2023-05-01

Theoretical And Computational Aspects Of Robust Cluster Analysis For Multivariate And High-Dimensional Datasets

Andrews Tawiah Anum
University of Texas at El Paso

THEORETICAL AND COMPUTATIONAL ASPECTS OF ROBUST CLUSTER ANALYSIS FOR MULTIVARIATE AND HIGH-DIMENSIONAL DATASETS

ANDREWS TAWIAH ANUM

Doctoral Program in Computational Science

APPROVED:

Michael Pokojovy, Ph.D., Chair

Abhijit Mandal, Ph.D.

Md Fashiar Rahman, Ph.D.

Xiaogang Su, Ph.D.

Tzu-Liang (Bill) Tseng, Ph.D.

Stephen L. Crites Jr, Ph.D.
Dean of the Graduate School

Dedication

To my

FATHER Andrews, MOTHER Eva, SISTERS Peninnah and Priscilla
and my ADVISOR Dr. Pokojovy

with love

THEORETICAL AND COMPUTATIONAL ASPECTS OF ROBUST CLUSTER ANALYSIS FOR MULTIVARIATE AND HIGH-DIMENSIONAL DATASETS

by

ANDREWS TAWIAH ANUM, MS

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Computational Science Program

THE UNIVERSITY OF TEXAS AT EL PASO

May 2023

Acknowledgements

I would like to thank my supervisor, Dr. Michael Pokojovy, and express my deepest gratitude for his unwavering support, guidance and encouragement throughout my doctoral journey. Your experiences and valuable insights have made a significant contribution to shaping the direction and quality of my research. Your guidance and mentorship have been invaluable in helping me grow as a researcher and as a human being. I will always be grateful for the knowledge and skills I gained through working with you, and for showing me grace on countless occasions. Every time I spoke with you, I felt how much you wanted me to be successful in my research, and in life in general. You encouraged me to be proactive by doing much more than the bare minimum. You are more than just a Thesis and Dissertation supervisor to me. In addition to research, you advised me on all aspects of life. In you I found a mentor, a supervisor, a collaborator and a life coach on top of it all. My success story in graduate school can not be shared or told without mentioning Dr. Michael Pokojovy.

My sincere thanks also go to members of my committee, Dr. Abhijit Mandal, Dr. Bill Tseng, Dr. Xiaogang Su and Dr. Md Fashiar Rahman for their valuable comments, constructive criticism and invaluable contributions to my dissertation. Your support and guidance helped me to complete this work. Your experience in respective fields and willingness to provide valuable information have been invaluable to my research.

I would also like to thank Dr. Michael Pokojovy, Dr. Su Chen, Dr. Bill Tseng and Dr. Xiaogang Su for helping me in my job search by writing strong recommendations letters in support of my academic applications. My job search would not have been successful without your massive support.

I would also like to thank the Director of the Computational Science Program, Dr. Ming-Ying Leung, for her advice on course selection and other academic affairs since I joined the Program. I would also like to thank all the teaching and administrative staff of the Department of Mathematical Sciences, especially Dr. María Barraza-Rios, for their support and

assistance throughout my research. Your willingness to make computing resources available to me is greatly appreciated.

I would like to thank my colleagues and fellow students for their support and encouragement throughout my time in the program. They gave me a sense of community and helped me stay motivated during tough times. I am also grateful to the Computational Science Students Association for making me President and Vice President for the 2022-2023 and 2021-2022 school years respectively. By serving in these positions, I gained important leadership skills that improved my interpersonal relationship skills. I also became more strategic in problem solving by serving in these positions.

I am incredibly grateful for the support of my family. Your unwavering encouragement and trust in me have been essential to my success. I would not have done it without your love and support. I am very grateful to you for always supporting me. My love for you is indescribable.

Finally, I would like to thank Prof. Francis Benyah from the University of Cape Coast (Ghana) and everyone who supported me on my doctoral path. Your support, guidance and encouragement have been invaluable and I am deeply grateful for your indirect contributions to my research. I now look forward to the next chapter of my life and look forward to continuing to grow and learn as a researcher and as an individual.

Abstract

Multivariate and high-dimensional datasets typically contain subgroups that may not be immediately apparent. To reveal these groups, cluster analysis is performed. Cluster analysis is an unsupervised machine learning technique commonly employed to partition a dataset into distinct categories referred to as clusters. The k-means algorithm is a prominent distance-based clustering method. Despite overwhelming popularity, the algorithm is not invariant under non-singular affine transformations and is not robust, i.e., can be unduly influenced by outliers. To address these deficiencies, we propose an alternative model-based clustering procedure by minimizing a "trimmed" variant of the negative log-likelihood function. We develop a "concentration step", vaguely reminiscent of the classical Lloyd's algorithm, that can iteratively reduce the objective function converging to local minimum in a finite number of steps. Being a local optimization technique, our algorithm depends on the choice of "warm-start." We develop a new sampling procedure to select appropriate warmstarts. For high-dimensional or sparse datasets, cluster covariances become ill-conditioned. Consequently, we equipped our proposed method with high-dimensional capabilities by using a regularization technique that replaces ill-conditioned covariances with well-conditioned counterparts. For $n > p$, a formal proof reveals that the objective function possesses the affine-invariant property under non-singular affine transformations rendering the procedure affine invariant. Extensive simulations for synthetic and real-world datasets are conducted to assess the performance of our algorithm with respect to multiple cluster quality metrics. Compared to such state-of-the-art competitors as k-means (or trimmed k-means) and `tclust`, empirical studies indicate competitiveness and oftentimes superiority of our algorithm.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Grouping similar objects is a task that we commonly encounter in our daily lives. Objects within a particular group are more alike to each other than they are to those in other groups. This concept forms the basis of many clustering algorithms and techniques that have been developed over the years. Clustering is a technique in unsupervised learning that aims to group similar data points together. Clustering can be abstractly defined as the process of identifying groups within a data set using a dissimilarity criterion (Khanmohammadi et al., 2017) or plainly defined as redistributing observational units in a data set into groups called clusters based on some similarity measure. The goal of clustering is to partition a given data set into meaningful subgroups (Ankerst et al., 2008), where each subgroup, or cluster, consists of similar data points. Clustering is performed to identify patterns in a dataset that may not be immediately apparent.

Clustering has a wide range of applications in various fields such as, image and speech recognition (Jain, 2010), market segmentation (Kaufman and Rousseeuw, 1990), anomaly detection (Chandola et al., 2009), medicine and healthcare (Kalyani, 2012; Xu and Wunsch, 2010; Andreopoulos et al., 2009), wireless sensor networks (Abbasi and Younis, 2007), bioinformatics (Masood and Khan, 2015), computer science, business and marketing, engineering, cybersecurity and many others. It is a powerful tool for exploring and understanding complex data sets and can provide insights that are not immediately apparent through other techniques. In healthcare, clustering is used in several ways to improve patient outcomes and inform medical research. One way clustering is used in healthcare is to identify patient subgroups with similar characteristics or outcomes. For example, clustering is used to group patients with similar symptoms, medical history, or treatment responses, which can

inform personalized treatment plans and improve patient outcomes. Clustering is also used to support medical research by identifying patient subgroups that may be underrepresented in clinical trials, and these subgroups are targeted for further research. Overall, clustering can help healthcare professionals improve patient outcomes by providing more personalized care, and it can support medical research by identifying new insights and opportunities for further study. In business and marketing, clustering have been used to inform targeted marketing campaigns, product development, customer recommendations and other business decisions. For example, a company may use clustering to group customers based on demographics, purchase history, or other relevant data, and then develop marketing strategies or products tailored to each group. Clustering can also be used to identify patterns or trends in customer behavior, which can help companies make predictions about future market conditions or identify new opportunities. Overall, clustering can help companies make more informed business decisions and improve their marketing efforts by better understanding their customers and target markets.

Clustering algorithms that exist can be grouped into four categories: partitional clustering, density-based clustering, hierarchical clustering and model-based clustering. Density-based clustering methods group densely populated data points from sparsely located data points. A density-based cluster is a set of data objects spread in the data space over a contiguous region of high density of objects, separated from other density-based clusters by contiguous regions of low density of objects (Kriegel et al., 2011). This method is particularly useful for identifying clusters of arbitrary shape in a dataset. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester et al., 1996) is a widely known density-based clustering technique, which uses two parameters: a distance threshold (`eps`) and a minimum number of points (`minPts`) to define clusters. DBSCAN starts by identifying a dense area of points, which is a region that contains at least `minPts` within a distance of `eps`. These points are considered as core points. Then the algorithm expands the cluster by adding all points that are within a distance of `eps` to the core points. Points that are not reachable from any core point are considered as noise or outliers. However, it can be sensitive

to the choice of parameters (`eps`, `minPts`) and it may not work well with high-dimensional data.

Hierarchical clustering methods involve building a binary merge tree and cluster observational units in a sample data into a tree-like structure called a dendrogram. The procedure begins from data elements stored at the leafs which proceeds by merging the two closest subsets until we reach the root of the tree containing all the elements of the dataset (Nielsen, 2016). Hierarchical clustering can be done in two major ways, i.e., agglomerative hierarchical clustering and the divisive hierarchical clustering. For agglomerative clustering, each observational unit in the data is assumed to be a cluster and, based on a similarity measure, pairs of points are merged into a cluster. The divisive hierarchical clustering, on the other hand, first considers all the data points as one cluster, and then considers all possible ways to divide the cluster into two clusters. The best division is selected and the splitting is done recursively until all clusters are formed. Single linkage, average linkage, and complete linkage are examples of distance measures used in hierarchical clustering. Single linkage creates clusters by connecting data points that are closest to each other, while complete linkage creates clusters by connecting data points that are most distant from each other. Average linkage creates clusters by connecting data points that have the smallest average distance between them (Landau et al., 2011). One of the main benefits of hierarchical clustering is that it can handle non-spherical clusters. However, it can be computationally expensive for large datasets and the choice of linkage method, which determines how clusters are merged or split, can greatly affect the final clustering results. Hierarchical clustering has also been used as initial clusters or partition for some clustering methods such as `Mclust` (Scrucca et al., 2016; Banfield and Raftery, 1993; Fraley and Raftery, 1998).

Partitional clustering involves splitting of the data set into non-overlapping groups by hyperplanes such that an objective function is minimized. The k-means and k-medoids algorithms are examples of partitioning algorithms. The k-medoids algorithm minimizes the sum of distances between data points and the medoid of their respective clusters (Kaufman and Rousseeuw, 1990). Partitional clustering is widely used in various fields such as image

processing, data mining, and market segmentation. The main advantage of partitional clustering is that it is computationally efficient for large datasets, and it can be used to identify clusters with distinct characteristics. However, it may not be suitable for datasets for which the number of clusters is not known in advance, and datasets with outliers.

The k-means algorithm aims to determine a set of centers, say k, in general, which divides the data into k groups so that the sum of squared distances from each point in the group to the center of the group is minimized (Kanungo et al., 2002). The algorithm starts by choosing k initial cluster centers at random and redistribute each observational unit to the cluster with closest center, and then updates the cluster centers based on the new assignments. This process is repeated until the assignments and cluster centers no longer change. One major problem of the k-means algorithm together with most traditional clustering techniques is its sensitivity of the clusters to units of measurement (Kumar and Orlin (2008)). Different units of measurement for same variables can cause the cluster structure to completely change. For this reason, k-means fails to obtain the right clusters if an affine transformation is applied to the data set. When a linear transformation is applied to the data, the data points are linearly arranged in the space and the k-means algorithm may produce unsatisfactory clusters. One approach that has been used to tackle this problem is by converting (normalizing) all variables (with units) into unitless variables (with the range [0, 1]) before applying a clustering technique. Knorr et al. (2001) argued that this way of solving the problem at hand is not a satisfactory solution because just one extreme value can cause other values to be contained in a small sub-range which can lead to "improvised" clusters. Outliers pose several problems to many measures that are estimated. The sample mean is one example of an estimator that is heavily affected by outliers. The sample mean a has breakdown point of zero implying that just one extreme positive or negative value is enough to "ruin" its estimating power. The k-means clustering method computes its cluster centroids using the sample mean which is non-robust thereby rendering the method susceptible to outliers. Thus, applying k-means to cluster a data set with outliers can lead to undesirable outcomes.

To address these challenges, several robust clustering methods and affine-invariant clustering methods are proposed in the literature like those by García-Escudero et al. (2008, 2010); Kumar and Orlin (2008), and more recently by Huang and Yang (2020) and references therein. We consider a clustering procedure which involves minimizing the "trimmed" negative log-likelihood of a Gaussian mixture model contaminated by up to $100 \cdot \alpha\%$ outliers. The resulting log-likelihood function has the affine-invariance property. Thus, the objective function, i.e., the log-likelihood function, is changed by a constant under any general non-singular linear transformations. A detailed proof of the affine-invariant property is given in Chapter 2. For given sample data $\boldsymbol{X} = (\boldsymbol{x}_i | i = 1, \ldots, n)$, where each $\boldsymbol{x}_i \in \mathbb{R}^p$, we search for a weight matrix $\boldsymbol{w}^*_{n \times K}$ where each $w_{ik} \in \{0, 1\}$, with $\sum_{k=1}^{K} w_{ik} = 1$ for each i, such that $\sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \geq n(1 - \alpha)$ for $0 \leq \alpha < 1$ where $\sum_{i=1}^{n} w_{ik} = n_k^{(\boldsymbol{w})}$ for each k, which minimizes a type of negative log-likelihood function. Cluster memberships for each $\boldsymbol{x}_i$ are then obtained from the weight matrix $\boldsymbol{w}^*$. The parameter α is known as a nominal "breakdown" point parameter that controls the percentage of points that remain unassigned. The procedure reduces to the non-robust case when $\alpha = 0$. In this case all data points will be assigned to appropriate clusters. When $\alpha \neq 0$, but some constant within its defined range, the robust procedure is used and $n \cdot \alpha\%$ data points are left unassigned.

We develop a concentration step, vaguely reminiscent of the classical Lloyd's algorithm, that can iteratively be used to minimize the trimmed log-likelihood objective. Following the ideas of Pokojovy and Jobe (2022), we unveil the equivalence between our proposed method and the well-known Frank-Wolfe gradient descent method, which, in turn, implies our algorithm converges at a local minimum of the objective function. The Frank-Wolfe gradient descent method requires solving a linear minimization problem over a convex domain. We discuss solution to this linear programming problem in Chapter 3. Being a local optimization technique, our algorithm depends on the choice of initial cluster partition or "warmstart." As a by-product, we develop a new affine-invariant sampling procedure to select appropriate warmstarts. This algorithm is also presented in Chapter 3. We extended our proposed method to situations where cluster covariance may be assumed equal with small cluster

sizes, and also to high-dimensional setting ($p > n$).

Multiple real-world and synthetic datasets with different configurations, including various cluster shapes and sizes, number of clusters and types of linear transformations, are analyzed to assess the performance of our proposed algorithm and compare it to k-means and `tclust`. Empirical results strongly indicate that our proposed cluster algorithm has the potential to serve as a robust affine-invariant, but still computationally attractive, alternative to the conventional k-means method.

Chapter 2

Model-Based Clustering

In this Chapter, we discuss the idea of model-based clustering and propose a way of clustering observations in a dataset into groups by minimizing a negative trimmed log-likelihood function of a Gaussian mixture model.

2.1 Problem Formulation

Model-based clustering is a type of clustering algorithm that uses a statistical model to identify the clusters in the data. The model is used to generate the cluster structure and to estimate the parameters of the clusters. Model-based clustering can give a probabilistic interpretation of the clustering results. This means that the model can assign a probability to each data point, indicating how likely it is to belong to a particular cluster. This allows for a more natural interpretation of the results, as well as providing a way to quantify the uncertainty in the clustering results. There are several popular models for model-based clustering. We will focus on Gaussian Mixture Models. The Gaussian Mixture Models is a convex combination of Gaussian densities and also a probabilistic model that assumes the observational units in the sample data set form K groups, where each group originates from a Gaussian distribution with individual mean and covariance. Given a sample dataset $\boldsymbol{X}$ of n observations $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ from a p-variate distribution modeled as a mixture of K Gaussian distributions, we want to determine for each observational unit $\boldsymbol{x}_i$ in the sample data set a K-dimensional binary random variable $\boldsymbol{y}$ which will represent the i-th cluster label. In the 1-of-K representation of the random variable $\boldsymbol{y}$, one particular element, y_k, is assigned a value of 1, while all others are assigned a value of 0. If the i-th observational unit

$\boldsymbol{x}_i$ belongs to the k-th cluster, $\boldsymbol{x}_i$ is assigned label $y_{ik} = 1$. The observational units in the sample data set are redistributed into K homogeneous subgroups once all $\boldsymbol{x}_i$'s are labeled.

Let $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ be a sample of n observations from a p-variate distribution modeled as a mixture of K Gaussian distributions given as a convex combination of Gaussian densities below

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \varphi_k\big(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big), \tag{2.1}$$

where $\pi_k \in [0, 1]$ is a mixing coefficient or mixture proportion such that the sum of all π_k's equal to 1, $\varphi_k\big(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big)$ represent the k-th Gaussian density with mean vector $\boldsymbol{\mu}_k$ and covariance matrices $\boldsymbol{\Sigma}_k$. The mean vector and covariance matrix determine the shape of the Gaussian distribution, while the mixture coefficient determines the proportion of the data that is assigned to each Gaussian distribution or cluster. The log-likelihood of the Gaussian mixture for a set of observational units $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ is given as

$$\log p(\boldsymbol{X}|\boldsymbol{\theta}) = \sum_{n=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \varphi\big(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big) \right) \tag{2.2}$$

with parameter $\boldsymbol{\theta} = \big((\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{k=1,\ldots,K}\big)$. For model-based clustering, we are concerned with redistributing or partitioning the observational units into clusters or estimating the parameter $\boldsymbol{\theta}$ such that the log-likelihood function in Equation (2.2) is maximized. One parameter estimation method that quickly comes to mind to maximize the log-likelihood function is the maximum likelihood framework. Maximum likelihood estimation can become problematic when applied to Gaussian Mixture Models when the data contains singularities. Since there are several components of the mixture, the presence of singularities can cause the log-likelihood to diverge to infinity. This is as a result of the tendency of some components shrinking onto specific data points and thereby increasing or decreasing the log-likelihood value. This phenomenon leads to severe over-fitting or poor generalization performance. Also, the maximum likelihood framework for Equation (2.2) has no closed-form analytical solution due to the summation over k in the logarithm. As a result, iterative methods are preferred in this setting over the maximum likelihood approach. Also, a popular method

that can be used to maximize the log-likelihood function in Equation (2.2) is known as the expectation–maximization (EM) framework of Dempster et al. (1977). See McLachlan and Krishnan (1997) and references therein. A robust regularized approach was proposed by García-Escudero et al. (2008, 2010). Alternative approaches include Kumar and Orlin (2008), Huang and Yang (2020), etc.

Suppose that we have in addition to the sample dataset $\boldsymbol{X}$, a K-dimensional binary random latent variable $\boldsymbol{y}$ in which an i-th observational unit originating from the k-th Gaussian is labeled $y_{ik} = 1$ and all others are assigned 0. Let $\boldsymbol{Y}$ then represent all of these 1-of-K representations. Then, $\boldsymbol{Y}$ has dimension $n \times K$. Consider the problem of maximizing the likelihood for the complete dataset $\{\boldsymbol{X}, \boldsymbol{Y}\}$. The likelihood function has the form

$$p(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{\theta}) = \prod_{n=1}^{n} \prod_{k=1}^{K} \pi_k^{y_{nk}} \varphi\big(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big)^{y_{nk}},$$

where y_{nk} is the k-th component of $\boldsymbol{y}_n$. The log-likelihood is then given as

$$\begin{aligned} \log p(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{\theta}) &= \sum_{n=1}^{n} \sum_{k=1}^{K} y_{nk} \log \left(\pi_k \varphi\big(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big)\right) \\ &= \sum_{n=1}^{n} \sum_{k=1}^{K} \log \left(\pi_k \varphi\big(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\big)\right). \end{aligned} \tag{2.3}$$

The log-likelihood in Equation (2.3) now has the summation over k on the logarithm of the Gaussian components. This form of the log-likelihood is easier to maximize.

Our aim is to develop a clustering method that is robust and invariant under non-singular affine transformation that puts p-variate data into K clusters with the assumption that each observation in the sample may originate from any one of the K possible multivariate normal distributions with means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ and covariance matrices $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k$. When all observations in the dataset are assigned to a cluster, we represent all the cluster labels $\boldsymbol{y}_i$, where $i = 1, \ldots n$, with the matrix $\boldsymbol{w}$ of dimension $n \times p$. We will call $\boldsymbol{w}$ the weight matrix of the sample dataset. We formulate the clustering problem as follows. Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ be a sample of n observations from a p-variate distribution modeled as a mixture of K Gaussian distributions with a weight matrix $\boldsymbol{w}$. Consider the negative log-likelihood function

given as:

$$H(\boldsymbol{w}) = -\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\log\Big(\pi_k^{(\boldsymbol{w})} f(\boldsymbol{x}_i|\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_k^{(\boldsymbol{w})})\Big). \tag{2.4}$$

Notice that the objective function now has minus in front of the summation implying that we will be minimizing the negative log-likelihood function rather than maximizing the log-likelihood. The objective function given in Equation (2.4) is a "trimmed" negative log-likelihood function as given in (García-Escudero et al., 2010, 2008). In Equation (2.4), k represents the index of the k-th cluster and i the index of the i-th observation, $|\mathcal{C}| = |\cup_{k=1}^{K}\mathcal{C}_k| = \lceil n(1-\alpha)\rceil$ where $\mathcal{C}_k$ is the k-th cluster, $\boldsymbol{w}\in\mathbb{R}^{n\times K}$ is a weight matrix. The k-th mixing proportion is represented by $\pi_k^{(\boldsymbol{w})}$ and the sum of all these mixing proportions equal to 1, i.e., $\sum_{k=1}^{K}\pi_k^{(\boldsymbol{w})} = 1$. Also,

$$f(\boldsymbol{x}_i|\boldsymbol{\mu},\boldsymbol{\Sigma}) = (2\pi)^{-p/2}\,|\boldsymbol{\Sigma}|^{-1/2}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

is the probability density function (pdf) of the multivariate Gaussian distribution. Further, $\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}$ is the mean of the k-th cluster defined as:

$$\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})} = \frac{1}{n_k^{(\boldsymbol{w})}}\sum_{i=1}^{n} w_{ik}\boldsymbol{x}_i, \tag{2.5}$$

and $\boldsymbol{S}_k^{(\boldsymbol{w})}$ is the covariance matrix associated with the k-th cluster defined as:

$$\begin{aligned}\boldsymbol{S}_k^{(\boldsymbol{w})} &= \frac{1}{n_k^{(\boldsymbol{w})}-1}\sum_{i=1}^{n} w_{ik}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})(\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'\\ &= \frac{1}{n_k^{(\boldsymbol{w})}-1}\sum_{i=1}^{n} w_{ik}\boldsymbol{x}_i\boldsymbol{x}_i' - \frac{n_k^{(\boldsymbol{w})}}{n_k^{(\boldsymbol{w})}-1}\bar{\boldsymbol{x}}_k\bar{\boldsymbol{x}}_k'.\end{aligned} \tag{2.6}$$

Beginning with hard clustering, we search for an optimal weight matrix $\boldsymbol{w}^*\in\{0,1\}^{n\times K}$ with

$$0\le\sum_{k=1}^{K} w_{ik}\le 1 \text{ for all } i,\ \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}\ge n(1-\alpha) \text{ and } \sum_{i=1}^{n} w_{ik}\ge n_k. \tag{2.7}$$

An optimal weight matrix is the weight matrix for the dataset that minimizes the negative log-likelihood function. For hard clustering, an observational unit is assigned probabilities,

strictly 0 or 1, of originating from any of the K clusters. The most probable cluster for the observational unit is assigned a probability of 1 and therefore, the weights $\boldsymbol{w}_{ik}$ are confined to 0 and 1. Relaxing the hard clustering condition allows assigning probabilities of originating from one of the K clusters, other than only 0 and 1, to an observational unit. Hence, the weights $\boldsymbol{w}_{ik} \in [0, 1]$. From hard clustering, we move to soft clustering by searching for an optimal weight matrix $\boldsymbol{w}^* \in [0, 1]^{n \times K}$ where $\sum_{k=1}^{K} w_{ik} = 1$ and $\sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik} \geq n(1 - \alpha)$ which minimizes the the objective function $H(\boldsymbol{w})$ such that:

$$\mathcal{C}_k \cap \mathcal{C}_l = \emptyset, \quad |\mathcal{C}_k| \geq p + 1, \quad \sum_{k=1}^{K} |\mathcal{C}_k| \geq n(1 - \alpha). \tag{2.8}$$

The first condition in Equation (2.8) ensures the clusters are pairwise disjoint. The second condition makes certain that there are at least $p+1$ points in each cluster guaranteeing that all determinants are non-zero provided the dataset is in general position. The last condition mandates that no more than αn points remain unassigned, allowing for up to $100 \cdot \alpha\%$ outliers. The parameter $\alpha \in [0, 1)$ is known as a nominal "breakdown" point parameter which controls what percentage of points that will not be assigned clusters. All data points are assigned and the procedure is non-robust, when $\alpha = 0$. When $\alpha \neq 0$, the procedure is expected to be robust since αn observational units end up not assigned.

