A Suboptimal Multi-Clustering Algorithm for Random Labeled Point Processes

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Saeideh Mogharehabed

Graduate Program in Electrical and Computer Science

The Ohio State University

2017

Master's Examination Committee:

Prof. Lori A. Dalton, Advisor

Prof. C. Emre Koksal
Copyrighted by

Saeideh Mogharehabed

2017
Abstract

Data clustering is an unsupervised learning analysis which tries to group dataset into subsets based on specified similarity criterion to clarify meaningful relations between data in each subset. Clustering is one of the most widely used techniques which can be applied in many aspects of science. The Bayes clustering algorithm is a relatively new method in the literature of cluster analysis. This method is an optimal model-based cluster analysis which provides a probabilistic framework and operates on a random labeled point process. The purpose of this study is to analyze and study a suboptimal multi-clustering algorithm based on the Bayes clustering method to decrease the computational cost. We present clustering results for synthetic data with multiple clusters which are generated by a Gaussian mixture model.
Dedication

To God Almighty who has been my source of inspiration and strength throughout my life.

To my loving husband, Ehsan, who has been a constant source of endless support and encouragement during my life in graduate school.

To my great parents and sisters for their never-ending love and spiritual support.
Acknowledgement

I would like to express my sincere gratitude to my advisor Professor Lori Dalton for her fruitful advises, patience and support during the course of this project. Her guidance helped me throughout the research and writing of this thesis.

I would also thank my husband, Ehsan, and my colleague, Ali Foroughipour, for the stimulating discussions, insightful comments and, continuous encouragement.
Vita

2010...............................................................B.S. Electrical Engineering, Isfahan University of Technology

2013...............................................................M.S. Biomedical Engineering, K. N. Toosi University of Technology

2015 to present ............................................Graduate Research Associate, Department of Electrical and Computer Science, The Ohio State University

Fields of Study

Major Field: Electrical and Computer Engineering
Table of Contents

Abstract ........................................................................................................................................... ii

Dedication ........................................................................................................................................ iii

Acknowledgement ........................................................................................................................ iv

Vita .................................................................................................................................................. v

Table of Contents ........................................................................................................................ vi

List of Figures .................................................................................................................................. viii

List of Tables ................................................................................................................................... x

Chapter 1: Introduction .................................................................................................................. 1
  1.1. Overview .................................................................................................................................. 1
  1.2. Literature review .................................................................................................................... 3
  1.3. Motivation ............................................................................................................................... 8
  1.4. Outline of thesis ..................................................................................................................... 9

Chapter 2: Bayes Clustering .......................................................................................................... 11
  2.1. Overview ................................................................................................................................ 11
  2.2.1. Label switching issue ......................................................................................................... 12
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.2. The Bayes clusterer</td>
<td>12</td>
</tr>
<tr>
<td>2.2.3. Gaussian mixture models</td>
<td>15</td>
</tr>
<tr>
<td>2.2.4. Probability mass functions PS(PS)</td>
<td>17</td>
</tr>
<tr>
<td>2.2.5. Cost function (cS)</td>
<td>19</td>
</tr>
<tr>
<td>Chapter 3: Multi-clustering Bayes Suboptimal Algorithm</td>
<td>20</td>
</tr>
<tr>
<td>3.1. Overview</td>
<td>20</td>
</tr>
<tr>
<td>3.2. Assumptions</td>
<td>20</td>
</tr>
<tr>
<td>3.3. Suboptimal algorithm</td>
<td>21</td>
</tr>
<tr>
<td>3.4. Partition Probabilities</td>
<td>26</td>
</tr>
<tr>
<td>Chapter 4: Simulations and results</td>
<td>31</td>
</tr>
<tr>
<td>4.1. Overview</td>
<td>31</td>
</tr>
<tr>
<td>4.2. Data generation</td>
<td>31</td>
</tr>
<tr>
<td>4.3. Experiments and results</td>
<td>33</td>
</tr>
<tr>
<td>Chapter 5: Conclusions and future works</td>
<td>57</td>
</tr>
<tr>
<td>Bibliography</td>
<td>59</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1 Examples of two-dimensional point sets with three clusters.......................... 2
Figure 2 Block diagram of Bayes clustering algorithm................................................. 13
Figure 3 Visualization of Dirichlet distribution for three clusters.............................. 29
Figure 4 Average empirical errors with respect to number of clusters......................... 37
Figure 5 Average run time with respect to number of clusters..................................... 37
Figure 6 Average empirical errors with respect to radius of ball search for $l = 2$ ........ 39
Figure 7 Average empirical errors with respect to radius of ball search for $l = 3$ ....... 40
Figure 8 Average empirical errors with respect to radius of ball search for $l = 4$ ....... 40
Figure 9 Average empirical errors with respect to radius of ball search for $l = 5$ ....... 41
Figure 10 Average empirical errors with respect to radius of ball search for $l = 6$ ....... 41
Figure 11 Average empirical errors with respect to $l$ for small point sets ($n=10$)........ 46
Figure 12 Average run time with respect to $l$ for small point sets ($n=10$).................. 46
Figure 13 Average empirical errors with respect to $l$ for small point sets ($n=100$)...... 47
Figure 14 Average run time with respect to $l$ for small point sets ($n=100$).............. 47
Figure 15 Average empirical errors with respect to dimension.................................... 49
Figure 16 Average run time with respect to dimension.................................................. 50
Figure 17 Average empirical errors with respect to sample size.................................. 51
Figure 18 Average run time with respect to sample size............................................. 52
Figure 19 Average empirical errors with respect to \( l \) with (false \( \psi \))................................. 53
Figure 20 Average empirical errors with respect to \( l \) with (false \( m \))................................. 54
Figure 21 Average empirical errors with respect to \( l \) with (false \( \nu \))................................. 55
Figure 22 Average empirical errors with respect to \( l \) with (false \( \kappa \))................................. 56
List of Tables

Table 1 Synthetic data clustering settings for simulations A, B, C, and F ........................ 33
Table 2 Settings for different simulations........................................................................ 34
Table 3 Results for average run times and standard deviation of run times for ............... 42
Table 4 Settings for Simulation D .................................................................................. 48
Table 5 Settings for Simulation E .................................................................................. 51
Chapter 1: Introduction

1.1. Overview

Data Clustering is an unsupervised learning analysis which tries to group data set in to subsets based on a specified similarity criterion to clarify meaningful relations between data in each subset [1]. That being said, the elements of one subset have a high degree of resemblance between themselves and low degree of resemblance with the elements of other subsets.

In cluster analysis, it is important to have compact and separated clusters in order to obtain a good performance of clustering. The clusters can have different sizes, shapes, and densities [2]. Figure 1-a illustrates three clusters with spherical shapes and equal volumes while Figure 1-b shows three clusters with different shapes, unequal volumes, and different orientation.
Clustering is one of the most widely used techniques which can be applied in many aspects of science. Various researches, in formats of papers, journals, and books, have been dedicated to the study of cluster analysis and applications of clustering. Clustering takes a vital role in various areas with specific goals such as:

- Biology for clustering genes using gene expression [5-7]
- Earth science for earthquake clustering [8]
- Computer vision for separating the parenchyma region from the chest in HRCT lung images [9]

The other applications of data clustering are found in document clustering, climate studies, information retrieval, psychology, archeology, and marketing [10 and 1].
1.2. Literature review

Generally, Clustering methods are divided into two main groups, heuristic and model-based clustering [11]. Heuristic methods cluster data based on a similarity criterion without assuming an underlying statistical model [12]. A similarity criterion measures the similarity between pairs of observations which can be a distance metric like Euclidian distance or a similarity function like dichotomous similarity function [13]. The idea of the heuristic clustering algorithm is to put similar objects to the same clusters and less similar objects to different clusters. The earliest studies introduced the cluster analysis based on the heuristic criteria. Kernighan et al. used graph theory as a heuristic criterion to partition arbitrary graphs [14]. Although heuristic methods are implemented in a fast and convenient way [15], their statistical properties are generally unresolved [19]. It means that in a heuristic algorithm we work with an objective function but we do not know how good it is or whether it works for another clustering problem or not. Actually, we do not have a broad perspective to see where the optimal stands and how far a heuristic algorithm is from the optimal answer.

Model-based clustering methods suggest a principled replacement to heuristic clustering algorithms [16]. In particular, the model-based cluster analysis solves the clustering problem in a principled probabilistic framework. In these methods, it is assumed that data were generated from a model which is a mixture of probability distributions. The purpose of these methods is to recover the underlying model, while each cluster corresponds to a component in a multivariate mixture distribution [17]. There have been a number of studies involving model-based clustering algorithms, including [12, 17, and 18].
Model-based clustering methods assume a probabilistic model for observations which allows clusters to be in different sizes, shapes, and orientations [14]. Also, in many research studies, model-based clustering methods have been shown to give clustering results with less error compared to heuristic-based methods (for instance, [11], [19], and [20]).

