N° d'ordre: 2014-05-TH

SUPÉLEC

ÉCOLE DOCTORALE STITS

Sciences et Technologies de l'information des Télécommunications et des Systèmes

THÈSE DE DOCTORAT

DOMAINE: STIC

Spécialité: Traitement du Signal

Soutenue le 28 janvier 2014

par:

Jingwen WU

Classification automatique à base de modèle et choix de modèles pour les données discrétisées

Directeur de thèse:	Gilles FLEURY	Professeur, Directeur de la Recherche et des Relations Industrielles, École Supérieure d'Électricité, France
Encadrant:	Hani HAMDAN	Processeur adjoint, École Supérieure d'Électricité, France
Président du jury:	Pierre DUHAMEL	Professeur, Directeur de Recherche, CNRS, France
Rapporteurs:	Francisco CHICLANA	Professeur, De Montfort University, Royaume-Uni
	Hamido FUJITA	Professeur, Iwate Prefectural University, Japon
Examinateurs:	Igor NIKIFOROV	Professeur, Université de Technologie de Troyes Laboratoire LM2S, France

Abstract

This thesis studies the Gaussian mixture model-based clustering approaches and the criteria of model selection for binned data clustering. Fourteen binned-EM algorithms and fourteen bin-EM-CEM algorithms of fourteen parsimonious Gaussian mixture models are developed. These new algorithms combine the advantages in computation time reduction of binning data and the advantages in parameters estimation simplification of parsimonious Gaussian mixture models. The complexities of the binned-EM and the bin-EM-CEM algorithms are calculated and compared to the complexities of the EM and the CEM algorithms respectively. In order to select the right model which fits well the data and satisfies the clustering precision requirements with a reasonable computation time, AIC, BIC, ICL, NEC, and AWE criteria, are adapted to binned data clustering with the developed binned-EM and bin-EM-CEM algorithms. The advantages of the proposed methods are illustrated through experimental studies.

Résumé

Cette thèse étudie les approches de classification automatique basées sur les modèles de mélange gaussiens et les critères de choix de modèles pour la classification automatique de données discrétisées. Quatorze algorithmes binned-EM et quatorze algorithmes bin-EM-CEM sont développés pour quatorze modèles de mélange gaussiens parcimonieux. Ces nouveaux algorithmes combinent les avantages des données discrétisées en termes de réduction du temps d'exécution et les avantages des modèles de mélange gaussiens parcimonieux en termes de simplification de l'estimation des paramètres. Les complexités des algorithmes binned-EM et bin-EM-CEM sont calculées et comparées aux complexités des algorithmes EM et CEM respectivement. Afin de choisir le bon modèle qui s'adapte bien aux données et qui satisfait les exigences de précision en classification avec un temps de calcul raisonnable, les critères AIC, BIC, ICL, NEC et AWE sont étendus à la classification automatique de données discrétisées lorsque l'on utilise les algorithmes binned-EM et bin-EM-CEM proposés. Les avantages des différentes méthodes proposées sont illustrés par des études expérimentales.

Acknowledgements

This PhD thesis was a precious opportunity for me in my career path. I cannot express how thankful I am now in writing this acknowledgement. When I look back at this three years, I see a long path, with so many people. This thesis would not have been possible without the help and support of these kind people.

Firstly, I must offer the profoundest gratitude to my director of thesis professor Gilles FLEURY and my supervisor professor Hani HAMDAN. Thank you both for having selected me as your PhD candidate and having confidence on me.

Professor FLEURY, thank you for all the fruitful discussions that we had and for your support. Thank you for giving your time in listening to me and in solving my problems. Thank you also for all your efforts of helping in finishing this thesis.

Professor HAMDAN, thank you for your three years guidance and for always pushing me to progress. Our collaboration made me much stronger both scientifically and mentally.

I would like to thank the members of the jury for their helpful comments and compliments on my thesis. I am so honored to have all these respectable professors in my jury. Thank you, professor Francisco CHICLANA and professor Hamido FUJITA, for having accepted to be my thesis reviewers and for coming from far to attend my defense. Thank you, professor Pierre DUHAMEL for having accepted to be the president of my defense and for having been supportive during the defense. Thank you, professor Igor NIKIFOROV, for being one of the jury member and for your good suggestions from the view of mathematician.

I would like to acknowledge the academic and technical support from the Ecole Supérieure d'Electricité (Supélec) , University of Paris-Sud and doctoral school STITS, and the financial support from the French Ministry of Higher Education and Research. I want to thank professor Gilles Duc who has followed my academic advancement throughout these years. And I thank Madame Anne Batalie for dealing with all the administrative procedures.

I am grateful to the department SSE, where there are so many kind people that I will never forget. Luc, thank you for not giving up in talking to me despite of my poor French. I feel so lucky to have you in the department to solve all my technical problems. Karine, what an elegant and lovely lady you are! Thank you for helping me a lot when I arrived in Supélec to facilitate the start of my PhD life. Alexandra, thank you for helping me with the trivial problems and being joyful everyday. VUONG, thank you for your passionate salutation every morning and the Vietnamese food that you prepared. Virginie, thank you for being there at the end of my thesis.

There are so many excellent professors in this department. Hana, I wouldn't have spoken French so well without your encouragement during these three years. And thank you so much for your academic advice. Arthur, thank you for being so fun and for helping me whenever I asked. Emmanuel, thank you for offering me help when I needed. Julien, José and Elisabeth, I appreciate our short but interesting chats at lunch. Thank you, Stéphane Font, for the administrative works and some interesting talks.

I am so lucky to have very kind and adorable colleagues in my office. I have spent a very good time with them. Romain, during these years, you have helped me so much. Every time I had a problem or a question academically or personally, you made every effort and you always found a solution. I am so lucky to have a friend like you and I have learned so much from you. Thank you specially for your support throughout my PhD study. Grégory, we have shared many events and they turned out to be good moments: PhD students' meetings, entrepreneurship class, cocktails, etc. I enjoyed talking with you and I have learned a lot from you. I appreciate a lot your accompany during my whole PhD study. Benoît, you are pure and joyful like an angel(I am serious). I felt delighted when you were around. I appreciate your curiosity and I like discussing with you. Thank you for making my days in the office very enjoyable. Lily, thank you for sharing with me the tips of living in France and being a good PhD student. Hope that you are happy in Cambridge and see you soon. I must specially thank Chantal, for the support and exchanges during our PhD and specially during our difficult time. I also want to thank Jad and Loïc, for having accompanied me this journey and for the good moments that we shared. I thank also Alireza, Catalin, Zhiguo, Régis and Wenmeng. I thank also my Chinese friends in Supélec: Zhong Yu, Chu Ning, Zheng Yuling, Chen Long, Ning Baozhu, Liu Junbin, Wang Jing, Wu Yanfu, Yu Peiqing, for being very kind and supportive with me.

I must thank my beautiful friend Francielle and her husbund Augusto for their support during this thesis. I thank also my good friends in China who have been loving me for more than a decade: Ye Shiyun, Deng Xiaojing and Ye Wenhua.

I want to thank my understanding fiancé Quentin who has been so patient, considerate and supportive with me during this important period. I don't remember how many times I became unreasonable because of my stress. Thank you for accepting all these and still being at my side. I also want to thank the family ROBINEAU, especially Hervé, Romy and Yvette, for treating me as a family member and giving me the support and the family warmth.

The most important, I want to thank my parents for their selfless love and endless support throughout my life. Dad, Mum, without you, I would never have been brave enough to pursue my dream in France. Thank you for always respecting and supporting

my choice. And thank you for always being there for me, listening to me and helping me out of all those mess. Dad, you are the model of my life. Thank you for teaching me the life and showing me the direction. I will keep your philosophy in mind and be a good girl. Mum, you are the most beautiful lady and the coolest mum. Thank you for being so gentle and attentive with me. I wouldn't have been so healthy far from home without your nagging. Thank you both for your education and upbringing...For you, I have too many things to thank that I cannot write them all here.

At last I want to thank my grand-parents, uncles, aunts and cousins who have been missing me and taking care of my parents during these years.

Contents

List of Figures

List of Tables

General introduction

Motivation

Clustering is the task of assigning a set of objects into several cohesive groups by measuring the similarity between objects using different measurements. It is an unsupervised learning process since that the number of groups and the forms of groups are unknown in advance. Cluster analysis becomes an important task in data analysis in the last decades. It can be used in many fields, including for example pattern recognition, image analysis, machine learning and information retrieval. It also arouses importance in technology by being applied to a wide variety of domains, such as human genetic clustering, medical imaging, market research and social networks.

In most cases, clustering done in practice is based largely on heuristic but intuitively reasonable procedures [1]. Some famous approaches are hierarchical clustering [2], partitional clustering including for instance the K-means algorithm [3] and model-based approach (see [4] [5] for instance). Among those approaches, model-based approach is commonly used, in which the data are clustered using some assumed modeling structure. This process helps to understand the data distribution. It has also been shown that some of the most popular heuristic clustering methods are in fact approximate estimation methods for some probability models [1]. For example, there are the k-means algorithm and Ward's method. This is one of the reasons why we focus on the model-based clustering in this thesis.

Finite mixture models are a type of density model which comprises a number of probability distributions. When used in clustering, each component probability distribution corresponds to a cluster. One prominent model is the Gaussian mixture model. A variety of approaches to the problem of mixture decomposition have been proposed. Many of them focus on maximum likelihood methods. To obtain data partition using mixture models, two main approaches are basically used: the mixture approach and the classification approach [6]. The mixture approach aims to estimate the mixture model parameters by maximizing the likelihood, and to deduce the data partition from the

estimated parameters. This can be done by the Expectation-Maximization (EM) algorithm [7]. The classification approach aims to estimate simultaneously the parameters and the data partition by maximizing the complete likelihood, and this can be done using the Classification Expectation-Maximization (CEM) algorithm [8].

The EM algorithm can be used to find the parameters of a statistical model in cases where the likelihood equations cannot be solved directly. Typically, there are unobserved data existing or assumed in the model. The EM algorithm is widely used due to its simple concept and its easy implementation. The CEM algorithm is a classification version of the EM algorithm. By inserting a classification step into the EM algorithm, the CEM algorithm obtains a less precise result than the EM algorithm but takes less computation time. However, if the data to be classified are well separated, both the EM and the CEM algorithms provide similar results.

Along with the development of information technology, the amount of data has increased explosively. The standard EM algorithm and the standard CEM algorithm become inefficient in some situations. Many methods are proposed to deal with a vast amount of data. For example, some variants seek to reduce the computational cost of the EM algorithm by reducing the time spent in the E-step [9], [10]. There are also variants known as Sparse EM (SpEM) and Lazy EM (LEM) [11]. One approach for reducing and especially controlling the computation time is to introduce binned data into the EM and CEM algorithms. The corresponding algorithms are called binned-EM [12] [13] and bin-EM-CEM [14]. Binning data is the process of dividing the data space into small regions called bins, then grouping the points (i.e. data) spatially in the bins according to their locations. The idea is to reduce the amount of data (the data size): from the number of points to the number of bins. Experimental results show that the binned-EM and bin-EM-CEM algorithms are respectively faster than the EM and CEM algorithms.

So far, the binned-EM and bin-EM-CEM algorithms are developed basing on the most general Gaussian mixture model, i.e. no restriction is placed on the variance matrices [13] [14]. For the classical EM and CEM algorithms, a set of parsimonious Gaussian mixture models were proposed by Celeux and Govaert [15], in order to fit different datasets. These models are generated according to an eigenvalue decomposition of the components' variance matrices proposed by Banfield and Raftery [16]. By placing certain restrictions on the variance matrices, these parsimonious models have less freedom degrees and thus are simpler. By developing the EM and CEM algorithms of these parsimonious Gaussian mixture models, different data distributions can be adapted by the corresponding parsimonious models which are more specific than the general one. Moreover, due to the simplified estimation, the EM and CEM algorithms become more efficient [15]. Same as in the binned data framework, using the most general model can

sometimes cost loss of time. So in this thesis, to simplify the parameters estimation so thus make the binned-EM and bin-EM-CEM algorithms more efficient, and also to better adapt different data distribution so thus get a more precise result, binned-EM algorithms and bin-EM-CEM algorithms of parsimonious Gaussian mixture models are developed.

Since there are so many potential models to be considered, an important question is raised by many researchers during study: which model should be applied while knowing nothing about the distribution of the dataset? It is important for unsupervised model-based clustering to choose the best model (including the number of clusters) which can precisely represent the data distribution in a reasonable period of time. To answer this question, in standard data framework, many criteria were proposed and studied. For example, there are classical criteria such as information criteria: Akaike Information Criterion (AIC) [17], AIC3 criterion [18] and ICOMP criterion [19]. Schwarz [20] has proposed the famous Bayesian Information Criterion (BIC). Later, Biernacki et al. [21] proposed the Integrated Completed Likelihood (ICL) criterion. Experimental results show that these criteria can successfully choose a relatively simple model which underlines and fits the data distribution. So in this thesis, several criteria are adapted and developed to select the best model for binned data clustering.

Objective and originality of the thesis

The first objective of our thesis is to develop new model-based clustering algorithms which combine the advantages in time reduction of binning data and the advantages in parameters estimation simplification of parsimonious Gaussian mixture models. The binned-EM algorithms of fourteen parsimonious Gaussian mixture models are developed [22] [23] [24]. By adapting EM algorithms of parsimonious Gaussian mixture models to binned data, binned-EM algorithms spend less computation time when dealing with big size datasets. The time complexities of the EM and binned-EM algorithms are calculated and compared. This comparison of time complexity is to show when and how binned-EM algorithm can be faster than the EM algorithm. At another side, the bin-EM-CEM algorithms of fourteen parsimonious Gaussian mixture models are also developed [25] [26]. These algorithms are supposed to be faster than the CEM algorithm. To better study the conditions when the bin-EM-CEM algorithm is faster than the CEM algorithm, the complexities of these two algorithms are calculated and compared. To study and illustrate the performances of the fourteen binned-EM algorithms and the fourteen bin-EM-CEM algorithms of parsimonious Gaussian mixture models, a variety of experiments on simulated data and real data are presented.

The second objective of this thesis is to select the right model for binned data clustering. A right model must fit well the data and satisfy the clustering precision requirements with a reasonable computation time. Several classic criteria are adapted to binned data clustering: BIC, ICL, AIC, NEC, and AWE criteria. They aim to choose a parsimonious model which optimizes the trade-off between the binned data fitting and the model complexity. In this thesis, we focus on BIC and ICL criteria. We associate these two criteria with the fourteen developed binned-EM algorithms and the fourteen developed bin-EM-CEM algorithms, in order to choose the right model for different datasets [27] [28]. Their choices of model and number of clusters are compared using simulated data and real data.

Outline of the thesis

This thesis is structured as follows:

Chapter 1 will highlight the main concepts used in the development of our new algorithms. The definition of clustering will be presented. Then some famous clustering methods will be discussed: hierarchical clustering, k-means algorithm and model-based approach. The Gaussian mixture model-based approach and the two most used model-based clustering approaches with their corresponding EM and CEM algorithms will be detailed. After, parsimonious Gaussian mixture models based on two different concepts will be presented. The AIC, AIC3, BIC, ICL, ICOMP, NEC, and AWE criteria will be described in standard data framework. The concept of binned data and how to obtain binned data from standard data will be discussed. At the end of this chapter, the concepts of the binned-EM algorithm and the bin-EM-CEM algorithm will be explained.

Chapters 2, 3, and 4 are the key chapters of the thesis.

In the Chapter 2, the fourteen binned-EM algorithms of fourteen parsimonious Gaussian mixture models for binned data clustering will be developed. The derivation of the binned-EM algorithm as well as the E-step and the M-step will be detailed. The complexity of binned-EM algorithm will be calculated and compared to the one of the EM algorithm. The parameter estimation by the binned-EM algorithm will be detailed in each case of fourteen parsimonious Gaussian mixture models. To illustrate the performances of these fourteen binned-EM algorithms, two experiments on simulated data will be presented. One experiment is on the data of different structures, while another one is on the data with different bin sizes. To show the practicality of binned-EM algorithm, two experiments on real data, where include French city clustering and image segmentation, will be shown and analysed.

In the Chapter 3, the fourteen bin-EM-CEM algorithms of fourteen parsimonious Gaussian mixture models for binned data clustering will be developed. The E-step, C-step, and M-step will be detailed. The complexity of the bin-EM-CEM algorithm will be calculated and compared to the one of the CEM algorithm. The parameters estimation, especially the variance matrix estimates for fourteen parsimonious Gaussian mixture models will be presented. To better study the performances of the fourteen bin-EM-CEM algorithms, one experiment on simulated data of different structures and another experiment on simulated data with different bin sizes will be shown. At the end of this chapter, two experiments on real data will be presented.

In the Chapter 4, several criteria for model selection will be studied in binned data clustering. The BIC and ICL criteria will be applied with both the fourteen binned-EM and the fourteen bin-EM-CEM algorithms. Experiment on simulated data will be shown in order to compare the model choices and choices of number of clusters obtained by BIC and ICL criteria. The performances of BIC and ICL criteria associated with the fourteen binned-EM and bin-EM-CEM algorithms will be studied on real data too. In order to better compare the model choices of BIC and ICL criteria associated with binned-EM and bin-EM-CEM algorithms, an additional experiment on simulated data will be presented. At the end of this chapter, AIC, AWE, and NEC criteria, will be adapted to binned data clustering. Experiments on simulated data of these three criteria will be shown. Their performances will be compared to the ones of BIC and ICL criteria.

Finally, the general conclusion and the prospective will be presented in the Chapter 4.8.

Chapter 1

State of the art

1.1 Introduction

This chapter aims to go through some basic concepts and some important developed methods in clustering before this thesis. The objective is to give an overview of the existing approaches, so thus to help readers to better understand this thesis' originality which will be presented in the new chapters.

Some related definitions and essential concepts in the domain of cluster analysis will be firstly presented in the Section 1.2. Classification will be distinguished from clustering by its definition. The typical applications of clustering will be listed. In the Section 1.3, three most commonly used approaches for clustering will be reviewed. These approaches are the hierarchical clustering, k-means algorithm and model-based approach. The advantages and the limitations of these approaches will be discussed. The Section 1.4 will present the Gaussian mixture model-based approach, which leads to the core of our study. The mixture approach and the classification approach for data clustering will be discussed. These two approaches are the most commonly used model-based clustering approaches. Their corresponding EM and CEM algorithms will be presented in the Section 1.5. The limitations of both algorithms will be discussed. EM algorithm's extensions will be briefly described.

During the study of model-based approach, some questions might be raised:

1. What are the potential models?
2. Which model is the best-fit for the data?
3. How many clusters?

To answer the first question, the Section 1.6 will present two different kinds of parsimonious Gaussian mixture models. These two kinds of models are based respectively on variance matrix parametrization and on factor analysis model. To answer the second and the third questions, the Section 1.7 will present various famous criteria for clustering model choice, including in particular the choice of number of clusters. The criteria that will be studied are AIC, BIC, ICL, ICOMP, NEC, and AWE criteria.

With the development of information technology, more and more data can be generated. In this situation, the classic EM and CEM algorithms become very slow in dealing with large size of data. The experimental results show that the computation time increases fast along with the increase of data amount. To improve this problem, an approach of extending classical algorithms to binned data was proposed. So in Section 1.8, the definition of binned data will be presented. New algorithms of extending the EM and CEM algorithms to binned data (Binned-EM and bin-EM-CEM) will be detailed. The Section 1.9 will conclude this chapter and lead to the need of our study presented in the Chapter 2.

1.2 What is clustering?

Assignment of a set of objects into several groups can be done through a supervised learning or an unsupervised learning. The supervised learning is called classification. Classification is the problem of identifying a set of observations into several categories, basing on the training result of a subset of observations whose belonging category is known. The unsupervised learning is defined as cluster analysis. It is also called clustering. Clustering is a process of putting a set of observations into several reasonable groups according to certain measure of similarity within each group.

Clustering is a principal technique for data analysis. It can be used in machine learning, pattern recognition, image analysis and information retrieval. It is also an important task in technology since it can be applied to many domains. For example:

- In computer science. Clustering is useful for image segmentation.
- In social network. It allows to recognize communities among large groups of people.
- In marketing. Clustering can put large number of clients into different groups according to their consuming needs, in order to use different publicity strategy.
- In health care. Cluster analysis can be used for medical imaging and medical resource decision making.

- In biology. It helps to cluster different types of plants or different species of animal according to their features.

Deal to its importance, many approaches were proposed to achieve the clustering propose. Different algorithm leads to different clustering result, depending on the definition of what constitutes a cluster in the algorithm. In this next section, three prominent approaches will be reviewed.

1.3 Common approaches for clustering

There are more than 100 published clustering algorithms. Three of the most well-known approaches are hierarchical clustering, k-means algorithm and model-based approach. Each of them bases on different idea of what makes a cluster. These three approaches are widely used in clustering because of their simple adapting abilities and their encouraging results. It is hard to tell which algorithm is the best. It depends on the dataset and the requirement of the clustering result.

1.3.1 Hierarchical clustering

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. It connects objects to clusters by calculating their connectivity. This approach does not give a single partitioning for the data. It provides several clustering results depending on the level at which the clusters merge. Hierarchical clustering can be agglomerative or divisive. Hierarchical agglomerative clustering (*i.e.* HAC) starts with individual observations and aggregates them into clusters. Hierarchical divisive clustering (*i.e.* HDC) starts with the complete dataset and divides it into partitions. In general, HAC is more often used than HDC. In this part, we will focus on the HAC.

Hierarchical clustering does not require us to pre-specify the number of clusters. The process of an HAC clustering of n objects is described as follows [29]:

1. We assume that each object constitutes a cluster. We begin from n clusters corresponding to n objects respectively. We compute the distances between each two clusters, which equals to the distances of each two objects.

2. Find out two most similar classes (A, B) and emerge them into one class $C = A \cup B$.

3. Remove the clusters A and B. Calculate the dissimilarities between the new class C and other classes.

4. Repeat step 2 and step 3 until all the objects are emerged into only one class of n objects.

Step 3 can be done by different methods. If we consider the shortest distance of any member of one cluster to any member of another cluster to be the distance between two clusters, this method is called single-linkage clustering. On the contrary, when the distance between two clusters is based on the two least similar points in these two clusters, the method is called complete-linkage clustering. The third method is called average-linkage clustering, where the distance between two clusters is the average of distances between members of these two clusters. In the Ward's method [2], at each step we find the pair of clusters that leads to a minimum increase in total within-cluster variance after merging. This increase is a weighted squared distance between cluster centers.

An HAC clustering is typically visualized as a dendrogram. Each object is represented by a horizontal line. The y-coordinate of the horizontal line is the similarity of the two clusters that were merged. A pre-specified number of clusters is not required in HAC. Each step of HAC represents different level of clustering. The root represents the whole dataset. An internal node represents a cluster at the present step. The height of the internal node represents the distance between its two child nodes.

An example of an HAC clustering is shown in the Figure 1.1. It stars with a set of five individual observations. At the beginning of the process, each observation is considered as one cluster, which is stated in the x-coordinate. The y-coordinate indicates the distance (or similarity) of the two clusters that are merged. After four steps (levels) of clustering, five observations are finally aggregated into one cluster.

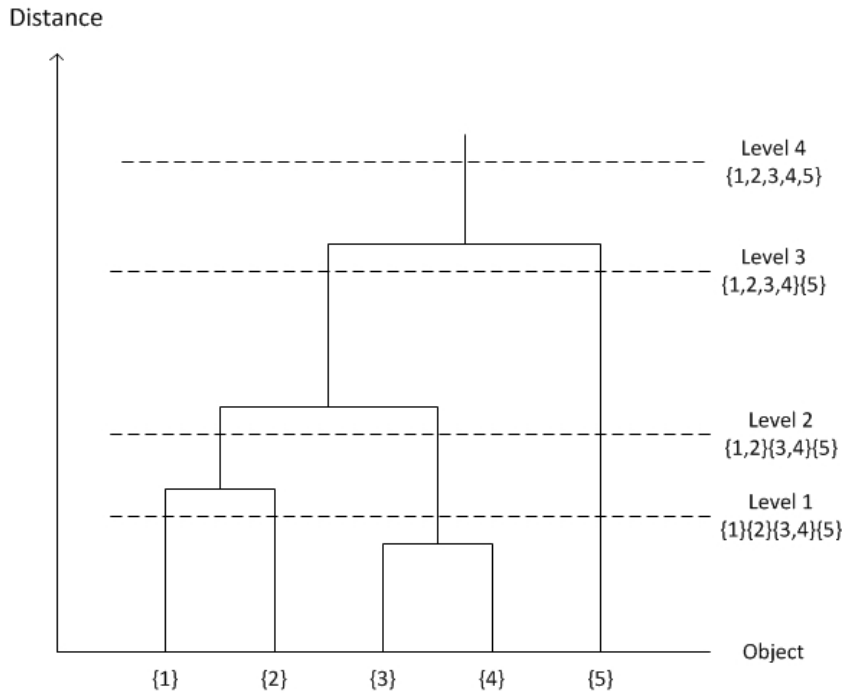


FIGURE 1.1: An example of hierarchical agglomerative clustering. It stars with five individual observations, goes through four levels of clustering to aggregates them into one cluster.

Hierarchical clustering has some limitations. For example, there is no explicit clusters and no optimal clusters can be defined. Despite of those limitations, many advantages make hierarchical clustering a widely used method: the number of clusters is not required in advance and there is no input parameter.

1.3.2 K-means algorithm

K-means [3] is one of the simplest and oldest unsupervised learning algorithms. It is also called Lloyd's algorithm [30], which is based on the centroid model. The objective of K-means clustering is simply to partition a set of observations into K clusters. The centroids of K clusters are defined in advance. The centroids are better to be well separated to each other so as to obtain a good clustering result. The procedure of the k-means algorithm can be summarized as follows:

1. Define K points in the whole space of the set of observations. These K points represent the initial centroids of K clusters.

2. Calculate the distances between the observations and the centroids.

3. Move each observation to the closest cluster.

4. After assigning all the observations, redefine the centroids of the K clusters.

5. Repeat steps 2, 3 and 4 until the positions of K centroids don't move anymore.

Comparing to the hierarchical clustering, K-means clustering has some advantages. With a large number of variables, K-Means may be computationally faster than hierarchical clustering (if K is small). K-Means may produce tighter clusters than hierarchical clustering, especially if the clusters are globular.

But at the same time, k-means clustering has several disadvantages too. For example, it is difficult to calculate the quality of the clustering result. It is hard to know what K should be. Different initial partitions can result in different final clusters. Sometimes k-means clustering converges to local optimum. The clustering result can highly depend on the initial centroids. Thus, several methods were proposed to improve this drawback. For example, there are the k-medoids algorithm [31] and the k-medians clustering [32]. The k-medoids algorithm chooses data points as centers and the k-medians clustering calculates the median instead of calculating the mean for each cluster to determine its centroid. Sometimes the k-means++ algorithm [33] is used to choose the initial values. In order to obtain the global optimum, it is common to run the K-means algorithm multiple times with different starting conditions. But sometimes K-means can be very slow to converge.

1.3.3 Model-based approach

The model-based approach aims to put the objects following the same distribution into the same group. This method supposes that the data follows certain probability distribution model. It is achieved by estimating the mixture model parameters which maximize the likelihood of potential model. This approach over-fits the dataset if there is no constraint on the complexity of the model. Despite its overfitting problem, model-based approach is strong comparing to the hierarchical clustering and k-means algorithm, because it provides not only the clusters, but also the mixture model which helps us to better understand the data distribution.

Two most commonly used model-based clustering approaches are mixture approach and classification approach. In the next section, we will focus on the Gaussian mixture model-based mixture approach and classification approach.

1.4 Model-based clustering approaches: mixture approach and classification approach

To obtain the model parameter estimation, we have two most well-established methods via maximum likelihood, namely the mixture approach (also called Maximum Likelihood estimation) and the classification approach (also called Classification Maximum Likelihood estimation). These two approaches are studied and compared by many authors. Generally speaking, the mixture approach is aimed to maximize the likelihood over the mixture model parameters via Estimation Maximization (EM) algorithm, while the classification approach is aimed to maximize the likelihood over the mixture model parameters and the origin identifying labels of each observation via Classification Estimation Maximization (CEM) algorithm.

The mixture approach was recommended and analyzed by R. A. Fisher between 1912 and 1922 [34]. Reviews of the development of maximum likelihood have been provided by a number of authors. Mixture approach estimates the parameters by using the Expectation Maximization (EM) algorithm and obtains the partition by using Maximum A Posteriori probability (MAP) estimation.

In the mixture approach, we estimate the parameters π_k and θ_k, in order to maximize the log-likelihood:

$$L(\mathbf{\Phi}|\mathbf{x}) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}; \theta_k)$$

In the restricted case where the mixing proportions π_k are considered to be equal, the log-likelihood has the form:

$$L(\mathbf{\Phi}|\mathbf{x}) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} f_k(\boldsymbol{x}; \theta_k)$$

The parameters θ_k are generally obtained by using the EM algorithm.

The classification approach is also called Classification Maximization Likelihood (CML), is proposed by Scott and Symons [6]. It aims to maximize the likelihood over the mixture model parameters and over the identifying labels of the mixture origin for each data. The mixture origin is denoted as $\boldsymbol{z}_i = (z_{ik}, k = 1, \ldots, K)$, where $z_{ik} = 1$ when $\boldsymbol{x}_i$ belongs to the component k, otherwise $z_{ik} = 0$. Two different CML criteria were proposed: the restricted CML criterion and the unrestricted one. In the restricted case, the mixing

proportions π_k are assumed to be equal. The CML criterion has the form [6]:

$$L_{CR} = \sum_{k=1}^{K} \sum_{x_i \in P_k} \log f(x_i, \mu_k, \Sigma_k)$$

where $P = (P_1, \ldots, P_K)$ is a partition of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ where $P_k = x_i / z_{ik} = 1$.

In the unrestricted case, the mixing proportions π_k of components are different. The CML criterion is [35]:

$$L_{CR} = \sum_{k=1}^{K} \sum_{x_i \in P_k} \log(\pi_k f(x_i, \mu_k, \Sigma_k))$$

CML approach is carried out basing on Classification Expectation Maximization (CEM) algorithm.

In the same study of Scott and Symons [6], they suggested that with little information of the size of clusters, the clustering procedures based on maximum likelihood may have a tendency to partition the sample into groups of similar size. The maximum likelihood estimates are discussed under two situations: equal covariance matrices and unequal covariance matrices. When we assume covariance matrices equal, maximizing the log likelihood function equals minimizing $|\mathbf{W}_y|$, the determinant of the within-groups sum of squares matrix. When the covariance matrices are very different among clusters, maximizing the likelihood equals to minimizing $tr(\mathbf{W}_y^{-1}\mathbf{B}_y)$. The algorithm was tested on the well-known Irish data [36]. The $\mathbf{W}_y$ criterion gave good results when the data is composed of two groups of equal size. And it shows that the criterion prefers to partition the sample into components of the nearly same size.

Usually, when studying these two approaches, the mixing proportion in the standard classification approach is restricted as equal, while there is not restriction on the mixing proportion in the mixture approach. Until 1981, Symons [35] proposed a general classification maximum likelihood criterion of free mixing proportions. In order to study the performance of both approaches under the same conditions, Celeux and Govaert [37] presented a comparison of the practical behaviors of the mixture and the classification approaches for clustering by maximum likelihood. They compared both approaches in two conditions of assuming the models respectively of equal mixing proportions and of unknown mixing proportions. The result turns out that the classification approach is favored in the case of small sample, and the mixture approach is preferred in dealing with large sample. They also found out that the approaches assuming equal mixing proportions are more reliable and stronger than un-restricted approaches in practical applications.

However, the CML approach has some limitations: firstly, it suggests the same features for all the clusters; secondly, it only fits for Gaussian distributions; thirdly, noise is not allowed in the hypothesis. To solve the first problem, parsimonious models were proposed according to a parametrization of the variance matrices of the mixture components proposed by Banfield and Raftery [16]. These parsimonious models will be emphasized in the later part of this chapter.

For the distribution restriction, a framework for the Uniform-Normal case is introduced. To deal with the noise, a Poisson process was discussed:

For a dataset, we should allow the possibility that some observations are not from any cluster of the model. We can consider this kind of data as noise. To include such observations, Banfield and Raftery [16] assume that the dataset arises from a Poisson process with intensity ν. The likelihood

$$L(\boldsymbol{\theta}, \boldsymbol{\gamma}) = \prod_{i=1}^{n} f_{\gamma_i}(\boldsymbol{x}_i; \boldsymbol{\theta})$$

is modified as:

$$L(\boldsymbol{\theta}, \nu, \boldsymbol{\gamma}) = \frac{(\nu A)^{n_0} e^{-\nu_0 A}}{n_0!} \prod_{i \in E} f_{\gamma_i}(\boldsymbol{x}_i; \boldsymbol{\theta})$$

where $E = \cup_{k=1}^{G} E_k$, $n_0 = n - \sum_{k=1}^{G} n_k$ and A is the hyper-volume of the region from which the data have been drawn.

Also, to deal with the noise problem, Banfield and Raftery [16] has also proposed the hierarchical clustering methods which require the shape parameter α to be defined before clustering and $\mathbf{A}_k = diag\{1, \alpha\}$.

Bensmail and Meulman [38] assumed a mixture of a Gaussian distribution satisfying and noise distributed as a homogeneous spatial Poisson process with constant rate π_0. The mixture distribution likelihood is:

$$p(\theta_1, \ldots, \theta_K; \pi_0, \pi_1, \ldots, \pi_K | \boldsymbol{x}) = \prod_{n=1}^{n} [\frac{\pi_0}{\Lambda} + \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}_i | \theta_k)]$$

where $\Lambda > 0$ is the volume of the finite domain which is defined as:

$$\Lambda = \prod_{j=1}^{p} (\max_{i=1,\ldots,n} \{x_{ij}\} - \min_{i=1,\ldots,n} \{x_{ij}\})$$

1.5 EM and CEM algorithms

1.5.1 EM algorithm

Estimation Maximization (EM) algorithm was first time officially proposed by Dempster et al. [7]. Later Wu [39] has corrected a flawed convergence analysis in this paper. The EM algorithm is widely used because of its simplicity and easy implementation. It is an efficient iterative procedure to compute the Maximum Likelihood (ML) estimate in the presence of missing (or unobserved) data.

Since it is not easy to maximize the log-likelihood directly, EM algorithm maximizes the expectation of complete log-likelihood instead. The complete data in EM algorithm are considered to be $(\mathbf{x}, \mathbf{z})$. $\mathbf{z}$ is the missing data indicating the mixture component origin label of each observation. $\mathbf{z} = (z_1, \ldots, z_n)$ where $z_i = k$ when x_i belongs to the component k. The complete log-likelihood takes the form:

$$CL(\mathbf{\Phi}, \mathbf{z}|\mathbf{x}) = \sum_{k=1}^{K} \sum_{k=1}^{K} z_{ik} \log \left(\pi_k f_k(\boldsymbol{x}; \theta_k)\right)$$

EM algorithm starts from the initial parameters $\theta^{(0)}$, then computes the Expectation step (E step) and the Maximization step (M step) iteratively:

- E step: Calculate the expected value of the complete log-likelihood function, with respect to the conditional distribution of $\mathbf{z}$ given $\mathbf{x}$ under the current estimate of the parameters $\mathbf{\Phi}$:

  $$Q(\mathbf{\Phi}|\mathbf{\Phi}^{(q)}) = E[\log CL(\mathbf{\Phi}, \mathbf{z}|\mathbf{x})]$$

  i.e. Calculate the posterior probabilities $t_{ik}^{(q)}$ of $\boldsymbol{x}_i$ belonging to the kth component:

  $$t_{ik}^{(q)} = \frac{\pi_k^{(q)} f_k(\boldsymbol{x}; \theta_k^{(q)})}{\sum_l \pi_l^{(q)} f_l(\boldsymbol{x}; \theta_l^{(q)})}$$

- M step: Find the parameter $\mathbf{\Phi}^{(q+1)}$ that maximizes the expectation:

  $$\mathbf{\Phi}^{(q+1)} = \arg\max_{\mathbf{\Phi}} Q(\mathbf{\Phi}|\mathbf{\Phi}^{(q)})$$

EM algorithm is widely used thanks to its conceptual simplicity and easy implementation. But EM algorithm has several limitations which need to be improved. For example, the fact that its result is highly dependant on the initial data. Sometimes it provides only the local optima. It might require many iterations and long computation time.

Large quantity of data and high dimensionality of data stress out this low execution problem. To improve these limitations, EM algorithm's extensions were proposed.

1.5.2 EM algorithm's extensions

To solve the slow convergence problem of the EM algorithm, Moore [9] proposed a very fast new algorithm basing on the multi-resolution kd-trees of Moore et al. [40]. This new algorithm succeeds in reducing the computational cost of EM-based clustering.

It is reported that, apart from the slow convergence, the EM algorithm has another limitation: the EM algorithm depends highly on the initial values. Bad initiation can lead to local maximum likelihood, instead of the global maximum. For this problem, several extension extensions of the EM algorithm were proposed.

One of the algorithms is called SEM algorithm which was proposed by Celeux and Diebolt [41] in order to improve some limitations of EM algorithm. In fact, a S step is inserted between the E and M steps of EM algorithm. In the S step, each observation x_i is assigned randomly to one of the clusters P_k by the posterior probabilities $t_k^m(x_i)$ that x_i belongs to P_k. It claims that the SEM algorithm has several improvements, with respect to the EM algorithm:

- It is sufficient to know an upper bound on the number of clusters;
- The result is independent from the initial parameters;
- The speed of convergence is improved.

Another algorithm is based on the SEM algorithm and it is called CAEM algorithm (Classification Annealing EM). Comparing to the SEM algorithm, it replaces the posterior probabilities $t_k^m(x_i)$ by the scores $s_k^m(x_i)$ which is defined as:

$$s_k^m(x_i) = \frac{\{p_k^m f(x_i, a_k^m)\}^{1/\tau_m}}{\sum_{k'=1}^{K}\{p_{k'}^m f(x_i, a_{k'}^m)\}^{1/\tau_m}}$$

where τ_m $(m \geq 0)$is a sequence of temperatures decreasing to zero when m tends to infinity from $\tau_0 = 1$.

Dasgupta and Raftery [42] has extended the model-based clustering methodology of Banfield and Raftery [16] mclust by refining the final partition using the EM algorithm which is called mclust-em. The mclust-em algorithm has two steps, firstly it executes mclust to obtain an initial partition of the data for each number of clusters. Secondly, it then executes the EM algorithm to obtain the maximized mixture likelihood in order to

make a model choice by BIC criterion. In mclust-em, the shape parameter α is estimated by including α at the M step of the EM algorithm. It turns out that mclust-em procedure outperforms the mclust procedure.

Fraley and Raftery [43] proposed a clustering methodology based on multivariate Gaussian mixtures in which BIC is used for comparison of models. Within this new approach, partitions of clusters are determined by a combination of hierarchical clustering and the EM algorithm. The agglomerative hierarchical clustering is used to initialize the EM algorithm. Noise and outliers are modeled by adding a Poisson process component. As real data experiment, they applied this new approach on a three-dimensional dataset used for diabetes diagnosis. The results showed that this approach gives much better performance than existing methods although it has two limitations: it cannot be applied to large datasets because of the big computation time problem; other models may be more suitable than multivariate normal distribution models in some situations.

Biernacki et al. [44] discussed how to choose starting values for the EM algorithm to get the highest likelihood in the framework of multivariate Gaussian mixture models. Their method basically considered a three-step (search/run/select) strategy for maximizing the likelihood. The idea is to firstly build a search method to generate p initial positions. Then run the EM algorithm for certain number of iterations at each initial position. At last select the result which provides the highest likelihood among the p results. The CEM and SEM algorithms can be used in the search step for finding the initial points. In this paper, a strategy of initiating EM algorithm by short runs of EM was also recommended. This strategy is simple and has good performance in many cases where no particular mixture model can be fitted to the data. But it is hard to define which strategy among those proposed one is the best and it is also difficult to tell which particular strategy should be used in each specific situation.

To solve the local optimum problem of EM algorithm, besides by choosing the starting values, another solution was proposed by Celeux and Govaert [8]. By setting the optimization-based clustering methods under the classification maximum likelihood approach, a general Classification EM algorithm was defined and studied by Celeux and Govaert [8]. Two stochastic algorithms are also developed, deriving from this general Classification EM algorithm. These two algorithms have less initial-position dependence compared to the classical optimization clustering algorithms. They are supposed to obtain the local optimum solutions from any initial position. However, both of the algorithms need a large number of iterations to ensure the best result.

1.5.3 CEM algorithm

CEM algorithm is considered as classification version of EM algorithm [8]. One significant difference between CEM and EM algorithms is that CEM algorithm inserts a classification step between Expectation step and Maximization step in order to accelerate the execution. Thus CEM algorithm is theoretically supposed to execute faster than EM algorithm. The procedure is described as follows:

- E-step (Expectation): Calculate the poster probabilities $t_{ik}^{(q)}$ of $\boldsymbol{x}_i$ belonging to the kth component, same as in the EM algorithm;
- C-step (Classification): Obtain $\boldsymbol{z}_i^{(q)}$ which indicates the mixture origin of each $\boldsymbol{x}_i$:

$$z_{ik}^{(q)} = \arg\max_k t_{ik}^{(q)}$$

  The biggest $t_{ik}^{(q)}$ (k=1,...,K) is replaced by 1, others are replaced by 0.
- M-step (Maximization): Find the parameter $\boldsymbol{\Phi}^{(q+1)}$ that maximizes the expectation.

1.6 Parsimonious models

Parsimony is a 'less is better' concept of frugality, economy, stinginess or caution in arriving at a hypothesis or course of action. The word derives from Middle English parcimony, from Latin parsimonia, from parsus, past participle of parcere: to spare. In science, parsimony is preference for the least complex explanation for an observation. Parsimony is also a factor in statistics: in general, mathematical models with the smallest number of parameters are preferred because each parameter introduced into the model adds some uncertainty to it. Different kinds of parsimonious models were proposed according to different theories. Here we will mention two groups of parsimonious Gaussian mixture models. Before introducing these parsimonious models, let's firstly present the definition of Gaussian mixture model.

1.6.1 Definition of Gaussian mixture model

A Gaussian mixture model is represented as a probability density function in the form of a weighted sum of Gaussian component densities. Equation 1.1 describes the distribution of an independent sample $\mathbf{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ issued from a K-component mixture

distribution:

$$f(\mathbf{x};\mathbf{\Phi}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x};\theta_k) \tag{1.1}$$

with $\mathbf{\Phi} = (\pi_1,\ldots,\pi_K,\theta_1,\ldots,\theta_K)$, where π_k $(k = 1,\ldots,K)$ denote the mixing proportions of the mixtures ($0 < \pi_k < 1$ and $\sum_{k=1}^{K} \pi_k = 1$), and $\theta_k = (\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)$ $(k = 1,\ldots,K)$ are the parameters of Gaussian distribution functions f_k of components: mean vectors $\boldsymbol{\mu}_k$ and variance matrices $\mathbf{\Sigma}_k$. The Gaussian distribution function f_k is defined in a d-dimensional space:

$$f_k(\boldsymbol{x};\boldsymbol{\mu}_k,\mathbf{\Sigma}_k) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}_k|^{1/2}} \exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\mathbf{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k))$$

Figure 1.2 gives an example of the density of a one-dimensional Gaussian mixture distribution with two components of equal proportions, while Figure 1.3 shows the density of a two-dimensional Gaussian mixture distribution with two components of equal proportions.

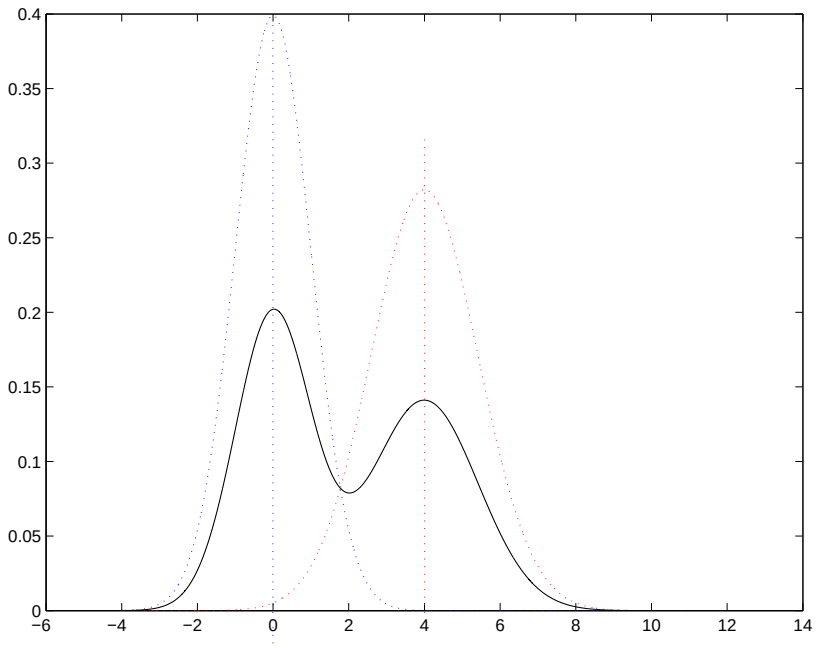


FIGURE 1.2: Density of a one-dimensional Gaussian mixture distribution with two components of equal proportions.

1.6.2 Models based on variance matrix parametrization

To be more adaptable to datasets of different distributions, parsimonious Gaussian mixture models were developed. They are achieved by applying a parametrization of the

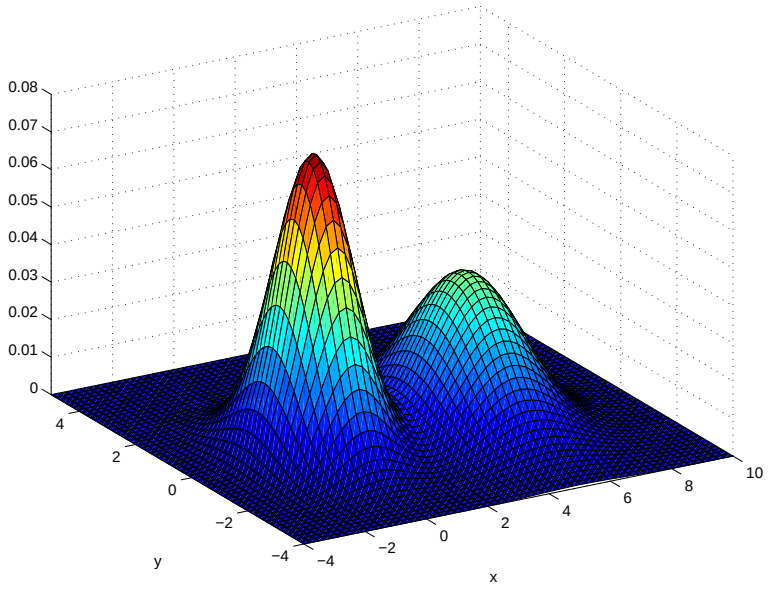


FIGURE 1.3: Density of a two-dimensional Gaussian mixture distribution with two components of equal proportions.

variance matrix $\mathbf{\Sigma}_k$ [16]:

$$\mathbf{\Sigma}_k = \lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T$$

where $\lambda_k = |\mathbf{\Sigma}_k|^{1/d}$ determines the volume of the kth cluster. Its orientation is determined by $\mathbf{D}_k$ which is the matrix of eigenvectors of $\mathbf{\Sigma}_k$. And $\mathbf{A}_k$ determines its shape. $\mathbf{A}_k$ is a diagonal matrix with the normalized eigenvalues of $\mathbf{\Sigma}_k$ in a decreasing order on the diagonal, and $|\mathbf{A}_k|$=1. In the Figure 1.4, an example of a Gaussian cluster in two dimensions is shown in the form of ellipse. The Gaussian cluster has rotated an angle of α from the horizontal line. Thus the variance matrix of this cluster is $\mathbf{D} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$. The semi-major axis equals $\sqrt{\lambda a}$ while the semi-minor axis equals $\sqrt{\frac{\lambda}{a}}$. The shape matrix $\mathbf{A} = \begin{pmatrix} a & 0 \\ 0 & 1/a \end{pmatrix}$.

According to this variance matrix decomposition, eight general parsimonious models were proposed [15]. They are $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$, $[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$, $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, $[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$, $[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$, $[\lambda \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$, $[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$. Diagonal family and spherical family of parsimonious models were proposed by putting more restrictions on certain parameters. If the angle α is defined to equal to 0 or $\pi/2$, the matrices $\mathbf{D}_k$ have only one entry, neither 1 or -1 in each row and each column, 0 elsewhere. Then $\mathbf{D}_k$ and $\mathbf{A}_k$ can be be merged into one matrix: $\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T = \mathbf{B}_k$. Thus, we have a simpler parametrization of variance matrix $\mathbf{\Sigma}_k = \lambda_k \mathbf{B}_k$. By doing this, we obtain four diagonal models: $[\lambda \mathbf{B}]$, $[\lambda_k \mathbf{B}]$, $[\lambda \mathbf{B}_k]$ and $[\lambda_k \mathbf{B}_k]$. To achieve a simpler family, the shapes of clusters are assumed

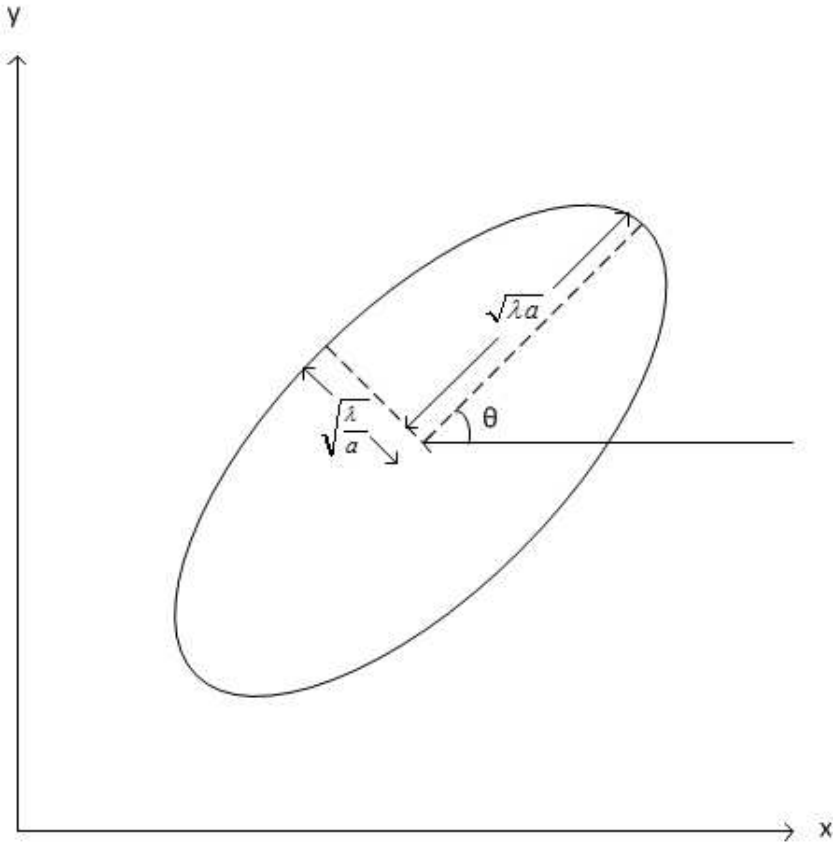


FIGURE 1.4: Interpretation of Gaussian mixture model parameter.

as spherical. $\mathbf{A}_k = diag(1,1) = \mathbf{I}$. There is no interest to consider the orientation in this situation. Then we have $\mathbf{\Sigma}_k = \lambda_k \mathbf{I}$. Then two more spherical models are generated: $[\lambda \mathbf{I}]$ and $[\lambda_k \mathbf{I}]$.

1.6.3 Models based on factor analysis model

Lately, Mcnicholas and Murphy [45] have proposed new models basing on assuming a latent Gaussian model which are closely related to the factor analysis model.

The factor analysis model assumes that a p-dimensional random vector X is modeled using a q-dimensional ($q < p$) vector of latent factor (*i.e.* unobserved factors). It is expressed in the form as follows [46]:

$$X = \mu + \mathbf{\Lambda} U + \epsilon$$

where $\mathbf{\Lambda}$ is a $p \times q$ matrix of factor loadings, the factors $U \sim N(0, \mathbf{I}_q)$ and $\epsilon \sim N(0, \mathbf{\Psi})$, where $\mathbf{\Psi} = diag(\psi_1, \ldots, \psi_p)$. The marginal distribution of X basing on this model is $N(\mu, \mathbf{\Lambda\Lambda}' + \mathbf{\Psi})$.

Later the factor analysis model was extended by developing the mixture of factor analyzers model, which is assumed as a mixture of Gaussian distributions with a factor analysis covariance structure [47]. The density function of factor analyzers model is:

$$f(x_i) = \sum_{g=1}^{G} \frac{\pi_g}{(2\pi)^{p/2}\mathbf{\Psi}_g^{1/2}} \exp\{ -\frac{1}{2}(x_i - \mu_g - \mathbf{\Lambda}_g u_i)^T \mathbf{\Psi}_g^{-1}(x_i - \mu_g - \mathbf{\Lambda}_g u_i)\}$$

where π_g denotes the proportion of the component g, μ_g the mean parameter, $\mathbf{\Lambda}_g$ the loading matrix and $\mathbf{\Psi}_g$ the noise matrix. The mixtures of factor analyzers model differs between the situations that the $\mathbf{\Psi}_g$ is constrained to be equal [47] or unequal [48]. Mcnicholas and Murphy [45] proposed to unify these Gaussian mixture models by applying constraints on the $\mathbf{\Lambda}_g$ and $\mathbf{\Psi}_g$ matrices, and if $\mathbf{\Psi}_g$ are isotropic: $\mathbf{\Psi}_g = \psi_g \mathbf{I}_p$ [49]. Thus, eight different parsimonious Gaussian mixture models were derived. These eight models are presented in the Table 1.1.

Model ID	Loading Matrix $\mathbf{\Lambda}_g$	Error Variance $\mathbf{\Psi}_g$	Isotropic
CCC	Constrained	Constrained	Constrained
CCU	Constrained	Constrained	Unconstrained
CUC	Constrained	Unconstrained	Constrained
CUU	Constrained	Unconstrained	Unconstrained
UCC	Unconstrained	Constrained	Constrained
UCU	Unconstrained	Constrained	Unconstrained
UUC	Unconstrained	Unconstrained	Constrained
UUU	Unconstrained	Unconstrained	Unconstrained

TABLE 1.1: Eight parsimonious Gaussian models with different covariance structures

The Alternating Expectation Conditional Maximization (AECM) algorithm [50], which is an extension of EM algorithm, is used for fitting these parsimonious Gaussian mixture models.

1.7 Criteria for model choice

Parsimonious Gaussian mixture models were proposed in order to better fit data of different distributions. When data follows a simple distribution in a two dimensional space, we could guess the potential model by observing. But if the data is of multi-dimension, or if the components of data are well-mixed, it is not easy to observe the right model which corresponds to the data. The experimental result of parsimonious model has shown that the best result is generally obtained by the model representing the data distribution [15]. Model choice becomes an essential subject in model-based

clustering. The choice of model concludes two contents: the choice of model and the choice of number of clusters.

In this thesis, model selection is restricted in choosing among the Gaussian mixture models. Why the choice of model is important? An over complex model leads to a complicate parameter estimation. An over simple model cannot represent correctly the data distribution. The importance of defining the number of clusters is also obvious: Too many clusters cannot provide a necessary clustering. For example, when the number of clusters is equivalent to the number of points, the meaning of clustering is lost in this case. But when there are too few clusters, some small clusters might be ignored or some overlapped clusters might be merged as one.