A new procedure for obtaining an optimal weight matrix $\boldsymbol{w}^*$ is proposed and presented in Algorithm 4. The new procedure is an iterative procedure we refer to as k-dets. The iterative step of k-dets is based on the idea of concentration step or C-step. Based on recent discovery of Pokojovy and Jobe (2022), our C-step uses Frank-Wolfe gradient method with largest step size ($\gamma = 1$) to iteratively reduce the objective function. Since our proposed method is an iterative procedure, it depends on initial data partition or "warmstart" (initial clusters). Thus, our proposed method begins by randomly selecting several initial partitions. For each of these initial partitions, the method iterates until a minima is reached leading to a candidate partition. The method the chooses the partition with the minimum objective value over all partitions. "Surprisingly", the algorithm always converges to a hard cluster membership matrix $\boldsymbol{w} \in \{0, 1\}^{n \times K}$. In addition to k-dets, we developed an affine-invariant

initial sampling method that produces an initial cluster partition for a given sample dataset. The procedure is given in Algorithm 3.

2.1.1 Affine-Invariance Property

One of the major strengths of our clustering algorithm is the affine-invariant property which comes from the objective function $H(\boldsymbol{w})$. The affine-invariant property ensures that when we find an optimal weight matrix for the sample data, then the same weight matrix is an optimal weight matrix for a general non-linear transformation of the sample dataset and vice versa. This is so because the objective function is changed only by a constant. A proof of this property is given below.

Theorem 1. *For any non-singular* $\boldsymbol{A} \in \mathbb{R}^{p\times p}$ *and* $\boldsymbol{b} \in \mathbb{R}^p$*, define* $\boldsymbol{y}_i = \boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{b}$*, where* $\boldsymbol{x}_i \in \mathbb{R}^p$*, if* $\boldsymbol{w}$ *is a minimizer of the objective function* $H(\boldsymbol{w}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ *then* $\boldsymbol{w}$ *is a minimizer of* $H(\boldsymbol{w}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$*.*

Proof. Let $\bar{\boldsymbol{y}}_k^{(\boldsymbol{w})}$ be the mean vector of the k-th cluster defined as:

$$\bar{\boldsymbol{y}}_k^{(\boldsymbol{w})} = \frac{1}{n_k^{(\boldsymbol{w})}} \sum_{i=1}^{n} w_{ik}\boldsymbol{y}_i. \tag{2.9}$$

Substituting $\boldsymbol{y}_i$ into Equation (2.9) and expanding we have

$$\begin{aligned}
\bar{\boldsymbol{y}}_k^{(\boldsymbol{w})} &= \frac{1}{n_k^{(\boldsymbol{w})}} \sum_{i=1}^{n} w_{ik}(\boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{b}) \\
&= \frac{1}{n_k^{(\boldsymbol{w})}} \sum_{i=1}^{n} (w_{ik}\boldsymbol{A}\boldsymbol{x}_i + w_{ik}\boldsymbol{b}) \\
&= \frac{1}{n_k^{(\boldsymbol{w})}} \sum_{i=1}^{n} w_{ik}\boldsymbol{A}\boldsymbol{x}_i + \frac{1}{n_k^{(\boldsymbol{w})}} \sum_{i=1}^{n} w_{ik}\boldsymbol{b} \\
&= \boldsymbol{A}\frac{1}{n_k^{(\boldsymbol{w})}} \sum_{i=1}^{n} w_{ik}\boldsymbol{x}_i + \boldsymbol{b}\frac{1}{n_k^{(\boldsymbol{w})}} \sum_{i=1}^{n} w_{ik} \\
&= \boldsymbol{A}\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})} + \boldsymbol{b}\frac{1}{n_k^{(\boldsymbol{w})}} n_k^{(\boldsymbol{w})}
\end{aligned}$$

$$= \boldsymbol{A}\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})} + \boldsymbol{b}.$$

The mean vector of the k-th cluster of the transformed data is defined in terms of the mean vector of the k-th cluster of the sample data. Let $\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})}$ and $\boldsymbol{S}_{\boldsymbol{y},k}^{(\boldsymbol{w})}$ be the covariance matrices of the k-th cluster of the sample data $\boldsymbol{X}$ and the transformed data $\boldsymbol{Y}$ respectively. Define $\boldsymbol{S}_{y,k}^{(\boldsymbol{w})}$ as:

$$\boldsymbol{S}_{\boldsymbol{y},k}^{(\boldsymbol{w})} = \frac{1}{n_k^{(\boldsymbol{w})} - 1} \sum_{i=1}^{n} \boldsymbol{w}_{ik}(\boldsymbol{y}_i - \bar{\boldsymbol{y}}_k^{(\boldsymbol{w})})(\boldsymbol{y}_i - \bar{\boldsymbol{y}}_k^{(\boldsymbol{w})})'. \tag{2.10}$$

Substituting $\boldsymbol{y}_i$ and $\bar{\boldsymbol{y}}_k^{(\boldsymbol{w})}$ into Equation (2.10) and expanding, we have

$$\begin{aligned}
\boldsymbol{S}_{\boldsymbol{y},k}^{(\boldsymbol{w})} &= \frac{1}{n_k^{(\boldsymbol{w})} - 1} \sum_{i=1}^{n} w_{ik}((\boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{b}) - (\boldsymbol{A}\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})} + \boldsymbol{b}))\big((\boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{b}) - (\boldsymbol{A}\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})} + \boldsymbol{b})\big)' \\
&= \frac{1}{n_k^{(\boldsymbol{w})} - 1} \sum_{i=1}^{n} w_{ik}(\boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})} - \boldsymbol{b})\big(\boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})} - \boldsymbol{b}\big)' \\
&= \frac{1}{n_k^{(\boldsymbol{w})} - 1} \sum_{i=1}^{n} w_{ik}(\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})\big(\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}\big)' \\
&= \frac{1}{n_k^{(\boldsymbol{w})} - 1} \sum_{i=1}^{n} w_{ik}\boldsymbol{A}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})\big(\boldsymbol{A}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})\big)' \\
&= \frac{1}{n_k^{(\boldsymbol{w})} - 1} \sum_{i=1}^{n} w_{ik}\boldsymbol{A}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'\boldsymbol{A}' \\
&= \frac{1}{n_k^{(\boldsymbol{w})} - 1} \boldsymbol{A} \sum_{i=1}^{n} w_{ik}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'\boldsymbol{A}' \\
&= \boldsymbol{A}\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})}\boldsymbol{A}'.
\end{aligned}$$

The covariance of the k-th cluster of the transformed data is now defined in terms of the covariance of the k-th cluster of the sample data set. Consider the probability density function of the multivariate Gaussian distribution defined below:

$$f(\boldsymbol{y}_i|\bar{\boldsymbol{y}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{\boldsymbol{y},k}^{(\boldsymbol{w})}) = (2\pi)^{-p/2} \left|\boldsymbol{S}_{\boldsymbol{y},k}^{(\boldsymbol{w})}\right|^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{y}_i - \bar{\boldsymbol{y}}_k^{(\boldsymbol{w})})'(\boldsymbol{S}_{\boldsymbol{y},k}^{(\boldsymbol{w})})^{-1}(\boldsymbol{y}_i - \bar{\boldsymbol{y}}_k^{(\boldsymbol{w})})\right). \tag{2.11}$$

Substituting $\boldsymbol{y}_i$, $\bar{\boldsymbol{y}}_k^{(\boldsymbol{w})}$ and $\boldsymbol{S}_{\boldsymbol{y},k}^{(\boldsymbol{w})}$ into Equation (2.11) and expanding gives

$$f(\boldsymbol{y}_i|\bar{\boldsymbol{y}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{\boldsymbol{y},k}^{(\boldsymbol{w})}) = (2\pi)^{-p/2} \left|\boldsymbol{A}\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})}\boldsymbol{A}'\right|^{-1/2} \times$$

$$
\begin{aligned}
&\times \exp\left(-\frac{1}{2}(\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'(\boldsymbol{A}\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})}\boldsymbol{A}')^{-1}(\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})\right) \\
&= (2\pi)^{-p/2}\left|\boldsymbol{A}\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})}\boldsymbol{A}'\right|^{-1/2} \times \\
&\times \exp\left(-\frac{1}{2}((\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'\boldsymbol{A}')(\boldsymbol{A}')^{-1}(\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})})^{-1}\boldsymbol{A}^{-1}(\boldsymbol{A}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}))\right) \\
&= (2\pi)^{-p/2}\left|\boldsymbol{A}\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})}\boldsymbol{A}'\right|^{-1/2}\exp\left(-\frac{1}{2}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'(\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})})^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})\right) \\
&= (2\pi)^{-p/2}\left|\boldsymbol{A}\right|^{-1/2}\left|\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})}\right|^{-1/2}\left|\boldsymbol{A}'\right|^{-1/2} \times \\
&\times \exp\left(-\frac{1}{2}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'(\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})})^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})\right) \\
&= (2\pi)^{-p/2}\left|\boldsymbol{A}\right|^{-1}\left|\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})}\right|^{-1/2}\exp\left(-\frac{1}{2}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'(\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})})^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})\right) \\
&= \left|\boldsymbol{A}\right|^{-1} f(\boldsymbol{x}_i|\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})}).
\end{aligned}
$$

The Gaussian density of the transformed data is now defined in terms of the Gaussian density of the sample data. The log-likelihood or the objective function $H(\boldsymbol{w}|\boldsymbol{y})$ is given as

$$
H(\boldsymbol{w}|\boldsymbol{y}) = -\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\log\Big(\pi_k^{(\boldsymbol{w})} f(\boldsymbol{y}_i|\bar{\boldsymbol{y}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{y,k}^{(\boldsymbol{w})})\Big). \tag{2.12}
$$

Substituting $f(\boldsymbol{y}_i|\bar{\boldsymbol{y}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{\boldsymbol{y},k}^{(\boldsymbol{w})})$ into Equation (2.12) and expanding yields

$$
\begin{aligned}
H(\boldsymbol{w}|\boldsymbol{y}) &= -\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\log\Big(\pi_k^{(\boldsymbol{w})}|\boldsymbol{A}|^{-1} f(\boldsymbol{x}_i; \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_k^{(\boldsymbol{w})})\Big) \\
&= -\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\log\big(|\boldsymbol{A}|^{-1}\big) - \sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\log\Big(\pi_k^{(\boldsymbol{w})} f(\boldsymbol{x}_i; \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_k^{(\boldsymbol{w})})\Big) \\
&= K\lceil n(1-\alpha)\rceil\log(|\boldsymbol{A}|) + H(\boldsymbol{w}|\boldsymbol{x}).
\end{aligned}
$$

The objective function $H(\boldsymbol{w}|\boldsymbol{y})$ is defined in terms of the objective function $H(\boldsymbol{w}|\boldsymbol{x})$ and some constant. Thus, a weight matrix $\boldsymbol{w}$ that minimizes the objective function $H(\boldsymbol{w}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ also minimizes the objective function $H(\boldsymbol{w}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$. □

2.2 The Gradient of the Log-Likelihood Function

The C-step is an important step in reducing the objective value at each iteration. The C-step of our proposed algorithm requires solving a linear minimization problem which utilizes the gradient of the objective function. In this section, we first simplify the objective function and then obtain the gradient by applying differentiation techniques. The objective function $H(\boldsymbol{w})$ given in Equation (2.4) can be expressed more explicitly as follows:

$$\begin{aligned} H(\boldsymbol{w}) &= -\sum_{k=1}^{K} \log \prod_{i \in \mathcal{C}_k} \pi_k^{(\boldsymbol{w})} f(\boldsymbol{x}_i; \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_k^{(\boldsymbol{w})}) \\ &= -\sum_{k=1}^{K} \log \pi_k^{|\mathcal{C}_k|} \prod_{i \in \mathcal{C}_k} f(\boldsymbol{x}_i; \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_k^{(\boldsymbol{w})}). \end{aligned}$$

By applying properties of logarithm and taking term by term summation we have:

$$\begin{aligned} H(\boldsymbol{w}) &= -\sum_{k=1}^{K} (|\mathcal{C}_k| \log \pi_k^{(\boldsymbol{w})} + \log \prod_{i \in \mathcal{C}_k} f(\boldsymbol{x}_i; \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_k^{(\boldsymbol{w})})) \\ &= -\Bigg(\sum_{k=1}^{K} |\mathcal{C}_k| \log \pi_k^{(\boldsymbol{w})} + \sum_{k=1}^{K} \log \prod_{i \in \mathcal{C}_k} f(\boldsymbol{x}_i; \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_k^{(\boldsymbol{w})}) \Bigg). \end{aligned} \tag{2.13}$$

An explicit representation of the first term of Equation (2.13) is given below:

$$\begin{aligned} \sum_{k=1}^{K} |\mathcal{C}_k| \log \pi_k^{(\boldsymbol{w})} &= \sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \log \left(\frac{n_k^{(\boldsymbol{w})}}{n} \right) \\ &= \sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \log \Big(n_k^{(\boldsymbol{w})} \Big) - \sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \log(n) \\ &= \sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \log \Big(n_k^{(\boldsymbol{w})} \Big) - \lceil n(1-\alpha) \rceil \log(\lceil n(1-\alpha) \rceil). \end{aligned}$$

The last term of Equation (2.13) is expressed as:

$$\sum_{k=1}^{K} \log \prod_{i \in \mathcal{C}_k} f(\boldsymbol{x}_i | \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_k^{(\boldsymbol{w})}) = \sum_{k=1}^{K} \log \prod_{i \in \mathcal{C}_k} (2\pi)^{-p/2} \left| \boldsymbol{S}_k^{(\boldsymbol{w})} \right|^{-1/2} \times \tag{2.14}$$

$$\times \exp\left(-\frac{1}{2}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'(\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})\right)$$
$$= \sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k} \log\left((2\pi)^{-p/2}\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|^{-1/2}\right) - \frac{1}{2}\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'\times$$
$$\times (\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}).$$

The first component of Equation (2.14) can be expressed more explicitly as:

$$\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k} \log\left((2\pi)^{-p/2}\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|^{-1/2}\right) = \sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k} \log\big((2\pi)^{-p/2}\big) + \sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k} \log\left(\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|^{-1/2}\right)$$
$$= \sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \log\big((2\pi)^{-p/2}\big) + \sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \log\left(\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|^{-1/2}\right)$$
$$= \lceil n(1-\alpha)\rceil \log\big((2\pi)^{-p/2}\big) + \sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \log\left(\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|^{-1/2}\right)$$
$$= \lceil n(1-\alpha)\rceil \log\big((2\pi)^{-p/2}\big) - \frac{1}{2}\sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \log\left(\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|\right).$$

The second component of Equation (2.14) can be expressed further as:

$$-\frac{1}{2}\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(w)})'(\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})$$
$$= -\frac{1}{2}\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\mathrm{tr}[(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'(\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})]$$
$$= -\frac{1}{2}\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\mathrm{tr}[(\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})']$$
$$= -\frac{1}{2}\sum_{k=1}^{K}\mathrm{tr}[(\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}\sum_{i\in\mathcal{C}_k}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})']$$
$$= -\frac{1}{2}\sum_{k=1}^{K}\mathrm{tr}((\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}\boldsymbol{S}_k^{(\boldsymbol{w})}(n_k^{(\boldsymbol{w})} - 1))$$
$$= -\frac{1}{2}\sum_{k=1}^{K}(n_k^{(\boldsymbol{w})} - 1)\mathrm{tr}(\mathbf{I})$$
$$= -\frac{1}{2}\left(\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\mathrm{tr}(\mathbf{I}) - \sum_{k=1}^{K}\mathrm{tr}(\mathbf{I})\right)$$

$$= -\frac{1}{2}(\lceil n(1-\alpha)\rceil p - Kp).$$

Substituting these expressions back into $H(\boldsymbol{w})$ and simplifying, we have

$$\begin{aligned}
H(\boldsymbol{w}) &= -\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\log\Big(\pi_k^{(\boldsymbol{w})} f(\boldsymbol{x}_i;\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})},\boldsymbol{S}_k^{(\boldsymbol{w})})\Big) \\
&= -\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\Big(n_k^{(\boldsymbol{w})}\Big) + \lceil n(1-\alpha)\rceil\log(\lceil n(1-\alpha)\rceil) \\
&\quad - \lceil n(1-\alpha)\rceil\log\big((2\pi)^{-p/2}\big) + \frac{1}{2}\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\Big(\Big|\boldsymbol{S}_k^{(\boldsymbol{w})}\Big|\Big) \\
&\quad + \frac{1}{2}(\lceil n(1-\alpha)\rceil p - Kp) \qquad (2.15)\\
&= -\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\Big(n_k^{(\boldsymbol{w})}\Big) + \frac{1}{2}\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\Big(\Big|\boldsymbol{S}_k^{(\boldsymbol{w})}\Big|\Big) \\
&\quad + \frac{1}{2}(\lceil n(1-\alpha)\rceil p - Kp) + \lceil n(1-\alpha)\rceil\log(\lceil n(1-\alpha)\rceil) \\
&\quad - \lceil n(1-\alpha)\rceil\log\big((2\pi)^{-p/2}\big) \\
&= -(I_f + C(n,p,K)),
\end{aligned}$$

where

$$\begin{aligned}
I_f &= \sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\Big(n_k^{(\boldsymbol{w})}\Big) - \frac{1}{2}\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\Big(\Big|\boldsymbol{S}_k^{(\boldsymbol{w})}\Big|\Big) \\
&= \sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\left(n_k^{(\boldsymbol{w})}\Big|\boldsymbol{S}_k^{(\boldsymbol{w})}\Big|^{-\frac{1}{2}}\right),
\end{aligned}$$

and

$$\begin{aligned}
C(n,p,K) &= -\frac{1}{2}(\lceil n(1-\alpha)\rceil p - Kp) - \lceil n(1-\alpha)\rceil\log(\lceil n(1-\alpha)\rceil) \\
&\quad + \lceil n(1-\alpha)\rceil\log\big((2\pi)^{-p/2}\big).
\end{aligned}$$

The objective function $H(\boldsymbol{w})$ is now expressed in a simpler form for term by term differentiation to obtain the gradient. Differentiating the second term of Equation (2.15) with respect

to w_{ik} yields a value of zero since none of the components depend on w_{ik} as given below:

$$\begin{aligned}\frac{\partial\left(C(n,p,K)\right)}{\partial w_{ik}} &= -\frac{1}{2}\frac{\partial\left(\lceil n(1-\alpha)\rceil p - Kp\right)}{\partial w_{ik}} - \frac{\partial\left(\lceil n(1-\alpha)\rceil \log(\lceil n(1-\alpha)\rceil)\right)}{\partial w_{ik}} + \\ &+ \frac{\partial\left(\lceil n(1-\alpha)\rceil \log\left((2\pi)^{-p/2}\right)\right)}{\partial w_{ik}} \\ &= 0.\end{aligned}$$

The derivative of Equations (2.5) and (2.6) with respect to w_{ik}, the mean vector of the k-th cluster and the covariance of the k-th cluster, can be obtained by applying chain and product rules:

$$\begin{aligned}\frac{\partial\left(\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}\right)}{\partial w_{ik}} &= \frac{1}{n_k^{(\boldsymbol{w})}}\boldsymbol{x}_i - \frac{1}{(n_k^{(\boldsymbol{w})})^2}\sum_{i=1}^{n} w_{ik}\boldsymbol{x}_i \\ \frac{\partial\left(\boldsymbol{S}_k^{(\boldsymbol{w})}\right)}{\partial w_{ik}} &= \frac{1}{n_k^{(\boldsymbol{w})}-1}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})' - \frac{\boldsymbol{S}_k^{(\boldsymbol{w})}}{n_k^{(\boldsymbol{w})}-1}.\end{aligned}$$

The derivative of the first component of Equation (2.15) is given as:

$$\begin{aligned}\frac{\partial\left(I_f\right)}{\partial w_{ik}} &= \frac{\partial\left(\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\left(n_k^{(\boldsymbol{w})}\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|^{-\frac{1}{2}}\right)\right)}{\partial w_{ik}} \\ &= \frac{\partial\left(\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\left(n_k^{(\boldsymbol{w})}\right)\right)}{\partial w_{ik}} - \frac{1}{2}\left(\frac{\partial\left(\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\left(\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|\right)\right)}{\partial w_{ik}}\right) \\ &= \frac{\partial\left(I_2\right)}{\partial w_{ik}} + \frac{\partial\left(I_1\right)}{\partial w_{ik}},\end{aligned}$$

where

$$\begin{aligned}\frac{\partial\left(I_2\right)}{\partial w_{ik}} &= \frac{\partial\left(\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\left(n_k^{(\boldsymbol{w})}\right)\right)}{\partial w_{ik}} \\ &= \frac{\partial\left(n_k^{(\boldsymbol{w})}\right)}{\partial w_{ik}}\log\left(n_k^{(\boldsymbol{w})}\right) + n_k^{(\boldsymbol{w})}\frac{\partial\left(\log\left(n_k^{(\boldsymbol{w})}\right)\right)}{\partial w_{ik}} \\ &= \left(\log\left(n_k^{(\boldsymbol{w})}\right) + n_k^{(\boldsymbol{w})}\frac{1}{n_k^{(\boldsymbol{w})}}\right) \\ &= 1 + \log\left(n_k^{(\boldsymbol{w})}\right),\end{aligned}$$

and

$$\begin{aligned}\frac{\partial\left(I_1\right)}{\partial w_{ik}} &= -\frac{1}{2}\left(\frac{\partial\left(\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\left(\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|\right)\right)}{\partial w_{ik}}\right)\\ &= -\frac{1}{2}\left(\frac{\partial\left(n_k^{(\boldsymbol{w})}\right)}{\partial w_{ik}}\log\left(\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|\right) + n_k^{(\boldsymbol{w})}\frac{\partial\left(\log\left(\left|\boldsymbol{S}_k^{(w)}\right|\right)\right)}{\partial w_{ik}}\right) \qquad (2.16)\\ &= -\frac{1}{2}\left(\log\left(\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|\right) + n_k^{(\boldsymbol{w})}\frac{\partial\log\left(\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|\right)}{\partial w_{ik}}\right).\end{aligned}$$

The second term in Equation (2.16) can be expressed as

$$\begin{aligned}\frac{\partial\log\left(|\boldsymbol{S}_k|\right)}{\partial w_{ik}} &= \mathrm{tr}\left((\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}\frac{\partial\left(\boldsymbol{S}_k^{(\boldsymbol{w})}\right)}{\partial w_{ik}}\right)\\ &= \mathrm{tr}\left((\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}\left(\frac{1}{n_k^{(\boldsymbol{w})}-1}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})(\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})' - \frac{\boldsymbol{S}_k^{(\boldsymbol{w})}}{n_k^{(\boldsymbol{w})}-1}\right)\right)\\ &= \frac{1}{n_k^{(\boldsymbol{w})}-1}\mathrm{tr}((\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})(\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})') - \frac{1}{n_k^{(\boldsymbol{w})}-1}\mathrm{tr}((\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}\boldsymbol{S}_k^{(\boldsymbol{w})})\\ &= \frac{1}{n_k^{(\boldsymbol{w})}-1}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})})'(\boldsymbol{S}_k^{(\boldsymbol{w})})^{-1}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}) - \frac{1}{n_k^{(\boldsymbol{w})}-1}\mathrm{tr}(\boldsymbol{I})\\ &= \frac{1}{n_k^{(\boldsymbol{w})}-1}D^2(\boldsymbol{x}_i;\bar{\boldsymbol{x}}_k^{(w)},\boldsymbol{S}_k^{(\boldsymbol{w})}) - \frac{p}{n_k^{(\boldsymbol{w})}-1},\end{aligned}$$

where $D^2(\boldsymbol{x}_i;\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\Sigma}}) := (\boldsymbol{x}_i-\hat{\boldsymbol{\mu}})'\hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{x}_i-\hat{\boldsymbol{\mu}})$ for $\boldsymbol{x}\in\mathbb{R}^p$, is the squared Mahalanobis distance. Hence, we have

$$\frac{\partial\left(I_1\right)}{\partial w_{ik}} = -\frac{1}{2}\log\left(\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|\right) - \frac{1}{2}\frac{n_k^{(\boldsymbol{w})}}{n_k^{(\boldsymbol{w})}-1}D^2(\boldsymbol{x}_i;\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})},\boldsymbol{S}_k^{(\boldsymbol{w})}) + \frac{1}{2}\frac{n_k^{(\boldsymbol{w})}p}{n_k^{(\boldsymbol{w})}-1}.$$

Thus, the gradient of $H(\boldsymbol{w})$ is given by

$$\begin{aligned}\frac{\partial\left(H(\boldsymbol{w})\right)}{\partial w_{ik}} &= -\frac{\partial\left(I_f\right)}{\partial w_{ik}} + \frac{\partial\left(C(n,p,k)\right)}{\partial w_{ik}}\\ &= -\Big(1+\log\left(n_k^{(\boldsymbol{w})}\right) - \frac{1}{2}\log\left(\left|\boldsymbol{S}_k^{(\boldsymbol{w})}\right|\right) - \frac{1}{2}\frac{n_k^{(\boldsymbol{w})}}{n_k^{(\boldsymbol{w})}-1}D^2(\boldsymbol{x}_i;\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})},\boldsymbol{S}_k^{(\boldsymbol{w})}) + \frac{1}{2}\frac{n_k^{(\boldsymbol{w})}p}{n_k^{(\boldsymbol{w})}-1}\Big).\end{aligned} \qquad (2.17)$$

Chapter 3

Connection with Frank-Wolfe Algorithm

In this Chapter, we discuss the C-step, and empirically demonstrate that by taking the maximum step size at each iteration, the C-step reduces the objective value. We also show how to solve the linear programming problem in our C-step. As a by-product of k-dets, we put forth the algorithm for randomly generating the initial cluster partition. We finally present our proposed algorithm, k-dets.