In the path of developing model-based methods, Dougherty et al [21] developed a probabilistic theory of clustering based on Bayes analysis. Later, Dalton et al [3] improved the framework by proposing a general analytic method for finding the Bayes clusterer together with a precise definition of error [3]. The strength of these methods is its ability to develop a theoretical framework for optimal cluster analysis along with a Bayesian decision theory and provide a definition for error, which as the result makes these methods superior to the other model-based methods.

Therefore, the main work of this thesis is based on the model-based Bayes clusterer developed by Dalton et al. [3]. Our purpose is to study and implement a suboptimal multi-clustering method which is explained in Chapter 3. This study investigates advantages, limitations, and challenges of the suboptimal multi-clustering algorithm.

The continue of this review is narrowed down to explain classical clustering algorithms which are used in this thesis such as expectation maximization (EM), fuzzy c-means (FCM), $k$-mean (KM), hierarchical. Furthermore, Bayes clustering algorithm which is the basis of this thesis is generally discussed.
1.2.1. Expectation maximization

Many researchers have proposed different clustering algorithms based on the expectation maximization algorithm [22]. EM algorithm assumes the data are generated from a Gaussian mixture model. This algorithm alternates between two main steps, expectation and maximization respectively. In the first step, an estimation of the expectation of log-likelihood is performed and in the second step, the parameters will be selected to maximize the expectation of this log likelihood. In the next round, the updated parameters are used to estimate a new expectation.

Based on the available data, different assumptions can be made for a Gaussian mixture model, for instance, spherical versus elliptical distributions and uniform or non-uniform prior probabilities [4].

1.2.2. k-means

$k$-means clustering method is one of the most widely used techniques in cluster analysis introduced in [23]. This algorithm automatically groups observations in $k$ clusters based on a similarity criterion. It starts with selecting $k$ points as $k$ centroids of clusters, then assigns each point to the closest cluster. In the next step, it updates the clusters and computes the new centroid for each cluster. The algorithm will converge when there is no change in clusters. The purpose of $k$-means algorithm is to minimize the within cluster variance which is the sum of squared deviations of each subset of points from its centroid [1].

$k$-means algorithm is known as one of the simplest clustering methods that performs well with so many data especially when the clusters are isolated and compact. However,
this algorithm suffers from getting trapped in the local minimum instead of resulting in the desired global minimum. One possible solution to overcome this drawback is to run algorithm several times with different random seeds and choose the most repeated partition as the desired result [4].

1.2.3. Fuzzy c-means

Fuzzy c-means algorithm groups a data set using iteratively searching for clusters with minimizing an objective function. Similar to k-means algorithm, this objective function computes the sum of squared deviations of each point from its cluster centroid with a multiplying membership factor. This membership factor is calculated based on the distance between a point and the centroid of a cluster and shows a degree on how a point belongs to a cluster. In each iteration, the centroid of clusters and membership factors are updated. These procedures are repeated until convergence is reached [4].

1.2.4. Hierarchical algorithm

Hierarchical algorithms cluster data based on a similarity criterion by creating a hierarchy of clusters which is represented by a two-dimensional diagram called dendrogram. They are categorized into two groups, agglomerative and divisive. The agglomerative methods consider each point as a singleton group initially and then in each iteration merge a pair of points based on a heuristic criterion until all points in a point set make a group. In contrast, the divisive methods move backward by assuming that all points are in one cluster and remove the connections (usually referred to as links) between them successively to find the best cluster. The divisive methods are more complex and can be
categorized into several subgroups based on the techniques that are selected to remove the undesirable links [24].

1.2.5. Bayes clustering

The Bayes clustering algorithm is an optimal model-based cluster analysis which provides a probabilistic framework and operates on a random labeled point process. The Bayes clustering algorithm requires to compute the probability mass function (PMF) for all possible partitions for a point set and calculate a pre-defined cost matrix. Then, the error is calculated by multiplying the cost matrix into the vector of probability mass functions for all possible partitions. The optimal partition is a partition associated with the minimum error [3].

One of the most significant differences between this algorithm and other methods (fuzzy c-means, hierarchical, and etc.) is that Bayes clustering method clusters data on a basis of natural definition of clustering error. They in [3] provided detailed mathematical expression for computing the probabilities of partitions for a point set in the presence of three Gaussian mixture models. These Gaussian models are discussed in the next chapter. They also suggested some techniques as computation reduction to mitigate the computation complexities. As it was stated in this work, the problem of huge computation cost is inevitable even with moderate size point sets [3]. To overcome this problem, they suggested several suboptimal methods such as suboptimal Pseed, suboptimal Pmax, suboptimal Pseed Fast, and suboptimal Pseed Faster. They implemented several simulations with small, moderate, and large point sets. They
made some simplifying assumptions in their simulations which are addressed in the next section [3].

1.3. Motivation

The original paper did not implement and study the performance of Bayes clusterer and the suboptimal algorithms under several conditions.

First of all, Bayes clusterer and the suboptimal algorithms were only implemented for two cluster problems. In many practical problems, data belongs to more than two clusters. Therefore, it is imperative to implement a suboptimal method which is able to identify more than two clusters in a data set. It is worthwhile to mention that as the number of clusters increases in a data set, finding the appropriate clusters becomes more challenging for a clustering algorithm, due to a more complex nature of the problem and larger search space for the desired partition. Hence, it is required to develop a clustering algorithm which can group more than two clusters, while coping with the foregoing issues.

Second, for simply implementing the simulations, they assumed that the size of each cluster is equal which is not necessarily true, especially when we work with real data. They made an assumption that the prior probability of each partition is uniform.

It should be noted that, although the implementations in [3] were not general and had these limiting issues, the theoretical framework of clustering in [3] is comprehensive and solves the above problems in the developed formulation for Bayes clusterer. The contribution of this thesis is in coding the algorithm in the original paper in matlab for multiple clusters and Dirichlet priors.
In this research, the focus is on overcoming the discussed problems in simulations by implementing and studying a suboptimal clustering method which is able to analyze more than two clusters. Allowing multiple clusters is a valuable and interesting feature in a clustering problem which can give us meaningful subsets for a data set.

Another particular aspect of this research is that it assumes prior probabilities on clusters based on Dirichlet distribution instead of using uniform distribution. By applying this distribution we can also control the number of inputs to manage the priors more convenient, i.e. instead of using a vector with size \( l^n \) (for a point set with \( n \) points and \( l \) clusters) for determining the prior probabilities for all partitions, we use an \( l \) length vector for \( \alpha_i \)'s to specify the priors. The details about the prior probabilities and Dirichlet distribution are discussed in Chapter 3.

Furthermore, the behavior of the suboptimal clustering algorithm under different number of clusters \( l \), number of points \( n \), dimensions \( d \), parameters of Gaussian model, and \( \alpha_i \)'s is addressed in this study.

It also reports technical implementation, computation, overflow/underflow problems.

1.4. Outline of thesis

This current work implements and studies a suboptimal multi-clustering method based on the Bayes clustering algorithm (optimal) to reduce the computation costs of this method. Also, it is shown that the suboptimal method maintains the performance near the optimal algorithm for a data set with a small number of clusters. Furthermore, the effects of hyperparameters for the model are discussed in the simulations.
The remainder of this research study is organized as follows. In Chapter 2 we explain the underlying probabilistic framework for Bayes clustering algorithm. This is followed by a brief formulation of Bayes clustering operators and a definition of error. Finally, the details about the Bayes clustering algorithm proposed by Dalton et al. [3] are discussed. Chapter 3 is devoted to describing the suboptimal multi-clustering algorithm called FCM-IB. Also, the details about the implementation of the proposed method, data generation, evaluation, and results are presented in Chapter 4. Chapter 5 summarizes the results of this work and draws conclusions. Furthermore, the limitations of the research are discussed and some suggestions are presented for the future work of this study.
Chapter 2: Bayes Clustering

2.1. Overview

The Bayes clustering algorithm is a relatively new method in the literature of cluster analysis. This method is a model-based cluster analysis which provides a probabilistic framework to find the optimal partition [3].

The main work of this thesis is based on the research developed by Dalton et al. [3]. This chapter first sums up the fundamental concepts of this probabilistic framework introduced by Dougherty et al. [21]. This is then followed by a brief formulation of Bayes clustering operators and a definition of error. Finally, the details about the Bayes clustering algorithm proposed by Dalton et al. [3] are discussed.

2.2. The probabilistic framework

In this section, we introduce a probabilistic framework to find an optimal clustering algorithm which groups an observed point set with minimum expected error. Then, we define two fundamental definitions that play an important role in the understanding of this framework, the random labeled point sets and the label operator. Also, it should be noted that in this study, observations are the points in a point set and each cluster is called a label.
The Clustering algorithm assigns a distinctive label function \( \phi_S \) to a point set \( S \). Let \( S \subset \mathbb{R}^d \) be a realization of a random point set \( \Xi \) in a space with \( d \) dimensions. Also, let \( \Lambda \) be a random labeling consists of all label functions \( \phi_S \) which cluster all points in a point set. Given a point set \( S \), a label function \( \phi_S \) is a deterministic function which maps each point in \( S \) to a label \( L = \{0, 1, \ldots, l - 1\} \). We can attribute a probability mass function \( P_S(\phi_S) \) to a family of label functions \( L^S \).

