A Non-MCMC Procedure for Fitting Dirichlet Process Mixture Models

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Mathematics and Statistics
University of Saskatchewan
Saskatoon

By
Zhengrong Li

Permission to Use

Abstract

Cluster analysis is concerned with partitioning cases into clusters such that the cases in a cluster are similar in terms of a set of variables. The K-Means Clustering Algorithm is a popular clustering method. It finds k clusters by choosing k data points as initial cluster centroids. Each data point is then assigned to the cluster with center that is closest to that point. In K-Means, the number of clusters has to be supplied in advance, which may be difficult in practice. A new method, the X-Means Clustering Algorithm, was proposed to solve this problem, which starts with an initial partition, then recursively runs a local K-Means in each cluster to split it until a lower Bayesian Information Criterion (BIC) value is reached compared with the previous larger cluster. However, this method would introduce a more severe local mode problem, that is, the previous inappropriate partition of cases cannot be corrected in the following local splitting. In this work, we develop a new algorithm that is based on Bayesian Dirichlet process mixture models, called the Non-MCMC DPM clustering algorithm. In the new clustering algorithm, we run the EM algorithm with all the cases to find a tentative partition, and then decide whether to accept the new partion with a criterion called DPC. We have tested our new clustering algorithm based on several simulated data sets, and found that it performs better than X-Means. We have also applied the algorithm to a real micorarray sample data set for predicting the class label (cancer or normal) based on the clustering results found by our new algorithm, and found that the prediction performance is comparable to state-of-the-art methods.

Acknowledgements

This thesis grew out of a MITACS internship project provided by MITACS and International Road Dynamics Inc. I am sincerely grateful to my supervisor, Dr. Longhai Li, for his invaluable advice and patient guidance. This thesis could not have been written without his constant help and support. I would also like to extend my appreciation to members of my committee: Dr. Chris Soteros, Dr. Bill Laverty, and Dr. Ian McQuillan, for their support and suggestions. I thank Department Mathematics and Statistics for the funding they provided through my Master degree program, and thank MITACS and International Road Dynamics Inc. for their funding towards my internship program. I am grateful to all of the professors, graduate students, and staff in the Department of Mathematics and Statistics, especially to my officemate, Qingxiang Yan, Lab Coordinator, Amos Lee and Manuela Haias, Graduate Secretary, Karen Ruston, Administrative Department Head Secretary, Margarita Santos, and Department Head, Raj Srinivasan. Lastly, and most importantly, I wish to thank my parents, Jianwei Li and Guifen Zhu. They bore me, raised me, support me, and love me. To them I dedicate this thesis.

Contents

List of Figures

Chapter 1

Introduction

Cluster Analysis is the assignment of objects (also called observations, or cases) into partitions called clusters so that the objects in the same clusters share similar characteristics ([13]). Various methods have been developed for Cluster Analysis. Some common methods are: Hierarchical Clustering Algorithm ([4]), K-Means Clustering Algorithm ([10]), Expectation-Maximization (EM) Algorithm ([3]), X-Means Clustering Algorithm ([21]), and Markov Chain Monte Carlo (MCMC) Dirichlet Process Mixture (DPM) Clustering Algorithm [see eg 19, 8, 16, 20, 22, 5, 12, 11, and the references therein]. Despite its popularity of general clustering, each of the above methods suffers several limitations. We developed a new clustering method, named *Non-Markov Chain Monte Carlo (Non-MCMC) Dirichlet Process Mixture (DPM) Clustering Algorithm*, which will be discussed throughout this thesis.

1.1 What is Cluster Analysis

Cluster analysis divides objects into groups (clusters) that behave similarly or show similar characteristics. Cluster analysis is an unsupervised learning technique because neither the number of clusters nor the groups themselves are known in advance. Cluster analysis has been used in many fields, such as engineering, economics, medicine, and market research.

Examples:

- **Medicine**

  Cluster analysis can be used to segment patients into various groups based on similarity of symptom. One group represents one symptom. A symptom is promoted to a *disease* once they find some actual causation underlying it. For example, suppose there is a study where a medical researcher has collected data on different measures of physical fitness (variables) for a sample of flu patients (observations). The researcher may want to cluster observations (patients) to detect clusters of patients with similar symptoms. At the same time, the researcher may want to cluster variables (fitness measurements) to detect clusters of measurements that appear to capture similar physical abilities.

- **Biology**

  Cluster analysis can be used to partition genes based on similar expression profile across tissues. Gene expression data is usually represented by a matrix, with rows corresponding to genes (observations), and columns corresponding to conditions (variables), experiments or time points. The content of the matrix is the expression levels of each gene under each condition. Each column contains the results obtained from a single array in a particular condition, and is called the profile of that condition. Each row vector is the expression pattern of a particular gene across all the conditions. The researcher may want to cluster observations (genes) to detect clusters of genes with similar expression profiles. For example, if a cluster of genes are all highly expressed in linear cell, but not in brain cells or skin cells, we may think they are involved in liver function.