3.1 Frank-Wolfe Gradient Method

A first-order optimization algorithm for constrained convex optimization, known as the conditional gradient method or the Frank-Wolfe algorithm, was introduced by Frank and Wolfe (1956). Consider the optimization problem given below:

$$f(\boldsymbol{s}) \to \min \text{ over } \boldsymbol{s} \in \mathcal{M} \tag{3.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is twice continuously differentiable over the compact and convex domain $\mathcal{M} \subset \mathbb{R}^d$. Algorithm 1 is the general Frank-Wolfe gradient descent method (Lacoste-Julien, 2016) for solving Equation (3.1). Lacoste-Julien (2016) showed that for Algorithm 1, the minimum Frank-Wolfe gap encountered after t iterations shows an $O(1/\sqrt{t})$ convergence. A detailed proof of the convergence for $\gamma_t \in \arg\min_{\gamma \in [0,1]} f(\boldsymbol{x}^{(t)} + \gamma_t \boldsymbol{d}_t)$ or $\gamma_t := \min\{\frac{g_t}{C}, 1\}$ is given in Theorem 1 of Lacoste-Julien (2016).

Algorithm 1: General Frank-Wolfe gradient method

Select $\boldsymbol{x}^{(0)} \in \mathcal{M}$ and tolerance threshold $\epsilon > 0$.
for $t = 0 \dots T$ **do**
    Compute $\boldsymbol{s}^{(t)} := \arg\min_{\boldsymbol{s} \in \mathcal{M}} \left\langle \boldsymbol{s}, \nabla f(\boldsymbol{x}^{(t)}) \right\rangle$, $\boldsymbol{d}_t := \boldsymbol{s}^{(t)} - \boldsymbol{x}^{(t)}$, and $g_t := \left\langle \boldsymbol{d}_t, -\nabla f(\boldsymbol{x}^{(t)}) \right\rangle$
    if $g_t \leq \epsilon$ **then**
        return $\boldsymbol{x}^{(t)}$
    end
    Compute $\gamma_t := \arg\min_{\gamma \in [0,1]} f(\boldsymbol{x}^{(t)} + \gamma_t \boldsymbol{d}_t)$
    Update $\boldsymbol{x}^{(t+1)} := \boldsymbol{x}^{(t)} + \gamma_t \boldsymbol{d}_t$
end
return $\boldsymbol{x}^{(T)}$

Putting problem (2.4) — (2.7) into the context of Equation (3.1), consider the optimization problem

$$H(\boldsymbol{w}) \to \min \text{ over } \boldsymbol{w} \in \mathcal{M}_\alpha \tag{3.2}$$

where $H : \mathbb{R}^{n \times K} \to \mathbb{R}$ is a continuously differentiable function defined on a compact domain

$$\mathcal{M}_\alpha = \left\{ \boldsymbol{w} \in [0,1]^{n \times K} \,\middle|\, \sum_{k=1}^{K} w_{ik} = 1, \sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \geq n(1-\alpha) \text{ and } \sum_{i=1}^{n} w_{ik} \geq n_k \right\}. \tag{3.3}$$

It has been demonstrated by Pokojovy and Jobe (2022) that the Frank-Wolfe gradient method is equivalent in certain situations to the concentration step or the C-step iteration. Consequently, we apply Frank-Wolfe gradient method as our C-step to iteratively reduce the objective function. A concrete algorithm of the Frank-Wolfe gradient method to solve problem (3.2) is given in Algorithm 2.

Given a dataset, the Frank-Wolfe method repeatedly updates an initial weight matrix until there is no further change in the weight matrix. At this stage, the resulting weight matrix is considered a locally optimal weight matrix. The Frank-Wolfe gradient method also requires a solution to a linear minimization problem at each step of the iteration, which

involves the gradient of the objective function. Later in this chapter, we will discuss solution to the linear programming problem and present a lemma pertaining to this solution. Given this solution, the descent direction at each step of the iteration can be calculated by finding the difference between the solution to the linear minimization problem and the weight matrix. Subsequently, the Frank-Wolfe gap (g_t), which is defined as the Frobenius product of the descent direction and the anti-gradient of the objective function, is computed at every step. The affine-invariance property of the Frank-Wolfe gap (Jaggi, 2013) makes our C-step, the Frank-Wolfe gradient method affine-invaraint. We employ the Frank-Wolfe gap, which does not have any norm constraints, to decide when the algorithm terminates. Specifically, the algorithm will end if the gap g_t falls below a small positive value ϵ. On the other hand, given that g_t is still greater than ϵ, the weight matrix will be updated by taking the maximum step size, and the process will be repeated until a local minimum is reached. Figure 3.1 presents a plot of the objective function against step size, $\gamma_t \in [0, 1]$, which shows that the objective function $H(\boldsymbol{w})$ is minimized at the maximum step $\gamma = 1$.

Algorithm 2: Frank-Wolfe gradient method

Let $\boldsymbol{w}^{(0)} \in \mathcal{M}_\alpha$ and tolerance threshold $\epsilon > 0$.
for $t = 0 \dots T$ **do**
    Compute $\boldsymbol{s}^{(t)} := \arg\min_{\boldsymbol{s} \in \mathcal{M}_\alpha} \left\langle \boldsymbol{s}, \nabla H(\boldsymbol{w}^{(t)}) \right\rangle_{\mathcal{F}}$ where $H(\boldsymbol{w}^{(t)})$ is as given in Equation (2.17).
    Let $\boldsymbol{d}_t := \boldsymbol{s}^{(t)} - \boldsymbol{w}^{(t)}$
    Compute $g_t := \left\langle \boldsymbol{d}_t, -\nabla H(\boldsymbol{w}^{(t)}) \right\rangle_{\mathcal{F}}$
    if $g_t \leq \epsilon$ **then**
        return $\boldsymbol{w}^{(t)}$
    end
    Take maximum step, $\gamma_t = 1$
    Update $\boldsymbol{w}^{(t+1)} := \boldsymbol{w}^{(t)} + \gamma_t \boldsymbol{d}_t$
end
return $\boldsymbol{w}^{(T)}$

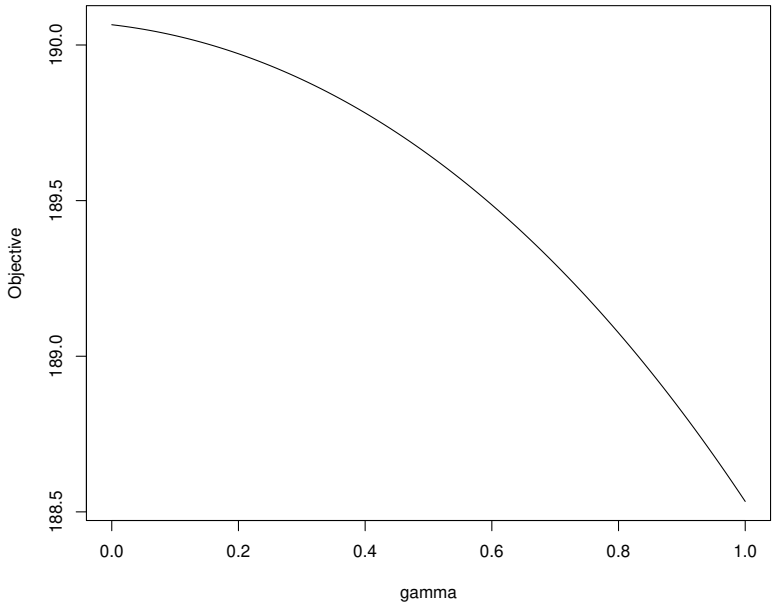

Figure 3.1: Objective function against step size γ

Theorem 2. *The minimum Frank-Wolfe gap g_t encountered at the t-th iteration of the Frank-Wolfe algorithm with maximum step size $\gamma_t = 1$ in Equation (3.2) is*

$$\tilde{g}_t \leq \frac{C}{\sqrt{t+1}} \qquad \text{for } t \geq 0 \tag{3.4}$$

where $C = C_{H,\mathcal{M}_\alpha} > 0$.

3.1.1 Solution to the Linear Programming Problem

To implement the Frank-Wolfe algorithm, it is necessary to solve a linear programming problem defined over the compact domain $\mathcal{M}_\alpha$. The approach to finding a solution to this linear minimization problem is explained below. In the context of Algorithm 2, consider the linear minimization problem stated on line 3. This problem seeks to find the solution matrix $\boldsymbol{s}^{(t)}$ that minimizes the expression $\left\langle \boldsymbol{s}, \nabla H(\boldsymbol{w}^{(t)}) \right\rangle_{\mathcal{F}}$, subject to the constraint that $\boldsymbol{s}$ belongs to the compact domain $\mathcal{M}_\alpha$. Let $\boldsymbol{s}^{(t)} \in [0,1]^{n \times K}$ be a solution to the linear programming

problem. At some $\boldsymbol{w}^{(t)}$, the dimension of the gradient of the objective function, denoted by $\nabla H(\boldsymbol{w}^{(t)})$, is $n \times K$, which is the same dimension of $\boldsymbol{s}^{(t)}$. We aim to obtain a solution matrix $\boldsymbol{s}^{(t)}$ that minimizes the Frobenius product between the gradient matrix and the solution matrix at the t-th iteration. The matrix $\boldsymbol{s}^{(t)}$ with components $s_{ik}^{(t)} = 1$ for all indices (i,k)'s of the gradient matrix with the minimum component on each row and 0 otherwise, is the solution matrix that minimizes the linear objective. See Equation (3.5) for the solution matrix to the linear programming problem.

Lemma 3. *For a given gradient $\nabla H(\boldsymbol{w}) \in \mathbb{R}^{n\times K}$, the solution to the linear programming problem on line 3 of Algorithm 2 is the matrix $\boldsymbol{s}^*$ with components $s_{ik}^* \equiv 1$ for all (i,k)'s corresponding to the row-column indices of the gradient matrix with minimum component and 0, otherwise.*

Proof. To show that the domain $\mathcal{M}_\alpha$ is non-empty, take $\boldsymbol{w} = (1,\ldots,1)/n(1-\alpha) \in \mathcal{M}_\alpha$. Therefore, $\mathcal{M}_\alpha$ is non-empty. The inclusion $\mathcal{M}_\alpha \subset [0,1]^{n\times K}$ implies that $\mathcal{M}_\alpha$ is bounded. Let $\boldsymbol{x}$ be a limit point of $\mathcal{M}_\alpha$. By definition, there exists a sequence $(\boldsymbol{x}_n)_n$ with $\boldsymbol{x}_n \in \mathcal{M}_\alpha$ such that $\boldsymbol{x}_n \to \infty$. But $\boldsymbol{x}_n \geq 0$ for all n which implies that $\boldsymbol{x} \geq 0$. Similarly, $\boldsymbol{x} \leq 1$ so $\boldsymbol{x} \in \mathcal{M}_\alpha$. Thus $\mathcal{M}_\alpha$ is closed since $\boldsymbol{x}$ was chosen arbitrary. Also, $\sum_{k=1}^K w_{ik} = 1$ and $\sum_{k=1}^K n_k^{(\boldsymbol{w})} \geq n(1-\alpha)$ in the limit. Since the objective function is a linear function it is continuous implying that there exists a minimum. Let $\boldsymbol{s}^* \in \mathcal{M}_\alpha$ be the minimizer of the linear programming problem.

For convenience, let $\boldsymbol{G}$ be the gradient where $\boldsymbol{G} \in \mathbb{R}^{n\times K}$ be fixed and the domain

$$\mathcal{M}_\alpha = \left\{ \boldsymbol{w} \in [0,1]^{n\times K} \middle| \sum_{k=1}^K w_{ik} = 1, \sum_{k=1}^K n_k^{(\boldsymbol{w})} \geq n(1-\alpha) \text{ and } \sum_{i=1}^n w_{ik} \geq n_k \right\}.$$

Let the linear programming problem be defined as

$$L(\boldsymbol{s}) = \big\langle \boldsymbol{G}, \boldsymbol{s} \big\rangle,$$

where $\big\langle \cdot, \cdot \big\rangle$ is the Frobenius inner product of $\boldsymbol{G}$ and $\boldsymbol{s}$. Let $k_i^* = \underset{k=1,\ldots,K}{\arg\min}\, G_{ik}$ where ties are broken arbitrary. Let $h = n(1-\alpha)$ and $G_{i_1^*}^* \leq \ldots \leq G_{i_h^*}^* \leq \ldots G_{i_n^*}^*$ where $\mathcal{I} = \{i_i^*, \ldots, i_n^*\}$ is

the permutation of the index set $\{1, \ldots, n\}$ and $G^*_{i^*_j} = G^*_{(j)}$. Let the optimal $\boldsymbol{s}$ be $\boldsymbol{s}^*$ and be defined below as

$$\boldsymbol{s}^*_{ik} = \begin{cases} 0 & \text{if } i \notin \{i^*_i, \ldots, i^*_h\} \\ 0 & \text{if } i \in \{i^*_i, \ldots, i^*_h\} \\ & \text{but } k \neq k^*_i \\ 1 & \text{if } i \in \{i^*_i, \ldots, i^*_h\} \\ & \text{and } k = k^*_i. \end{cases} \tag{3.5}$$

Observing that $\langle \boldsymbol{G}, \boldsymbol{s}^* \rangle \leq \langle \boldsymbol{G}, \boldsymbol{s} \rangle$ for all $\boldsymbol{s} \in \mathcal{M}_\alpha$ holds true, proves the assertion. A detailed proof is given in Anum (2021). □

3.2 Initial Clustering Algorithm

Iterative methods, as the name suggests, depend on the quality of solution at a previous step to obtain a quality solution at a current time step. As a result, many iterative methods suffer from initial starting condition effects. "Bad" initial starting choices (initial choices that are not sufficiently close to the true solution) or "warmstarts" can cause iterative methods to either converge to a suboptimal solution or fail to converge at all. In other situations, the number of iterations required to reach convergence (either an optimal or suboptimal) can be severely impacted.

Clustering methods are typically iterative methods that require the choice of initial data partition as an input. Many techniques used for initial partition of a data are documented in the literature (See MacQueen (1967); Forgy (1965); Kaufman and Rousseeuw (1990); Pena et al. (1999), etc.) Most of these procedures (for splitting data into groups) have the same underlying problem of being dependent on units of measurements. Consequently, there is no guarantee that the use of these initial partitioning techniques will not compromise the affine-invariance property of the final clusters. To have some layer of protection, we propose an initial data splitting technique called `kdetsInit`, given in Algorithm 3, for our k-dets

algorithm. At the start of the algorithm, all observational units in the dataset are considered to belong to a single cluster. The algorithm determines the size of each cluster after splitting (via the hypergeometric distribution). During the splitting step, an existing cluster $\mathcal{C}_i$ is selected at random and divided into two parts (not necessarily equal). Thus, to obtain K clusters, the algorithm performs $(K-1)$ division steps. The algorithm randomly selects an observational unit $\boldsymbol{x}^*$ from $\mathcal{C}_i$ without replacement and iterate the following steps. Another observational unit $\boldsymbol{x}_j$ is selected from $\mathcal{C}_i$ without replacement. The directional vector $\boldsymbol{d}_j$ is computed as the difference between $\boldsymbol{x}_j$ and $\boldsymbol{x}^*$. We find the orthonormal vector $\boldsymbol{u}_j$ to the directional vetor $\boldsymbol{d}_j$ using the Gram-Schmidt orthonormalization process. The cluster $\mathcal{C}_i$ is projected onto $\boldsymbol{u}_j$ to obtain a vector $\boldsymbol{z}_j$. We compute the robust scale of the projection $\boldsymbol{z}_j$ using the univariate MCD and we keep track of the minimum and maximum variance. The algorithm iterates until the ratio of the minimum and maximum variance is smaller than some small positive ϵ. The algorithm then partitions $\mathcal{C}_i$ into two subsets such that one subset $S_{\tilde{n}_1}$ has cluster size $\tilde{n}_1$ and the other subset $S_{\tilde{n}_2}$ has cluster size $\tilde{n}_2$ such that $\tilde{n}_1 + \tilde{n}_2 = n$. If we are searching for more than two clusters, the algorithm then randomly selects any of the subsets $S_{\tilde{n}_i}$ and repeats the process until the data are split in K clusters.

3.3 The *k*-dets Algorithm

We now formally present our proposed k-dets clustering algorithm given in Algorithm 4. The algorithm is an iterative procedure like many other clustering algorithms. It begins by invoking Algorithm 3 to split the data into an initial partition. An initial weight matrix is obtained for the initial partition of the dataset. The weight matrix has dimension $n \times K$ where each row sum is equal to one and each column sum equals the number of observations, n_k, in cluster $\mathcal{C}_k$. The algorithm then iterates the concentration step or the C-step as follows. The gradient of the objective function at $\boldsymbol{w}^{(m)}$ is computed. When there are no outliers and $\alpha = 0$ can be assumed, the non-robust version of the procedure (including all obsevational units in the dataset) is used by updating the weight matrix with Algorithm 2,

Algorithm 3: `kdetsInit`$(x_1, \ldots, x_n; K)$

`NumClust` $= 1$, $\mathcal{C}_1 = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^p$.

From multinomial distribution $\mathsf{Mult}(1, K, (p_1, \ldots, p_k))$, draw cluster sizes $n_1, \ldots, n_K$ to achieve after splitting such that $\sum_{i=1}^{K} n_i = n$.

while *`NumClust`* $< K$ **do**

  Select a cluster $\mathcal{C}_i$ randomly, $1 \leq i \leq$ `NumClust`, for splitting

  Draw $\boldsymbol{x}^*$ from $\mathcal{C}_i$ randomly without replacement

  for $j = 1, \ldots, |\mathcal{C}_i|$ **do**

    Draw $\boldsymbol{x}_j$ from $\mathcal{C}_i$ without replacement and compute the directional vector $\boldsymbol{d}_j = \boldsymbol{x}_j - \boldsymbol{x}^*$

    Find the vector $\boldsymbol{u}_j$ that is orthonormal to $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_j$ using Gram-Schmidt orthonormalization process:

    (i) $\boldsymbol{u}_j = \boldsymbol{d}_j - \sum_{m=1}^{j-1} \Pi_{\boldsymbol{u}_m}(\boldsymbol{d}_j)$, where the projection operator $\Pi_{\boldsymbol{u}_m}(\boldsymbol{d}_j) := \left(\frac{\boldsymbol{u}'\boldsymbol{d}}{\boldsymbol{u}'\boldsymbol{u}}\right)\boldsymbol{u}$,

    (ii) $\boldsymbol{u}_j = \frac{\boldsymbol{u}_j}{\|\boldsymbol{u}_j\|}$, for $\|\boldsymbol{u}_j\| \neq 0$.

    Project $\mathcal{C}_i$ on $\boldsymbol{u}_j$ via $\boldsymbol{z}_j = \Pi_{\boldsymbol{u}_j}(C_i)$ and compute $\hat{\sigma}_j^2 = \hat{\sigma}_{\mathrm{MCD}}^2(\boldsymbol{z}_j)$ (cf. Rousseeuw and Driessen (1999))

    Let $\hat{\sigma}_{\min}^2 = \min\{\hat{\sigma}_1^2, \ldots, \hat{\sigma}_j^2\}$ and $\hat{\sigma}_{\max}^2 = \max\{\hat{\sigma}_1^2, \ldots, \hat{\sigma}_j^2\}$

    if $\frac{\hat{\sigma}_{\min}^2}{\hat{\sigma}_{\max}^2} < \epsilon$ **then**

      $\boldsymbol{z}^* = \boldsymbol{z}_{j-1}$ (with $\boldsymbol{z}_0 = \boldsymbol{z}_1$ for $j = 1$)

      break

    end

  end

  Partition $\mathcal{C}_i = S_{\tilde{n}_1} \cup S_{\tilde{n}_2}$ and $S_{\tilde{n}_1} \cap S_{\tilde{n}_2} = \emptyset$ such that $\{\boldsymbol{x}_i \in S_{\tilde{n}_1} | z_i^* \leq h\}$ and $\{\boldsymbol{x}_i \in S_{\tilde{n}_2} | z_i^* > h\}$ where $|S_{\tilde{n}_1}| + |S_{\tilde{n}_2}| = |\mathcal{C}_i|$

  Assign $\mathcal{C}_i = S_{\tilde{n}_1}$

  `NumClust` $=$ `NumClust` $+ 1$

  $\mathcal{C}_{\texttt{NumClust}} = S_{\tilde{n}_2}$

end

and ensure that each cluster has at least $(p+1)$ observational units. A new partition is formed from the updated weight matrix and the objective value is computed. If there are outliers in the dataset, $\alpha > 0$, the robust procedure is used by ordering the minimum gradient in every i-th row in increasing order, and selecting the top $\lceil n(1-\alpha) \rceil$ observations. The weight matrix is updated leaving $n\alpha$ points unassigned. A new cluster membership is formed and the objective value is computed. This process is repeated n_{rep} times and the membership with the minimum objective value is selected as our optimal partition. Figure 3.2 shows that the objective function reduces at each iteration and converges in a finite number of steps.

Algorithm 4: k-dets$(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n; K, \alpha)$

1 Generate, say, $n_{\text{rep}} = 100$ random warmstarts, i.e.,
$\mathcal{C}^0 = (\mathcal{C}_1^0, \ldots, \mathcal{C}_K^0), \ldots, \mathcal{C}^{n_{\text{rep}}} = (\mathcal{C}_1^{n_{\text{rep}}}, \ldots, \mathcal{C}_K^{n_{\text{rep}}})$, using Algorithm 3 and obtain the initial weight matrix $\boldsymbol{w} \in [0,1]^{n \times K}$ from the initial partition.

2 For $m = 0, 1, \ldots,$ iterate the concentration step:

2.1 Compute the gradient $d_{ik}^{(m)}$ at $\boldsymbol{w}^{(m)}$.

2.2 Update the weight matrix $\boldsymbol{w}^{(m+1)}$ using Algorithm 2 ensuring that there are at least $(p+1)$ points in each cluster.

2.3 Form a new partition $\mathcal{C}^{(m+1)} = (\mathcal{C}_1^{(m+1)}, \ldots, \mathcal{C}_K^{(m+1)})$ from $\boldsymbol{w}^{(m+1)}$ and compute objective value $H(\mathcal{C}^{(m+1)})$.

2.4 Stop as soon as $H(\mathcal{C}^{(m)}) = H(\mathcal{C}^{(m+1)})$ or $\mathcal{C}^{(m)} = \mathcal{C}^{(m+1)}$.

3 Starting Step 2 from each warmstart, select final partition with the smallest value of objective $H(\mathcal{C})$.

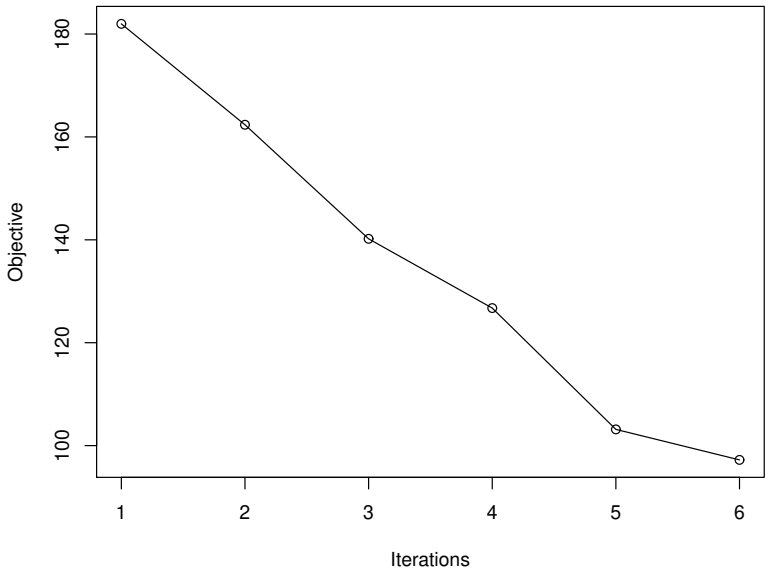

Figure 3.2: Objective function against iteration

Lemma 4. *Let $\mathcal{C}^0 = (C_1^0, \ldots, C_K^0)$ be an arbitrary (admissible) initial clustering. Apply the concentration step in Algorithm 4 to obtain $\mathcal{C}^1 = (C_1^1, \ldots, C_K^1)$. Then $H(\mathcal{C}^1) \leq H(\mathcal{C}^0)$.*

Conjecture 5. *Arguing similar to Theorem 1 of Rousseeuw and Driessen (1999), the C-step (Step 2) of Algorithm 4 is expected to reduce $H(\mathcal{C})$ leading to (monotonic) convergence in a finite number of steps.*

Chapter 4

Simulation Study

In this chapter, we discuss some popular cluster performance measures and perform several simulation studies using uncontaminated and contaminated datasets with different configurations to assess the performance of our proposed method. We compare the results from applying our proposed method to k-means when there are no outliers in the dataset and to tk-means when there is some contamination of the dataset, and `tclust` from the `tclust` package in `R` (Fritz et al., 2012). The methods are applied $n_{\text{rep}} = 100$ replications on the datasets and the average of the results are taken.

4.1 Performance Measures

The fundamental idea of clustering is to identify patterns in a sample dataset that may not be immediately apparent to businesses or researchers learning about a dataset. Consequently, a wide variety of clustering techniques have been developed in literature. These clustering techniques, when applied, produces cluster labels to the observational units of the sample data. Nevertheless, having just the cluster labels to the observational units does not tell the cluster quality or cluster "accuracy". To evaluate the quality of clustering results, several clustering performance measures have been developed. We will discuss some of the commonly used clustering performance measures in this section. The selected clustering metrics require reference partitions or ground truth labels and have values ranging from 0 to 1.