We can concise a random labeled point process by a pair \((\Xi, \Lambda)\), in which \( \Xi \) is a point process and \( \Lambda \) is a random labeling.

2.2.1. Label switching issue

The label switching problem has been discussed in [3]. This problem is inherent to clustering. In fact, the main goal of a cluster operator \( \zeta \) is to partition a point set \( S \) regardless of the actual labeling. Therefore, a family of cluster operators called \( F_\zeta \) can group points in a same way with different combinations of labels. That is, all permutations of labels that induce similar partitions are equivalent.

2.2.2. The Bayes clusterer

The main idea of Bayes clusterer introduced in [3] is to provide a principled statistical framework and cluster data based on a probability model. It is assumed that all elements in a data set are originated from a finite mixture model. A block diagram in Figure 2 depicts this probabilistic framework in brief. In the following, each part is fully described.
The error of partition $\zeta(S)$ in [3] is defined by the minimum proportion of mislabeled points in $S$ over all label functions including $\zeta(S)$, which can also be written in the form

Figure 2 Block diagram of Bayes clustering algorithm.
\[ E_\zeta(S) = \sum_{P_S \in \mathcal{K}_S} c_S(\zeta(S), P_S) P_S(P_S), \quad (2.7) \]

where \( \mathcal{K}_S \) is the set of all possible partitions defined for a point set \( S \) and \( c_S \) is a natural partition cost function showing the number of mislabeled points between two partitions which are given as inputs. This natural cost function is given by

\[ C_S(Q_S, P_S) = \frac{1}{\eta(s)} \min_{\phi_{S,Q_S} \in G_{Q_S}} \sum_{x \in S} I_{\phi_{S,P_S}(x) \neq \phi_{S,Q_S}(x)}. \quad (2.8) \]

where \( \eta(s) \) is the number of points in \( S \) and \( G_{Q_S} \) refers to the label switching problem explained in Section 2.2.1 and is a family of label functions which induce a specific partition \( Q_S \). The second term in equation 2.7 is \( P_S(P_S) \) which is the probability mass function (PMF) of partition \( P_S \) and is given by

\[ P_S(P_S) = \sum_{\phi_S \in G_{P_S}} P_S(\phi_S), \quad (2.9) \]

where partition \( P_S \) can be any partition in the set of all possible partitions \( \mathcal{K}_S \) defined on \( S \) and \( \phi_S \) can be any label function that induces partition \( P_S \), which forms the set \( G_{P_S} \). Having said that, the PMF of a partition equals the sum of PMFs of all label functions which induce that partition.

Bayes clusterer \( \zeta^*(S) \), is an optimal clustering operator which has the minimum error over all points in \( S \). In general, following Equation (2.7), the Bayes clustering can be formulated as an optimization problem.

\[ \zeta^*(S) = \arg \min_{\zeta(S) \in \mathcal{K}_S} \sum_{P_S \in \mathcal{K}_S} c_S(\zeta(S), P_S) P_S(P_S). \quad (2.10) \]
Therefore, for finding the optimal partition using the Bayes clusterer, we need to calculate two terms in Equation (2.10), the probability mass function $P_S(\mathcal{P}_S)$ and the cost function $c_S$. In order to compute $P_S(\mathcal{P}_S)$, we need to know our finite mixture models. Then, we explain how to calculate $P_S(\mathcal{P}_S)$ and $c_S$ [3].

2.2.3. Gaussian mixture models

A Gaussian mixture model is a probability density function with distribution parameters that comprises weighted sum of Gaussian components (Gaussian models) [25]. In fact, we assume that all points in a point set $S$ are originated independently from a mixture of a finite number of these Gaussian components. Next, we briefly discuss three Gaussian models that were introduced in the original paper. For further information on these models, see [3].

A. Gaussian distributions with known means and known covariances

In this model, a point set is generated by a finite number of Gaussian distributions with known means and known covariances where a group of points with cluster $i$ is related to a distribution with parameter $\rho_i = \{\mu_i, \Sigma_i\}$. In particular, the probability density function (PDF) for each point $x \in S$ with cluster $i$ and parameter $\rho_i$ is corresponding to

$$f_i(x; \rho_i) \sim \mathcal{N}(\mu_i, \Sigma_i)$$

$$f_i(x; \rho_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \times \exp\left(-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i)\right).$$

(2.11)

It is worthwhile to mention that $\mu_i$ is a real vector with $d$ elements and $\Sigma_i$ is a symmetric positive definite matrix of size $d \times d$. 

15
B. Gaussian distributions with Gaussian means and known covariances

Under this model, a point set is generated by $l$ Gaussian distributions similar to Model A. The only difference is that means of these distributions are not known and have independent Gaussian distributions with mean $m_i$ and covariance $\frac{1}{\nu_i} \Sigma_i$; where $i \in \{1, \ldots, l\}$, $m_i$ (a real vector with $d$ elements) and $\nu_i$ (a positive real number) are two fixed hyperparameters. Therefore, the PDF for each point $x \in S$ in cluster $i$ with parameter $\rho_i$ is characterized by

$$f_i(x; \rho_i) \sim \mathcal{N}(\mu_i, \Sigma_i)$$

where $\mu_i \sim \mathcal{N}\left(m_i, \frac{1}{\nu_i} \Sigma_i\right)$.

C. Gaussian distributions with Normal-Inverse-Wishart means and covariances

For this model, both means and covariances are random. The means are generated by Gaussian distributions and the covariances are generated from an inverse Wishart distribution, which generally is called Normal-Inverse-Wishart means and covariances. In this model, for each $i \in \{1, \ldots, l\}$ we define four hyperparameters $m_i, \nu_i, \kappa_i$ and $\psi_i$; $m_i, \nu_i$ are described in Model B, $\kappa_i$ is a real number greater than $d - 1$ and $\psi_i$ is a symmetric positive definite matrix of size $d \times d$. Hence, the distribution for a point $x \in S$ with cluster $i$ and parameter $\rho_i$ is given by

$$f_i(x; \rho_i) = \mathcal{N}(\mu_i, \Sigma_i),$$

where $\mu_i \sim \mathcal{N}\left(m_i, \frac{1}{\nu_i} \Sigma_i\right)$ and $\Sigma_i \sim \mathcal{W}^{-1}(\kappa_i, \psi_i)$. Equation (2.12) represents the inverse Wishart distributions of $\Sigma_i$. 

16
\[ f(\Sigma_i) = \frac{|\psi_i|^{\frac{k_i}{2}}}{2^{\frac{k_i}{2}} \Gamma_d(\frac{k_i}{2})} |\Sigma_i|^{-\frac{k_i + d + 1}{2}} \exp \left( -\frac{1}{2} \text{tr}(\psi_i \Sigma_i^{-1}) \right), \]  

where \( \Gamma_d \) is the multivariate Gamma function.

2.2.4. Probability mass functions \( P_S(\mathcal{P}_S) \)

For calculating \( P_S(\mathcal{P}_S) \), according to Equation (2.9) we need to find \( P_S(\phi_S) \) for \( \phi_S \in L^S \) which can be computed by the following equation

\[ P_S(\phi_S) = P(\phi_S|S) \propto P(\phi_S)f(S|\phi_S), \]  

where \( P(\phi_S) \) is a known prior defined on labels and \( f(S|\phi_S) \) is the likelihood of a point set \( S \) given label function \( \phi_S \). The following equation holds by considering a distribution parameter \( \rho \) with prior density \( f(\rho) \)

\[ P_S(\phi_S) \propto P(\phi_S) \int f(S|\phi_S, \rho)f(\rho)d\rho. \]  

Furthermore, another assumption is that in a point set \( S \) with \( l \) clusters, each point \( x \in S \) is independently generated from a label-\( i \)-conditional distribution for a label function \( \phi_S \) and a collection of distribution parameters \( \rho = \{\rho_1, \ldots, \rho_l\} \) in which \( \rho_i \) is a distribution parameter related to label \( i \). It is also presumed that \( \rho_i \) has prior probability \( f(\rho_i) \) and \( \rho_i \)'s are independent from each other. Thus the following equation is obtained

\[ P_S(\phi_S) \propto P(\phi_S) \prod_{l=1}^{l} \int \left( \prod_{r \geq 1} f_i(x; \rho_i) \right)f(\rho_i) d\rho_i. \]
where \( n_i \) in the number of points with label \( i \) and \( S_i \) is the subset of \( S \) associated with label \( i \). From now on we use the expression \( L_i(S_i) \) by

\[
L_i(S_i) = \int (\prod_{x \in S_i} f_i(x; \rho_i)) \ f(\rho_i) d\rho_i.
\]  

(2.16)

Therefore, Given the Gaussian model, we can compute \( L_i(S_i) \) and accordingly \( f(S|\phi_S) \).