Many criteria were proposed for clustering model choice. There are classical criteria such as information criteria: Akaike Information criterion (AIC) [17], AIC3 criterion [18] and ICOMP criterion [19]. Schwarz [20] has proposed the famous Bayesian Information criterion (BIC). Later Biernacki et al. [21] proposed a BIC-like criterion: Integrated Completed Likelihood (ICL) criterion. There is also classification criterion NEC [51].

Information criteria (AIC, AIC3 and BIC) are based on the maximum likelihood with certain penalty of the number of parameters of the model. As the complexity of the model increases, the model becomes more capable of adapting to the characteristics of the data and the maximum likelihood is bigger. Thus, selecting the best fit model by selecting the one that maximizes the likelihood certainly leads to choose the most complex model. A balance between the data information represented by the model and the complexity of the model needs to be defined.

1.7.1 Criteria based on maximum likelihood

It is said that the maximum likelihood is not a good criterion in choosing the right model for clustering. Let's review the definition of likelihood to understand why. In mixture model, $\mathbf{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is assumed to be an independent sample issued from a K-component mixture distribution defined on $\mathbb{R}^p$:

$$f(\mathbf{x}; \mathbf{\Phi}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}; \theta_k)$$

with $\mathbf{\Phi} = (\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K)$, where π_k $(k = 1, \ldots, K)$ denote the mixing proportions of the mixtures ($0 < \pi_k < 1$ and $\sum_{k=1}^{K} \pi_k = 1$), and $\theta_k = (\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)$ $(k = 1, \ldots, K)$ are the parameters of Gaussian distribution functions f_k of components: mean vectors

$\boldsymbol{\mu}_k$ and variance matrices $\boldsymbol{\Sigma}_k$. The log-likelihood of the sample is defined:

$$L(M,K) = \sum_{i=1}^{n} \ln(\sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x};\theta_k))$$

where M represents the mixture model which maximizes the likelihood.

From its definition, $L(M,K)$ increases along with the complexity of the mixture model M and the number of components K. Doing model choice by $L(M,K)$ only results in getting the most complex model with the most number of components. Thus, some approximate maximum likelihood criteria were proposed.

There is the fuzzy classification likelihood:

$$C(M,K) \quad = \quad L(M,K) - E(M,K)$$

with the entropy term:

$$E(M,K) \quad = \quad -\sum_{k=1}^{K}\sum_{i=1}^{n} t_{ik} \log t_{ik}$$

where t_{ik} is the estimated conditional probability that x_i arises from the kth mixture component.

In the context of choosing the number of clusters in clustering approach, classification likelihood (CL) method was studied. CL is a penalized term of the likelihood:

$$CL(\theta, \boldsymbol{z}) = L(\theta) - LP(\theta, \boldsymbol{z})$$

where $LP(\theta, \boldsymbol{z}) = -\sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik} \ln(t_{ik})$. And CL is also a penalization of the k-means criterion $tr(\mathbf{W}_k)$. The relationship between CL and k-means criteria is:

$$CL_K = -\frac{nd}{2} \ln(tr(\mathbf{W}_K)) - n \ln(K) + cst.$$

Since the classification likelihood is a penalized form of likelihood and K-means, it can be considered as an approach to choosing the number of clusters. Several strategies are suggested:

1. CL criterion. Maximize directly $CL(\theta, \boldsymbol{z})$
2. CLM criterion. First estimate the parameters θ by maximum likelihood and then penalize the maximum log-likelihood by the term LP_K, where $LP_K = \min_{\boldsymbol{z}} LP(\hat{\theta}_L(K), \boldsymbol{z})$.

3. An extension of CL which is called CL2. It refuses all number of clusters K where at least one of cluster contains points equal or less than free parameters.
4. An another strategy CLM2 extended from CLM in a similar way as CL2 from CL.

where CLM criterion is defined as follows:

$$CLM(M,K) \quad = \quad L(M,K) - EC(M,K)$$

where $EC(M,K)$ is a kind of entropy term:

$$EC(M,K) \quad = \quad -\sum_{k=1}^{K}\sum_{i=1}^{n} z_{ik} \log t_{ik}$$

These four strategies of using classification likelihood are tested in experiment. The result shows that the four strategies perform well on data of well-separated clusters of equal mixing proportions .

1.7.2 AIC criterion

The Akaike Information criterion, proposed by Akaike [17] is a measure of the quality of a model for the dataset. It is generally considered as the first model selection criterion:

$$AIC(M) = -2L_{max}(M) + 2v(M)$$

where $L_{max}(M)$ is the maximum of log-likelihood for the estimated model M and $v(M)$ is the number of the free parameters in this model M.

Basing on AIC criterion, Bozdogan [18] proposed a modified AIC criterion AIC3 which has the form:

$$AIC3(M,K) \quad = \quad -2L(M,K) + 3\nu(M,K)$$

1.7.3 BIC criterion

The Bayesian Information criterion is another penalized criterion which is highly related to the AIC criterion. In the Bayesian framework, choosing a better fit model is to choose the model of the highest posterior probability. In the Bayes' theorem, the posterior probability of the model M_l for dataset $\mathbf{x}$ is:

$$P(M_l|\mathbf{x}) = \frac{f(\mathbf{x}|M_l)P(M_l)}{\sum_r f(\mathbf{x}|M_r)P(M_r)}$$

where $f(\mathbf{x}|M_l)P(M_l)$ is the integral likelihood of the model M_l and $P(M_l)$ is its prior probability. So if all the prior probabilities of all the models are assumed to be the same, choosing the model of the highest posterior probability leads to choosing the model with the biggest integrated likelihood.

By adding the parameters to increase the likelihood usually leads to over-fitting problem. Similar to the AIC criterion, to solve this problem, the BIC criterion introduces a penalty term of the number of the model parameters into the maximum likelihood. The BIC criterion is used to compare models with different parameters and with different number of clusters.

The formula of the BIC criterion is:

$$BIC(M) = -2L_{max}(M) + v(M)\ln(n)$$

where n is the number of points contained in the dataset.

1.7.4 ICL criterion

If the suitable model is not considered in the clustering process, BIC criterion tends to overestimate the number of clusters [52]. To deal with this problem, an Integrated Completed Likelihood (ICL) criterion was proposed [21]. We know that BIC is an approximation of maximum log-likelihood. The completed log-likelihood can be considered as a penalized log-likelihood where the penalty is a measure of the ability of a Gaussian mixture model to provide a fitted partition to the data [21]:

$$CL(M) = L(M) + \underbrace{\sum_{k=1}^{K}\sum_{i=1}^{n} z_{ik}\log t_{ik}}_{\leq 0}$$

where

$$t_{ik} = \frac{\pi_k f(\boldsymbol{x}_i|\boldsymbol{a}_k)}{\sum_l \pi_l f(\boldsymbol{x}_i|\boldsymbol{a}_l)}$$

indicating the conditional probability that $\boldsymbol{x}_i$ arises from the kth mixture component.

So ICL is defined as:

$$ICL(M) = -2LM_{max}(M) + v(M)\ln(n)$$

It can also be written as:

$$\log f(\mathbf{x}, \mathbf{z}|m, K) \approx -2\log f(\mathbf{x}, \mathbf{z}|m, K, \widehat{\theta}) + v_{m,K}\ln(n)$$

where m indicates the form of a Gaussian mixture, K the number of clusters and

$$\widehat{\theta} = \arg\max_{\theta} f(\mathbf{x}, \mathbf{z}|m, K, \theta)$$

But the unobserved data $\mathbf{z}$ is missing, thus $\widehat{\theta}$ is replaced by:

$$\widehat{\theta} = \arg\max_{\theta} f(\mathbf{x}|m, K, \theta)$$

and $\mathbf{z}$ is replaced with $\hat{\mathbf{z}} = MAP(\widehat{\theta})$.

1.7.5 ICOMP, NEC and AWE criteria

Bozdogan [19] proposes to use a measure of non-linear complexity involving the Fisher information matrix F_n^{-1} and the number of parameters. The complexity of van Emden is:

$$g(M) = v(M) \ln[\frac{tr(F_n^{-1})(M)}{v(M)}] - \ln|F_n^{-1}(M)|$$

So Informational Complexity Criterion (*i.e.* ICOMP) criterion is :

$$ICOMP(M) = -2L(M) + v(M) \ln[\frac{tr(F_n^{-1}(M))}{v(M)}] - \ln|F_n^{-1}(M)|$$

Celeux and Soromenho [51] has proposed Normalized Entropy Criterion (NEC). The NEC is in fact derived from a relation between the log-likelihood $L(M)$ and the classification log-likelihood $LC(M)$:

$$CL(M, K) = L(M, K) - E(M, K) \tag{1.2}$$

where

$$E(M, K) = \underbrace{-\sum_{k=1}^{K}\sum_{i=1}^{n} z_{ik} \log t_{ik}}_{\geq 0}$$

and

$$t_{ik} = \frac{\pi_k f(\boldsymbol{x}_i|\boldsymbol{a}_k)}{\sum_l \pi_l f(\boldsymbol{x}_i|\boldsymbol{a}_l)}$$

The entropy $E(M, K)$ cannot directly be a criterion for model choice because $L(M, K)$ increases along with the complexity of the model M. So $E(M, K)$ needs to be normalized. From Equation 2.14, we have:

$$1 = \frac{L(M, K) - L(M, 1)}{LM(M, K) - LM(M, 1)} + \frac{E(M, K) - E(M, 1)}{LM(M, K) - L(M, 1)}$$

when $K > 1$. With $E(M, 1) = 0$, we have the NEC criterion:

$$NEC(M, K) = \frac{E(M, K)}{LM(M, K) - L(M, 1)} \tag{1.3}$$

From the Equation 2.14, NEC criterion cannot compare the situation between $K > 1$ and $K = 1$. To decide if $K = 1$ by NEC criterion, we need to apply an procedure which was proposed by Celeux and Soromenho [51]:

- Estimate the parameters of the Gaussian mixture function by maximizing the likelihood, with different value of K ($2 \leq K \leq K_{sup}$) where K_{sup} indicating a upper bound of the number of components. $NEC(M, K)$ is minimized with K^*: $NEC(M, K^*)$.
- Estimate the parameters of a K^* components' Gaussian mixture with equal means $\mu_1 = \ldots = \mu_{K^*} = \overline{\mu}$ where $\overline{\mu}$ is the sample mean of the data. $\widetilde{L}(1)$ and $\widetilde{E}(M, 1)$ are respectively the corresponding log-likelihood and entropy. The result of NEC criterion with $K = 1$ is:

  $$NEC(M, 1) = \frac{\widetilde{E}(M, 1)}{\widetilde{L}(M, 1) - L(M, 1)}$$

  $K = K^*$ when $NEC(M, K^*) \leq NEC(M, 1)$, otherwise $K = 1$.

Another criterion of approximation of Bayes factor is called Approximate Weight of Evidence (AWE) criterion:

$$AWE(K) = -2CL(K) + 2v(K)(\frac{3}{2} + \ln(n))$$

1.8 Binned data clustering

The EM and CEM algorithms become a useful tool for cluster analysis in many domains. But along with the development of technology, the amount of observations keeps increasing. In this case, both EM and CEM algorithms take a lot of computation time. In order to increase the speed of the EM and CEM algorithms, people have proposed to introduce binned data into clustering approaches.

1.8.1 What is binned data?

Binning is way to group a set of data into a smaller number of bins. It is a data preprocessing technique used to reduce the effects of minor observation errors. It is also used

to reduce the amount of data in order to accelerate the clustering process. Moreover, many datasets can only be presented in the form of binned data deal to measure machine of limited precision, such as red blood cell data [13]. Binned data clustering is useful in dealing with this kind of data.

To get binned data, the idea is to divide the overall space $\mathbb{R}^p$ into v sub-spaces which are also called bins with a partition $(\boldsymbol{\mathcal{H}}_1, \ldots, \boldsymbol{\mathcal{H}}_v)$. The amount of data within each bin is calculated and is called the 'frequency' of the bin. The Figure 1.5 explains the transformation from standard data to binned data.

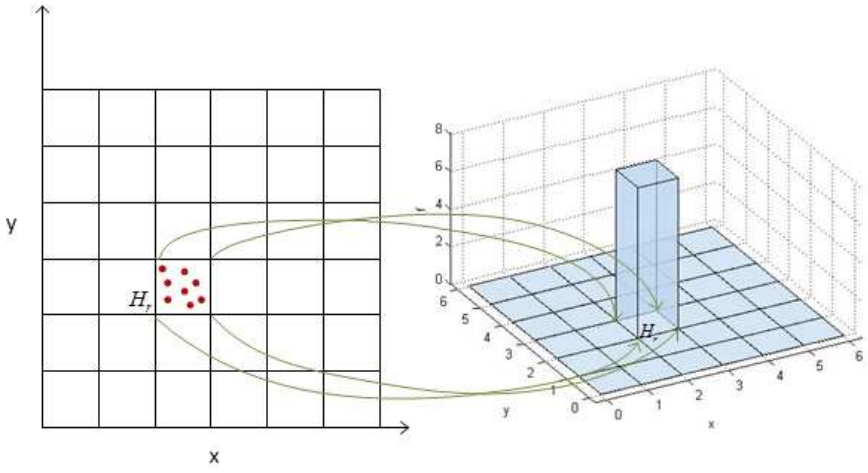


FIGURE 1.5: How to get binned data from standard data.

The locations of bins and their frequencies become the only information about data after transformation. In this case, the number of observations is reduced to the number of bins. Figures 1.6, 1.7 and 1.8 demonstrate an example showing how to obtain binned data.

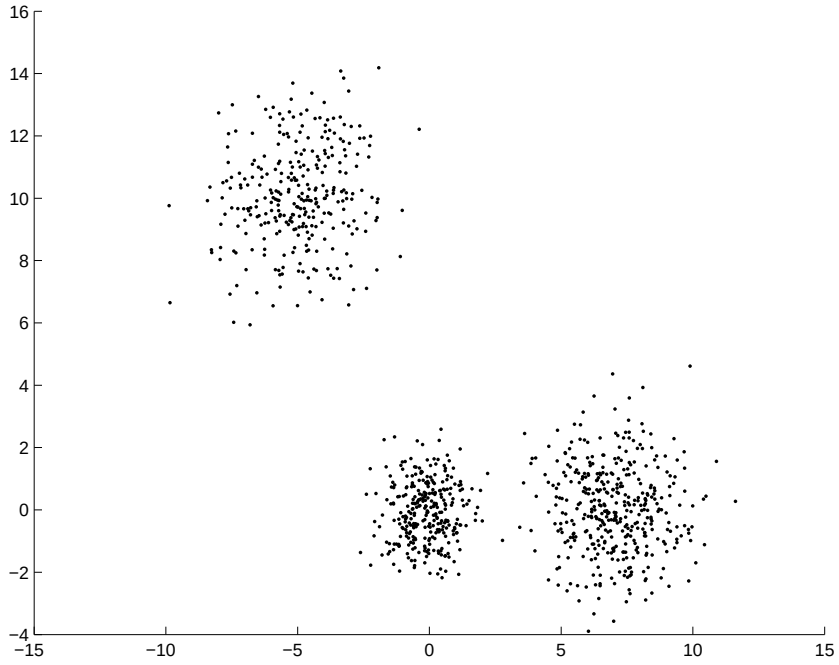


FIGURE 1.6: Simulated data according to Gaussian mixture model with three clusters.

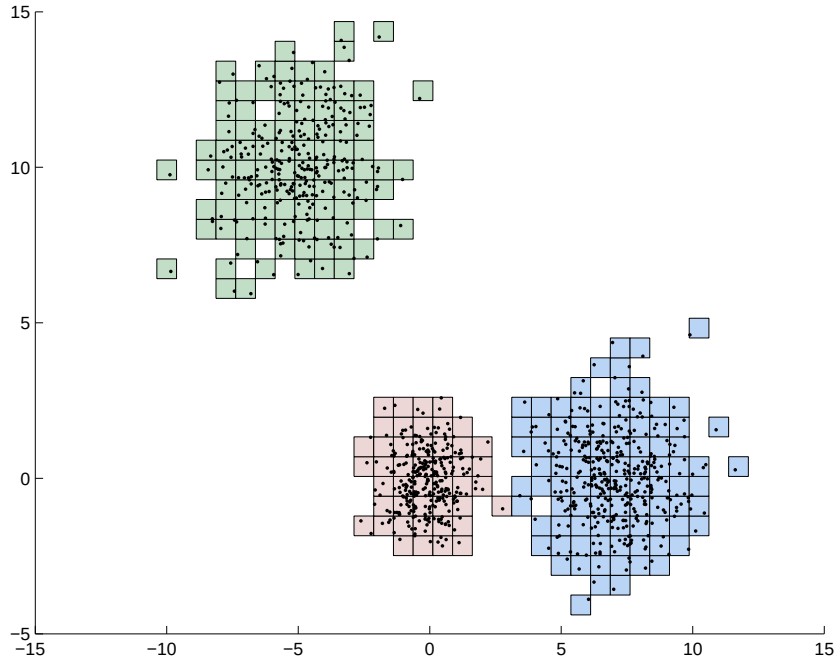


FIGURE 1.7: Binned data in the view of 2d corresponding to the data in Figure 1.6.

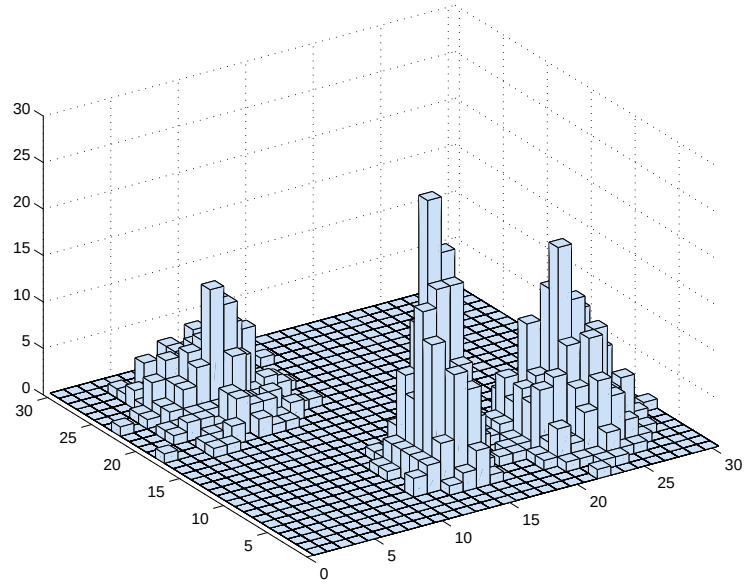


FIGURE 1.8: Binned data in the view of frequencies corresponding to the data in Figure 1.6.

Figure 1.6 describes a set of standard data simulated according to Gaussian mixture model with three clusters of different volumes. In Figure 1.7, the space is divided into small bins of size $= 0.5 \cdot 0.5$ (shown by grid). Only the non-empty-bins are shown in the figure, which are much less than the data. Figure 1.8 presents the three-dimensional view of binned data which are transformed from the standard data (individual observations) in Figure 1.6. The two dimensional plane at the bottom indicates the space of the standard data. The height of each bar in the two dimensional histogram represents the frequency of each bin.

1.8.2 Binned-EM algorithm

The idea of applying EM algorithm to binned data (binned-EM) was first time mentioned by Dempster et al. [7]. Then the one-dimensional binned-EM algorithm was developed by McLachlan and Jones [12] and later Cadez et al. [13] extended it to the multi-dimensional case.

In the paper of Cadez et al. [13], he proposed a general solution to the problem of fitting a multivariate mixture density model to binned and truncated data. The overall sample space $\boldsymbol{\mathcal{H}}$ is divided into v disjoint subspaces $\boldsymbol{\mathcal{H}}_j$, which are also called bins. There are only r non-empty bins. The likelihood is:

$$\log(L) \quad = \quad \sum_{j=1}^{r} n_j \log(P_j) - n \log(P)$$

where $n = \sum_{j=1}^{r} n_j$. And P and P_j indicates the integrals of the probability density function (PDF) of bins:

$$P_j = P_j(\Phi) = \int_{\boldsymbol{\mathcal{H}}_j} f(x;\Phi)\mathrm{d}\boldsymbol{x}$$

$$P = P(\Phi) = \sum_{j=1}^{r} P_j = \int_{\boldsymbol{\mathcal{H}}} f(x;\Phi)\mathrm{d}\boldsymbol{x}$$

Since the implementation of binned-EM algorithm can be slow, in order to reduce the execution time, several straightforward numerical techniques are also proposed in the paper of Cadez et al. [13]: Romberg integration was used to select the order of integration, so as to converge faster; two special bins are defined to cover the truncated regions: one covers the space from the last bin to ∞, another from $-\infty$ to the first bin; an additional heuristic is added into the algorithm to make EM algorithm converge in less time. Experimental results on simulated data suggest that the proposed methods can save significant computation time with no loss in the accuracy of the parameter estimates. And an application of the proposed approach to diagnosis of iron deficiency anemia was briefly described, in the context of binned and truncated bivariate measurements of volume and hemoglobin concentration from an individual's red blood cells.

Similar as the EM algorithm, binned-EM algorithm aims to obtain the parameters estimate which maximizes the likelihood. In binned data framework, each bin is assumed to belong to one cluster. The log-likelihood of binned-EM algorithm takes the form:

$$L(\boldsymbol{\Phi}) = \sum_{r=1}^{v} n_r \log\Big(\sum_{k=1}^{K} \pi_k^{(q+1)} \int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x};\theta_k^{(q)})d\boldsymbol{x}\Big) + \log(c)$$

In binned data clustering, maximizing the expectation of the complete likelihood equals maximizing the likelihood. But the former method is easier. The complete likelihood contains observed data and missing data. The missing data is the label vector $\boldsymbol{z}_r$ $(r = 1,\ldots,v)$ which indicates the origin labels of the bins $\boldsymbol{\mathcal{H}}_r$. The complete log-likelihood is:

$$\begin{aligned} L(\boldsymbol{\Phi};\boldsymbol{a},\boldsymbol{z}) &= \log(p(\boldsymbol{a},\boldsymbol{z};\boldsymbol{\Phi})) \\ &= \sum_{r=1}^{v} n_r \log\big(\pi_{z_r} \int_{\boldsymbol{\mathcal{H}}_r} f_{z_r}(\boldsymbol{x};\theta_{z_r})\mathrm{d}\boldsymbol{x}\big) + \log(c) \end{aligned}$$

To maximize the expectation of the complete log-likelihood $E(L(\boldsymbol{\Phi};\boldsymbol{a},\boldsymbol{z}))$, similar as the EM algorithm, the binned-EM algorithm executes the Expectation step and the Maximization step iteratively. The derivation of binned-EM algorithm will be detailed in the Chapter 2.

1.8.3 Bin-EM-CEM algorithm

A classification EM algorithm for binned data (bin-EM-CEM, noted as binned-EM-CEM Samé et al. [14]) was first time proposed by Samé et al. [14]. The derivation of bin-EM-CEM algorithm is detailed in this paper. Numerical experiments are shown in order to compare the bin-EM-CEM algorithm with the classical CEM algorithm. The result shows that the bin-EM-CEM algorithm spends less CPUtime than CEM algorithm when dealing with large number of data with a appropriate size of bins. From the aspect of quality, bin-EM-CEM algorithm approaches to CEM algorithm when the bins become small enough. They proposed an approximation of ICL for bin-EM-CEM algorithm to select the best number of clusters:

$$ICL_b(K) \quad = \quad (L(\hat{\Phi}; a, \hat{z}) - n\log(S) - \log(c)) - \frac{v_K}{2}\log(n)$$

where $\hat{\Phi}$ and $\hat{z}$ are estimations by bin-EM-CEM with K clusters, S is the surface of each bin. And

$$v_K = (2K) + 2K + (K-1) = 5K - 1$$

A bin-EM-CEM application was presented to detect damaged sones on the surface of a gas tank, by using acoustic emissions.

In 2004, Hamdan and Govaert [53] have addressed the problem of taking into account data imprecision in the mixture model clustering of binned data by binned-EM algorithm. An original method to fit the binning data procedure to imprecise data was developed. The idea is to model imprecise data by multivariate uncertainty zones and to assign each uncertainty zone to several bins with proportions proportional to its overlapping volumes with the bins. To overcome the long computation time problem of binned-EM algorithm, Hamdan [54] applied classification approach of binned data based on bin-EM-CEM algorithm. The complete log-likelihood criterion is:

$$L(\boldsymbol{\Phi}; \boldsymbol{a}, \boldsymbol{z}) = \sum_{r=1}^{v} \log(\pi_{z_r} \int_{\boldsymbol{\mathcal{H}}_r} f_{z_r}(\boldsymbol{x}; \theta_{z_r}) \mathrm{d}\boldsymbol{x}) + \log \frac{n!}{\prod_{r=1}^{v} n_r!}$$

where $\boldsymbol{\Phi} = (\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K)$, π_k $(k = 1, \ldots, K)$ are the proportions of the mixture, θ_k $(k = 1, \ldots, K)$ are the parameters of each component and z_r $(r = 1, \ldots, v)$ are the origin component of bin $\boldsymbol{\mathcal{H}}_r$. A flaw diagnosis application to acoustic emission control was presented. The partitions obtained by the algorithms CEM, int-EM-CEM, bin-EM-CEM and int-bin-EM-CEM are compared.

Later in 2009, another fast algorithm devoted to binned data clustering was proposed by Samé [55]. The proposed approach is called bin-CEM algorithm, which is inspired

by CEM algorithm. In the framework of grouped data, this approach estimates simultaneously the missing data $\boldsymbol{x}, \boldsymbol{z}$ and the parameter vector $\boldsymbol{\Phi}$ by maximizing the complete data log-likelihood. And the complete data log-likelihood can be rewritten as followed:

$$L_c(\boldsymbol{\Phi}; \boldsymbol{a}, \boldsymbol{z}) = \sum_{r=1}^{v} \sum_{s=1}^{n_r} \log \left(\pi_{z_{rs}} f_{z_{rs}}(\boldsymbol{x}_{rs}; \theta_{z_{rs}}) \right)$$

The standard CEM, bin-CEM and bin-EM (EM applied to grouped data) algorithms are compared in the experiment study. The experimental result shows that, under the condition that the number of bins per dimension is 40, the execution time of the bin-CEM and bin-EM algorithms stay almost constant along with the increase of sample size, since they only depend on the number of bins. The proposed algorithm bin-CEM outperforms the CEM and bin-CEM algorithms in terms of computation time.

In the Chapter 3, the derivation of bin-EM-CEM algorithm will be detailed.

1.9 Conclusion

This chapter presented the basic and important concepts which our following study bases on.

In this chapter, firstly we presented the definition of cluster analysis. Three main clustering approaches were reviewed. They are the hierarchical clustering, k-means algorithm and model-based clustering. Then, we highlighted the model-based clustering approaches basing on Gaussian mixture models. The mixture approach and the classification approaches were presented. We also studied their corresponding algorithms: EM and CEM. In order to simplify the clustering process, two kinds of parsimonious Gaussian mixture models were presented. One kind of parsimonious models is based on variance matrix parametrization. And anther group of parsimonious models is based on factor analysis model. Since the core of our thesis is model selection for data clustering, commonly used criteria for model selection were presented in detail. The AIC, BIC, ICL, ICOMP, NEC, and AWE criteria, for standard data, were detailed. Finally, a main concept of this thesis was presented: binned data. The definition of binned data was detailed. The concepts of the EM algorithm applied to binned data (binned-EM) and the CEM algorithm applied to binned data (bin-EM-CEM) were explained.

After giving a comprehensive introduction on the main concepts we use in our study, in the following three chapters, we will present the main contribution of our study. In the next two chapters, we aim to combine the advantages of binning data with the advantages of parsimonious models, so as to simplify the parameters estimation and to

save some computation time. The binned-EM algorithms of fourteen Gaussian mixture models will be developed in the Chapter 2, and the bin-EM-CEM algorithms of fourteen Gaussian mixture models will be developed in the Chapter 3. The Chapter 4 shows how to adapt AIC, BIC, ICL, ICOMP, NEC, and AWE criteria to binned data clustering.

Chapter 2

Parsimonious Gaussian mixture models for binned data clustering and the corresponding binned-EM algorithms

2.1 Introduction

Mixture approach is one of the most commonly used model-based clustering approaches. It aims to maximize the likelihood over the mixture model parameters. The mixture model parameters estimates are obtained by maximizing the likelihood using Expectation Maximization (EM) algorithm. And the partition is estimated by maximum a posteriori (MAP) rule (see [15] for example). The EM algorithm was firstly introduced by Dempster et al. [7], and corrected by Wu [39] on its flawed convergence analysis. Because of its simple conception and easy implementation, the EM algorithm has been widely used for maximum likelihood estimation specially when some variables are unobserved. On the other side, the EM algorithm has some limitations. One limitation is that the EM algorithm becomes very slow when dealing with large datasets. This problem has been discussed and some suggestions to accelerate the EM algorithm have been proposed [56].

With the increase of the number of mixture components and the dimensionality of data, too many parameters to estimate is one of the reasons of slow computation. Simplifying the model complexity can reduce the number of parameters. This approach can be efficient in solving the long computation time problem. Banfield and Raftery [16]

suggested a way of decomposition of the variance matrix of each component of a Gaussian mixture model. Following this parametrization, fourteen parsimonious models were proposed [15]. The experimental result showed that these models can detect many data distributions, instead of using the most complex model.

Besides that, binned data was introduced into EM algorithm. This new approach has two advantages: firstly, it accelerates the speed of the EM algorithm by reducing the amount of data; secondly, it adapts to the binned data which exist naturally due to the limited precision of the measuring equipment. Applying EM algorithm to binned data was first time mentioned by Dempster et al. [7]. The one-dimensional binned-EM algorithm was developed by McLachlan and Jones [12]. Then Cadez et al. [13] extended binned-EM algorithm to the multi-dimensional case. The experimental results showed that the binned-EM algorithm is faster than the EM algorithm without losing much precision.

In order to combine the advantages of parsimonious models and binned data with EM algorithm, this chapter will present the new EM algorithms of parsimonious Gaussian mixture models applied to binned data (binned-EM algorithms). Firstly, the Section 2.2 will give an overview of the classic EM algorithm for standard data. The complexity of the EM algorithm will be calculated. The Section 2.3 will describe the fourteen parsimonious Gaussian mixture models. The Section 2.4 will present the derivation of the binned-EM algorithm. The computational complexity of the binned-EM algorithm will be calculated and compared to the one of the EM algorithm. Parameters estimations corresponding to different parsimonious models will be detailed in the Section 2.5. In the Section 2.6, experiments of binned-EM algorithms on simulated data and real data will be shown and analyzed. Finally, we will summarize this chapter and lead to the next chapter.

2.2 Mixture approach for standard data

In mixture model-based clustering, mixture approach aims to maximize the likelihood over the mixture model parameters. The parameters are usually estimated by the EM algorithm, and the data partition is obtained by the MAP rule. In this section, we will review the classical EM algorithm and the MAP rule.

2.2.1 The EM algorithm

In this part, a general review of the Estimation Maximization (EM) algorithm will be given. The EM algorithm is an iterative method which aims to maximize likelihood and

estimate the corresponding model parameters. After being proposed several times in special circumstances, the EM algorithm was formally defined by Dempster et al. [7]. But the convergence analysis of Dempster et al. [7] was flawed and Wu [39] corrected it in 1983.

Assume that a set of observed data $\mathbf{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is generated according to a mixture distribution. $\mathbf{x}$ is also called the incomplete data. The complete data includes $\mathbf{x}$ as well as missing data $\mathbf{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$. If we assume that each $\boldsymbol{x}_i$ is independent identically distributed (i.i.d.), this model follows a distribution:

$$p(\mathbf{x}|\theta) = \sum_{i=1}^{n} p(\boldsymbol{x}_i|\theta) = L(\theta|\mathbf{x})$$

where θ is the unknown parameters, and $L(\theta|\mathbf{x})$ is called the likelihood function of θ given $\mathbf{x}$.

In the case of Gaussian mixture model, the model is composed by K components. Each component follows respectively a Gaussian distribution. $L(\theta|\mathbf{x})$ takes the form:

$$P(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k p_k(\mathbf{x}|\theta_k)$$

where π_k is the mixing proportion of the k component, $p_k(\mathbf{x}|\theta_k)$ is the distribution of the k component with parameter $\theta_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, $\boldsymbol{\mu}_k$ is the mean vector of the component and $\boldsymbol{\Sigma}_k$ its variance matrix.

We want to find the maximum likelihood estimate (MLE) $\widehat{\theta}$ such that $L(\widehat{\theta}|\mathbf{x})$ is a maximum. In order to estimate θ, log-likelihood function is introduced:

$$L(\theta|\mathbf{x}) = \log P(\mathbf{x}|\theta)$$

Henceforth $L(\theta|\mathbf{x})$ is the notation of log-likelihood. Since $\log(x)$ strictly increases along with x, the value θ maximizing $P(\mathbf{x}|\theta)$ also maximizes $\log\big(P(\mathbf{x}|\theta)\big)$. But maximum of $L(\theta|\mathbf{x})$ is generally uneasy to obtain. The EM algorithm seeks to obtain the MLE by maximizing the expectation of the complete log-likelihood $L(\theta|\mathbf{x}, \mathbf{z})$ which bases on the complete data $(\mathbf{x}, \mathbf{z})$. The EM algorithm repeats two steps iteratively until convergence:

- Expectation step (E-step): Calculate the expectation of the complete log-likelihood

$$Q(\theta, \theta^{(q)}) = E(L(\theta; \mathbf{x}, \mathbf{z}))$$

For the case of Gaussian mixture model, this equals to computing the conditional probabilities of each component k:

$$t_{ik}^{(q)} = \frac{\pi_k^{(q)} f_k(\boldsymbol{x}_i; \theta_k^{(q)})}{t_i^{(q)}}$$

where

$$t_i^{(q)} = \sum_{k=1}^{K} \pi_k^{(q)} f_k(\boldsymbol{x}_i; \theta_k^{(q)})$$

- Maximization step (M-step): Find the parameters $\hat{\theta}$ which maximize $Q(\theta, \theta^{(q)})$:

$$\hat{\theta} = \arg\max_{\theta} Q(\theta, \theta^{(q)})$$

We obtain the parameters of Gaussian mixture model:

$$\pi_k^{(q+1)} = \frac{1}{n} \sum_{i=1}^{n} t_{ik}^{(q)}$$

$$\boldsymbol{\mu}_k^{(q+1)} = \frac{1}{\sum_{i=1}^{n} t_{ik}^{(q)}} \sum_{i=1}^{n} t_{ik}^{(q)} \boldsymbol{x}_i$$

The result of $\boldsymbol{\Sigma}$ in the M-step varies according to the parsimonious model [15]. For example, for the model $[\lambda \mathbf{DAD}^T]$, the estimate of $\boldsymbol{\Sigma}$ has the form:

$$\boldsymbol{\Sigma}_k^{(q+1)} = \frac{1}{\sum_{i=1}^{n} t_{ik}^{(q)}} \sum_{i=1}^{n} t_{ik}^{(q)} (\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(q+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(q+1)})^T$$

2.2.2 The complexity of EM algorithm

The efficiency is one of the important characteristics of an algorithm. The CPUtime that an algorithm consumes represents the efficiency of the algorithm. But this quantity differs due to many factors. These factors are, for example, the way of programming, the processor and the memory of the computer and the random result of one experiment.

Time complexity of an algorithm quantifies the amount of the time taken by the algorithm. It is commonly expressed by a function of inputs, using the Big-O notation. In order to better study the time consumption of the EM algorithm and to compare with the other algorithms, in this part, we will calculate the time complexity of the EM algorithm.

Three parameters affect the computation time of EM algorithm. They are the amount of data n, the number of space dimension d and the number of clusters K. The EM

algorithm based on different model takes different computation time. As our study focus on the fourteen parsimonious Gaussian mixture models, we will calculate the time complexity of the EM algorithm based on one of these models. Among these models, we choose the model $[\lambda \mathbf{DAD}^T]$ because of its simplicity and generality.

The EM algorithm of the model $[\lambda \mathbf{DAD}^T]$ is presented in the Algorithm 1:

Algorithm 1 EM algorithm

$q \leftarrow 0$
Initialize $\boldsymbol{\pi}^{(0)}$ and $\theta^{(0)} = \{\boldsymbol{\mu}^{(0)}, \boldsymbol{\Sigma}^{(0)}\}$.
repeat
  for $i = 1 : n$ **do**
    $t_i^{(q)} \leftarrow \sum_{k=1}^{K} \pi_k^{(q)} f_k(\boldsymbol{x}_i; \theta_k^{(q)})$
    for $k = 1 : K$ **do**
      $t_{ik}^{(q)} \leftarrow \frac{\pi_k^{(q)} f_k(\boldsymbol{x}_i; \theta_k^{(q)})}{t_i^{(q)}}$
    end for
  end for
  for $k = 1 : K$ **do**
    $\pi_k^{(q+1)} \leftarrow \frac{1}{n} \sum_{i=1}^{n} t_{ik}^{(q)}$
    $\boldsymbol{\mu}_k^{(q+1)} \leftarrow \frac{1}{\sum_{i=1}^{n} t_{ik}^{(q)}} \sum_{i=1}^{n} t_{ik}^{(q)} \boldsymbol{x}_i$
    $\boldsymbol{\Sigma}_k^{(q+1)} \leftarrow \frac{\sum_{k=1}^{K} \sum_{i=1}^{n} t_{ik}^{(q)} (\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(q+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(q+1)})^T}{n}$
  end for
  $q \leftarrow q + 1$
until $\frac{L^{(q+1)} - L^{(q)}}{L^{(q)}} < \varepsilon$
$\widehat{\boldsymbol{\pi}} \leftarrow \boldsymbol{\pi}^{(q+1)}$, $\widehat{\theta} \leftarrow \theta^{(q+1)}$

From the Algorithm 1, the EM algorithm follows an iterative process, and ends until $\frac{L^{(q+1)} - L^{(q)}}{L^{(q)}} < \varepsilon$. It is uneasy to know exactly when the EM algorithm stops. The speed of the convergence depends on the dataset, the initiation and the threshold ε. In other word, the computational complexity of EM algorithm approaches infinity. But in reality, the EM algorithm usually stops in few seconds or more. To be able to calculate the time complexity, we suppose that EM algorithm stops in N iterates. If we note the complexity of each iterate as $O(T)$, then the complexity of the EM algorithm is about $O(NT)$.

To facilitate the calculation, each iterate is divided into three parts:

- Calculate n times $t_i^{(q)}$.
- Calculate $n * K$ times $t_{ik}^{(q)}$.
- Calculate K times $\pi_k^{(q+1)}$, $\boldsymbol{\mu}_k^{(q+1)}$ and $\boldsymbol{\Sigma}_k^{(q+1)}$.

To calculate $t_i^{(q)}$ equals to calculate K times this equation:

$$\pi_k^{(q)} f_k(\boldsymbol{x}_i; \theta_k^{(q)}) = \frac{\pi_k^{(q)}}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_k^{(q)}|^{1/2}} \exp(-1/2(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(q)})\boldsymbol{\Sigma}_k^{(q)^{-1}}(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(q)})^T) \tag{2.1}$$

To calculate the complexity of the Equation 2.1, the difficult part is to calculate $|\boldsymbol{\Sigma}_k^{(q)}|^{1/2}$, $\exp(x)$ and $\boldsymbol{\Sigma}_k^{(q)^{-1}}$.

To calculate the determinant of one $d \times d$ matrix using LU decomposition costs $O(d^3)$ complexity. To obtain the square root of S, we can use the Newton's method. Because finding $\sqrt{S}$ is the same as solving the equation $f(x) = x^2 - S = 0$. We can obtain the approximate result by doing the equation 2.2 calculation iteratively:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - S}{2x_n} = \frac{1}{2}(x_n + \frac{S}{x_n}) \tag{2.2}$$

In the Equation 2.2, each iterate needs 4 basic operations. If the root being sought has multiplicity greater than one, the convergence rate is merely linear (errors reduced by a constant factor at each step) unless special steps are taken. The speed of convergence depends on the defined threshold or how we want it to stop. It might take a long time until convergence. To simplify, we suppose it stops in n_1 iterates. Thus, the complexity of obtaining the square root is approximately $O(4n_1)$. So the complexity of $|\boldsymbol{\Sigma}_k^{(q)}|^{1/2}$ is about $O(d^3 + 4n_1)$.

To calculate $\exp(x)$, we use Taylor series:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \tag{2.3}$$

The condition to define when to stop the multiplication is flexible. So we use truncated Taylor series. We suppose that the result of e^x becomes stable until $n = n_2$. Thus the Equation 2.3 becomes:

$$e^x = \sum_{n=0}^{n_2} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^{n_2}}{n_2!} \tag{2.4}$$

The complexity of the Equation 2.4 is $O(n_2^2)$.

To calculate the inversion of $\boldsymbol{\Sigma}_k^{(q)}$, using Gaussian elimination takes complexity of $O(d^3)$.

After adding up all the operations, calculating the Equation 2.1 takes complexity of $T = O(2d^3 + 2d^2 + \frac{5}{2}d + 4n_1 + n_2^2)$. According to the Big-O notation, T can be also considered as $T = O(d^3 + 4n_1 + n_2^2)$.

Since the complexity of $t_i^{(q)}$ is calculated, the complexity of the rest of the EM algorithm is easy to find out. The complexity of each part is listed in the Table 2.1.

Parameter	Times	Complexity
$t_i^{(q)}$	n	$O(d^3K + 4n_1K + n_2^2K)$
$t_{ik}^{(q)}$	$n * K$	$O(1)$
$\pi_k^{(q+1)}$	K	$O(n)$
$\boldsymbol{\mu}_k^{(q+1)}$	K	$O(nd + 2n)$
$\boldsymbol{\Sigma}_k^{(q+1)}$	K	$O(3dnK)$

TABLE 2.1: Decomposition of complexity of the EM algorithm.

From the Table 2.1, we can conclude that the complexity of the EM algorithm is approximately $O(d^3KnN + 3K^2ndN + 4n_1KnN + n_2^2KnN + 4nKN + dnKN)$. According to the definition of the Big-O notation, the complexity of EM algorithm can also be noted as $O(d^3KnN + 3K^2ndN + 4n_1KnN + n_2^2KnN)$

2.3 Parsimonious models

In general, the greater the number of simplifying assumptions made about the essential structure of the real world, the simpler the model. One goal of the science is to create simple models that have a great deal of explanatory power. Such models are called parsimonious models. Parsimonious models provide a simpler and clearer image of the structure of data. They simplify the process of parameter estimation of the model and save the computation time.

In order to obtain parsimonious Gaussian mixture models, a way of decomposition of the variance matrix $\boldsymbol{\Sigma}_k$ was proposed by Banfield and Raftery [16]: $\boldsymbol{\Sigma}_k = \lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T$, where $\lambda_k = |\boldsymbol{\Sigma}_k|^{1/d}$ determining the volume of the kth cluster, $\mathbf{D}_k$ which is the matrix of eigenvectors of $\boldsymbol{\Sigma}_k$ determining its orientation, and a diagonal matrix $\mathbf{A}_k$ determining its shape. $\mathbf{A}_k$ has the normalized eigenvalues of $\boldsymbol{\Sigma}_k$ in a decreasing order on the diagonal, and $|\mathbf{A}_k|$=1. Eight general parsimonious Gaussian mixture models were generated by allowing none, some or all of the parameters to vary among clusters: $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$, $[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$, $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, $[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$, $[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$, $[\lambda \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$, $[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$. Besides, by putting certain restrictions on the orientation matrices $\mathbf{D}_k$ and the shape matrices $\mathbf{A}_k$ respectively, we can obtain two other more simplified families of parsimonious Gaussian mixture models. By assuming the orientation of clusters as horizontal or vertical, the orientation matrices $\mathbf{D}_k$ have exactly one entry

1 or -1 in each row and each column and 0 elsewhere. In this case, there is no interest of studying the variations on the orientation matrices. We can simplify a part of the variance matrices as: $\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T = \mathbf{B}_k$, where $\mathbf{B}_k$ is a diagonal matrix with $|\mathbf{B}_k| = 1$. As the variance matrices $\mathbf{\Sigma}_k$ become $\mathbf{\Sigma}_k = \lambda_k\mathbf{B}_k$, we have four diagonal models: $[\lambda\mathbf{B}]$, $[\lambda_k\mathbf{B}]$, $[\lambda\mathbf{B}_k]$ and $[\lambda_k\mathbf{B}_k]$. In another case, we assume the shape of clusters are spherical. The shape matrices are always diag(1,1), and the variations on the orientation matrices are not necessary. Then we have two most simple models in the spherical family: $[\lambda\mathbf{I}]$ and $[\lambda_k\mathbf{I}]$ [15].

In order to show the difference among fourteen models, samples of these fourteen parsimonious Gaussian mixture models are presented in Figures 2.1, 2.2, 2.3. For simplicity, these simulated data are generated in a two-dimensional space, and contain two components of the same proportions.

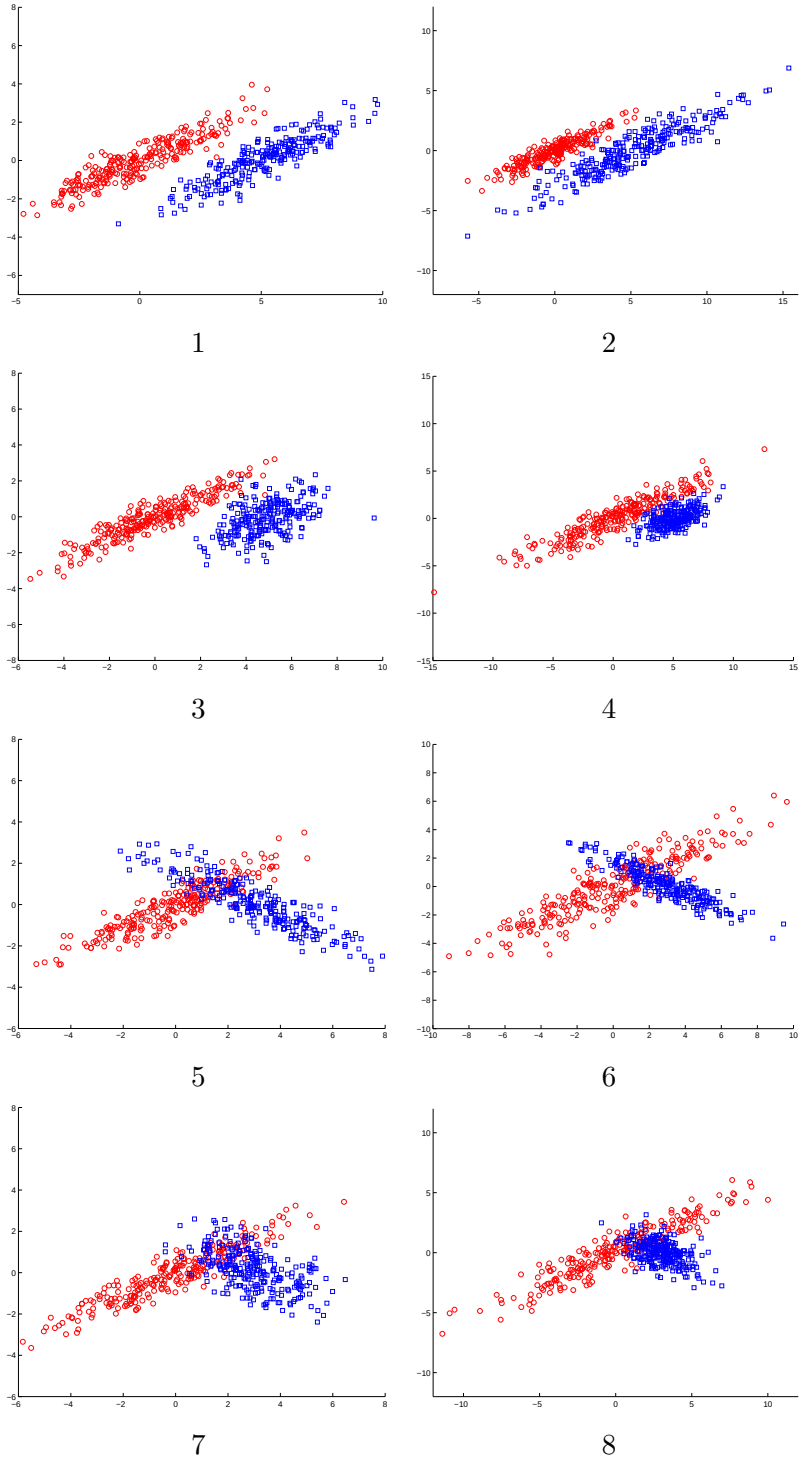


FIGURE 2.1: Eight general models: 1.$[\lambda \mathbf{DAD}^T]$, 2.$[\lambda_k \mathbf{DAD}^T]$, 3.$[\lambda \mathbf{DA}_k \mathbf{D}^T]$, 4.$[\lambda_k \mathbf{DA}_k \mathbf{D}^T]$, 5.$[\lambda \mathbf{D}_k \mathbf{AD}_k^T]$, 6.$[\lambda_k \mathbf{D}_k \mathbf{AD}_k^T]$, 7.$[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$, 8.$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$.

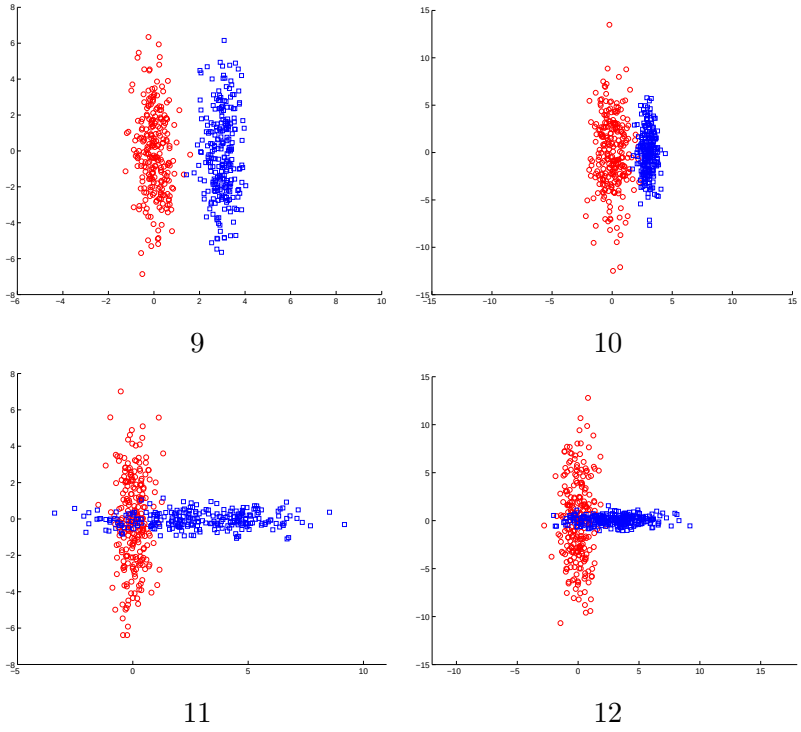


FIGURE 2.2: Four diagonal models: 9.[$\lambda\mathbf{B}$], 10.[$\lambda_k\mathbf{B}$], 11.[$\lambda\mathbf{B}_k$], 12.[$\lambda_k\mathbf{B}_k$].

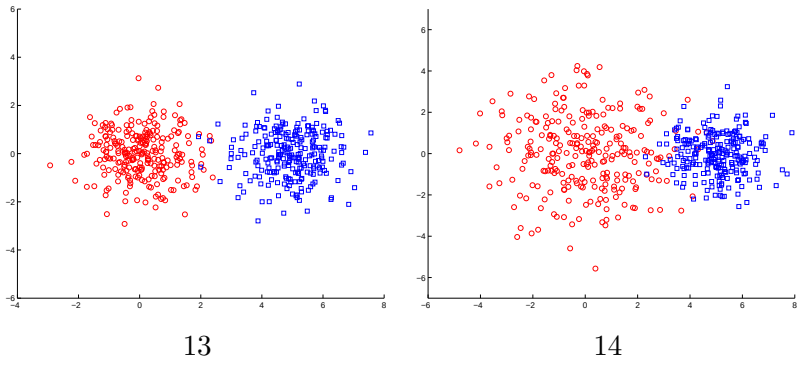


FIGURE 2.3: Two spherical models: 13.[$\lambda\mathbf{I}$], 14.[$\lambda_k\mathbf{I}$].

In the Figure 2.1, samples of eight general models are presented. The general family has the highest freedom degree of the parameters comparing to the other families. For the models of the general family, orientation, shape and volume can be different among clusters. The model [$\lambda\mathbf{DAD}^T$] is the simplest and the most restrictive model in the general family since all the components in this model have the same characteristics (same volume, shape and orientation). Conversely, the model [$\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T$] has all the freedoms of the components. Thus it is more flexible as well as more complex than the other models in this family.

To obtain the diagonal models, a restriction on the orientation is added. From the Figure 2.2, we can see that for this kind of model, the orientation of the components are defined to be vertical or horizontal. This added restriction simplifies the model but also limits the model choice.

The simplest parsimonious models are spherical models. In this kind of models, all the components are supposed to be of spherical shape. Only one parameters to estimate in this model: the volume.

These fourteen models have a hierarchical relationship which is shown in the Figure 2.4, from the most complex model to the simplest model:

Depending on the model complexity, each model has different combination of free parameters. The more complex model has more parameters. The Table 2.2 presents the number of parameters for these fourteen models:

Family	Model	Number of parameters
General	$[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$	$\alpha + \beta$
	$[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$	$\alpha + \beta + K - 1$
	$[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	$\alpha + \beta + (K-1)(d-1)$
	$[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	$\alpha + \beta + (K-1)d$
	$[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$\alpha + K\beta - (K-1)d$
	$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$\alpha + K\beta - (K-1)(d-1)$
	$[\lambda \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$\alpha + K\beta - (K-1)$
	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$\alpha + K\beta$
Diagonal	$[\lambda \mathbf{B}]$	$\alpha + d$
	$[\lambda_k \mathbf{B}]$	$\alpha + d + K - 1$
	$[\lambda \mathbf{B}_k]$	$\alpha + Kd - K + 1$
	$[\lambda_k \mathbf{B}_k]$	$\alpha + Kd$
Spherical	$[\lambda \mathbf{I}]$	$\alpha + 1$
	$[\lambda_k \mathbf{I}]$	$\alpha + d$

TABLE 2.2: Number of free parameters of fourteen models. Where $\alpha = Kd + K - 1$ for the unrestricted case, $\alpha = Kd$ for the restricted case. And $\beta = (d(d+1)/2)$.

We cannot tell which model is the best among these fourteen models. It depends on the practical needs and the distribution of the data. EM algorithms of these fourteen models were developed [15]. In the next section, we will present how to develop the binned-EM algorithms of these fourteen models.