4.1.1 Cluster Accuracy (CA)

A very basic cluster performance measure is the so called cluster "accuracy." We obtain the cluster "accuracy" by first taking all possible permutations of the resulting clusters from the clustering methods and compare each of these possible permutations with the ground truth clusters by computing the confusion matrix and then the accuracy given below:

$$\text{CA} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

The highest accuracy from these matches is selected. Positive in this scenario means one cluster label, and negative is considered as the other cluster label for binary clusters or the remaining cluster labels for multi-class (more than two cluster labels). A positive prediction occurs when an observations' label in the ground truths is the same in the resulting cluster labels. True positive (TP) is the number of correct positive predictions, false positive (FP) is the number of incorrect positive predictions, false negative (FN) is the number of incorrect negative predictions and true negative (TN) is the number of correct negative predictions. For datasets with two clusters, TP, FP, FN and TN are directly obtained from the confusion matrix and the accuracy is obtained using the formula above. For datasets with multi-class, the confusion matrix is given as a combination of all classes. In such situation, obtaining TP, FP, FN and TN are not as direct as in the binary class. In this case, we find TP, FP, FN and TN for individual classes and compute the corresponding accuracy. The overall accuracy is the average of all accuracies.

4.1.2 Rand Index (RI)

The Rand Index measures the similarity between the clustering results or cluster labels from a clustering algorithm and the true class labels of the observational units. The calculation of Rand Index involves obtaining the number of pairs of data points that are either in the same cluster in both the clustering result and the true class labels or in different clusters in both the clustering result and the true class labels. The Rand Index also takes into consideration the number of pairs of points that are misrepresented. It is calculated as the ratio of the

number of pairs of points that are correctly clustered and the sum of the number of pairs of points that are correctly and incorrectly clustered which provides a more comprehensive measure of similarity. The formula is given as:

$$\text{RI} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}},$$

where true positives (TP) is the number of pairs of data points that are in the same cluster in both the true and predicted clustering, true negative (TN) is the number of pairs of data points that are in different clusters in both the true and predicted clustering, false negative (FN) is the number of pairs of data points that are in the same cluster in the true clustering but in different clusters in the predicted clustering, and false positive (FP) is the number of pairs of data points that are in different clusters in the true clustering but in the same cluster in the predicted clustering. The Rand Index value ranges from 0 to 1, with a value of 1 indicating a perfect match between the clustering result and the true class labels. A Rand Index value of 0 indicates that the clustering result and the ground truth labels do not agree. The Rand Index performance measure was introduced by Rand (1971).

4.1.3 Purity

Purity (Manning, 2008) is a clustering performance measure that assesses how well the clustering result matches the true class labels. Purity measures the frequency of the most common class label in each cluster and takes the average over all clusters. The formula for computing purity is given as follows:

$$\text{purity}(\mathcal{C}, \mathcal{G}) = \frac{1}{N} \sum_{k=1,\ldots,K} \max_{j=1,\ldots,K} |C_k \cap G_j|,$$

where $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ is the set of clusters obtained from the clustering technique and $\mathcal{G} = \{G_1, G_2, \ldots, G_j\}$ is the ground truth cluster set. The elements C_k and G_j in sets $\mathcal{C}$ and $\mathcal{G}$ represent all the observational units in the k-th cluster from the clustering technique applied and the j-th cluster ground truth. Here, $|\cdot|$ is the number of observations in the intersection. The value of purity ranges from 0 to 1, with a value of 1 indicating a perfect clustering result.

4.1.4 Normalized Mutual Information (NMI)

Another popular performance measure is based on the idea of mutual information hence called the normalized mutual information (NMI). This metric measures the degree of agreement between the ground truth labels and the resulting cluster assignments. Computing this metric is two-fold: first compute the mutual information between the ground truth labels and then the resulting cluster assignments. This information is then normalized via the entropy. The NMI formula is given below:

$$\text{NMI}(\mathcal{C}, \mathcal{G}) = \frac{2I(\mathcal{C}, \mathcal{G})}{H(\mathcal{C}) + H(\mathcal{G})},$$

where $\mathcal{C}$ is the cluster assignments from the clustering technique and $\mathcal{G}$ is the ground truth labels, $I(\mathcal{C}, \mathcal{G})$ is the mutual information between the two assignments, $H(\mathcal{C})$ is the entropy of the ground truth labels. The mutual information is mathematical expressed as:

$$\begin{aligned} I(\mathcal{C}, \mathcal{G}) &= \sum_{k=1}^{K} \sum_{j=1}^{K} P(C_k \cap G_j) \log \frac{P(C_k \cap G_j)}{P(C_k)P(G_j)} \\ &= \sum_{k=1}^{K} \sum_{j=1}^{K} \frac{|C_k \cap G_j|}{N} \log \frac{N|C_k \cap G_j|}{|C_k||G_k|}, \end{aligned}$$

where $P(C_k)$, $P(G_j)$, and $P(C_k \cap G_j)$ represent the probabilities that an observational unit is assigned to cluster C_k, G_j, and their intersection. The entropy is given as:

$$\begin{aligned} H(\mathcal{C}) &= -\sum_{k=1}^{K} P(C_k) \log P(C_k) \\ &= -\sum_{k=1}^{K} \frac{|C_k|}{N} \log \frac{|C_k|}{N}. \end{aligned}$$

The NMI score also ranges from 0 to 1. The NMI was first introduced by Strehl and Ghosh (2002).

4.2 Uncontaminated Datasets

4.2.1 Clusters on the Vertices of a Unit Square

- ***Description of dataset $\boldsymbol{X}$.***

  Huang and Yang (2020) (cf. Chapter 5) provides some simulated datasets for testing the performance of their proposed method. We will adopt three of their simulated datasets. We generated the initial synthetic dataset to consist of four distinct clusters, each with its center at one of the four vertices of a unit square, resulting in well-separated clusters. A total of 80 bivariate observations were generated from $\mathcal{N}(\boldsymbol{\mu}_k, (1/16)\boldsymbol{I}_2)$ for $k = 1, 2, 3, 4$ where $\boldsymbol{\mu}_k$ is the mean vector of the k-th cluster and $\boldsymbol{I}_2$ is the 2×2 identity matrix, with 20 observations located at each vertex $\boldsymbol{\mu}_k$. To generate this data set, the `rmvnorm` function from the `mvtnorm` package in `R` was used. We set a seed of 800 for reproducibility of this dataset. The left panel of Figure 4.1 shows the plot of $\boldsymbol{X}$.

- ***Description of dataset $\boldsymbol{Y}$.***

  The second dataset is a transformed version of the initial population $\boldsymbol{X}$. The linear transformation of $\boldsymbol{X}$ is given as $\boldsymbol{Y} = \boldsymbol{X} \times \begin{pmatrix} 3 & 0 \\ 0 & 1/3 \end{pmatrix}$. The first feature or column of $\boldsymbol{X}$ is multiplied or stretched by a factor of three whiles the second column is shrinked by a factor of three. A visualization of the transformed dataset show four clusters that may not be immediately distinguishable, as two clusters on the left appear to be almost merged into a single cluster, and the same is true for the two clusters on the right. A plot of dataset $\boldsymbol{Y}$ is shown in the middle panel of Figure 4.1. Since the original population has four clusters, we will search for four clusters when we apply various clustering techniques in our analysis.

- ***Description of dataset $\boldsymbol{Z}$.***

  For dataset $\boldsymbol{Z}$, another transformation of the initial population $\boldsymbol{X}$ was taken by multi-

plying $\boldsymbol{X}$ by a non-singular matrix $\begin{pmatrix} 4.1 & 2.1 \\ 1.9 & 1.1 \end{pmatrix}$. Unlike previous transformation, the off-diagonal elements of this non-singular matrix are non-zero. The features of $\boldsymbol{Z}$ are a linear combinations of the features of the initial population. The plot of $\boldsymbol{Z}$ depicted in the right panel of Figure 4.1 indicates that the transformed features follow a linear trend, while the four clusters appear to be poorly separated, rendering the structure more challenging to cluster.

- ***Analysis of datasets $\boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{Z}$.***

  With the datasets now generated (see Figure 4.1), our next step involves analyzing them via applying our proposed k-dets clustering algorithm, the widely used k-means technique and `tclust`. For all these synthetic datasets, we will search for four clusters as in the initial population (see left panel of Figure 4.1). It is worth mentioning that since there are no outliers in these datasets, every observational unit will be allocated or assigned a cluster. Following the application of all the clustering techniques (mentioned above) on the datasets, we will evaluate the quality of the clustering results using the performance measures detailed in Section 4.1. To ensure a more accurate comparison of the quality of the clustering results, we will apply the clustering methods several times on the datasets, $n_{\text{rep}} = 100$ replications, and compute the average of the outcomes.

  Given that the clusters are distinctly separated in $\boldsymbol{X}$ (see left panel of Figure 4.1), it comes as no surprise that all the clustering methods applied are able to correctly identify all four subgroups in this dataset. All three methods had perfect clustering performances. Figure 4.2 shows the projected cluster results from all the methods. Applying k-means clustering to the dataset $\boldsymbol{Y}$ (which is an affine transformation of $\boldsymbol{X}$) to identify four clusters yields low accuracy, as shown in Table 4.1. This is because clusters 1 (colored green) and 4 (colored black), as well as clusters 2 (colored

red) and 3 (colored blue), appear to be joined together. Consequently, k-means fails to distinguish between cluster 1 and cluster 4, as well as between cluster 2 and cluster 3, resulting in a high rate of misclassification. The `tclust` method had a better clustering performance than k-means. In contrast, k-dets successfully identified all four true clusters. Figure 4.3 shows best cluster results from the methods in comparison to the ground truth clusters. Also, for dataset $\boldsymbol{Z}$, k-dets successfully identified all four clusters. Both k-means and `tclust` had a better clustering performance when applied to dataset $\boldsymbol{Z}$ than when applied to dataset $\boldsymbol{Y}$. Figure 4.4 shows best projected cluster results from the methods in comparison to the ground truth clusters. Because of the affine-invariance property of k-dets, the clustering results for dataset $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ when k-det was applied are the same. Since the $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are transformations of $\boldsymbol{X}$, or have different units of measurements in comparison to $\boldsymbol{X}$, both k-means and `tclust` view them as different datasets and thereby the difference in their clustering results. Table 4.1 reports the average cluster accuracy (ACA), average Rand index (ARI), average purity (AP) and average normalized mutual information (ANMI) values computed based on $n_{\text{rep}} = 100$ replications.

Table 4.1: ACA, ARI, AP and ANMI comparisons for synthetic data.

	$\boldsymbol{X}$				$\boldsymbol{Y}$				$\boldsymbol{Z}$			
Methods	ACA	ARI	AP	ANMI	ACA	ARI	AP	ANMI	ACA	ARI	AP	ANMI
k-means	1.000	1.000	1.000	1.000	0.575	0.753	0.575	0.516	0.938	0.941	0.938	0.850
`tclust`	1.000	1.000	1.000	1.000	0.810	0.886	0.810	0.803	0.940	0.944	0.940	0.855
k-dets	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

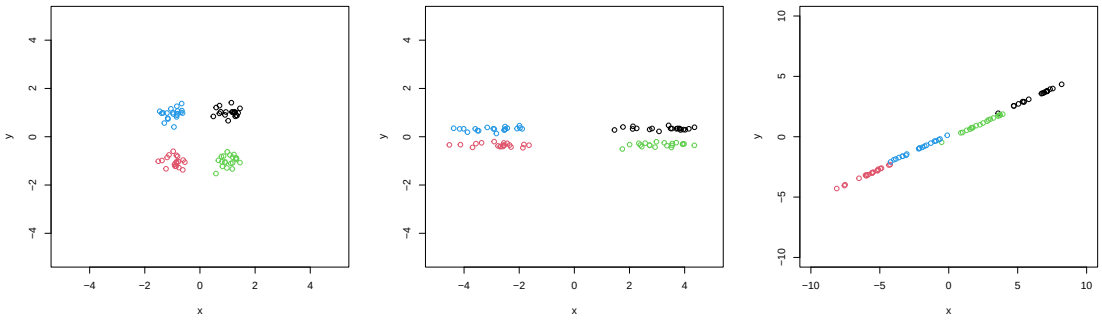

Figure 4.1: Scatterplots of the synthetic datasets. The plot in the left panel represents $\boldsymbol{X}$, displaying the four clusters positioned at the vertices of the unit square; the middle panel plot is a plot of $\boldsymbol{Y}$ which is a transformation of $\boldsymbol{X}$; similarly, the right panel plot is a transformation of $\boldsymbol{X}$ with the features now following a linear trend.

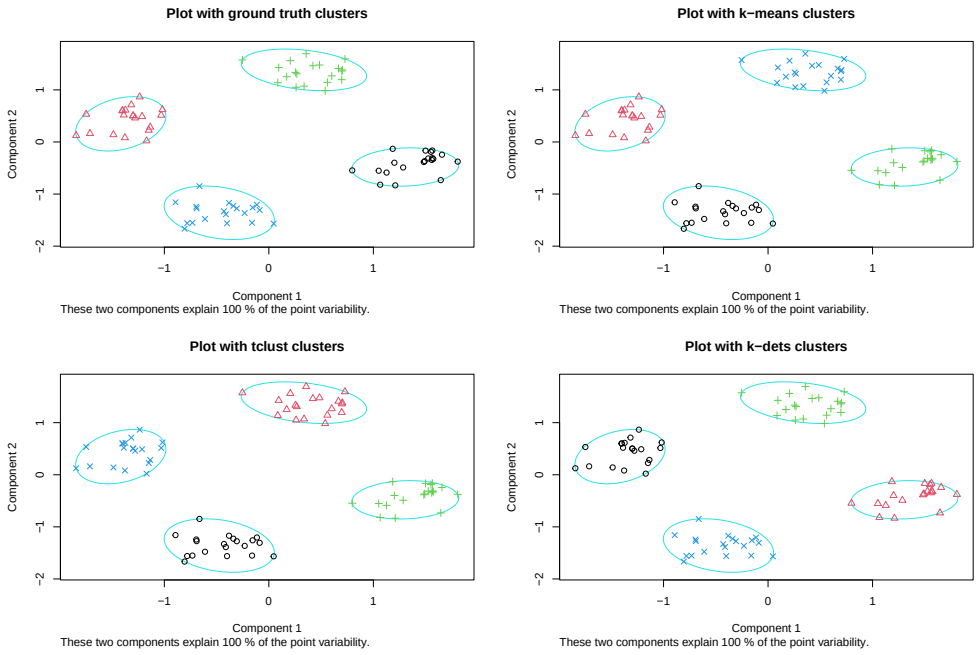

Figure 4.2: Plots of dataset $\boldsymbol{X}$ and projected resulting clusters. The top left panel displays $\boldsymbol{X}$ with four distinguishable clusters, while the top right and bottom left panels display best clusters obtained using the k-means and `tclust` methods, respectively. Finally, the bottom right panel displays the best k-dets clusters.

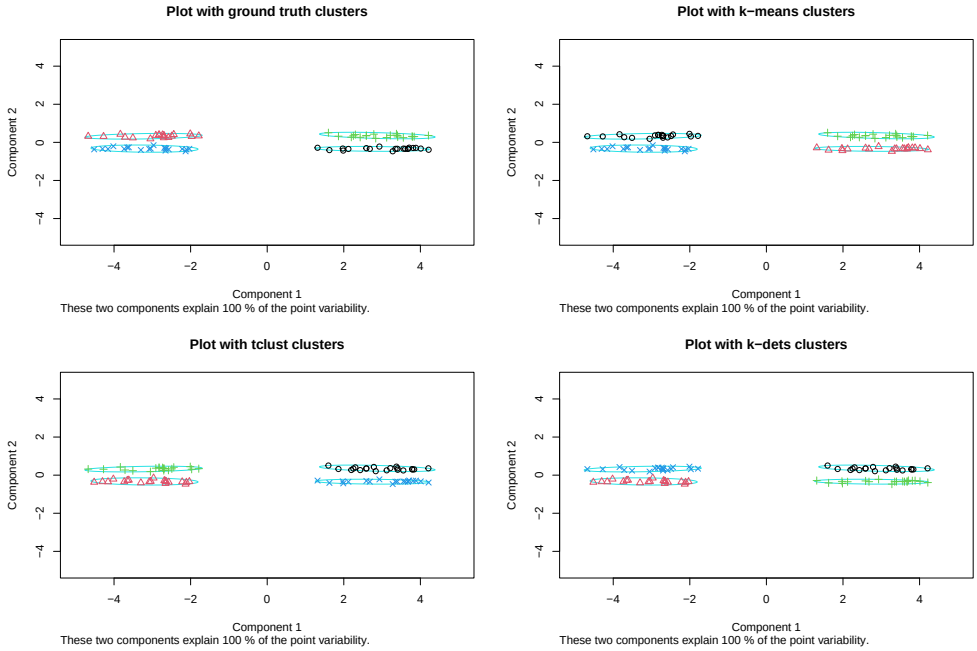

Figure 4.3: Plots of dataset $\boldsymbol{Y}$ and resulting clusters. The top left panel displays the transformed dataset $\boldsymbol{Y}$ with ground truth clusters. The top right panel shows the best clusters obtained using the k-means algorithm, while the bottom left panel illustrates the best clusters obtained by `tclust` clustering method. Finally, the bottom right panel depicts the best cluster partition from k-dets.

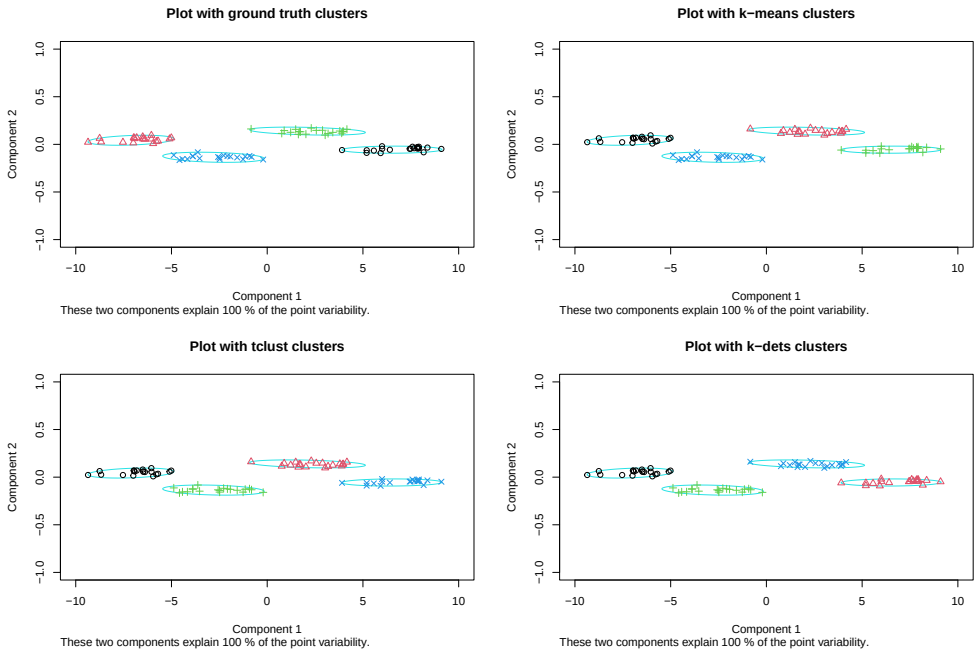

Figure 4.4: Plots of dataset $\boldsymbol{Z}$ and resulting clusters. The top left panel is a plot of the transformed dataset $\boldsymbol{Z}$ with ground truth clusters. The top right panel shows the best clustering assignments obtained using the k-means, while the bottom left panel displays the best clustering assignments obtained from `tclust`. Finally, the bottom right panel depicts the k-dets.

4.2.2 Miscellaneous Simulated Data

To further demonstrate the performance of our proposed k-dets clustering algorithm, we simulate the selected series of datasets adopted from Kumar and Orlin (2008). One of these datasets has three clusters, while the remaining six datasets have two clusters each. Each cluster consists of 100 observational units with two or three features. These datasets are generated from the Gaussian distribution using different configurations, including various cluster shapes (e.g., spherical and elliptical) and sizes, number of clusters. The datasets are generated using the `rmvnorm` function from the `mvtnorm` package in `R`, with different parameters of the normal distribution as described below.

M.1 ***Two spherical clusters of equal sizes***:

Cluster 1 has mean vector $\boldsymbol{\mu}_1 = (0, 0)'$ and covariance matrix $\boldsymbol{\Sigma}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Cluster 2 has mean vector $\boldsymbol{\mu}_2 = (3, 0)'$ and covariance matrix $\boldsymbol{\Sigma}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

The dataset M.1 contains 200 bivariate observations from a Gaussian distribution, which are divided into two spherical clusters with 100 observations each. A visualization of M.1 is displayed in the top left panel of Figure 4.5. We set a seed of 10 for reproducibility of the data. The plot shows that the clusters are not immediately apparent due to overlapping observations. The resulting clusters obtained through the k-means, `tclust` and k-dets methods are displayed in the top right panel, bottom left panel, and bottom right panel of Figure 4.5, respectively. It is worth mentioning that since the clustering methods are applied several times, Figure 4.5 plots the best partition from each of the methods. Table 4.2 reports the average cluster accuracy (ACA), average Rand index (ARI), average purity (AP) and average normalized mutual information (ANMI) values computed based on 100 replications.

M.2 ***Two spherical clusters of different sizes***:

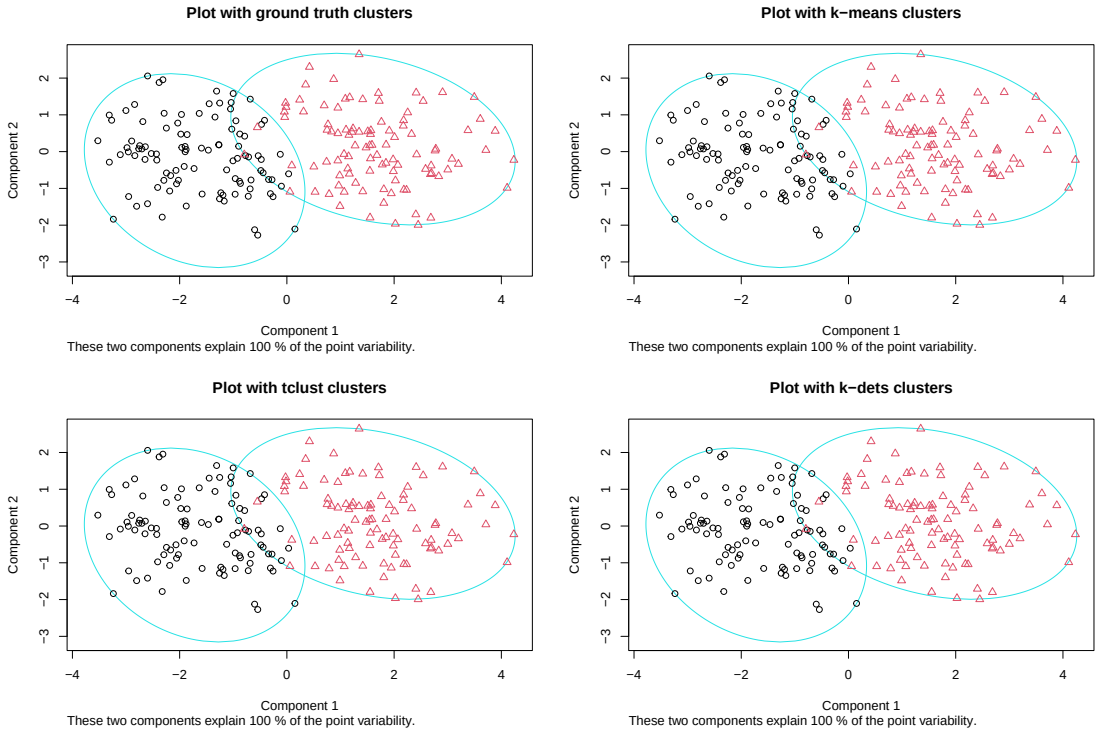

Figure 4.5: The top left panel shows dataset M.1, which consists of two spherical clusters of equal size. The resulting clusters (projected) obtained using k-means are displayed in the top right panel. The bottom left panel illustrates the resulting clusters (projected) obtained from `tclust`, while the bottom right panel depicts clustering results (projected) from k-dets.

Cluster 1 has mean vector $\boldsymbol{\mu}_1 = (0, 0)'$ and covariance matrix $\boldsymbol{\Sigma}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Cluster 2 has mean vector $\boldsymbol{\mu}_2 = (20, 0)'$ and covariance matrix $\boldsymbol{\Sigma}_2 = \begin{pmatrix} 50 & 0 \\ 0 & 50 \end{pmatrix}$.

The dataset M.2 consists of 200 observations drawn from a Gaussian distribution. Each observational unit is bivariate. The observations are well separated into two spherical clusters, each containing 100 observations. The clusters are centered around $\boldsymbol{\mu}_1 = (0, 0)'$ and $\boldsymbol{\mu}_2 = (20, 0)'$, and differ in sphere size. The cluster centered around $\boldsymbol{\mu}_2$ is more spread out. A plot of this dataset can be found in the top left panel

of Figure 4.6, which shows the two well separated clusters. The top right panel of Figure 4.6 shows the resulting clusters obtained using k-means, the bottom left panel of Figure 4.6 shows the resulting clusters from `tclust`, whiles the bottom right panel shows resulting clusters from k-dets. The best partition out of all replications from the three methods are plotted in Figure 4.6. Table 4.2 provides numerical summary of the quality of the clustering results of the three clustering methods. A seed of 10 was set for reproducibility of the dataset.