In this thesis, we use the Normal-Inverse-Wishart Gaussian model (Model C) which was previously explained in the Section 2.2.3. In the following, we skip the mathematical relations for the Model A and Model B stated in Section 2.2.3, and just focus on the formulation for third model.

Following the procedures in [3], we first define \( \psi_i^* \) for label \( i \) and \( n_i \geq 1 \)

\[
\psi_i^* = (n_i - 1)\tilde{\Sigma}_i + \frac{n_i n_i}{n_i + \nu_i} (\tilde{\mu}_i - m_i)(\tilde{\mu}_i - m_i)^T,
\]

(2.18)

where \( \tilde{\mu}_i \) and \( \tilde{\Sigma}_i \) are sample mean and sample covariance of all points in cluster \( i \) respectively and \( m_i \) and \( \nu_i \) are hyperparameters which were defined previously. If \( n_i = 1 \), Equation (2.18) turns to

\[
\psi_i^* = \frac{\nu_i}{1 + \nu_i} (\tilde{\mu}_i - m_i)(\tilde{\mu}_i - m_i)^T.
\]

(2.19)

Also, if \( n_i = 0 \), we adjust \( \psi_i^* \) as a zero matrix of size \( d \times d \).

Therefore, by defining \( \psi_i^* \) in Equation (2.18), the term \( L_i(S_i) \) for a \( d \)-dimensional point set \( S \) with \( l \) clusters and Normal-Inverse-Wishart means and covariances is given by

\[
L_i(S_i) = \frac{\mid \nu_i \mid^\frac{d}{2} \mid \psi_i \mid^\frac{k_i}{2} \Gamma_d(k_i) \Gamma_d(k_i + n_i) \Gamma_d(k_i + \nu_i)}{\mid n_i + \nu_i \mid^\frac{d}{2} \mid \psi_i + \psi_i^* \mid^\frac{k_i + n_i}{2} \Gamma_d(k_i + \nu_i + n_i) \Gamma_d(k_i + n_i) \Gamma_d(k_i + \nu_i + n_i)}.
\]

(2.20)

where \( \nu_i \) and \( k_i \) are hyperparameters, and \( \mid \cdot \mid \) points out to a determinant. Substituting (2.20) in (2.15), we obtain
\[
P_S(\phi_S) \propto P(\phi_S) \prod_{i=1}^{l} \frac{|\nu_i|^\frac{d}{2} \Gamma_d \left(\frac{\kappa_i + n_i}{2}\right) |\psi_i|^\frac{\kappa_i}{2}}{|n_i + \nu_i|^\frac{d}{2} \Gamma_d \left(\frac{\kappa_i}{2}\right) |\psi_i + \psi_i^*|^\frac{\kappa_i + n_i}{2}}.
\]

2.2.5. Cost function \((c_S)\)

In order to find the Bayes clusterer we need to compute the cost function \(c_S\) to solve Equation (2.10). Equation (2.8) describes the partition cost function as well. In the research was carried out by Dalton et al. [3], they used a specific cost function named natural cost function. Based on the fact stated in [3], it is possible to use different cost functions depending on generalization of the performance. It should be noted that different cost functions implicate different choices of actions. For example, for zero-one cost function which considers one unit cost for any incorrect decision and zero cost for correct decisions, we choose the maximum a posteriori estimator (MAP). In the context of clustering with the Bayesian decision theory considering zero-one cost, the optimal decision is to select the partition with maximum posterior probability.

To follow the purpose of finding the Bayes partition which has the minimum defined error, two expressions are defined, the candidate partitions \(C_s = \{Q^1, ..., Q^{|c_S|}\} \subseteq K_S\) coupled with the reference partitions \(R_s = \{P^1, ..., P^{|R_S|}\} \subseteq K_S\). The candidate partitions make the search space and the reference partitions are partitions with known probabilities. In a brute force search, we set both candidate partitions and reference partitions equal \(K_S\).
Chapter 3: Multi-clustering Bayes Suboptimal Algorithm

3.1. Overview

In this chapter, a suboptimal clustering algorithm called FCM-IB is discussed. This algorithm is presented in [3] as “suboptimal Pseed” that approximates the Bayes clustering algorithm introduced in [3]. The main objective of this study is to develop a new code and analyze the ability to cluster more than two groups.

Before we explain the suboptimal method, it is essential to address some assumptions that are applied in the suboptimal method.

3.2. Assumptions

First, we need to define the hamming distance between two partitions of equal length. Hamming distance is the number of points in two partitions with different labels. In this thesis, the distance between two partitions is given by minimum hamming distance, which can be defined as the minimum numbers of mismatch labels in two partitions when we consider the label switching issue. This measuring equals to \( \eta(S)c_S(\mathcal{P}^1, \mathcal{P}^2) \), where \( \eta(S) \) is the cardinality of a point set, \( S \), and \( c_S(\mathcal{P}^1, \mathcal{P}^2) \) is the cost defined between \( \mathcal{P}^1 \) and \( \mathcal{P}^2 \) according to Equation 2.8.
Second, it is assumed that the probabilities of partitions near each other are close. Therefore, when we look for high probability partitions, we can confine the search space to the neighborhood of the partition with maximum probability [3].

3.3. Suboptimal algorithm

This suboptimal method comprises two main stages. The first stage (call Algorithm 1) is about finding the partitions within the neighborhood with radius $k$ (a given fixed positive integer) of the seed partition. These neighbor partitions are in the closed ball with a center at seed partition and a radius $k$. As described before, the distance between partitions is measured via minimum hamming distance and considers the label switching problem. In the second stage (call Algorithm 2), the scaled probabilities (known as un-normalized probabilities) of all partitions, which are acquired from the first stage are computed. These partitions include the seed partition and the neighboring partitions. Then, the partition with maximum probability is selected by comparing these scaled probabilities. Generally, this is done iteratively by repeating these two stages until there is no improvement and the algorithm is converged to the highest probability partition or a stopping condition is met.

Here $\mathcal{P}_S = \{S_1, S_2, ..., S_l\}$ denotes the entire point set, $S$, with $l$ clusters, and $S_i \subset S$ the set of points that belongs to cluster $i$. Our goal is to identify these intrinsic separate groups $S_1, S_2, ..., S_l$.

The algorithm is initialized by finding an initial seed partition with $l$ clusters using fuzzy $c$-means (FCM) algorithm. In fact, the suboptimal algorithm is sometimes sensitive to initial seed because the objective function (Probability of all partitions, $P_S(\mathcal{P}_S)$) is a nonconvex function.
That is to say, an improper initial seed may lead the algorithm to get trapped in local optima. FCM has the advantage of being really fast, and has shown a reasonable performance under the Gaussian model we have implemented.

In the next step, after finding the initial seed, given a fixed integer $k$ and $l$, we use Algorithm 1 to find the neighboring partitions of the initial seed. This algorithm was discussed in [3]. It is worthwhile to mention that finding all partitions within the closed ball with a center at seed partition and a radius $k$ is equivalent to the set of partitions with hamming distance at most $k$ from seed partition.

---

**Algorithm 1**: Find the neighbors partitions in a closed ball with center at seed partition $\mathcal{P}_i$ and radius $k$ with $l$ clusters (including the seed partition)

**Data**: $\mathcal{P}_i = \text{the initial seed}$

$k = \text{radius of ball search or the radius of neighborhood}$

$l = \text{number of clusters}$

**Result**: $\mathcal{R}_s = \text{partitions within the closed ball with center at seed partition and radius } k$

\[
\mathcal{R}_s = \{\} ; \\
\text{for } j = 1 \text{ to } k \text{ do} \\
\quad \mathcal{X}_s = \text{set of all partitions with hamming distance } j \text{ from } \mathcal{P}_i ; \\
\quad \mathcal{R}_s = \{\mathcal{R}_s, \mathcal{X}_s\} ; \\
\text{end} \\
\mathcal{R}_s = \{\mathcal{R}_s, \mathcal{P}_i\} \\
---
As a matter of fact, the complexity and computation runtime of the suboptimal algorithm are constrained by the radius $k$. If $k = 0$ the initial seed partition is the only reference partition $\mathcal{R}_s = \mathcal{P}_I$. If $k = n - 1$ the search space includes all possible partitions.

When this step is finished, we need to find the scaled probabilities of all partitions acquired from the previous stage using Algorithm 2 which was proposed in [3]. As we described in Chapter 2, we use the Normal-Inverse-Wishart (NIW) means and covariances model to compute the probability of each partition. Therefore, proceeding to use Algorithm 2, it is necessary to determine the hyperparameters for this model.