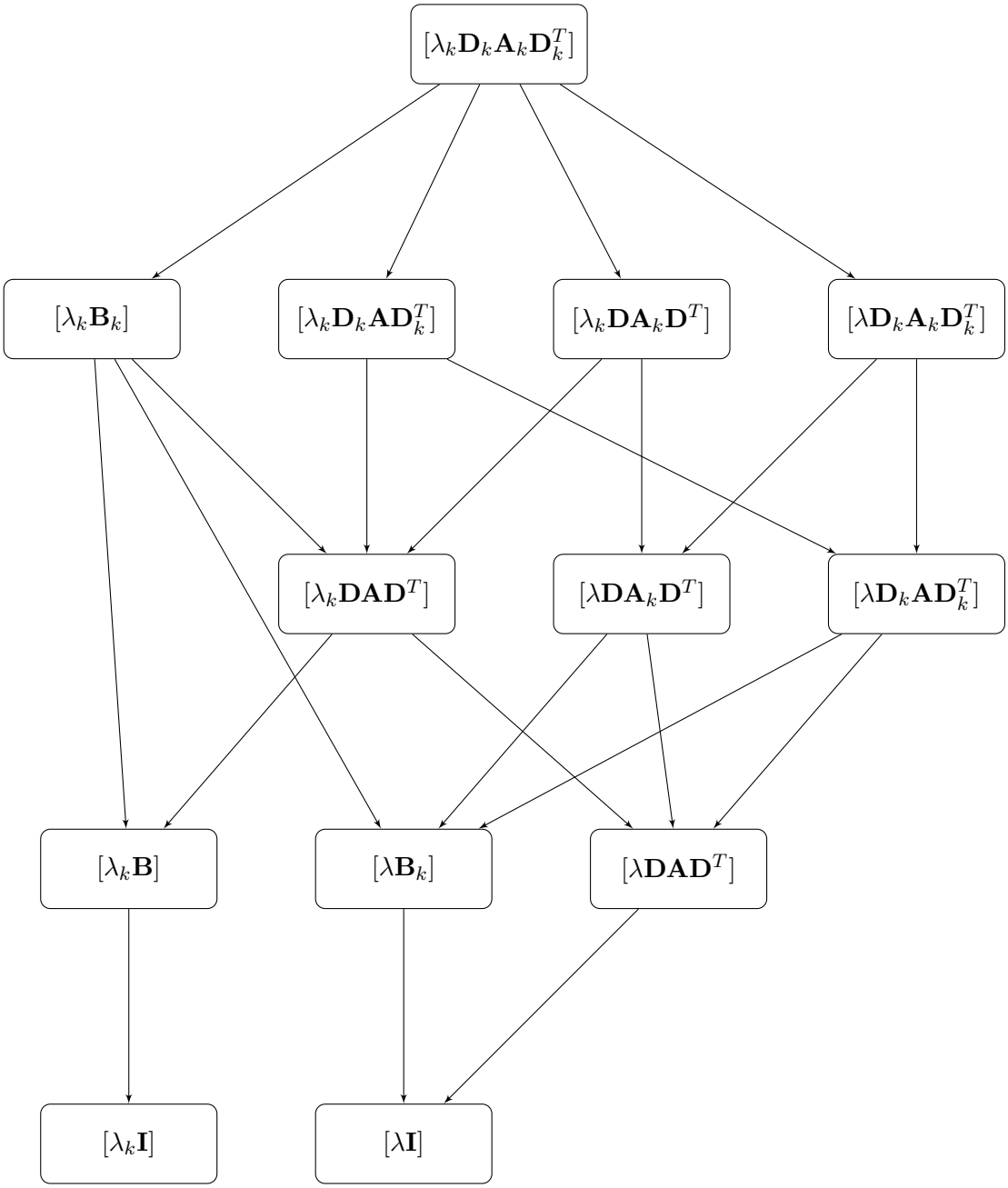


FIGURE 2.4: Hierarchical relationship of fourteen parsimonious Gaussian mixture models.

2.4 Binned-EM algorithm

2.4.1 The likelihood

We suppose that $\mathbf{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is a sample issued from a K-component mixture distribution defined on a d-dimensional space ($\mathbb{R}^d$):

$$f(\mathbf{x}; \boldsymbol{\Phi}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}; \theta_k)$$

with $\boldsymbol{\Phi} = (\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K)$, where π_k $(k = 1, \ldots, K)$ are the mixing proportions ($0 < \pi_k < 1$ and $\sum_{k=1}^{K} \pi_k = 1$), and $\theta_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ $(k = 1, \ldots, K)$ are the parameters of Gaussian distribution functions f_k of components: mean vectors $\boldsymbol{\mu}_k$ and variance matrices $\boldsymbol{\Sigma}_k$.

Similar as the EM algorithm for standard data, the EM algorithm for binned data *i.e.* the binned-EM algorithm, also provides a theoretical framework that enables us to iteratively maximize the observed likelihood (the one which includes only the observed data) by maximizing the expectation of the complete likelihood (the one which includes both the missing data and the observed data). Firstly, before applying the binned-EM algorithm, standard data are transformed into binned data. The overall sample space $\mathbb{R}^d$ is divided into v bins with a subspace partition $(\boldsymbol{\mathcal{H}}_1, \ldots, \boldsymbol{\mathcal{H}}_v)$. We assume that the only observed data is a set of frequencies n_r $(r = 1, \ldots, v)$ where each frequency n_r indicates the number of $\boldsymbol{x}_i$ belonging to the bin $\boldsymbol{\mathcal{H}}_r$. The vector $\boldsymbol{a} = (n_1, \ldots, n_v)$ is the vector of frequencies, with $\sum_{r=1}^{v} n_r = n$. In order to maximize the likelihood $L(\boldsymbol{\Phi})$, the binned-EM algorithm maximizes the expectation of complete likelihood $L_C(\boldsymbol{\Phi})$ which is built on the concept of the complete data $(\mathbf{x}, \mathbf{z}) = \big((\boldsymbol{x}_1, \boldsymbol{z}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{z}_n)\big)$. The label vectors $\boldsymbol{z}_i$ $(i = 1, \ldots, n)$ have the form $\boldsymbol{z}_i = (z_{i1}, \ldots, z_{iK})$, where each z_{ik} $(k \in \{1, \ldots, K\})$ values 1 if $\boldsymbol{x}_i$ arises from the mixture component k and values 0 otherwise. The complete log-likelihood can be written as follows:

$$\begin{aligned} L_C(\boldsymbol{\Phi}) &= L(\boldsymbol{\Phi}; \mathbf{x}, \mathbf{z}) \\ &= \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log\big(\pi_k f_k(\boldsymbol{x}_i; \theta_k)\big) + \log(c) \end{aligned}$$

where c is a constant and does not depend on $\boldsymbol{\Phi}$.

Since there is no information about the exact location of the data in each bin, we assume that all the data in the same bin belong to the same cluster. Thus, the label vectors $\boldsymbol{z}_i$ $(i = 1, \ldots, n)$ can be transformed into $\boldsymbol{z}_r$ $(r = 1, \ldots, v)$ where $\boldsymbol{z}_r \in \{1, \ldots, K\}$ $(r \in \{1, \ldots, v\})$ is the label of the bin $\boldsymbol{\mathcal{H}}_r$ *i.e.* the mixture component to which $\boldsymbol{\mathcal{H}}_r$

belongs; the label vectors $\boldsymbol{z}_r$ $(r = 1, \ldots, v)$ may have the form $\boldsymbol{z}_r = (z_{r1}, \ldots, z_{rK})$, where each z_{rk} $(k \in \{1, \ldots, K\})$ values 1 if $\boldsymbol{\mathcal{H}}_r$ is assumed as belonging to the mixture component k and values 0 otherwise.

The expectation of complete log-likelihood $Q(\boldsymbol{\Phi}, \boldsymbol{\Phi}^{(q)})$ is expressed as follows.

$$\begin{aligned} Q(\boldsymbol{\Phi}, \boldsymbol{\Phi}^{(q)}) &= E(L_C(\boldsymbol{\Phi})) \\ &= \sum_{k=1}^{K}\sum_{r=1}^{v} n_r \Big(\ln(\pi_k) p_{k/r}^{(q)} \\ &\quad + \frac{\pi_k^{(q)}}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} \ln\left(f_k(\boldsymbol{x}; \theta_k)\right) f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x} \Big) \end{aligned} \tag{2.5}$$

where

$$p_r^{(q)} = P(\boldsymbol{x} \in \boldsymbol{\mathcal{H}}_r | \boldsymbol{\Phi}^{(q)}) = \sum_{k=1}^{K} \pi_k^{(q)} \int_{\boldsymbol{\mathcal{H}}_r} f_k\Big(\boldsymbol{x}; \theta_k^{(q)}\Big) \mathrm{d}\boldsymbol{x}$$

and

$$\begin{aligned} p_{k/r}^{(q)} &= P\Big(z_k = 1 | \boldsymbol{x} \in \boldsymbol{\mathcal{H}}_r, \boldsymbol{\Phi}^{(q)}\Big) \\ &= \frac{\pi_k^{(q)} \int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}{p_r^{(q)}} \end{aligned}$$

2.4.2 The E-step and the M-step

Binned-EM algorithm executes two steps iteratively until convergence: the E-step (Expectation) and the M-step (Maximization). At the E-step, the $p_{k/r}^{(q)}$ and $p_r^{(q)}$ are calculated for all the k and r:

$$p_r^{(q)} = \sum_{k=1}^{K} \pi_k^{(q)} \int_{\boldsymbol{\mathcal{H}}_r} f_k\Big(\boldsymbol{x}; \theta_k^{(q)}\Big) \mathrm{d}\boldsymbol{x}$$

and

$$p_{k/r}^{(q)} = \frac{\pi_k^{(q)} \int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}{p_r^{(q)}}$$

At the M-step, the parameters $\boldsymbol{\Phi} = (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ maximizing $Q(\boldsymbol{\Phi}, \boldsymbol{\Phi}^{(q)})$ are estimated:

$$\pi_k^{(q+1)} = \frac{\sum_{r=1}^{v} n_r p_{k/r}^{(q)}}{n}$$

$$\boldsymbol{\mu}_k^{(q+1)} = \frac{\sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} \boldsymbol{x} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}{\sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}$$

To find the model parameters $\boldsymbol{\Sigma}_k^{(q+1)}$ maximizing $Q(\boldsymbol{\Phi}, \boldsymbol{\Phi}^{(q)})$, firstly, we select a part of $Q(\boldsymbol{\Phi}, \boldsymbol{\Phi}^{(q)})$ which includes $\boldsymbol{\Sigma}_k$:

$$\begin{aligned} F(\boldsymbol{\Sigma}_k) &= \sum_{k=1}^{K} \sum_{r=1}^{v} n_r \frac{\pi_k^{(q)}}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} \ln\left(f_k(\boldsymbol{x}; \theta_k)\right) f_k\left(\boldsymbol{x}; \theta_k^{(q)}\right) \mathrm{d}\boldsymbol{x} \\ &= \sum_{k=1}^{K} \sum_{r=1}^{v} n_r \frac{\pi_k^{(q)}}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} -\frac{1}{2}\big(\ln|\boldsymbol{\Sigma}_k| \\ &\quad +(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\big) f_k\left(\boldsymbol{x}; \theta_k^{(q)}\right) \mathrm{d}\boldsymbol{x} + C \end{aligned} \tag{2.6}$$

where C is a constant.

To maximize the $Q(\boldsymbol{\Phi}, \boldsymbol{\Phi}^{(q)})$ (Equation 2.5) leads to minimizing $B(\boldsymbol{\Sigma}_k)$ which is transformed from $F(\boldsymbol{\Sigma}_k)$:

$$\begin{aligned} B(\boldsymbol{\Sigma}_k) &= \sum_{k=1}^{K} \sum_{r=1}^{v} \pi_k^{(q)} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} \left(\ln|\boldsymbol{\Sigma}_k| + \left(\boldsymbol{x} - \boldsymbol{\mu}_k^{(q+1)}\right)^T \right. \\ &\quad \left. \cdot \boldsymbol{\Sigma}_k^{-1}\left(\boldsymbol{x} - \boldsymbol{\mu}_k^{(q+1)}\right)\right) f_k\left(\boldsymbol{x}; \theta_k^{(q)}\right) \mathrm{d}\boldsymbol{x} \\ &= \sum_{k=1}^{K} \pi_k^{(q)} \ln|\boldsymbol{\Sigma}_k| \sum_{r=1}^{v} \left(\frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k\left(\boldsymbol{x}; \theta_k^{(q)}\right) \mathrm{d}\boldsymbol{x} \right) \\ &\quad + \sum_{k=1}^{K} \pi_k^{(q)} \mathrm{tr}\left(\boldsymbol{\Sigma}_k^{-1} \mathbf{G}_k^{(q+1)}\right) \end{aligned} \tag{2.7}$$

where

$$\mathbf{G}_k^{(q+1)} = \sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} \left(\boldsymbol{x} - \boldsymbol{\mu}_k^{(q+1)}\right)\left(\boldsymbol{x} - \boldsymbol{\mu}_k^{(q+1)}\right)^T \cdot f_k\left(\boldsymbol{x}; \theta_k^{(q)}\right) \mathrm{d}\boldsymbol{x}$$

($\mathrm{tr}(\mathbf{X})$: the trace of the matrix $\mathbf{X}$ which is defined as the sum of the elements on the main diagonal of $\mathbf{X}$).

Minimizing the Equation (2.7) leads to the estimating result of the variance matrix $\boldsymbol{\Sigma}_k^{(q+1)}$, which is different depending on each parsimonious Gaussian mixture model. The variance matrix estimate for fourteen models will be detailed in the Section 2.5.

2.4.3 The complexity of binned-EM algorithm

In the sub-Section 2.2.2, we have calculated the complexity of the EM algorithm of the model $[\lambda \mathbf{DAD}^T]$. In this part, we will calculate the complexity of the binned-EM algorithm of the same model, in order to compare with the EM algorithm.

Firstly, in the Algorithm 2, the binned-EM algorithm of model $[\lambda \mathbf{DAD}^T]$ is shown:

Algorithm 2 Binned-EM algorithm

$q \leftarrow 0$

Initialize $\boldsymbol{\pi}^{(0)}$ and $\theta^{(0)} = \{\boldsymbol{\mu}^{(0)}, \boldsymbol{\Sigma}^{(0)}\}$.

repeat

for $r = 1 : v$ **do**

$p_r^{(q)} \leftarrow \sum_{k=1}^{K} \pi_k^{(q)} \int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}$

for $k = 1 : K$ **do**

$p_{k/r}^{(q)} \leftarrow \frac{\pi_k^{(q)} \int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}{p_r^{(q)}}$

end for

end for

for $k = 1 : K$ **do**

$\pi_k^{(q+1)} \leftarrow \frac{\sum_{r=1}^{v} n_r p_{k/r}^{(q)}}{n}$

$\boldsymbol{\mu}_k^{(q+1)} \leftarrow \frac{\sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\mathcal{H}_r} \boldsymbol{x} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}{\sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}$

$\boldsymbol{\Sigma}_k^{(q+1)} \leftarrow \frac{\sum_{k=1}^{K} \pi_k^{(q)} \sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\mathcal{H}_r} (\boldsymbol{x} - \boldsymbol{\mu}_k^{(q+1)})(\boldsymbol{x} - \boldsymbol{\mu}_k^{(q+1)})^T f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}{n}$

end for

$q \leftarrow q + 1$

until $\frac{L^{(q+1)} - L^{(q)}}{L^{(q)}} < \varepsilon$

$\widehat{\boldsymbol{\pi}} \leftarrow \boldsymbol{\pi}^{(q+1)}$, $\widehat{\theta} \leftarrow \theta^{(q+1)}$

Same as the EM algorithm, the binned-EM algorithm also executes the iterates until it converges. The stopping moment for binned-EM algorithm is flexible depending on the condition, more precisely, the ε. Same as what we did to EM algorithm, we suppose that the binned-EM algorithm stops in N iterates.

In order to facilitate the calculation, each iterate is divided into three parts:

- Calculate v times $p_r^{(q)}$.
- Calculate $v * K$ times $p_{k/r}^{(q)}$.
- Calculate K times $\pi_k^{(q+1)}$, $\boldsymbol{\mu}_k^{(q+1)}$ and $\boldsymbol{\Sigma}_k^{(q+1)}$.

In the Section 2.2, we calculated the complexity of

$$f_k(x_i;\theta_k^{(q)}) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}_k^{(q)}|^{1/2}} \exp(-1/2(x_i-\boldsymbol{\mu}_k^{(q)})\mathbf{\Sigma}_k^{(q)^{-1}}(x_i-\boldsymbol{\mu}_k^{(q)})^T) \tag{2.8}$$

which is approximately $O(d^3+4n_1+n_2^2)$.

In the binned-EM algorithm, to calculate $p_r^{(q)}$, we need to calculate:

$$\int_{\mathcal{H}_r} f_k(x;\theta_k^{(q)})\mathrm{d}\boldsymbol{x} = \int_{\mathcal{H}_r} \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}_k^{(q)}|^{1/2}} \tag{2.9}$$

$$\exp(-1/2((x-\boldsymbol{\mu}_k^{(q)})\mathbf{\Sigma}_k^{(q)^{-1}}(x-\boldsymbol{\mu}_k^{(q)})^T)\mathrm{d}\boldsymbol{x} \tag{2.10}$$

To calculate one dimensional numerical integral, we use the trapezoidal rule:

$$\int_b^a f(x)\mathrm{d}x \approx (b-a)\frac{f(a)+f(b)}{2} \tag{2.11}$$

To obtain a more accurate approximation, we can divide the interval $[a,b]$ into n subintervals. We add up all the approximations of the subintervals. This method is called the composite trapezoidal rule. So the Equation 2.11 can be extended into:

$$\int_b^a f(x)\mathrm{d} \approx \frac{b-a}{n}(\frac{f(a)}{2}+\sum_{k=1}^{n-1}(f(a+k\frac{b-a}{n}))+\frac{f(b)}{2}) \tag{2.12}$$

To compute integrals in multiple dimensions, one approach is to phrase the multiple integral as repeated one-dimensional integrals by appealing to Fubini's theorem:

$$\int_{A\times B} f(x,y)d(x,y) = \int_A (\int_B f(x,y)dy)\mathrm{d}x = \int_B (\int_A f(x,y)dx\mathrm{d}x)\mathrm{d}y$$

This approach requires the function evaluations to grow exponentially as the number of dimensions increases.

So to calculate $\int_{\mathcal{H}_r} f_k(x;\theta_k^{(q)})\mathrm{d}\boldsymbol{x}$, we suppose that we divide the interval on each dimension into l_l $(l=1,\ldots,d)$ subintervals. This means that, we execute $f_k(x;\theta_k^{(q)})$ function $(l_1+1)(l_2+1)\cdots(l_d+1)$ times. The complexity of calculating the Equation 2.9 is about $O((l_1+1)(l_2+1)\cdots(l_d+1)(d^3+4n_1+n_2^2))$.

For the rest of the binned-EM algorithm, we apply the same methods we mentioned before and in the Section 2.2. We list the complexity of each part of binned-EM algorithm in the Table 2.3:

Parameter	Times	Complexity
$p_r^{(q)}$	v	$O((l_1+1)(l_2+1)\cdots(l_d+1)(d^3K+4n_1K+n_2^2K))$
$p_{k/r}^{(q)}$	$v*K$	$O(1)$
$\pi_k^{(q+1)}$	K	$O(2v)$
$\boldsymbol{\mu}_k^{(q+1)}$	K	$O((l_1+1)(l_2+1)\cdots(l_d+1)v)$
$\boldsymbol{\Sigma}_k^{(q+1)}$	K	$O((l_1+1)(l_2+1)\cdots(l_d+1)3dvK)$

TABLE 2.3: Decomposition of complexity of the binned-EM algorithm.

From the Table 2.3, we can conclude that the complexity of the binned-EM algorithm is approximately $O((l_1+1)(l_2+1)\cdots(l_d+1)(d^3KvN+4n_1KvN+n_2^2KvN+3dvK^2N))$.

Comparing the complexity of the EM algorithm and the complexity of the binned-EM algorithm, if we suppose that the binned-EM algorithm is faster than the EM algorithm, we have the inequality function:

$$O(d^3KnN+3K^2dnN+4n_1KnN+n_2^2KnN) > O((l_1+1)(l_2+1)\cdots(l_d+1)(d^3KvN+4n_1KvN+n_2^2KvN+3dvK^2N))$$

Thus we obtain the condition that binned-EM algorithm is faster than the EM algorithm:

$$n > (l_1+1)(l_2+1)\cdots(l_d+1)v \tag{2.13}$$

If the amount of data n meets the condition $n > (l_1+1)(l_2+1)\cdots(l_d+1)v$, using the binned-EM algorithm is faster than using the EM algorithm.

2.5 Parsimonious models for binned-EM algorithm

In this section, we will present the derivation of variance matrix estimate of fourteen parsimonious models.

2.5.1 The general models

In the general family, there are eight models. They are: $[\lambda\mathbf{DAD}^T]$, $[\lambda_k\mathbf{DAD}^T]$, $[\lambda\mathbf{DA}_k\mathbf{D}^T]$, $[\lambda_k\mathbf{DA}_k\mathbf{D}^T]$, $[\lambda\mathbf{D}_k\mathbf{AD}_k^T]$, $[\lambda_k\mathbf{D}_k\mathbf{AD}_k^T]$, $[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$, $[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$.

Model $[\lambda \mathbf{DAD}^T]$. After equation simplification, minimizing Equation (2.7) equals to minimization of

$$\mathbf{M}_1(\mathbf{\Sigma}) = n \ln |\mathbf{\Sigma}| + \mathrm{tr}\Big(\mathbf{\Sigma}^{-1}\mathbf{G}^{(q+1)}\Big)$$

where

$$\mathbf{G}^{(q+1)} = \sum_{k=1}^{K} \pi_k^{(q)} \mathbf{G}_k^{(q+1)}$$

and the obtained estimated variance matrix $\mathbf{\Sigma}$ is then

$$\mathbf{\Sigma}^{(q+1)} = \frac{\mathbf{G}^{(q+1)}}{n}$$

Model $[\lambda_k \mathbf{DAD}^T]$. In this situation, it is easier to express it as $\mathbf{\Sigma}_k = \lambda_k \mathbf{C}$ with $\mathbf{C} = \mathbf{DAD}^T$. Minimizing Equation (2.7) equals to minimizing

$$\begin{aligned} M_2(\lambda_k, \mathbf{C}) &= \sum_{k=1}^{K} d \ln(\lambda_k) \sum_{r=1}^{v} \pi_k^{(q)} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k\Big(\boldsymbol{x}; \theta_k^{(q)}\Big) \mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{k+1}^{K} \pi_k^{(q)} \frac{1}{\lambda_k} \mathrm{tr}\Big(\mathbf{G}_k^{(q+1)} \mathbf{C}^{-1}\Big) \end{aligned}$$

We obtain λ_k and $\mathbf{C}$ minimizing M_2 by the following iterative process:

- Keep $\mathbf{C}$ fixed, the λ_k's minimizing $M_2(\lambda_k, \mathbf{C})$ are

$$\lambda_k^{(q+1)} = \frac{\mathrm{tr}(\mathbf{G}_k^{(q+1)} \mathbf{C}^{-1})}{d \sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k\Big(\boldsymbol{x}; \theta_k^{(q)}\Big) \mathrm{d}\boldsymbol{x}}$$

- keep λ_k's fixed, the matrix $\mathbf{C}$ minimizing $M_2(\lambda_k, \mathbf{C})$ is

$$\mathbf{C}^{(q+1)} = \frac{\sum_{k=1}^{K} \pi_k^{(q)} \frac{1}{\lambda_k} \mathbf{G}_k^{(q+1)}}{|\sum_{k=1}^{K} \pi_k^{(q)} \frac{1}{\lambda_k} \mathbf{G}_k^{(q+1)}|^{1/d}}$$

Model $[\lambda \mathbf{DA}_k \mathbf{D}^T]$. For this model, firstly is to minimize:

$$\begin{aligned} M_3(\lambda, \mathbf{D}, \mathbf{A}_k) &= d \ln(\lambda) \sum_{k=1}^{K} \sum_{r=1}^{v} n_r \frac{\pi_k^{(q)}}{p_r^{(q)}} \int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x} \\ &\quad + \frac{1}{\lambda} \sum_{k=1}^{K} \pi_k^{(q)} \mathrm{tr}(\mathbf{G}_k^{(q+1)} \mathbf{D} \mathbf{A}_k^{-1} \mathbf{D}^T) \end{aligned}$$

Minimizing M_3 leads to minimizing $\sum_{k=1}^{K} \pi_k^{(q)} tr(\mathbf{G}_k \mathbf{D}\mathbf{A}_k^{-1}\mathbf{D}^T)$ and calculation of λ. λ can be calculated directly:

$$\lambda^{(q+1)} = \frac{\sum_{k=1}^{K} \pi_k^{(q)} tr(\mathbf{G}_k^{(q+1)} \mathbf{D}\mathbf{A}_k^{-1}\mathbf{D}^T)}{nd} \tag{2.14}$$

We minimize $\sum_{k=1}^{K} \pi_k^{(q)} tr(\mathbf{G}_k^{(q+1)} \mathbf{D}\mathbf{A}_k^{-1}\mathbf{D}^T)$ using the following iterative method:

• Keeping $\mathbf{D}$ fixed, from Corollary A.5 of the Appendix A, we get

$$\mathbf{A}_k^{(q+1)} = \frac{diag(\mathbf{D}^T \pi_k^{(q)} \mathbf{G}_k^{(q+1)} \mathbf{D})}{|diag(\mathbf{D}^T \pi_k^{(q)} \mathbf{G}_k^{(q+1)} \mathbf{D})|^{1/d}} \tag{2.15}$$

• Keeping $\mathbf{A}_1^{(q+1)}, \ldots, \mathbf{A}_K^{(q+1)}$ fixed, we adapt an algorithm of Flury aiming to minimize $f(\mathbf{D}) = \sum_{k=1}^{K} \pi_k^{(q)} tr(\mathbf{G}_k^{(q+1)} \mathbf{D}\mathbf{A}_k^{-1}\mathbf{D}^T)$: First initial a solution $\mathbf{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_d)$. For any couple $(l, m)(l(\neq)m) \in 1, \ldots, d$, we find a corresponding couple $(\boldsymbol{\delta}_l, \boldsymbol{\delta}_m)$ which are orthogonal vectors, linear combination of $\boldsymbol{d}_l$ and $\boldsymbol{d}_m$, minimizing the criterion $f(\mathbf{D})$. We have

$$\begin{aligned}
\sum_{k=1}^{K} \mathrm{tr}(\mathbf{D}\pi_k^{(q)} \mathbf{A}_k^{-1}\mathbf{D}^T \mathbf{G}_k^{(k+1)}) &= \sum_{k=1}^{K} \sum_{j=1}^{d} \frac{\boldsymbol{d}_j^T \mathbf{G}_k^{(q+1)} \pi_k^{(q)} \boldsymbol{d}_j}{a_k^j} \\
&= \sum_{k=1}^{K} \frac{\boldsymbol{d}_l^T \mathbf{G}_k^{(q+1)} \pi_k^{(q)} \boldsymbol{d}_l}{a_k^l} + \sum_{k=1}^{K} \frac{\boldsymbol{d}_m^T \mathbf{G}_k^{(q+1)} \pi_k^{(q)} \boldsymbol{d}_m}{a_k^m} \\
&\quad + \sum_{k=1}^{K} \sum_{j \neq l,m} \frac{\boldsymbol{d}_j^T \mathbf{G}_k^{(q+1)} \pi_k^{(q)} \boldsymbol{d}_j}{a_k^j} \\
&= S(\boldsymbol{d}_l, \boldsymbol{d}_m) + \sum_{k=1}^{K} \sum_{j \neq l,m} \frac{\boldsymbol{d}_j^T \mathbf{G}_k^{(q+1)} \pi_k^{(q)} \boldsymbol{d}_j}{a_k^j}
\end{aligned}$$

Thus, it equals to find $(\boldsymbol{\delta}_l, \boldsymbol{\delta}_m)$ minimizing $S(\boldsymbol{d}_l, \boldsymbol{d}_m)$. We can write

$$\boldsymbol{\delta}_l = (\boldsymbol{d}_l, \boldsymbol{d}_m)\boldsymbol{q}_1$$

$$\boldsymbol{\delta}_m = (\boldsymbol{d}_l, \boldsymbol{d}_m)\boldsymbol{q}_2$$

where $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are two orthogonal vectors of R^2. We have

$$\begin{aligned} S(\boldsymbol{\delta}_l, \boldsymbol{\delta}_m) &= \sum_{k=1}^{K} \frac{\boldsymbol{q}_1^T (\boldsymbol{d}_l, \boldsymbol{d}_m)^T \mathbf{G}_k^{(q+1)} \pi_k^{(q)} (\boldsymbol{d}_l, \boldsymbol{d}_m) \boldsymbol{q}_1}{a_k^l} \\ &\quad + \sum_{k=1}^{K} \frac{\boldsymbol{q}_2^T (\boldsymbol{d}_l, \boldsymbol{d}_m)^T \mathbf{G}_k^{(q+1)} \pi_k^{(q)} (\boldsymbol{d}_l, \boldsymbol{d}_m) \boldsymbol{q}_2}{a_k^m} \\ &= \sum_{k=1}^{K} \frac{\boldsymbol{q}_1^T \mathbf{Z}_k \boldsymbol{q}_1}{a_k^l} + \sum_{k=1}^{K} \frac{\boldsymbol{q}_2^T \mathbf{Z}_k \boldsymbol{q}_2}{a_k^m} \end{aligned}$$

where

$$\mathbf{Z}_k = (\boldsymbol{d}_l, \boldsymbol{d}_m)^T \mathbf{G}_k^{(q+1)} \pi_k^{(q)} (\boldsymbol{d}_l, \boldsymbol{d}_m)$$

Denoting $\mathbf{Q} = (\boldsymbol{q}_1, \boldsymbol{q}_2)$, we get

$$\boldsymbol{q}_1^T \mathbf{Z}_k \boldsymbol{q}_1 + \boldsymbol{q}_2^T \mathbf{Z}_k \boldsymbol{q}_2 = tr(\mathbf{Q}^T \mathbf{Z}_k \mathbf{Q}) = tr(\mathbf{Z}_k)$$

And the problem reduces to the optimization of

$$S(\boldsymbol{d}_l, \boldsymbol{dm}) = \sum_{k=1}^{K} \frac{\boldsymbol{q}_1^T \mathbf{Z}_k \boldsymbol{q}_1}{a_k^l} + \sum_{k=1}^{K} \frac{tr(\mathbf{Z}_k - \boldsymbol{q}_1^T \mathbf{Z}_k \boldsymbol{q}_1)}{a_k^m}$$

which is equivalent to the minimization of

$$\boldsymbol{q}_1^T \{ \sum_{k=1}^{K} (\frac{1}{a_k^l} - \frac{1}{a_k^m}) \mathbf{Z}_k \} \boldsymbol{q}_1$$

Hence, $\boldsymbol{q}_1$ is the second eigenvector of the matrix $\sum_{k=1}^{K} (\frac{1}{a_k^l} - \frac{1}{a_k^m}) \mathbf{Z}_k$. Repeat the procedure above until $f(\mathbf{D})$ converges.

Model $[\lambda_k \mathbf{D} \mathbf{A}_k \mathbf{D}^T]$. Writing $\boldsymbol{\Sigma}_k = \mathbf{D} \mathbf{A}_k \mathbf{D}^T$ where $|\mathbf{A}_k| = |\boldsymbol{\Sigma}_k|$ is more efficient. Minimizing equation (2.7) leads to the minimization of

$$\begin{aligned} M_4(\lambda_k, \mathbf{D}, \mathbf{A}_k) &= \sum_{k=1}^{K} \ln(|\mathbf{A}_k|) \sum_{r=1}^{v} n_r \frac{\pi_k^{(q)}}{p_r^{(q)}} \int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{k=1}^{K} \pi_k^{(q)} \mathrm{tr}(\mathbf{G}_k^{(q+1)} \mathbf{D} \mathbf{A}_k^{-1} \mathbf{D}^T) \end{aligned}$$

As previously presented, the minimization of M_4 can be achieved in the similar way:

• Keeping $\mathbf{D}$ fixed, from Corollary A.7 of the Appendix A, we get

$$\mathbf{A}_k^{(q+1)} = \frac{diag(\mathbf{D}\mathbf{G}_k^{(q+1)}\mathbf{D}^T)}{\sum_{r=1}^{v} n_r \frac{\pi_k^{(q)}}{p_r^{(q)}} \int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}$$

• Keeping fixed $\mathbf{A}_1, \ldots, \mathbf{A}_K$, it can be making use of the same algorithm described above by minimizing $\sum_{k=1}^{K} \pi_k^{(q)} tr(\mathbf{D}\mathbf{A}_k^{-1}\mathbf{D}^T\mathbf{G}_k)$.

Model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$. Minimizing Equation (2.7) equals to minimizing

$$\begin{aligned} M_5(\lambda, \mathbf{D}_k, \mathbf{A}) &= d\ln(\lambda) \sum_{k=1}^{K}\sum_{r=1}^{v} n_r \frac{\pi_k^{(q)}}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k\Big(\boldsymbol{x}; \theta_k^{(q)}\Big) \mathrm{d}\boldsymbol{x} \\ &\quad + \frac{1}{\lambda} \sum_{k=1}^{K} \pi_k^{(q)} \mathrm{tr}\Big(\mathbf{G}_k^{(q+1)} \mathbf{D}_k \mathbf{A}^{-1} \mathbf{D}_k^{-1}\Big) \end{aligned}$$

Considering the eigenvalue decomposition $\mathbf{G}_k^{(q+1)} = \mathbf{L}_k^{(q+1)} \boldsymbol{\Omega}_k^{(q+1)} \mathbf{L}_k^{-1(q+1)}$, $k = 1, \ldots, K$, of the symmetric definite positive matrix $\mathbf{G}_k$ with the eigenvalues in the diagonal matrix $\boldsymbol{\Omega}_k$ in decreasing order,we have

$$\begin{aligned} M_5(\lambda, \mathbf{D}_k, \mathbf{A}) &= \frac{1}{\lambda} \sum_{k=1}^{K} \pi_k^{(q)} \mathrm{tr}\Big(\mathbf{D}_k^{-1} \mathbf{L}_k^{(q+1)} \boldsymbol{\Omega}_k^{(q+1)} \mathbf{L}_k^{-1(q+1)} \mathbf{D}_k \mathbf{A}^{-1}\Big) \\ &\quad + d\ln(\lambda) \sum_{k=1}^{K}\sum_{r=1}^{v} n_r \frac{\pi_k^{(q)}}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k\Big(\boldsymbol{x}; \theta_k^{(q)}\Big) \mathrm{d}\boldsymbol{x} \end{aligned}$$

From Theorem A.1 of Appendix A, we get $\mathbf{D}_k = \mathbf{L}_k$, and we have

$$\begin{aligned} M_5(\lambda, \mathbf{L}_k, \mathbf{A}) &= \frac{1}{\lambda} \sum_{k=1}^{K} \pi_k^{(q)} \mathrm{tr}\Big(\boldsymbol{\Omega}_k^{(q+1)} \mathbf{A}^{-1}\Big) \\ &\quad + d\ln(\lambda) \sum_{k=1}^{K}\sum_{r=1}^{v} n_r \frac{\pi_k^{(q)}}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k\Big(\boldsymbol{x}; \theta_k^{(q)}\Big) \mathrm{d}\boldsymbol{x} \end{aligned}$$

From this, we deduce $\mathbf{A}$ and λ:

$$\mathbf{A}^{(q+1)} = \frac{\sum_{k=1}^{K} \pi_k^{(q)} \boldsymbol{\Omega}_k^{(q+1)}}{|\sum_{k=1}^{K} \pi_k^{(q)} \boldsymbol{\Omega}_k^{(q+1)}|^{1/d}}$$

$$\lambda^{(q+1)} = \frac{|\sum_{k=1}^{K} \pi_k^{(q)} \boldsymbol{\Omega}_k^{(q+1)}|^{1/d}}{n}$$

Model $[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$. Minimizing Equation (2.7) equals to minimizing

$$\begin{aligned} M_6(\lambda_k, \mathbf{D}_k, \mathbf{A}) &= d \sum_{k=1}^{K} \pi_k^{(q)} \ln(\lambda_k) \sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k\Big(\boldsymbol{x}; \theta_k^{(q)}\Big) \mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{k=1}^{K} \pi_k^{(q)} \frac{1}{\lambda_k} \mathrm{tr}\Big(\mathbf{G}_k^{(k+1)} \mathbf{D}_k \mathbf{A}^{-1} \mathbf{D}_k^{-1}\Big) \end{aligned}$$

Use again the eigenvalue decomposition $\mathbf{G}_k^{(q+1)} = \mathbf{L}_k^{(q+1)} \mathbf{\Omega}_k^{(q+1)} \mathbf{L}_k^{-1(q+1)}$. The minimization of M_6 is achieved iteratively:

$$\lambda_k^{(q+1)} = \frac{\mathrm{tr}\Big(\mathbf{\Omega}_k^{(q+1)} \mathbf{A}^{-1}\Big)}{d \sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k\Big(\boldsymbol{x}; \theta_k^{(q)}\Big) \mathrm{d}\boldsymbol{x}}$$

$$\mathbf{D}_k = \mathbf{L}_k$$

and

$$\mathbf{A}^{(q+1)} = \frac{\sum_{k=1}^{K} \pi_k^{(q)} \frac{1}{\lambda_k} \mathbf{\Omega}_k^{(q+1)}}{|\sum_{k=1}^{K} \pi_k^{(q)} \frac{1}{\lambda_k} \mathbf{\Omega}_k^{(q+1)}|^{1/d}}$$

Model $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$. We consider the variance matrix as $\mathbf{\Sigma}_k = \lambda \mathbf{C}_k$ where $\mathbf{C}_k = \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T$. Then, minimizing Equation (2.7) equals to minimizing

$$M_7(\lambda, \mathbf{C}_k) = dn \ln(\lambda) + \frac{1}{\lambda} \sum_{k=1}^{K} \pi_k^{(q)} \mathrm{tr}\Big(\mathbf{C}_K^{-1} \mathbf{G}_k^{(q+1)}\Big)$$

Simple calculation give us:

$$\mathbf{C}_k^{(q+1)} = \frac{\mathbf{G}_k^{(q+1)}}{|\mathbf{G}_k^{(q+1)}|^{1/d}}$$

and

$$\lambda^{(q+1)} = \frac{\sum_{k=1}^{K} \pi_k^{(q)} |\mathbf{G}_k^{(q+1)}|^{1/d}}{n}$$

Model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$. This is the most general parsimonious model. Minimizing Equation (2.7) equals to minimizing

$$\begin{aligned} M_8(\mathbf{\Sigma}_k) &= \sum_{k=1}^{K} \pi_k^{(q)} \ln|\mathbf{\Sigma}_k| \sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k\Big(\boldsymbol{x}; \theta_k^{(q)}\Big) \mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{k=1}^{K} \pi_k^{(q)} \mathrm{tr}\Big(\mathbf{\Sigma}_k^{-1} \mathbf{G}_k^{(q+1)}\Big) \end{aligned}$$

and the variance matrices $\mathbf{\Sigma}_k^{(q+1)}$ are

$$\mathbf{\Sigma}_k^{(q+1)} = \frac{\mathbf{G}_k^{(q+1)}}{\sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}$$

2.5.2 The diagonal models

In the diagonal family, there are four models: $[\lambda\mathbf{B}]$, $[\lambda_k\mathbf{B}]$, $[\lambda\mathbf{B}_k]$ and $[\lambda_k\mathbf{B}_k]$.

Model $[\lambda\mathbf{B}]$. Minimizing Equation (2.7) leads to the minimization of

$$\begin{aligned} M_9(\lambda, \mathbf{B}) &= d\ln(\lambda) \sum_{k=1}^{K} \pi_k^{(q)} \sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x} \\ &\quad + \frac{1}{\lambda} \sum_{k=1}^{K} \pi_k^{(q)} \mathrm{tr}(\mathbf{B}^{-1}\mathbf{G}_k^{(q+1)}) \end{aligned}$$

To estimate the result, we used Corollary A.5 of the Appendix A. We get:

$$\mathbf{B}^{(q+1)} = \frac{diag(\sum_{k=1}^{K} \pi_k^{(q)} \mathbf{G}_k^{(q+1)})}{|diag(\sum_{k=1}^{K} \pi_k^{(q)} \mathbf{G}_k^{(q+1)})|^{1/d}}$$

and

$$\lambda^{(q+1)} = \frac{|diag(\sum_{k=1}^{K} \pi_k^{(q)} \mathbf{G}_k^{(q+1)})|^{1/d}}{n}$$

Model $[\lambda_k\mathbf{B}]$. In this situation, minimizing Equation (2.7) leads to the minimization of

$$\begin{aligned} M_{10}(\lambda_k, \mathbf{B}) &= d\sum_{k=1}^{K} \pi_k^{(q)} \ln(\lambda_k) \sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{k=1}^{K} \pi_k^{(q)} \frac{1}{\lambda_k} \mathrm{tr}(\mathbf{B}^{-1}\mathbf{G}_k^{(q+1)}) \end{aligned}$$

The minimization of the function M_{10} has to be performed iteratively.

• Keep $\mathbf{B}$ fixed, the λ_k's minimizing M_{10} are

$$\lambda_k^{(q+1)} = \frac{\mathrm{tr}(\mathbf{B}^{-1}\mathbf{G}_k^{(q+1)})}{d\sum_{r=1}^{v} \frac{n_r}{p_r^{(q)}} \int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}$$

• Keep the volumes λ_k's fixed, the matrix $\mathbf{B}$ minimizing M_{10} is minimizing $\sum_{k=1}^{K} \frac{\pi_k^{(q)}}{\lambda_k} \text{tr}(\mathbf{B}^{-1}\mathbf{G}_k^{(q+1)})$, thus, we have

$$\mathbf{B}^{(q+1)} = \frac{diag(\sum_{k=1}^{K} \frac{\pi_k^{(q)}}{\lambda_k}\mathbf{G}_k^{(q+1)})}{|diag(\sum_{k=1}^{K} \frac{\pi_k^{(q)}}{\lambda_k}\mathbf{G}_k^{(q+1)})|^{1/d}}$$

Model $[\lambda\mathbf{B}_k]$. In this situation, minimizing Equation (2.7) leads to the minimization of

$$\begin{aligned} M_{11}(\lambda, \mathbf{B}_k) &= d\ln(\lambda)\sum_{k=1}^{K}\pi_k^{(q)}\sum_{r=1}^{v}\frac{n_r}{p_r^{(q)}}\int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x};\theta_k^{(q)})\mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{k=1}^{K}\pi_k^{(q)}\frac{1}{\lambda}\text{tr}(\mathbf{B}_k^{-1}\mathbf{G}_k^{(q+1)}) \end{aligned} \tag{2.16}$$

It can be deduced directly from Equation (2.16) that:

$$\mathbf{B}_k^{(q+1)} = \frac{diag(\mathbf{G}_k^{(q+1)})}{|diag(\mathbf{G}_k^{(q+1)})|^{1/d}}$$

and

$$\lambda^{(q+1)} = \frac{\sum_{k=1}^{K}\pi_k^{(q)}|diag(\mathbf{G}_k^{(q+1)})|^{1/d}}{n}$$

Model $[\lambda_k\mathbf{B}_k]$. In this situation, minimizing Equation (2.7) leads to the minimization of

$$\begin{aligned} M_{12}(\lambda_k, \mathbf{B}_k) &= d\sum_{k=1}^{K}\pi_k^{(q)}\ln(\lambda_k)\sum_{r=1}^{v}\frac{n_r}{p_r^{(q)}}\int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x};\theta_k^{(q)})\mathrm{d}\boldsymbol{x} \\ &\quad + \sum_{k=1}^{K}\pi_k^{(q)}\frac{1}{\lambda_k}\text{tr}(\mathbf{B}_k^{-1}\mathbf{G}_k^{(q+1)}) \end{aligned} \tag{2.17}$$

It can be deduced directly from Equation (2.17) that:

$$\mathbf{B}_k^{(q+1)} = \frac{diag(\mathbf{G}_k^{(q+1)})}{|diag(\mathbf{G}_k^{(q+1)})|^{1/d}}$$

and

$$\lambda_k^{(q+1)} = \frac{|diag(\mathbf{G}_k^{(q+1)})|^{1/d}}{\sum_{r=1}^{v}\frac{n_r}{p_r^{(q)}}\int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x};\theta_k^{(q)})\mathrm{d}\boldsymbol{x}}$$

2.5.3 The spherical models

There are two models in the spherical family: $[\lambda\mathbf{I}]$, $[\lambda_k\mathbf{I}]$.

Model $[\lambda\mathbf{I}]$. In this situation, minimizing Equation (2.7) leads to the minimization of

$$\begin{aligned} M_{13}(\lambda) &= d\ln\lambda\sum_{k=1}^{K}\sum_{r=1}^{v}\pi_k^{(q)}\frac{n_r}{p_r^{(q)}}\int_{\boldsymbol{\mathcal{H}}_r} f_k\big(\boldsymbol{x};\theta_k^{(q)}\big)\mathrm{d}\boldsymbol{x} \\ &\quad +\sum_{k=1}^{K}\frac{\pi_k^{(q)}}{\lambda}\mathrm{tr}\big(\mathbf{G}_k^{(q+1)}\big) \\ &= nd\ln(\lambda)+\frac{1}{\lambda}\mathrm{tr}\big(\mathbf{G}^{(q+1)}\big) \end{aligned}$$

and

$$\mathbf{G}^{(q+1)}=\sum_{k=1}^{K}\pi_k^{(q)}\mathbf{G}_k^{(q+1)}$$

so we get

$$\lambda^{(q+1)}=\frac{\mathrm{tr}\big(\mathbf{G}^{(q+1)}\big)}{nd}$$

Model $[\lambda_k\mathbf{I}]$. In this situation, minimizing Equation (2.7) leads to the search of volume vector $\lambda=(\lambda_1,\ldots,\lambda_K)$ minimizing

$$\begin{aligned} M_{14}(\lambda_k) &= d\sum_{k=1}^{K}\pi_k^{(q)}\ln(\lambda_k)\sum_{r=1}^{v}\frac{n_r}{p_r^{(q)}}\int_{\boldsymbol{\mathcal{H}}_r} f_k\big(\boldsymbol{x};\theta_k^{(q)}\big)\mathrm{d}\boldsymbol{x} \\ &\quad +\sum_{k=1}^{K}\pi_k^{(q)}\frac{1}{\lambda_k}\mathrm{tr}\big(\mathbf{G}_k^{(q+1)}\big) \end{aligned}$$

and we get

$$\lambda_k^{(q+1)}=\frac{\mathrm{tr}\big(\mathbf{G}_k^{(q+1)}\big)}{d\sum_{r=1}^{v}\frac{n_r}{p_r^{(q)}}\int_{\boldsymbol{\mathcal{H}}_r} f_k\big(\boldsymbol{x};\theta_k^{(q)}\big)\mathrm{d}\boldsymbol{x}}$$

2.6 Experiments on simulated data

The numerical experiment is divided into three parts. The first part aims to compare the performances of the binned-EM algorithms of fourteen models on simulated data of different distributions. In the second part, we will study how the size of bin affects the result of binned-EM algorithm by changing the bin size. In the last part, we will test binned-EM algorithm on real applications: French cities clustering and image segmentation. To simplify the experiment and in order to be able to show the results in this paper visually, the simulated data are all defined in a two-dimensional space ($\mathbb{R}^2$).

2.6.1 Experiment on simulated data of different structures

The first experiment is to study the performances of different parsimonious models applied to data of different structures. According to fourteen parsimonious models, we simulate fourteen samples of different structure. Each sample size is of 3000. Fourteen versions of EM algorithm (each version associates to one parsimonious model) are applied to each sample for 30 times with random initiations. The best result among 30 results of each algorithm model is considered as the final result. The process of the experiment is explained in the Figure 2.5:

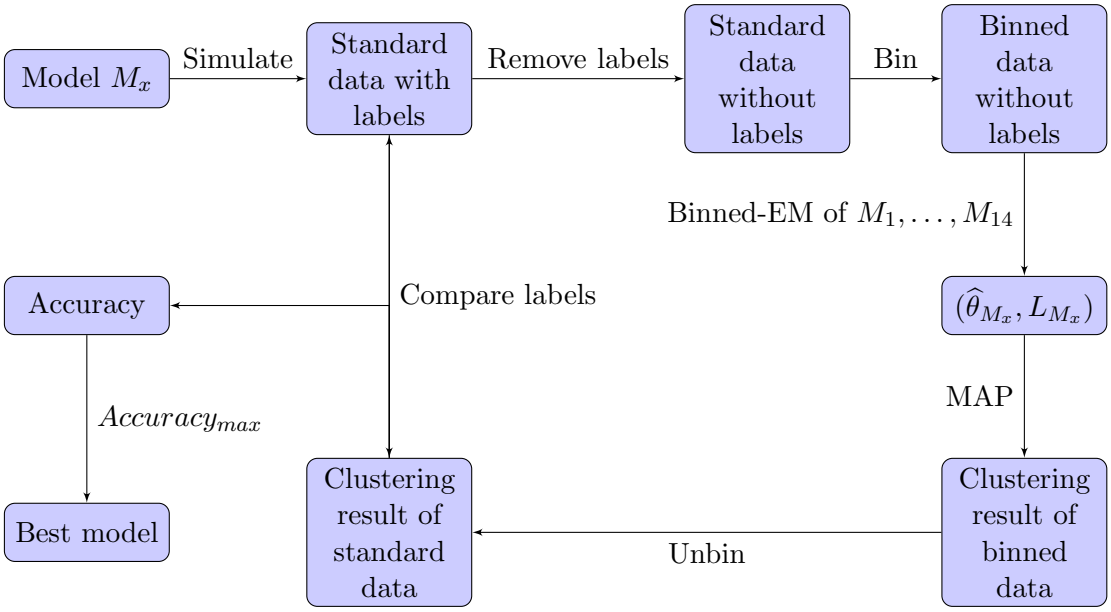


FIGURE 2.5: Experiment process of binned-EM algorithms of fourteen parsimonious models applying to simulated data.

Since the volumes, shapes and orientations are different among fourteen models, the separation of clusters within each model is controlled and defined by the distance value, which indicates the distance between two mixture components:

$$\delta = \sqrt{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T (\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2})^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)}$$

To obtain binned data, we divide the space into small bins of square shape with length= 0.5. Because of the different volumes and different centers of clusters in each data structure, the space is cut into different numbers of bins depending on the volumes of samples, which will be detailed in the description of the characteristics of dataset.

The simulated data contains two components of equal mixing portions in two-dimensional space. Characteristics of each structure of data are described as follows:

- Data structure 1 is generated according to the model $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$ with $\lambda = 1$, $\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A} = Diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-1.1, 0)$, $\boldsymbol{\mu}_2 = (1.2, 0)$, $\delta = 2.97$, Number of bins$= 23 \times 17$.

- Data structure 2 is generated according to the model $[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$ with $\lambda_1 = 1$, $\lambda_2 = 3$, $\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A} = Diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-1.5, 0)$, $\boldsymbol{\mu}_2 = (1.5, 0)$, $\delta = 2.74$, Number of bins$= 29 \times 28$.

- Data structure 3 is generated according to the model $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$ with $\lambda = 1$, $\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A}_1 = Diag(1, 1)$, $\mathbf{A}_2 = Diag(2, 1/2)$, $\boldsymbol{\mu}_1 = (-1.5, 0)$, $\boldsymbol{\mu}_2 = (1.5, 0)$, $\delta = 3.0$, Number of bins$= 21 \times 15$.

- Data structure 4 is generated according to the model $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$ with $\lambda_1 = 1$, $\lambda_2 = 2$, $\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A}_1 = Diag(1, 1)$, $\mathbf{A}_2 = Diag(2, 1/2)$, $\boldsymbol{\mu}_1 = (-1.5, 0)$, $\boldsymbol{\mu}_2 = (2, 0)$, $\delta = 2.93$, Number of bins$= 22 \times 20$.

- Data structure 5 is generated according to the model $[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$ with $\lambda = 1$, $\mathbf{D}_1 = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$, $\mathbf{D}_2 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A} = Diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-2, 0)$, $\boldsymbol{\mu}_2 = (1.5, 0)$, $\delta = 3.04$, Number of bins$= 23 \times 24$.

- Data structure 6 is generated according to the model $[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$ with $\lambda_1 = 3$, $\lambda_2 = 1$, $\mathbf{D}_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{D}_2 = Diag(1, 1)$, $\mathbf{A} = Diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-5, 0)$, $\boldsymbol{\mu}_2 = (5, 0)$, $\delta = 3.79$, Number of bins$= 47 \times 29$.

- Data structure 7 is generated according to the model $[\lambda \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$ with $\lambda = 1$, $\mathbf{D}_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{D}_2 = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$, $\mathbf{A}_1 = Diag(3, 1/3)$, $\mathbf{A}_2 = Diag(2, 1/2)$, $\boldsymbol{\mu}_1 = (-1.4, 1)$, $\boldsymbol{\mu}_2 = (1.5, -1)$, $\delta = 3.09$, Number of bins$= 19 \times 20$.

- Data structure 8 is generated according to the model $[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$ with $\lambda_1 = 2, \lambda_2 = 1$, $\mathbf{D}_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{D}_2 = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$, $\mathbf{A}_1 = Diag(2, 1/2)$, $\mathbf{A}_2 = Diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-1.8, 1)$ $\boldsymbol{\mu}_2$=(1.7,-1), $\delta = 3.05$, Number of bins$= 24 \times 27$.

- Data structure 9 is generated according to the model $[\lambda \mathbf{B}]$ with $\lambda = 1$, $\mathbf{B} = Diag(1/2, 2)$, $\boldsymbol{\mu}_1 = (-1, 0)$, $\boldsymbol{\mu}_2 = (1.1, 0)$, $\delta = 2.97$, Number of bins$= 13 \times 20$.

- Data structure 10 is generated according to the model $[\lambda_k \mathbf{B}]$ with $\lambda_1 = 1$, $\lambda_2 = 3$, $\mathbf{B} = Diag(1/3, 3)$, $\boldsymbol{\mu}_1 = (-4, 0)$, $\boldsymbol{\mu}_2 = (4, 0)$, $\delta = 3.27$, Number of bins$= 48 \times 12$.

- Data structure 11 is generated according to the model $[\lambda \mathbf{B}_k]$ with $\lambda = 1$, $\mathbf{B}_1 = Diag(1/2, 2)$, $\mathbf{B}_2 = Diag(1/3, 3)$, $\boldsymbol{\mu}_1 = (-1, 0)$, $\boldsymbol{\mu}_2 = (1, 0)$, $\delta = 3.09$, Number of bins$= 18 \times 24$.
- Data structure 12 is generated according to the model $[\lambda_k \mathbf{B}_k]$ with $\lambda_1 = 1$, $\lambda_2 = 3$, $\mathbf{B}_1 = Diag(2, 1/2)$, $\mathbf{B}_2 = Diag(4, 1/4)$, $\boldsymbol{\mu}_1 = (-3, 0)$, $\boldsymbol{\mu}_2 = (3, 0)$, $\delta = 3.03$, Number of bins$= 42 \times 11$.
- Data structure 13 is generated according to the model $[\lambda \mathbf{I}]$ with $\lambda_1 = 1$, $\boldsymbol{\mu}_1 = (-1.5, 0)$, $\boldsymbol{\mu}_2 = (1.5, 0)$, $\delta = 2.97$, Number of bins$= 27 \times 18$.
- Data structure 14 is generated according to the model $[\lambda_k \mathbf{I}]$ with $\lambda_1 = 1$, $\lambda_2 = 3$, $\boldsymbol{\mu}_1 = (-2.1, 0)$, $\boldsymbol{\mu}_2 = (2.1, 0)$, $\delta = 2.97$, Number of bins$= 28 \times 22$.

We evaluate the performance of each model by the accuracy, the CPU time and the standard deviation of accuracy. Accuracy indicates the percentage of the data which are correctly classified, while the CPU time is the amount of time for which a central processing unit (CPU) was used for our algorithm computation. The result is displayed in Tables 2.4, 2.5 and 2.6. The result in bold is the best result for the corresponding dataset.