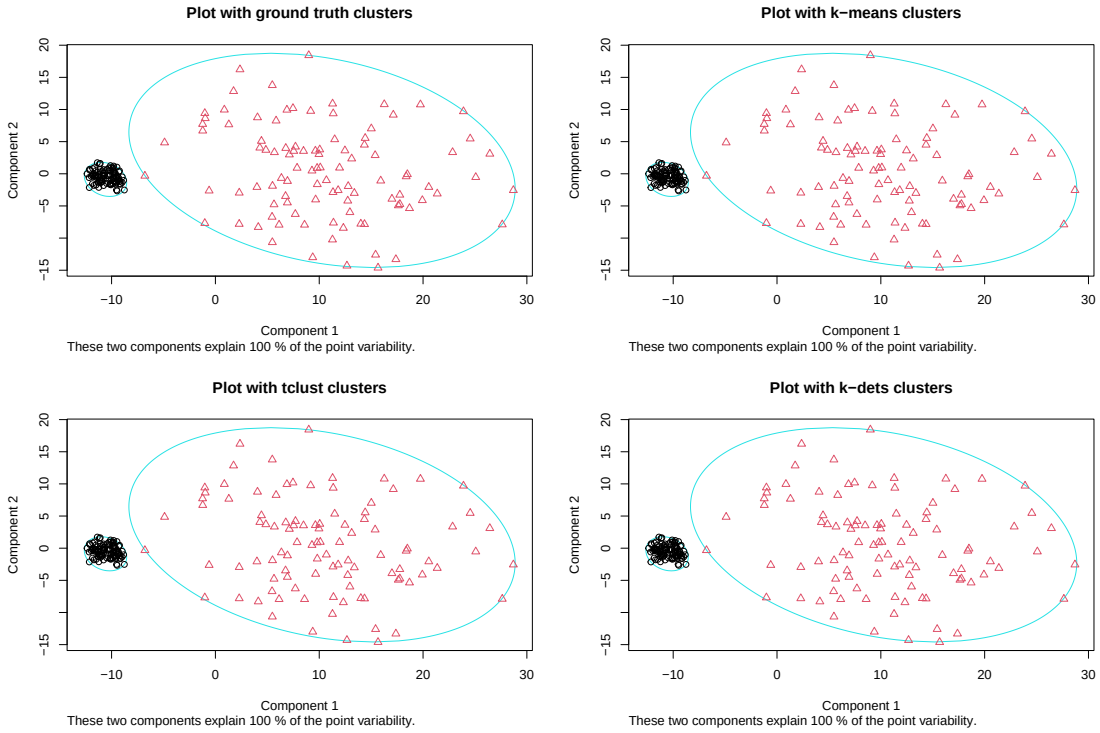

Figure 4.6: The top left panel displays data M.2 with the ground truth clusters. The top right panel illustrates the resulting clusters (projected) obtained using k-means. The bottom left panel shows the resulting clusters (projected) obtained from `tclust`, and the bottom right panel depicts resulting clusters (projected) obtained from k-dets.

M.3 ***One spherical and one elliptical cluster***:

Cluster 1 has mean vector $\boldsymbol{\mu}_1 = (0, 0)'$ and covariance matrix $\boldsymbol{\Sigma}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Cluster 2 has mean vector $\boldsymbol{\mu}_2 = (5, 0)'$ and covariance matrix $\boldsymbol{\Sigma}_2 = \begin{pmatrix} 5 & 6 \\ 6 & 9 \end{pmatrix}$.

M.3 also consists of 200 bivariate observations forming two well separated clusters: one elliptical and one spherical. The clusters have centres around $\boldsymbol{\mu}_1 = (0, 0)'$ and $\boldsymbol{\mu}_2 = (5, 0)'$. The cluster centered at $\boldsymbol{\mu}_2 = (5, 0)'$ has an elliptical shape. The top left panel of Figure 4.7 shows a plot of M.3 with the ground truth clusters. The resulting projected cluster assignments from k-means, `tclust` and k-dets are shown in the top right panel, bottom left panel and bottom right panel of Figure 4.7. The average of the performance measures over $n_{\text{rep=100}}$ replications are reported in Table 4.2. A seed, 10, was set for reproducibility of the dataset.

M.4 ***Two elliptical clusters of equal shapes and sizes***:

The mean vector for cluster 1 is $\boldsymbol{\mu}_1 = (0, 0)'$, and the corresponding covariance matrix is $\boldsymbol{\Sigma}_1 = \begin{pmatrix} 5 & 6 \\ 6 & 9 \end{pmatrix}$. Cluster 2 has mean vector $\boldsymbol{\mu}_2 = (5, 0)'$ and covariance matrix $\boldsymbol{\Sigma}_2 = \begin{pmatrix} 5 & 6 \\ 6 & 9 \end{pmatrix}$.

Dataset M.4 was also generated from a Gaussian distribution. M.4 also has 200 observations, each with two features. There are two elliptical clusters of equal shapes and sizes in this data, each of which contains 100 observations. The clusters have centers around $\boldsymbol{\mu}_1 = (0, 0)'$ and $\boldsymbol{\mu}_2 = (5, 0)'$. A pictorial representation of the well separated clusters of dataset M.4 is given in the top left panel of Figure 4.8. Top right panel and bottom left panel of Figure 4.8 are graphs of best resulting projected clusters from k-means and `tclust` clustering methods. The bottom right panel shows the best partition (projected) of M.4 when k-dets was applied. The average of the performance measures are given in Table 4.2. A seed, 10, was set for the reproducibility of the

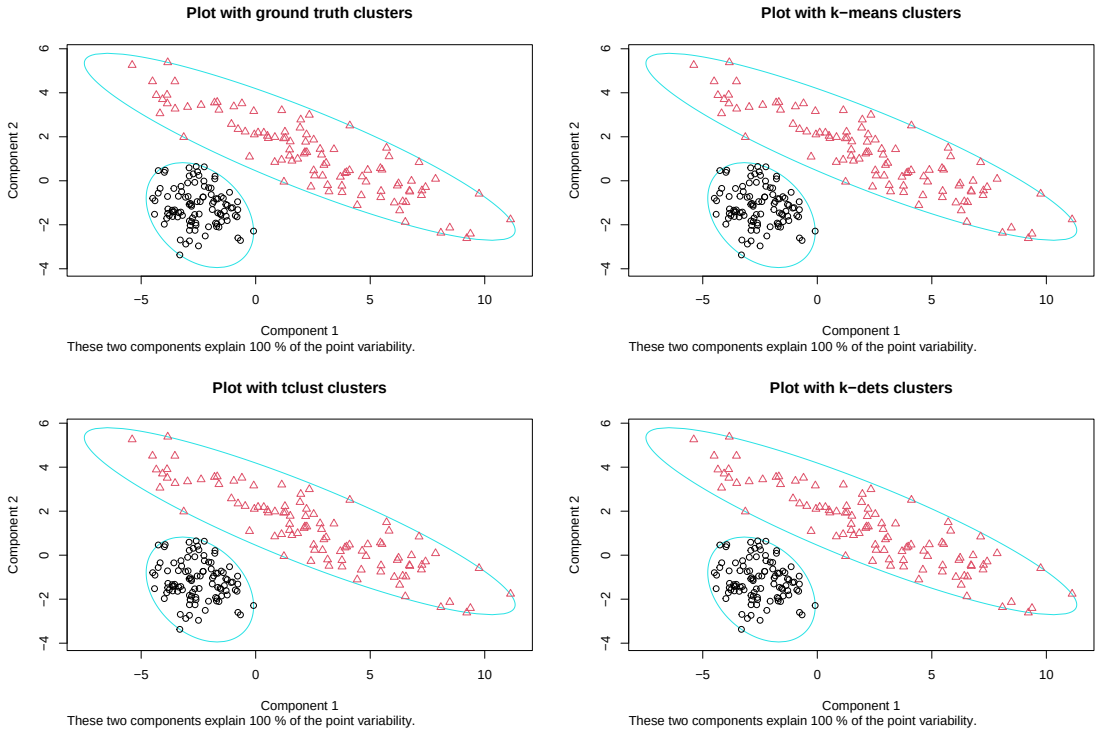

Figure 4.7: In the top left panel, we see data M.3, consisting of one spherical cluster and one elliptical cluster. The top right panel displays the best resulting clusters (projected) obtained using k-means technique. The bottom left panel illustrates the best resulting clusters (projected) obtained from `tclust`. Finally, the bottom right panel shows the best results (projected) from k-dets.

results.

M.5 ***Two elliptical clusters of different shapes and sizes***:

Cluster 1 has mean vector $\boldsymbol{\mu}_1 = (0, 0)'$ and covariance matrix $\boldsymbol{\Sigma}_1 = \begin{pmatrix} 6 & -7 \\ -7 & 9 \end{pmatrix}$. The mean vector for cluster 2 is $\boldsymbol{\mu}_2 = (5, 0)'$ and the corresponding covariance matrix is $\boldsymbol{\Sigma}_2 = \begin{pmatrix} 10 & 7 \\ 7 & 5 \end{pmatrix}$. Dataset M.5 comprises 200 bivariate observations that are sampled from a Gaussian distribution, where two elliptical clusters exist in the dataset, each containing 100 observations. These elliptical clusters present in M.5 differ in size and

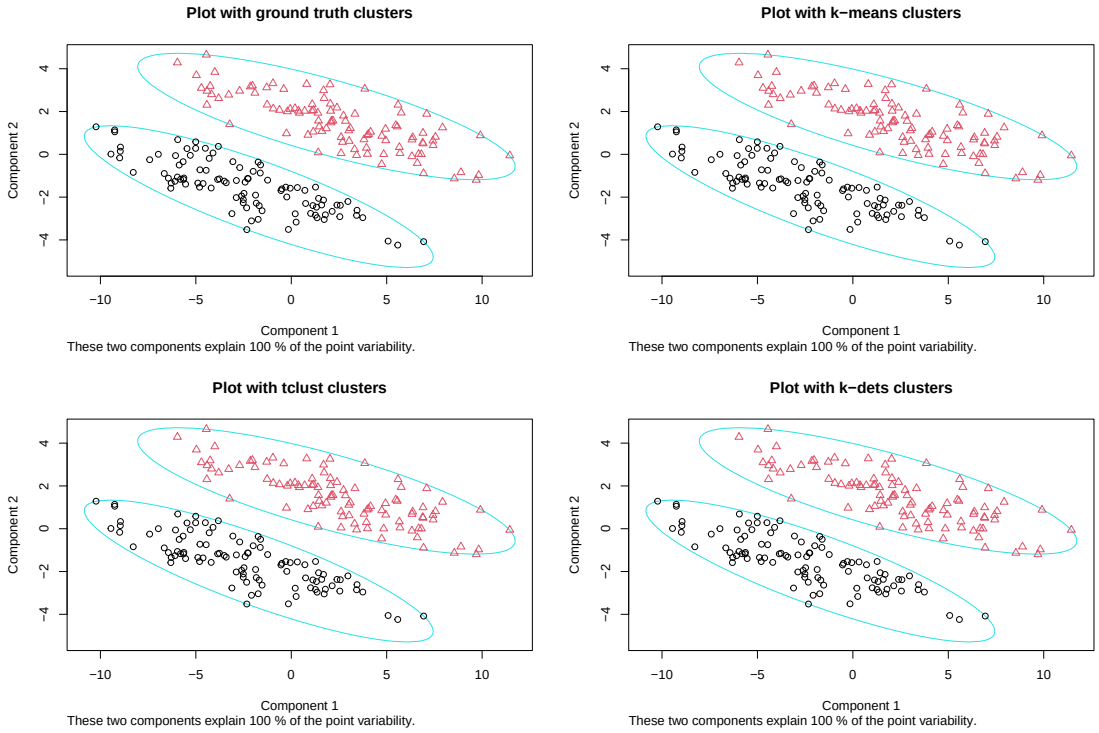

Figure 4.8: Dataset M.4 is plotted in the top left panel with ground truth clusters. The plot shows two elliptical clusters of equal shapes. The top right plot is the best clustering results (projected) from k-means and the bottom left plot is the best clustering results from `tclust`. The bottom right plot the best partition (projected) from k-dets.

shape. The centers of the clusters are located at $\boldsymbol{\mu}_1 = (0, 0)'$ and $\boldsymbol{\mu}_2 = (5, 0)'$. As shown in the top left panel of Figure 4.9, the two elliptical clusters overlap. The best resulting clusters (projected) obtained using the k-means and `tclust` methods are presented in the top right and bottom left panels of Figure 4.9. The bottom right panel plots the best resulting clusters (projected) from k-dets. Table 4.2 reports the average of the performance metrics for the three clustering methods. A seed, 10, was set for reproducibility.

M.6 ***Three elliptical clusters of different shapes and sizes***:

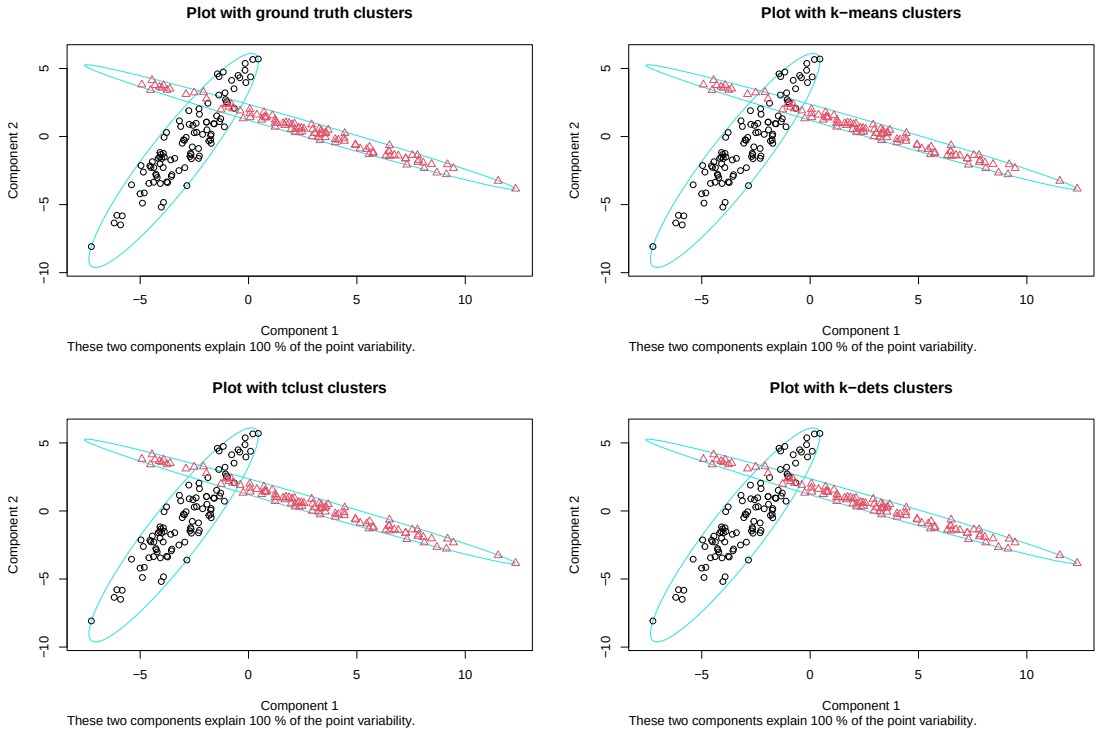

Figure 4.9: A plot of M.5 (top left panel) with ground truth clusters, best resulting clusters from k-means (top right panel), `tclust` (bottom left panel) and k-dets (bottom right panel).

Cluster 1 has mean vector $\boldsymbol{\mu}_1 = (0, 0)'$ and covariance matrix $\boldsymbol{\Sigma}_1 = \begin{pmatrix} 6 & -6 \\ -6 & 9 \end{pmatrix}$.

Cluster 2 has mean vector $\boldsymbol{\mu}_2 = (10, 0)'$ and covariance matrix $\boldsymbol{\Sigma}_2 = \begin{pmatrix} 12 & 7 \\ 7 & 5 \end{pmatrix}$.

Cluster 3 has mean vector $\boldsymbol{\mu}_3 = (0, 5)'$ and covariance matrix $\boldsymbol{\Sigma}_3 = \begin{pmatrix} 13 & -7 \\ -7 & 5 \end{pmatrix}$.

Dataset M.6 is distinct from other synthetic datasets in that it consists of three clusters, each with 100 bivariate observations. These clusters are elliptical in shape, with varying sizes and shapes, and are centered around $\boldsymbol{\mu}_1 = (0, 0)'$, $\boldsymbol{\mu}_2 = (10, 0)'$, and $\boldsymbol{\mu}_3 = (0, 5)'$. Figure 4.10 presents a plot of this data, revealing that the clusters overlap. The best

resulting clusters (projected) from applying k-means, `tclust` and k-dets are displayed in the top right, bottom left and bottom right panels of Figure 4.10, and Table 4.2 presents the quality of the clustering methods. A seed, 10, was set for reproducibility.

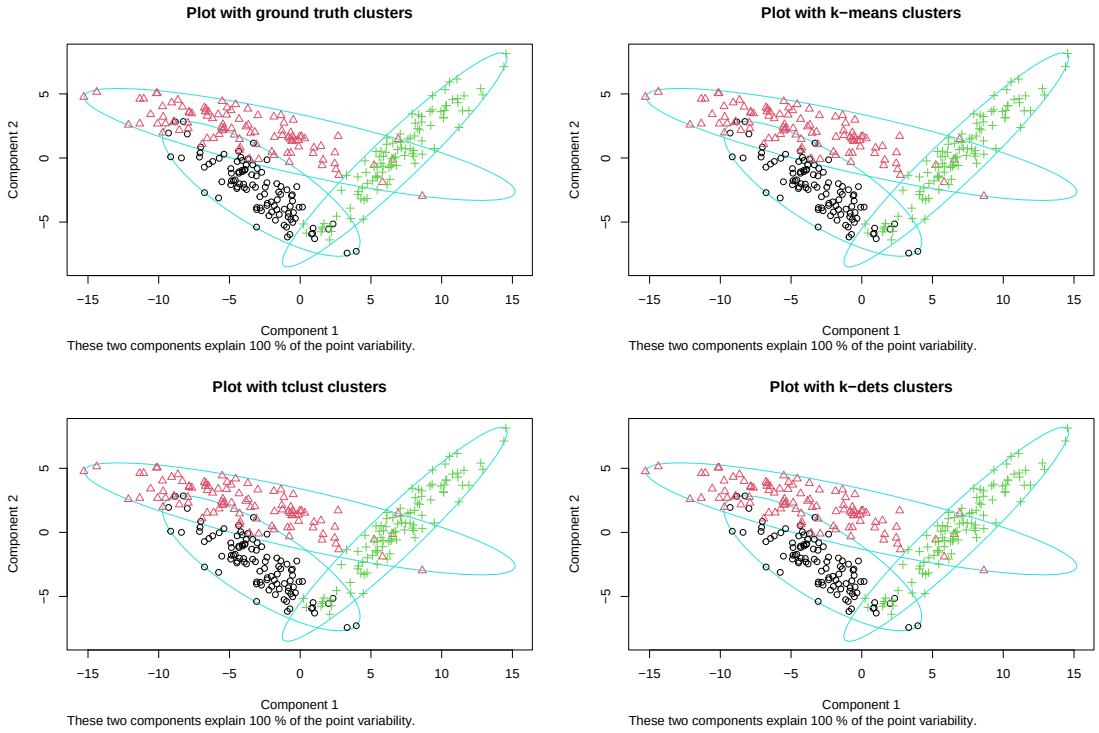

Figure 4.10: A plot of M.6 (top left panel) with ground truth clusters, best resulting clusters from k-means (top right panel), `tclust` (bottom left panel) and k-dets (bottom right panel).

M.7 ***Two elliptical cluster of different shapes and sizes***:

The mean vector for cluster 1 is $\boldsymbol{\mu}_1 = (0, 0, 0)'$ and the corresponding covariance matrix is $\boldsymbol{\Sigma}_1 = \begin{pmatrix} 6 & 3 & 4 \\ 3 & 8 & 2 \\ 4 & 2 & 5 \end{pmatrix}$. Cluster 2 has mean vector $\boldsymbol{\mu}_2 = (0, 0, 5)'$ and covariance matrix

$$\boldsymbol{\Sigma}_2 = \begin{pmatrix} 10 & 2 & 4 \\ 2 & 6 & 2 \\ 4 & 2 & 2 \end{pmatrix}.$$

The dataset labeled M.7 comprises 200 trivariate observations that have been randomly generated from a Gaussian distribution. This dataset contains two elliptical clusters, each containing 100 observations. These clusters differ in size and shape and are centered around $\boldsymbol{\mu_1} = (0, 0, 0)'$ and $\boldsymbol{\mu_2} = (0, 0, 5)'$. A 2D plot of the dataset is shown in the top left panel of Figure 4.11, which demonstrates that the two elliptical clusters overlap and the clusters are not immediately apparent. However, this may not be the case when the dataset is plotted in 3D. The top right and bottom left panels of Figure 4.11 illustrate the best clusters resulting (projected) from the application of the k-means and `tclust` methods. The bottom right panel shows the best clusters resulting (projected) from applying k-dets on M.7. Table 4.2 displays the average of the performance metrics. A seed, 10, is set for reproducibility of the dataset.

Table 4.2: ACA, ARI, AP and ANMI comparisons for series of synthetic datasets.

	ACA			API			AP			ANMI		
Datasets	k-means	`tclust`	k-dets	k-means	`tclust`	k-dets	k-means	`tclust`	k-dets	k-means	`tclust`	k-dets
M.1	0.960	0.980	0.980	0.923	0.961	0.961	0.960	0.980	0.980	0.796	0.878	0.878
M.2	0.960	0.995	0.999	0.923	0.990	0.998	0.960	0.995	0.999	0.796	0.960	0.992
M.3	0.850	1.000	1.000	0.744	1.000	1.000	0.850	1.000	1.000	0.511	1.000	1.000
M.4	0.735	1.000	1.000	0.608	1.000	1.000	0.735	1.000	1.000	0.168	1.000	1.000
M.5	0.825	0.950	0.970	0.710	0.905	0.942	0.825	0.950	0.970	0.464	0.761	0.808
M.6	0.680	0.923	0.926	0.701	0.905	0.908	0.680	0.923	0.926	0.387	0.745	0.750
M.7	0.790	0.985	1.000	0.667	0.970	1.000	0.790	0.985	1.000	0.265	0.902	1.000

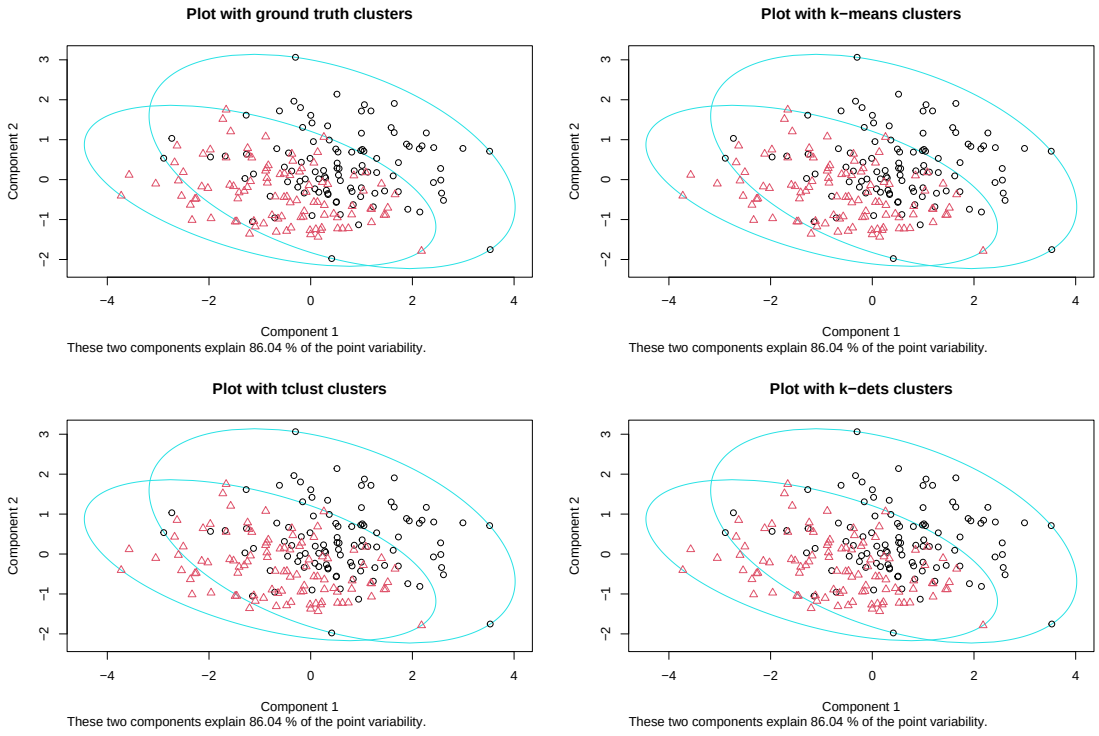

Figure 4.11: A plot of M.7 (top left panel) with ground truth clusters, best resulting clusters from k-means (top right panel), `tclust` (bottom left panel) and k-dets (bottom right panel).