---

**Algorithm 2**: Find the highest probability partition

**Data**: $\mathcal{R}_s$ = a set of partitions

$l$ = number of clusters

$\rho'_i = \{m_i, \nu_i, \kappa_i, \psi_i, \alpha_i\}$ = a set of hyperparameters for NIW model and prior

**Result**: $\mathcal{P} = \text{local highest probability partition}$

**Do**

$P_S = \text{scaled probabilities of } \mathcal{R}_s \text{ using } \rho'_i$;

$P = \max \{P_S\}$;

$\mathcal{P} = \text{the partition corresponding to } P$

---

In the following, the partition with the highest probability is recognized and considered as the new seed partition for Algorithm 1. We repeat this procedure until there is no partition with a probability higher than the probability of the seed partition or a stopping condition is met. We call this procedure iterative ball search (IB) algorithm. The convergence of this IB algorithm to a local maximum in the finite number of repetitions is
guaranteed due to two reason. First, the search space contains a finite number of partitions, second, the probabilities of the new seeds are increasing. We can also control this by a parameter in the algorithm called \( max\_itr \). This parameter shows the maximum number of iteration for the algorithm until it converges. If this parameter is infinity there is no limitation and the algorithm can freely repeat until the convergence achieved.

A heuristic solution is that using a number of initial seed partitions (call these partitions \( \mathcal{P}_{I_1}, \mathcal{P}_{I_2}, ..., \mathcal{P}_{Ir} \)), then finding the final the local optimal partition for each, and keeping the best final partition which has the highest probability among all of them. In the suboptimal algorithm, we use the parameter \( r \) as the number of initial seed partitions.

We have combined these algorithms into one algorithm, and we have called it FCM-IB algorithm. The FCM-IB algorithm is shown below.
Algorithm FCM-IB: The suboptimal multi-clustering algorithm

**Data:** $S$ = a point set

$l$ = number of clusters

$r$ = number of repetitions

$k$ = radius of ball search or the radius of neighborhood

$p_i' = \{m_i, \nu_i, \kappa_i, \psi_i, \alpha_i\}$ = a set of hyperparameters for NIW model and prior

$max_{itr}$ = maximum number of iterations

**Result:** $P'$ = local highest probability partition

$t = 0$;

$R_s' = \{}$;

**for** $i = 1$ to $r$ **do**

$P_i = \text{FCM}(S, l)$;

$P(0) = \text{NAN}(1, n)$;

**repeat**

$t = t + 1$;

$R_s = \text{Algorithm 1}(P_i)$;

$P(t) = \text{Algorithm 2}(R_s, l, p_i')$;

**until** $P(t) = P(t + 1)$ or $t > max_{itr}$;

$P_i = P(t)$;

$R_s' = \{R_s', P_i\}$;

**end**

$P' = \text{Algorithm 2}(R_s', l, p_i')$;

Now we will explain how to compute the probability of each partition, as we presented in Algorithm 2. These probabilities are derived in three steps. In the first step, the likelihood of each label function is calculated under the NIW Gaussian model. In the second step, the prior probabilities of these label functions are computed. Based on these two steps,
probabilities of partitions are derived. In the next section, we will explain the details of these steps.

3.4. Partition Probabilities

Based on Equation (2.9), (2.15) and (2.16) we have

\[
P_S(P_S) = \sum_{\phi_S \in G_{P_S}} P_S(\phi_S) \propto \sum_{\phi_S \in G_{P_S}} P(\phi_S) f(S|\phi_S)
\]  

\[
= \sum_{\phi_S \in G_{P_S}} (P(\phi_S) \prod_{l=1}^{l} L_l(S_l)) .
\]  

As can be seen, for computing the probabilities we need to compute two main terms, the prior probability for each label function \( P(\phi_S) \) and the likelihood \( L_l(S_l) \).

3.4.1. Likelihood

In this thesis, we use the NIW Gaussian model (Model C), which was previously explained in the Section 2.2.3. Based on this model, according to Equation (2.20), likelihood function for each label function is calculated.

3.4.2. Prior probabilities of label functions

In this research, we use Dirichlet-multinomial distribution to produce the prior probabilities for each label function \( \phi_S \).

The Dirichlet distribution, denoted by \( Dir(\alpha) \), is a continuous multivariate probability distribution. This distribution is a function of a vector \( \alpha \) and a vector \( p. \alpha \) is a \( 1 \times l \) vector,
each a positive real number, \( \alpha_i > 0 \). Also, \( \mathbf{p} \) is a vector of independent variables of size \( 1 \times l \) in an l-dimensional simplex. Each member of \( \mathbf{p} \), \( p_i \), is a value between zero and one. In addition, \( \mathbf{p} \) is a normalized vector, i.e. the sum of \( p_i \) values is equal to one. In the context of clustering problems, \( p_i \)'s are the probabilities that an individual point is assigned cluster \( i \), and \( l \) is the total number of existing clusters. In mathematical terms, \( \text{Dir}(\alpha) \) of order \( l \geq 2 \), is expressed by

\[
f(p_1, ..., p_l; \alpha_1, ..., \alpha_l) = \frac{\Gamma\left(\sum_{i=1}^{l} \alpha_i\right)}{\prod_{i=1}^{l} \Gamma(\alpha_i)} \prod_{i=1}^{l} p_i^{\alpha_i-1},
\]

(3.2)

where \( \Gamma(\cdot) \) denotes the gamma function.

Utilizing Dirichlet-multinomial distribution with different parameters for generating the prior probabilities gives us the ability to generate different priors not just uniform priors.

For a point set \( S \) with \( n \) points and \( l \geq 2 \) clusters, given the assumption that the probabilities of the points belonging to any cluster, \( p_i \)'s are independent, the probability of a label function \( \phi_S \), with \( n_1, n_2, ..., n_l \) points in the \( l \) clusters, is obtained by

\[
P(\phi_S | \mathbf{p}) = p_1^{n_1} \cdot p_2^{n_2} \cdot ... \cdot p_l^{n_l} = \prod_{i=1}^{l} p_i^{n_i}
\]

(3.3)

where \( p_1, p_2, ..., p_l \) are not deterministic and are generated from a Dirichlet distribution with parameter \( \alpha = (\alpha_1, \alpha_2, ..., \alpha_l) \) given by

\[
p_1, p_2, ..., p_l \sim \text{Dir}(\alpha_1, \alpha_2, ..., \alpha_l)
\]
\[ P(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_l; \mathbf{\alpha}_1, ..., \mathbf{\alpha}_l) = \frac{\Gamma\left(\sum_{i=1}^{l} \alpha_i\right)}{\prod_{i=1}^{l} \Gamma(\alpha_i)} \prod_{i=1}^{l} p_i^{\alpha_i-1}, \]  

(3.4)

Therefore according to the theorem of total probability, \( P(\phi_S) \) can be calculated as

\[ P(\phi_S) = \int P(\phi_S|\mathbf{p}) P(\mathbf{p}) d\mathbf{p} = \frac{\Gamma\left(\sum_{i=1}^{l} \alpha_i\right)}{\prod_{i=1}^{l} \Gamma(\alpha_i)} \int \prod_{i=1}^{l} p_i^{\alpha_i-1} \prod_{i=1}^{l} p_i^{n_i} d\mathbf{p} \]  

(3.5)

\[ = \frac{\Gamma\left(\sum_{i=1}^{l} \alpha_i\right)}{\prod_{i=1}^{l} \Gamma(\alpha_i)} \int \prod_{i=1}^{l} p_i^{n_i+\alpha_i-1} d\mathbf{p}. \]

Since for a Dirichlet distribution with parameter \( \mathbf{\alpha} + \mathbf{n} \) we have,

\[ \int \left(\frac{\Gamma\left(\sum_{i=1}^{l} n_i + \alpha_i\right)}{\prod_{i=1}^{l} \Gamma(\alpha_i + n_i)} \right) \prod_{i=1}^{l} p_i^{\alpha_i+n_i-1} d\mathbf{p} = 1 \]  

(3.6)

Therefore,

\[ P(\phi_S) = \frac{\Gamma\left(\sum_{i=1}^{l} \alpha_i\right)}{\prod_{i=1}^{l} \Gamma(\alpha_i)} \frac{\prod_{i=1}^{l} \Gamma(\alpha_i + n_i)}{\Gamma\left(\sum_{i=1}^{l} \alpha_i + n_i\right)}. \]  

(3.7)

It is worthwhile to mention that the parameters \( \alpha_i \)'s in Dirichlet distribution are crucial as they control the distribution of clusters. If we choose unequal \( \alpha_i \)'s, we are biased toward specific clusters as it can be seen in Figure 3 (figure drawn from a part of a code shared by Obeid [27]).
Figure 3 Visualization of Dirichlet distribution for three clusters with $\alpha_i$'s equal to (a) $\alpha = (1,1,1)$ (b) $\alpha = (4,4,4)$ (c) $\alpha = (15,15,15)$ (d) $\alpha = (15,4,4)$ (e) $\alpha = (2,2,7)$ (f) $\alpha = (1,13,1)$ (courtesy of Obeid [27])
Furthermore, the mean and covariance for Dirichlet distribution are defined by

\[ E[p_i] = \frac{\alpha_i}{\sum_{i=1}^{l} \alpha_i} \]  \hspace{1cm} (3.8)

\[ Var[p_i] = \frac{\alpha_i(\sum_{i=1}^{l} \alpha_i - \alpha_i)}{(\sum_{i=1}^{l} \alpha_i)^2(\sum_{i=1}^{l} \alpha_i + 1)} \]  \hspace{1cm} (3.9)

where \( \alpha_i \)'s have equal values of \( \alpha \), \( E[p_i] \) and \( Var[p_i] \) are as follows

\[ E[p_i] = \frac{1}{l} \]  \hspace{1cm} (3.10)

\[ Var[p_i] = \frac{l - 1}{l^2(l\alpha + 1)} \]  \hspace{1cm} (3.11)