According to the result, we can analyze in detail as follows: For data of structure $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$, the models in general family provide accuracies over 0.900, much higher than diagonal and spherical families. The best result is obtained by the model $[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$ as well as the model $[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$, instead of the model $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$ which underlies the data structure. These two models allow more freedom in the cluster parameters (different volumes, different volumes and orientation) than the model $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$. We want to mention that the right model $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$ provides a slightly better model result than the most general model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$. This outcome shows the suitability of parsimonious models for different datasets.

The highest accuracy for data structure $[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$ is obtained by its own feature model $[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$, as what is expected. Besides, the model $[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ and the most general model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ also provide high accuracies with low computation time. Not surprisingly the diagonal and spherical models perform disappointingly.

For data structure $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, it is of interest to see that all the models which allow different shapes in clusters provides good results: $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$, $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$. Among these models, the corresponding model $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$ has the highest accuracy.

When clustering data of structure $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, the right model $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$ has the highest accuracy and the least CPUtime. All the other models are able to provide a

Model Structure \ Data Structure		$[\lambda\mathbf{DAD}^T]$	$[\lambda_k\mathbf{DAD}^T]$	$[\lambda\mathbf{DA}_k\mathbf{D}^T]$	$[\lambda_k\mathbf{DA}_k\mathbf{D}^T]$	$[\lambda\mathbf{D}_k\mathbf{AD}_k^T]$
$[\lambda\mathbf{DAD}^T]$	Accuracy(σ)	0.926(0.0052)	0.914(0.0051)	0.918(0.0071)	0.921(0.0034)	0.925(0.0050)
	CPUtime	27.2	26.5	26.1	35.4	51.7
$[\lambda_k\mathbf{DAD}^T]$	Accuracy(σ)	0.959(0.0042)	**0.928(0.0019)**	0.902(0.0122)	0.903(0.0072)	0.927(0.0222)
	CPUtime	23.4	26.6	25.9	38.7	73.2
$[\lambda\mathbf{DA}_k\mathbf{D}^T]$	Accuracy(σ)	**0.960(0.0052)**	0.906(0.0072)	**0.929(0.0080)**	0.897(0.0093)	0.935(0.0145)
	CPUtime	23.3	48.0	28.2	38.8	57.3
$[\lambda_k\mathbf{DA}_k\mathbf{D}^T]$	Accuracy(σ)	0.913(0.0250)	0.812(0.0113)	0.927(0.0087)	**0.922(0.0080)**	0.826(0.0411)
	CPUtime	55.2	69.4	49.5	21.6	53.7
$[\lambda\mathbf{D}_k\mathbf{AD}_k^T]$	Accuracy(σ)	0.925(0.0050)	0.892(0.0055)	0.897(0.0142)	0.913(0.0065)	**0.960(0.0067)**
	CPUtime	51.7	111.2	38.8	39.5	33.2
$[\lambda_k\mathbf{D}_k\mathbf{AD}_k^T]$	Accuracy(σ)	**0.960(0.0052)**	0.927(0.0022)	0.895(0.0112)	0.909(0.0048)	0.950(0.0132)
	CPUtime	24.2	26.2	26.9	39.9	99.7
$[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	Accuracy(σ)	0.925(0.0051)	0.893(0.0065)	0.928(0.0070)	0.917(0.0093)	0.886(0.0430)
	CPUtime	50.4	90.0	49.4	80.7	45.3
$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	Accuracy(σ)	0.925(0.0052)	0.923(0.0085)	0.922(0.0067)	0.917(0.0053)	0.953(0.0026)
	CPUtime	27.2	26.4	26.1	38.7	27.8
$[\lambda\mathbf{B}]$	Accuracy(σ)	0.725(0.0232)	0.627(0.0046)	0.916(0.0128)	0.901(0.0065)	0.923(0.0095)
	CPUtime	45.0	64.3	55.9	38.8	43.1
$[\lambda_k\mathbf{B}]$	Accuracy(σ)	0.718(0.0007)	0.691(0.0189)	0.921(0.0056)	0.909(0.0086)	0.846(0.0094)
	CPUtime	53.8	149.3	53.8	55.9	76.0
$[\lambda\mathbf{B}_k]$	Accuracy(σ)	0.728(0.0227)	0.746(0.0228)	0.915(0.0061)	0.897(0.0085)	0.864(0.0081)
	CPUtime	44.1	85.3	55.9	38.7	31.1
$[\lambda_k\mathbf{B}_k]$	Accuracy(σ)	0.7175(0.0007)	0.721(0.0094)	0.912(0.0107)	0.910(0.0095)	0.850(0.0077)
	CPUtime	54.5	159.2	69.7	55.7	77.8
$[\lambda\mathbf{I}]$	Accuracy(σ)	0.734(0.0209)	0.746(0.172)	0.913(0.0153)	0.869(0.0106)	0.928(0.0065)
	CPUtime	39.9	71.6	69.7	132.6	53.5
$[\lambda_k\mathbf{I}]$	Accuracy(σ)	0.724(0.0140)	0.758(0.0098)	0.918(0.0064)	0.916(0.0112)	0.930(0.0070)
	CPUtime	39.9	68.8	37.4	42.1	52.0

TABLE 2.4: Accuracy, CPUtime and Standard deviation of accuracy (in parentheses) of binned-EM algorithm on simulated data 1

Data structure / Model Structure		$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda\mathbf{B}]$	$[\lambda_k\mathbf{B}]$
$[\lambda\mathbf{D}\mathbf{A}\mathbf{D}^T]$	Accuracy(σ)	0.941(0.0057)	0.925(0.0051)	0.925(0.0052)	0.725(0.0232)	0.948(0.0062)
	CPUtime	29.3	50.4	27.2	45.0	26.8
$[\lambda_k\mathbf{D}\mathbf{A}\mathbf{D}^T]$	Accuracy(σ)	0.925(0.0039)	0.817(0.0122)	0.988(0.0019)	0.908(0.0159)	0.959(0.0019)
	CPUtime	52.7	34.0	64.7	37.7	48.5
$[\lambda\mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	Accuracy(σ)	0.887(0.0215)	0.824(0.0087)	0.976(0.0065)	0.900(0.0131)	0.948(0.0063)
	CPUtime	53.2	34.8	69.1	26.1	26.5
$[\lambda_k\mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	Accuracy(σ)	0.933(0.0053)	0.787(0.0142)	0.966(0.0062)	0.783(0.0243)	0.946(0.0075)
	CPUtime	29.2	34.5	45.5	25.9	40.8
$[\lambda\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	Accuracy(σ)	0.950(0.0031)	0.980(0.0027)	0.967(0.0184)	0.896(0.0084)	0.948(0.0063)
	CPUtime	79.4	64.0	107.0	47.9	26.7
$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	Accuracy(σ)	**0.958(0.0045)**	0.787(0.0115)	0.991(0.0083)	0.907(0.0142)	0.959(0.0022)
	CPUtime	71.5	34.8	66.6	26.6	48.7
$[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	Accuracy(σ)	0.948(0.0055)	**0.983(0.0013)**	0.917(0.0110)	0.919(0.0111)	0.948(0.0059)
	CPUtime	69.5	88.5	43.7	34.2	26.6
$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	Accuracy(σ)	0.957(0.0056)	0.983(0.0036)	**0.992(0.0014)**	0.921(0.0079)	0.960(0.0030)
	CPUtime	61.0	40.7	67.7	86.9	48.5
$[\lambda\mathbf{B}]$	Accuracy(σ)	0.817(0.0072)	0.917(0.0151)	0.906(0.0211)	**0.923(0.0096)**	0.949(0.0066)
	CPUtime	124.7	26.2	26.5	41.2	26.6
$[\lambda_k\mathbf{B}]$	Accuracy(σ)	0.910(0.0065)	0.913(0.0072)	0.945(0.0081)	0.916(0.0158)	**0.960(0.0032)**
	CPUtime	204.9	64.4	64.5	72.5	69.7
$[\lambda\mathbf{B}_k]$	Accuracy(σ)	0.829(0.0050)	0.865(0.0070)	0.863(0.0080)	0.836(0.0159)	0.948(0.0064)
	CPUtime	149.4	35.4	40.4	18.3	26.5
$[\lambda_k\mathbf{B}_k]$	Accuracy(σ)	0.952(0.0014)	0.923(0.0117)	0.969(0.0053)	0.900(0.0155)	0.960(0.0035)
	CPUtime	93.3	66.0	64.8	50.3	69.7
$[\lambda\mathbf{I}]$	Accuracy(σ)	0.857(0.0076)	0.928(0.0072)	0.926(0.0068)	0.920(0.0055)	0.939(0.0066)
	CPUtime	133.0	28.5	48.1	54.2	69.4
$[\lambda_k\mathbf{I}]$	Accuracy(σ)	0.919(0.0096)	0.911(0.0060)	0.945(0.0048)	0.920(0.0076)	0.956(0.0046)
	CPUtime	45.5	36.7	51.0	50.9	48.3

TABLE 2.5: Accuracy, CPUtime and Standard deviation of accuracy (in parentheses) of binned-EM algorithm on simulated data 2

Model Structure \ Data structure		$[\lambda\mathbf{B}_k]$	$[\lambda_k\mathbf{B}_k]$	$[\lambda\mathbf{I}]$	$[\lambda_k\mathbf{I}]$
$[\lambda\mathbf{DAD}^T]$	Accuracy(σ)	0.828(0.0194)	0.903(0.0073)	0.934(0.0209)	0.924(0.0140)
	CPUtime	15.1	37.9	39.9	39.3
$[\lambda_k\mathbf{DAD}^T]$	Accuracy(σ)	0.834(0.0186)	0.930(0.0072)	0.931(0.0062)	0.945(0.0038)
	CPUtime	17.3	38.1	26.7	48.6
$[\lambda\mathbf{DA}_k\mathbf{D}^T]$	Accuracy(σ)	0.823(0.0152)	0.926(0.0060)	0.931(0.0062)	0.9334(0.0071)
	CPUtime	16.0	37.7	26.5	48.7
$[\lambda_k\mathbf{DA}_k\mathbf{D}^T]$	Accuracy(σ)	0.821(0.0203)	0.911(0.0108)	0.929(0.0066)	0.942(0.0060)
	CPUtime	14.9	45.9	58.9	29.0
$[\lambda\mathbf{D}_k\mathbf{AD}_k^T]$	Accuracy(σ)	0.865(0.0071)	0.908(0.0076)	0.885(0.0099)	0.862(0.0105)
	CPUtime	35.2	73.4	47.8	47.9
$[\lambda_k\mathbf{D}_k\mathbf{AD}_k^T]$	Accuracy(σ)	0.865(0.0084)	0.916(0.0084)	0.931(0.0062)	0.945(0.0036)
	CPUtime	39.9	38.1	26.7	50.2
$[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	Accuracy(σ)	0.865(0.0060)	0.922(0.0048)	**0.982(0.0136)**	0.952(0.0099)
	CPUtime	32.1	54.5	51.3	34.3
$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	Accuracy(σ)	0.864(0.0048)	0.934(0.0070)	0.973(0.0090)	**0.969(0.0099)**
	CPUtime	39.3	37.6	66.3	45.7
$[\lambda\mathbf{B}]$	Accuracy(σ)	0.838(0.0187)	0.921(0.0058)	0.920(0.0102)	0.837(0.0263)
	CPUtime	17.4	71.1	38.3	37.0
$[\lambda_k\mathbf{B}]$	Accuracy(σ)	0.763(0.0197)	0.893(0.0130)	0.931(0.0062)	0.945(0.0040)
	CPUtime	35.0	163.0	41.6	70.4
$[\lambda\mathbf{B}_k]$	Accuracy(σ)	0.861(0.0058)	0.927(0.0060)	0.819(0.0151)	0.805(0.0216)
	CPUtime	27.1	64.6	26.1	27.8
$[\lambda_k\mathbf{B}_k]$	Accuracy(σ)	**0.870(0.0057)**	**0.942(0.0053)**	0.931(0.0062)	0.945(0.0037)
	CPUtime	65.4	64.4	41.7	70.7
$[\lambda\mathbf{I}]$	Accuracy(σ)	0.819(0.0169)	0.907(0.0097)	0.928(0.0071)	0.928(0.0066)
	CPUtime	27.0	30.8	28.6	47.8
$[\lambda_k\mathbf{I}]$	Accuracy(σ)	0.792(0.0237)	0.924(0.0068)	0.913(0.0079)	0.945(0.0048)
	CPUtime	27.2	37.4	91.2	42.8

TABLE 2.6: Accuracy, CPUtime and Standard deviation of accuracy (in parentheses) of binned-EM algorithm on simulated data 3

good result. This is because the difference between two clusters doesn't show a strong character in the dataset.

Data of structure $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ contains clusters of different orientations. The models which contain clusters of different orientations obtain good result (accuracy more than 0.950): $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$, $[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$, $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$. And the best result is still obtained by the right model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$.

The best result of cluster analysis of data of structure $[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ is obtained by its own feature model. The most general model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ provides the second best result. The other general models give a better result than the diagonal and spherical models. Surprisingly the model $[\lambda_k \mathbf{B}_k]$ provides high accuracy too because the samples we generated has the structure closed to $[\lambda_k \mathbf{B}_k]$ and two clusters in the data are well separated.

For the data of structure $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$, models $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ and $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ have very high accuracy 0.983, yet the model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ spends less computation time than the model $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$.

The data contains clusters with different volumes, orientations and shapes ($[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$) is the most difficult case to deal with. In this situation, undoubtedly the most complex model obtains the highest accuracy because it is the only model which can represent correctly the structure of the data.

And it is interesting to notice that among the general family, the models allowing different volumes have worse performances than the models restricting the same volumes. Some parsimonious models which are simpler than the structure of data can still obtain good results because those models can be closed to the structure of the simulated data. And this is also because the two clusters are well separated in our simulated dataset. For example, data simulated according to the model $[\lambda_k \mathbf{D} \mathbf{A}_k \mathbf{D}^T]$ are very close to the structure of the model $[\lambda_k \mathbf{D} \mathbf{A}_k \mathbf{D}^T]$.

For diagonal data structure $[\lambda \mathbf{B}]$, we notice that as the other general models, its own feature model $[\lambda \mathbf{B}]$ provides the best result, even the other models who is more complex than $[\lambda \mathbf{B}]$ cannot obtain better result.

The two clusters in the data simulated according to model $[\lambda_k \mathbf{B}]$ are well separated, which leads to all the results of fourteen models are almost the same. The models $[\lambda_k \mathbf{B}]$, $[\lambda_k \mathbf{B}_k]$ and $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ which have the closest structure to the data, provide the highest accuracy.

The model $[\lambda \mathbf{B}_k]$ is not the model obtaining the highest accuracy for the data structured of $[\lambda \mathbf{B}_k]$. But it obtains a good result when taking less CPUtime. The model $[\lambda_k \mathbf{B}_k]$

appears to be the best model for data structured $[\lambda_k \mathbf{B}_k]$. The diagonal family and spherical family show equivalent performances as the general family in this case.

When data is simulated according to the spherical models $[\lambda \mathbf{I}]$ and $[\lambda_k \mathbf{I}]$, models $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ and $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ provide the highest precision respectively. Models $[\lambda \mathbf{I}]$ and $[\lambda_k \mathbf{I}]$ are very simple, but they still can do a good cluster analysis with high accuracy and less computation time than the general models. Generally speaking, the general models provide good result on the data simulated according to the diagonal models and spherical models. Because each diagonal and spherical model can be described by several general models, but in a more complicated form.

After all the analysis above, we can conclude as follows:

In most of the case, the result shows that the model which can represent the data structure obtains the best highest accuracy. Generally speaking, the general models provide better results than the diagonal and spherical models when the data is simulated according to the general model. And naturally, when the the data generated according to diagonal models, diagonal models offer the best result. It explains that it's not necessary to use the most general model for all the data clustering. When the clusters are well separated, the difference among fourteen models could be reduced. When the clusters are well-mixed, choosing the right model plays a more important role in achieving a good clustering result. We notice that in general the standard deviation of a model is lower when this model provides higher accuracy. It means that when the model is more suitable for the data, the result is even more stable and more reliable. In this paper, we need to compare the fourteen models at the same levels, so we program all the models at the same manner which contains certain numerical multidimensional integration. Due to the methods in programming and the capability of calculators, the CPUtime in this paper cannot be compared to the experimental results of EM algorithm and binned-EM algorithm in previous papers. As indicated and proved in the papers of Cadez et al. [13], Samé et al. [14], Samé [55], binning data in the mixture and classification approach helps in reducing computation time. Also, in the previous section, we have calculated and compared the computational complexities of EM algorithm and binned-EM algorithm. Thus here in this section, the CPUtime is only to give a reference among fourteen models and to show the difference of computation time of different models. This experiment shows that if we are able to choose the right parsimonious model for the data, we can receive higher accuracy and spend less computation time.

2.6.2 Experiment on simulated data with different bin sizes

As we know, in binned-EM algorithm, we assume that the only information of the data is the frequencies of bins. The change in bin size can directly affect on the frequencies of bins. It plays an inevitably important role in binned-EM algorithm.

Logically speaking, smaller bins lead to higher accuracy but more computation time. The situation reaches to extreme when all the frequencies of non-empty bins equal 1. In this case, the number of non-empty bins equals the number of observations. On the contrary, bigger bins lead to worse accuracy but less CPUtime. From this experiment, we will see how the accuracy and computation time of binned-EM algorithm vary with the bin size.

All the samples are simulated according to the model $[\lambda \mathbf{DAD}^T]$, with $\lambda = 1$, $\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A} = diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-1.3, 0)$ and $\boldsymbol{\mu}_2 = (1.5, 0)$. To obtain the binned data, we divide the space into small bins of square form. 11 sizes of bins will be considered. The side-length of bins varies from 0.2 to 0.7 with interval of 0.05. So the numbers of bins are 56×45, 48×34, 35×32, 32×25, 29×21, 26×20, 24×18, 21×15, 19×15, 17×14, 16×13. For each bin size, we simulate a sample of $size = 3000$. On each sample we apply binned-EM algorithm of model $[\lambda \mathbf{DAD}^T]$ 30 times with random initiation. We evaluate each performance by accuracy and CPUtime. The best result from the 30 results of each size of bins is considered as the final result.

The results are shown in the Table 2.7 and the Figures 2.6 and 2.7.

Bin size	Accuracy	CPUtime	bin num.	non-empty-bin num.
$0.2 \cdot 0.2$	0.9637	70	2976	751
$0.25 \cdot 0.25$	0.9632	50	1518	517
$0.3 \cdot 0.3$	0.9630	38	1189	401
$0.35 \cdot 0.35$	0.9615	30	744	304
$0.4 \cdot 0.4$	0.9611	24	660	253
$0.45 \cdot 0.45$	0.9593	19	535	210
$0.5 \cdot 0.5$	0.9603	16	428	174
$0.55 \cdot 0.55$	0.9582	14	349	149
$0.6 \cdot 0.6$	0.9576	12	286	132
$0.65 \cdot 0.65$	0.9557	11	234	114
$0.7 \cdot 0.7$	0.9549	10	213	103

TABLE 2.7: Result of binned-EM algorithm with different size of bins on simulated data.

From the Table 2.7 and the Figure 2.6, we can see the accuracy of binned-EM algorithm keeps steady at a high level while slightly reducing from 0.9637 to 0.9549 with the increasing of bin size from $0.2 \cdot 0.2$ to $0.7 \cdot 0.7$. From Figure 2.7, it is shown that CPUtime decreases when the bins become bigger. Because our algorithm only depends on the bins which are not empty. When the bins are bigger, there are less bins as well as non-empty-bins and surely it costs less computation time. By comparing the

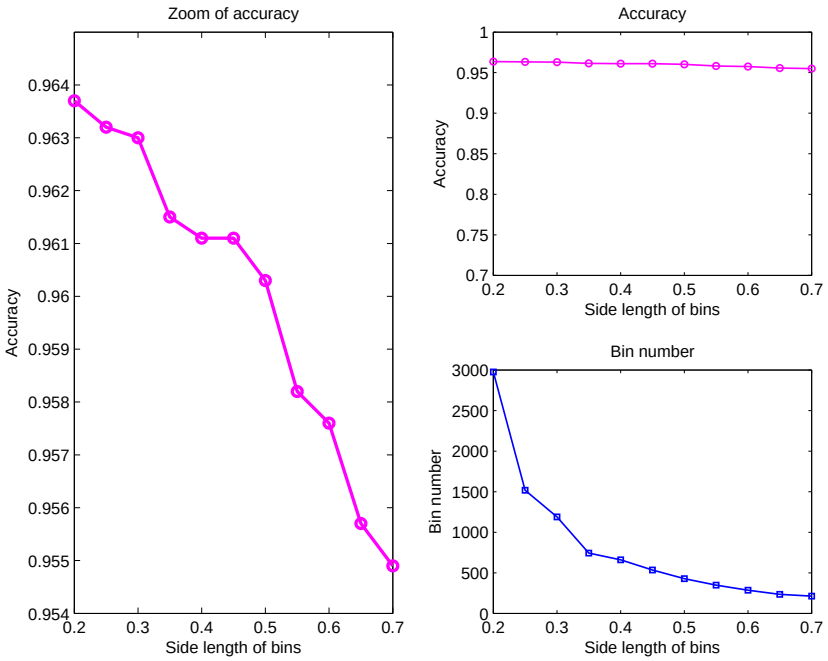


FIGURE 2.6: Accuracy and bin number of binned-EM algorithm applied on simulated data with different size of bins.

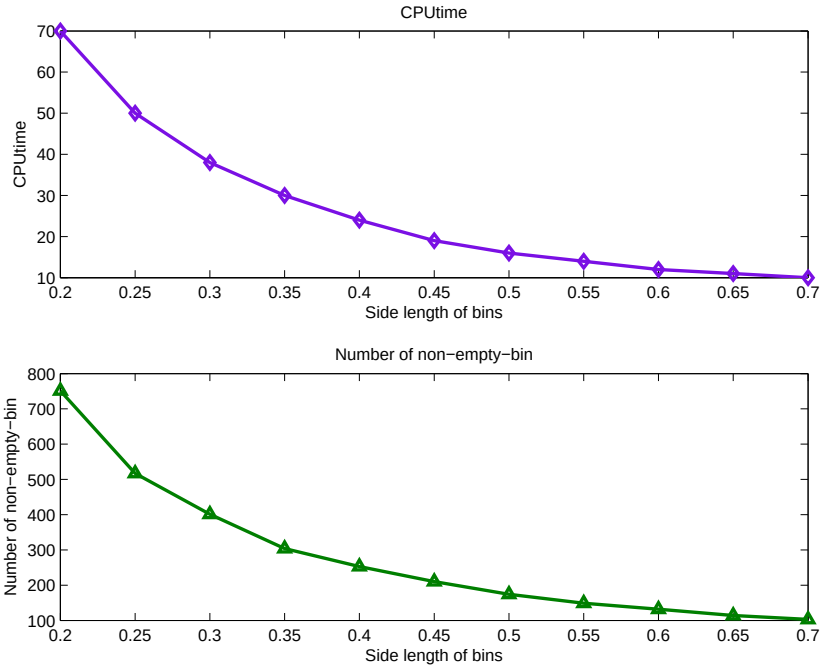


FIGURE 2.7: CPUtime and non-empty-bin number of binned-EM algorithm applied on simulated data with different size of bins.

two small figures in Figure 2.7, we notice that the decrease of CPUtime trends to the similar tendency as the decrease of non-empty-bin number. So we can conclude that the computation time of binned-EM algorithm can be reduced by enlarging the size of bins, with certain loss of precision.

2.7 Experiments on real data

2.7.1 French city clustering

Experiment on real data can test the algorithm's practical ability in real application. To test our algorithm, as the real dataset, we have the population and population density (population/surface) of 1193 cities from three departments in France: Meuse, Nord and Val-de-Marne. These cities have different characteristic in population and population density due to different locations. Meuse is a rural department with a small population and low population density. In the opposite, Val-de-Marne, situated to southeast of Paris, is a department of high population density and Nord is the most populous department in France. We assume that we only have the information about the population and density of these 1193 cities and we need to group them into three clusters by binned-EM algorithm. The result of binned-EM algorithm will be compared to the actually clusters in reality. Accuracy will be calculated and CPUtime will be recorded.

Figure 2.8 displays 1193 observations concerning the log-population and log-density of cities from Meuse (500 observations), Nord (652 observations) and Val-de-Marne (47 observations). From the figure, we can tell that there are three clusters. It is obvious that the structure of each cluster is neither spherical nor diagonal. Thus there is no need to test also the binned-EM algorithms of diagonal models and spherical models. We only apply eight general models on this real dataset: $[\lambda\mathbf{DAD}^T]$, $[\lambda_k\mathbf{DAD}^T]$, $[\lambda\mathbf{DA}_k\mathbf{D}^T]$, $[\lambda_k\mathbf{DA}_k\mathbf{D}^T]$, $[\lambda\mathbf{D}_k\mathbf{AD}_k^T]$, $[\lambda_k\mathbf{D}_k\mathbf{AD}_k^T]$, $[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$ and $[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$. Each model is applied on the real data for 30 times. We pick up the best accuracy from 30 results as the final result of the model.

The result is displayed in Table 2.8 and Figures 2.9 and 2.10.

From Figure 2.9, three departments are basically correctly clustered. From Figure 2.10, the observations which are incorrectly clustered mostly locate in the intersection of two departments. For the point of accuracy, both model $[\lambda_k\mathbf{DA}_k\mathbf{D}^T]$ and model $[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$ obtain the highest accuracy: 0.813. For the point of computation time, model $[\lambda_k\mathbf{DA}_k\mathbf{D}^T]$ consumes the least CPUtime among the eight models: 33.7, while

model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ takes the second least CPUtime: 58.42. Thus, we can consider the best model in clustering the French cities by its population and population density is model $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$. It proves the practical application ability of binned-EM algorithm and the advantage of parsimonious models in saving computation time.

Model	Accuracy	Time
$[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$	0.780	58.72
$[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$	0.801	110.78
$[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	0.789	85.60
$[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	0.813	33.07
$[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	0.755	59.80
$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	0.791	60.11
$[\lambda \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	0.752	221.65
$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	0.813	58.42

TABLE 2.8: Accuracy, CPUtime of binned-EM algorithm on French cities clustering.

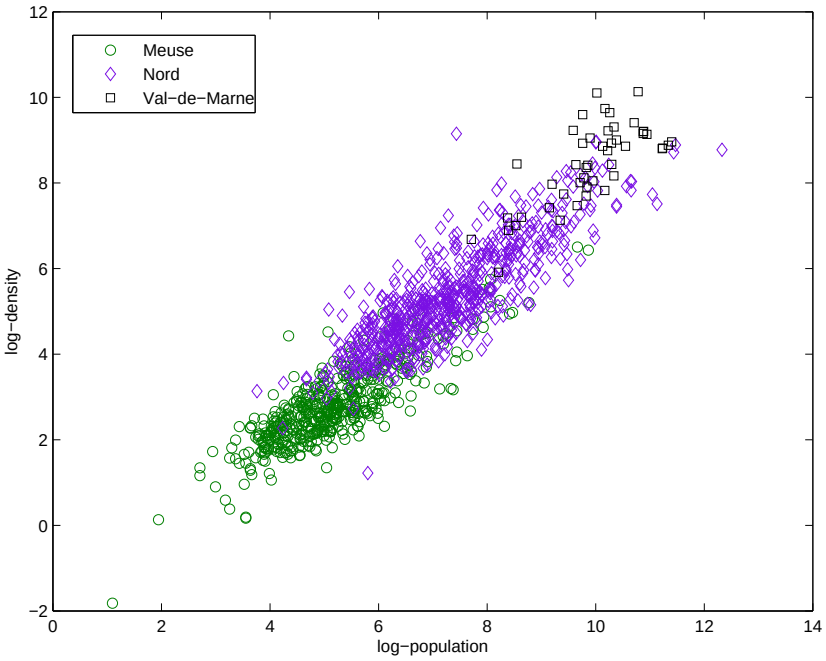


FIGURE 2.8: Log-population and log-density of 1193 cities from three departments in France.

2.7.2 Image segmentation

Image segmentation, feature extraction and object recognition constitute three main objectives in computer vision theory. The feature extraction and object recognition are based on image segmentation. The result of image segmentation will directly affect the subsequent feature extraction and object recognition. Image segmentation is a process of extracting meaningful features or regions from image. These features can be the

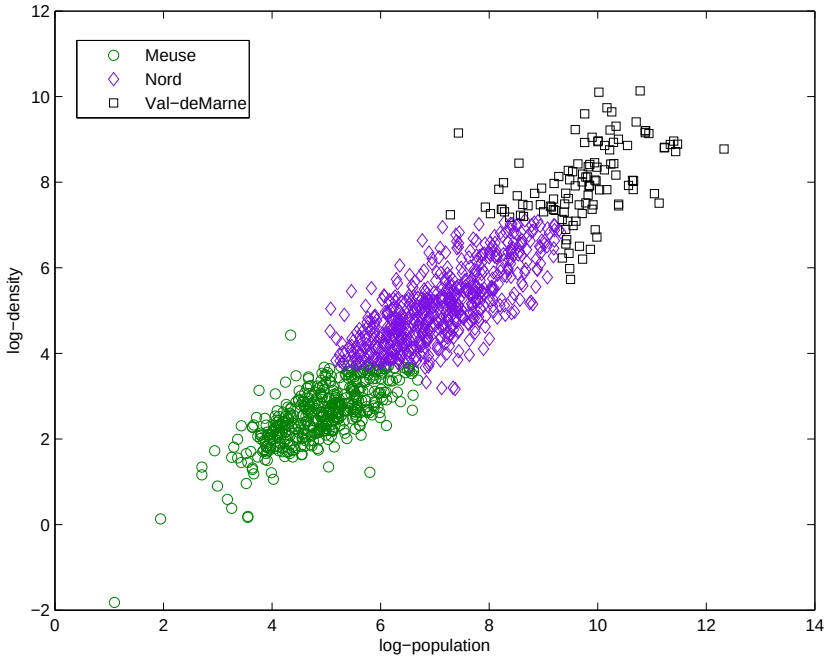


FIGURE 2.9: Binned-EM algorithm result on real data on French cities.

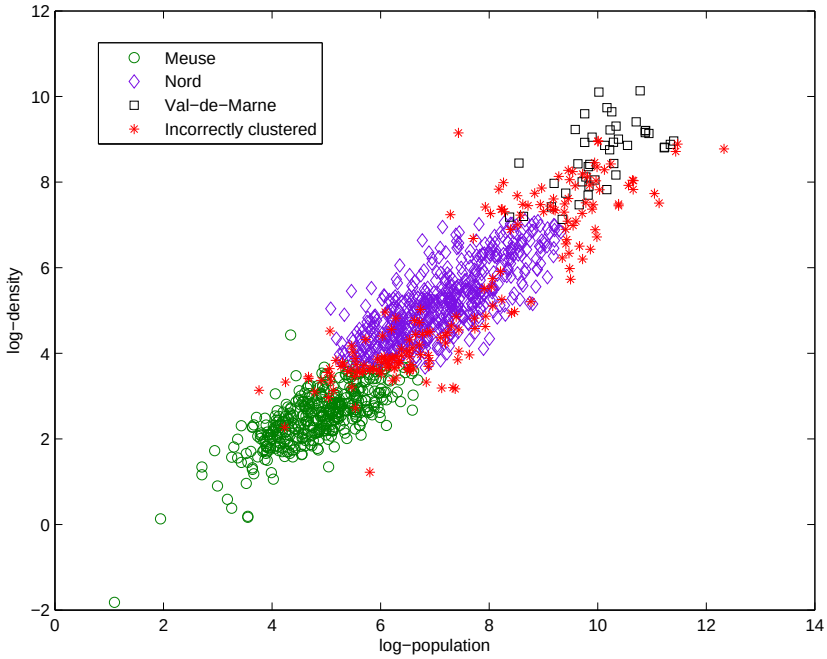


FIGURE 2.10: Incorrectly clustered points of binned-EM algorithm result on real data on French cities.

original image features, such as the gray value of the pixel, the object contour, the color and reflection characteristics. The purpose of image segmentation is to divide the image into several disjoint regions so that each region has the consistency and adjacent areas have significant difference in property. This helps to better understand and analyze the image.

Image segmentation is widely used in all the areas of image processing. For example, it can be used in medical imaging including locating tumors and other pathologies. It can also be used in face detection, fingerprint recognition, etc. Many methods of image segmentation were proposed. Image segmentation techniques can be classified into two broad families: region-based, and contour-based approaches. Region-based approaches try to find partitions of the image pixels into sets corresponding to coherent image properties such as brightness, color and texture. Contour-based approaches detects the places where the gray level or structure has a mutation, where indicating the end of a region, but also the place to start another region.

Many other methods for image segmentation were proposed. One of them is called clustering method. Feature space clustering method for image segmentation is to express the pixels by corresponding points in the feature space. Partition the feature space according to the accumulation of the points in the feature space. Then map them back to the original image space, to obtain the segmentation result. Where, K-means, Fuzzy C-means clustering (FCM) algorithm are the most commonly used clustering algorithms.

In this part, we will apply binned-EM algorithm to image segmentation. As we know, all the color can be represented in RGB Color space. In this experiment, we will convert image from RGB color space to $L*a*b*$ color space. Thus, colors will be segmented in an automated fashion using the $L*a*b*$ color space. The $L*a*b*$ color space (also known as CIELAB or CIE $L*a*b*$) enables you to quantify the visual differences among colors. The $L*a*b*$ color space is derived from the CIE XYZ tristimulus values. The $L*a*b*$ space consists of a luminosity layer $'L*'$, chromaticity-layer $'a*'$ indicating where color falls along the red-green axis, and chromaticity-layer $'b*'$ indicating where the color falls along the blue-yellow axis. All of the color information is in the $'a*'$ and $'b*'$ layers.

We are going to do image segmentation of the image in the Figure 2.11:

FIGURE 2.11: Original image.

From the image, we can tell that there are four main colors: year(the sleds), white(the sky and the snow), gray(the mountain and the shadow) and black(the trees). Thus in the following process, we will cluster the image pixels into four clusters basing on the color.

After converting the image into $L*a*b*$ color space. Since the color information exists in the $'a*b*'$ space, the data become pixels with $'a*'$ and $'b*'$ values. The Figure 2.12 shows the 154401 image pixels:

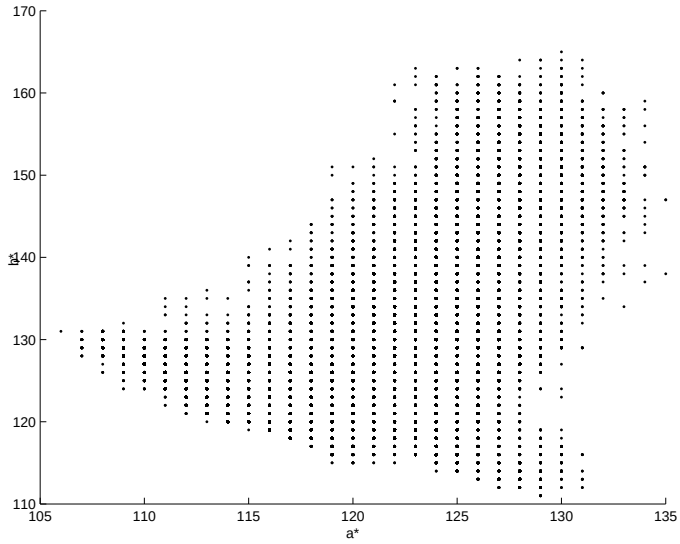
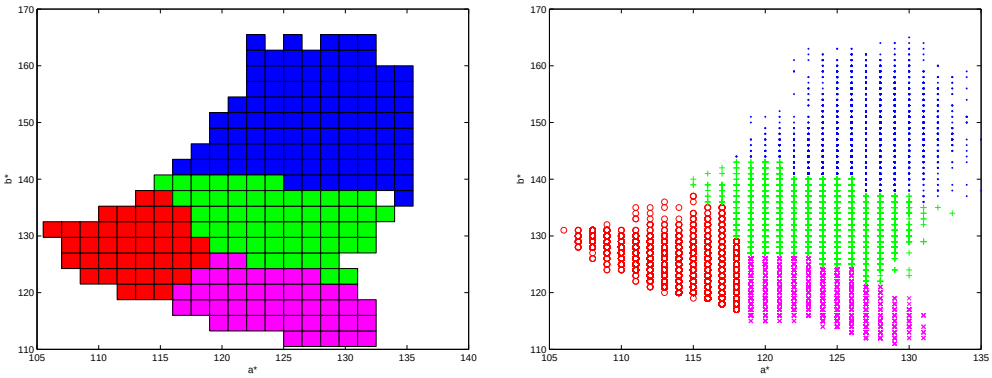


FIGURE 2.12: Image pixel represented in the $'a * b*'$ space.

From the Figure 2.12, it is hard to tell which model it corresponds to. We try the general model $[\lambda \mathbf{DAD}^T]$. Figure 2.13 shows the clustering result by binned-EM algorithm of model $[\lambda \mathbf{DAD}^T]$ with $20 * 20$ bins.

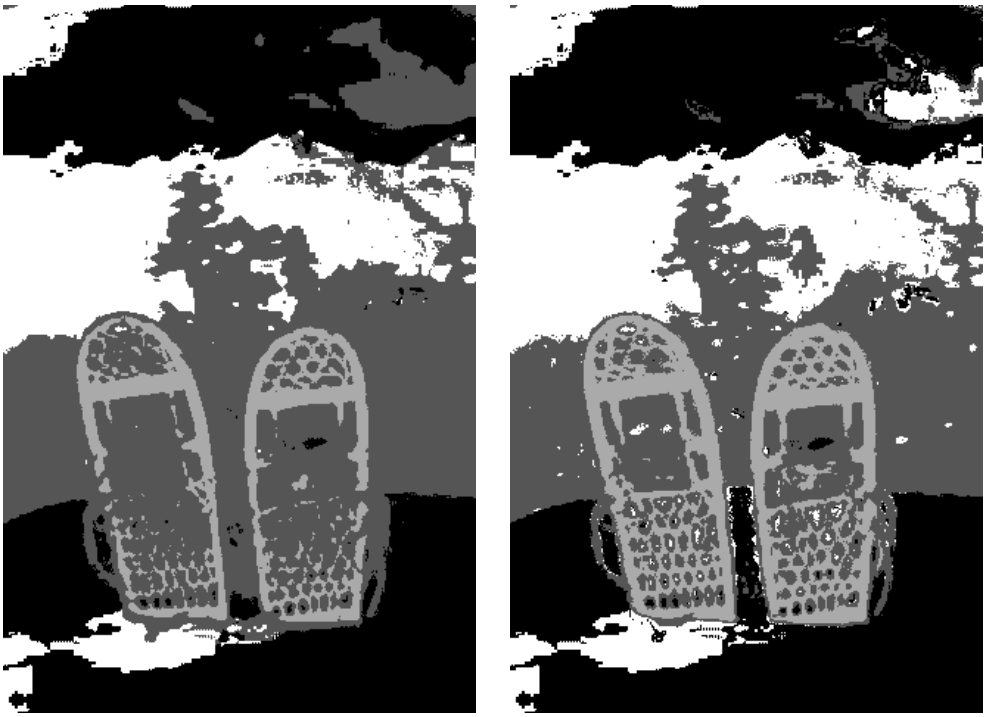


$Clusteringresultinbinneddataform$ $Clusteringresultinstandarddataform$

FIGURE 2.13: Clustering result by binned-EM algorithm of model $[\lambda \mathbf{DAD}^T]$ with $20 * 20$ bins.

2.7.2.1 With different sizes of bin

In the Figures 2.14 and 2.15 shows the image segmentation result by binned-EM algorithm of model $[\lambda \mathbf{DAD}^T]$ with different size of bins. The bin size changes among 5, 10, 20, 30, 40 and 50 bins per dimension. This aims to see how size of bins affects the result of image segmentation.



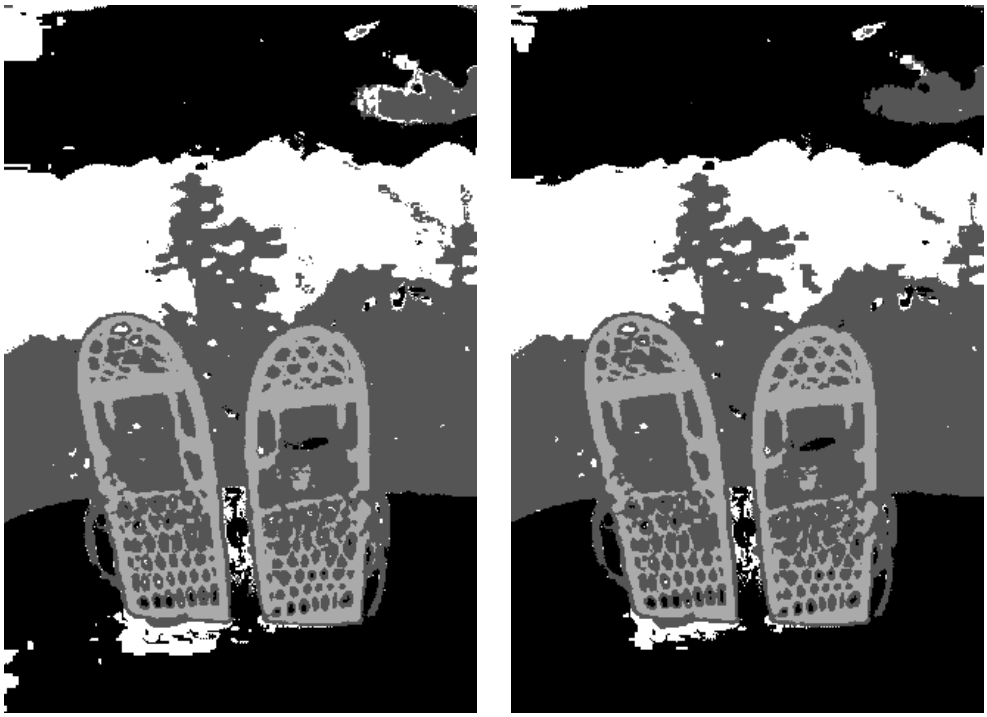
$5 * 5bins$

$10 * 10bins$

$20 * 20bins$

$30 * 30bins$

FIGURE 2.14: Result of image segmentation by binned-EM algorithm of model $[\lambda \mathbf{DAD}^T]$ with different bin size(1).



$40 * 40 bins$ $50 * 50 bins$

FIGURE 2.15: Result of image segmentation by binned-EM algorithm of model $[\lambda \mathbf{DAD}^T]$ with different bin size(2).

From these two figures, we can have some conclusion on the comparison among six results. We can see that when bin size if of 5 bins per dimension, the segmentation result is very general. Some details are missing. For example, we cannot see the cushion and the weave pattern in the sleds. The result of 10 bins per dimension provides more details than 5 bins. The weave pattern in the sleds is clearer. The snow between two sleds is separated from the trees. Snow points on the trees are identified. The result with 20 bins per dimension is close to the one with 10 bins. The separation line between the trees and the mountain is cleaner. More snow spots are detected. The shadow part gets smaller. The results with 30 and 40 bins per dimension are almost the same with each other. The shadows of the sleds in these two figures are almost disappear. The snow ground between two sleds is detected.

From the result, we can see that image segmentation result is more detailed and more correct with smaller bins. But binned-EM algorithm with big bins can also obtain a general image segmentation result which is good enough. Why? As we can see in the Figure 2.12, many pixels which have the same color are overlapped. And the pixels which have similar color are centralized. In this case, grouping the pixels into bins helps largely in reducing the amount of data and the computation time. Some information

about the image segmentation by binned-EM algorithm with different size of bins are shown in the Table 2.9:

Info. / Bin size	L_M	CPUtime(s)	Num. of non-empty bins
5 bins/dimension	-5.43×10^3	30	20
10 bins/dimension	-3.47×10^4	50	67
20 bins/dimension	-6.88×10^4	149	234
30 bins/dimension	-8.67×10^4	203	480
40 bins/dimension	-1.43×10^5	258	610
50 bins/dimension	-1.64×10^5	316	746

TABLE 2.9: Information of the image segmentation by binned-EM algorithm with different size of bins.

The Table 2.9 shows that, bigger bins result in bigger maximum likelihood. We know that there are 154401 pixels in the image. If we use classical EM algorithm, we have 154401 data to deal with. But by applying binned-EM algorithm, the number of bins can be changed according to the bin size. In the Table 2.9, when the size of bins is 5 bins per dimension, we only have 20 non-empty bins to deal with. Reducing 154401 points to 20 bins, helps a lot in computation time saving. Along with the increase of number of bins, of course the computation time increases too.

Considering the computation time, let's look back to the image segmentation quality. We can say that 20 bins per dimension is a good size of bin for image segmentation of this image basing on color.

2.7.2.2 With different models

In this part, we will compare the performance of binned-EM algorithm of parsimonious models on image segmentation. From the Figure 2.12, we cannot tell the distribution of the data corresponds to which model. We suppose that the general models are more suitable for this dataset according to the Figure 2.12. Thus we only choose the eight general models to cluster the pixels of this image. According to the experimental result of the first experiment, binned-EM algorithm with 20 bins per dimension can obtain a good image segmentation result with less computation time. So in this experiment, the bin size is fixed at 20 bins along each dimension. We suppose that there are four main colors in this image. By applying the binned-EM algorithm of eight general models to the pixels clustering, we obtain the image segmentation results which are shown in the Figures 2.16 and 2.17 :

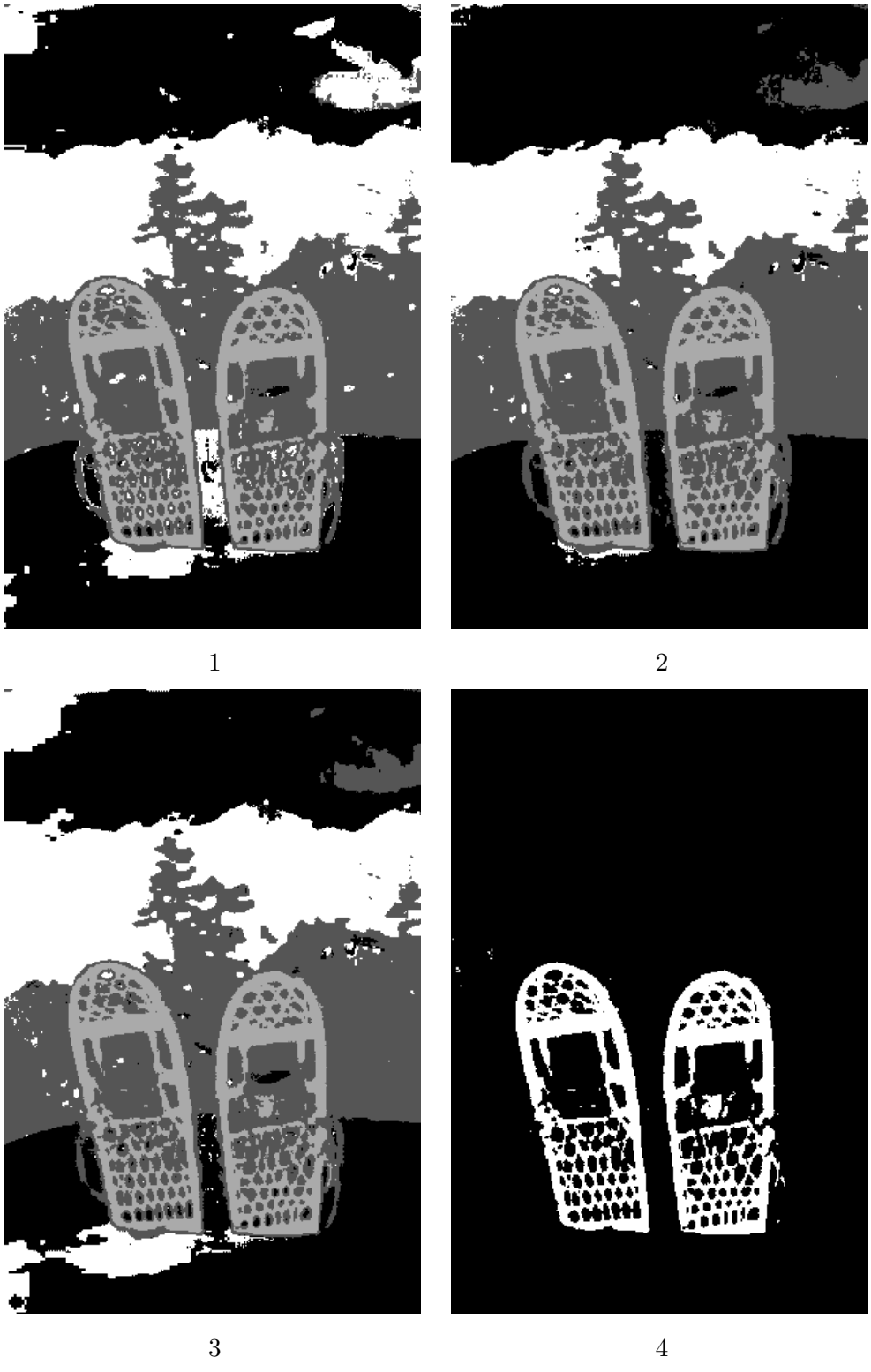
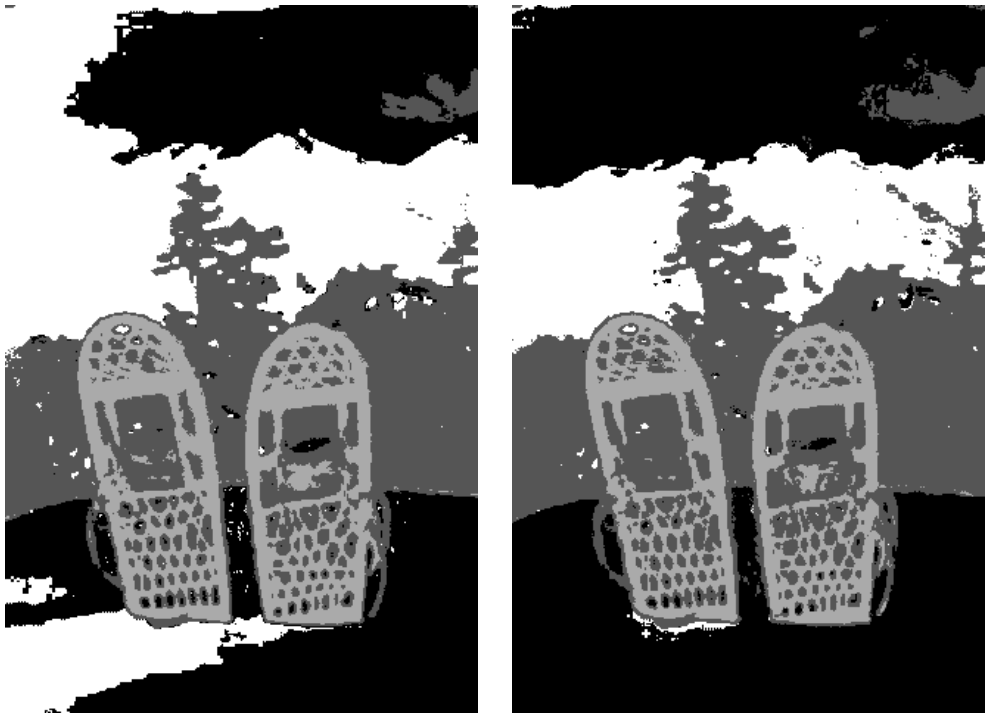


FIGURE 2.16: Result of image segmentation by binned-EM algorithm of four general models: 1. $[\lambda \mathbf{DAD}^T]$, 2. $[\lambda_k \mathbf{DAD}^T]$, 3. $[\lambda \mathbf{DA}_k \mathbf{D}^T]$, 4.$[\lambda_k \mathbf{DA}_k \mathbf{D}^T]$



5

6

7

8

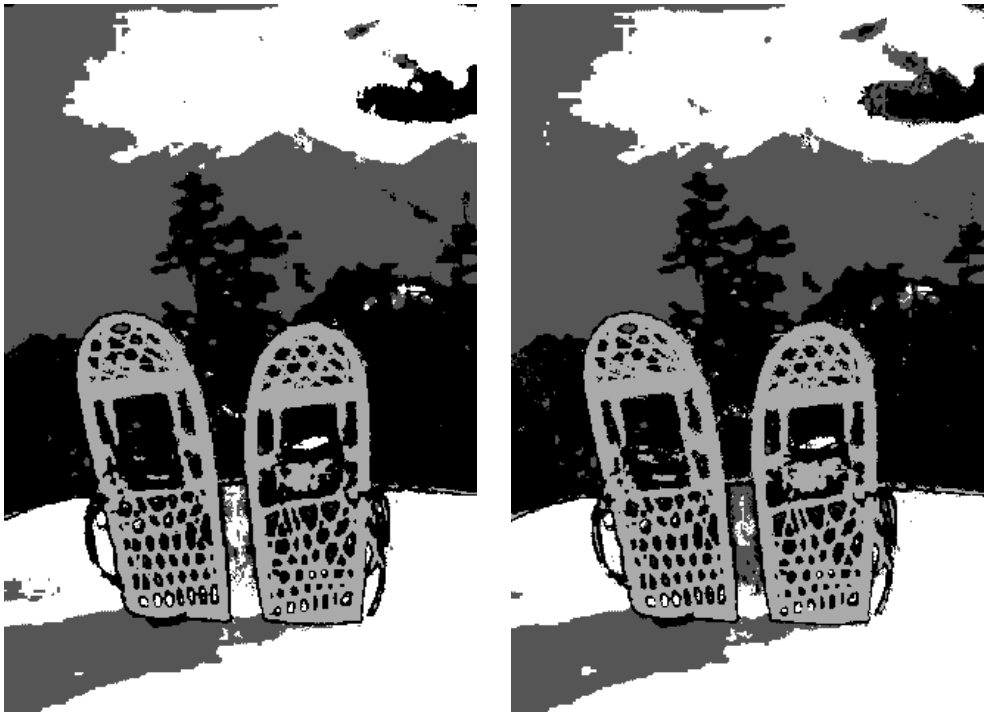
FIGURE 2.17: Result of image segmentation by binned-EM algorithm of four general models: 5. $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$, 6. $[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$, 7. $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$, 8. $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$

From the Figures 2.16 and 2.17, we would say that the model $[\lambda \mathbf{D} \mathbf{A}_k \mathbf{D}^T]$ and the model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ are two best models for the image segmentation of this image. In these

two image segmentation result, there is a clear separation between the sky and the mountain, between the mountain and the trees, between the trees and the snow ground. The shadow is detected. The snow between two sleds is separated from the trees. The weave pattern in the sleds is clear. All the main subjects are clearly shown. The other models give a general idea about the image. Different faults are shown in the result of different models. The model $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$ is the worst one among these eight general models. We can only see two sleds in the result.

2.7.2.3 Comparison with classical EM algorithm and k-means algorithm

Binned-EM algorithm is developed by applying EM algorithm to binned data. Binned-EM algorithm aims to save some computation time by grouping data in bins. An experiment on real data is important to show this point. The goal of this experiment is to show the binned-EM algorithm is faster than the classical EM algorithm when applied to image segmentation.