4.3 Contaminated Data

4.3.1 `M5data`: Three Groups with Different Scales

The preceding section demonstrated the comparison between k-dets, `tclust` and the most popular clustering technique using various synthetic datasets that were free from any form of contamination. In this section, we will showcase another advantage of the k-dets algorithm, which was one of the primary motivations for its proposal. We will demonstrate how the proposed algorithm performs when data contamination is present and compare it to trimmed k-means (tk-means), which is considered a robust version of k-means algorithm, and `tclust`. For this analysis, we utilized the `M5data` from the `R` package `tclust` (viz., Fritz et al. (2012) and García-Escudero et al. (2008)). The dataset comprises 1800 bivariate normal data, which contains three distinct clusters, with an additional 200 observations of background noise surrounding the Gaussian components. The "clean" observations are unevenly distributed among the three clusters, with a distribution ratio of 1:2:2. The three clusters have centers at $\boldsymbol{\mu}_1 = (0, 8)', \boldsymbol{\mu}_2 = (8, 0)'$ and $\boldsymbol{\mu}_3 = (-8, -8)'$ with corresponding covariance matrices $\boldsymbol{\Sigma_1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \boldsymbol{\Sigma_2} = \begin{pmatrix} 45 & 0 \\ 0 & 30 \end{pmatrix}$, and $\boldsymbol{\Sigma_3} = \begin{pmatrix} 15 & -10 \\ -10 & 15 \end{pmatrix}$. The background noise, on the other hand, was generated from a uniform distribution, where the qualifying points were those with squared Mahalanobis distances greater than $\chi^2_{2,0.975}$. In Figure 4.12 (top-left panel), the `M5data` plot shows that two of the clusters are joined together or have significant overlap, with outliers enclosing the three clusters. We aim to identify the three true clusters while not assigning 10% of the observations in the dataset (the background noise). tk-means, `tclust` and our proposed algorithm k-dets were applied to this dataset by taking several replications, each time computing the performance metrics discussed in Section 4.1. Table 4.3 reports the average results.

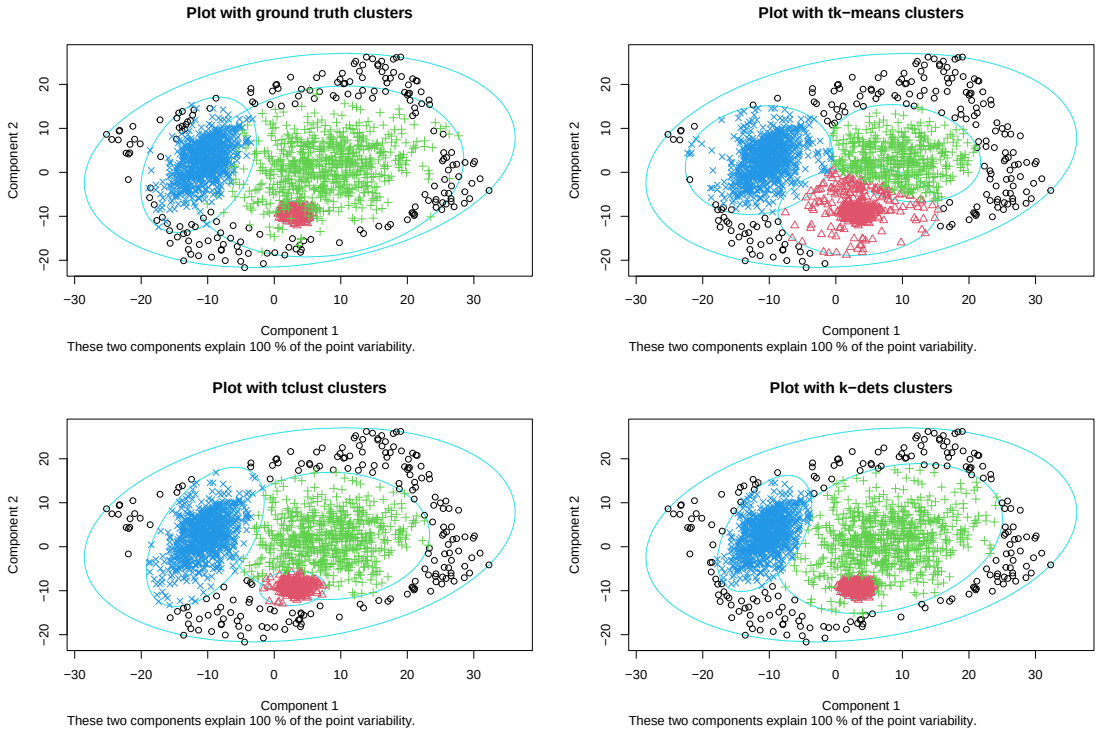

Figure 4.12: A plot of `M5data` (top left panel) with ground truth clusters, best resulting clusters (projected) from tk-means (top right panel), `tclust` (bottom left panel), and k-dets clusters (bottom right panel).

Table 4.3: ACA, ARI, AP and ANMI comparison for M5data

Methods	ACA	ARI	AP	ANMI
tk-means	0.853	0.865	0.853	0.678
`tclust`	0.933	0.932	0.933	0.802
k-dets	0.958	0.957	0.958	0.853

4.3.2 M.6 with Background Noise

In Subsection 4.2.2, a thorough description of dataset M.6 is given. We contaminate this dataset by randomly replacing 15% (45 points) of the observational units with background

noise to demonstrate the strength of k-dets, in terms of robustness. The background noise or outliers are randomly sampled from a uniform distribution over $\big[\lfloor \min(\boldsymbol{X}) \rfloor - 0.5 \times (\max(\boldsymbol{X}) - \min(\boldsymbol{X})), \lceil \max(\boldsymbol{X}) \rceil + 0.5 \times (\max(\boldsymbol{X}) - \min(\boldsymbol{X}))\big]$, where $\boldsymbol{X}$ is the dataset, to enclose the clusters. Observations whose squared mahalanobis distances greater than $\chi^2_{2,0.95}$ are considered as background noise. At $\alpha = 0.15$, all methods were applied several times to this dataset and we assess their performance by reporting the average of the results for each of the performance metric. See Table 4.4 for these statistics. Figure 4.13 is a plot of the data and the best resulting clusters (projected) from the three methods.

Table 4.4: ACA, ARI, AP and ANMI comparison for M.6 with background noise

Methods	ACA	ARI	AP	ANMI
tk-means	0.693	0.771	0.693	0.520
`tclust`	0.933	0.930	0.933	0.804
k-dets	0.938	0.935	0.938	0.814

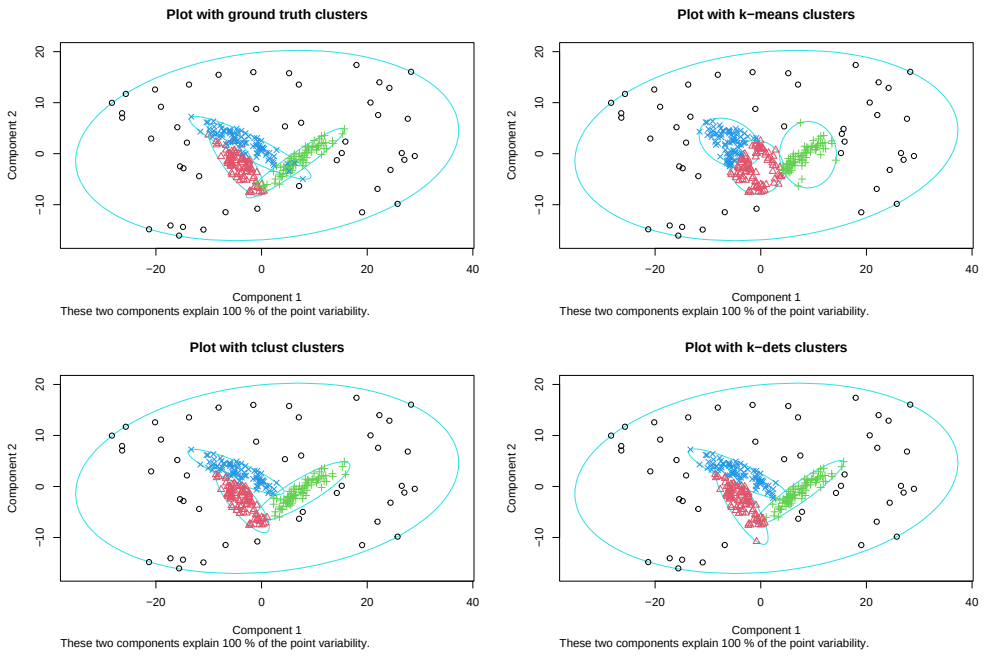

Figure 4.13: A plot of M.6 dataset with background noise with ground truth clusters (top left panel), best resulting clusters (projected) from tk-means (top right panel), `tclust` (bottom left panel) and k-dets (bottom right panel).

Chapter 5

Real Data Exploration

In this chapter, we conduct an analysis of a real-world dataset that is free from contamination, the Iris dataset. Additionally, we analyze one contaminated data set, which is the Iris data set with some level of contamination. The performance of our proposed algorithm, k-dets, is assessed and compared to k-means (for uncontaminated data) and tk-means (for contaminated data) and `tclust`.

5.1 Uncontaminated Data

5.1.1 Iris Data

The Iris dataset was initially presented by Fisher (1936) and has since been widely utilized for testing and benchmarking new algorithms, including those for clustering and classification. We will also adopt the Iris dataset to assess the performance of our proposed cluster method, k-dets, in comparison to k-means and `tclust`. This dataset consists of 150 plants, divided into three subgroups of Iris species: Setosa, Versicolor, and Virginica, with each group containing 50 plants. The dataset is multivariate, and it records four measurements on each Iris plant sampled: petal length, petal width, sepal length, and sepal width. A graph of the dataset (see Figure 5.1) shows that the Setosa group are well separated from the remaining two groups. The Vesicolor and Virginica groups overlap making the two groups not immediately apparent. We expect all the methods to be able to reveal the Setosa group without any challenges. The real challenge would be to separate the overlapping groups. As in other examples, we will apply the methods several times and take the average of

the results. See Table 5.1 for summary of results. The best resulting clusters (projected) obtained from the methods, as well as the ground truth, are presented in Figure 5.2.

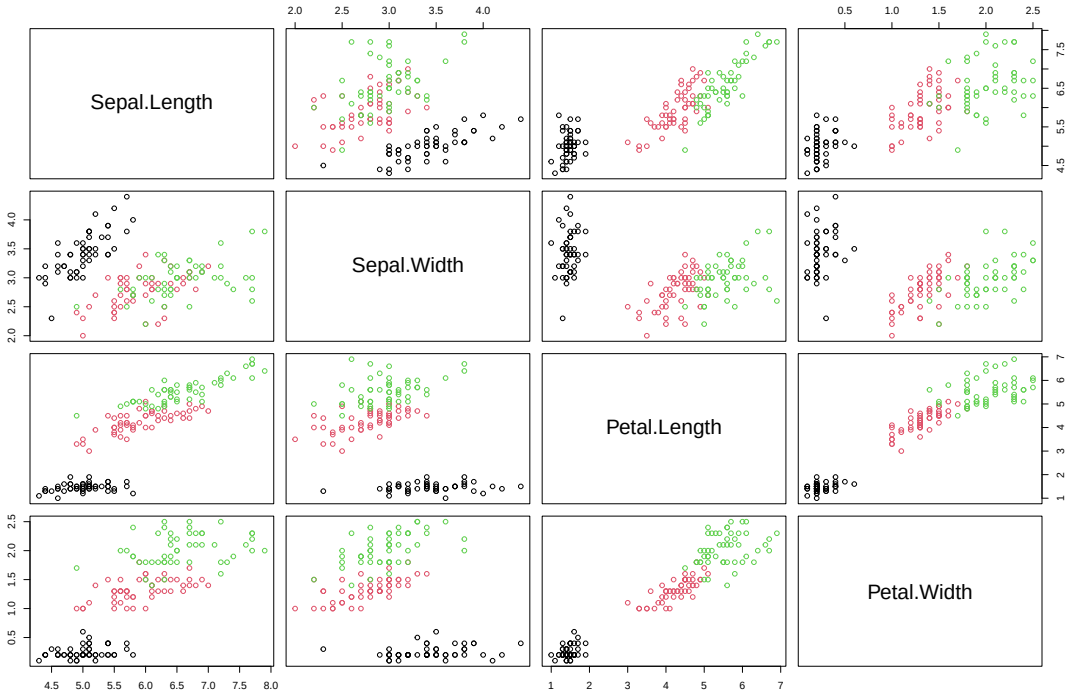

Figure 5.1: Scatter plot matrix of Iris data (Setosa, Versicolor, Virginica)

Table 5.1: ACA, ARI, AP and ANMI comparison for k-means, `tclust` and k-dets on Iris dataset

Methods	ACA	ARI	AP	ANMI
k-means	0.893	0.880	0.893	0.758
`tclust`	0.979	0.973	0.979	0.928
k-dets	0.971	0.963	0.971	0.904

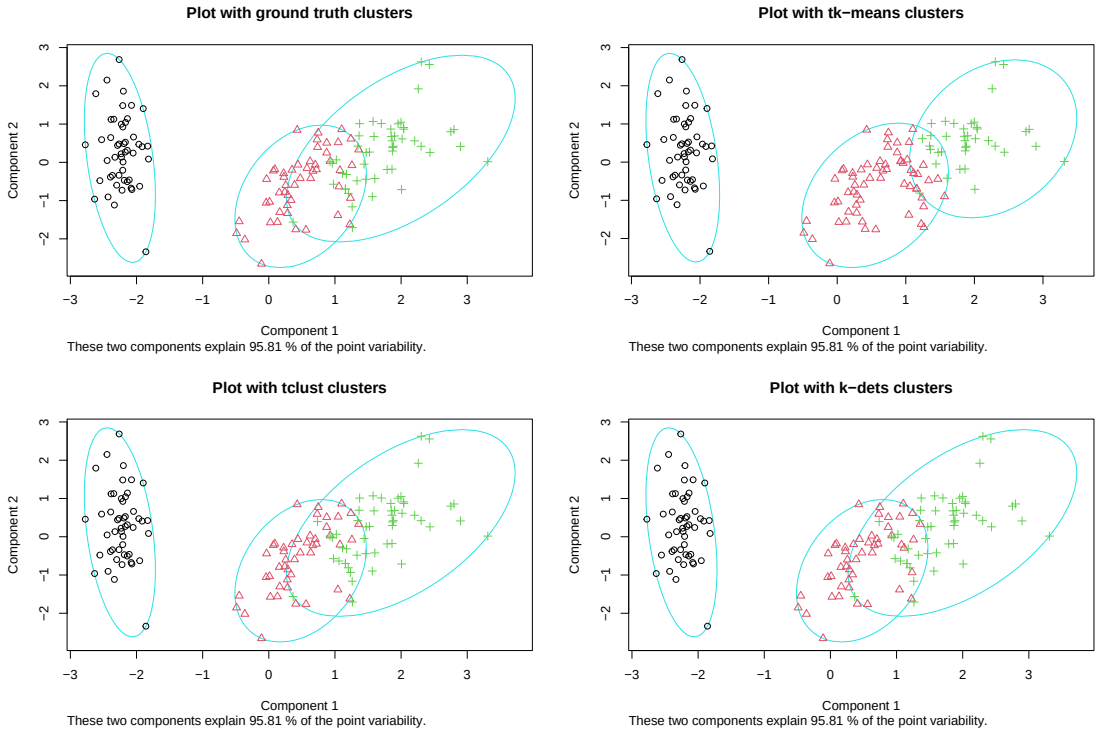

Figure 5.2: Resulting clusters for Iris data. Top left panel is the ground truth plot; top right panel represents the best resulting clusters (projected) from tk-means, bottom left panel is the best clustering (projected) from `tclust`; bottom right is the best clustering (projected) from k-dets.

5.2 Contaminated Data

5.2.1 Iris Data with Contamination

In Subsection 5.1.1, we introduced the Iris dataset. In this Subsection, we will now randomly contaminate the Iris dataset with background noise sampled from a uniform distribution over $\big[\lfloor \min(\boldsymbol{X}) \rfloor - 0.5 \times (\max(\boldsymbol{X}) - \min(\boldsymbol{X})), \lceil \max(\boldsymbol{X}) \rceil + 0.5 \times (\max(\boldsymbol{X}) - \min(\boldsymbol{X}))\big]$. We consider data points with squared mahalanobis distances to the cluster centers greater than $\chi^2_{4,0.95}$ as background noise, and for this dataset, such points account for 20% of the observation units. Accordingly, we applied the three methods to search for three true clusters at $\alpha = 0.20$. We

repeat this for a total of 100 times and measure the quality of the clusters from the all the methods. Table 5.2 reports the average of the cluster quality for the methods. Figure 5.4 is a plot of the data and the best resulting clusters (projected) from the methods applied.

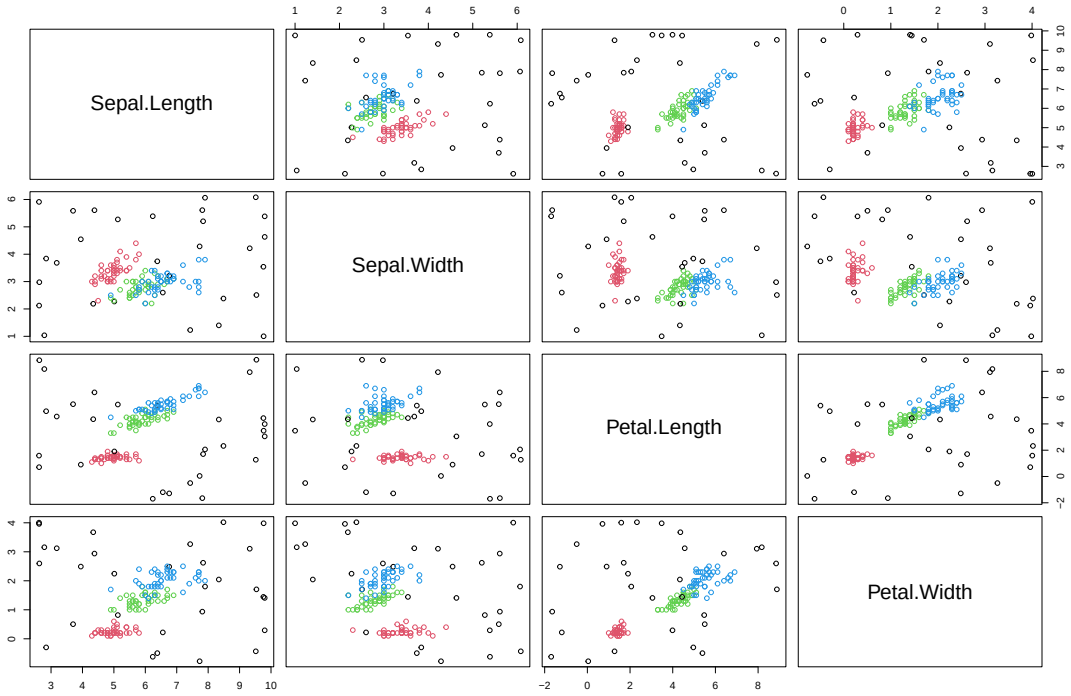

Figure 5.3: Contaminated (20%) Iris dataset.

Table 5.2: ACA, ARI, AP and ANMI comparison for tk-means, `tclust` and k-dets on contaminated Iris data

Methods	ACA	ARI	AP	ANMI
tk-means	0.913	0.922	0.913	0.842
`tclust`	0.905	0.916	0.905	0.860
k-dets	0.949	0.956	0.949	0.914

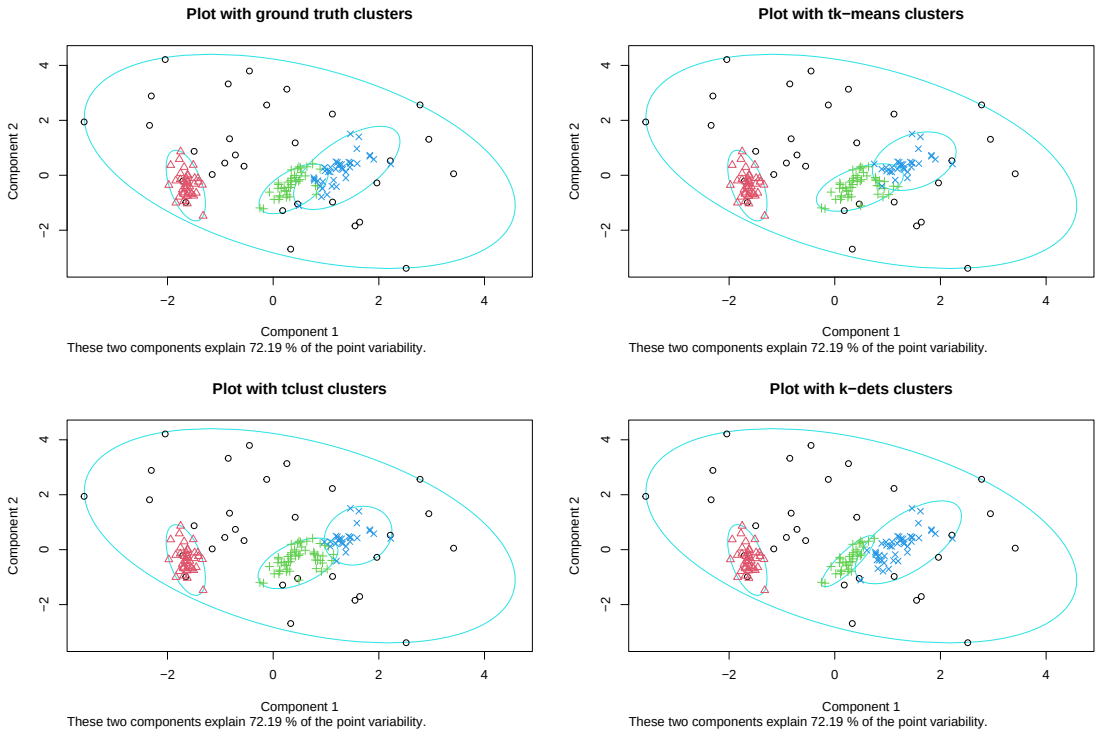

Figure 5.4: Top left panel is the ground truth clusters of the contaminated Iris data; top right panel represents the best resulting clusters (projected) from tk-means, bottom left panel is the best clustering (projected) from `tclust`; bottom right is the best clustering (projected) from k-dets.

Chapter 6

Equal Covariance Assumption

The framework in Chapter 2 assumes that the cluster covariances are different, i.e., $\boldsymbol{S}_k \neq \boldsymbol{S}_{k'}$ for $k \neq k'$. When we have enough reason(s) to suspect that the clusters are structured in a way that the covariance matrices are similar and n is small, we use an estimate of the covariance called the pooled covariance, $\boldsymbol{S}_{\text{pooled}}$, which involves the individual cluster covariances. The objective function in Equation (2.4), which is an approximation of a trimmed negative log-likelihood function is then defined in terms of the pooled covariance given below

$$H(\boldsymbol{w}) = -\sum_{k=1}^{K}\sum_{i\in C_k} \log\Big(\pi_k^{(\boldsymbol{w})} f(\boldsymbol{x}_i; \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{\text{pooled}}^{(\boldsymbol{w})})\Big) \tag{6.1}$$

with constraints as before. Each cluster mean vector and covariance are define as before, see Equations (2.5) and (2.6). The pooled covariance is defined as the weighted average of the individual covariances

$$\boldsymbol{S}_{\text{pooled}}^{(\boldsymbol{w})} = \frac{\sum_{k=1}^{K}(n_k - 1)\boldsymbol{S}_k^{(\boldsymbol{w})}}{\sum_{k=1}^{K}(n_k - 1)} \tag{6.2}$$

where n_k and $\boldsymbol{S}_k^{(\boldsymbol{w})}$ are respectively, the number of observations and the covariance associated to the k-th cluster. Following Theorem 1, a similar proof can be shown that the affine-invariance property of the objective function in Equation 6.1 is not destroyed.