These equations show that as \( \alpha \) becomes smaller, the variance gets larger. Thus, it is probable that \( p_i \)'s take values within a wide range from zero to one (i.e. the probability of having values within a wide range from zero to one is non-negligible). On the contrary, as \( \alpha \) gets larger, the variance decreases and the probabilities of \( p_i \)'s become more for values around \( E[p_i] \), which is equal to \( \frac{1}{l} \). Consequently, \( p_i \)'s are close to uniform for a large \( \alpha \) [26].
Chapter 4: Simulations and results

4.1. Overview

In this chapter, we explain different simulations using to analyze and study the suboptimal clustering algorithm (FCM-IB) compared to the Bayes clusterer, MAP and other classical clustering algorithms fuzzy c-means (FCM), k-means (KM), hierarchical clustering with single linkage (Hier. (Si)), hierarchical clustering with average linkage (Hier. (Av)), hierarchical clustering with complete linkage (Hier. (Co)), Expectation maximization (EM), and randomly clustered points (Random).

4.2. Data generation

We generate two-dimensional random labeled point process under the Gaussian mixture model described in Section 2.2.3 with multiple clusters, i.e. \( l \geq 2 \). This model is derived from Normal-inverse-Wishart distribution with Gaussian means and inverse-Wishart covariances.

First, we assume that

\[
p = [p_1, p_2, ..., p_l] \sim Dir(\alpha_1, \alpha_2, ..., \alpha_l)
\]
where \( p \) is the vector of prior probabilities and each \( p_i \) is the probability of cluster \( i \) (or label \( i-1 \)).

In Dirichlet distribution, if all \( \alpha_i \)'s are equal, we have a symmetric Dirichlet distribution such that \( E[p_i] = \frac{1}{l} \). We choose \( \alpha_i \)'s to be equal since we do not want to become biased towards a specific cluster. Because in a real problem, we do not have any information about the clusters and the number of points in each cluster. Intuitively, the high values of \( \alpha_i \)'s indicate the prior probabilities that are generated from Dirichlet, variate dense around \( E[p_i] = \frac{1}{l} \). On the contrary, a small value of \( \alpha_i \) generates prior probabilities \( (p = [p_1, p_2, ..., p_l]) \) from a sparse evenly distributed distributions (see Figure 3).

Therefore, the prior probabilities generated by small \( \alpha_i \)'s are non-uniform which make these simulations different from the simulations in [3] where the priors were uniform. As it was discussed in Chapter 1, assuming uniform prior probabilities is not necessarily true especially in real data.

Therefore, we set \( \alpha_i = 1 \ (i = 1, 2, ..., l) \), which means the number of points in each cluster is sparse and varies from zero to \( n \). The idea behind this is to test the performance of the suboptimal algorithm with arbitrary number of points in each cluster not just with equal clusters which is a special case. Therefore, the prior probabilities of each cluster are produced in this step.

Next, the number of points in each cluster are drawn from a multinomial distribution with parameters \( p \) and \( n \). The parameter \( p \) is a vector of prior probabilities which are drawn from a Dirichlet distribution with hyperparameters \( \alpha_i \)'s. As a result, for a point set with \( n \) points and \( l \) clusters, the number of points in each cluster, \( n_1, n_2, ..., n_l \), can be any non-
negative integers from zero to $n$, such that $n_1 + n_2 + \ldots + n_l = n$. That is, we have $n_1$ points with label ‘0’, $n_2$ points with label ‘1’ to $n_l$ points with label ‘$l - 1$’. Given $n_1, n_2, \ldots, n_l$, we compose the actual label vector (true labels) randomly. Then, given the hyperparameters of the model $(m_i, \nu_i, \kappa_i$ and $\psi_i$, $i = 1, 2, \ldots, l$), for generating a set $S$, the points are drawn from $l$ Normal-inverse-Wishart distributions followed by a vector of true labels showing the cluster of each point in a set $S$. Therefore, a $d$-dimensional point set with $n$ points is defined by a matrix of numerical values of size $n \times d$, such that each row represents a point. This is followed by a vector of random labels called true label.

Table 1 depicts settings used for generating point sets. If two vectors (or two points) have the same labels, they are in the same cluster.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$m$</th>
<th>$\psi$</th>
<th>$\nu$</th>
<th>$\kappa$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_1 = 1$</td>
<td>$m_1 = [0,\ldots,0]_{1\times d}$</td>
<td>$\psi_1 = 0.5 . I_d$</td>
<td>$\nu_1 = d + 2$</td>
<td>$\kappa_1 = d + 2$</td>
</tr>
<tr>
<td>$\alpha_2 = 1$</td>
<td>$m_2 = [1,\ldots,1]_{1\times d}$</td>
<td>$\psi_2 = 0.5 . I_d$</td>
<td>$\nu_2 = d + 2$</td>
<td>$\kappa_2 = d + 2$</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$\alpha_l = 1$</td>
<td>$m_l = [l - 1,\ldots,l - 1]_{1\times d}$</td>
<td>$\psi_l = 0.5 . I_d$</td>
<td>$\nu_l = d + 2$</td>
<td>$\kappa_l = d + 2$</td>
</tr>
</tbody>
</table>

4.3. Experiments and results

The error computes for each set by counting the mismatch labels between the actual label vector and the label vector resulted from the clustering algorithm (output label) and then dividing by $n$. It should be noted that for finding the error we should consider the label switching issue. In fact, for a label function with $l$ clusters without any empty clusters,
there are $l! − 1$ equivalent label functions. If there exists any empty clusters, the amount of equivalent label functions becomes smaller. Therefore, the error is the minimum value of all amounts obtained from counting the mismatch labels between the all equivalent output vectors and the actual label vector. For this calculation, we use Hungarian algorithm which is a combinatorial optimization algorithm [27].

Then, average empirical error for each algorithm is obtained by averaging these errors.

Table 2 Settings for different simulations

<table>
<thead>
<tr>
<th>Simulations</th>
<th># sets for each setting</th>
<th>$d$</th>
<th>$n$</th>
<th>$l$</th>
<th>$k$</th>
<th>max_itr</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation A</td>
<td>1000</td>
<td>2</td>
<td>10</td>
<td>2,3,…,7</td>
<td>1</td>
<td>Inf</td>
<td>1</td>
</tr>
<tr>
<td>Simulation B</td>
<td>1000</td>
<td>2</td>
<td>10</td>
<td>2,3,…,6</td>
<td>1,2,…,4</td>
<td>inf</td>
<td>1</td>
</tr>
<tr>
<td>Simulation C</td>
<td>1000</td>
<td>2</td>
<td>10,100</td>
<td>2,3,…,7</td>
<td>1</td>
<td>10,100,inf</td>
<td>1</td>
</tr>
<tr>
<td>Simulation D</td>
<td>1000</td>
<td>100,200,…,1000</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>inf</td>
<td>1</td>
</tr>
<tr>
<td>Simulation E</td>
<td>1000</td>
<td>10</td>
<td>10,20,…,100</td>
<td>2</td>
<td>1</td>
<td>inf</td>
<td>1</td>
</tr>
<tr>
<td>Simulation F</td>
<td>1000</td>
<td>2</td>
<td>10</td>
<td>2,3,…,7</td>
<td>1</td>
<td>Inf</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: inf = infinity

A set of experiments are implemented, addressing the performance with respect to (A) number of clusters ($l$), (B) radius of the ball search as hamming distance ($k$), (C) maximum number iteration (max_itr), (D) dimension, (E) sample size ($n$), and (F) false assuming hyperparameters in the simulations. Table 2 summarizes the settings for these simulations.
A. Simulation A (effect of different number of clusters)

In the first series of experiments (*simulation A*), we evaluate the performance of the suboptimal algorithm and other classical clustering algorithms versus the number of clusters \( l \) on small point sets of size \( n = 10 \) with arbitrary unknown sized clusters. To this end, we use 1000 sets under the Normal-inverse-Wishart model for each cluster and compute the empirical errors. Classical methods are used as benchmark criteria. FCM, KM, and Hierarchical algorithms are implemented by built-in MATLAB functions with correct number of clusters, \( l \). Also, EM algorithm is implemented using the code written by Michael chen with Correct \( l \) [29]. The idea of random clustering method is identical to the randomly generating true labels for a point set and is written by the author of this thesis.