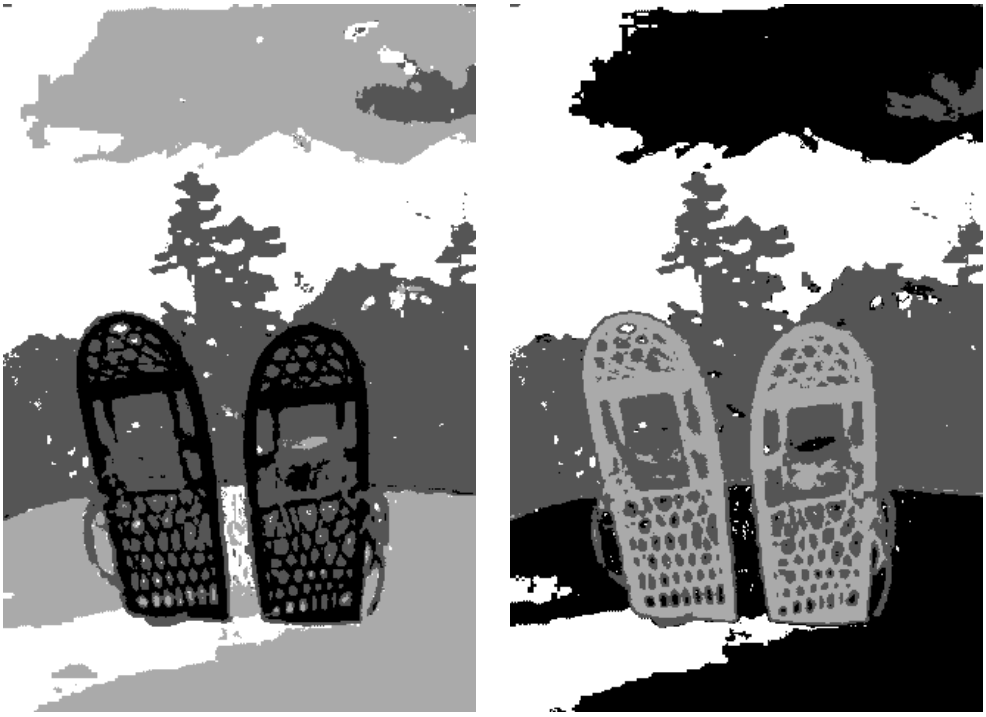
EM $binned - EM$

FIGURE 2.18: comparison between the result by EM algorithm and the result by binned-EM algorithm

From the Figure 2.18, the result of classical EM algorithm and the result of the binned-EM algorithm of the model $[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$ are very similar to each other. We can say that,

binned-EM algorithm doesn't lose obvious precision comparing to the EM algorithm in image segmentation. But, the binned-EM algorithm takes only 119 seconds comparing to 347 seconds for the EM algorithm. This result highlights the advantage of binned-EM algorithm in saving computation time without much loss of precision of clustering, comparing to the EM algorithm.

At the end, we would like to show also the image segmentation result by k-means algorithm. This aims to compare the clustering quality between k-means algorithm and the binned-EM algorithm for image segmentation. The Figure 2.19 shows the comparison between the result by k-means algorithm and the result by binned-EM algorithm of model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$:



$k - means$ $binned - EM$

FIGURE 2.19: comparison between the result by k-means algorithm and the result by binned-EM algorithm

From the Figure 2.19, two results of two algorithms are similar to each other. The result of binned-EM algorithm of model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ is slightly better than the result of k-means algorithm, according to some details. The snow between two sleds is better detected by binned-EM algorithm. The weave pattern in the sleds is clearer in the result of binned-EM algorithm. And the mistake on the right-top corner is smaller in the binned-EM algorithm result. We should notice that, in binned-EM algorithm, all the 154401 pixels are grouped into 234 non-empty bins. Even the k-means algorithm deals with 154401

pixels one by one, binned-EM algorithm provides a higher precision only by dealing with 234 bins.

2.8 Conclusion

This chapter focused on the application of the EM algorithms of parsimonious models to binned data clustering, which led to fourteen binned-EM algorithms. At the beginning, we reviewed the EM algorithm for standard data. The complexity of the EM algorithm was calculated. Then the fourteen parsimonious Gaussian mixture model were presented. Examples of these models were illustrated. After, we developed binned-EM algorithms of fourteen parsimonious Gaussian mixture models. The derivation of binned-EM algorithm of each model was detailed. The complexity of the binned-EM algorithm is calculated and compared with the one of the EM algorithm. We obtained a condition when binned-EM algorithm is faster than the EM algorithm. This result not only shows that the binned-EM algorithm is faster than the EM algorithm when the data amount is big enough, it also helps us to know in which situation binned-EM algorithm should be applied instead of the EM algorithm. The maximum likelihood estimates of model parameters for fourteen parsimonious models were discussed. We compared and studied the performances of these new algorithms by numerical experiments on simulated data and real data. We can conclude the result as follows:

Firstly, the parsimonious Gaussian mixture model which represents the data structure obtains the best result. It shows that even if the parsimonious models are simpler than the most complex model, they are able to well fit different datasets. And because the binned-EM algorithms of parsimonious Gaussian mixture models are more precise and strict in model parameters estimation, they are able to obtain better result than the one with the most complex model. Secondly, by simplifying the parameter estimation, the parsimonious models help in saving computation time. Thirdly, at another side, the computation time depends also on the size of bins. So by enlarging the bin size, we can reduce and control the computation time of clustering. But it risks in losing certain precision. Finally, by running our new algorithms on French department dataset and on image segmentation, it shows that binned-EM algorithms of parsimonious models have a good performance in practice and takes less computation time than the classic EM algorithm.

In the next chapter, we will develop the bin-EM-CEM algorithms of fourteen parsimonious Gaussian mixture models.

Chapter 3

Parsimonious Gaussian mixture models for binned data clustering and the corresponding bin-EM-CEM algorithms

3.1 Introduction

Another common model-based clustering approach is the classification approach, which was proposed by Symons [35]. It aims to maximize the complete likelihood by the Classification Estimation Maximization (CEM) algorithm over the mixture parameters and over the origin labels indicating the component that each observation comes from.

The CEM algorithm is regarded as the classification version of the EM algorithm while maximizing the complete likelihood. In the CEM algorithm, a classification step is inserted between the E-step and the M-step of the EM algorithm using a maximum a posteriori (MAP) principle. Each step of the CEM algorithm will be detailed in this chapter. Thanks to this classification step, the CEM algorithm is faster than the EM algorithm. But this advantage in computation time is not enough when facing to data of big quantity. The execution time of the CEM algorithm still increases significantly along with the data size. To resolve this problem, a classification EM algorithm for binned data (bin-EM-CEM) was developed by Samé et al. [14].

At another side, the fourteen parsimonious Gaussian mixture models can contribute to offer a good solution for this computation time problem. These models are more simplified than the most general model. By using these models, the clustering process can

be accelerated. The CEM algorithms of fourteen parsimonious models were developed by Celeux and Govaert [15]. The result turned out to be encouraging. The parsimonious models adapt to different datasets. And until now, the bin-EM-CEM algorithms of parsimonious models are missing.

Thus, the objective of this chapter is to introduce and develop the CEM algorithms for binned data clustering (bin-EM-CEM algorithms)of fourteen parsimonious Gaussian mixture models.

This chapter is organized as follows:

In the Section 3.2, we will review the classification approach for standard data and its corresponding algorithm: the CEM algorithm. The computation complexity of the CEM algorithm will be calculated in this part. Then in the Section 3.3, the derivation of the bin-EM-CEM algorithm will be detailed. The computation complexity of the bin-EM-CEM algorithm will be calculated and compared with the one of the CEM algorithm. The Section 3.4 will present the estimation of variance matrices of fourteen parsimonious models, which varies according to different models. In the Section 3.5, we will show two experiments on simulated data. The first experiment is to compare the performances of bin-EM-CEM algorithms of fourteen models on different simulated data. The second experiment aims to study how the bin-EM-CEM algorithm behaves differently when the size of bins changes. The Section 3.6 will show two experiments on real datasets. The first experiment is French city clustering and in the second experiment the bin-EM-CEM algorithms is applied to the image segmentation. The last Section 3.7 will conclude this chapter and open the discussion of the next chapter.

3.2 Classification approach for standard data

In the classification maximum likelihood (CML) approach, the indicators z_i identifying the origin of $\boldsymbol{x}_i$ are considered as missing parameters. Different from the mixture maximum likelihood (ML) approach, CML approach maximizes the likelihood basing on the mixture model parameters and the data labels.

In standard data framework, the classification approach assumed that the mixing proportions are equal in the mixture model. In 1981, Symons [35] has proposed a general classification approach which imposes no restriction on the mixing proportions. Theoretically, this unrestricted approach is expected to outperform the restricted one. But some numerical experiments in Symons [35] show that the unrestricted classification approach has a tendency to overstate the size of the larger clusters. This tendency is also proved in the paper of Bryant [57] on the subject of clustering of large sample. Bryant showed

that asymptotically the unrestricted classification approach did not classify at all when the components are badly separated or when there is huge difference among the mixing proportions. Celeux and Govaert [37] has compared the classification approaches with equal mixing proportions and with free mixing proportions. The result showed that the restricted classification approach is preferable to the unrestricted one.

3.2.1 The likelihood

In the classification maximum likelihood approach, the complete data is composed by $(\mathbf{x}, \boldsymbol{z}) = \{(\boldsymbol{x}_1, z_1), \ldots, (\boldsymbol{x}_n, z_n)\}$ where the unknown parameter z_i indicates the origin component where the individual $\boldsymbol{x}_i$ comes from. $z_i = k$ if $\boldsymbol{x}_i$ comes from the kth component. In the restricted version of CML approach, the proportions π_k's are assumed to be equal. Thus we have the form of restricted CML approach as follows:

$$L_{CR} = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in P_k} \log\Big(\big(f(\boldsymbol{x}_i, \theta_k)\big)\Big)$$

where $P = (P_1, \ldots, P_K)$ is a partition of the sample $\boldsymbol{x}_i, \ldots, \boldsymbol{x}_n$, and $P_k = \{\boldsymbol{x}_i | z_i = k\}$. $\theta_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ are the parameters of the Gaussian distribution function f, where $\boldsymbol{\mu}_k$'s are the mean and $\boldsymbol{\Sigma}_k$'s are the variance matrices.

The unrestricted CML approach with free proportions π_k's is presented as follows:

$$L_C = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in P_k} \log\big(\pi_k f(\boldsymbol{x}_i, \theta_k)\big)$$

It is proved [6] that maximizing the restricted CML criterion is to minimizing the within-group scatter matrix $|\mathbf{W}|$:

$$\mathbf{W} = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in P_k} (\boldsymbol{x}_i - \overline{\boldsymbol{x}}_k)(\boldsymbol{x}_i - \overline{\boldsymbol{x}}_k)'$$

where

$$\overline{\boldsymbol{x}}_k = \frac{1}{\#P_k} \sum_{\boldsymbol{x}_i \in P_k} \boldsymbol{x}_i$$

It can also be proved that maximizing the unrestricted CML criterion is to minimizing:

$$n \log(|\mathbf{W}|) - 2 \sum_{k=1}^{K} \#P_k \log\{\#P_k\}$$

3.2.2 The CEM algorithm

To obtain the model parameters which optimizes the CML criteria, we can apply the CEM algorithm. In this part, we give out an example of the unrestricted CML criterion. Beginning with an initial partition, CEM algorithm computes the following three steps iteratively:

- E step. Compute the posterior probabilities $t_{ik}^{(q)}$ $(1 \leq i \leq n; 1 \leq k \leq K)$:

$$t_{ik}^{(q)} = \frac{\pi_k^{(q)} f_k(\mathbf{x}; \theta_k^{(q)})}{\sum_l \pi_l^{(q)} f_l(\mathbf{x}; \theta_l^{(q)})}$$

  for the unrestricted criterion.

- C step. Assign each $\boldsymbol{x}_i$ to the cluster which provides the maximum posterior probability $t_k(\boldsymbol{x}_i)$. This equals obtaining $\boldsymbol{z}_i^{(q)}$ which indicates the mixture origin of each $\boldsymbol{x}_i$:

$$z_i^{(q)} = \arg\max_k t_{ik}^{(q)}$$

- M-step (Maximization): Find the parameter $\mathbf{\Phi}^{(q+1)}$ that maximizes the expectation.

We can obtain the mixture model parameters:

$$\pi_k^{(q+1)} = \frac{1}{n} \sum_{i=1}^{n} z_{ik}^{(q)}$$

$$\boldsymbol{\mu}_k^{(q+1)} = \frac{1}{\sum_{i=1}^{n} z_{ik}^{(q)}} \sum_{i=1}^{n} z_{ik}^{(q)} \boldsymbol{x}_i$$

The result of variance matrix $\mathbf{\Sigma}_k^{(q+1)}$ differs according to the chosen parsimonious model. We will detail the variance matrix estimate later. For example, for the CEM algorithm of model $[\lambda \mathbf{DAD}^T]$, we have:

$$\mathbf{\Sigma}_k^{(q+1)} = \frac{\sum_{k=1}^{K} \sum_{i=1}^{n} z_{ik}^{(q)} (\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(q+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(q+1)})^T}{n}$$

3.2.3 The complexity of CEM algorithm

Complexity calculation of CEM algorithm helps us to study the efficiency of the algorithm. Instead of running many experiments under different conditions, complexity

calculation also leads to easier comparison of the time consumption with other algorithms.

Celeux and Govaert [15] have developed CEM algorithms of fourteen parsimonious Gaussian mixture models. Since the calculations of these algorithms are similar, in this part we only study the complexity of one general model $[\lambda \mathbf{DAD}^T]$. The CEM algorithm of model $[\lambda \mathbf{DAD}^T]$ is presented in the Algorithm 3:

Algorithm 3 CEM algorithm

$q \leftarrow 0$

Initialize $\boldsymbol{\pi}^{(0)}$ and $\theta^{(0)} = \{\boldsymbol{\mu}^{(0)}, \boldsymbol{\Sigma}^{(0)}\}$.

repeat

  for $i = 1 : n$ **do**

    $t_i^{(q)} \leftarrow \sum_{k=1}^{K} \pi_k^{(q)} f_k(\boldsymbol{x}_i; \theta_k^{(q)})$

    for $k = 1 : K$ **do**

      $t_{ik}^{(q)} \leftarrow \frac{\pi_k^{(q)} f_k(\boldsymbol{x}_i; \theta_k^{(q)})}{t_i^{(q)}}$

    end for

  end for

  for $i = 1 : n$ **do**

    $z_i^{(q)} \leftarrow \arg\max_k t_{ik}^{(q)}$

  end for

  for $k = 1 : K$ **do**

    $\pi_k^{(q+1)} \leftarrow \frac{1}{n} \sum_{i=1}^{n} z_{ik}^{(q)}$

    $\boldsymbol{\mu}_k^{(q+1)} \leftarrow \frac{1}{\sum_{i=1}^{n} z_{ik}^{(q)}} \sum_{i=1}^{n} z_{ik}^{(q)} \boldsymbol{x}_i$

    $\boldsymbol{\Sigma}_k^{(q+1)} \leftarrow \frac{\sum_{k=1}^{K} \sum_{i=1}^{n} z_{ik}^{(q)} (\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(q+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{(q+1)})^T}{n}$

  end for

  $q \leftarrow q + 1$

until $\frac{L^{(q+1)}(\boldsymbol{x},\boldsymbol{z}) - L^{(q)}(\boldsymbol{x},\boldsymbol{z})}{L^{(q)}(\boldsymbol{x},\boldsymbol{z})} < \varepsilon$

$\widehat{\boldsymbol{z}} \leftarrow \boldsymbol{z}^{(q)}$, $\widehat{\boldsymbol{\pi}} \leftarrow \boldsymbol{\pi}^{(q+1)}$, $\widehat{\theta} \leftarrow \theta^{(q+1)}$

Same as the EM algorithm, the CEM algorithm repeats several calculations until one condition is satisfied: $\frac{L^{(q+1)}(\boldsymbol{x},\boldsymbol{z}) - L^{(q)}(\boldsymbol{x},\boldsymbol{z})}{L^{(q)}(\boldsymbol{x},\boldsymbol{z})} < \varepsilon$. It is hard to define the moment when the CEM algorithm stops. It depends on the data distribution, the initiation and the threshold ε. To simplify the complexity calculation, let's suppose that the CEM algorithm stops in N iterates. Each iterate can be seen as four parts:

- Calculate n times $t_i^{(q)}$.
- Calculate $n * K$ times $t_{ik}^{(q)}$.

- Calculate n times $z_i^{(q)}$
- Calculate K times $\pi_k^{(q+1)}$, $\boldsymbol{\mu}_k^{(q+1)}$ and $\boldsymbol{\Sigma}_k^{(q+1)}$.

The first part and the second part is exactly the same as in the EM algorithm. Details of the complexity calculation of these two parts can be seen in the Chapter 2. According to the result, the complexity of calculation of $t_i^{(q)}$ is $T = O(d^3 + 4n_1 + n_2^2)$. Since $\pi_k^{(q)} f_k(\boldsymbol{x}_i; \theta_k^{(q)})$ and $t_i^{(q)}$ are already known, the complexity of computation of $t_{ik}^{(q)}$ is only $O(1)$. To obtain $z_i^{(q)}$, is to find out the maximum of $t_{ik}^{(q)}$. The worst case is that the maximum locates at the last position. To check all the data and then assign the origin to each $\boldsymbol{x}_i$, it needs $O(2K)$ complexity.

We suppose that there are n_k data belong to the cluster k. When obtaining the parameters $\pi_k^{(q+1)}$, we need to calculate only $n_k - 1$ times the addition of $z_{ik}^{(q)}$ instead of $n - 1$ times. Because the rest $z_{ik}^{(q)}$ equal zero. The same situation can be applied to the calculation of $\boldsymbol{\mu}_k^{(q+1)}$ and $\boldsymbol{\Sigma}_k^{(q+1)}$. That is also the reason why the CEM algorithm is faster then the EM algorithm.

The complexities of each part of the CEM algorithm are summarized in the Table 3.1:

Parameter	Times	Complexity
$t_i^{(q)}$	n	$O(d^3K + 4n_1K + n_2^2K)$
$t_{ik}^{(q)}$	$n * K$	$O(1)$
$z_i^{(q)}$	n	$O(2K)$
$\pi_k^{(q+1)}$	K	$O(n_k)$
$\boldsymbol{\mu}_k^{(q+1)}$	K	$O(n_k d + 2n_k)$
$\boldsymbol{\Sigma}_k^{(q+1)}$	K	$O(3dn_kK)$

TABLE 3.1: Decomposition of complexity of the CEM algorithm.

From the Table 3.1, we can conclude that the complexity of the CEM algorithm of model $[\lambda \mathbf{DAD}^T]$ is approximately $O(d^3KnN + 4n_1KnN + n_2^2KnN + 3dK^2n_kN + 3KnN + dKn_kN + 3Kn_kK)$. According to the definition of the Big-O notation, the complexity of the CEM algorithm can also be noted as $O(d^3KnN + 4n_1KnN + n_2^2KnN + 3dK^2n_kN)$. Comparing to the EM algorithm, CEM algorithm has smaller complexity. This result corresponds to the reality that the CEM algorithm is faster than the EM algorithm.

3.3 The bin-EM-CEM algorithm

3.3.1 The likelihood

We assume that $\mathbf{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is an independent sample issued from a K-component mixture distribution defined on $\mathbb{R}^d$:

$$f(\mathbf{x}; \boldsymbol{\Phi}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}; \theta_k)$$

with $\boldsymbol{\Phi} = (\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K)$, where π_k $(k = 1, \ldots, K)$ denote the mixing proportions of the mixtures ($0 < \pi_k < 1$ and $\sum_{k=1}^{K} \pi_k = 1$), and $\theta_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ $(k = 1, \ldots, K)$ are the parameters of Gaussian distribution functions f_k of components: mean vectors $\boldsymbol{\mu}_k$ and variance matrices $\boldsymbol{\Sigma}_k$. Vector $\mathbf{z} = (z_1, \ldots, z_n)$ is the class label of $\mathbf{x}$, where $z_i = 1, \ldots, K$ for $i = 1, \ldots, n$. $z_i = k$ when $\boldsymbol{x}_i$ comes from the kth component.

The whole sample space $\mathbb{R}^d$ is divided into v bins with a partition $(\boldsymbol{\mathcal{H}}_1, \ldots, \boldsymbol{\mathcal{H}}_v)$ and we assume that the only observed information is a set of frequencies n_r $(r = 1, \ldots, v)$ where each frequency n_r indicates the number of $\boldsymbol{x}_i$ belonging to the bin $\boldsymbol{\mathcal{H}}_r$. The set of frequencies is denoted by vector $\boldsymbol{a} = (n_1, \ldots, n_v)$, with $\sum_{r=1}^{v} n_r = n$.

The probability that $\boldsymbol{x}$ belongs to bin $\boldsymbol{\mathcal{H}}_r$ is denoted by:

$$p_r(\boldsymbol{\Phi}) = P(\boldsymbol{x} \in \boldsymbol{\mathcal{H}}_r | \boldsymbol{\Phi}) = \sum_{k=1}^{K} \pi_k \int_{\boldsymbol{\mathcal{H}}_r} f_k\Big(\boldsymbol{x}; \theta_k\Big) \mathrm{d}\boldsymbol{x}$$

and the probability that $\boldsymbol{x}$ belonging to the bin $\boldsymbol{\mathcal{H}}_r$ comes from component k of the mixture is denoted by:

$$p_{k/r}(\boldsymbol{\Phi}) \quad = \quad \frac{\pi_k \int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x}; \theta_k) \mathrm{d}\boldsymbol{x}}{p_r(\boldsymbol{\Phi})}$$

The only observed information vector $\boldsymbol{a}$ follows a multinomial distribution

$$p(\boldsymbol{a}, \boldsymbol{\Phi}) = c \prod_{r=1}^{v} (p_r(\boldsymbol{\Phi}))^{n_r}$$

where $c = n! / \prod_{r=1}^{v} n_r!$.

According to the space division with a partition $(\boldsymbol{\mathcal{H}}_1, \ldots, \boldsymbol{\mathcal{H}}_v)$, the complete information of the data can be denoted as $(\mathbf{x}, \mathbf{z}) = ((\boldsymbol{x}_{r1}, z_{r1}), \ldots, (\boldsymbol{x}_{rn_r}, z_{rn_r}))$ for $r = 1, \ldots, v$, where $\boldsymbol{x}_{rs}$ points to the sth data in bin $\boldsymbol{\mathcal{H}}_r$. The density of each point $\boldsymbol{x}_{rs}$ is

$\pi_{z_{rs}} f_{z_{rs}}(\boldsymbol{x}_{rs}; \theta_{z_{rs}})/p_r(\boldsymbol{\Phi})$ and the complete data probability function is

$$p(\boldsymbol{a}, \mathbf{x}, \boldsymbol{z}; \boldsymbol{\Phi}) = c \prod_{r=1}^{v} \prod_{s=1}^{n_r} \pi_{z_{rs}} f_{z_{rs}}(\boldsymbol{x}_{rs}; \theta_{z_{rs}})$$

The complete log-likelihood is:

$$L(\boldsymbol{\Phi}; \boldsymbol{a}, \mathbf{x}, \boldsymbol{z}) = \sum_{k=1}^{K} \sum_{r=1}^{v} \sum_{s=1}^{n_r} z_{k_{rs}} \log\left(\pi f_k(\boldsymbol{x}_{rs}; \theta_k)\right) + \log(c)$$

where $z_{k_{rs}} = 1$ if $z_{rs} = k$ and 0 otherwise.

Since there is no information about the exact location of the data within each bin, we assume that all the data comes from the same component within each bin. So $p_r(\boldsymbol{\Phi})$ can be expressed as:

$$p_r(\boldsymbol{\Phi}) = \pi_{z^r} \int_{\boldsymbol{\mathcal{H}}_r} f_{z^r}\Big(\boldsymbol{x}; \theta_{z^r}\Big) \mathrm{d}\boldsymbol{x}$$

Then we have the joint density function as follows:

$$p(\boldsymbol{a}, \boldsymbol{z}; \boldsymbol{\Phi}) = c \prod_{r=1}^{v} (\pi_{z^r} \int_{\boldsymbol{\mathcal{H}}_r} f_{z^r}(\boldsymbol{x}; \theta_{z^r}) \mathrm{d}\boldsymbol{x})^{n_r}$$

and the complete log-likelihood can be expressed as:

$$L(\boldsymbol{\Phi}; \boldsymbol{a}, \boldsymbol{z}) = \sum_{r=1}^{v} n_r \log\left(\pi_{z_r} \int_{\boldsymbol{\mathcal{H}}_r} f_{z^r}(\boldsymbol{x}; \theta_{z^r})\right) + \log(c)$$

3.3.2 The E-step, C-step, and M-step

The bin-EM-CEM algorithm aims to maximize $L(\boldsymbol{\Phi}; \boldsymbol{a}, \boldsymbol{z})$ and starts from a random initialization $\boldsymbol{\Phi}^{(0)}$. It follows two steps iteratively until it convergence.

Step 1 (Expectation and Classification): calculate

$$\boldsymbol{z}^{(q+1)} = \arg\max_{\boldsymbol{z}} L(\boldsymbol{\Phi}^{(q)}; \boldsymbol{a}, \boldsymbol{z})$$

we have

$$\begin{aligned} z_r^{(q+1)} &= \arg\max_{1 \le k \le K} (\log(\pi_k^{(q)} \int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x})) \\ &= \arg\max_{1 \le k \le K} P_{k/r}(\boldsymbol{\Phi}^{(q)}) \end{aligned} \tag{3.1}$$

From Equation (3.1), step 1 can be divided into step E (Expectation) and step C (Classification) [58]:

At the E step, we calculate $p_{k/r}^{(q)}$ for all the k, r;

At the C step, we obtain the partition $\boldsymbol{z}^{(q+1)}$ by maximizing $p_{k/r}^{(q)}$: $z_r^{(q+1)} = \arg\max_k P_{k/r}^{(q)}$, which means for each r, we replace the biggest $p_{k/r}^{(q)}$ by 1, and 0 for the others.

Step 2 (Maximization): calculate

$$\boldsymbol{\Phi}^{(q+1)} = \arg\max_{\boldsymbol{\Phi}} L(\boldsymbol{\Phi}; \boldsymbol{a}, \boldsymbol{z}^{(q+1)})$$

Because it is not easy to maximize $L(\boldsymbol{\Phi}; \boldsymbol{a}, \boldsymbol{z})$ directly, thus we apply an internal EM algorithm to obtain this maximization. As the EM algorithm, we maximize the expectation of the complete log-likelihood instead of log-likelihood:

$$\begin{aligned} Q(\boldsymbol{\Phi}, \boldsymbol{\Phi}^{(q)}) &= E(L(\boldsymbol{\Phi}; \boldsymbol{a}, \boldsymbol{x}, \boldsymbol{z})|\boldsymbol{a}, \boldsymbol{z}^{(q+1)}; \boldsymbol{\Phi}^{(q)}) \\ = E(\sum_{k=1}^{K}\sum_{r=1}^{v}\sum_{s=1}^{n_r} z_{krs} \log\left(\pi_k f_k(\boldsymbol{x}_{rs}; \theta_k)\right) &+ \log(c)|\boldsymbol{z}^{(q+1)}; \boldsymbol{\Phi}^{(q)}) \end{aligned}$$

In the cycle of the inner EM algorithm, let's denote:

$$Q(\boldsymbol{\Phi}, \boldsymbol{\Phi}^*) \quad = \quad E(L(\boldsymbol{\Phi}; \boldsymbol{a}, \boldsymbol{x}, \boldsymbol{z})|\boldsymbol{a}, \boldsymbol{z}^{(q+1)}; \boldsymbol{\Phi}^*)$$

where

$$\boldsymbol{\Phi}^* = \boldsymbol{\Phi}^{(q)}$$

Then we have:

$$\begin{aligned} Q(\boldsymbol{\Phi}, \boldsymbol{\Phi}^*) \quad = \quad & \sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r (\log(\pi_k) + \frac{1}{\int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^*) dx} \\ & \cdot \int_{\mathcal{H}_r} \log(f_k(\boldsymbol{x}; \theta_k^*)) f_k(\boldsymbol{x}; \theta^*) \mathrm{d}\boldsymbol{x}) + \log(c) \end{aligned} \tag{3.2}$$

Maximizing Equation (3.2) equals maximizing:

$$\begin{aligned} \mathbf{A} \quad = \quad & \sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \frac{1}{\int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^*) \mathrm{d}\boldsymbol{x}} \int_{\mathcal{H}_r} \log(f_k(\boldsymbol{x}; \theta_k^*)) f_k(\boldsymbol{x}; \theta_k^*) \mathrm{d}\boldsymbol{x} \\ = \quad & \sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \frac{1}{\int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^*) \mathrm{d}\boldsymbol{x}} \int_{\mathcal{H}_r} (-\log(2\pi)^{n/2} - \log|\boldsymbol{\Sigma_k}|^{1/2} \\ & -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)' \boldsymbol{\Sigma_k}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k)) f_k(\boldsymbol{x}; \theta_k^*) \mathrm{d}\boldsymbol{x} \end{aligned}$$

And finally it leads to the minimization of

$$\begin{aligned} B &= \sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \frac{1}{\int_{\mathcal{H}_r} f_k(\boldsymbol{x};\theta_k^*)dx} \int_{\mathcal{H}_r} (\log|\boldsymbol{\Sigma}_k| \\ &\quad +(\boldsymbol{x}-\boldsymbol{\mu}_k)'\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k))f_k(\boldsymbol{x};\theta_k^*)\mathrm{d}\boldsymbol{x} \\ &= \sum_{k=1}^{K} tr(\boldsymbol{\Sigma}_k^{-1}\mathbf{G}_k^{**}) + \sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log(\boldsymbol{\Sigma}_k) \end{aligned} \tag{3.3}$$

where

$$\mathbf{G}_k^{**} = \sum_{r=1}^{v} \frac{z_{kr}^{(q)} n_r}{p_{r/k}^*} \int_{\mathcal{H}_r} (\boldsymbol{x}-\boldsymbol{\mu}_k^{**})'(\boldsymbol{x}-\boldsymbol{\mu}_k^{**}) f_k(\boldsymbol{x};\theta_k^*)\mathrm{d}\boldsymbol{x}$$

and

$$p_{r/k}^* = \int_{\mathcal{H}_r} f_k(\boldsymbol{x};\theta_k^*)dx$$

We get

$$\pi_k^{**} = \frac{\sum_{r=1}^{v} n_r z_{rk}^{(q)}}{n}$$

and

$$\boldsymbol{\mu}_k^{**} = \frac{\sum_{r=1}^{v} \frac{n_r z_{rk}^{(q)}}{p_{r/k}^*} \int_{\mathcal{H}_r} f_k(\boldsymbol{x};\theta_k^*)\mathrm{d}\boldsymbol{x}}{\sum_{v=1}^{v} n_r z_{rk}^{(q)}}$$

where

$$\boldsymbol{\Phi}^{**} = \boldsymbol{\Phi}^{(q+1)}$$

The result of $\boldsymbol{\Sigma}_k^{**}$ depends on the parsimonious models.

3.3.3 The complexity of bin-EM-CEM algorithm

In order to compare with the complexity of the CEM algorithm, we will calculate the computation complexity of bin-EM-CEM algorithm in this part. The bin-EM-CEM algorithm of model $[\lambda\mathbf{DAD}^T]$ is presented in the Algorithm 4:

Algorithm 4 Bin-EM-CEM algorithm

$q \leftarrow 0$

Initialize $\boldsymbol{\pi}^{(0)}$ and $\theta^{(0)} = \{\boldsymbol{\mu}^{(0)}, \boldsymbol{\Sigma}^{(0)}\}$.

repeat

for $r = 1 : v$ **do**

$p_r^{(q)} \leftarrow \sum_{k=1}^{K} \pi_k^{(q)} \int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}$

for $k = 1 : K$ **do**

$p_{k/r}^{(q)} \leftarrow \frac{\pi_k^{(q)} \int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}}{p_r^{(q)}}$

end for

end for

for $r = 1 : v$ **do**

$z_r^{(q)} \leftarrow \arg\max_k p_{k/r}^{(q)}$

end for

$\boldsymbol{\pi}^* = \boldsymbol{\pi}^{(q)}, \theta^* = \theta^{(q)}$

repeat

for $r = 1 : v$ **do**

$p_{r/k}^* \leftarrow \int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \theta_k^{(q)}) \mathrm{d}\boldsymbol{x}$

end for

for $k = 1 : K$ **do**

$\pi_k^{**} \leftarrow \frac{\sum_{r=1}^{v} n_r z_{kr}^{(q)}}{n}$

$\boldsymbol{\mu}_k^{**} \leftarrow \frac{\sum_{r=1}^{v} \frac{n_r z_{kr}}{p_{r/k}^*} \int_{\mathcal{H}_r} \boldsymbol{x} f_k(\boldsymbol{x}; \theta_k^*) \mathrm{d}\boldsymbol{x}}{\sum_{r=1}^{v} n_r z_{kr}^{(q)}}$

$\boldsymbol{\Sigma}_k^{**} \leftarrow \frac{\sum_{k=1}^{K} \sum_{r=1}^{v} \frac{n_r z_{kr}^{(q)}}{p_{r/k}^*} \int_{\mathcal{H}_r} (\boldsymbol{x} - \boldsymbol{\mu}_k^{**})(\boldsymbol{x} - \boldsymbol{\mu}_k^{**})^T f_k(\boldsymbol{x}; \theta_k^*) \mathrm{d}\boldsymbol{x}}{\sum_{k=1}^{K} \sum_{r=1}^{v} z_{kr}^{(q)} n_r}$

$* \leftarrow **$

end for

until $\frac{L^{**}(\boldsymbol{a}, \boldsymbol{z}^{(q)}) - L^*(\boldsymbol{a}, \boldsymbol{z}^{(q)})}{L^*(\boldsymbol{a}, \boldsymbol{z}^{(q)})} < \varepsilon$

$\boldsymbol{\pi}^{(q+1)} = \boldsymbol{\pi}^{**}, \theta^{(q+1)} = \theta^{**}$

$q \leftarrow q + 1$

until $\boldsymbol{z}^{(q)} = \boldsymbol{z}^{(q-1)}$

$\widehat{\boldsymbol{z}} \leftarrow \boldsymbol{z}^{(q)}$, $\widehat{\boldsymbol{\pi}} \leftarrow \boldsymbol{\pi}^{(q+1)}$, $\widehat{\theta} \leftarrow \theta^{(q+1)}$

From the Algorithm 4, similar as the CEM algorithm, bin-EM-CEM algorithm executes a big loop until the result doesn't change anymore. This iterate can take a very long time. It is difficult to know when exactly it will stop. To be able to compare with the EM, CEM, binned-EM algorithms, we also define that the bin-EM-CEM algorithm stops in N iterates.

Each iterate can be considered as five small parts:

- Calculate v times $p_r^{(q)}$.
- Calculate $v * K$ times $p_{k/r}^{(q)}$.
- Calculate v times $z_r^{(q)}$.
- Calculate once $\boldsymbol{\pi}^*$ and θ^*.
- Execute an inner EM algorithm. We suppose that the inner EM algorithm stops in N_2 iterates. For each iterate, we execute:
  - Calculate v times $p_{r/k}^*$.
  - Calculate K times π_k^{**}, $\boldsymbol{\mu}_k^{**}$ and $\boldsymbol{\Sigma}_k^{**}$.

From the sub-Section 2.4.3, we obtained the complexity to calculate $p_r^{(q)}$ is $O((l_1+1)(l_2+1)\cdots(l_d+1)(d^3K+4n_1K+n_2^2K))$. Since the $\pi_k^{(q)}\int_{\mathcal{H}_r} f_k(\boldsymbol{x};\theta_k^{(q)})\mathrm{d}\boldsymbol{x}$ and $p_r^{(q)}$ are already known, to obtain $p_{k/r}^{(q)}$ only takes complexity of $O(1)$. To get $z_r^{(q)}$ is to scan over $p_{k/r}^{(q)}$ and choose the maximum. Then assign 1 to $z_{k_m r}^{(q)}$ if $p_{k_m/r}^{(q)}$ is the maximum. Assign 0 to the rest. This process takes $O(2k)$ complexity. Assigning $\boldsymbol{\pi}^{(q)}$ to $\boldsymbol{\pi}^*$ and $\theta^{(q)}$ to θ^* needs $O(K+dK+d^2K)$ complexity.

We suppose that the inner EM algorithm stops in N_2 iterates. The calculation of the complexity of obtaining π_k^{**}, $\boldsymbol{\mu}_k^{**}$ and $\boldsymbol{\Sigma}_k^{**}$ is very similar as the corresponding one of binned-EM algorithm. Details can refer in the sub-Section 2.4.3.

The complexity of each part of bin-EM-CEM algorithm for each iterate is listed in the Table 3.2:

Parameter	Times	Complexity
$p_r^{(q)}$	v	$O((l_1+1)(l_2+1)\cdots(l_d+1)(d^3K+4n_1K+n_2^2K))$
$p_{k/r}^{(q)}$	$v*K$	$O(1)$
$z_r^{(q)}$	v	$O(2k)$
$\boldsymbol{\pi}^*$, θ^*	1	$O(K+dK+d^2K)$
$p_{r/k}^*$	$N_2 v$	$O(1)$
$\pi_k^{(q+1)}$	$N_2 K$	$O(2v_k)$
$\boldsymbol{\mu}_k^{(q+1)}$	$N_2 K$	$O((l_1+1)(l_2+1)\cdots(l_d+1)v_k)$
$\boldsymbol{\Sigma}_k^{(q+1)}$	$N_2 K$	$O((l_1+1)(l_2+1)\cdots(l_d+1)3dv_kK)$

TABLE 3.2: Decomposition of complexity of the bin-EM-CEM algorithm.

From the Table 3.2, the complexity of bin-EM-CEM algorithm is approximately $O((l_1+1)(l_2+1)\cdots(l_d+1)(d^3KvN+4n_1KvN+n_2^2KvN+3dv_kK^2NN_2))$. Comparing to the

CEM algorithm, if n satisfies the condition:

$$n > (l_1 + 1)(l_2 + 1) \cdots (l_d + 1)v \tag{3.4}$$

then bin-EM-CEM algorithm is faster than the CEM algorithm.

3.4 Bin-EM-CEM algorithms of parsimonious models

3.4.1 The general models

Model $[\lambda \mathbf{DAD}^T]$. For this most common model, maximizing equation (3.2) equals the minimization of

$$M_1(\mathbf{\Sigma}) = \sum_{k=1}^{K} tr(\mathbf{\Sigma}^{-1}\mathbf{G}_k^{**}) + \sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log(\mathbf{\Sigma})$$

variance matrix $\mathbf{\Sigma}$ is estimated by

$$\mathbf{\Sigma}^{**} = \frac{\sum_{k=1}^{K} \mathbf{G}_k^{**}}{\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

Model $[\lambda_k \mathbf{DAD}^T]$. In this situation, we put $\mathbf{\Sigma}_k = \lambda_k \mathbf{C}$ with $\mathbf{C} = \mathbf{DAD}^T$. Maximizing equation (3.2) equals to the minimization of

$$M_2(\lambda_k, \mathbf{C}) = \sum_{k=1}^{K} \frac{1}{\lambda_k} tr(\mathbf{C}^{-1}\mathbf{G}_k^{**}) + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log|\lambda_k|$$

To find λ_k^{**} and $\mathbf{C}$, an iteration has to be performed:

- Keep $\mathbf{C}$ fixed, the λ_k^{**} are

$$\lambda_k^{**} = \frac{tr(\mathbf{G}_k^{**}\mathbf{C}^{-1})}{d\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

- keep λ_k^{**} fixed, the matrix $\mathbf{C}$ is

$$\mathbf{C}^{**} = \frac{\sum_{k=1}^{K} \frac{1}{\lambda_k}\mathbf{G}_k^{**}}{|\sum_{k=1}^{K} \frac{1}{\lambda_k}\mathbf{G}_k^{**}|^{1/d}}$$

Model $[\lambda \mathbf{DA}_k \mathbf{D}^T]$. Maximizing equation (3.2) leads to the minimization of

$$M_3(\lambda, \mathbf{D}, \mathbf{A}_k) = \frac{1}{\lambda}\sum_{k=1}^{K} tr(\mathbf{G}_k^{**}\mathbf{D}\mathbf{A}_k^{-1}\mathbf{D}^T) + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log(\lambda)$$

To minimize M_3 is to calculate λ and minimize $\sum_{k=1}^{K} tr(\mathbf{G}_k^{**}\mathbf{D}\mathbf{A}_k^{-1}\mathbf{D}^T)$ using an iterative method as following. First step for λ

$$\lambda^{**} = \frac{\sum_{k=1}^{K} tr(\mathbf{G}_k^{**}\mathbf{D}\mathbf{A}_k^{-1}\mathbf{D}^T)}{d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r} \tag{3.5}$$

The second step is to minimize $\sum_{k=1}^{K} tr(\mathbf{G}_k^{**}\mathbf{D}\mathbf{A}_k^{-1}\mathbf{D}^T)$:

• Keeping $\mathbf{D}$ fixed, from Corollary A.5 of the Appendix A, we get

$$\mathbf{A}_k^{**} = \frac{diag(\mathbf{D}^T\mathbf{G}_k^{**}\mathbf{D})}{|diag(\mathbf{D}^T\mathbf{G}_k^{**}\mathbf{D})|^{1/d}}$$

• Keeping $\mathbf{A}_1^{(q+1)}, \ldots, \mathbf{A}_K^{(q+1)}$ fixed, we adapt an algorithm of Flury aiming to minimize $f(\mathbf{D}) = \sum_{k=1}^{K} \pi_k^{(q)} tr(\mathbf{G}_k^{(q+1)}\mathbf{D}\mathbf{A}_k^{-1}\mathbf{D}^T)$: First initial a solution $\mathbf{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_d)$. For any couple $(l, m)(l(\neq)m) \in 1, \ldots, d$, we find a corresponding couple $(\boldsymbol{\delta}_l, \boldsymbol{\delta}_m)$ which are orthogonal vectors, linear combination of $\boldsymbol{d}_l$ and $\boldsymbol{d}_m$, minimizing the criterion $f(\mathbf{D})$. We have

$$\begin{aligned}
\sum_{k=1}^{K} \mathrm{tr}(\mathbf{D}\pi_k^{(q)}\mathbf{A}_k^{-1}\mathbf{D}^T\mathbf{G}_k^{(k+1)}) &= \sum_{k=1}^{K}\sum_{j=1}^{d} \frac{\boldsymbol{d}_j^T\mathbf{G}_k^{(q+1)}\pi_k^{(q)}\boldsymbol{d}_j}{a_k^j} \\
&= \sum_{k=1}^{K} \frac{\boldsymbol{d}_l^T\mathbf{G}_k^{(q+1)}\pi_k^{(q)}\boldsymbol{d}_l}{a_k^l} + \sum_{k=1}^{K} \frac{\boldsymbol{d}_m^T\mathbf{G}_k^{(q+1)}\pi_k^{(q)}\boldsymbol{d}_m}{a_k^m} + \sum_{k=1}^{K}\sum_{j\neq l,m} \frac{\boldsymbol{d}_j^T\mathbf{G}_k^{(q+1)}\pi_k^{(q)}\boldsymbol{d}_j}{a_k^j} \\
&= S(\boldsymbol{d}_l, \boldsymbol{d}_m) + \sum_{k=1}^{K}\sum_{j\neq l,m} \frac{\boldsymbol{d}_j^T\mathbf{G}_k^{(q+1)}\pi_k^{(q)}\boldsymbol{d}_j}{a_k^j}
\end{aligned}$$

Thus, it equals to find $(\boldsymbol{\delta}_l, \boldsymbol{\delta}_m)$ minimizing $S(\boldsymbol{d}_l, \boldsymbol{d}_m)$. We can write

$$\boldsymbol{\delta}_l = (\boldsymbol{d}_l, \boldsymbol{d}_m)\boldsymbol{q}_1$$

$$\boldsymbol{\delta}_m = (\boldsymbol{d}_l, \boldsymbol{d}_m)\boldsymbol{q}_2$$

where $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are two orthogonal vectors of R^2. We have

$$\begin{aligned}
S(\boldsymbol{\delta}_l, \boldsymbol{\delta}_m) &= \sum_{k=1}^{K} \frac{\boldsymbol{q}_1^T(\boldsymbol{d}_l, \boldsymbol{d}_m)^T\mathbf{G}_k^{(q+1)}\pi_k^{(q)}(\boldsymbol{d}_l, \boldsymbol{d}_m)\boldsymbol{q}_1}{a_k^l} \\
&\quad + \sum_{k=1}^{K} \frac{\boldsymbol{q}_2^T(\boldsymbol{d}_l, \boldsymbol{d}_m)^T\mathbf{G}_k^{(q+1)}\pi_k^{(q)}(\boldsymbol{d}_l, \boldsymbol{d}_m)\boldsymbol{q}_2}{a_k^m} \\
&= \sum_{k=1}^{K} \frac{\boldsymbol{q}_1^T\mathbf{Z}_k\boldsymbol{q}_1}{a_k^l} + \sum_{k=1}^{K} \frac{\boldsymbol{q}_2^T\mathbf{Z}_k\boldsymbol{q}_2}{a_k^m}
\end{aligned}$$

where

$$\mathbf{Z}_k = (\boldsymbol{d}_l, \boldsymbol{d}_m)^T \mathbf{G}_k^{(q+1)} \pi_k^{(q)} (\boldsymbol{d}_l, \boldsymbol{d}_m)$$

Denoting $\mathbf{Q} = (\boldsymbol{q}_1, \boldsymbol{q}_2)$, we get

$$\boldsymbol{q}_1^T \mathbf{Z}_k \boldsymbol{q}_1 + \boldsymbol{q}_2^T \mathbf{Z}_k \boldsymbol{q}_2 = tr(\mathbf{Q}^T \mathbf{Z}_k \mathbf{Q}) = tr(\mathbf{Z}_k)$$

And the problem reduces to the optimization of

$$S(\boldsymbol{d}_l, \boldsymbol{d_m}) = \sum_{k=1}^{K} \frac{\boldsymbol{q}_1^T \mathbf{Z}_k \boldsymbol{q}_1}{a_k^l} + \sum_{k=1}^{K} \frac{tr(\mathbf{Z}_k - \boldsymbol{q}_1^T \mathbf{Z}_k \boldsymbol{q}_1)}{a_k^m}$$

which is equivalent to the minimization of

$$\boldsymbol{q}_1^T \{ \sum_{k=1}^{K} (\frac{1}{a_k^l} - \frac{1}{a_k^m}) \mathbf{Z}_k \} \boldsymbol{q}_1$$

Hence, $\boldsymbol{q}_1$ is the second eigenvector of the matrix $\sum_{k=1}^{K} (\frac{1}{a_k^l} - \frac{1}{a_k^m}) \mathbf{Z}_k$. Repeat the procedure above until $f(\mathbf{D})$ converge.

Model $[\lambda_k \mathbf{D} \mathbf{A}_k \mathbf{D}^T]$. For this case, writing $\Sigma_k = \mathbf{D} \mathbf{A}_k \mathbf{D}^T$ where $|\mathbf{A}_k| = |\mathbf{\Sigma}_k|$ is more convenient. Maximizing equation (3.2) equals the minimization of

$$M_4(\mathbf{D}, \mathbf{A}_k) = \sum_{k=1}^{K} tr(\mathbf{D} \mathbf{A}_k^{-1} \mathbf{D}^T \mathbf{G}_k^{**}) + \sum_{k=1}^{K} \sum_{r=1}^{v} z_{kr}^{(q)} n_r \log |\mathbf{A}_k|$$

As previously presented, the minimization of M_4 can be achieved in the similar way:

• Keeping $\mathbf{D}$ fixed, from Corollary A.7 of the Appendix A, we get

$$\mathbf{A}_k^{**} = \frac{diag(\mathbf{D} \mathbf{G}_k^{**} \mathbf{D}^T)}{\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

• For fixed $\mathbf{A}_1^{**}, \ldots, \mathbf{A}_K^{**}$, it can be making use of the same algorithm described above since minimizing M_4 is equivalent to minimize $\sum_{k=1}^{K} tr(\mathbf{D} \mathbf{A}_k^{-1} \mathbf{D}^T \mathbf{G}_k^{**})$.

Model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$. Maximizing equation (3.2) equals the minimization of

$$M_5(\lambda, \mathbf{D}_k, \mathbf{A}) = \frac{1}{\lambda} \sum_{k=1}^{K} tr(\mathbf{D}_k \mathbf{A}^{-1} \mathbf{D}_k^T \mathbf{G}_k^{**}) + d \sum_{k=1}^{K} \sum_{r=1}^{v} z_{kr}^{(q)} n_r \log |\lambda|$$

Considering for $k = 1, \ldots, K$ the eigenvalue decomposition $\mathbf{G}_k^{**} = \mathbf{L}_k^{**}\mathbf{\Omega}_k^{**}L_k^{T**}$ of the symmetric definite positive matrix $\mathbf{G}_k$ with the eigenvalues in the diagonal matrix $\mathbf{\Omega}_k$ in decreasing order, we have

$$\begin{aligned} M_5(\lambda, \mathbf{D}_k, \mathbf{A}) &= \frac{1}{\lambda}\sum_{k=1}^{K} tr(\mathbf{D}_k^T\mathbf{L}_k^{**}\mathbf{\Omega}_k^{**}\mathbf{L}_k^{T**}\mathbf{D}_k\mathbf{A}^{-1}) \\ &\quad + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log|\lambda| \end{aligned}$$

From Theorem A.1 of Appendix A, we get $\mathbf{D}_k = \mathbf{L}_k$, and we have

$$M_5(\lambda, \mathbf{D}_k, \mathbf{A}) = \frac{1}{\lambda}\sum_{k=1}^{K} tr(\mathbf{\Omega}_k^{**}\mathbf{A}^{-1}) + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log|\lambda| \tag{3.6}$$

From which, we deduce the optimal $\mathbf{A}$ and λ

$$\mathbf{A}^{**} = \frac{\sum_{k=1}^{K}\mathbf{\Omega}_k^{**}}{|\sum_{k=1}^{K}\mathbf{\Omega}_k^{**}|^{1/d}}$$

$$\lambda^{**} = \frac{|\sum_{k=1}^{K}\mathbf{\Omega}_k^{**}|^{1/d}}{\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

Model $[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$. Use again the eigenvalue decomposition $\mathbf{G}_k^{(q+1)} = \mathbf{L}_k^{(q+1)}\mathbf{\Omega}_k^{(q+1)}\mathbf{L}_k^{T(q+1)}$. Maximizing equation (3.2) leads to the minimization of

$$M_6(\lambda_k, \mathbf{D}_k, \mathbf{A}) = \sum_{k=1}^{K}\frac{1}{\lambda_k} tr(\mathbf{\Omega}_k^{**}\mathbf{A}^{-1}) + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log|\lambda_k|$$

The minimization of M_6 has to be achieved iteratively:

$$\mathbf{A}^{**} = \frac{\sum_{k=1}^{K}\frac{1}{\lambda_k}\mathbf{\Omega}_k^{**}}{|\sum_{k=1}^{K}\frac{1}{\lambda_k}\mathbf{\Omega}_k^{**}|^{1/d}}$$

and

$$\lambda_k^{**} = \frac{tr(\mathbf{\Omega}_k^{**}\mathbf{A}^{-1})}{d\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

Model $[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$. We write $\mathbf{\Sigma}_k = \lambda\mathbf{C}_k$ where $\mathbf{C}_k = \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^{-1}$. Then, maximizing equation (3.2) equals to minimize

$$M_7(\lambda, \mathbf{C}_k) = \frac{1}{\lambda}\sum_{k=1}^{K} tr(\mathbf{C}_k^{-1}\mathbf{G}_k^{**}) + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log|\lambda|$$

Simple calculation give us:

$$\mathbf{C}_k^{**} = \frac{\mathbf{G}_k^{**}}{|\mathbf{G}_k^{**}|^{1/d}}$$

and

$$\lambda^{**} = \frac{\sum_{k+1}^{K}|\mathbf{G}_k^{**}|^{1/d}}{\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

Model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$. This is the most general situation. Maximizing equation (3.2) leads to the minimization of

$$M_8(\mathbf{\Sigma}_k) = \sum_{k=1}^{K} tr(\mathbf{\Sigma}_k^{-1}\mathbf{G}_k^{**}) + \sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log \mathbf{\Sigma}_k$$

and the variance matrices $\mathbf{\Sigma}_k^{(q+1)}$ are estimated by

$$\mathbf{\Sigma}_k^{**} = \frac{\mathbf{G}_k^{**}}{\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

3.4.2 The diagonal models

For the diagonal family, the orientation is assumed to be horizontal or vertical. So the orientation matrices are either $\mathbf{D}_k = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ or $\mathbf{D}_k = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$, then $\mathbf{\Sigma}_k = \lambda_k \mathbf{A}_k$ or $\mathbf{\Sigma}_k = \lambda_k \mathbf{A}_k^T$. We write $\mathbf{\Sigma}_k = \lambda_k \mathbf{B}_k$ where $\mathbf{B}_k$ is a diagonal matrix with $|\mathbf{B}_k| = 1$. Then the four diagonal models are $[\lambda \mathbf{B}]$, $[\lambda_k \mathbf{B}]$, $[\lambda \mathbf{B}_k]$ and $[\lambda_k \mathbf{B}_k]$.

Model $[\lambda \mathbf{B}]$. In this situation, maximizing equation (3.2) leads to the minimization of

$$M_9(\lambda, \mathbf{B}) = \frac{1}{\lambda}\sum_{k=1}^{K} tr(\mathbf{B}^{-1}\mathbf{G}_k^{**}) + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log|\lambda| \tag{3.7}$$

To estimate the result, we used Corollary A.5 of the Appendix A

$$\mathbf{B}^{**} = \frac{diag(\sum_{k=1}^{K}\mathbf{G}_k^{**})}{|diag(\sum_{k=1}^{K}\mathbf{G}_k^{**})|^{1/d}}$$

and

$$\lambda^{**} = \frac{|diag(\sum_{k=1}^{K}\mathbf{G}_k^{**})|^{1/d}}{\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

Model $[\lambda_k \mathbf{B}]$. In this situation, maximizing equation (3.2) leads to the minimization of

$$M_{10}(\lambda_k, \mathbf{B}) = \sum_{k=1}^{K} \frac{1}{\lambda_k} tr(\mathbf{B}^{-1}\mathbf{G}_k^{**}) + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log|\lambda_k|$$

The minimization of the function M_{10} has to be performed iteratively. • Keep the volumes λ_k's fixed, the matrix $\mathbf{B}$ minimizing M_{10} is minimizing $\sum_{k=1}^{K} \frac{1}{\lambda_k} tr(\mathbf{B}^{-1}\mathbf{G}_k^{**})$, thus, we have

$$\mathbf{B}^{**} = \frac{diag(\sum_{k=1}^{K} \frac{1}{\lambda_k}\mathbf{G}_k^{**})}{|diag(\sum_{k=1}^{K} \frac{1}{\lambda_k}\mathbf{G}_k^{**})|^{1/d}}$$

• When the matrix $\mathbf{B}$ is kept fixed, the λ_k's minimizing B_{10} are

$$\lambda_k^{**} = \frac{tr(\mathbf{B}^{-1}\mathbf{G}_k^{**})}{d\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

Model $[\lambda \mathbf{B}_k]$. In this situation, maximizing equation (3.2) leads to the minimization of

$$M_{11}(\lambda, \mathbf{B}_k) = \frac{1}{\lambda}\sum_{k=1}^{K} tr(\mathbf{B}_k^{-1}\mathbf{G}_k^{**}) + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log|\lambda|$$

From which it follows that

$$\mathbf{B}_k^{**} = \frac{diag(\mathbf{G}_k^{**})}{|diag(\mathbf{G}_k^{**})|^{1/d}}$$

and

$$\lambda^{**} = \frac{\sum_{k=1}^{K} |diag(\mathbf{G}_k^{**})|^{1/d}}{\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

Model $[\lambda_k \mathbf{B}_k]$. In this situation, maximizing equation (3.2) leads to the minimization of

$$M_{12}(\lambda_k, \mathbf{B}_k) = \sum_{k=1}^{K} \frac{1}{\lambda_k} tr(\mathbf{B}_k^{-1}\mathbf{G}_k^{**}) + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log|\lambda_k|$$

From which it follows that

$$\mathbf{B}_k^{**} = \frac{diag(\mathbf{G}_k^{**})}{|diag(\mathbf{G}_k^{**})|^{1/d}}$$

and

$$\lambda_k^{**} = \frac{|diag(\mathbf{G}_k^{**})|^{1/d}}{\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

3.4.3 The spherical models

In spherical family, we assume that the shape of clusters are spherical. Thus the shape matrices are always $diag(1,1)$. Then the variations on the orientation matrices are not necessary. In this case, we have two spherical parsimonious models: $\mathbf{\Sigma}_k = \lambda\mathbf{I}$ and $\mathbf{\Sigma}_k = \lambda_k\mathbf{I}$, where $\mathbf{I}$ denotes the identity matrix.

