouette width above. Intuitively, Silhouette tends to ''favor'' clusters with low variation when the squared Euclidean dissimilarity measure is used.

We can measure the variation within the k^{th} cluster by the contribution from the k^{th} cluster, WSS_k, to WSS:

$$WSS_k = \sum_{l \in C_k} ||\tilde{\mathbf{x}}_l - \bar{\tilde{\mathbf{x}}}_k||^2.$$

One can see that $b(i)$ and $a(i)$, defined above, are increasing functions of the variation within the i^{th} neighbor and the variation within the cluster to which the i^{th} case belongs, respectively. To see this, let $dis(i,C_k)$ be the average dissimilarity of the i^{th} case to all cases in C_k. Then $b(i)$ is the minimum of $dis(i,C_k)$ over C_k such that k is not $IC1(i)$. This $dis(i,C_k)$ is not only a function of the distance from the i^{th} case to the k^{th} cluster center but also a function of WSS_k:

$$\begin{aligned}
dis(i,C_k) &= \text{average dissimilarity of the } i^{th} \text{ case to all cases in } C_k \\
&= \frac{1}{|C_k|} \sum_{l \in C_k} ||\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_l||^2 \\
&= \frac{1}{|C_k|} \left(\sum_{l \in C_k} ||\tilde{\mathbf{x}}_i - \bar{\tilde{\mathbf{x}}}_k||^2 + (\tilde{\mathbf{x}}_i - \bar{\tilde{\mathbf{x}}}_k)^T \sum_{l \in C_k} (\bar{\tilde{\mathbf{x}}}_k - \tilde{\mathbf{x}}_l) + \sum_{l \in C_k} ||\bar{\tilde{\mathbf{x}}}_k - \tilde{\mathbf{x}}_l||^2 \right) \\
&= ||\tilde{\mathbf{x}}_i - \bar{\tilde{\mathbf{x}}}_k||^2 + \frac{1}{|C_k|} WSS_k.
\end{aligned}$$

One can also show that $a(i)$ is a function of both $WSS_k^{(-i)}/(|C_k|-1)$ and $||\tilde{\mathbf{x}}_i - \bar{\tilde{\mathbf{x}}}_k^{(-i)}||^2$, where $WSS_k^{(-i)}$ and $\bar{\tilde{\mathbf{x}}}_k^{(-i)}$ represent the WSS_k and $\bar{\tilde{\mathbf{x}}}_k$ calculated without the i^{th} case, respecitvely. Therefore $b(i)$ could be smaller than $a(i)$ even if the i^{th} case is clustered in the closest cluster center when the variation of the closest cluster is considerably large. Then Silhouette suggests to change the cluster belonging of the i^{th} case to its neighbor. This situation can be seen in the simple example in

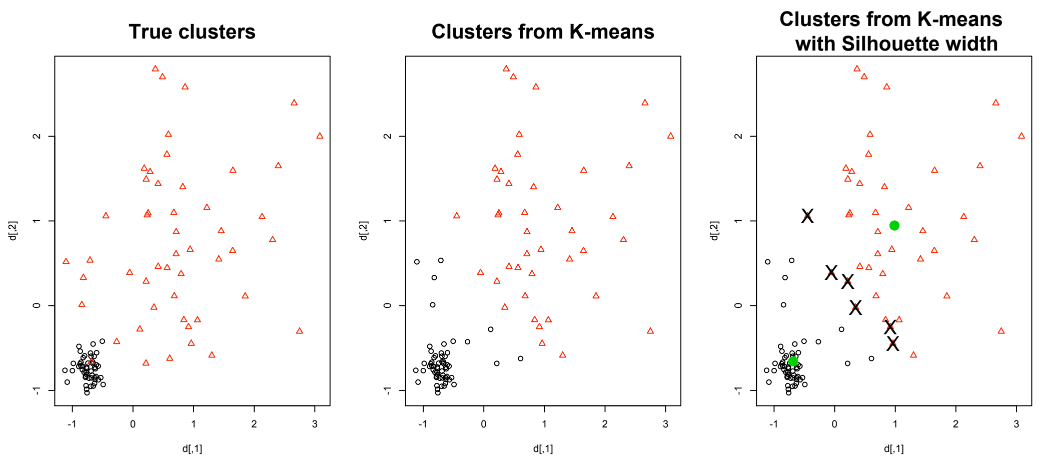


Figure A.1: The example of negative Silhouette.

Figure A.1. The first panel shows the generated cases from two clusters. The cases are colored by true cluster labels. Clearly one cluster has much larger variation than another. The second panel and the third panel show the same cases colored by the partition from 2-means. In the third panel, the cases with negative silhouette widths are indicated by ‘‘X’’. The circles indicate the cluster centers from 2-means. Although the cases are clustered in the clusters with the closest cluster centers, Silhouette suggests that they should rather be placed in the neighbor.

We think that it is odd to favor the cluster with less variation to measure how well cases are clustered. A direct comparison of the distance between a case to its cluster center and the distance between the case to its second closest cluster might be better to measure how well cases are clustered. For this reason, we use *rev.silhouette* for the data analyses in Chapter 5.