The result of each clustering algorithm is a vector of labels that determines the partition for the points in the set. Furthermore, as we explained in Section 3.3, for the suboptimal method, we also need to determine \( r, k, \) and \( max_itr \), which are the number of repetitions for repeating the algorithm, the radius of ball search for determining the neighboring partitions, and the maximum number of iterations for IB algorithm respectively. We set these parameters equal to \( r = 1, k = 1, \) and \( max_itr = infinity \) as stated in Table 2. Figure 4 shows average empirical error as a function of the number of clusters \( l \). As can be seen from the figure, the suboptimal algorithm (FCM-IB) based on the equation for the optimal clustering method is superior to all classical clustering methods for each number of clusters. Although in the classical clustering methods, the performance deteriorates considerably when the number of clusters increases, the FCM-IB algorithm depicts a stable performance.
Moreover, we compare the performance of the suboptimal method with Bayes clustering algorithm (optimal algorithm) with natural cost for $l = 2$ and $l = 3$ and maximum a posteriori (MAP) algorithm for $l = 2, 3$, and 4. The computational costs and run time of the optimal algorithm and MAP algorithm are prohibitive even for point sets with moderate sizes. Considering this fact, we simulate the optimal algorithm for only $l = 2, 3$ and MAP for only $l = 2, 3$, and 4. Figure 4 suggests that the performance of the suboptimal algorithm is near optimal and MAP algorithm for the small number of clusters, while it reduces the computation complexities and run time greatly as it can be seen from Figure 5. This fact suggests that the maximum probability partition is often the optimal answer and FCM-IB is able to find the maximum probability partition with $r = 1, k = 1$, and $max_itr = infinity$ instead of getting trapped in local optimum. It also tells us that using the constrained search space for candidate partitions instead of brute force increases the error slightly.

The graphs for average run time with respect to the number of clusters are indicated in Figure 5, where the y-axis is in log scale. The suboptimal algorithm (FCM-IB) requires little time for $l = 2$, where it is comparable to classical clustering methods. As the number of clusters increases, the search space for optimal partitions becomes larger which leads to computation complexities and greater run time.
Figure 4 Average empirical errors with respect to number of clusters

Figure 5 Average run time with respect to number of clusters
B. Simulation B (effect of different radius of ball search)

In the next series of the simulations (Simulation B), the purpose is to show that how the suboptimal clustering algorithm (FCM-IB) works when the radius of ball search (k) increases from k = 1 to k = 4. Similar to Simulation A, we generate 1000 sets for each number of clusters with the settings stated in Table 2. As the radius of the ball search increases, the number of partitions becomes larger. Hence, we need to deal with massive computation costs of RAM usage, and CPU run time especially with a large number of clusters. Figure 6 to Figure 10 display average empirical error and average run time of FCM-IB along with other classical algorithms as a function of k for each cluster l = 2, 3, ..., 6. As it can be seen from these figures, for the suboptimal algorithm, we can conclude that increasing the radius of ball search, k, enhances the performance of the suboptimal algorithm, because the space for searching the partition with maximum probability gets larger and hence, the likelihood of getting trapped in local optima becomes smaller. In our model, increasing k does not result in a considerable improving the performance because the sample size is small (n = 10) and then the number of all partitions are small. It is also trivial that increasing k leads to increasing the run time and RAM usage considerably as by increasing the search space the computations take more time and memory to execute. Therefore, we continue the simulations with k = 1 to save the run time and RAM usage.
Table 3 indicates the graphs for average run time and the standard deviation of run time for the suboptimal method with $l = 2, 3, \ldots, 6$. We utilized an 8 core 3.50 GHz Intel Xenon machine with 16 Gigabyte RAM for these simulations.

Figure 6 Average empirical errors with respect to radius of ball search for $l = 2$
Figure 7 Average empirical errors with respect to radius of ball search for $l = 3$

Figure 8 Average empirical errors with respect to radius of ball search for $l = 4$
Figure 9 Average empirical errors with respect to radius of ball search for $l = 5$

Figure 10 Average empirical errors with respect to radius of ball search for $l = 6$
Table 3 Results for average run times and standard deviation of run times for each algorithm

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1</th>
<th>d</th>
<th>n</th>
<th>k</th>
<th>Mean Run Time (Seconds)</th>
<th>Std Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCM-IB (S, pt, theta, k=1,2,3,4, max_itr=inf, r=1)</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>1</td>
<td>0.0120</td>
<td>0.0080</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>0.0247</td>
<td>0.0118</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>3</td>
<td>0.0529</td>
<td>0.0206</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>4</td>
<td>0.0863</td>
<td>0.0300</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>1</td>
<td>0.0219</td>
<td>0.0099</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>0.0559</td>
<td>0.0209</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>3</td>
<td>0.1961</td>
<td>0.0617</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>4</td>
<td>0.6949</td>
<td>0.1823</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>1</td>
<td>0.0677</td>
<td>0.0299</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>0.2452</td>
<td>0.0816</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>3</td>
<td>7.0043</td>
<td>2.4920</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>4</td>
<td>27.10</td>
<td>6.3812</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>1</td>
<td>1.2932</td>
<td>0.6299</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>12.0525</td>
<td>5.9741</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>3</td>
<td>90.5644</td>
<td>24.4236</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>4</td>
<td>2960.6120</td>
<td>773.5814</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>1</td>
<td>2.7206</td>
<td>1.0495</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>17.2661</td>
<td>4.5885</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>3</td>
<td>1490.7453</td>
<td>367.1776</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>4</td>
<td>12691.8863</td>
<td>1987.1217</td>
</tr>
</tbody>
</table>

Table 3 continued
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1</th>
<th>d</th>
<th>n</th>
<th>k</th>
<th>Mean Run (Seconds)</th>
<th>Std Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayes</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td></td>
<td>4.5515</td>
<td>0.0590</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td></td>
<td>3593.9</td>
<td>235.1961</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td></td>
<td>81.54</td>
<td>3.50202</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0018</td>
<td>0.0005</td>
</tr>
<tr>
<td>MAP</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0416</td>
<td>0.0180</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td></td>
<td>1.7469</td>
<td>0.2193</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td></td>
<td>81.54</td>
<td>3.50202</td>
</tr>
<tr>
<td>K-means</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0047</td>
<td>0.0062</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0061</td>
<td>0.0018</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0050</td>
<td>0.0016</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0054</td>
<td>0.0017</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0053</td>
<td>0.0016</td>
</tr>
<tr>
<td>Expectation Maximization</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0048</td>
<td>0.0231</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0054</td>
<td>0.0158</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0042</td>
<td>0.0037</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0031</td>
<td>0.0019</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0028</td>
<td>0.0015</td>
</tr>
<tr>
<td>Random</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0005</td>
<td>0.0010</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0004</td>
<td>0.0001</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0004</td>
<td>0.0001</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0004</td>
<td>0.0001</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td></td>
<td>0.0004</td>
<td>0.0001</td>
</tr>
<tr>
<td>Algorithm</td>
<td>l</td>
<td>d</td>
<td>n</td>
<td>k</td>
<td>Mean Run Time (Seconds)</td>
<td>Std Run Time</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>----</td>
<td>-------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Fuzzy c-means</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0034</td>
<td>0.0164</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0034</td>
<td>0.0034</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0044</td>
<td>0.0157</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0037</td>
<td>0.0036</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0040</td>
<td>0.0025</td>
</tr>
<tr>
<td>Hierarchical, single</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.000006</td>
<td>0.000017</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.000005</td>
<td>0.000003</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.000005</td>
<td>0.000002</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.000005</td>
<td>0.000002</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.000005</td>
<td>0.000002</td>
</tr>
<tr>
<td>Hierarchical, average</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.000005</td>
<td>0.000002</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0017</td>
<td>0.0009</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0015</td>
<td>0.0007</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0017</td>
<td>0.0006</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0018</td>
<td>0.0005</td>
</tr>
<tr>
<td>Hierarchical, complete</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0018</td>
<td>0.0004</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0011</td>
<td>0.0008</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0017</td>
<td>0.0005</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0015</td>
<td>0.0005</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>0.0017</td>
<td>0.0006</td>
</tr>
</tbody>
</table>
C. Simulation C (effect of different \textit{max_itr})

The purpose of these simulations is to demonstrate the effect of different \textit{max_itr’s} on the performance of the suboptimal algorithm. As in Section 3.3 stated, \textit{max_itr} controls the number of partitions in the ball search. This parameter shows the maximum number of iterations for the algorithm until it converges. If this parameter is infinity, the algorithm can freely be repeated until the convergence achieved. For these simulations, similar to previous simulations, we use 1000 sets for each number of clusters. We ran the suboptimal algorithm for different number of \textit{max_itr} $=10, 100, \text{infinity}$. To show the effect of this parameter on small size point sets and moderate size point sets, we use $n = 10$ and $n = 100$. For the settings used in this simulation, see Table 2. Average empirical error as a function of $l$ for different numbers of \textit{max_itr} for point sets with $n = 10$ and $n = 100$ are illustrated in Figure 11 and Figure 13. Figure 11 is similar to Figure 4 except two lines for \textit{max_itr} $=10,100$. It can be concluded from these figures that if the sample size is small, there is no considerable difference in increasing \textit{max_itr}, since the suboptimal algorithm converges quickly. As the number of points increases, the algorithm needs greater \textit{max_itr} to converge to the optimal partition. Figure 12 reveals that the average run time are almost identical for different \textit{max_itr} in the simulation with $n = 10$, while it can be seen from Figure 14 that the increase in \textit{max_itr} increases run time slightly because in moderate size point sets the convergence needs more iterations.
Figure 11 Average empirical errors with respect to $l$ for small point sets ($n=10$)

Figure 12 Average run time with respect to $l$ for small point sets ($n=10$)
Figure 13 Average empirical errors with respect to $l$ for small point sets ($n=100$)

Figure 14 Average run time with respect to $l$ for small point sets ($n=100$)
D. Simulation E (effect of high dimension)

In this section, the goal is to evaluate the performance of FCM-IB algorithm in high dimension. We simulate FCM-IB algorithm and other classical clustering algorithms with a synthetic data set based on the random labeled point process. The settings for this series of simulation is shown in Table 4.