Model $[\lambda\mathbf{I}]$. In this situation, maximizing equation (3.2) leads to the minimization of

$$M_{13}(\lambda) = \frac{1}{\lambda}\sum_{k=1}^{K} tr(\mathbf{G}_k^{**}) + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log|\lambda| \tag{3.8}$$

where

$$\mathbf{G}_k^{**} = \sum_{r=1}^{v} \frac{z_{kr}^{(q)} n_r}{p_{r/k}^{*}} \int_{\mathcal{H}_r} (\boldsymbol{x} - \boldsymbol{\mu}_k^{**})'(\boldsymbol{x} - \boldsymbol{\mu}_k^{**}) f_k(\boldsymbol{x}; \theta_k^{*}) \mathrm{d}\boldsymbol{x}$$

So we get

$$\lambda^{**} = \frac{\sum_{k=1}^{K} tr(\mathbf{G}_k^{**})}{d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

Model $[\lambda_k\mathbf{I}]$. In this situation, maximizing equation (3.2) leads to the minimization of

$$M_{14}(\lambda_k) = \sum_{k=1}^{K} \frac{1}{\lambda_k} tr(\mathbf{G}_k^{**}) + d\sum_{k=1}^{K}\sum_{r=1}^{v} z_{kr}^{(q)} n_r \log|\lambda_k| \tag{3.9}$$

And we get

$$\lambda_k^{**} = \frac{tr(\mathbf{G}_k^{**})}{d\sum_{r=1}^{v} z_{kr}^{(q)} n_r}$$

3.5 Experiments on simulated data

3.5.1 Experiment of bin-EM-CEM algorithms of fourteen models

In this experiment, we study how these fourteen models perform differently on data of different distributions. According to fourteen parsimonious models, data of fourteen distributions are simulated. To simplify the experiment and display the main comparison, the simulated data are generated in a two-dimensional space (in $\mathbb{R}^2$) with two components of equal mixing proportions. According to each model, 30 samples of $size = 5000$ are generated. Fourteen versions of bin-EM-CEM algorithm (each version associates to one parsimonious models) are applied on each sample. The average of the results of 30 samples is considered as the final result of the model. We define the size of each bin as

$0.5 \cdot 0.5$. Thus, all the space for each sample are cut into subspaces of same size. The number of bins depends on the volume of clusters in each sample, which will be detailed in the description of sample distribution. Since each model has different attribute, to define distance between two mixture components, we use distance value δ:

$$\delta = \sqrt{(\mu_1 - \mu_2)^T (\frac{\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2}{2})^{-1} (\mu_1 - \mu_2)}$$

The parameters of each structure of data are described in the Tables 3.3 and 3.4.

We evaluate the performance of each model by the accuracy and the standard deviation of accuracy. Accuracy is achieved by comparing the cluster result with the correct clustering, indicating the percentage of the data which are correctly clustered. The results are displayed in Tables 3.5 and 3.6.

According to the result, we can analyze as follows:

For the data generated according to the model $[\lambda \mathbf{DAD}^T]$, all the general models obtain the similar good results (around 0.9688), while the diagonal and spherical models can only obtain accuracies from 0.8109 to 0.8197. Because all the general models can adapt to this simplest general model distribution, and the models in the diagonal and spherical families are too simple to provide a good clustering result.

Generally speaking, general models have better performance than the diagonal and spherical models on clustering the data of distribution $[\lambda_k \mathbf{DAD}^T]$. Not surprisingly the model $[\lambda_k \mathbf{DAD}^T]$ provides the highest accuracy. At the same time, we notice that generally all the general models allowing different volumes obtain a better result than the other general models which require the same volumes of all the components. This situation also happens to the diagonal and spherical families: $[\lambda_k \mathbf{B}]$, $[\lambda_k \mathbf{B}_k]$ and $[\lambda_k \mathbf{I}]$ have a higher accuracy than $[\lambda \mathbf{B}]$, $[\lambda \mathbf{B}_k]$ and $[\lambda \mathbf{I}]$.

For the data containing clusters of different shapes, of distribution $[\lambda \mathbf{DA}_k \mathbf{D}^T]$, undoubtedly its own feature model $[\lambda \mathbf{DA}_k \mathbf{D}^T]$ provides the best result. The other good results following are obtained by the general models suggesting different shapes of clusters: $[\lambda_k \mathbf{DA}_k \mathbf{D}^T]$, $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ and $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$.

For the data of distributions $[\lambda_k \mathbf{DA}_k \mathbf{D}^T]$, $[\lambda_k \mathbf{D}_k \mathbf{AD}_k^T]$ and $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$, the best results are obtained respectively by their own feature model. The second best results of these three types of data are provided by the most general model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$. The reason is: except for the feature models, which are exactly the same as the data distribution, only the most general model is able to estimate the model parameters correctly. But when a data distribution is very similar to another model (with slight difference among

Parameters	$[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$	$[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$	$[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	$[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	$[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k \mathbf{I}]$
λ	$\lambda = 1$	$\lambda_1 = 1$ $\lambda_2 = 5$	$\lambda = 1$	$\lambda_1 = 1$ $\lambda_2 = 2$	$\lambda = 1$	$\lambda_1 = 3$ $\lambda_2 = 1$	$\lambda_1 = 1$ $\lambda_2 = 3$
$\mathbf{D}$	$\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$	$\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$	$\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$	$\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$	$\mathbf{D}_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$ $\mathbf{D}_2 = Diag(1,1)$	$\mathbf{D}_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$ $\mathbf{D}_2 = Diag(1,1)$	
$\mathbf{A}$	$\mathbf{A} = Diag(3, 1/3)$	$\mathbf{A} = Diag(3, 1/3)$	$\mathbf{A}_1 = Diag(1,1)$ $\mathbf{A}_2 = Diag(2, 1/2)$	$\mathbf{A}_1 = Diag(1,1)$ $\mathbf{A}_2 = Diag(2, 1/2)$	$\mathbf{A} = Diag(3, 1/3)$	$\mathbf{A} = Diag(3, 1/3)$	
$\boldsymbol{\mu}$	$\boldsymbol{\mu}_1 = (-1.5, 0)$ $\boldsymbol{\mu}_2 = (1.5, 0)$	$\boldsymbol{\mu}_1 = (-2, 0)$ $\boldsymbol{\mu}_2 = (2, 0)$	$\boldsymbol{\mu}_1 = (-1.5, 0)$ $\boldsymbol{\mu}_2 = (1.5, 0)$	$\boldsymbol{\mu}_1 = (-1.5, 0)$ $\boldsymbol{\mu}_2 = (2, 0)$	$\boldsymbol{\mu}_1 = (-2, 1)$ $\boldsymbol{\mu}_2 = (1, 0)$	$\boldsymbol{\mu}_1 = (-2, 1)$ $\boldsymbol{\mu}_2 = (2, 0)$	$\boldsymbol{\mu}_1 = (-2, 0)$ $\boldsymbol{\mu}_2 = (2, 0)$
δ	$\delta = 3.80$	$\delta = 2.98$	$\delta = 3.00$	$\delta = 2.93$	$\delta = 3.19$	$\delta = 3.07$	$\delta = 2.83$
Bin num.	26×19	39×41	22×16	24×24	27×16	37×33	28×23

TABLE 3.3: Parameters of structures of simulated data 1.

Parameters	$[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda \mathbf{B}]$	$[\lambda_k \mathbf{B}]$	$[\lambda \mathbf{B}_k]$	$[\lambda_k \mathbf{B}_k]$	$[\lambda \mathbf{I}]$
λ	$\lambda = 1$	$\lambda_1 = 3$ $\lambda_2 = 1$	$\lambda = 1$	$\lambda_1 = 1$ $\lambda_2 = 3$	$\lambda = 1$	$\lambda_1 = 1$ $\lambda_2 = 2$	$\lambda = 2$
$\mathbf{D}$	$\mathbf{D}_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$ $\mathbf{D}_2 = Diag(1,1)$	$\mathbf{D}_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$ $\mathbf{D}_2 = Diag(1,1)$					
$\mathbf{A}$	$\mathbf{A}_1 = Diag(3, 1/3)$ $\mathbf{A}_2 = Diag(1,1)$	$\mathbf{A}_1 = Diag(3, 1/3)$ $\mathbf{A}_2 = Diag(1,1)$					
$\mathbf{B}$			$\mathbf{B} = Diag(3, 1/3)$	$\mathbf{B} = Diag(3, 1/3)$	$\mathbf{B}_1 = Diag(2, 1/2)$ $\mathbf{B}_2 = Diag(4, 1/4)$	$\mathbf{B}_1 = Diag(2, 1/2)$ $\mathbf{B}_2 = Diag(4, 1/4)$	
$\boldsymbol{\mu}$	$\boldsymbol{\mu}_1 = (-2, 0)$ $\boldsymbol{\mu}_2 = (1, 0)$	$\boldsymbol{\mu}_1 = (-2, 0)$ $\boldsymbol{\mu}_2 = (2, 0)$	$\boldsymbol{\mu}_1 = (-2.8, 2.8)$ $\boldsymbol{\mu}_2 = (2.8, 0)$	$\boldsymbol{\mu}_1 = (-4, 0)$ $\boldsymbol{\mu}_2 = (4, 0)$	$\boldsymbol{\mu}_1 = (-2.5, 0)$ $\boldsymbol{\mu}_2 = (3, 0)$	$\boldsymbol{\mu}_1 = (-3.6, 0)$ $\boldsymbol{\mu}_2 = (-3.6, 0)$	$\boldsymbol{\mu}_1 = (-2.1, 0)$ $\boldsymbol{\mu}_2 = (2.1, 0)$
δ	$\delta = 3.00$	$\delta = 3.10$	$\delta = 3.23$	$\delta = 3.27$	$\delta = 3.18$	$\delta = 3.22$	$\delta = 2.90$
Bin num.	22×18	30×31	35×9	52×14	34×10	42×9	27×22

TABLE 3.4: Parameters of structures of simulated data 2.

Algorithm \ Data Structure	$[\lambda\mathbf{DAD}^T]$	$[\lambda_k\mathbf{DAD}^T]$	$[\lambda\mathbf{DA}_k\mathbf{D}^T]$	$[\lambda_k\mathbf{DA}_k\mathbf{D}^T]$	$[\lambda\mathbf{D}_k\mathbf{AD}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{AD}_k^T]$	$[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda\mathbf{DAD}^T]$	0.9687(.0025)	0.8900(.0058)	0.8997(.0073)	0.9154(.0058)	0.8013(.0522)	0.7017(.0170)	0.8367(.0269)
$[\lambda_k\mathbf{DAD}^T]$	0.9686(.0021)	**0.9532(.0030)**	0.8979(.0095)	0.9069(.0085)	0.8708(.0142)	0.9320(.0084)	0.8540(.0287)
$[\lambda\mathbf{DA}_k\mathbf{D}^T]$	0.9689(.0027)	0.8906(.0073)	**0.9252(.0032)**	0.8692(.0114)	0.8870(.0109)	0.8218(.0197)	0.8847(.0091)
$[\lambda_k\mathbf{DA}_k\mathbf{D}^T]$	0.9689(.0027)	0.9528(.0035)	0.9194(.0056)	**0.9402(.0056)**	0.8737(.0141)	0.9490(.0049)	0.8946(.0099)
$[\lambda\mathbf{D}_k\mathbf{AD}_k^T]$	0.8966(.0064)	0.9134(.0053)	0.9103(.0074)	0.9268(.0048)	**0.9527(.0043)**	0.9087(.0086)	0.9243(.0230)
$[\lambda_k\mathbf{D}_k\mathbf{AD}_k^T]$	0.9689(.0025)	0.9528(.0027)	0.9096(.0056)	0.9120(.0059)	0.9378(.0061)	**0.9608(.0028)**	0.9037(.0106)
$[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	0.9688(.0024)	0.8876(.0062)	0.9181(.0084)	0.9164(.0085)	0.9402(.0056)	0.9506(.0032)	**0.9259(.0052)**
$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	0.9689(.0026)	0.9521(.0033)	0.9141(.0065)	0.9204(.0059)	0.9386(.0067)	0.9606(.0027)	0.9223(.0060)
$[\lambda\mathbf{B}]$	0.8197(.0058)	0.7747(.0039)	0.9140(.0081)	0.9034(.0105)	0.8614(.0105)	0.8957(.0112)	0.9016(.0106)
$[\lambda_k\mathbf{B}]$	0.8192(.0071)	0.8712(.0066)	0.9073(.0120)	0.9172(.0064)	0.8591(.0064)	0.9051(.0081)	0.9071(.0054)
$[\lambda\mathbf{B}_k]$	0.8182(.0053)	0.7820(.0053)	0.9057(.0123)	0.9060(.0072)	0.8971(.0081)	0.8918(.0118)	0.8979(.0121)
$[\lambda_k\mathbf{B}_k]$	0.8211(.0087)	0.8713(.0068)	0.9057(.0121)	0.9069(.0104)	0.9041(.0108)	0.9020(.0073)	0.9015(.0090)
$[\lambda\mathbf{I}]$	0.8124(.0081)	0.7180(.0141)	0.9134(.0105)	0.8539(.0087)	0.8424(.0083)	0.8524(.0085)	0.8652(.0175)
$[\lambda_k\mathbf{I}]$	0.8109(.0076)	0.8674(.0075)	0.9099(.0082)	0.9193(.0059)	0.8306(.0112)	0.9239(.0065)	0.8953(.0139)

TABLE 3.5: Accuracy and standard deviation of accuracy of bin-EM-CEM algorithms on simulated data.

Algorithm \ Data Structure	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda\mathbf{B}]$	$[\lambda_k\mathbf{B}]$	$[\lambda\mathbf{B}_k]$	$[\lambda_k\mathbf{B}_k]$	$[\lambda\mathbf{I}]$	$[\lambda_k\mathbf{I}]$
$[\lambda\mathbf{D}\mathbf{A}\mathbf{D}^T]$	0.7507(.0162)	0.9439(.0037)	0.9343(.0083)	0.9393(.0048)	0.9371(.0076)	0.9238(.0055)	0.8999(.0091)
$[\lambda_k\mathbf{D}\mathbf{A}\mathbf{D}^T]$	0.9144(.0201)	0.9439(.0038)	0.9577(.0019)	0.9322(.0057)	0.9261(.0072)	0.9236(.0055)	0.9306(.0054)
$[\lambda\mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	0.7863(.0101)	0.9438(.0037)	0.9292(.0057)	0.9471(.0053)	0.9488(.0035)	0.9237(.0054)	0.9021(.0108)
$[\lambda_k\mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	0.8631(.0124)	0.9429(.0024)	0.9573(.0026)	0.9471(.0053)	0.9487(.0035)	0.9238(.0055)	0.9313(.0051)
$[\lambda\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	0.9688(.0024)	0.9440(.0037)	0.9363(.0056)	0.9389(.0045)	0.9391(.0068)	**0.9241(.0053)**	0.9064(.0079)
$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	0.9351(.0064)	0.9439(.0038)	0.9579(.0019)	0.9393(.0048)	0.9382(.0073)	0.9236(.0055)	0.9312(.0051)
$[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	0.9197(.0067)	0.9434(.0026)	0.9306(.0057)	0.9470(.0054)	0.9488(.0035)	0.9239(.0053)	0.9014(.0083)
$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	**0.9495(.0029)**	0.9429(.0024)	0.9574(.0026)	0.9471(.0054)	0.9487(.0035)	0.9236(.0055)	0.9311(.0051)
$[\lambda\mathbf{B}]$	0.8222(.0057)	**0.9440(.0038)**	0.9400(.0040)	0.9393(.0048)	0.9401(.0075)	0.9236(.0055)	0.9131(.0082)
$[\lambda_k\mathbf{B}]$	0.9364(.0049)	0.9439(.0038)	**0.9581(.0017)**	0.9394(.0048)	0.9401(.0074)	0.9236(.0055)	0.9328(.0041)
$[\lambda\mathbf{B}_k]$	0.8298(.0068)	0.9438(.0037)	0.9424(.0064)	**0.9471(.0053)**	0.9487(.0035)	0.9236(.0054)	0.9097(.0082)
$[\lambda_k\mathbf{B}_k]$	0.9370(.0058)	0.9429(.0024)	0.9575(.0025)	0.9471(.0053)	**0.9488(.0035)**	0.9236(.0055)	0.9333(.0037)
$[\lambda\mathbf{I}]$	0.7717(.0114)	0.9433(.0034)	0.9380(.0061)	0.9402(.0039)	0.9420(.0052)	0.9236(.0055)	0.9045(.0064)
$[\lambda_k\mathbf{I}]$	0.9350(.0058)	0.9436(.0037)	0.9569(.0031)	0.9446(.0047)	0.9439(.0063)	0.9236(.0055)	**0.9336(.0055)**

TABLE 3.6: Accuracy and standard deviation of accuracy of bin-EM-CEM algorithms on simulated data.

the volumes or orientations or shape), this model which cannot exactly represent the sample distribution can still obtain a good result.

For data generated according to model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$, the three highest accuracies are given by the models allowing different orientations among clusters: $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$, $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ and $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$.

When dealing with the data composed of the clusters of different volumes, different orientations and different shapes, model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ is the best model.

The data simulated according to the model $[\lambda \mathbf{B}]$ is two identical diagonal elliptical clusters. All the general and diagonal models are capable in finding clusters in this type of data. Hence, the results are almost the same. This proves that when the data distribution is not complicated, we can use simple model instead of complex model with many parameters.

The data of model $[\lambda_k \mathbf{B}]$ contains clusters of different volumes. In this case, all the models which suggest different volumes provide better results than the models which don't. Because the data is in the shape of ellipse which is closed to spherical shape too, thus even the spherical models can have high accuracies: 0.9380 and 0.9569.

The data of model $[\lambda \mathbf{B}_k]$ that we simulated are two different diagonal clusters of same volumes. It can also be considered as two clusters of same volumes, same vertical or horizontal orientations and different shapes, which is proved by the fact that all the general models which allow different shapes have good results. This theory also can be applied on the data of model $[\lambda_k \mathbf{B}_k]$ to explain why other models can provide good results.

When the data distribution is very simple, such as $[\lambda \mathbf{I}]$, there is small difference among the results of the fourteen models.

The model of $[\lambda_k \mathbf{I}]$ is giving the best accuracy for the data simulated according to the model $[\lambda_k \mathbf{I}]$. This shows again the clustering ability of parsimonious models.

After the analysis above, we can conclude as follows: The best results are always obtained by the models which can perfectly represent the distribution of data. The models which are more complex than the data distribution but contain the data distribution can also provide good clustering results. For instance, models $[\lambda_k \mathbf{D} \mathbf{A} \mathbf{D}^T]$, $[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ and $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ perform well on dealing with the data of distribution $[\lambda_k \mathbf{D} \mathbf{A} \mathbf{D}^T]$. Generally speaking, when dealing with data structured according to general models, diagonal and spherical families have worse performance than general family. Because diagonal and spherical models are too simple to meet the requirement in estimating the parameters of a more complex model. In the cases where the model is simpler than the

data distribution but closed to the data distribution, the model can still provide a good result. Specially when the clusters of data are well separated, differences of the results among fourteen models are diminished. We notice that in general the standard deviation of the accuracy of a model is low when the accuracy is high. The result of diagonal family proves again the fact that the model which is exactly the same as the data distribution can give the best result for the data. The good performance of spherical family indicates that parsimonious models are capable of finding reasonable groups of data instead of using the complex models.

3.5.2 Experiment of bin-EM-CEM algorithm with different sizes of bin

The essential of bin-EM-CEM algorithm is to estimate the parameters maximizing the likelihood for binned data. Thus the size of bin is a very important information which affects directly the clustering result. This experiment shows how the accuracy and CPUtime will change when the size of bin is different.

For example, in the Figure 3.1 there is four samples with different bin size in a view of 3d. These samples are generated according to the same model of two clusters in a two dimensional space.

Comparing to the model distribution in the Figure 3.2,

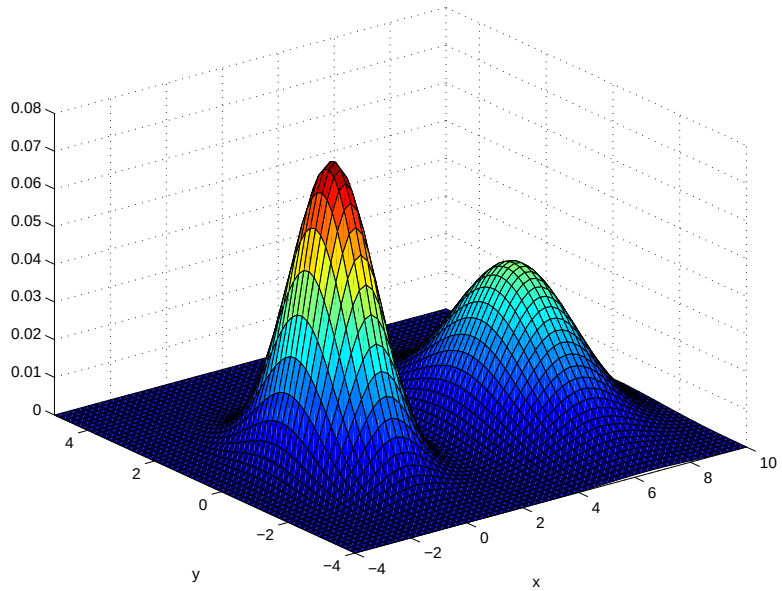


FIGURE 3.2: Density of a Gaussian mixture distribution of two clusters in a two dimensional space.

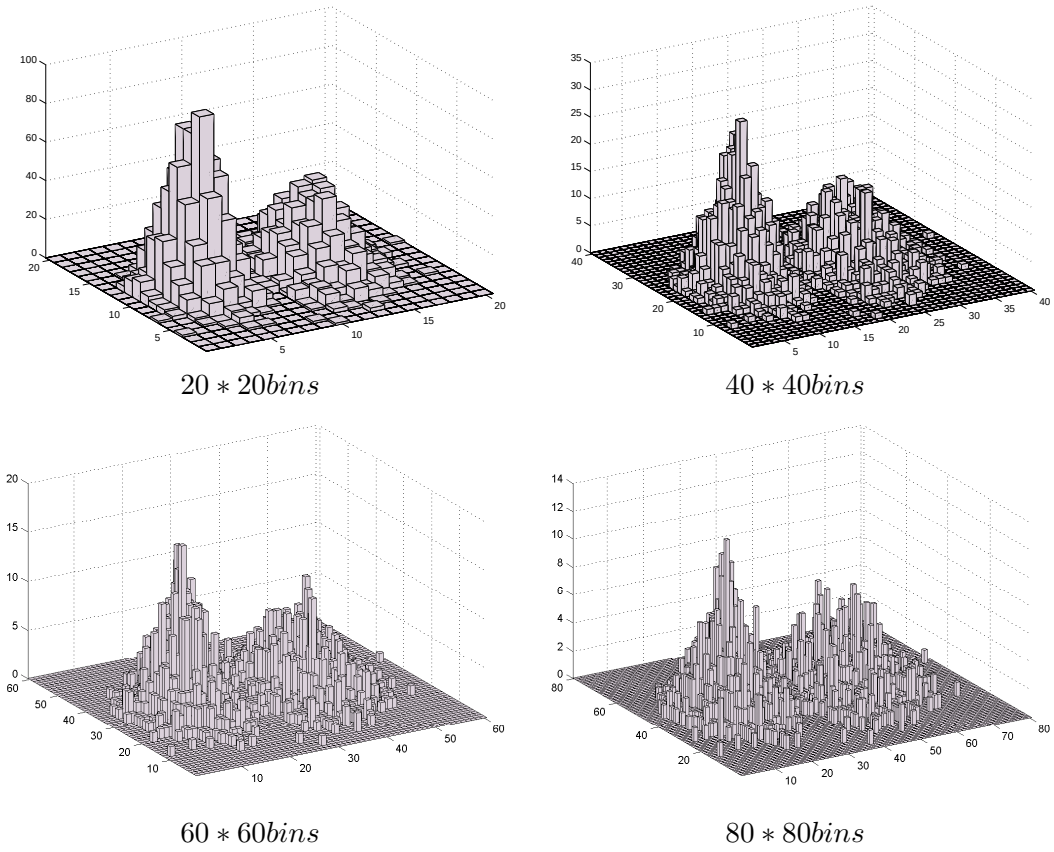


FIGURE 3.1: Four samples generated according to the same model with different bin size.

we can notice that, along with the increase of the number of bins, the binned data is closer to the real distribution of the underlying model. When the bin size is bigger, the data distribution shown in the figure is more general.

We define the size of bins by the number of bins per dimension. We consider the number of bins from 10 to 100 with an interval of 10. For each bin size, 30 samples of size= 5000 are generated according to the model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ with $\lambda = 1$, $\mathbf{D}_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{D}_2 = Diag(1,1)$, $\mathbf{A} = Diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-2, 1)$, $\boldsymbol{\mu}_2 = (1, 0)$ and $\delta = 3.1$. Bin-EM-CEM algorithm of model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ is applied on each sample. The average of 30 samples is considered as the final result of each bin size. The accuracy and CPUtime will be two evaluation factors. Accuracy indicates the percentage of the data which are correctly classified, while the CPU time is the amount of time for which a central processing unit (CPU) was used for our algorithm computation.

The results are displayed in Table 3.7 and Figure 3.3.

Bins num.	Accuracy	CPUtime	Bin num.	Non-empty-bin num.	Bin size
100	0.9439	353	10000	1827	0.13 · 0.09
90	0.9435	305	8100	1453	0.15 · 0.10
80	0.9430	259	6400	1186	0.17 · 0.12
70	0.9427	217	4900	1129	0.19 · 0.13
60	0.9424	176	3600	914	0.22 · 0.15
50	0.9420	137	2500	769	0.28 · 0.18
40	0.9412	115	1600	476	0.33 · 0.23
30	0.9407	84	900	319	0.46 · 0.30
20	0.9389	45	400	160	0.68 · 0.45
10	0.9299	5	100	53	0.36 · 0.89

TABLE 3.7: Result of bin-EM-CEM algorithm with different size of bins on simulated data.

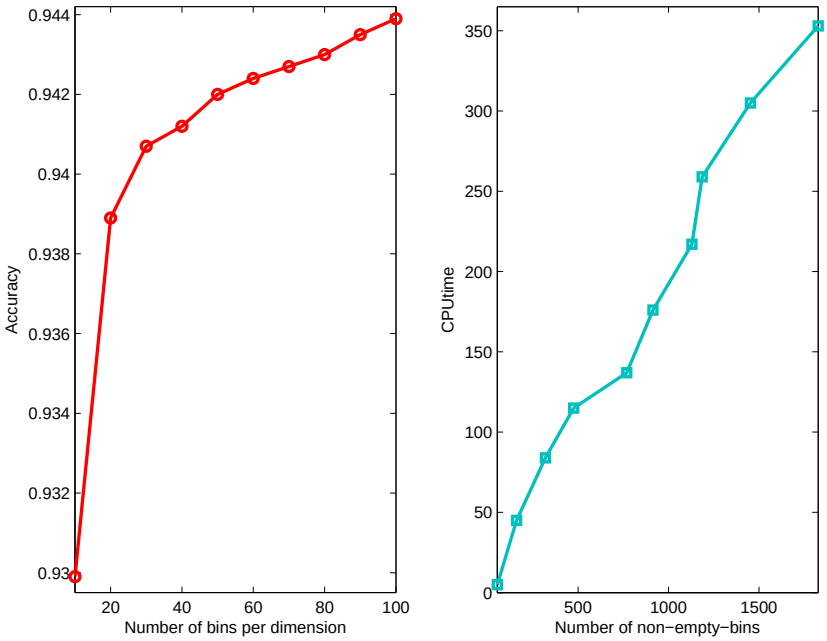


FIGURE 3.3: Result of bin-EM-CEM algorithm on simulated data with different size of bins.

From the result and the Figure 3.3, the accuracy increases significantly along with the number of bins per dimension. When the number of bins changes from 10 to 20, there is a relatively big increase in the accuracy: from 0.9299 to 0.9389. The speed of the increasing trend slows down until that the number of bins is 40, then the accuracy increases stably. Even when we divide the space only in 10 bins per dimension, the result is still satisfying with accuracy of 0.9299 and only 5 seconds. The CPUtime increase at a stable pace along with the number of non-empty-bins deal to our programming considering only the non-empty-bins. When the number of bins per dimension is 100, the CPUtime increase up to 353 seconds. But deal to the methods in programming and the type of calculators, the CPUtime in this paper cannot be compared to the experimental results of CEM algorithm and binn-EM-CEM algorithm in previous papers. As indicated and proved in the papers of Hamdan and Samé [53], [54], [14], [55], binning data in the classification approach helps in reducing computation time. Despite this, presenting the CPUtime in this paper is helpful in comparing and studying how the computation change when we modify the size of bins.

We can conclude that the computation time of bin-EM-CEM algorithm can be reduced by modifying the size of bins with certain loss of accuracy. It is important to find the balance between gaining the computation time and losing precision of the result according to the real situation.

3.6 Experiments on real data

3.6.1 French city clustering

The characteristic of bin-EM-CEM algorithms of fourteen parsimonious models were studied and presented in the two previous experiments, by applying on simulated data. And we can't neglect the importance of the performance of these algorithms in real life. In this experiment, we will test the practical application ability of these fourteen bin-EM-CEM algorithms on real dataset. As real dataset, we have the population and the population density (population/surface) of 1193 French cities. They are respectively from three different departments: Meuse (500 cities), Nord (652 cities) and Val-de-Marne (47 cities). Meuse is a rural department with a small population and low population density. In the opposite, Val-de-Marne, situated to southeast of Paris, is a department of high population density and Nord is the most populous department in France. We try to cluster these 1193 cities by bin-EM-CEM algorithms, with the only information of population and population density. As we already know the origin department of each city, we compare it with the clustering result in order to obtain the accuracy. At the same time, the computation time is noted down for comparison among the algorithms.

Figure 3.4 displays 1193 observations concerning the log-population and log-density of cities from Meuse (500 observations), Nord (652 observations) and Val-de-Marne (47 observations). The space is divided into $50 \cdot 50$ bins. Each model is applied on the data for 30 times. The average of 30 results as the final result of the model.

The result is displayed in Table 3.8 and Figure 3.5.

Figure 3.5 shows the one of the clustering results of bin-EM-CEM algorithms. Compare Figure 3.4 and Figure 3.5, three departments are basically correctly clustered. Red stars between every two clusters are the points which were incorrectly clustered. It shows the difficulty of clustering the data which locates in the mixing area of two clusters.

From Table 3.8, the accuracies of fourteen algorithms are all above 0.7913. We can say that all the algorithms of different models are capable of putting these French cities in the right group. We notice that the general models perform better than the diagonal models and spherical models. It means that general family is more suitable for the data

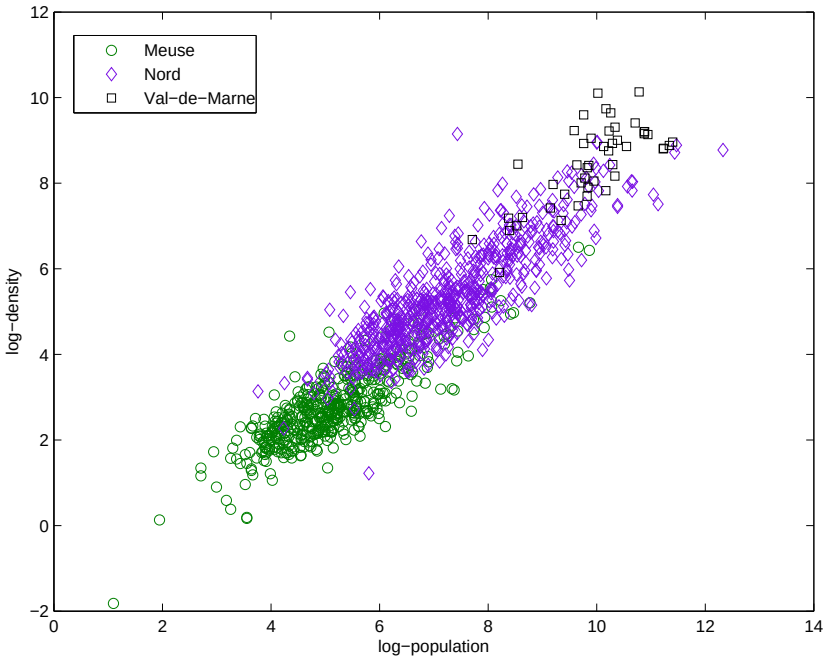


FIGURE 3.4: Log-population and log-density of 1193 cities from three departments in France.

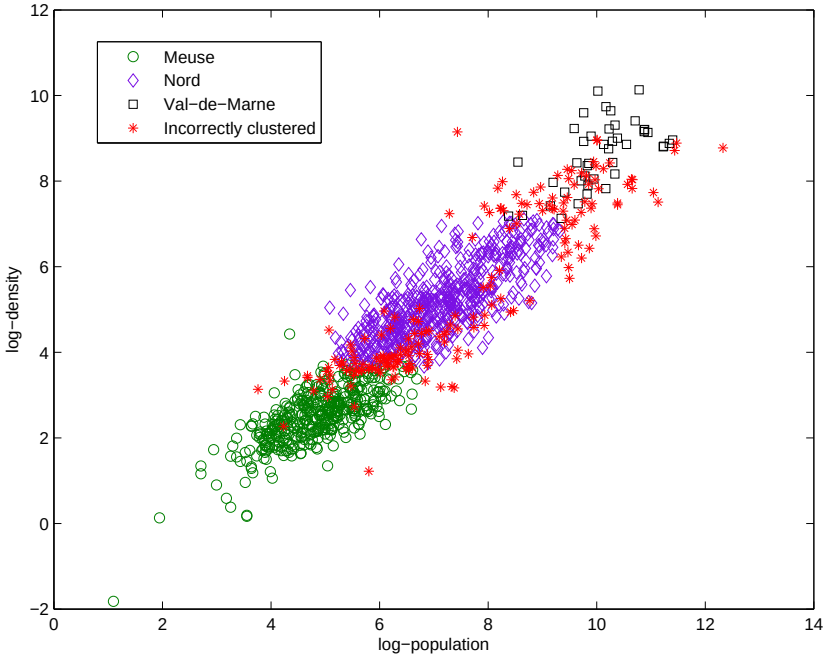


FIGURE 3.5: Incorrectly clustered points of bin-EM-CEM algorithm result on real data on French cities.

Model	Accuracy	Time
$[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$	0.8047	64.0
$[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$	0.8022	64.0
$[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	0.7913	85.4
$[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	0.7888	85.6
$[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	0.8189	65.4
$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	**0.8374**	89.0
$[\lambda \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	0.8256	64.2
$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	0.8215	64.2
$[\lambda \mathbf{B}]$	0.7913	**42.4**
$[\lambda_k \mathbf{B}]$	0.7913	42.6
$[\lambda \mathbf{B}_k]$	0.7913	42.5
$[\lambda_k \mathbf{B}_k]$	0.7913	42.5
$[\lambda \mathbf{I}]$	0.7988	57.3
$[\lambda_k \mathbf{I}]$	0.8005	57.3

TABLE 3.8: Accuracy, CPUtime of bin-EM-CEM algorithm on French cities clustering.

distribution than the other two families in this case. Model $[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$ obtains the highest accuracy: 0.8374, but it also takes the longest computation time. Model $[\lambda \mathbf{B}]$ takes the least CPUtime but it provides the lowest accuracy. It is hard to define which model is the best one for the real dataset. It depends on the practical needs in reality: high precision or fast computation process. In this case, we could try to find out the best model by some criterion. This would be studied in our following research.

3.6.2 Image segmentation

In the experiment part of the Chapter 2, we introduced applying binned-EM algorithm to image segmentation. In this part, we will apply bin-EM-CEM algorithm of parsimonious models to image segmentation. The goal is to compare the performance of the bin-EM-CEM algorithm of parsimonious models on real application.

The Figure 3.6 shows the image to be processed:

FIGURE 3.6: Original image.

This is an image of colorful lanterns for Chinese Mid-Autumn festival. There are three main colors of lanterns in the image. The background is black, with several blurred silhouettes. On the right up corner ornaments some blue lights.

The image is converted from RGB color space to $L*a*b*$ color space, where a luminosity layer $'L*'$, chromaticity-layer $'a*'$ indicating where color falls along the red-green axis, and chromaticity-layer $'b*'$ indicating where the color falls along the blue-yellow axis. After this, we can represent the image in a two dimensional $'a*b*'$ space, as shown in the Figure 3.7:

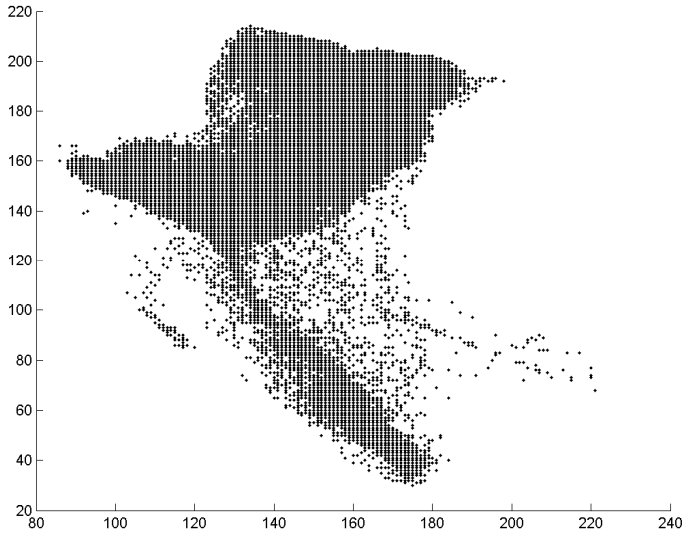


FIGURE 3.7: Image pixel represented in the $'a*b*'$ space.

There are totally 426400 pixels in the image, so 426400 points in the Figure 3.7. As we analyzed, there are mainly five colors in the image. but it is difficult to find out

five clusters in the Figure 3.7. We will apply bin-EM-CEM algorithms of parsimonious models to discovered these clusters.

3.6.2.1 With different models

From the Figure 3.7, the distribution model for this dataset is not evident. We can assume that the data don't follow the diagonal models, either the spherical models. Thus, we will apply bin-EM-CEM algorithms of eight general models to the dataset, in order to find out the most suitable model.

To obtain binned data, the space is divided into 20 bins per dimension. We suppose that the number of clusters is known as 5. The image segmentation result is shown in the Figure 3.8:

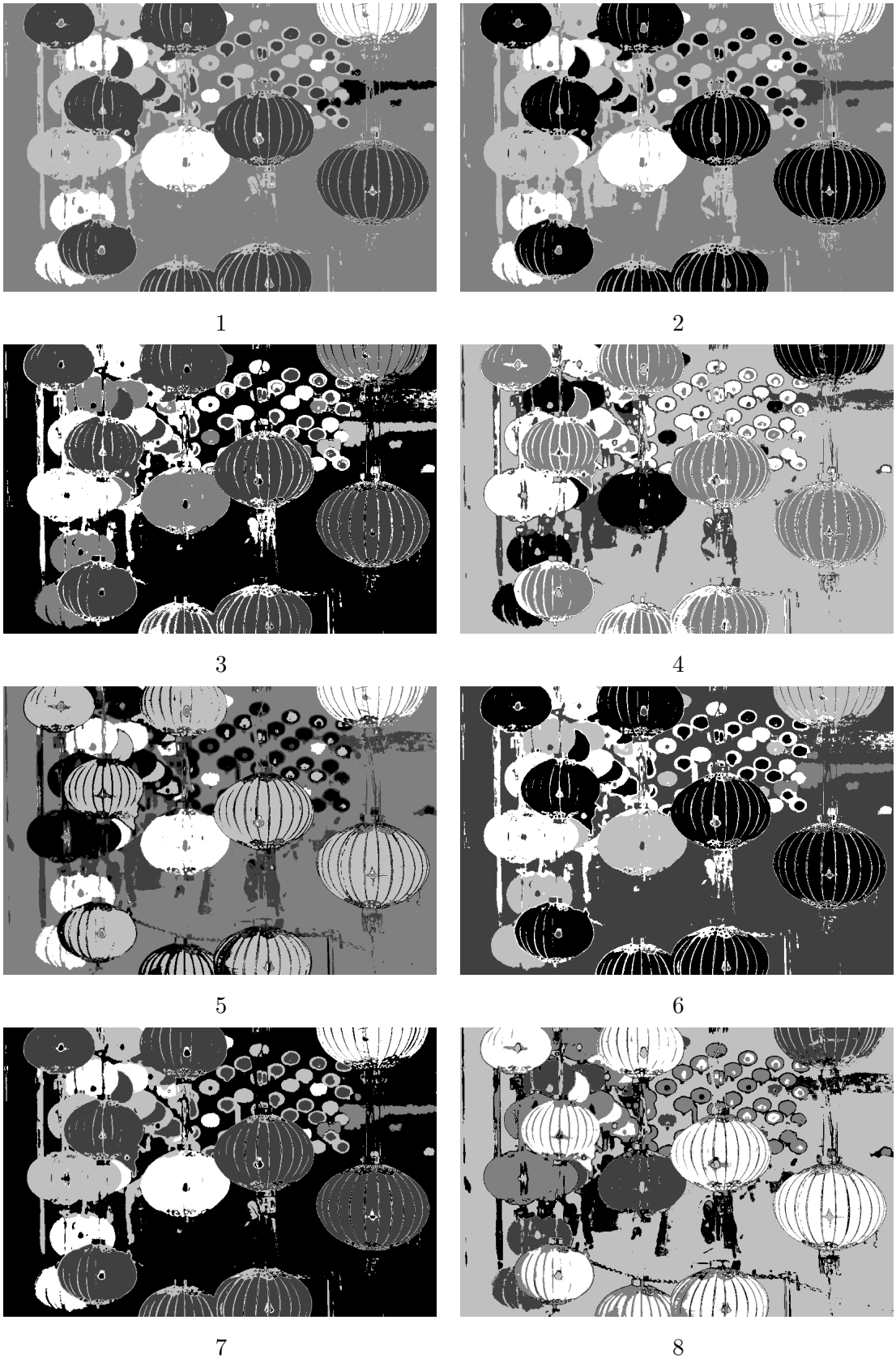


FIGURE 3.8: Result of image segmentation of Figure 3.6 by bin-EM-CEM algorithm of eight general models: 1. $[\lambda \mathbf{DAD}^T]$, 2. $[\lambda_k \mathbf{DAD}^T]$, 3. $[\lambda \mathbf{DA}_k \mathbf{D}^T]$, 4. $[\lambda_k \mathbf{DA}_k \mathbf{D}^T]$, 5. $[\lambda \mathbf{D}_k \mathbf{AD}_k^T]$, 6. $[\lambda_k \mathbf{D}_k \mathbf{AD}_k^T]$, 7. $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$, 8. $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$

In the Figure 3.8, all the eight models succeeded in separating the main colors. The worse result is obtained by the model $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$. In the result of the model $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, the bunch of blue light is not detected. Except for this model, all the results are similar. The differences only exist in the small details. For example, the center of a small lantern is detected in the models $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$ and $[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$, but not in the other models. But these details are not very important for the result. We can say that, except for the model $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, the other seven general models are suitable for image segmentation of this image.

To define the best model, we consider also the maximum complete likelihood and the computation time. Since the size of bin is the same among eight bin-EM-CEM algorithms of eight models, the 426400 pixels are grouped into 215 non empty bins for all the models. The Table 3.9 shows these information of eight models:

Model \ Info.	LM_{max}	CPUtime(s)
$[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$	-5.45×10^5	23
$[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$	-4.65×10^5	15
$[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	-3.85×10^5	22
$[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	-4.75×10^5	29
$[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	-3.15×10^5	14
$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	-3.15×10^5	15
$[\lambda \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	-2.92×10^8	23
$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	-2.85×10^5	22

TABLE 3.9: Information of the image segmentation of Figure 3.6 by bin-EM-CEM algorithms of eight general models.

From the Table 3.9, the more complex the model is, the higher maximum complete likelihood it obtains. The most complex model $[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$ provides the highest maximum complete likelihood. But it takes 22 seconds, which is much longer than 14 seconds, which is the time for the model $[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$. To conclude, if the objective is to segmenting the image in the shortest time, the model $[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$ is the best model. If the objective is to obtain the best image segmentation result, the model $[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$ should be chosen.

The Figure 3.9 shows the clustering result of the dataset by the bin-EM-CEM algorithm of the model $[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$ with 20 bins per dimension:

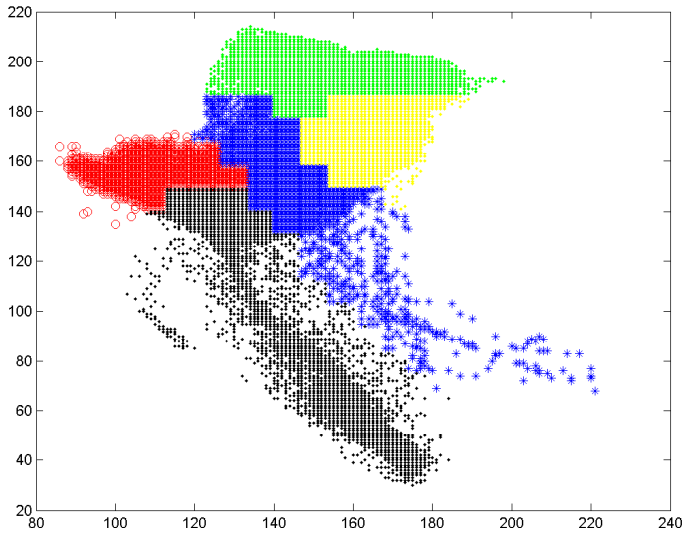


FIGURE 3.9: The clustering result of the dataset by the bin-EM-CEM algorithm of the model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ with 20 bins per dimension.

3.6.2.2 With different bin sizes

Different size of bins in bin-EM-CEM algorithm leads to different clustering result. We don't have a method to define the best bin size for dataset. In this part, we will study the influence of the bin size on the image segmentation result. We will process the same image as in the Figure 3.6. According to the comparison and analysis in the previous part, the model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ is chosen as the objective model. We will apply the bin-EM-CEM algorithms of model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ to the dataset. The only change is the size of bins. It changes among 5, 10, 15, 20, 25, 30, 35 and 40 bins per dimension.

The Figure 3.10 shows the image segmentation results of the Figure 3.6 by the bin-EM-CEM algorithms with different bin size:

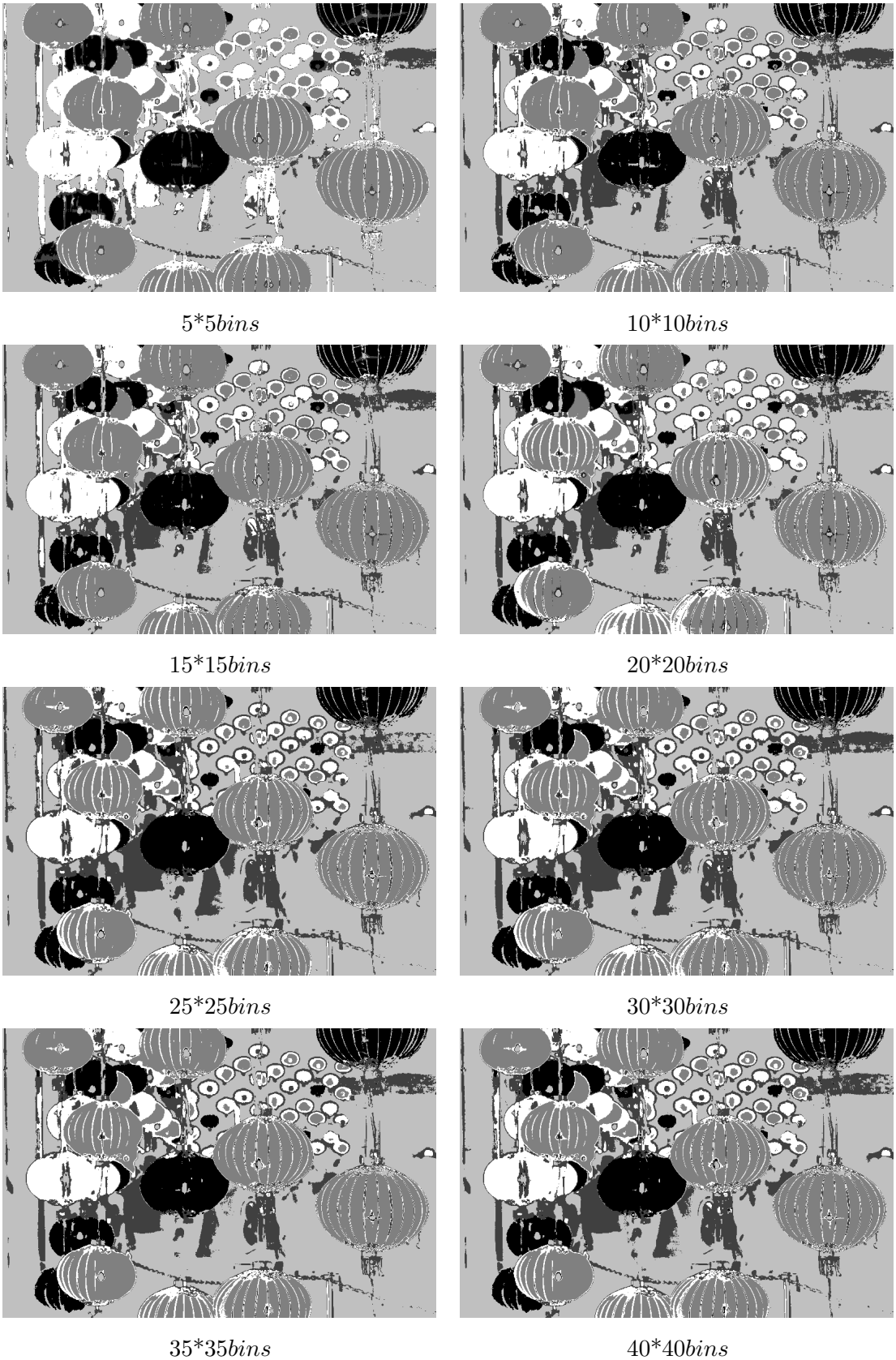


FIGURE 3.10: Image segmentation results of the Figure 3.6 by the bin-EM-CEM algorithms of the model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ with different bin size

From the Figure 3.10, the image segmentation results are very similar to each other among these eight situations. We can see that when the bin size is 5 bins per dimension, some details are missing. When the space is divided into 10 bins per dimension, we already obtain a very good image segmentation result by bin-EM-CEM algorithm. There are not obvious difference among the results with 10, 15, 20, 25, 30, 35 and 40 bins per dimension.

Let's see in the Table 3.10 some data about the experiment:

Info. / Bin size	L_M	CPUtime(s)	Num. of non-empty bins
5 bins/dimension	-7.20×10^4	4	19
10 bins/dimension	-1.69×10^5	10	63
15 bins/dimension	-2.26×10^5	13	128
20 bins/dimension	-3.15×10^5	14	215
25 bins/dimension	-3.60×10^5	24	326
30 bins/dimension	-3.80×10^5	31	433
35 bins/dimension	-3.99×10^5	41	575
40 bins/dimension	-4.02×10^5	51	723

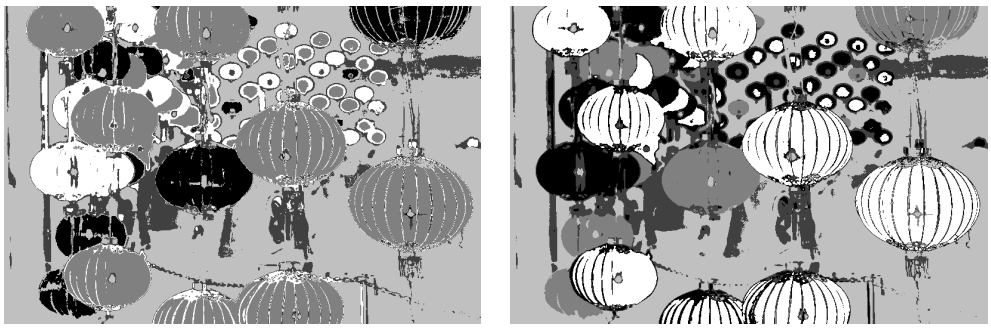
TABLE 3.10: Information of the image segmentation by bin-EM-CEM algorithm of the model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ with different size of bins.

The maximum complete likelihood increases along with the decrease of the number of non-empty bins. This is easy to proved and understand. Thus the highest maximum complete likelihood doesn't represent the best bin size. The CPUtime increases along with the increase of the number of non-empty bins. This is naturel. There is not a lot to explain here.

Considering that the image segmentation result is good enough when the space is divided into 10 bins per dimension, we can say that the best bin size is 10 bins per dimension, because it takes very little time to obtain a satisfying result.

3.6.2.3 Comparison with classical CEM algorithm

This part will compare the results of bin-EM-CEM algorithm and CEM algorithm. The CEM algorithm is of the model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$. The bin-EM-CEM algorithm is of the same model with 10 bins per dimension. The comparison is shown in the Figure 3.11:



$bin - EM - CEM$ CEM

FIGURE 3.11: comparison between the result by CEM algorithm and the result by bin-EM-CEM algorithm

The result of the bin-EM-CEM algorithm is almost the same as the one of CEM algorithm. It shows that even the 426400 pixels are grouped into only 63 bins for the bin-EM-CEM algorithm, the quality of image segmentation is not affected. And the bin-EM-CEM algorithm has bigger advantage than the CEM algorithm in computation time. The CEM algorithm takes 38 seconds, while the bin-EM-CEM algorithm takes only 10 seconds.

3.7 Conclusion

In this chapter, we developed fourteen bin-EM-CEM algorithms of parsimonious Gaussian mixture models. The classic CEM algorithm for the standard data was first reviewed. In order to compare the computation times of the CEM and the bin-EM-CEM algorithms, we calculated the computation complexities of these two algorithms for the simplest general model. The comparison shows that the bin-EM-CEM algorithm takes less computation time than the CEM algorithm if the amount of data is large enough. An inequality is given out to define the condition when the bin-EM-CEM algorithm is faster than the CEM algorithm. Derivation of the bin-EM-CEM algorithm and the parameter estimates for fourteen models were detailed. The estimations of the variance matrices of the mixture components differ among fourteen models. At the end of this chapter, we studied and compared the performances of the bin-EM-CEM algorithms of different models in several experiments on simulated data and real data. Some important remarks can be outlined as follows:

- The best clustering result is most of the time obtained by the right model which has the same distribution as the data. This result highlights that the parsimonious models help in improving the clustering precision.