Let $\boldsymbol{S}_{\boldsymbol{x},k}^{(\boldsymbol{w})}$ be the covariance of the k-th cluster of the sample data $\boldsymbol{X}$ and $\boldsymbol{S}_{\boldsymbol{y},k}^{(\boldsymbol{w})}$ be the covariance of the k-th cluster of the transformed data $\boldsymbol{Y}$, $\boldsymbol{S}_{\boldsymbol{x},\text{pooled}}^{(\boldsymbol{w})}$ be the pooled covariance for the sample data, $\boldsymbol{S}_{\boldsymbol{x},\text{pooled}}^{(\boldsymbol{w})}$ be the pooled covariance of $\boldsymbol{Y}$ defined as:

$$\boldsymbol{S}_{\boldsymbol{y},\text{pooled}}^{(\boldsymbol{w})} = \frac{\sum_{k=1}^{K}(n_k - 1)\boldsymbol{S}_{\boldsymbol{y},k}^{(\boldsymbol{w})}}{\sum_{k=1}^{K}(n_k - 1)}$$

$$
\begin{aligned}
&= \frac{(n_k-1)\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{y},1}}{\sum_{k=1}^{K}(n_k-1)} + \frac{(n_k-1)\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{y},2}}{\sum_{k=1}^{K}(n_k-1)} + \ldots + \frac{(n_k-1)\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{y},K}}{\sum_{k=1}^{K}(n_k-1)} \\
&= \frac{(n_k-1)\boldsymbol{A}\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},1}\boldsymbol{A}'}{\sum_{k=1}^{K}(n_k-1)} + \frac{(n_k-1)\boldsymbol{A}\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},2}\boldsymbol{A}'}{\sum_{k=1}^{K}(n_k-1)} + \ldots + \frac{(n_k-1)\boldsymbol{A}\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},K}\boldsymbol{A}'}{\sum_{k=1}^{K}(n_k-1)} \\
&= \boldsymbol{A}\frac{\sum_{k=1}^{K}(n_k-1)\boldsymbol{S}^{(\boldsymbol{w})}_{x,k}}{\sum_{k=1}^{K}(n_k-1)}\boldsymbol{A}' \\
&= \boldsymbol{A}\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}}\boldsymbol{A}'.
\end{aligned}
$$

The pdf of the multivariate normal distribution of the transformed data with the pooled covariance can be expressed as:

$$
\begin{aligned}
f(\boldsymbol{y}_i|\bar{\boldsymbol{y}}^{(\boldsymbol{w})}_k, \boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{y},\text{pooled}}) &= (2\pi)^{-p/2}\left|\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{y},\text{pooled}}\right|^{-1/2}\exp\left(-\frac{1}{2}(\boldsymbol{y}_i-\bar{\boldsymbol{y}}^{(\boldsymbol{w})}_k)'(\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{y},\text{pooled}})^{-1}(\boldsymbol{y}_i-\bar{\boldsymbol{y}}^{(\boldsymbol{w})}_k)\right) \\
&= (2\pi)^{-p/2}\left|\boldsymbol{A}\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}}\boldsymbol{A}'\right|^{-1/2}\times \\
&\times\exp\left(-\frac{1}{2}(\boldsymbol{A}\boldsymbol{x}_i-\boldsymbol{A}\bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k)'(\boldsymbol{A}\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}}\boldsymbol{A}')^{-1}(\boldsymbol{A}\boldsymbol{x}_i-\boldsymbol{A}\bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k)\right) \\
&= (2\pi)^{-p/2}\left|\boldsymbol{A}\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}}\boldsymbol{A}'\right|^{-1/2}\times \\
&\times\exp\left(-\frac{1}{2}((\boldsymbol{x}_i-\bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k)'\boldsymbol{A}')(\boldsymbol{A}')^{-1}(\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}})^{-1}\boldsymbol{A}^{-1}(\boldsymbol{A}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k))\right) \\
&= (2\pi)^{-p/2}\left|\boldsymbol{A}\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}}\boldsymbol{A}'\right|^{-1/2}\exp\left(-\frac{1}{2}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k)'(\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}})^{-1}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k)\right) \\
&= (2\pi)^{-p/2}\left|\boldsymbol{A}\right|^{-1/2}\left|\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}}\right|^{-1/2}\left|\boldsymbol{A}'\right|^{-1/2}\times \\
&\times\exp\left(-\frac{1}{2}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k)'(\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}})^{-1}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k)\right) \\
&= (2\pi)^{-p/2}\left|\boldsymbol{A}\right|^{-1}\left|\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}}\right|^{-1/2}\exp\left(-\frac{1}{2}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k)'(\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}})^{-1}(\boldsymbol{x}_i-\bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k)\right) \\
&= \left|\boldsymbol{A}\right|^{-1} f(\boldsymbol{x}_i|\bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k, \boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{x},\text{pooled}}).
\end{aligned}
$$

The pdf of the multivariate normal of the transformed data $\boldsymbol{Y}$ is now expressed as a product of a constant and the pdf of the multivariate normal of the original data $\boldsymbol{X}$. This new expression can be used to rewrite the objective function $H(\boldsymbol{w}|\boldsymbol{y})$ as a constant term plus

$H(\boldsymbol{w}|\boldsymbol{x})$. The objective function $H(\boldsymbol{w}|\boldsymbol{y})$ can then be expressed as

$$\begin{aligned}
H(\boldsymbol{w}|\boldsymbol{y}) &= -\sum_{k=1}^{K}\sum_{i\in C_k}\log\Big(\pi_k^{(\boldsymbol{w})} f(\boldsymbol{y}_i|\bar{\boldsymbol{y}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{y,\text{pooled}}^{(\boldsymbol{w})})\Big) \\
&= -\sum_{k=1}^{K}\sum_{i\in C_k}\log\Big(\pi_k^{(\boldsymbol{w})}|\boldsymbol{A}|^{-1} f(\boldsymbol{x}_i;\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{\boldsymbol{x},\text{pooled}}^{(\boldsymbol{w})})\Big) \\
&= -\sum_{k=1}^{K}\sum_{i\in C_k}\log\big(|\boldsymbol{A}|^{-1}\big) - \sum_{k=1}^{K}\sum_{i\in C_k}\log\Big(\pi_k^{(\boldsymbol{w})} f(\boldsymbol{x}_i;\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{\boldsymbol{x},\text{pooled}}^{(\boldsymbol{w})})\Big) \\
&= K\lceil n(1-\alpha)\rceil \log(|\boldsymbol{A}|) + H(\boldsymbol{w}|\boldsymbol{x}).
\end{aligned}$$

Since the objective function $H(\boldsymbol{w}|\boldsymbol{y})$ is expressed in terms of $H(\boldsymbol{w}|\boldsymbol{x})$ and is changed by a constant, the affine-invariant property of the objective function is preserved when the pooled covariance is used. Therefore, if $\boldsymbol{w}$ is a minimizer of the objective function $H(\boldsymbol{w}|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)$ with the pooled covariance $\boldsymbol{S}_{\boldsymbol{x},\text{pooled}}$, then $\boldsymbol{w}$ is a minimizer of $H(\boldsymbol{w}|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n)$ with the pooled covariance $\boldsymbol{S}_{\boldsymbol{y},\text{pooled}}$.

6.1 Derivative of the Objective Function

The objective function in Equation 6.1 can be expressed similar to what we had before (see Equation (2.15)), except that this is in terms of the pooled covariance.

$$\begin{aligned}
H(\boldsymbol{w}) &= -\sum_{k=1}^{K}\sum_{i\in C_k}\log\Big(\pi_k^{(\boldsymbol{w})} f(\boldsymbol{x}_i;\bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{\text{pooled}}^{(\boldsymbol{w})})\Big) \\
&= -\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\Big(n_k^{(\boldsymbol{w})}\Big) + \frac{1}{2}\sum_{k=1}^{K} n_k^{(\boldsymbol{w})}\log\Big(\Big|\boldsymbol{S}_{\text{pooled}}^{(\boldsymbol{w})}\Big|\Big) + \\
&+ \frac{1}{2}(\lceil n(1-\alpha)\rceil p - Kp) + \lceil n(1-\alpha)\rceil\log(\lceil n(1-\alpha)\rceil) - \lceil n(1-\alpha)\rceil\log\big((2\pi)^{-p/2}\big) \\
&= -(I_f + C(n,p,K)),
\end{aligned}$$

where

$$I_f = \sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \log\Big(n_k^{(\boldsymbol{w})}\Big) - \frac{1}{2}\sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \log\Big(\Big|\boldsymbol{S}_{\text{pooled}}^{(\boldsymbol{w})}\Big|\Big)$$
$$= \sum_{k=1}^{K} n_k^{(\boldsymbol{w})} \log\left(n_k^{(\boldsymbol{w})} \Big|\boldsymbol{S}_{\text{pooled}}^{(\boldsymbol{w})}\Big|^{-\frac{1}{2}}\right),$$

and

$$C(n,p,K) = -\frac{1}{2}(\lceil n(1-\alpha)\rceil p - Kp) - \lceil n(1-\alpha)\rceil \log(\lceil n(1-\alpha)\rceil)$$
$$- \frac{\lceil n(1-\alpha)\rceil p}{2}\log(2\pi).$$

The gradient of the objective function is obtained by differentiating the objective function with respect to $\boldsymbol{w}_{ik}$. Again, since $C(n,p,K)$ is a constant its derivative is zero, i.e., $\frac{\partial(C(n,p,K))}{\partial \boldsymbol{w}_{ik}} = 0$. Next, we need to get the derivative of the remaining term I_f. By following the differentiating of I_f in Chapter 2,

$$\frac{\partial\,(I_f)}{\partial w_{ik}} = 1 + \log\Big(n_k^{(\boldsymbol{w})}\Big) - \frac{1}{2}\log\Big(\Big|\boldsymbol{S}_{\text{pooled}}^{(\boldsymbol{w})}\Big|\Big) - \frac{1}{2}\frac{n_k^{(\boldsymbol{w})}}{\sum_{k=1}^{K}(n_k^{(\boldsymbol{w})}-1)} D^2(\boldsymbol{x}_i; \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{\text{pooled}}^{(\boldsymbol{w})}) +$$
$$+ \frac{1}{2}\frac{n_k^{(\boldsymbol{w})} p}{\sum_{k=1}^{K}(n_k^{(\boldsymbol{w})}-1)}.$$

Therefore,

$$\frac{\partial\,(H(\boldsymbol{w}))}{\partial w_{ik}} = -\frac{\partial\,(I_f)}{\partial w_{ik}} + \frac{\partial\,(C(n,p,k))}{\partial w_{ik}}$$
$$= -\Big(1 + \log\Big(n_k^{(\boldsymbol{w})}\Big) - \frac{1}{2}\log\Big(\Big|\boldsymbol{S}_{\text{pooled}}^{(\boldsymbol{w})}\Big|\Big) - \frac{1}{2}\frac{n_k^{(\boldsymbol{w})}}{\sum_{k=1}^{K}(n_k^{(\boldsymbol{w})}-1)} D^2(\boldsymbol{x}_i; \bar{\boldsymbol{x}}_k^{(\boldsymbol{w})}, \boldsymbol{S}_{\text{pooled}}^{(\boldsymbol{w})}) +$$
$$+ \frac{1}{2}\frac{n_k^{(\boldsymbol{w})} p}{\sum_{k=1}^{K}(n_k^{(\boldsymbol{w})}-1)}\Big).$$

6.2 Example

In the examples below, we generated several datasets from Gaussian distribution with different configurations, i.e., location and scale parameters, and number of clusters. The dataset generation models are given below in Subsection 6.2.1. We then created other versions of

the generated datasets via a random affine transformation, and random contamination. We apply all methods (tk-means, `tclust` and k-dets) on these datasets for a total of 100 replications and the average results are reported. In Subsection 6.2.2, we analyzed the glass vessels datasets to benchmark the equal variance assumption.

6.2.1 Simulated Data

N1 We simulated a dataset with two clusters, each of size 10. The dataset is bivariate. We applied the clustering methods on this dataset and also on an affine transformation of this dataset. A plot of the data and its transformation is given in Figure 6.1. The average results are presented in Tables 6.1 and 6.2. After applying all the methods on this dataset, k-means had the least performance results for all metrics used and `tclust` had the least performance results for the transformed dataset. Once again, the affine-invariance property of our proposed method is evident as our clustering results is unchanged for both datasets.

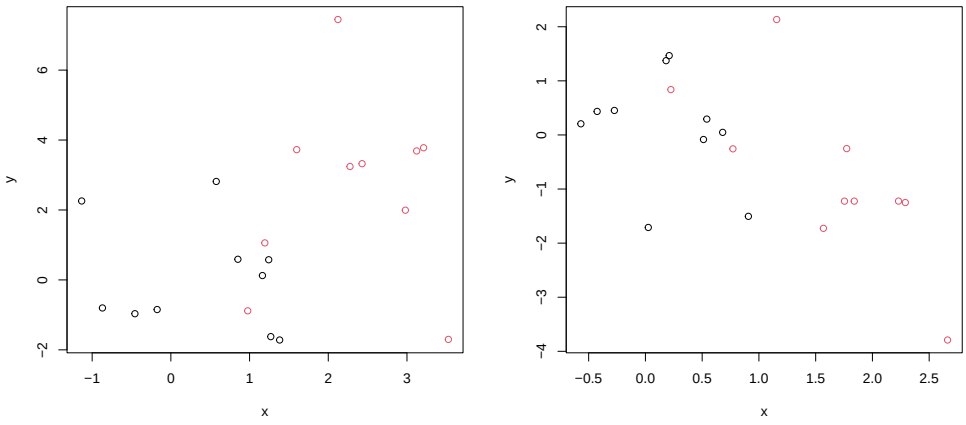

Figure 6.1: Simulated clusters with equal covariance (right panel is a random transformation).

N2 ***Two elliptical clusters of same shapes and sizes***:

Table 6.1: ACA, ARI, AP and ANMI comparison for k-means, `tclust` and k-dets on simulated data

Methods	ACA	ARI	AP	ANMI
k-means	0.750	0.605	0.750	0.192
`tclust`	0.810	0.689	0.810	0.424
k-dets	0.880	0.779	0.880	0.576

Table 6.2: ACA, ARI, AP and ANMI comparison for k-means, `tclust` and k-dets on transformed simulated data

Methods	ACA	ARI	AP	ANMI
k-means	0.800	0.663	0.800	0.300
`tclust`	0.715	0.592	0.715	0.216
k-dets	0.880	0.779	0.880	0.576

The mean vectors for the two clusters are $\boldsymbol{\mu}_1 = (0,0)'$ and $\boldsymbol{\mu}_2 = (0,5)'$. Both clusters have the same covariance $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \begin{pmatrix} 13 & -7 \\ -7 & 5 \end{pmatrix}$. The dataset N2 comprises 300 observations where each cluster contains half of the observations. We created two versions of this dataset by introducing 5% and 10% background noise into N2. We take random affine transformations of these three datasets and the transformed datasets are labeled N2.T, N2.T05 (dataset with 5% contamination) and N2.T10. We apply all three methods on these datasets and report the average results in Table 6.3.

N3 ***Two spherical clusters of same shapes and sizes***:

The mean vector for cluster 1 is $\boldsymbol{\mu_1} = (0,0,0,0,0)^T$ and the mean vector for cluster 2 is $\boldsymbol{\mu}_1 = (-5,5,-5,5,-5)^T$. The two clusters have the same covariance $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = 50\boldsymbol{I}_{5\times 5}$. Each cluster has 200 observations. As in the case of dataset N2, two versions of N3 are created by introducing 5% and 10% background noise. We then take random

Table 6.3: ARI, AP and ANMI comparison for synthetic data.

	N2.T			N2.T05			N2.T10		
Methods	ARI	AP	ANMI	ARI	AP	ANMI	ARI	AP	ANMI
tk-means	0.584	0.707	0.127	0.607	0.695	0.201	0.630	0.693	0.244
`tclust`	0.620	0.746	0.185	0.649	0.748	0.316	0.678	0.754	0.391
k-dets	0.967	0.983	0.878	0.953	0.974	0.853	0.956	0.973	0.863

affine transformations of these three datasets and label these datasets N3.T, N3.T05 (dataset with 5% contamination) and N3.T10. We apply all three methods on these datasets and report the average results in Table 6.4.

Table 6.4: ARI, AP and ANMI comparison for synthetic data.

	N3.T			N3.T05			N3.T10		
Methods	ARI	AP	ANMI	ARI	AP	ANMI	ARI	AP	ANMI
tk-means	0.499	0.512	0.000	0.546	0.546	0.204	0.589	0.566	0.297
`tclust`	0.985	0.992	0.937	0.985	0.992	0.944	0.987	0.992	0.953
k-dets	0.990	0.995	0.955	0.949	0.968	0.866	0.947	0.962	0.871

6.2.2 Real Data

- ***Glass Vessels Dataset***:

  The glass vessels dataset (Janssens et al., 1998; Varmuza and Filzmoser, 2016) is a built-in dataset available in the `chemometrics` package in `R`. The dataset is made up of 13 different measurements on 180 archaeological glass vessels. There are four classes in this dataset of sizes 145, 15, 10 and 10. We considered this dataset because three of the clusters have small counts. The average results are reported in Table 6.5.

Table 6.5: ACA, ARI, AP and ANMI comparison for k-means, `tclust` and k-dets on glass vessels dataset

Methods	ACA	ARI	AP	ANMI
k-means	0.540	0.662	0.540	0.661
`tclust`	0.555	0.663	0.555	0.661
k-dets	0.666	0.720	0.666	0.645

Chapter 7

High-Dimensional Data

Recent datasets emerging from several fields including, but not limited to, medicine and healthcare are high-dimensional which need thorough analysis to obtain very useful information. A dataset containing numerous variables on which observations are measured is termed high-dimensional. What number of variables makes a data high dimensional? There is no specific threshold on the amount of variables. Usually this large number can be carefully selected by the researcher and also varies depending on the field of research. Ideally a high-dimensional dataset is one that presents several computational challenges. As a result, the "curse of dimensionality" has been an open research problem and algorithms should take into consideration the problems associated with high dimensional datasets.

Visualization of datasets is very important in any statistical analysis we perform. Many hidden patterns and information can be unlocked through data visualization. However, many dependable statistical tools that are available can visualize data to at most 3D making it impossible to visualize high-dimensional datasets. The question of whether more variables measured on an observation helps improve analysis of the data is still an open question. However, it is likely that more noise may be added to the dataset with an increase in the number of variables. Also, increasing the number of variables increases the dimensionality and it is possible that the dataset may become sparse. It would be easier to redistribute data points into clusters if all variables in the datasets contribute to identifying observations that are similar to each other and dissimilar from other observations. However, it is even not the case for datasets that are not high-dimensional. Some of the variables in high-dimensinoal datasets may not contribute at all to identifying similar objects. It is also likely that most of the variables may be correlated which may cause "improvised" clusters.

7.1 Regularized Covariances

We equipped our proposed clustering method with high-dimensional capabilities by using a regularization technique reminiscent of that proposed by Boudt et al. (2020). Following the proposal in this sequel, we will replace the covariance $\boldsymbol{S}_k^{(\boldsymbol{w})}$ for each cluster by a regularized covariance estimate $\boldsymbol{S}_{\text{reg},k}^{(\boldsymbol{w})}$ which is defined in terms of cluster covariance $\boldsymbol{S}_k^{(\boldsymbol{w})}$, a target matrix $\boldsymbol{T}_k$ and a regularization parameter ρ_k:

$$\boldsymbol{S}_{\text{reg},k}^{(\boldsymbol{w})} = \rho_k \boldsymbol{T}_k + (1-\rho_k)\boldsymbol{S}_k^{(\boldsymbol{w})}. \tag{7.1}$$

The regularized covariance is a convex combination of $\boldsymbol{S}_k^{(\boldsymbol{w})}$ and $\boldsymbol{T}_k$, a predetermined symmetric positive definite target matrix. Note that these two matrices have the same dimension. The regularization parameter is allowed to belong in the interval $[0, 1]$. When $\rho_k = 0$, $\boldsymbol{S}_{\text{reg},k}^{(\boldsymbol{w})}$ is simply $\boldsymbol{S}_k^{(\boldsymbol{w})}$, the covariance matrix of each cluster and when $\rho_k = 1$, $\boldsymbol{S}_{\text{reg},k}^{(\boldsymbol{w})}$ reduces to the target matrix $\boldsymbol{T}_k$. Spectral decomposition permits us to decompose the target matrix into matrices containing its eigenvalues and corresponding eigenvectors:

$\boldsymbol{T}_k = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}'$

$\boldsymbol{\Lambda}$: a diagonal matrix containing eigenvalues of $\boldsymbol{T}_k$,

$\boldsymbol{Q}$: an orthogonal matrix containing corresponding eigenvectors of $\boldsymbol{T}_k$.

To guide our choice of target matrix $\boldsymbol{T}_k$, a symmetric positive definite matrix that is well-conditioned can be used. A few choices are given. For a diagonal matrix, an identity matrix can be used, also a diagonal matrix whose diagonal elements are robust scale estimates can be used. For a non-diagonal matrix, a rank correlation matrix of the data can be used (Boudt et al., 2020). Based on the decomposition of the target matrix, the regularized covariance can be expressed as

$$\boldsymbol{S}_{\text{reg},k}^{(\boldsymbol{w})} = \boldsymbol{Q}\boldsymbol{\Lambda}^{1/2}[\rho_k \boldsymbol{I} + (1-\rho_k)\boldsymbol{S}_{\boldsymbol{U},k}^{(\boldsymbol{w})}]\boldsymbol{\Lambda}^{1/2}\boldsymbol{Q}' \tag{7.2}$$

where $\boldsymbol{S}_{\boldsymbol{U},k}^{(\boldsymbol{w})} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}'\boldsymbol{S}_k^{(\boldsymbol{w})}\boldsymbol{Q}\boldsymbol{\Lambda}^{-1/2}$. The precision matrix of the regularized covariance matrix is then

$$(\boldsymbol{S}_{\text{reg},k}^{(\boldsymbol{w})})^{-1} = \boldsymbol{Q}\boldsymbol{\Lambda}^{-1/2}[\rho_k \boldsymbol{I} + (1-\rho_k)\boldsymbol{S}_{\boldsymbol{U},k}^{(\boldsymbol{w})}]^{-1}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}'. \tag{7.3}$$

The gradient of the objective function involves a term that computes the squared Mahalanobis distances. This distance metric requires the inverse of the covariance matrix in its computation. Equation (7.3) is necessary for computing the gradient of the objective function. However, for high dimensional data especially when $p \gg n$, taking the inverse of $\boldsymbol{S}^{(\boldsymbol{w})}_{\text{reg},k}$ implies taking the inverse of a $p \times p$ matrix which is computationally expensive. On the strength of the Sherman-Morrison-Woodbury identity (Sherman and Morrison, 1950; Woodbury, 1950; Bartlett, 1951), a more computationally efficient way to invert $\boldsymbol{S}^{(\boldsymbol{w})}_{\text{reg},k}$ is put forth in Equation (7.4) for any $n_k < p$. Let

$$\begin{aligned}
\boldsymbol{B} &= \rho_k \boldsymbol{I} + (1-\rho_k)\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{U},k} \\
&= \rho_k \boldsymbol{I} + (1-\rho_k)\boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}'\boldsymbol{S}^{(\boldsymbol{w})}_{k}\boldsymbol{Q}\boldsymbol{\Lambda}^{-1/2} \\
&= \rho_k \boldsymbol{I} + (1-\rho_k)\boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}'\frac{\boldsymbol{Z}'\boldsymbol{Z}}{n^{(\boldsymbol{w})}_k - 1}\boldsymbol{Q}\boldsymbol{\Lambda}^{-1/2} \\
&= \rho_k \boldsymbol{I} + (1-\rho_k)\boldsymbol{U}\boldsymbol{V},
\end{aligned}$$

where $\boldsymbol{U} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}'\frac{\boldsymbol{Z}'}{n^{(\boldsymbol{w})}_k - 1}$, $\boldsymbol{V} = \boldsymbol{Z}\boldsymbol{Q}\boldsymbol{\Lambda}^{-1/2}$ and $\boldsymbol{Z} = (\boldsymbol{X}_k - \bar{\boldsymbol{x}}^{(\boldsymbol{w})}_k)$ with $\boldsymbol{X}_k$ containing the observations of the k-th cluster. Then by applying the Sherman-Morrison-Woodbury identity, we have,

$$\begin{aligned}
\boldsymbol{B}^{-1} &= \left[\frac{1}{\rho_k}\boldsymbol{I}_p - \frac{1}{\rho_k}\boldsymbol{I}_p\frac{(1-\rho_k)}{n^{(\boldsymbol{w})}_k - 1}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}\boldsymbol{Z}'\left(\boldsymbol{I}_{n_k} + \boldsymbol{Z}\boldsymbol{Q}\boldsymbol{\Lambda}^{-1/2}\frac{1}{\rho_k}\boldsymbol{I}_p\frac{(1-\rho_k)}{n^{(\boldsymbol{w})}_k - 1}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}'\boldsymbol{Z}'\right)^{-1} \times \right. \\
&\quad \left. \times \boldsymbol{Z}\boldsymbol{Q}\boldsymbol{\Lambda}^{-1/2}\frac{1}{\rho_k}\boldsymbol{I}_p\right] \\
&= \left[\frac{1}{\rho_k}\boldsymbol{I}_p - \frac{1}{\rho_k^2}\frac{(1-\rho_k)}{n^{(\boldsymbol{w})}_k - 1}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}\boldsymbol{Z}'\left(\boldsymbol{I}_{n_k} + \frac{1}{\rho_k}\frac{(1-\rho_k)}{n^{(\boldsymbol{w})}_k - 1}\boldsymbol{Z}\boldsymbol{Q}\boldsymbol{\Lambda}^{-1}\boldsymbol{Q}'\boldsymbol{Z}'\right)^{-1}\boldsymbol{Z}\boldsymbol{Q}\boldsymbol{\Lambda}^{-1/2}\right].
\end{aligned}$$

Thus, the precision matrix of the regularized covariance of the k-th cluster is given as

$$(\boldsymbol{S}^{(\boldsymbol{w})}_{\text{reg},k})^{-1} = \boldsymbol{Q}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{B}^{-1}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}'. \tag{7.4}$$

Equation (7.4) is computationally efficient as compared to Equation (7.3) owing to the fact that a $p \times p$ matrix is not inverted but rather an $n_k \times n_k$ matrix which is of a lower dimension.