Table 4 Settings for Simulation D

<table>
<thead>
<tr>
<th>Model</th>
<th>m</th>
<th>ψ</th>
<th>ν</th>
<th>κ</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation E (n=10, l=2, d=100,200,…,1000)</td>
<td>$m_1 = [0, \ldots, 0]_{1 \times d}$</td>
<td>$\psi_1 = I_d$</td>
<td>$\nu_1 = d + 2$</td>
<td>$\kappa_1 = d + 2$</td>
<td>$\alpha_1 = 1$</td>
</tr>
<tr>
<td></td>
<td>$m_2 = [0, \ldots, 0]_{1 \times d}$</td>
<td>$\psi_2 = I_d$</td>
<td>$\nu_2 = d + 2$</td>
<td>$\kappa_2 = d + 2$</td>
<td>$\alpha_2 = 1$</td>
</tr>
</tbody>
</table>

Figure 15 shows that when the number of dimensions increases, the average error remains almost constant. One possible reason might be that the probability of getting trapped in the local maximum does not approach zero as the dimension increases. More investigations on this issue could form a topic of research for the future studies. Therefore, we do not use high dimension in our simulations with the exception of Simulation E. Also, the average empirical errors in Figure 15 illustrate that the suboptimal method based on the equation for Bayes clusterer shows superior performance compared to these methods. Graphs of average run time are provided in Figure 16.

Another important issues in high dimension are technical implementation on machine, such as, huge computation, overflow/underflow problems. Based on Equation (2.20), we are required to compute gamma function for some values of kappa and a determinant of a
huge matrix of size $d \times d$ for term $\psi_l + \psi_l^*$. To avoid under flow/over flow in code, we take log.

Figure 15 Average empirical errors with respect to dimension
E. Simulation E (effect of high sample size)

Simulation E shows the performance of suboptimal method with respect to sample size. To evaluate this, we draw 1000 sets from our model using the settings in Table 5. Figure 17 indicates the average empirical error with respect to sample size for the suboptimal method and other classical clustering algorithms. The graphs show that FCM-IB based on the equation for Bayes clusterer outperforms other classical clustering algorithms, while it maintains a low rate of error. Noticeably, graphs of average run time in Figure 18 indicate that average run time increases with \( n \), making FCM-IB much slower than other algorithms for large \( n \) (\( n \geq 100 \)).
Table 5 Settings for Simulation E

<table>
<thead>
<tr>
<th>Model</th>
<th>m</th>
<th>ψ</th>
<th>ν</th>
<th>𝜅</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation E, (n=10,20,\ldots,100, l=2, d=10)</td>
<td>(m_1 = [0, \ldots, 0]_{1 \times d})</td>
<td>(\psi_1 = 0.5 \cdot l_d)</td>
<td>(\nu_1 = 1)</td>
<td>(\kappa_1 = d + 2)</td>
<td>(\alpha_1 = 10)</td>
</tr>
<tr>
<td></td>
<td>(m_2 = [0, \ldots, 0]_{1 \times d})</td>
<td>(\psi_2 = 0.5 \cdot l_d)</td>
<td>(\nu_2 = 1)</td>
<td>(\kappa_2 = d + 2)</td>
<td>(\alpha_2 = 10)</td>
</tr>
</tbody>
</table>

Figure 17 Average empirical errors with respect to sample size

The Average Empirical Error [%]

<table>
<thead>
<tr>
<th></th>
<th>FCM-IB</th>
<th>KM</th>
<th>EM</th>
<th>Random</th>
<th>FCM</th>
<th>Hier.(Si)</th>
<th>Hier.(Av)</th>
<th>Hier.(Co)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample size</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
</tr>
</tbody>
</table>
F. Simulations F (effect of false modeling hyperparameters)

In this part, we investigate the effect of hyperparameters for a model. For this purpose, we design four different experiments such that in each of them we assume a false hyperparameter. In fact, the settings for these sets of experiments are identical to Simulation A. In the first experiment, we assume only a false covariance hyperparameter \( (\psi_i, i = 1, 2, \ldots, l) \) to observe the effect of this assumption while other hyperparameters are correct. Therefore, we assume the false covariances hyperparameter \( (\psi_i, i = 1, 2, \ldots, l) \) are given by

\[
\psi_i = 2 \cdot I_d, \ i = 1, 2, \ldots, l
\] (4.1)
As shown in Figure 19, even though we assume false $\psi$, the performance is not degraded considerably (specially for small numbers of clusters which the error rate is close to FCM-IB with correct hyperparameters), though this is not true in all cases.

![Figure 19 Average empirical errors with respect to $l$ with (false $\psi$)](image)

Next, in the second experiment, we consider the effect of assuming false mean hyperparameters. Instead of $m_1 = [0, 0]$, $m_2 = [1, 1]$, and to $m_i = [l - 1, l - 1]$, it is falsely assumed that we have $m_1 = 3 \cdot [0, 0]$, $m_2 = 3 \cdot [1, 1]$, and to $m_i = 3 \cdot [l - 1, l - 1]$. As follows from the Figure 20, the performance of FCM-IB with false values of $m$ deteriorates, though this is not true in all cases.
Figure 20 Average empirical errors with respect to $l$ with (false $m$)

Now, we consider the effect of assuming false $\nu$ hyperparameters on the performance of the suboptimal method. Instead of $\nu_i = d + 2, i = 1, 2, ..., l$, we falsely assume that we have $\nu_i = 10 \cdot (d + 2), i = 1, 2, ..., l$. This hyperparameter determines the variance for the distribution of mean,

$$\mu_i \sim \mathcal{N} \left( m_i, \frac{1}{\nu_i} \Sigma_i \right).$$

It can be seen from Figure 21 that error rate decreases slightly with such false values of $\nu_i$, though this is not true in all cases.
The final experiment in this section outlines the performance of suboptimal method while we assume false kappa which is \( \kappa_i = 10 \cdot (d + 2), i = 1, 2, ..., l \). Considering Figure 22, we see performance slightly deteriorates when a false assumption about \( \kappa \) is made, though this is not true in all cases.
Figure 22 Average empirical errors with respect to $l$ with (false $\kappa$)
Chapter 5: Conclusions and future works

In this research study, we have implemented and studied a suboptimal multi-clustering algorithm based on the Bayes clustering algorithm [3] under a random labeled point process. The Bayes clustering algorithm requires to compute the probability mass function for all possible partitions for a point set and calculate a pre-defined cost matrix. These procedures are significantly time and RAM consuming. Therefore, a suboptimal method is vital to make the Bayes Clustering algorithm feasible. Also, allowing multiple clusters is a valuable and interesting feature in a clustering problem which can give us meaningful subsets for a data set. Another particular aspect of this research is that it assumes prior probabilities on clusters based on Dirichlet distribution.

We demonstrated that this algorithm outperforms other classical clustering algorithms. It has also been shown that, in some cases, the performance of the algorithm is near optimal clustering method while it reduces the computation complexities and run time greatly.

We also tested the suboptimal algorithm when the radius of the ball search $k$ increases. The result suggested an improvement in the performance of the suboptimal method. Increasing $k$ makes the search space larger and avoids getting trapped in the local maxima, but requires more computations. Furthermore, other simulations have illustrated that increasing $max_itr$ might reduce average empirical error of the suboptimal algorithm. All
experiments were executed using synthetic data set with multiple clusters in two-dimensional space.

It should be noted that running FCM-IB for a point set with very large number of clusters even in small samples may not be practical. Also, the great computation costs do not enable us to cluster large point sets with multiple clusters. Further research may yield improvements in these issues. There are several remarkable extensions to this work. First, it may be worthwhile to simulate the algorithm with real data and ensure that the algorithm is applicable to real data. In real data, we only have the observations without any further information such as the model and its hyperparameters. Therefore, developing a suboptimal algorithm to find the optimal clusters within the sight of model uncertainty is really important. Second, high dimensional multi-clustering will be a significant activity since real data are often presented in high dimensional space such as gene expression data.
Bibliography