- **Marketing**

  Cluster analysis can be used in marketing to segment the market and determine target markets. For example, for a marketing research of vacation travelling, customers can be clustered into three groups: 1) The demanders - they want top-notch service; 2) The escapists - they just want to escape from their busy lives and relax; 3) The educationalist - they want to see new things, visit new countries, or experience new cultures. These three groups are our clusters. Different marketing strategy and pricing can be determined for each individual group in order to achieve maximum profit for the marketing companies.

1.2 Review of Common Clustering Methods

There are probably over 100 published clustering algorithms at the moment. We will review some of the most prominent examples of clustering algorithms here.

Hierarchical Clustering: Hierarchical Clustering Algorithm is based on the idea that objects are more related to nearby objects than the objects that are farther away ([4]). Therefore, clusters are formed by connecting objects based on their distance. Clustering results produced by Hierarchical Clustering depend on the way distances are computed. User also needs to choose the linkage criterion to use. Common choices are single-linkage clustering, which uses the minimum of object distance), and complete linkage clustering, which uses the maximum of object distances. While Hierarchical Clustering method is quite easy to understand, they are not very robust towards outliers. Outliers will either show up as additional clusters or even cause other clusters to merge.

K-Means Clustering Algorithm: K-Means Clustering Algorithm is a method of cluster analysis which aims to partition n objects into k clusters in which each object belongs to the cluster with the nearest mean, such that the squared distances from the cluster are minimized ([10]). It first chooses k objects as random as initial cluster centroids. Each object is assigned to the cluster that is closest to that object. Each cluster centroid is replaced by the mean of all objects that belongs to that cluster. Repeat this process until the cluster centroids converge. One major concern about K-Means Clustering Algorithm is that the number of clusters k needs to be defined in advance, which is hard to archive in practice. Also, clustering results are very sensitive to the choices of initial cluster centroids.

Expectation-Maximization (EM) Algorithm: EM Algorithm performs clustering by fitting a given data set with a mixture of Gaussian distributions ([3]). The data set is fitted with a fixed number of Gaussian distributions. The parameters for these Gaussian distributions are randomly chosen. Each data point is assigned to the Gaussian distribution that it has the highest probability of coming from. Mean and variance for each Gaussian distribution are then iteratively updated until they converge to a local optimal. EM Algorithm has its limitations as well. Similar to K-Means Algorithm, EM algorithm requires user to define the number of clusters (number of Gaussian distributions in this case), and sometimes it is hard to fit the data set with a mixture of Gaussian distributions.

X-Means Clustering Algorithm: X-Means Clustering Algorithm is an improved version of K-Means Clustering Algorithm. Instead of forming clusters using distance, it searches the cluster locations and number of clusters by optimizing the Bayesian Information Criterion (BIC) ([21]). X-Means algorithm starts with splitting the data set into a certain number of clusters which equal the lower bound of the number of clusters using the K-Means Clustering Algorithm. It recursively splits each centroid into two children by running a *local* K-Means Clustering Algorithm until the splitting results in a lower BIC value or reach the upper bound of the number of clusters. One major limitation of X-Means Clustering Algorithm is the local mode problem, that is, it does not have the ability to correct previous inappropriate partitions. Once an inappropriate partition is formed in the splitting process, it stays there forever.

Markov Chain Monte Carlo (MCMC) Dirichlet Process Mixture (DPM) Clustering Algorithm: During the past decade, methods based on fitting Dirichlet process mixture models have become very popular for cluster analysis in many different areas, [see eg 19, 8, 16, 20, 22, 5, 12, 11, and the references therein]. In DPM models, one treats all the possible partitions with varying numbers of clusters as unknown "parameters", and assigns them a joint prior distribution, called Dirichlet process prior, which penalizes the partitions with overly large number of clusters. One then simulates Markov chains for sampling from the posterior distribution of all possible partitions, as well as other parameters used for modelling the distribution of original data. An attractiveness of fitting DPM models is an automatic mechanism for finding an appropriate number of clusters. However, it is difficult to use MCMC samples for cluster analysis once the label switching problem ([25]) has occurred during MCMC simulation, in which the identities of partitions are switched to another permutation of the identities from one iteration to another iteration. The new method proposed in this thesis is also based on DPM models, but instead of MCMC simulation, we use a sophisticated optimization method for searching a local mode of the posterior of "all possible partitions", called Dirichlet Process Criterion (DPC). There will be not label switching problem in our algorithm, and the fitting results can be used directly for cluster analysis.