- The execution time of the bin-EM-CEM algorithm can be reduced by enlarging the size of bins. But this might lead to certain loss of accuracy, it will be convenient to have a good trade-off between the reduction of size of bins and the accuracy requirements.
- The bin-EM-CEM algorithms of parsimonious Gaussian mixture models perform well on French department data and on image segmentation. In image segmentation, even with big bin size, the bin-EM-CEM algorithms provide a good result. And the bin-EM-CEM algorithms have evident advantages in computation comparing to the CEM algorithms when applying to image segmentation.
- A criterion is needed to choose the best model which fits well the data and satisfy the clustering accuracy requirements with a reasonable computation time.

To answer the last remark, in the next chapter, we will discuss some commonly used criteria to choose the model for data clustering and adapt them to binned data framework.

Chapter 4

Criteria for binned data model-based clustering

4.1 Introduction

As we introduced in the previous chapters, model-based clustering becomes a common approach in cluster analysis. Without any information on the data structure, choosing the right model is decidedly an important step. Various criteria have been proposed to measure a model's suitability to the dataset. They are often supposed to find a balance between the suitability and the model complexity.

Bayesian Information Criterion (BIC), introduced by Schwarz [20], is a common-used criterion for model selection basing on likelihood function. In the Bayesian framework, under the condition that the models have the same prior probabilities, selecting the model of the highest posterior probability is equivalent to selecting the model of the largest integrated likelihood [21]. But when fitting mixture models, the likelihood tends to increase along with the model complexity *i.e.* by adding mixture components and mixture model parameters, and it finally results in over-fitting. To solve this problem, BIC introduces a penalty term for the model complexity to the integrated likelihood of the model. Thus, the choice gets more reasonable and more close to the data structure.

But BIC criterion has one limitation: because it was not designed for clustering purpose when doing the model selection, if the right model is not considered as the potential model, BIC criterion will tend to overestimate the correct size [59]. For this reason, an Integrated Completed Likelihood (ICL) criterion was proposed by Biernacki et al. [21] to compensate this limitation. ICL criterion in fact is a BIC approximation of the integrated complete likelihood. The difference between BIC and ICL criteria is that

ICL criterion is based on complete likelihood and contains an additional penalty term, thus ICL prefers well-separated clusters. The corresponding consequence is that BIC criterion might over-estimate the number of mixture components whilst ICL criterion under-estimates the number of mixture components. Generally speaking, for standard data clustering, both BIC and ICL criteria are proved to be useful in choosing the model and consequently the number of the components.

Other common criteria such as AIC, AWE, and NEC criteria, are also commonly used in this domain. They are frequently mentioned for assessing mixture models and compared with BIC and ICL criteria.

We developed binned-EM and bin-EM-CEM algorithms of different parsimonious Gaussian mixture models in the Chapters 2 and 3, but no criterion is applied to model selection for binned data clustering. In this chapter, we adapt several common criteria to binned data so as to select the model and the number of mixture components for binned data clustering.

We mainly focus on BIC and ICL criteria applied to binned data clustering. In the Section 4.2, BIC and ICL criteria will be adapted to binned data clustering using binned-EM algorithm. The Section 4.3 will present the corresponding experimental results on simulated data and real data. And in the Section 4.4, we will adapt BIC and ICL criteria to cluster analysis of binned data using bin-EM-CEM algorithm. Related experiments will be presented in the Section 4.5. Then, comparison among BIC and ICL criteria applied to binned-EM and bin-EM-CEM algorithms will be presented in the Section 4.6. AIC, AWE, and NEC criteria, will be adapted to binned data clustering in the Section 4.7. A conclusion summarizing the main results will be given in the Section 4.8.

4.2 BIC and ICL criteria for binned data clustering by binned-EM algorithm

The manner to choose the best model for clustering the data in framework of binned data is the same as in the standard data framework: to choose the model which maximizes the integrated likelihood. As the likelihood increases by adding parameters, the most general model gives the biggest likelihood but it is also the most complicated model with the most parameters. In some cases it is not necessary to use a complex model when dealing with simple data structure. BIC and ICL criteria provide a penalty term of model complexity. In this section, we review the BIC and ICL criteria and adapt BIC and ICL criteria to binned data clustering.

4.2.1 Bayesian information criterion (BIC)

$$L(\boldsymbol{\Phi};\boldsymbol{a}) \approx L(\hat{\boldsymbol{\Phi}};\boldsymbol{a}) - \frac{v_{m,K}}{2} log(n)$$

where $L(\boldsymbol{\Phi};\boldsymbol{a})$ is the log-likelihood which has the form:

$$\begin{aligned} L(\boldsymbol{\Phi};\boldsymbol{a}) &= \log(p(\boldsymbol{a};\boldsymbol{\Phi})) \\ &= \sum_{r=1}^{v} n_r \log(\sum_{k=1}^{K} \pi_k \int_{\boldsymbol{\mathcal{H}}_r} f_k(\boldsymbol{x};\theta_k)\mathrm{d}\boldsymbol{x}) + \log(c) \end{aligned}$$

and $\hat{\boldsymbol{\Phi}}$ is the maximum likelihood estimate of $\boldsymbol{\Phi}$:

$$\hat{\boldsymbol{\Phi}} = \arg\max_{\boldsymbol{\Phi}} L(\boldsymbol{\Phi};\boldsymbol{a})$$

which is obtained by binned-EM algorithm [22], [23], [24]. And $v_{m,K}$ is the number of parameters to estimate in the model m with K components, which is listed in detail in the paper of Celeux and Govaert [15].

4.2.2 Integrated completed likelihood criterion (ICL)

ICL criterion was proposed by considering the integrated likelihood of the complete data [21]:

$$L(\boldsymbol{\Phi};\boldsymbol{a},\boldsymbol{z}) \approx L(\hat{\boldsymbol{\Phi}};\boldsymbol{a},\boldsymbol{z}) - \frac{v_{m,K}}{2} log(n)$$

where $L(\boldsymbol{\Phi};\boldsymbol{a},\boldsymbol{z})$ is the complete log-likelihood:

$$\begin{aligned} L(\boldsymbol{\Phi};\boldsymbol{a},\boldsymbol{z}) &= \log(p(\boldsymbol{a},\boldsymbol{z};\boldsymbol{\Phi})) \\ &= \sum_{r=1}^{v} n_r \log\left(\pi_{z_r} \int_{\boldsymbol{\mathcal{H}}_r} f_{z_r}(\boldsymbol{x};\theta_{z_r})\mathrm{d}\boldsymbol{x}\right) + \log(c) \end{aligned}$$

But in this paper, we apply ICL on binned-EM algorithm instead of bin-EM-CEM algorithm. An important detail in this paper is that: $\hat{\boldsymbol{\Phi}}$ isn't obtained by bin-EM-CEM algorithm, but the same as in BIC, which is obtained by binned-EM algorithm:

$$\hat{\boldsymbol{\Phi}} = \arg\max_{\boldsymbol{\Phi}} L(\boldsymbol{\Phi};\boldsymbol{a})$$

Since $\boldsymbol{z}$ is unknown, we replace $\boldsymbol{z}$ by $\hat{\boldsymbol{z}} = MAP(\hat{\boldsymbol{\theta}})$.

4.3 Experiments of BIC and ICL criteria with binned-EM algorithm

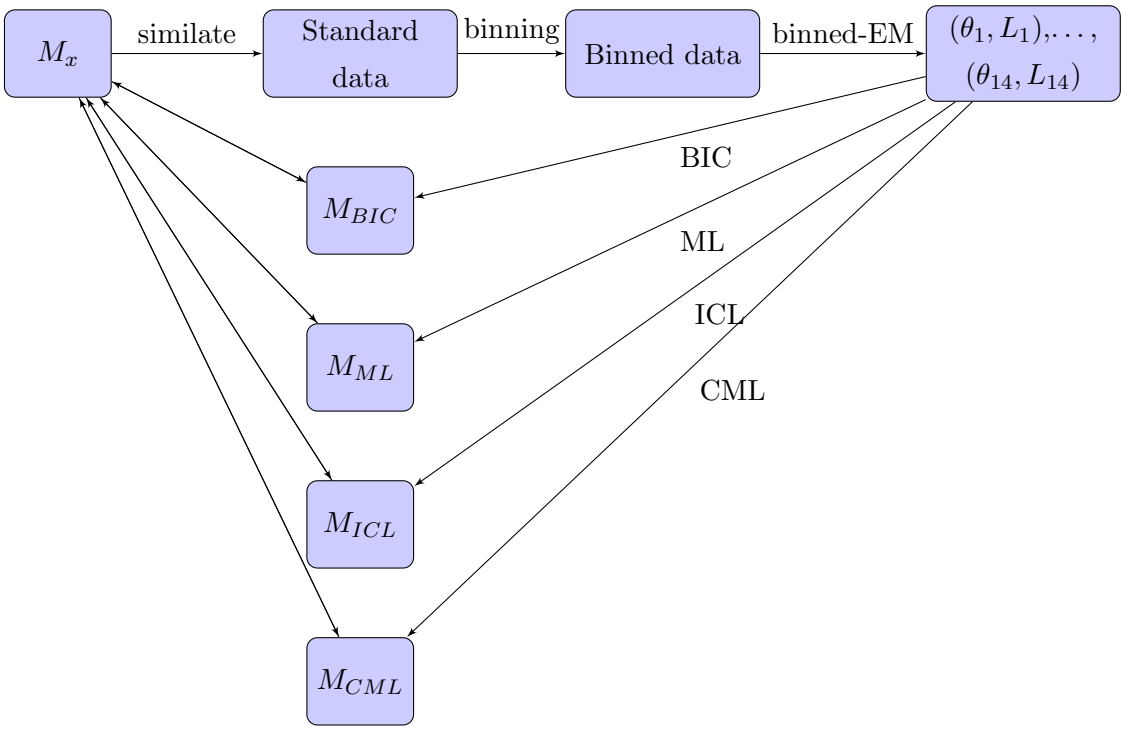
4.3.1 Experiments on simulated data

4.3.1.1 Choice of model

In this part, we compare the performance of BIC and ICL criteria in choosing the model for binned data clustering. The number of clusters is considered to be known, BIC and ICL criteria make a model choice among the fourteen parsimonious models.

As simulated data, we generate fourteen types of data according to fourteen parsimonious Gaussian mixture models with two clusters. For each type of simulated data, we generate 30 samples of size= 3000. Fourteen binned-EM algorithms of parsimonious models are applied to each sample once. The result which appears the most among 30 results will be considered as the final result for the corresponding data type. We compare the result of maximum log-likelihood estimation, maximum completed log-likelihood estimation, BIC and ICL criteria.

The process of the experiment is shown in the following flow chart:



Since the volumes, shapes and orientations are different among fourteen models, the separation level of clusters within each model is controlled and defined by the distance

value, which indicates the distance between two mixture components:

$$\delta = \sqrt{(\mu_1 - \mu_2)^T (\frac{\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2}{2})^{-1} (\mu_1 - \mu_2)}$$

To obtain binned data, we divide the space into small bins of $size = 0.5 \cdot 0.5$. Because of the different volumes and different centers of clusters in each data structure, the space is cut into different numbers of bins per dimension depending on the volumes of samples, which we will detail in the description of the characteristics of data.

The data simulated contains two components of equal mixing portions in two-dimensional space. Characteristics of each structure of data are described as follows:

- Data structure 1 is generated according to the model $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$ with $\lambda = 1$, $\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A} = Diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-1.1, 0)$, $\boldsymbol{\mu}_2 = (1.2, 0)$, $\delta = 2.97$, Number of bins$= 23 \times 17$.

- Data structure 2 is generated according to the model $[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$ with $\lambda_1 = 1$, $\lambda_2 = 3$, $\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A} = Diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-1.5, 0)$, $\boldsymbol{\mu}_2 = (1.5, 0)$, $\delta = 2.74$, Number of bins$= 29 \times 28$.

- Data structure 3 is generated according to the model $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$ with $\lambda = 1$, $\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A}_1 = Diag(1, 1)$, $\mathbf{A}_2 = Diag(2, 1/2)$, $\boldsymbol{\mu}_1 = (-1.5, 0)$, $\boldsymbol{\mu}_2 = (1.5, 0)$, $\delta = 3.0$, Number of bins$= 21 \times 15$.

- Data structure 4 is generated according to the model $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$ with $\lambda_1 = 1$, $\lambda_2 = 2$, $\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A}_1 = Diag(1, 1)$, $\mathbf{A}_2 = Diag(2, 1/2)$, $\boldsymbol{\mu}_1 = (-1.5, 0)$, $\boldsymbol{\mu}_2 = (2, 0)$, $\delta = 2.93$, Number of bins$= 22 \times 20$.

- Data structure 5 is generated according to the model $[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$ with $\lambda = 1$, $\mathbf{D}_1 = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$, $\mathbf{D}_2 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A} = Diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-2, 0)$, $\boldsymbol{\mu}_2 = (1.5, 0)$, $\delta = 3.04$, Number of bins$= 23 \times 24$.

- Data structure 6 is generated according to the model $[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$ with $\lambda_1 = 3$, $\lambda_2 = 1$, $\mathbf{D}_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{D}_2 = Diag(1, 1)$, $\mathbf{A} = Diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-5, 0)$, $\boldsymbol{\mu}_2 = (5, 0)$, $\delta = 3.79$, Number of bins$= 47 \times 29$.

- Data structure 7 is generated according to the model $[\lambda \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ with $\lambda = 1$, $\mathbf{D}_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{D}_2 = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$, $\mathbf{A}_1 = Diag(3, 1/3)$, $\mathbf{A}_2 = Diag(2, 1/2)$, $\boldsymbol{\mu}_1 = (-1.4, 1)$, $\boldsymbol{\mu}_2 = (1.5, -1)$, $\delta = 3.09$, Number of bins= 19×20.

- Data structure 8 is generated according to the model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ with $\lambda_1 = 2, \lambda_2 = 1$, $\mathbf{D}_1 = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{D}_2 = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}$, $\mathbf{A}_1 = Diag(2, 1/2)$, $\mathbf{A}_2 = Diag(3, 1/3)$, $\boldsymbol{\mu}_1 = (-1.8, 1)$ $\boldsymbol{\mu}_2$=(1.7,-1), $\delta = 3.05$, Number of bins= 24×27.

- Data structure 9 is generated according to the model $[\lambda \mathbf{B}]$ with $\lambda = 1$, $\mathbf{B} = Diag(1/2, 2)$, $\boldsymbol{\mu}_1 = (-1, 0)$, $\boldsymbol{\mu}_2 = (1.1, 0)$, $\delta = 2.97$, Number of bins= 13×20.

- Data structure 10 is generated according to the model $[\lambda_k \mathbf{B}]$ with $\lambda_1 = 1$, $\lambda_2 = 3$, $\mathbf{B} = Diag(1/3, 3)$, $\boldsymbol{\mu}_1 = (-4, 0)$, $\boldsymbol{\mu}_2 = (4, 0)$, $\delta = 3.27$, Number of bins= 48×12.

- Data structure 11 is generated according to the model $[\lambda \mathbf{B}_k]$ with $\lambda = 1$, $\mathbf{B}_1 = Diag(1/2, 2)$, $\mathbf{B}_2 = Diag(1/3, 3)$, $\boldsymbol{\mu}_1 = (-1, 0)$, $\boldsymbol{\mu}_2 = (1, 0)$, $\delta = 3.09$, Number of bins= 18×24.

- Data structure 12 is generated according to the model $[\lambda_k \mathbf{B}_k]$ with $\lambda_1 = 1$, $\lambda_2 = 3$, $\mathbf{B}_1 = Diag(2, 1/2)$, $\mathbf{B}_2 = Diag(4, 1/4)$, $\boldsymbol{\mu}_1 = (-3, 0)$, $\boldsymbol{\mu}_2 = (3, 0)$, $\delta = 3.03$, Number of bins= 42×11.

- Data structure 13 is generated according to the model $[\lambda \mathbf{I}]$ with $\lambda_1 = 1$, $\boldsymbol{\mu}_1 = (-1.5, 0)$, $\boldsymbol{\mu}_2 = (1.5, 0)$, $\delta = 2.97$, Number of bins= 27×18.

- Data structure 14 is generated according to the model $[\lambda_k \mathbf{I}]$ with $\lambda_1 = 1$, $\lambda_2 = 3$, $\boldsymbol{\mu}_1 = (-2.1, 0)$, $\boldsymbol{\mu}_2 = (2.1, 0)$, $\delta = 2.97$, Number of bins= 28×22.

The model choice result is shown in the Tables 4.1 and 4.2.

According to the result of maximum log-likelihood, the most complex model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ is the best model for 10 data types: $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$, $[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$, $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, $[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$, $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$, $[\lambda \mathbf{B}]$, $[\lambda_k \mathbf{B}]$, $[\lambda \mathbf{B}_k]$, $[\lambda \mathbf{I}]$, $[\lambda_k \mathbf{I}]$. This result accords to the theory that the most complex model gives the highest likelihood. For the data type of $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$ and $[\lambda_k \mathbf{B}_k]$, the biggest likelihood is obtained by its own feature model $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$ and $[\lambda_k \mathbf{B}_k]$ respectively. For the data type of $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, the biggest likelihood comes from the result of model $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, which is a little more complicated than the data structure.

Comparing to the maximum log-likelihood, BIC criterion succeeds in choosing the right model which is exactly the same as the data structure for eight data types: $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$, $[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$, $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, $[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$, $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$, $[\lambda_k \mathbf{B}]$, $[\lambda_k \mathbf{B}_k]$ and $[\lambda_k \mathbf{I}]$. For data

Data type \ criteria	BIC	ML
$[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$	$[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]*$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$	$[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]*$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	$[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]*$	$[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]*$
$[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$
$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]*$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k \mathbf{B}_k]$	$[\lambda_k \mathbf{B}_k]$
$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$
$[\lambda \mathbf{B}]$	$[\lambda_k \mathbf{B}_k]$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda_k \mathbf{B}]$	$[\lambda_k \mathbf{B}]*$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda \mathbf{B}_k]$	$[\lambda_k \mathbf{B}_k]$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda_k \mathbf{B}_k]$	$[\lambda_k \mathbf{B}_k]*$	$[\lambda_k \mathbf{B}_k]*$
$[\lambda \mathbf{I}]$	$[\lambda_k \mathbf{I}]$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda_k \mathbf{I}]$	$[\lambda_k \mathbf{I}]*$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$

TABLE 4.1: Result of BIC criterion and maximum log-likelihood estimation with binned-EM algorithm. (* indicates the correct model)

Model Structure \ criteria	ICL	CML
$[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$	$[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]*$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$	$[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]*$	$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$
$[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	$[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]*$	$[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]*$
$[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$
$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]*$	$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]*$
$[\lambda \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k \mathbf{B}_k]$	$[\lambda_k \mathbf{B}_k]$
$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$
$[\lambda \mathbf{B}]$	$[\lambda_k \mathbf{B}]$	$[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$
$[\lambda_k \mathbf{B}]$	$[\lambda_k \mathbf{B}]*$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda \mathbf{B}_k]$	$[\lambda_k \mathbf{B}_k]$	$[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda_k \mathbf{B}_k]$	$[\lambda_k \mathbf{B}_k]*$	$[\lambda_k \mathbf{B}_k]*$
$[\lambda \mathbf{I}]$	$[\lambda_k \mathbf{I}]$	$[\lambda_k \mathbf{I}]$
$[\lambda_k \mathbf{I}]$	$[\lambda_k \mathbf{I}]*$	$[\lambda_k \mathbf{I}]*$

TABLE 4.2: Result of ICL criterion and complete maximum log-likelihood estimation with binned-EM algorithm. (* indicates the correct model)

type of $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, $[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$ and $[\lambda \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$, BIC has the same choice as the maximum log-likelihood. BIC tends to choose the most complex model of diagonal family $[\lambda_k \mathbf{B}_k]$ for two data type of diagonal family: $[\lambda \mathbf{B}]$, $[\lambda \mathbf{B}_k]$. For the simplest data structure of $[\lambda \mathbf{I}]$, BIC prefers model $[\lambda_k \mathbf{I}]$.

For the result of maximum completed log-likelihood, the most complex model $[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$ gives the biggest completed log-likelihood for six types of data: $[\lambda \mathbf{D}\mathbf{A}\mathbf{D}^T]$, $[\lambda_k \mathbf{D}\mathbf{A}\mathbf{D}^T]$, $[\lambda_k \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, $[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$, $[\lambda_k \mathbf{B}]$ and $[\lambda \mathbf{B}_k]$. The biggest completed log-likelihood is obtained by the right model for the following four data types: $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$, $[\lambda_k \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$, $[\lambda_k \mathbf{B}_k]$ and $[\lambda_k \mathbf{I}]$. For the data type of $[\lambda \mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$, $[\lambda \mathbf{B}]$ and $[\lambda \mathbf{I}]$, the highest completed log-likelihood is given by a model which is more complex than the data structure.

ICL criterion has the almost same result as BIC criterion except that it chooses the model $[\lambda_k \mathbf{B}]$ for the data type of $[\lambda \mathbf{B}]$.

We can conclude as follows: Both BIC and ICL are able to choose the right model for most of the types of data. The result also shows that BIC and ICL are helpful in selecting a more model which can obtain a good clustering result and more simplified compared to the most complex model $[\lambda_k \mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$. Generally speaking, for the data generated according to the general family, BIC and ICL criteria favor the models from general family. And it happens the same to diagonal family and spherical family. The result also shows that BIC and ICL criteria have almost the same behaviors in the context that the number of clusters is defined in advance. It is logical since the main difference between BIC and ICL criteria is that BIC criterion temps to over-estimate the number of clusters. Meanwhile ICL temps to select reasonably less number of clusters for the clustering objective.

4.3.1.2 Choice of number of clusters

This part of experiment aims to study how BIC and ICL choose the model especially the number of clusters. 30 datasets are simulated according the model $[\lambda \mathbf{D}\mathbf{A}_k\mathbf{D}^T]$ with two clusters. The parameters are defined as: $\mathbf{D} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$, $\mathbf{A}_1 = Diag(1,1)$, $\mathbf{A}_2 = Diag(2,1/2)$, $\boldsymbol{\mu}_1 = (-1.5,0)$, $\boldsymbol{\mu}_2 = (1.5,0)$. 14 binned-EM algorithms of 14 models are applied on each dataset. The size of bins is defined as $50 \cdot 50$. The potential number of data are 1, 2 and 3. The model selected is the model with a number of clusters which leads to the highest BIC and ICL result. Table 4.3 presents the model choice of BIC and ICL criteria, their corresponding approximation and CPUtime.

From the result, when the number of clusters is assumed to be 2, both BIC and ICL criteria can select the right model. But when the number of clusters are 1 or 3, BIC

Clusters Num.		BIC	ICL	Time
1	Model	$[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$	12s
	Result	-736.6	-736.6	
2	Model	$[\lambda \mathbf{D} \mathbf{A}_k \mathbf{D}^T]$	$[\lambda \mathbf{D} \mathbf{A}_k \mathbf{D}^T]$	13s
	Result	-465.1	-681.93	
3	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$	28s
	Result	-533.7	-1063.2	

TABLE 4.3: Model choice and choice of number of clusters of BIC and ICL criteria.

and ICL criteria cannot make the right choice. They either choose a wrong model $[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$, either choose the most complex model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ which is much more complex than the right model $[\lambda \mathbf{D} \mathbf{A}_k \mathbf{D}^T]$. Since the model $[\lambda \mathbf{D} \mathbf{A}_k \mathbf{D}^T]$ with 2 clusters provides the highest BIC and ICL approximation, we can say that if the number of clusters is unknown, BIC and ICL criteria are able to choose the right model with the correct number of clusters.

4.3.2 Experiments on real data

4.3.2.1 French city clustering

BIC and ICL criteria adapted to binned data clustering seem to perform well on simulated data. To further test these two criteria on real dataset, we consider a set of population and population density of 1199 French cities. These French cities belong to three departments: Meuse (500 observations), Nord (652 observations) and Val-de-Marne (47 observations). According to different location of three departments, these cities have different characteristic of population, as what is shown in Figure 4.1. Meuse is a rural department with a small population and low population density; in the opposite, Val-de-Marne, situated to southeast of Paris, is a department in France; Nord is the most populous department in France. Fourteen binned-EM algorithms of parsimonious Gaussian mixture models with different number of clusters (2, 3 and 4) are applied to the real data. BIC and ICL choose the right model as well as the number of clusters. Each algorithm are executed 30 times. The most selected model and number of clusters are considered as the final choice. The result of BIC and ICL criteria are compared with the result of maximum log-likelihood and maximum complete log-likelihood. The result is shown in Table 4.4.

From the result, BIC and ML criteria have chosen three as the number of cluster, while ICL and CML criteria have preferred two clusters. BIC criterion chooses the combination of model $[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ with 3 clusters. And ML obtains the best result with the most complex model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$. ICL criterion has a different choice from BIC criterion. It

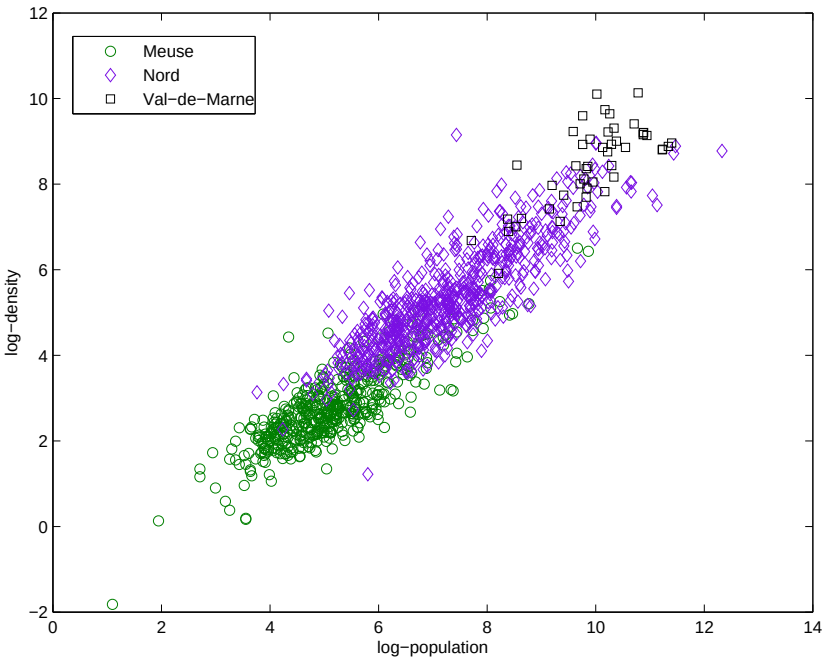


FIGURE 4.1: Log-population and log-density of 1199 cities from three departments in France.

Clusters Num.		2	3	4
BIC	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$*	$[\lambda \mathbf{D} \mathbf{A} \mathbf{D}^T]$
	Result	-677.9	**-672.6**	-692.3
ML	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$*	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$
	Result	-638.9	**-612.9**	-628.6
ICL	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$*	$[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$
	Result	**-818.2**	-883.9	-968.7
CML	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$*	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda \mathbf{D} \mathbf{A} \mathbf{D}^T]$
	Result	**-779.3**	-823.7	-948.4

TABLE 4.4: Model choice and choice of number of clusters of BIC and ICL criteria for real dataset.

selects the most complex model also $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ with only 2 clusters. CML provides the same choice. From the accuracy point of view, BIC has made a good choice. It has selected a model which represents the data structure as well as a correct number of clusters. But from the clustering point of view, ICL performs properly.

4.3.2.2 Image segmentation

In this part, we will apply BIC and ICL criteria to image segmentation by binned-EM algorithms of parsimonious Gaussian mixture models. This experiment aims to study if BIC and ICL criteria can provide the right number of colors and find out the distribution of the colors of the image.

The image to be processed is the same as the one in the experiment part of Chapter 2, which is shown in the Figure 4.2:

FIGURE 4.2: Original image.

After converting the image into $L*a*b*$ color space. Since the color information exists in the $'a*b*'$ space, the data become pixels with $'a*'$ and $'b*'$ values. The Figure 4.3 shows the 154401 image pixels:

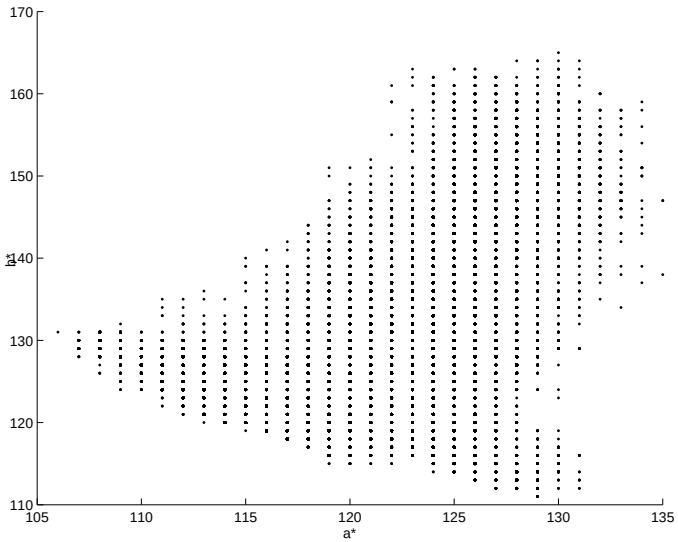


FIGURE 4.3: Image pixel represented in the $'a*b*'$ space.

From the Figure 4.3, it is difficult to tell the number of clusters and the distribution of the clusters. Thus we apply the binned-EM algorithms of fourteen parsimonious Gaussian mixture models to this dataset. The potential numbers of clusters are among 3, 4 and 5. The BIC and ICL criteria are applied to choose the model and the number of clusters. To obtain binned data, the space is divided into 20 bins per dimension.

Clusters Num.		BIC	ICL
3	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$
	Result	-5.97×10^4	-7.13×10^4
4	Model	$[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$
	Result	-5.75×10^4	-6.79×10^4
5	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$
	Result	-4.16×10^4	-5.62×10^4
6	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$
	Result	-3.97×10^4	-5.61×10^4
7	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$
	Result	-3.18×10^4	-5.15×10^4
8	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$
	Result	-3.03×10^4	-5.25×10^4

TABLE 4.5: The choice of model and number of clusters of BIC and ICL criteria with binned-EM algorithm for image segmentation of Figure 4.2.

From the Table 4.5, the result of BIC criterion increases when the number of clusters increases. The result of ICL criterion increases when the number of clusters increases until 7. There are many colors in the image (more than 7 colors). These colors look the similar but actually different. More clusters means a finer grouping of colors. Thus more clusters approaches closer the real clustering of colors, so as a better image segmentation. when the number of clusters is 4, both of BIC and ICL criteria have chosen the model $[\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$. Except for this case, both of BIC and ICL criteria have chosen the model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ for the other numbers of clusters. The image segmentation with different number of cluster is shown in the Figure 4.4:

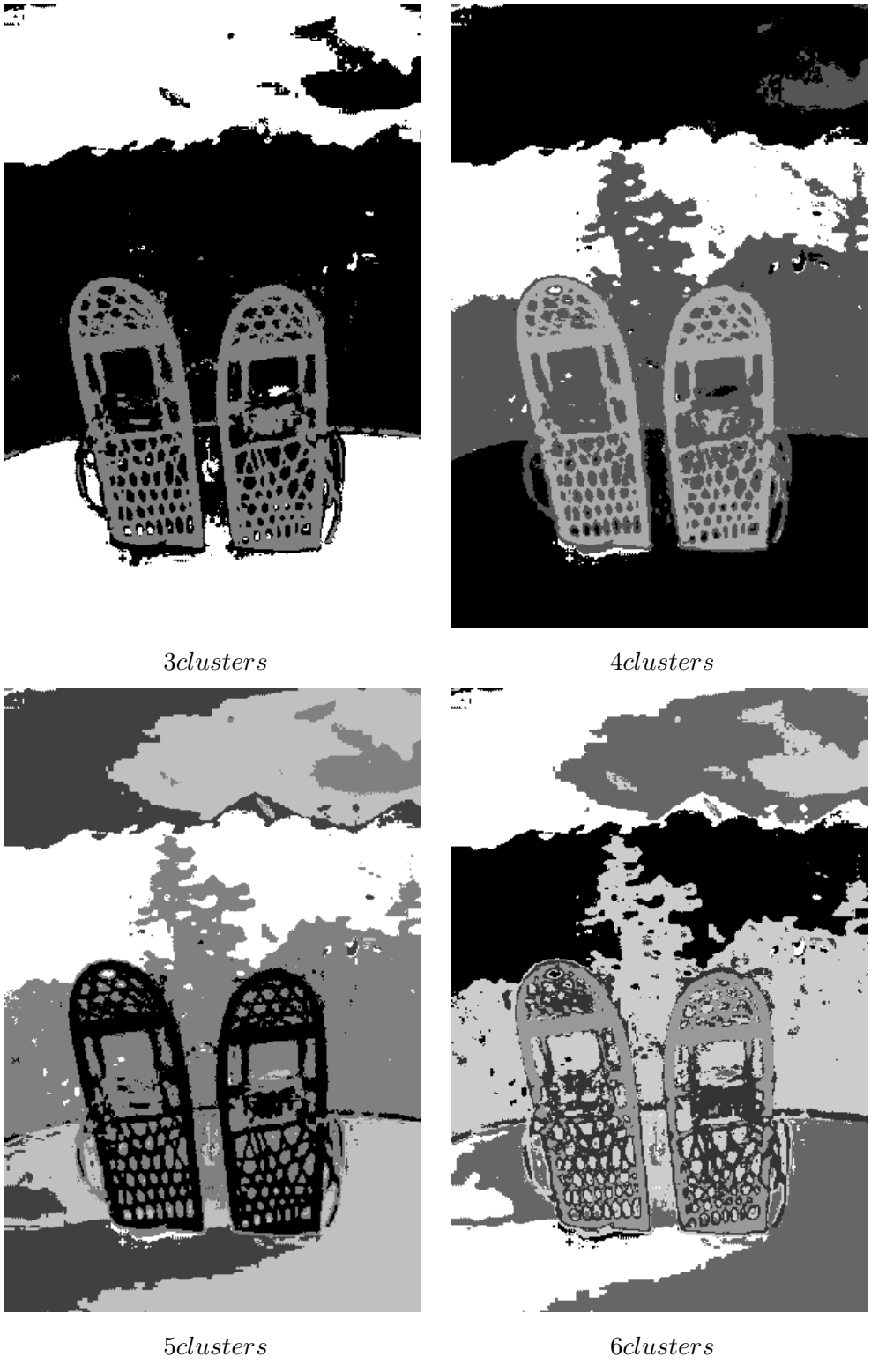


FIGURE 4.4: Result of image segmentation by binned-EM algorithm with different number of clusters.

4.4 BIC and ICL criteria for binned data clustering by bin-EM-CEM algorithm

4.4.1 Bayesian information criterion (BIC)

BIC criterion is to introduce a penalty term depending on the model complexity to maximum value of the log-likelihood, so as to find the best model which provide high integrated likelihood with respectively reasonable parameters.

$$L(\boldsymbol{\Phi};\boldsymbol{a}) \approx L(\hat{\boldsymbol{\Phi}},\boldsymbol{a}) - \frac{\upsilon_{m,K}}{2} log(n)$$

where $\hat{\boldsymbol{\Phi}}$ is the maximum likelihood estimate of $\boldsymbol{\Phi}$, which is usually obtained by binned-EM algorithm, but in this article, $\hat{\boldsymbol{\Phi}}$ is obtained by bin-EM-CEM algorithm:

$$\hat{\boldsymbol{\Phi}} = \arg\max_{\boldsymbol{\Phi}} L(\boldsymbol{\Phi};\boldsymbol{a},\boldsymbol{z})$$

$\upsilon_{m,K}$ is the number of parameters to estimate in the model m with K components. The corresponding information can be found in the paper of Celeux and Govaert [15].

4.4.2 Integrated completed likelihood criterion (ICL)

ICL criterion was proposed by considering the integrated likelihood of the complete data [21]:

$$L(\boldsymbol{\Phi};\boldsymbol{a},\boldsymbol{z}) \approx L(\hat{\boldsymbol{\Phi}};\boldsymbol{a},\boldsymbol{z}) - \frac{\upsilon_{m,K}}{2} log(n)$$

where $\hat{\boldsymbol{\Phi}}$ is obtained by bin-EM-CEM algorithm:

$$\hat{\boldsymbol{\Phi}} = \arg\max_{\boldsymbol{\Phi}} L(\boldsymbol{\Phi};\boldsymbol{a},\boldsymbol{z})$$

4.5 Experiments of BIC and ICL criteria with bin-EM-CEM algorithm

4.5.1 Experiments on simulated data

In this section, we compare the behavior of BIC and ICL criteria in three parts. The first part, we apply BIC and ICL criteria to two simulated data with different overlapping rates. This experiment is very similar to the first experiment of the article of Biernacki

[21] in order to have a comparison with BIC and ICL applied to standard data clustering model selection. In the second experiment, we increase the data size from 400 to 2000 and 3600. Decision of BIC and ICL criteria on different amounts of data will be studied. In the third part, we will study how the performances of BIC and ICL criteria change following with different size of bins.

4.5.1.1 Different overlappings

In the first part, we will compare the choice of BIC and ICL in model selection of binned data clustering. In order to compare with the BIC and ICL behaviors on standard data, we generate two types of data similar to the simulated data in the experiment of Biernacki's paper [21].

The first type of data consists of three Gaussian components in a two dimensional space. Three components have the same volumes and the same shapes. The orientations of the first component and the third component are the same. The orientation of the seconde component is different. The parameters of the Gaussian mixture are:

$$n = 400, p_1 = 0.25, p_2 = 0.25, p_3 = 0.5$$

$$\mu_1 = \mu_2 = (0, 0), \mu_3 = (8, 0)$$

$$\lambda_1 = \lambda_2 = \lambda_3 = 1$$

$$\mathbf{A}_1 = \mathbf{A}_2 = \mathbf{A}_3 = diag(10, 1/10)$$

$$\mathbf{D}_1 = \mathbf{D}_3 = \begin{pmatrix} \cos(\pi/2) & \sin(\pi/2) \\ -\sin(\pi/2) & \cos(\pi/2) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

$$\mathbf{D}_2 = \begin{pmatrix} \cos(\frac{5}{12}\pi) & \sin(\frac{5}{12}\pi) \\ -\sin(\frac{5}{12}\pi) & \cos(\frac{5}{12}\pi) \end{pmatrix} = \begin{pmatrix} 0.9659 & 0.2588 \\ -0.2588 & 0.9659 \end{pmatrix}$$

So the angle between the orientations of the first component and the seconde component is 15 degrees.

The seconde type of data have the same parameters as the first type data except for the orientation of the seconde cluster:

$$\mathbf{D}_2 = \begin{pmatrix} \cos(\pi/3) & \sin(\pi/3) \\ -\sin(\pi/3) & \cos(\pi/3) \end{pmatrix} = \begin{pmatrix} 0.5 & 0.866 \\ -0.866 & 0.5 \end{pmatrix}$$

In this case, the angle between the orientations of the first component and the seconde component is 30 degrees.

For each type of data, we generate 30 samples. Fourteen bin-EM-CEM algorithms of fourteen parsimonious models are applied on each samples with random initial parameters. The potential models are fourteen parsimonious mixture models with 2, 3 or 4 components. The best result of BIC and ICL among 30 samples is considered as the final result of each data type respectively.

For the first type of data, the BIC criterion has chosen the right model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ with 3 clusters. And the ICL criterion prefer the most complex model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ with 2 clusters.

For the seconde type of data, both the BIC and ICL criteria chose the right model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ with 3 clusters.

BIC and ICL are able to detect the right model for binned data clustering. ICL tends to choose fewer number of components in order to obtain a more reasonable clustering result. BIC and ICL criteria applied to binned data perform similarly as applying to standard data.

4.5.1.2 Different amounts of data

The seconde experiment aims to study how would the result of BIC and ICL criteria would change when the amount of data gets bigger. Besides the result from the first experiment, in this experiment, we consider two more data amounts: 2000 and 3600. The data are simulated according to the parameters same of the first type of data in the first experiment. The size of bin is still fixed at 40 bins per dimension. For each amount of data, we simulate 30 samples. Then fourteen parsimonious models are applied on each sample. The one with the best BIC and ICL result among the 30 samples is considered as the final result. The result of BIC and ICL criteria (including the model choice and the choice of number of clusters) and the CPU time of the whole clustering process are shown in the Table 4.6. The CPU time indicates the amount of time for which a central processing unit (CPU) was used for processing instructions of a computer program.

Data amount	BIC choice	ICL choice	CPUtime
400	$[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T](3)$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T](2)$	13.3s
2000	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T](2)$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T](2)$	16.4s
3600	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T](2)$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T](2)$	19.6s

TABLE 4.6: Model choice(number of clusters in the parenthesis) by BIC and ICL and CPU time of binned data clustering on different amount of dataset.

As mentioned earlier in the paper, when dealing with data of 400 observations, the BIC criterion succeeds in choosing the right model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ with the right number of clusters 3 while the ICL prefers a more complex model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ with only 2 clusters. When the amount of data increases to 2000 or 3600, both the BIC and the ICL criteria have chosen the most complex model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ with 2 clusters as the first choice. It seems that with the same size of bins, the increase of the number of observations doesn't help in choosing the right model for binned data clustering. We should mention that except for the model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ with 2 clusters, both BIC and ICL criteria consider the right model $[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T]$ with 3 clusters as seconde choice.

Considering the aspect of computation time, the CPU time increases slower than the size of data. With the increase of number of data from 400, to 2000 until 3600, there is only about $3s$ more of computation time at each increase. We can say that data binning has slow down the increasing tendency of time along with the increase of data.

4.5.1.3 Different sizes of bin

The size of bins is a very important factor in binned data clustering which can affect the result of clustering as well as model selection. In this part, except for the size of bins and the amount of data, other parameters of simulated data are the same as the seconde experiment. The size of bins changes from 30 to 40 and 50 bins per dimension. The result is shown in Table 4.7 From the result, when the size of bins is too big(30

Size of bins	BIC choice	ICL choice	CPUtime
30	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T](2)$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T](2)$	21.1s
40	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T](2)$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T](2)$	16.4s
50	$[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T](3)$	$[\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T](3)$	19.6s

TABLE 4.7: Model choice(number of clusters in the parenthesis) by BIC and ICL and CPU time of binned data clustering with different size of bins.

bins per dimension), BIC and ICL criteria cannot succeed in detecting the right data structure. And moreover, it takes more computation time than the case of 40 bins per dimension. Because bigger bins lead to more difficulties for bin-EM-CEM algorithm to obtain model parameters. When the bin is smaller(50 bins per dimension), BIC and ICL criteria are able to choose the right model for dataset. It is normal that it takes more time than 40 bins per dimension, because smaller bins means more bins to deal with.

4.5.2 Experiments on real data

4.5.2.1 French city clustering

As real dataset, we have the population and the population density (population/surface) of 1199 French cities. They are respectively from three different departments: Meuse (500 cities), Nord (652 cities) and Val-de-Marne (47 cities). Meuse is a rural department with a small population and low population density. In the opposite, Val-de-Marne, situated to southeast of Paris, is a department of high population density and Nord is the most populous department in France. The true partition of these French city is displayed in the Figure 4.5. On this dataset, fourteen parsimonious models with $K = 2, 3, 4$ are considered as potential models. The biggest value of BIC and ICL among all the potential models is considered as the result of BIC and ICL criteria. The clustering result is displayed in the Figure 4.6. BIC criterion has chosen the model

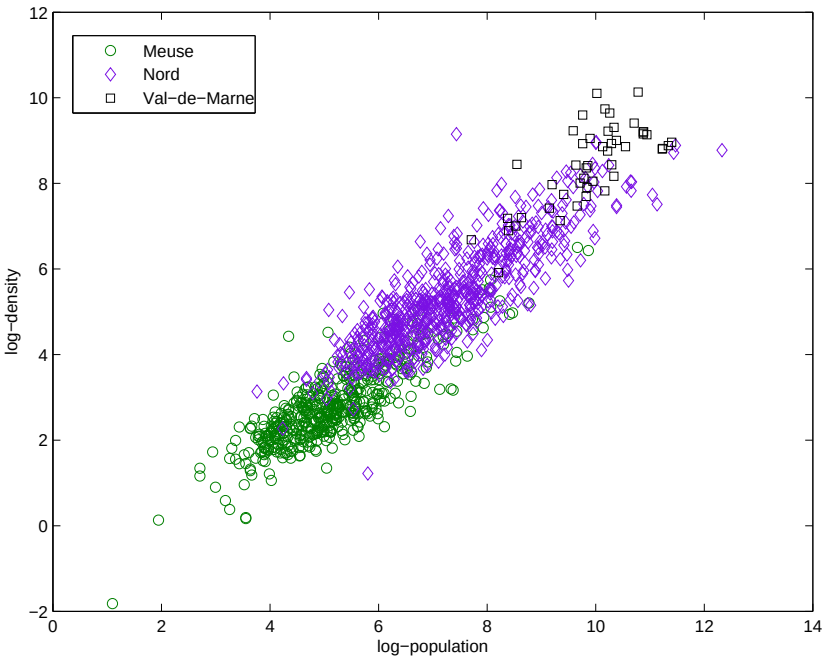


FIGURE 4.5: True clustering of 1199 cities from three departments in France.

$[\lambda_k \mathbf{DAD}^T]$ with 3 clusters, and ICL criterion favors a simpler model $[\lambda_k \mathbf{DAD}^T]$ with 2 clusters. It makes sense from the structure of the dataset. From the Figure 4.5, three clusters have the similar orientations and shapes. The volumes are different, especially the population of Val-de-Marne is much smaller than the other two departments. So ICL criterion has the tendency to put the Nord and Val-de-Marne together as one cluster.

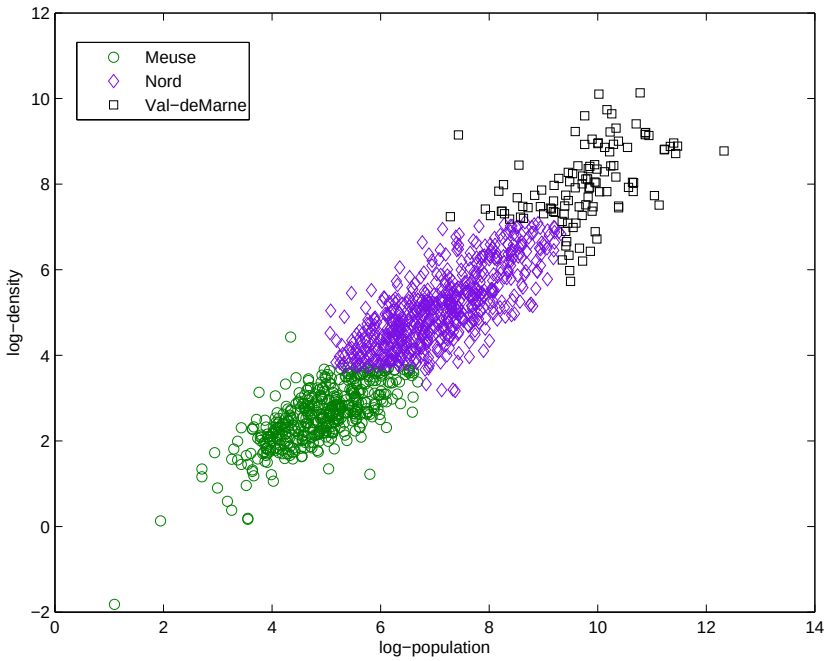


FIGURE 4.6: Result of bin-EM-CEM algorithm with BIC of 1199 cities from three departments in France.

4.5.2.2 Image segmentation

The goal of this experiment is to study how the BIC and ICL criteria perform on the application of image segmentation by bin-EM-CEM algorithm. We still use the lantern image which was used in the Chapter 3. This image is shown again in the Figure 4.7:

FIGURE 4.7: Original image.

The size of bin cannot be defined by BIC and ICL criteria. So defining the bin size is not our objective in this experiment. We suppose that the space is always divided into 20 bins per dimension. The fourteen bin-EM-CEM algorithms of parsimonious models will be applied to the dataset which represents the color of the image pixels. In the

bin-EM-CEM algorithm, the number of clusters are supposed to be 4, 5, 6, 7, 8 and 9. So BIC and ICL criteria will choose the model with the number of cluster which suppose to be the best for the dataset. The result is shown in the Table 4.8:

Clusters Num.		BIC	ICL	L	CL	CPUtime
4	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	20
	Result	-3.36×10^5	-3.50×10^5	-3.49×10^5	-3.36×10^5	
5	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	22
	Result	-2.92×10^5	-3.02×10^5	-2.78×10^5	-2.92×10^5	
6	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	26
	Result	-5.91×10^4	-7.13×10^4	-5.88×10^4	-7.11×10^4	
7	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	31
	Result	-5.16×10^4	-6.65×10^4	-5.13×10^4	-6.63×10^4	
8	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	35
	Result	-4.07×10^4	-5.59×10^4	-4.04×10^4	-5.56×10^4	
9	Model	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	$[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$	40
	Result	-4.01×10^4	-5.65×10^4	-3.55×10^4	-5.44×10^4	

TABLE 4.8: The choice of model and number of clusters of BIC and ICL criteria with bin-EM-CEM algorithm for image segmentation of Figure 4.2.

From the Table 4.8, the result of BIC and maximum likelihood increase along with the increase of the number of clusters. So the choice of BIC criterion so far is the model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ with 9 clusters. At another side, the maximum complete likelihood increases when the number of clusters gets bigger. But ICL criterion prefers the model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ with 8 clusters instead of 9 clusters. And the CPUtime increases when more clusters are considered.

We can analyze this result as follows. There are many colors in the Figure 4.2. Many colors are similar, but they are still different colors. From the result, 9 clusters seem not to be enough to represent the colors in the image. That's why the maximum likelihood, the BIC and the maximum complete likelihood increase along with the increase of the number of clusters. More clusters lead to a more precise clustering, which might be closer to the reality. But in some situations, we don't need to separate the colors in such details. According to the result of ICL criterion, 8 is an acceptable number of clusters for this dataset. 8 color clusters are enough to express the information of this image. In this case, maybe we can say that the BIC criterion overestimates the number of clusters. Or maybe the ICL criterion simplifies the clustering result. Which criterion is better depending on our needs and goals.

Among those 6 results with different bin size, we will show four of them in the Figure 4.8 to give an example.

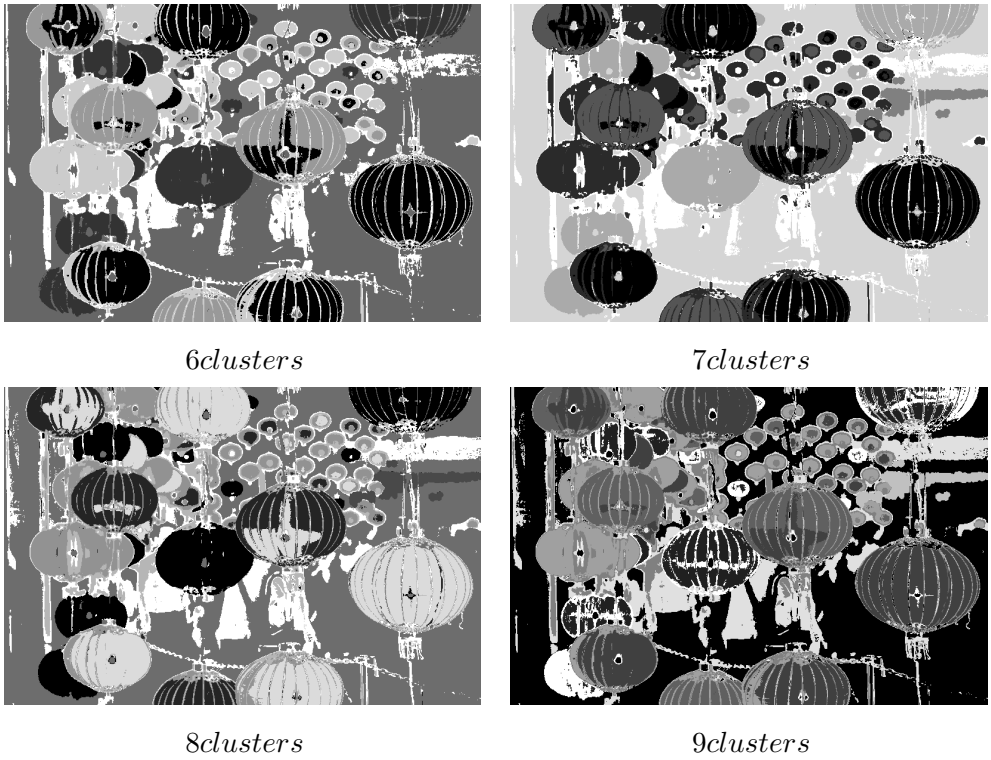


FIGURE 4.8: Result of image segmentation of Figure 4.7 by bin-EM-CEM algorithm with different number of clusters.

Comparing the four image segmentation results in the Figure 4.8, we can see that more details are shown when the number of clusters is higher. For example, the frame of each lantern is clearly shown in the last sub-figure with 9 clusters. And of course, when we suppose less number of clusters, some small information are lost. For example, when the number of clusters is 6, the blue lights at the right-up corner of the image is not shown in the result. Less number of clusters leads to a clearer image segmentation result, while higher number of clusters leads to a more complex result.

To analyze the model choice of BIC and ICL criteria, the clustering results by bin-EM-CEM algorithm of model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ with 6, 7, 8 and 9 clusters are shown in the Figure 4.9:

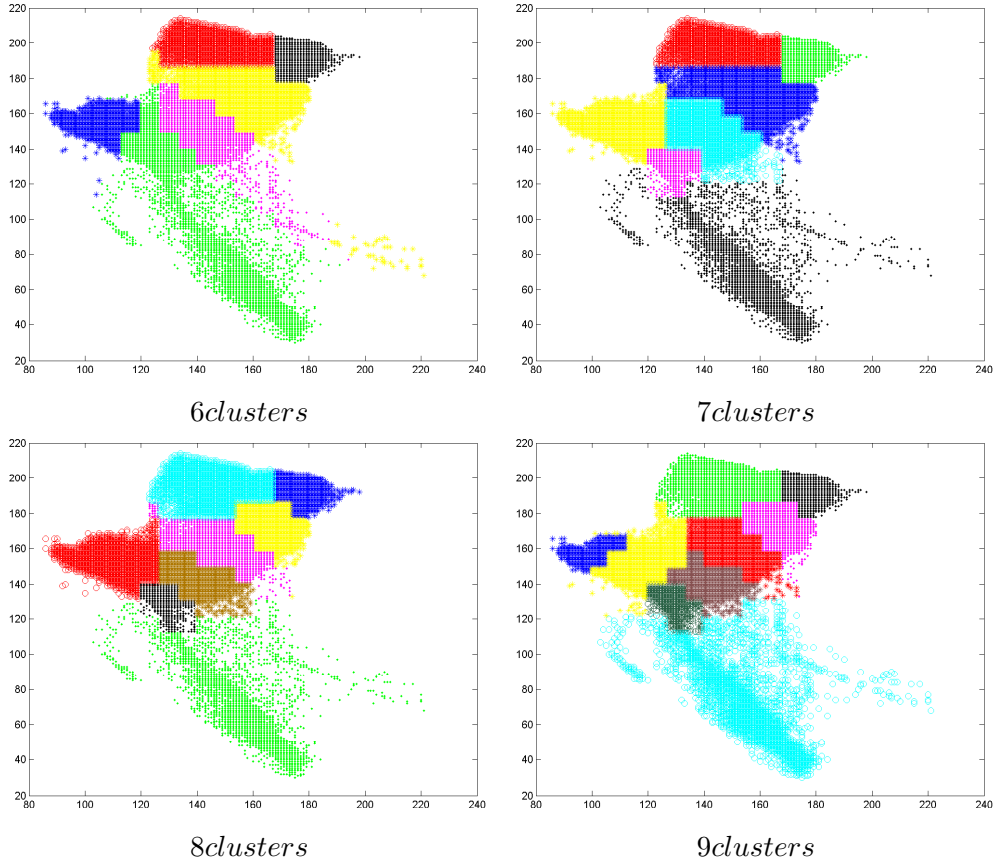


FIGURE 4.9: Clustering result of the pixels color by bin-EM-CEM algorithm with different number of clusters.