For any non-singular $\boldsymbol{A} \in \mathbb{R}^{p\times p}$, and any q-norm, the condition number of $\boldsymbol{A}$ is defined as the product of the q-th norm of $\boldsymbol{A}$ and the q-th norm of $\boldsymbol{A}^{-1}$ as in Equation (7.5). If $\boldsymbol{A}$ is high-dimensional, computing its inverse can be computationally prohibitive. Oftentimes, $\boldsymbol{A}^{-1}$ may not even exist. Consequently, we express the condition number of $\boldsymbol{A}$ in terms of its eigenvalues by the definition of matrix norms and eigenvalue equation:

$$\begin{aligned}
\kappa(\boldsymbol{A}) &= \|\boldsymbol{A}\|\left\|\boldsymbol{A}^{-1}\right\| \\
&= \max_{\boldsymbol{x}\neq 0}\frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \times \max_{\boldsymbol{x}\neq 0}\frac{\left\|\boldsymbol{A}^{-1}\boldsymbol{x}\right\|}{\|\boldsymbol{x}\|} \\
&= \max_{\boldsymbol{x}\neq 0}\frac{\|\boldsymbol{\lambda}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \times \max_{\boldsymbol{x}\neq 0}\frac{\left\|\boldsymbol{\lambda}^{-1}\boldsymbol{x}\right\|}{\|\boldsymbol{x}\|} \\
&= \max_{\boldsymbol{x}\neq 0}\frac{|\boldsymbol{\lambda}|\|\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \times \max_{\boldsymbol{x}\neq 0}\frac{\left|\boldsymbol{\lambda}^{-1}\right|\|\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \\
&= \max\left\{|\boldsymbol{\lambda}|\right\} \times \max\left\{\left|\boldsymbol{\lambda}^{-1}\right|\right\} \\
&= \frac{\max\left\{|\boldsymbol{\lambda}|\right\}}{\min\left\{|\boldsymbol{\lambda}|\right\}}.
\end{aligned} \tag{7.5}$$

Note here that the absolute value of the eigenvalues are taken before the maximum and minimum eigenvalues are selected because $\boldsymbol{A}$ is not assumed positive definite.

A bound on the condition number of the regularized covariance matrix is necessary to ensure the matrix is well-conditioned (Won et al., 2013), i.e., the condition number is not bigger than some threshold κ_c. Since $\boldsymbol{T}_k$, $\boldsymbol{Q}$ and $\boldsymbol{\Lambda}$ are all fixed, the eigenvalues of $\boldsymbol{S}^{(\boldsymbol{w})}_{\text{reg},k}$ equalling $\rho_k + (1-\rho_k)\lambda_i$, $i = 1,\ldots,p$, where $\boldsymbol{\lambda} = (\lambda_1,\ldots,\lambda_p)$ holds the eigenvalues of $\boldsymbol{S}^{(\boldsymbol{w})}_{\boldsymbol{U},k}$ suffices. Accordingly, following Equation (7.5), the condition number for each cluster regularized covariance is

$$\kappa(\boldsymbol{S}^{(\boldsymbol{w})}_{\text{reg},k}) = \frac{\rho_k + (1-\rho_k)\max\left\{\boldsymbol{\lambda}\right\}}{\rho_k + (1-\rho_k)\min\left\{\boldsymbol{\lambda}\right\}}. \tag{7.6}$$

Following the proposal or recommendation of Boudt et al. (2020), for each regularized covariance matrix, find a minimum value of ρ_k which is non-negative for which $\kappa(\boldsymbol{S}^{(\boldsymbol{w})}_{\text{reg},k}) \leq \kappa_c$, where κ_c is chosen to be 50. We propose an algorithm named `RegCov` for the regularized covariance similar to Boudt et al. (2020). The underlying difference between our proposed

and that of Boudt et al. (2020) is that we do not standardize the p variables as standardization may destroy the cluster structures. If a cluster covariance $\boldsymbol{S}_k^{(\boldsymbol{w})}$ is not well-conditioned, `RegCov` (Algorithm 5) will be used to obtain a regularized cluster covariance. On the other hand, if a cluster covariance $\boldsymbol{S}_k^{(\boldsymbol{w})}$ is well-conditioned, ρ_k is set to null and no regularization of the covariance is needed.

Algorithm 5: *RegCov*$(\boldsymbol{S}_k^{(\boldsymbol{w})})$

1 Choose a target matrix $\boldsymbol{T}_k$ and decompose as $\boldsymbol{T}_k = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}'$

2 Compute $\boldsymbol{S}_{\boldsymbol{U},k}^{(\boldsymbol{w})} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{Q}'\boldsymbol{S}_k^{(\boldsymbol{w})}\boldsymbol{Q}\boldsymbol{\Lambda}^{-1/2}$

3 Find the smallest value of $\rho_k \in (0, 1]$ for which $\kappa(\boldsymbol{S}_{\text{reg},k}^{(\boldsymbol{w})}) \leq \kappa_c$

4 Compute the regularized covariance $\boldsymbol{S}_{\text{reg},k}^{(\boldsymbol{w})}$

Algorithm 6: k-dets$(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n; K, \alpha)$

1 Generate, say, $n_{\text{rep}} = 100$ random warmstarts, i.e.,
$\mathcal{C}^0 = (\mathcal{C}_1^0, \ldots, \mathcal{C}_K^0), \ldots, \mathcal{C}^{n_{\text{rep}}} = (\mathcal{C}_1^{n_{\text{rep}}}, \ldots, \mathcal{C}_K^{n_{\text{rep}}})$, using Algorithm 3 and obtain the initial weight matrix $\boldsymbol{w} \in [0, 1]^{n \times K}$ from the initial partition.

2 For $m = 0, 1, \ldots,$ iterate the concentration step:

2.1 If any $\boldsymbol{S}_k^{(\mathcal{C}^{(m)})}$ is not well-conditioned, compute $\boldsymbol{S}_{\text{reg},k}^{(\mathcal{C}^{(m)})}$ using Algorithm 5.

2.2 Compute the gradient at $\boldsymbol{w}^{(m)}$ using $\boldsymbol{S}_{\text{reg},k}^{(\boldsymbol{w})}$.

2.3 Update the weight matrix $\boldsymbol{w}^{(m+1)}$ using Algorithm 2 leaving $n\alpha$ points unassigned.

2.4 Form a new partition $\mathcal{C}^{(m+1)} = (\mathcal{C}_1^{(m+1)}, \ldots, \mathcal{C}_K^{(m+1)})$ from $\boldsymbol{w}^{(m+1)}$ and compute objective value $H(\mathcal{C}^{(m)})$.

2.5 Stop as soon as $H(\mathcal{C}^{(m)}) = H(\mathcal{C}^{(m+1)})$ or $\mathcal{C}^{(m)} = \mathcal{C}^{(m+1)}$.

3 Starting Step 2 from each warmstart, select final partition with the smallest value of objective $H(\mathcal{C})$.

7.2 Example

7.2.1 Synthetic Data

- We simulated a data with 60 observational units and 300 features. This dataset contains three groups each with 20 observations. The `tclust` method was not considered in this comparison due to some unexpected errors in results. The best clustering for each of the methods is presented in Figure 7.1 and the average results are given in Table 7.1.

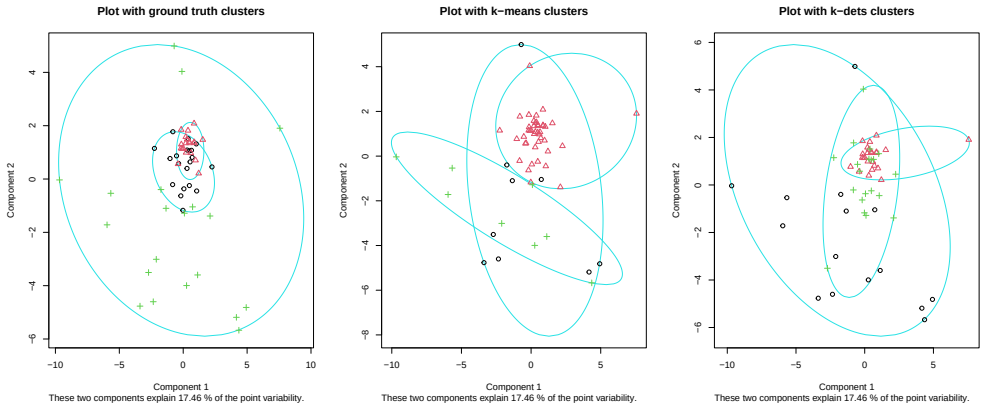

Figure 7.1: Simulated clusters with equal covariances.

Table 7.1: ACA, ARI, AP and ANMI comparison for k-means and k-dets

Methods	ACA	ARI	AP	ANMI
k-means	0.483	0.637	0.483	0.488
k-dets	0.850	0.824	0.850	0.626

7.2.2 Real Datasets

- ***Phenyl Dataset***

  The Phenyl dataset accompanies the `chemometrics` package in `R`. The dataset is made up of mass spectra obtained from 600 chemical compounds. The mass spectra have

been transformed into 658 variables representing various spectra features. Among these 600 compounds, one-half of them contain a phenyl substructure and one-half do not have this substructure. Therefore, there are two subgroups present in this dataset. Again, `tclust` was not considered in this comparison since the method returns all the observations as one cluster. The average results are reported in Table 7.2.

Table 7.2: ACA, ARI, AP and ANMI comparison for k-means and k-dets on phenyl dataset

Methods	ACA	ARI	AP	ANMI
k-means	0.822	0.707	0.823	0.334
k-dets	0.822	0.707	0.823	0.334

Chapter 8

Conclusions

We proposed a clustering algorithm based on minimizing the "trimmed" negative log-likelihood function. This algorithm addresses clustering challenges faced by k-means, a popular clustering algorithm. Our proposed algorithm is robust and tends to be affine-invariant in situations where the dataset is non-sparse. We developed a concentration step, vaguely reminiscent of the classical Lloyd's algorithm, that can iteratively be used to minimize the log-likelihood objective function. Following the ideas of Pokojovy and Jobe (2022), we discovered equivalence between our proposed method and the well-known Frank-Wolfe gradient method, which, in turn, implies our algorithm converges to a local minimum of the objective function. Being a local optimization technique, our algorithm depends on the choice of initial cluster or "warmstart." As a by-product, we developed a new affine-invariant sampling procedure to draw initial partitions or warmstarts. We considered the case where the subgroups or clusters of a dataset have same covariance structure. In such situation, the pooled covariance is used instead of the individual covariance matrices. Our proposed clustering method was also extended to be applicable for high-dimensional datasets which will serve fields or disciplines that usually analyze high-dimensional datasets. Multiple real and synthetic data with different configurations, including various cluster shapes and sizes, number of clusters and types of linear transformations, are analyzed to assess the performance of our proposed algorithm in comparison to our reference clustering method, k-means, and `tclust`. Empirical results strongly indicate that our proposed cluster algorithm is oftentimes better or head-to-head to the reference methods and is computationally attractive alternative to the conventional k-means method.

Chapter 9

Future Research

- **Regularization for low-dimensional setting**.

  So far, we performed regularization to obtain well-conditioned covariance matrices for high-dimensional or sparse datasets. For multivariate (non-sparse and specifically for $n > p$), it is still possible for cluster covariance matrices to become ill-conditioned. To this end, we will extend the idea of regularization to all kinds of datasets.

- **Equal covariance assumption for high-dimensional setting**.

  So far, we assume equal covariance for multivariate datasets ($n > p$) with small n. For high-dimensional or sparse datasets with small n, we will also assume equal covariances.

- **Choosing optimal k**.

  The choice of number of clusters, k, is an important decision to make in cluster analysis. For datasets with cluster labels, k is known apriori. However, most dataset do not have cluster labels which makes clustering an unsupervised learning. Thus, we will develop a method to help us decide on what k to choose for our analysis.

- **Functional data clustering**.

  Functional datasets are encountered in several fields of study, such as engineering, medicine and healthcare, and economics. Each data point of the various synthetic and real datasets we have used in our simulation studies is an observation with some recorded features. Functional data points, on the other hand, are functions representing curves or surfaces belonging to an infinite-dimensional space. Infinite dimensionality is a major source of difficulties in analyzing functional datasets. Possible approaches to

solving this problem include approximating the curves or surfaces in a finite-dimension space and then applying cluster algorithms to finite-dimensional approximations.

- **Extend to co-clustering**

  Co-clustering, also referred to as bi-clustering, serves as a data analysis technique with the objective of clustering both the rows and columns of a given sample data matrix. Unlike conventional clustering methods, co-clustering identifies subsets of rows and columns that demonstrate comparable patterns or relationships. By simultaneously clustering both dimensions, co-clustering has the ability to uncover cohesive subgroups or co-occurring patterns that are unique to specific subsets of rows and columns. Co-clustering proves particularly beneficial when the data exhibit a block-like structure, signifying the presence of subsets of rows and columns with similar characteristics. This situation may arise, for instance, in gene expression analysis, where genes might exhibit co-regulation under specific conditions, or in text mining, where documents could be associated with specific topics.

References

Abbasi, A. A. and Younis, M. (2007). A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30(14-15):2826–2841.

Andreopoulos, B., An, A., Wang, X., and Schroeder, M. (2009). A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefings in Bioinformatics*, 10(3):297–314.

Ankerst, M., Breunig, M., Kriegel, H., Ng, R., and Sander, J. (2008). Ordering points to identify the clustering structure. In *Proc. ACM SIGMOD*, volume 99.

Anum, A. T. (2021). A new algorithm for robust affine-invariant clustering. Master's thesis, The University of Texas at El Paso.

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

Bartlett, M. S. (1951). An inverse matrix adjustment arising in discriminant analysis. *The Annals of Mathematical Statistics*, 22(1):107–111.

Boudt, K., Rousseeuw, P. J., Vanduffel, S., and Verdonck, T. (2020). The minimum regularized covariance determinant estimator. *Statistics and Computing*, 30(1):113–128.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: series B (methodological)*, 39(1):1–38.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188.

Forgy, E. W. (1965). Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21:768–769.

Fraley, C. and Raftery, A. E. (1998). How many clusters? Which clustering method? Answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588.

Frank, M., Wolfe, P., et al. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

Fritz, H., Garcia-Escudero, L. A., and Mayo-Iscar, A. (2012). tclust: An R package for a trimming approach to cluster analysis. *Journal of Statistical Software*, 47:1–26.

García-Escudero, L. A., Gordaliza, A., Matrán, C., and Mayo-Iscar, A. (2008). A general trimming approach to robust cluster analysis. *The Annals of Statistics*, 36(3):1324–1345.

García-Escudero, L. A., Gordaliza, A., Matrán, C., and Mayo-Iscar, A. (2010). A review of robust clustering methods. *Advances in Data Analysis and Classification*, 4(2-3):89–109.

Huang, H.-H. and Yang, J. (2020). Affine-transformation invariant clustering models. *Journal of Statistical Distributions and Applications*, 7(1):1–24.

Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pages 427–435. PMLR.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666.

Janssens, K. H., Deraedt, I., Schalm, O., and Veeckman, J. (1998). Composition of 15–17th century archaeological glass vessels excavated in Antwerp, Belgium. *Mikrochimica Acta 15 (Suppl.)*, pages 253–267.

Kalyani, P. (2012). Approaches to partition medical data using clustering algorithms. *International Journal of Computer Applications*, 49(23).

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley, Canada.

Khanmohammadi, S., Adibeig, N., and Shanehbandy, S. (2017). An improved overlapping k-means clustering method for medical applications. *Expert Systems with Applications*, 67:12–18.

Knorr, E. M., Ng, R. T., and Zamar, R. H. (2001). Robust space transformations for distance-based operations. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 126–135.

Kriegel, H.-P., Kröger, P., Sander, J., and Zimek, A. (2011). Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240.

Kumar, M. and Orlin, J. B. (2008). Scale-invariant clustering with minimum volume ellipsoids. *Computers & Operations Research*, 35(4):1017–1029.

Lacoste-Julien, S. (2016). Convergence rate of Frank-Wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*.

Landau, S., Leese, M., Stahl, D., and Everitt, B. S. (2011). *Cluster Analysis*. John Wiley & Sons.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Symposium on Mathematics and Probability, 5th, Berkeley*, volume 1, pages 281–297. University of California Press, Berkeley, CA.

Manning, C. D. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Masood, M. A. and Khan, M. (2015). Clustering techniques in bioinformatics. *IJ Modern Education and Computer Science*, 1:38–46.

McLachlan, G. J. and Krishnan, T. (1997). *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics. John Wiley & Sons.

Nielsen, F. (2016). Hierarchical clustering. In *Introduction to HPC with MPI for Data Science*, pages 195–211. Springer.

Pena, J. M., Lozano, J. A., and Larranaga, P. (1999). An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040.

Pokojovy, M. and Jobe, J. M. (2022). A robust deterministic affine-equivariant algorithm for multivariate location and scatter. *Computational Statistics & Data Analysis*, 172:107475.

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

Rousseeuw, P. J. and Driessen, K. V. (1999). A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223.

Scrucca, L., Fop, M., Murphy, T. B., and Raftery, A. E. (2016). mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):289.

Sherman, J. and Morrison, W. J. (1950). Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127.

Strehl, A. and Ghosh, J. (2002). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617.

Varmuza, K. and Filzmoser, P. (2016). *Introduction to Multivariate Statistical Analysis in Chemometrics*. CRC Press `https://doi.org/10.1201/9781420059496`.

Won, J. H., Lim, J., Kim, S. J., and Rajaratnam, B. (2013). Condition-number-regularized covariance estimation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):427–450.

Woodbury, M. (1950). Inverting modified matrices. *Memo. Rep*, 42:106.

Xu, R. and Wunsch, D. C. (2010). Clustering algorithms in biomedical research: A review. *IEEE Reviews in Biomedical Engineering*, 3:120–154.

Appendix A

R Code

A.1 Objective Function

```
###OBJECTIVE FUNCTION
Obj.Hw <- function(x.data, w.matrix){
    n <- nrow(x.data)
    d <- ncol(x.data)
    K <- ncol(w.matrix)
    C <- -0.5*(n*d - K*d) - n*log(n) - 0.5*d*n*log(2*pi)
    n.k <- colSums(w.matrix)
    logdets <- matrix(rep(0, K), nrow = 1, ncol = K)
    I.1 <- n.k * log(n.k)
    for (m in 1:K) {
        logdets[m] <- unlist(determinant(n1ccov(x.data,
            w.matrix[,m], method = "unbiased")$cov,
            logarithm = TRUE)$modulus )[1]
    }
    I.2 <- n.k * logdets
    Hw <- -(sum(I.1) - 0.5*sum(I.2) + C)
    return(Hw)
}
```

A.2 Weight Matrix Code

```
###WEIGHT MATRIX
weight.matrix <- function(x.data, Cid.vec){
    u.Cid <- unique(Cid.vec)
    w.mat <-matrix(c(rep(0, NROW(x.data)*length(u.Cid))),
      nrow = NROW(x.data), ncol = length(u.Cid) )
    for (j in 1:ncol(w.mat)) {
        ai <- which(Cid.vec == j)
        w.mat[ai,j] <- 1
    }
        return(w.mat)
}
```

A.3 Cluster Membership

```
###CLUSTER MEMBERSHIP
clust.memb <- function(w.matrix){
    n.wm <- NROW(w.matrix)
    memb <- rep(0, n.wm)
    for (g in 1:n.wm){
        memb[g] <- which.max(w.matrix[g, ])
    }
    return(memb)
}
```

A.4 Solution of the Linear Programming Problem

```
###SOLUTION MATRIX TO THE LINEAR MINIMIZATION PROBLEM
SM <- function(grad.matrix){
    ww.new <- matrix(rep(0, NROW(grad.matrix)*NCOL(grad.matrix)),
        nrow = NROW(grad.matrix), ncol = NCOL(grad.matrix))
    if(anyNA(grad.matrix) == TRUE){
        ww.new <- matrix(NA, nrow = NROW(grad.matrix),
            ncol = NCOL(grad.matrix))
        warning("gradient matrix has NA's")
    }else{
        for (l in 1:nrow(grad.matrix)){
            ww.new[l, which.min(grad.matrix[l, ])] <- 1
        }
    }
    return(ww.new)
}
```

A.5 Performance Assessment

```
###PERFORMANCE ASSESSMENT
assessment.perf <- function(groundtruth, cluster.result){
    clust.num <- unique(cluster.result)
    nc.perm <- permutations(length(clust.num))
    max.accuracy <- .Machine$double.xmin
    for (j in 1:nrow(nc.perm)) {
        cluster.result.perm <- rep(0, length(cluster.result))
        for (i in 1:ncol(nc.perm)) {
            cluster.result.perm[which( cluster.result == i) ] <-
```

```
                nc.perm[j,i]
            }
        acc <- confusionMatrix(as.factor(groundtruth),
            as.factor(cluster.result.perm))
        cm.accuracy <- as.numeric(unlist(acc$overall)[1])
        #get the accuracy
        if(cm.accuracy >= max.accuracy){
            max.accuracy <- cm.accuracy
            fin.cluster <- cluster.result.perm
            conf.mat.inf <- acc
        }
        }
        return(list( max.accuracy = max.accuracy, fin.cluster =
            fin.cluster, conf.mat.inf = conf.mat.inf))
}
```

A.6 Weighted Covariance

```
n1ccov <- function (x, wt.k, center = TRUE, method =
    c("unbiased", "ML")){
    if (is.data.frame(x))
        x <- as.matrix(x)
    else if (!is.matrix(x))
        stop("'x' must be a matrix or a data frame")
    if (!all(is.finite(x)))
        stop("'x' must contain finite values only")
    n <- nrow(x)
```

```
    if (with.wt.k <- !missing(wt.k)) {
        if (length(wt.k) != n)
            stop("length of 'wt' must equal the number of
rows in 'x'")
        if (any(wt.k < 0) || (s <- sum(wt.k)) == 0)
            wt.k <- rep(NA, length(wt.k))
    }
    if (is.logical(center)){
        center <- if (center)
        colSums(wt.k * x)/sum(wt.k)
        else 0
    }else {
        if (length(center) != ncol(x))
            stop("length of 'center' must equal the number of
columns in 'x'")
    }
    x <- sqrt(wt.k) * sweep(x, 2, center, check.margin = FALSE)
    cov <- switch(match.arg(method), unbiased = crossprod(x)
        /(sum(wt.k) - 1), ML = crossprod(x)/sum(wt.k^2))
    y <- list(cov = cov, center = center, n.obs = n)
    if (with.wt.k)
        y$wt.k <- wt.k
    return(y)
}
```

A.7 Weighted Mean

```
ncmean <- function(x.data, w.matrix, K.k){
    nk = colSums(w.matrix)
    nk = nk[K.k]
    xbar.k <- colSums(w.matrix[,K.k]*x.data)/nk
    return(list(xbar.k = xbar.k, nk = nk))
}
```

A.8 Gradient of the Objective Function

```
##GRADIENT FUNCTION BASED ON WEIGHT MATRIX
dH_dw <- function(x.data, w.matrix, numerical_flag = FALSE){
    d <- NCOL(x.data)
    n <- NROW(x.data)
    K <- NCOL(w.matrix)
    n.k <- colSums(w.matrix)
    dH.mat <- matrix(0, nrow = n, ncol = K)
    means <- matrix(rep(0, K*d ), nrow = K, ncol = d)
    covs <- array(0.0, dim = c(d, d, K))
    invcovs <- array(0.0, dim = c(d, d, K))
    if(numerical_flag){
        h <- 1E-6
        dH.mat2 <- Obj.Hw(x.data, w.matrix)
        for (j in 1:K){
            for (i in 1:n){
                w.matrixph <- w.matrix
                w.matrixph[i,j] <- w.matrixph[i,j] + h
                dH.mat1 <- Obj.Hw(x.data, w.matrixph)
```

```
            dH.mat[i,j] <- (dH.mat1 - dH.mat2)/(h)
        }
    }
}else{
    for (q in 1:K ){
        means[q, ] <- ncmean(x.data, w.matrix, q)$xbar.k
        covs[, ,q] <- n1ccov(x.data, w.matrix[,q])$cov
        if(anyNA(covs[, ,q]) == TRUE){
            return((dH.mat[, q] <- NaN))
            break
        }
        if( rcond(covs[, ,q], norm = "I") <=
            .Machine$double.eps^0.5){
            invcovs[, ,q] <- solve( matrix(0.00001*diag(d),
                nrow = d, byrow= T))
        }else{
            invcovs[, ,q] <- solve(covs[, ,q])
        }
    }
    for (j in 1:K){
        d1.j <- mahalanobis(x.data, means[j,], invcovs[, , j],
            inverted = TRUE)
        d2.j <- d
        dH.mat[ ,j] <- -(1 + log(n.k[j])
        - 0.5*(unlist(determinant(covs[, ,j],
            logarithm = TRUE)$modulus)[1])
            -0.5*(n.k[j] / (n.k[j] - 1 ))*(d1.j - d2.j))
    }
```

```
    }
    return(dH.mat)
}
```

Curriculum Vitae

Andrews Tawiah Anum graduated from Nungua Senior High School, Ghana in 2012. He holds a Bachelor of Science (BS) degree in Mathematics and Statistics from the University of Cape Coast (UCC). Andrews was awarded the maiden Deans Award for best Mathematics and Statistics student in 2015. He followed that up by claiming an equally-prestigious Vice Chancellor's Award for best Mathematics and Statistics student a year later. He was appointed Teaching Assistant at the Mathematics and Statistics Department at the University of Cape Coast during his national service. He was involved in programs like National Mathematics Camp, Mathematics sensitization to High Schools, and Seminars, as a way of increasing students interests and knowledge in Mathematics and its applications and also brought numerous Mathematicians of diverse backgrounds together. These workshops helped him improved his interpersonal skills and challenged his creativity and Mathematical thinking. He began his graduate studies at University of Texas at El Paso in fall 2017. Andrews served as a Graduate Teaching Assistant at the Department of Mathematical Sciences while pursuing his Master's degree. During his second year in graduate school, Andrews started his thesis work titled "*Robust Statistical Inference for the Gaussian Distribution*" which was supervised by his mentor, Dr. Michael Pokojovy. After obtaining his MS degree, he was admitted into the Computational Science Program to pursue his doctoral degree. Dr. Pokojovy also supervised Andrews' present PhD dissertation which is titled "*Theoretical and Computational Aspects of Robust Cluster Analysis for Multivariate and High-Dimensional Datasets*." Andrews has accepted a tenure-track position as an Assistant Professor of Statistics at the University of New England starting fall 2023.

Email: atanum@miners.utep.edu

Website: https://sites.google.com/view/aanum/home