From the first sub-figure in the Figure 4.9, we can see that each cluster has different volume. Some clusters have diagonal orientation and some of them are of general orientation. Different shapes exist, where includes spherical and general shapes. The model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ can be considered as the best model for this dataset. When the number of cluster increases, some clusters are divided into two clusters. Thus the volumes of some clusters tend to be similar and the shapes of some clusters tend to be spherical. For this reason, the result of bin-EM-CEM algorithm of model $[\lambda_k \mathbf{I}]$ is very similar to the result of model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$. But BIC and ICL criteria prefer the model $[\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T]$ because this model can express the clustering result more precisely.

4.6 Comparison among BIC and ICL criteria of binned-EM and bin-EM-CEM algorithms

In this chapter, by adapting BIC and ICL criteria to the binned-EM and bin-EM-CEM algorithms, we developed four new criteria: BIC criterion with binned-EM algorithm,

ICL criterion with binned-EM algorithm, BIC criterion with bin-EM-CEM algorithm and ICL criterion with bin-EM-CEM algorithm. It is of interest to make a comparison among these four criteria.

In this experiment, we simulate three different datasets. Each time we simulate a Gaussian mixture with five components. The objective is to observe the ability of finding the number of clusters of the four criteria. Thus each component follows the simplest spherical model. The difference among three datasets is the location of the one component which leads to different distance between two components. It aims to see if our four criteria can correctly tell the number of clusters with different mixed levels. Here we give out the common model parameters of the three simulated datasets:

$$n = 5000, K = 5, p_1 = p_2 = p_3 = p_4 = p_5 = 0.2$$

$$\mu_1 = (0, 5), \mu_2 = (5, 0), \mu_3 = (5, 10), \mu_4 = (9, 5),$$

$$\Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_4 = \Sigma_5 = diag(1, 1)$$

The distance between two clusters is defined by value δ:

$$\delta = \sqrt{(\mu_1 - \mu_2)^T (\frac{\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2}{2})^{-1} (\mu_1 - \mu_2)}$$

For the first situation, two clusters are well mixed together:

$$\delta = 1, \mu_5 = (10, 5)$$

In the second situation, the distance between the two clusters is greater than in the first situation:

$$\delta = 1.5, \mu_5 = (10.5, 5)$$

We have the best-separated mixture in the third situation:

$$\delta = 2, \mu_5 = (11, 5)$$

These three situations of datasets are shown in the Figures 4.10, 4.11 and 4.12.

For each situation, we estimate 30 samples. We apply binned-EM algorithm and bin-EM-CEM algorithm on each sample and then decide the model choice (including the number clusters) by BIC and ICL criteria. The size of bins in binned-EM and bin-EM-CEM algorithms is unified in the whole experiment. The space is divided into 40×40 bins. For each criterion, the model which is chosen the most is considered as the final choice of the criterion. The result is presented in the Table.

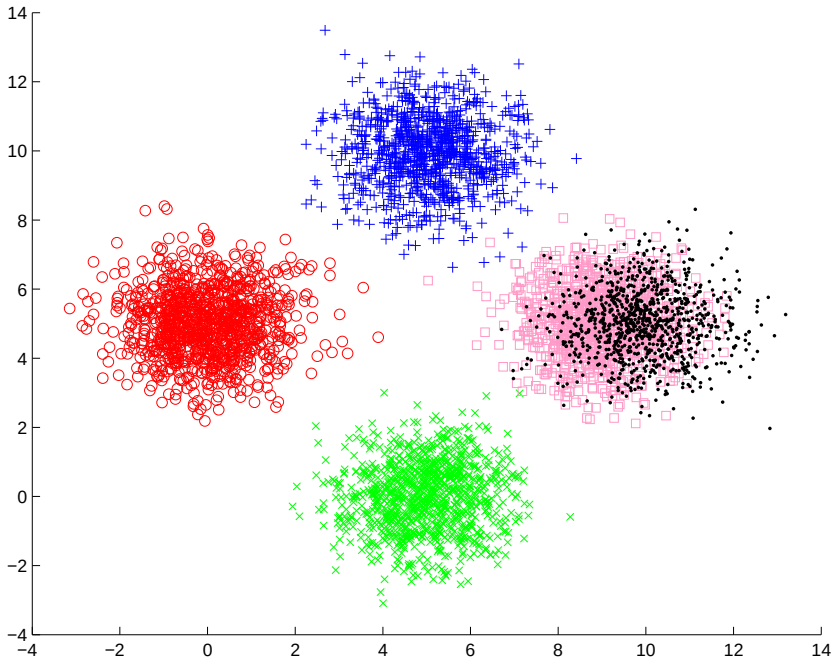


FIGURE 4.10: An sample simulated according to Gaussian mixture model with five clusters. The distance between two overlapping components $\delta = 1$.

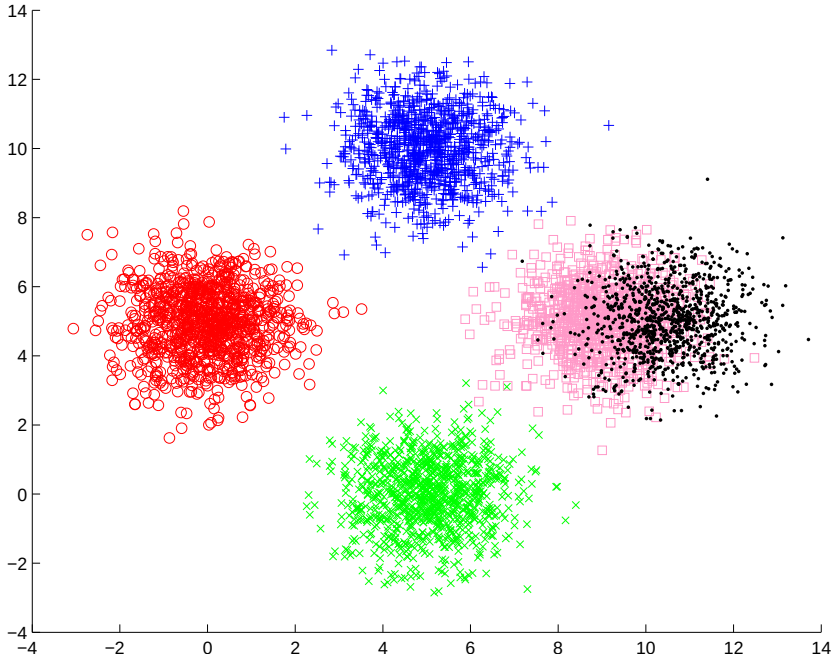


FIGURE 4.11: An sample simulated according to Gaussian mixture model with five clusters. The distance between two overlapping components $\delta = 1.5$.

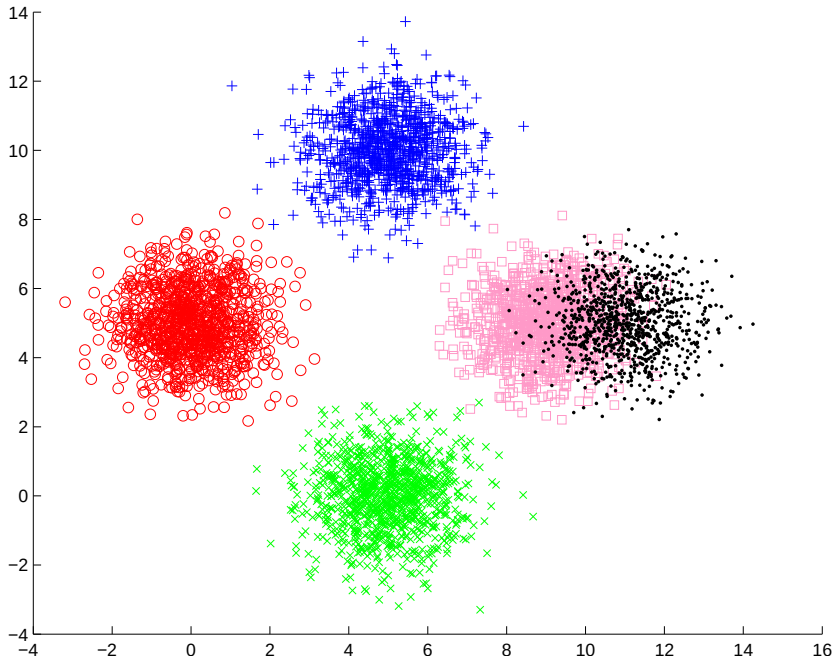


FIGURE 4.12: An sample simulated according to Gaussian mixture model with five clusters. The distance between two overlapping components $\delta = 2$.

Criterion \ Distance	$\delta = 1$		$\delta = 1.5$		$\delta = 2$	
	Model	Num.	Model	Num.	Model	Num.
$BIC_{binned-EM}$	$[\lambda_k \mathbf{I}]$	5	$[\lambda_k \mathbf{B}_k]$	4	$[\lambda \mathbf{I}]$	5
$ICL_{binned-EM}$	$[\lambda_k \mathbf{I}]$	4	$[\lambda_k \mathbf{B}_k]$	4	$[\lambda \mathbf{I}]$	5
$BIC_{bin-EM-CEM}$	$[\lambda_k \mathbf{I}]$	4	$[\lambda_k \mathbf{B}_k]$	4	$[\lambda \mathbf{I}]$	5
$ICL_{bin-EM-CEM}$	$[\lambda_k \mathbf{I}]$	4	$[\lambda_k \mathbf{B}_k]$	4	$[\lambda \mathbf{I}]$	5

TABLE 4.9: Comparison of model choice of four criteria.

In all the simulated datasets of three situations, there are three well-separated clusters and two another clusters well-mixed. We denote these two clusters as C_1 and C_2. When $distance = 1$, C_1 and C_2 are too mixed that it seems they are only one cluster. This is also shown in the Figure 4.10. In this case, it is not easy to tell the number of clusters. $BIC_{binned-EM}$ has chosen the right cluster number 5 and the model $[\lambda_k \mathbf{I}]$, which is slightly more complex than the real model $[\lambda \mathbf{I}]$. $ICL_{binned-EM}$ chose the same model as $BIC_{binned-EM}$, $[\lambda_k \mathbf{I}]$. But $ICL_{binned-EM}$ prefers to consider clusters C_1 and C_1 as one cluster.

When $distance = 1.5$, clusters C_1 and C_2 are less overlapped. From the Figure 4.11, since two clusters are half overlapped, thet are combined as one diagonal cluster. In this case, both of $BIC_{binned-EM}$ and $ICL_{binned-EM}$ criteria chose the most complex diagonal model $[\lambda_k \mathbf{B}_k]$ and 4 clusters. It is reasonable because without any information

of dataset, it is easy to suppose that there are three spherical clusters and one diagonal cluster. The diagonal cluster has twice volume as the spherical clusters.

Finally when $distance = 2$, C_1 and C_2 are better separated. In this case, both of $BIC_{binned-EM}$ and $ICL_{binned-EM}$ criteria can tell the right model and the correct number of clusters.

4.7 AIC, AWE, and NEC criteria applied to binned data clustering

4.7.1 AIC criterion and its derivation

The Akaike information criterion (AIC), proposed by Akaike [17], is generally regarded as the first model selection criterion. It is another measure of a model's suitability to the dataset. It takes the form as follows:

$$AIC(M, K) = 2v_{M,K} - 2L(M)$$

where $v_{m,K}$ is the number of the free parameters of the model M, and $L(M)$ indicates the maximum log-likelihood of the model M.

The index takes into account both the statistical fitting and the number of parameters that have to be estimated to achieve this particular degree of fit, by imposing a penalty for decreasing the number of parameters. We will review the derivation of AIC criterion hereafter.

AIC is based on a simple information theory. We suppose that the data is generated following an unknown model f. To represent f, we consider two candidate models: g_1 and g_2. Suppose that we know the model f, we can get the information lost when using g_1 or g_2 to represent f by calculating the kullback-Leiber divergence [60]. The best model is the one who leads to less information loss. But in fact, we don't know the true model f, [17] showed that by AIC criterion, we can estimate how much more information is lost by g_1 than g_2.

To measure the distance between the potential model g and the true model f, we use the expected Kullback-Leiber information $E_y[I(f, g(\cdot|\widehat{\theta}(y)))]$, which is aimed to minimize:

$$\min_{g \in G} E_y[I(f, g(\cdot|\widehat{\theta}(y)))]$$

We have:

$$E_y[I(f, g(\cdot|\widehat{\theta}(y)))] = \int_\Omega f(x)\log(f(x))dx - \underbrace{\int_\Omega f(y)[\int_\Omega f(x)\log(g(x|\widehat{\theta}(y)))dx]dy}_{E_yE_x[\log(g(x|\widehat{\theta}(y)))]}$$

where G is the collection of potential models, $\widehat{\theta}$ is the Maximum Likelihood Estimation (MLE) based on model g and data y, and y is a random sample from the density function $f(x)$.

Minimizing $E_y[I(f, g(\cdot|\widehat{\theta}(y)))]$ leads to maximize $E_yE_x[\log(g(x|\widehat{\theta}(y)))]$. An approximately unbiased estimate of $E_yE_x[\log(g(x|\widehat{\theta}(y)))]$ for large sample and the best-fit model is:

$$L(M) - v_{M,K}$$

AICc is AIC with a correction for finite sample sizes. It is used when the sample size n and the number of parameters $v_{M,K}$ has the relation: $\frac{n}{v_{M,K}} < 40$. It takes the form:

$$AICc = AIC + \frac{2v_{M,K}(v_{M,K}+1)}{n - v_{M,K} - 1}$$

where n is the sample size and K is the number of free parameters.

Compared to the BIC criterion, the AIC penalizes the number of parameters less strongly. A comparison of AIC/AICc and BIC is given by Burnham and Anderson [61].

4.7.2 AWE criterion

Approximate Weight of Evidence (AWE) is an approximation of Bayes factor. Bayes factor, which is noted as B_0, indicates the relation between the integrated likelihood of two models M and M_0.

$$B_0 = \frac{P(X|M)}{P(X|M_0)}$$

where the likelihood

$$P(X/M) = \int P(\theta|M)P(X|\theta, M)d\theta$$

with θ the model parameter vector.

To calculate the Bayes factor, we use the Schwarz criterion [20]:

$$S = 2L(M) - 2L(M_0) - (v(M) - v(M_0))\log(n) \tag{4.1}$$

where $L(M)$ is the maximum likelihood of the model M and $v(M)$ is the number of free parameters in this model. S is a rough approximation of $2\log(B_0)$. Under certain conditions, we have:

$$\frac{S - 2\log(B_0)}{2\log(B_0)} \longrightarrow 0$$

Here let's remember that, to obtain BIC criterion, we remove the constants in the Equation 4.1 and takes $-S$ as criterion:

$$BIC(M) = -2L(M) + v(M)\log(n)$$

To obtain another criterion, Banfield and Raftery [16] have given another approximation of Bayes factor B_0. Contrary with BIC criterion, this approximation limits the choice of number of class. The integrated likelihood $P(\mathbf{x}/K)$, corresponding to a K-component model, has the form as follows:

$$P(\mathbf{x}/K) = \sum_{\boldsymbol{z}} \int f(\mathbf{x}, \boldsymbol{z}|\theta, K)\pi(\theta|K)d\theta$$

with $f(\mathbf{x}, \boldsymbol{z}|\theta, K)$ the complete likelihood of $(\theta, \boldsymbol{z})$ for K classes, and $\pi(\theta|K)$ the prior of θ for K classes. Being within the framework of hierarchical clustering, it uses the fact that K classes is the result from the agglomeration of two classes among $K+1$ classes. The proportions of classes are required to be equal. Using the asymptotic approximation [62], we obtain:

$$-2\log(B_0) \approx -2CL(K) + 2CL(1) + 2(v(K) - v(1))(\frac{3}{2} + \log(n)) \tag{4.2}$$

where B_0 is the Bayes factor of a model of K classes against a model of only one class. $CL(K)$ is the maximum complete likelihood for K classes. By removing the constants in the Equation 4.2, we obtain the AWE criterion:

$$AWE(K) = -2CL(K) + 2v(K)(\frac{3}{2} + \log(n))$$

4.7.3 NEC criterion

NEC criterion's full name is Normalized Entropy Criterion. It was proposed by [51] to estimate the number of clusters arising from a mixture model.

In the mixture model, a dataset $\mathbf{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is assumed to follow a probability distribution of K-component:

$$f(\mathbf{x}; \mathbf{\Phi}) = \sum_{k=1}^{K} \pi_k f_k(\boldsymbol{x}; \theta_k)$$

with $\mathbf{\Phi} = (\pi_1, \ldots, \pi_K, \theta_1, \ldots, \theta_K)$, where π_k $(k = 1, \ldots, K)$ denote the mixing proportions of the mixtures ($0 < \pi_k < 1$ and $\sum_{k=1}^{K} \pi_k = 1$), and $\theta_k = (\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)$ $(k = 1, \ldots, K)$ are the parameters of Gaussian distribution functions f_k of components: mean vectors $\boldsymbol{\mu}_k$ and variance matrices $\mathbf{\Sigma}_k$.

The maximized log-likelihood of the dataset $\mathbf{x}$ is

$$L(K) = \sum_{i=1}^{n} \log[\sum_{k=1}^{K} \widehat{\pi}_k f_k(\boldsymbol{x}_i; \widehat{\theta}_k)]$$

where $\widehat{\pi}_k$ and $\widehat{\theta}_k$ denote the maximum likelihood estimates of π_k and θ_k respectively.

The NEC criterion is derived from a relation between the log-likelihood $L(K)$ and a classification type log-likelihood $C(K)$:

$$L(K) = C(K) + E(K) \tag{4.3}$$

where

$$C(K) = \sum_{k=1}^{K} \sum_{i=1}^{n} t_{ik} \log[\widehat{\pi}_k f_k(\boldsymbol{x}_i; \widehat{\theta}_k)]$$

with

$$t_{ik} = \frac{\widehat{\pi}_k f_k(\boldsymbol{x}_i; \widehat{\theta}_k)}{\sum_{j=1}^{K} \widehat{\pi}_j f_j(\boldsymbol{x}_i; \widehat{\theta}_j)}$$

denoting the conditional probability that $\boldsymbol{x}_i$ arises from the kth mixture component.

In the Equation 4.3, the entropy $E(K)$ has the form

$$E(K) = -\sum_{k=1}^{K} \sum_{i=1}^{n} t_{ik} \log(t_{ik}) \geq 0$$

which measures the overlap of the mixture components. The entropy $E(K)$ cannot be used directly as a criterion to obtain the number of clusters, because $L(K)$ is an increasing function of K and should be normalized. Thus we have

$$1 = \frac{C(K - C(1))}{L(K) - L(1)} + \frac{E(K) - E(1)}{L(K) - L(1)}, K > 1 \tag{4.4}$$

From the Equation 4.4, we can get the NEC criterion:

$$NEC(K) = \frac{E(K)}{L(K) - L(1)}$$

In order to adapt NEC criterion to binned data framework, we adjust some parameters:

$$E(K) = -\sum_{k=1}^{K}\sum_{r=1}^{v} p_{r/k}(\mathbf{\Phi}) \log(p_{r/k}(\mathbf{\Phi}))$$

where

$$p_{r/k}(\mathbf{\Phi}) = \frac{\widehat{\pi}_k \int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \widehat{\theta}_k) dx}{\sum_{k=1}^{K} \widehat{\pi}_k \int_{\mathcal{H}_r} f_k(\boldsymbol{x}; \widehat{\theta}_k) dx}$$

From the definition of the NEC criterion, $L(K)$ cannot be $L(1)$. So we cannot compare the cases when $K = 1$ with when $K > 1$ by the NEC criterion. To deal with this problem, Celeux and Soromenho [51] has proposed a procedure which is reviewed in the Chapter 1. But this method is restricted to Gaussian mixtures and shows a disadvantage [52]. Biernacki et al. [63] proposed a simpler and general procedure to deal with this problem.

4.7.4 Numerical Experiments

In this part, we propose to study and compare the performances of all the criteria we mentioned above: likelihood L, complete likelihood CL, BIC, ICL, AIC, AWE, and NEC criteria. The experiments are divided into two parts. In the first part, we focus on the choice of number of clusters. In this case, the model is considered as already known and follows a simple model. In the second part, the model is considered as unknown but the number of clusters is fixed. This part aims to study the ability of model choice of all the criteria.

4.7.4.1 Choice of number of clusters

The simulated data is generated according to a model which we used in the Subsection 4.6: a Gaussian mixture model with five components. Each component has the same structure: the simplest spherical model [$\lambda\mathbf{I}$]. Three of the components are well separated with each other. The other two components are overlapped to certain level. Three groups of sample we simulated correspond to three levels of overlapping of theses two components. The level of overlapping is referred to the distance between two

components:

$$\delta = \sqrt{(\mu_1 - \mu_2)^T (\frac{\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2}{2})^{-1} (\mu_1 - \mu_2)}$$

So the model parameters of these three groups of simulated data are detailed in the Table 4.10: In the algorithm, the potential model is fixed. It is same as the real model

Common parameters	$n = 5000, K = 5, p_1 = p_2 = p_3 = p_4 = p_5 = 0.2$ $\mu_1 = (0,5), \mu_2 = (5,0), \mu_3 = (5,10), \mu_4 = (9,5),$ $\Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_4 = \Sigma_5 = diag(1,1)$	
Different parameters	Group 1	$\delta = 1, \mu_5 = (10,5)$
	Group 2	$\delta = 1.5, \mu_5 = (10.5,5)$
	Group 3	$\delta = 2, \mu_5 = (11,5)$

TABLE 4.10: Model parameters of simulated data.

[$\lambda\mathbf{I}$]. For each group of dataset, we simulate 30 datasets. For each dataset, we apply binned-EM and bin-EM-CEM algorithms, so as to get the maximum likelihood and the maximum complete likelihood. Then we use the criteria to choose the number of cluster. The number which is chosen the most is considered as the choice of the corresponding criterion. The bin size is still fixed at $40bins \times 40bins$.

The result is shown in the Table 4.11:

Criteria \ Dataset	Group 1	Group 2	Group 3
L	4	5	5
LM	4	4	5
BIC	4	5	5
ICL	4	4	5
AIC	4	4	5
AWE	4	4	4
NEC	4	5	5

TABLE 4.11: Result of choice of number of clusters by L, LM, BIC, ICL, AIC, AWE, and NEC criteria.

From the result, when the two components are too overlapped (distance $\delta = 1$), all the criteria consider that there are only four criteria. When the distance $\delta = 1.5$, L, BIC and NEC criteria succeeded to point out the right number of clusters. The CL, ICL, AIC and AWE criteria favor 5 clusters. We notice that L has the same behavior as BIC, and CL has the same behavior as ICL. Same as in the standard framework, in binned data clustering, L and BIC criteria still perform better than CL and ICL criteria. Under

the condition that the model is fixed as $[\lambda\mathbf{I}]$, AIC and AWE criteria couldn't make the right choice. In the third group of simulated dataset, the two connected clusters are well separated. In this case, all the criteria are able to detect five clusters.

4.7.4.2 Choice of model

In this part, we study the model choice for binned data clustering of all the criteria introduced in this chapter. Under the condition that the number of clusters is known, the difference of choice of different criteria is the model selection. For the experiment, we simulate data according to fourteen parsimonious Gaussian mixture models. Follow each model, we generate 30 datasets of size= 3000. To apply binned-EM algorithm, the whole space is divided into 40bins×40bins. Details of model parameters see in the Section 4.3. The model which has been chosen by the most times is selected as the final choice of corresponding criterion. The result is shown in the Table 4.12.

From the result, we can see that BIC, ICL and NEC criteria have exactly the same result in this experiment. They are able to choose the right model for 8 different data structure. Compared to BIC, ICL and NEC criteria, AIC criterion behaves less outstanding. It favors the most complex model $[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$ for the dataset simulated according to the models $[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$, $[\lambda_k\mathbf{B}]$ and $[\lambda_k\mathbf{B}]$. Among all these criteria, AWE criterion gives the best result. It give 9 correct answers out of 14 (maximum 8 right answers for other criteria). AWE succeeded detect one data structure that other criteria can detect: $[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$. If we have to measure the performance of these criteria, we would give out this order: AWE, (BIC, ICL, NEC), AIC, CML, ML, from the best criterion to the worst one.

4.8 Conclusion

This chapter studied the model selection for binned data clustering. Our aim was to know which model one should use to get a good clustering result, without knowing the data structure. To reach this goal, in this chapter, several classical criteria were adapted to binned data clustering. These criteria aim to select the right model among fourteen parsimonious Gaussian mixture models, as well as the number of clusters. A right model must fit well the data and meet the clustering precision requirements with a reasonable computation time.

First of all, we focused on the BIC and ICL criteria. Basing on the BIC and ICL criteria for standard data, we proposed four new associations for model selection in binned data

Data Model \ Criteria	ML	BIC	CML	ICL	AIC	AWE	NEC
$[\lambda\mathbf{DAD}^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda\mathbf{DAD}^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda\mathbf{DAD}^T]*$	$[\lambda\mathbf{DAD}^T]*$	$[\lambda\mathbf{DAD}^T]*$	$[\lambda\mathbf{DAD}^T]*$
$[\lambda_k\mathbf{DAD}^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{DAD}^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k\mathbf{DAD}^T]*$	$[\lambda_k\mathbf{DAD}^T]*$	$[\lambda_k\mathbf{DAD}^T]*$	$[\lambda_k\mathbf{DAD}^T]*$
$[\lambda\mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	$[\lambda\mathbf{D}\mathbf{A}_k\mathbf{D}^T]*$	$[\lambda\mathbf{D}\mathbf{A}_k\mathbf{D}^T]*$	$[\lambda\mathbf{D}\mathbf{A}_k\mathbf{D}^T]*$	$[\lambda\mathbf{D}\mathbf{A}_k\mathbf{D}^T]*$	$[\lambda\mathbf{D}\mathbf{A}_k\mathbf{D}^T]*$	$[\lambda\mathbf{D}\mathbf{A}_k\mathbf{D}^T]*$	$[\lambda\mathbf{D}\mathbf{A}_k\mathbf{D}^T]*$
$[\lambda_k\mathbf{D}\mathbf{A}_k\mathbf{D}^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$
$[\lambda\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$
$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]*$
$[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{B}_k]$	$[\lambda_k\mathbf{B}_k]$	$[\lambda_k\mathbf{B}_k]$	$[\lambda_k\mathbf{B}_k]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{B}_k]$
$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]*$
$[\lambda\mathbf{B}]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{B}_k]$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k\mathbf{B}]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{B}]$	$[\lambda_k\mathbf{B}_k]$
$[\lambda_k\mathbf{B}]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{B}]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{B}]*$	$[\lambda_k\mathbf{D}_k\mathbf{A}\mathbf{D}_k^T]$	$[\lambda_k\mathbf{B}]*$	$[\lambda_k\mathbf{B}]*$
$[\lambda\mathbf{B}_k]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{B}_k]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{B}_k]$	$[\lambda_k\mathbf{B}_k]$	$[\lambda_k\mathbf{B}_k]$	$[\lambda_k\mathbf{B}_k]$
$[\lambda_k\mathbf{B}_k]$	$[\lambda_k\mathbf{B}_k]*$	$[\lambda_k\mathbf{B}_k]*$	$[\lambda_k\mathbf{B}_k]*$	$[\lambda_k\mathbf{B}_k]*$	$[\lambda_k\mathbf{B}_k]*$	$[\lambda_k\mathbf{B}_k]*$	$[\lambda_k\mathbf{B}_k]*$
$[\lambda\mathbf{I}]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{I}]$	$[\lambda_k\mathbf{I}]$	$[\lambda_k\mathbf{I}]$	$[\lambda_k\mathbf{I}]$	$[\lambda_k\mathbf{I}]$	$[\lambda_k\mathbf{I}]$
$[\lambda_k\mathbf{I}]$	$[\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k^T]$	$[\lambda_k\mathbf{I}]*$	$[\lambda_k\mathbf{I}]*$	$[\lambda_k\mathbf{I}]*$	$[\lambda_k\mathbf{I}]*$	$[\lambda_k\mathbf{I}]*$	$[\lambda_k\mathbf{I}]*$

Table 4.12: Model choice of ML, CML, BIC, ICL, AIC, AWE, and NEC criteria.

clustering using the binned-EM algorithms and bin-EM-CEM algorithms. In fact, each of the criteria BIC and ICL was associated with either fourteen binned-EM algorithms or fourteen bin-EM-CEM algorithms. Experiments on simulated data and real data were presented to compare the results in model choice, including the choice of the number of clusters, of these four criteria. The experimental results showed that:

Generally speaking, all these four new associations are able to choose the best-fit model for different datasets. Compared to the BIC criterion when applied to binned data, the ICL criterion prefers to choose less number of clusters with a more complex model. This behavior remains the same as in the standard data framework. The size of bins affects the model choice of BIC and ICL criteria. Smaller bins lead to a better-fit model, but cost more consumption of computation time.

In this chapter, we also adapted AIC, AWE, and NEC criteria, to model selection in binned data clustering. To test the performance of these criteria, we divided the experiments into two parts: the first part was to study the choice of number of clusters and the second part was to study the choice of model. The results showed that when the model is provided, the BIC and NEC criteria perform slightly better than the other criteria (ICL, AIC, and AWE) in choosing the number of clusters. And when the number of clusters is known, the BIC, ICL, AIC, AWE, and NEC criteria, can choose the right model in most of the cases. More precisely, the AWE criterion outperforms the other criteria. The BIC, ICL, and NEC criteria, have almost the same behaviors in model choice. The AIC criterion has a slightly less outstanding performance compared to the other criteria in our experiments.

General conclusion and prospective

General conclusion

In this thesis, we studied the application of EM and CEM algorithms of parsimonious Gaussian mixture models to binned data clustering, and its associated model selection.

In cluster analysis, mixture-model-based clustering approach is one of the most important approaches. It helps to discover and understand the data structure by assuming potential models. The two most commonly used mixture-model-based clustering approaches are the mixture approach and the classification approach. The mixture approach aims to maximize the likelihood of mixture model parameters, using the EM algorithm, and to deduce the data partition from the estimated mixture model parameters. The classification approach aims to maximize the complete likelihood *i.e.* the likelihood of the mixture model parameters and the data labels, using the CEM algorithm. Along with the development of information technology, the amount of data increased explosively. In this case, the EM and CEM algorithms suffer from a long computation time problem, while dealing with data of large size. Thus, one of the main objectives of this thesis was to find a solution to obtain a good data clustering result within a reasonable computation time.

In standard data framework, fourteen parsimonious Gaussian mixture models were proposed according to a parametrization of variance matrices of clusters. These models have less free parameters comparing to the most general one. They can adapt efficiently to different data structures and obtain a good clustering result. At another side, binned data was introduced into clustering in order to reduce the amount of data (the data size): from the number of points to the number of bins. Moreover, the size of bins can be modified according to our actual needs. Thus, the computation time can be reduced and especially controlled by defining intelligently the size of bins.

So in this thesis, we developed fourteen binned-EM algorithms and fourteen bin-EM-CEM algorithms of fourteen parsimonious Gaussian mixture models. These new algorithms combine the advantages of binning data in time reduction and the advantages of parsimonious Gaussian mixture models in simplifying the parameters estimation. The complexities of EM and binned-EM algorithms were calculated and compared. The comparison result showed that the binned-EM algorithm is faster than the EM algorithm when the data size is big enough. The complexities of CEM and bin-EM-CEM algorithms were also calculated and compared. The bin-EM-CEM algorithm is faster than the CEM algorithm when the data size increases and satisfies a condition. This condition is defined by an inequality which is given out in this thesis.

While using binned-EM algorithms and bin-EM-CEM algorithms, different parsimonious Gaussian mixture models lead to different variance matrices estimations. Due to this specific and precise estimation, parsimonious models can better-fit different datasets than the most complex model. Also, by applying these parsimonious Gaussian mixture models, the parameter estimations are simplified. Thus, while combining the advantages of binned data and parsimonious models, binned-EM and bin-EM-CEM algorithms of parsimonious Gaussian mixture models can fit well the data within a reasonable computation time.

Numerical experiments of the fourteen binned-EM and the fourteen bin-EM-CEM algorithms of parsimonious Gaussian mixture models applied to different datasets were performed and analyzed. The result showed that the parsimonious model representing exactly the data structure obtained the highest accuracy and spent less computation time. This result implies that the parsimonious models simplify the parameters estimation and then they are also able to adapt to datasets of different structures and provide a good clustering result. We also studied an experiment of binned-EM and bin-EM-CEM algorithms with different bin sizes. The result showed that bigger bins lead to less computation time with acceptable loss of precision. The applications of binned-EM and bin-EM-CEM algorithms of parsimonious Gaussian mixture models to image segmentation were analysed. Our algorithms obtained a good image segmentation result in little computation time. Specially compared to the EM and CEM algorithms, our algorithms have evident advantage in computation time while obtaining almost the same image segmentation result.

The BIC and ICL criteria are two well-known criteria for clustering model choice. The idea is to introduce a penalty term for the model complexity to the maximum value of the likelihood (for BIC) and to the maximum value of the complete likelihood (for ICL). Since the right model can obtain the best clustering result, the model selection is an important step. Thus, in this thesis, to select the best model, we have extended the BIC

and ICL criteria to binned data clustering and we associated them with binned-EM and bin-EM-CEM algorithms. These four associations (either BIC or ICL criteria associated to either binned-EM or bin-EM-CEM algorithms) are able to choose the right model. The ICL criterion prefers to choose less number of clusters with a more complex model comparing to the BIC criterion.

Besides this, we also adapted the AIC, NEC, and AWE criteria, to binned data clustering. The experimental results showed that these criteria can choose the right model and the correct number of clusters. The AWE criterion outperforms the other criteria. The BIC, ICL, and NEC criteria, have almost the same behaviors in model choice.

Prospective

In the continuation of this work, we can see the perspectives in three directions:

- Adapt other parsimonious mixture models basing on other criteria to binned data. For example, the models based on factor analysis model were mentioned in the Chapter 1. These models can be adapted to binned data.
- Develop new criterion which is better adapted to binned data clustering. Many criteria were developed for standard data clustering and they perform very well. In this thesis, we didn't adapt all the possible criteria to binned data clustering. It would be interesting to adapt other criteria to binned data clustering and to have a deep comparison.
- Study the ways of binning data in order to obtain a better clustering result and less computation time. It remains interesting works to do in improving the speed of binned data clustering. The ways of how to bin data are essential in this subject.

Appendix A

Theorem proving

Theorem A.1. *The orthogonal matrix* $\mathbf{Q}$ *minimizing* $\mathrm{tr}\Big(\mathbf{QAQ}^{-1}\mathbf{B}\Big)$ *where* $\mathbf{A}$ *and* $\mathbf{B}$ *are diagonal matrices, with general diagonal term* α_j *and* β_j *such that* $\alpha_1 \geqslant \alpha_2 \geqslant \ldots \geqslant \alpha_d$ *and* $\beta_1 \leqslant \beta_2 \leqslant \ldots \leqslant \beta_d$*, is the identity matrix and the minimized value is* $\mathrm{tr}\Big(\mathbf{AB}\Big) = \sum_{j=1}^{d} \alpha_j\beta_j$.

Proof. Let $\alpha'_1, \ldots, \alpha'_d$ be the general terms of the diagonal of the matrix $\mathbf{QAQ}^{-1}$. Since $\mathbf{Q}$ is orthogonal the matrix $\mathbf{QAQ}^{-1}$ is symmetric and we have from Lemma A.2 below

$$\begin{aligned} \alpha'_1 + \ldots + \alpha'_c &\leqslant \alpha_1 + \ldots + \alpha_c \qquad (1 \leqslant c < d) \\ \alpha'_1 + \ldots + \alpha'_d &\leqslant \alpha_1 + \ldots + \alpha_d. \end{aligned}$$

Now, $\mathrm{tr}\Big(\mathbf{QAQ}^{-1}\mathbf{B}\Big) = \sum_{j=1}^{d} \alpha'_j\beta_j$ ($\mathbf{B}$ diagonal). Thus

$$\begin{aligned} \mathrm{tr}\Big(\mathbf{AB}\Big) &= \sum_{j=1}^{d-1} \alpha'_j\beta_j + \Big(\sum_{j=1}^{d} \alpha_j - \sum_{j=1}^{d-1} \alpha'_j\Big)\beta_d \\ &= \sum_{j=1}^{d-1} \alpha'_j\beta_j + \sum_{j=1}^{d-1} (\alpha_j - \alpha'_j)\beta_d + \alpha_d\beta_d \\ &\geqslant \sum_{j=1}^{d-1} \alpha'_j\beta_j + \sum_{j=1}^{d-1} (\alpha_j - \alpha'_j\beta_{d-1}) + \alpha_d\beta_d \end{aligned}$$

Since $\sum_{j=1}^{d-1}(\alpha_j - \alpha'_j) \geqslant 0$ and $\beta_d \geqslant \beta_{d-1}$. Repeating the same argument, we get

$$\begin{aligned} \mathrm{tr}\Big(\mathbf{AB}\Big) &= \sum_{j=1}^{d-2} \alpha'_j\beta_j + \sum_{j=1}^{d-2} (\alpha_j - \alpha'_j)\beta_{d-2} \\ &\quad + \alpha_{d-1}\beta_{d-1} + \alpha_d\beta_d \end{aligned}$$

and finally,

$$\mathrm{tr}\Big(\mathbf{AB}\Big) \geqslant \sum_{j=1}^{d} \alpha_j \beta_j$$

Hence $\sum_{j=1}^{d} \alpha_j \beta_j$ is a lower bound of $\mathrm{tr}\Big(\mathbf{QAQ}^{-1}\mathbf{B}\Big)$. Since this bound is reached as $\mathbf{Q} = \mathbf{I}$, the proof is complete. □

Lemma A.2. *Let* $\mathbf{A}$ *be a real, d-dimensional, symmetric, positive matrix and let* q *be the associated quadratic form. Let* $\lambda_1 \geqslant \ldots \geqslant \lambda_d$ *and* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d$ *be respectively the eigenvalues and the eigenvectors of* $\mathbf{A}$*. For any orthonormal basis* $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_d$ *of* $\mathbb{R}^d$*, we have*

$$\sum_{j=1}^{d} q(\boldsymbol{y}_j) = \sum_{j=1}^{d} q(\boldsymbol{x}_j) \quad (= \sum_{j=1}^{d} \lambda_j) \tag{A.1}$$

$$\forall k = 1, \ldots, d \quad \sum_{j=1}^{k} q(\boldsymbol{y}_j) \leqslant \sum_{j=1}^{k} q(\boldsymbol{x}_j) \ \ (= \sum_{j=1}^{k} \lambda_j) \tag{A.2}$$

Proof. Equation (A.1) follows from the decomposition of the vectors $\boldsymbol{y}_j (1 \leqslant i \leqslant d)$ on the basis of the eigenvectors of $\mathbf{A}$

$$\forall i = 1, \ldots, d \qquad \boldsymbol{y}_i = \sum_{j=1}^{d} y_i^j \boldsymbol{x}_j.$$

Then, we have

$$\forall i = 1, \ldots, d \qquad q(\boldsymbol{y}_i) = \sum_{j=1}^{d} (y_i^j)^2 q(\boldsymbol{x}_j)$$

$$\sum_{i=1}^{d} q(\boldsymbol{y}_i) = \sum_{i=1}^{d} \sum_{j=1}^{d} (y_i^j)^2 q(\boldsymbol{x}_j) = \sum_{j=1}^{d} (\sum_{i=1}^{d} (y_i^j)^2) q(\boldsymbol{x}_j)$$

And, Equation (A.1) is derived from the fact that the y_i^j's are the coordinates, in a an orthonormal basis of vectors, of a vector with norm 1.

The Equation (A.2) can be proved by induction on k.

$$\begin{aligned} q(\boldsymbol{y}_1) &= q(\sum_{j=1}^{d} y_i^j \boldsymbol{x}_j) = \sum_{j=1}^{d} (y_1^j)^2 q(\boldsymbol{x}_j) = \sum_{j=1}^{d} (y_1^j)^2 \lambda_j \\ &\leqslant \sum_{j=1}^{d} (y_1^j)^2 \lambda_1 = \lambda_1 \end{aligned}$$

Assume that Equation (A.2) is true for $k-1$. Let F_k be the space generated by $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$ and let E_{k-1} be the space generated by $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k-1}$. dim $F_k = k$ and dim $E_{k-1}^{\perp} =$

$d+1-k$. As a consequence $G = F_k \bigcap E_{k-1}^{\perp} \neq \emptyset$. Let $\boldsymbol{y}$ be a normed vector in G. Let $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{k-1}$, $\boldsymbol{v}$ be an orthonormal basis of F. Applying Equation (A.1), we have

$$\sum_{j=1}^{k} q(\boldsymbol{y}_j) \quad = \quad \sum_{j=1}^{k-1} q(\boldsymbol{z}_j) + q(\boldsymbol{v}) \tag{A.3}$$

Since Equation (A.2) is true for $k-1$, we have

$$\sum_{j=1}^{k-1} q(\boldsymbol{z}_j) \leqslant \sum_{j=1}^{k-1} q(\boldsymbol{x}_j)$$

Since $\boldsymbol{v} \in E_{k-1}^{\perp}$, it is possible to write

$$\boldsymbol{v} = \sum_{j=k}^{d} v_j \boldsymbol{x}_j$$

$$q(\boldsymbol{v}) = \sum_{j=k}^{d} (v_j)^2 q(\boldsymbol{x}_j) \leqslant (\sum_{j=k}^{d} (v_j)^2) \lambda_k = \lambda_k = q(\boldsymbol{x}_k)$$

Then, from Equation (A.3), we have

$$\sum_{j=1}^{k} q(\boldsymbol{y}_j) \leqslant \sum_{j=1}^{k} q(\boldsymbol{x}_j)$$

and the proof of the lemma is complete. □

Theorem A.3. *The symmetric positive matrix* $\mathbf{M}$ *of dimension* $d \times d$ *and* $|\mathbf{M}| = 1$ *minimizing* $\mathrm{tr}(\mathbf{M})$ *is identity matrix* $\mathbf{I}$*, and the minimized value is* d*.*

Proof. If we note $\lambda_1, \ldots, \lambda_d$ as the eigenvalues of the symmetric matrix M, the problem reduces to minimizing $\sum_i \lambda_i$ under the constrain of $\prod_i \lambda_i = 1$. Knowing that all of the eigenvalues are positive, we apply the method of Lagrange multipliers. So the problem equals the minimization of

$$g(\lambda_1, \ldots, \lambda_d) \quad = \quad \sum_i \lambda_i - \lambda(\prod_i \lambda_i - 1)$$

partially deducing by derivative

$$g'_{\lambda_j}(\lambda_1, \ldots, \lambda_p) = 1 - \lambda \frac{\prod_i \lambda_i}{\lambda_j} = 0 \qquad \forall j$$

remember

$$\lambda_j = \lambda \prod_i \lambda_i \qquad \forall j$$

and using the constrain $\prod_j \lambda_j = 1$, we have $\lambda = \frac{1}{\prod_i \lambda_i}$, so $\lambda_i = 1 \qquad \forall i$. Finally we get the result. □

Corollary A.4. *The symmetric positive matrix* $\mathbf{M}$ *of dimension* $d \times d$ *and* $|\mathbf{M}| = 1$ *minimizing* $\mathrm{tr}(\mathbf{Q}\mathbf{M}^{-1})$ *where* $\mathbf{Q}$ *is a symmetric positive definite matrix is*

$$\mathbf{M} = \frac{\mathbf{Q}}{|\mathbf{Q}|^{1/d}}$$

and the minimized value is $d|\mathbf{Q}|^{1/d}$.

Proof. Make $\mathbf{N} = |\mathbf{Q}|^{-(1/d)}\mathbf{Q}\mathbf{M}^{-1}$, so $|\mathbf{N}| = 1$. The problem equals to the minimization of $\mathrm{tr}(\mathbf{N})$. As the result of Theorem A.3, we have the solution $\mathbf{N} = \mathbf{I}$, that is $|\mathbf{Q}|^{-(1/d)}\mathbf{Q}\mathbf{M}^{-1} = \mathbf{I}$. The result can be deduced easily. □

Corollary A.5. *The diagonal matrix* $\mathbf{M}$ *of dimension* $d \times d$ *and* $|\mathbf{M}| = 1$ *minimizing* $\mathrm{tr}(\mathbf{Q}\mathbf{M}^{-1})$ *where* $\mathbf{Q}$ *is a symmetric positive definite matrix is*

$$\mathbf{M} = \frac{diag(\mathbf{Q})}{|diag(\mathbf{Q})|^{1/d}}$$

and the minimized value is $d|diag(\mathbf{Q})|^{1/d}$.

Proof. If $\mathbf{M}$ is diagonal matrix, $\mathbf{M}^{-1}$ is also a diagonal matrix, then we have $\mathrm{tr}(\mathbf{Q}\mathbf{M}^{-1}) = \mathrm{tr}(diag(\mathbf{Q})\mathbf{M}^{-1})$. From the proof of Corollary A.4, we have $\mathbf{M} = \frac{diag(\mathbf{Q})}{|diag(\mathbf{Q})|^{1/d}}$. □

Theorem A.6. *To minimize* $\mathrm{tr}(\mathbf{Q}\mathbf{M}^{-1}) + \alpha \ln |\mathbf{M}|$ *where* $\mathbf{Q}$ *is a symmetric positive definite matrix and* α *is a positive real number, the result of the* $d \times d$ *symmetric matrix* $\mathbf{M}$ *is* $\mathbf{M} = (1/\alpha)\mathbf{Q}$.

Proof. Considering $\mathbf{M} = |\mathbf{M}|^{1/d}N$ with $\mathbf{N} = 1$, we have

$$\mathrm{tr}(\mathbf{Q}\mathbf{M}^{-1}) + \alpha \ln |\mathbf{M}| = \frac{1}{|\mathbf{M}|}\mathrm{tr}(\mathbf{Q}\mathbf{N}^{-1}) + \alpha \ln |\mathbf{M}|$$

From Corollary A.4, we have

$$N = |\mathbf{Q}| - (1/d)\mathbf{Q}$$

Thus,

$$\mathrm{tr}(\mathbf{Q}\mathbf{M}^{-1}) + \alpha \ln |\mathbf{M}| \quad = \quad \frac{d|\mathbf{Q}|^{1/d}}{|\mathbf{M}|^{1/d}} + \alpha \ln |\mathbf{M}|$$

We set:

$$\frac{d|\mathbf{Q}|^{1/d}}{|\mathbf{M}|^{1/d}} + \alpha \ln |\mathbf{M}| \quad = \quad 0 \tag{A.4}$$

The result of derivative of function (A.4) with respect to $|\mathbf{M}|$ is $|\mathbf{M}| = (\frac{1}{\alpha^{(d)}})|\mathbf{Q}|$, so the final result is $\mathbf{M} = (\frac{1}{\alpha})\mathbf{Q}$ □

Corollary A.7. *To minimize* $\mathrm{tr}(\mathbf{Q}\mathbf{M}^{-1}) + \alpha \ln |\mathbf{M}|$ *where* $\mathbf{Q}$ *is a symmetric positive definite matrix and* α *is a positive real number, the result of the* $d \times d$ *diagonal matrix* $\mathbf{M}$ *is* $\mathbf{M} = (1/\alpha^d) diag(\mathbf{Q})$.

Proof. If M is diagonal matrix, M^{-1} is also a diagonal matrix, then we have $\mathrm{tr}(\mathbf{Q}\mathbf{M}^{-1}) + \alpha \ln |\mathbf{M}| = \mathrm{tr}(diag(\mathbf{Q})\mathbf{M}^{-1}) + alpha \ln |\mathbf{M}|$. From the proof of Corollary A.6, we have $\mathbf{M} = \frac{1}{\alpha^d} diag(\mathbf{Q})$. □

Bibliography

[1] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97:611–631, 2000.

[2] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[3] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.

[4] H. H. Bock. Probabilistic models in cluster analysis. *Computational Statistics & Data Analysis*, 23:5–28, 1996.

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

[6] A. L. Scott and M. J. Symons. Clustering methods based on likelihood ratio criteria. *Biometrics*, 27:387–397, 1971.

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

[8] G. Celeux and G. Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics & Data Analysis*, 14(3):315–332, 1992.

[9] A. W. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In *Advances in Neural Information Processing Systems*, pages 543–549. MIT Press, 1999.

[10] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178, 2000.

[11] F.X. Jollois and M. Nadif. Speed-up for the expectation-maximization algorithm for clustering categorical data. *Journal of Global Optimization*, 37(4):513–525, 2007.

[12] G. J. McLachlan and P. N. Jones. Fitting mixture models to grouped and truncated data via the EM algorithm. *Biometrics*, 44(2):571–578, 1988.

[13] I. V. Cadez, P. Smyth, G. J. McLachlan, and C. E. McLaren. Maximum likelihood estimation of mixture densities for binned and truncated multivariate data. *Machine Learning*, 47(1):7–34, 2002.

[14] A. Samé, C. Ambroise, and G. Govaert. A classification EM algorithm for binned data. *Computational Statistics & Data Analysis*, 51(2):466–480, 2006.

[15] G. Celeux and G. Govaert. Gaussian parsimonious clustering models. *Pattern Recognition*, 28(5):781–793, 1995.

[16] J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803–821, 1993.

[17] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[18] H. Bozdogan. Model selection and Akaike's Information Criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, 52:345–370, 1987.

[19] H. Bozdogan. On the information-based measure of covariance complexity and its application to the evaluation of multivariate linear models. *Communications in Statistics, Theory and Methods*, 19:221–278, 1990.

[20] G. Schwarz. Estimating the dimension of a model. *Annals of statistics*, 6(2):461–464, 1978.

[21] C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):719–725, 2000.

[22] H. Hamdan and J. Wu. EM algorithm of spherical models for binned data. In *IEEE International Symposium on Signal Processing and Information Technology*, pages 99–105, Bilbao, Spain, 14-17 december 2011.

[23] J. Wu and H. Hamdan. Parsimonious Gaussian mixture models of diagonal family for binned data clustering: mixture approach. In *IEEE International Symposium on Computational Intelligence and Informatics*, pages 385–390, Budapest, Hungary, 21-22 november 2011.

[24] J. Wu and H. Hamdan. Parsimonious Gaussian mixture models of general family for binned data clustering: mixture approach. In *IEEE International Symposium on Applied Machine Intelligence and Informatics*, pages 283–288, Herl'any, Slovakia, 26-28 january 2012.

[25] H. Hamdan and J. Wu. Bin-EM-CEM algorithms of spherical parsimonious Gaussian mixture models for binned data clustering. In *IEEE International Conference on Intelligent Engineering Systems*, pages 187–192, Costa Rica, 19-21 june 2013.

[26] J. Wu and H. Hamdan. Bin-EM-CEM algorithms of general parsimonious Gaussian mixture models for binned data clustering. In *IEEE International Conference on Computational Cybernetics*, pages 17–22, Tihany, Hungary, 8-10 july 2013.

[27] J. Wu and H. Hamdan. Model choice for binned-EM algorithms of fourteen parsimonious Gaussian mixture models by BIC and ICL criteria. In *IEEE International Conference on System Science and Engineering*, pages 351–356, Budapest, Hungary, 4-6 july 2013.

[28] H. Hamdan and J. Wu. Model selection with BIC and ICL criteria for binned data clustering by bin-EM-CEM algorithms. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 3133–3138, Manchester, United Kingdom, 13-16 october 2013.

[29] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 2(3):241–254, 1967.

[30] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[31] Y. Dodge. *Statistical Data Analysis Based on the L B1 S-norm and Related Methods*. Statistics for industry and technology. Springer-Verlag, 2002.

[32] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice Hall, Inc., 1988.

[33] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[34] J. Aldrich. R. A. Fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3):162–176, 1997.

[35] M.J. Symons. Clustering criteria and multivariate normal mixture. *Biometrics*, 37: 35–43, 1981.

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

[37] G. Celeux and G. Govaert. Comparison of the mixture and the classification maximum likelihood in cluster analysis. *Journal of Statistical Computation and Simulation*, 47:127–146, 1993.

[38] H. Bensmail and J. J. Meulman. Model-based clustering with noise: Bayesian inference and estimation. *Journal of Classification*, pages 49–76, 2003.

[39] C. F. J. Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

[40] A. W. Moore, J. G. Schneider, and K. Deng. Efficient locally weighted polynomial regression predictions. In *International Conference on Machine Learning*, pages 236–244, 1997.

[41] G. Celeux and D. Diebolt. The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Computational Statistics Quarterly*, 2(1):73–82, 1985.

[42] A. Dasgupta and A. E. Raftery. Detecting features in spatial point processes with clutter via model-based clustering. *Journal of the American Statistical Association*, 93:294–302, 1998.

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

[44] C. Biernacki, G. Celeux, and G. Govaert. Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. *Computational Statistics & Data Analysis*, 41(3-4):561–575, 2003.

[45] P. D. Mcnicholas and T. B. Murphy. Parsimonious Gaussian mixture models. *Statistics and Computing*, 18(3):285–296, 2008.

[46] D. J. Bartholomew and M. Knott. *Latent Variable Models and Factor Analysis (Kendall's Library of Statistics*. London: Edward Arnold, second edition, 1999.

[47] Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical report, University of Toronto, 1997.

[48] G. Mclachlan and D. Peel. *Finite Mixture Models*. Wiley-Interscience, 2000.

[49] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.

[50] X. L. Meng and D. van Dyk. The EM algorithm–an old folk-song sung to a fast new tune. *Journal of the Royal Statistical Society. Series B (Methodological)*, 59 (3):511–567, 1997.

[51] G. Celeux and G. Soromenho. An entropy criterion for assessing the number of clusters in a mixture model. *Journal of Classification*, 13(2):195–212, 1996.

[52] C. Biernacki and G. Govaert. Using the classification likelihood to choose the number of clusters. *Computing Science and Statistics*, 29(2), 1997.

[53] H. Hamdan and G. Govaert. The fitting of binned data clustering to imprecise data. In *IEEE International Conference on Information & Communication Technologies: from Theory to Applications*, pages 1–6, Damascus, Syria, 19-23 april 2004.

[54] H. Hamdan. Mixture model clustering of binned uncertain data: the classification approach. In *IEEE International Conference on Information & Communication Technologies: from Theory to Applications*, pages 1645–1650, Damascus, Syria, 24-28 april 2006.

[55] A. Samé. Grouped data clustering using a fast mixture-model-based algorithm. In *IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, USA, 2009.

[56] G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Chichester: Wiley, 1997.

[57] P. Bryant. Large-sample results for optimization based clustering methods. *Journal of Classification*, pages 31–44, 1991.

[58] H. Hamdan. *Développement de méthodes de classification pour le contrôle par émission acoustique d'appareils à pression*. PhD thesis, Université de Technologie de Compiègne, 2005.

[59] C. Biernacki. *Choix de modèles en classification*. PhD thesis, Université de Technologie de Compiègne, 1997.

[60] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

[61] K. P. Burnham and D. R. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer-Verlag, second edition, 2002.

[62] J.H. Wolfe. Pattern clustering by multivariate mixture analysis. *Multivariate Behavioral Research*, 5:329–350, 1970.

[63] C. Biernacki, G. Celeux, and G. Govaert. An improvement of the NEC criterion for assessing the number of clusters in a mixture model. *Pattern Recognition Letters*, 20(3):267–272, 1999.