1.3 Contribution of This Thesis

Each of the above clustering methods suffers several limitations as discussed in Section (1.2). We developed a new clustering method, named *Non-Markov Chain Monte Carlo (Non-MCMC) Dirichlet Process Mixture (DPM) Clustering Algorithm*, with the aim to propose solutions for the above limitations. In short, the Non-MCMC DPM Clustering Algorithm starts with considering the entire data set as 1 cluster. It first fits the data set with a Bayesian DPM model, and calculate the DPC value for this model. Then it splits the data set into 2 clusters using Principal Component Analysis. EM algorithm is applied over the new partition to update the cluster assignment. DPC value for the new partition is also calculated. If the new DPC value is greater than the old DPC value, new partition is accepted. Then we keep splitting each cluster one by one until no more new partitions can be formed. The advantages of Non-MCMC DPM Clustering Algorithm over the methods discussed in section (1.2) are: First of all, it has the ability to determine the optimum number of clusters without any user inputs. Secondly, it uses DPC as a model selection criterion, which is more accurate than the linkage criterion used in Hierarchical Clustering Algorithm, sum-of-squares criterion used in K-Means Clustering Algorithm, and BIC used in X-Means Clustering Algorithm. Thirdly, it uses EM algorithm over the entire data set when a split is performed, and the cluster assignment is updated every time after running EM algorithm. Therefore, if some data points are assigned to the inappropriate clusters in the previous steps, it has the ability to re-assign them to the appropriate clusters in the following steps. Last but not least, it can search a partition that has reached a local mode. We will apply the Non-MCMC DPM Clustering Algorithm over several simulated data sets and a sample prostate cancer data set. The clustering results obtained by Non-MCMC DPM Clustering Algorithm and the comparison with other clustering methods will be discussed in this thesis.

1.4 Outline of the Remainder of the Thesis

Chapter (2) is a preliminary discussion for Hierarchical Clustering Algorithm. Chapter (3) focuses on how to use EM Algorithm and X-Means Algorithm to perform clustering. We also provide a discussion on their limitations. Chapter (4) discusses the MCMC DPM Clustering Algorithm and its limitations. We then introduce the Non-MCMC DPM Clustering Algorithm, and its applications to several simulated data sets and a prostate microarray data set. Comparisons between Non-MCMC DPM Clustering Algorithm and other existing methods are also presented in this chapter. Chapter (5) begins with a summary of the thesis. It then provides directions that remain to be explored.

Chapter 2
Hierarchical Clustering

The most common algorithm for cluster analysis is Hierarchical Clustering. Assume we have n observations. In Hierarchical Clustering, we start with n clusters, one for each observation. At each step, we merge an observation or a cluster of observations into another cluster. We repeat this process until all of the observations are merged into a single cluster. We can also perform hierarchical clustering in a reverse order. That is, we start with a single cluster containing all n observations. At each step, we split a cluster into two clusters until we have n clusters with a single observation each.

2.1 Similarity Measurement

The goal of cluster analysis is grouping similar observations together. How can we determine the similarity of two observations? The most common method used here is to measure the distance between two observations. Common distance functions are listed as follows from [1]:

- **Euclidean Distance**

  The Euclidean distance between two points $x = (x_1, x_2, \ldots, x_p)'$ and $y = (y_1, y_2, \ldots, y_p)'$ is defined as

  $$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_p - y_p)^2} = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2}. \qquad (2.1)$$

- **Mahalanobis Distance**

  The Mahalanobis distance between two points $x = (x_1, x_2, \ldots, x_p)'$ and $y = (y_1, y_2, \ldots, y_p)'$

is defined as

$$d(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)}. \tag{2.2}$$

Where S is the covariance matrix of x and y.

- **Absolute Distance**

  The absolute distance between two points $x = (x_1, x_2, \ldots, x_p)'$ and $y = (y_1, y_2, \ldots, y_p)'$ is defined as

$$d(x,y) = \sum_{i=1}^{p}(|x_i - y_i|). \tag{2.3}$$

- **0 - 1 Distance for Discrete Data**

  The absolute distance between two discrete data points $x = (x_1, x_2, \ldots, x_p)'$ and $y = (y_1, y_2, \ldots, y_p)'$ is defined as

$$d(x,y) = \sum_{i=1}^{p} \mathrm{I}(x_i \neq y_i). \tag{2.4}$$

2.2 Methods Of Hierarchical Clustering

Hierarchical clustering has two approaches. As discussed in [13], the first approach, called *Agglomerative* hierarchical approach, starts with n clusters, and merges two closest clusters into one new cluster at each step. The end result is a single cluster containing all the data points. The second approach is called *Divisive* hierarchical approach. This approach starts with a single cluster containing the entire data set, splits a cluster into two new clusters at each step, and ends up with n clusters which contain one data point each.

Since the agglomerative hierarchical approach requires to find two closest clusters at each step, the question arises here, how to determine which of the two clusters are closest. We can solve this question by measuring the similarity of two clusters. Then the steps of agglomerative hierarchical clustering become:

1. Assign every case to its own cluster.

2. Repeat the following until all cases are in one cluster.

   a. Find two clusters with the biggest similarity.

   b. Replace these two clusters with one cluster containing all cases from both clusters.

There are several different different agglomerative hierarchical clustering methods because of the different ways of defining similarity between clusters.

- **Single Linkage Clustering**

  Single linkage method is one of the easiest agglomerative hierarchical clustering methods. It is also called *nearest neighbor* method. Assume we have two clusters A and B. The distance between them is defined as the minimum distance between an observation in A and a observation in B as follows:

$$D(A, B) = \min\{d(y_i, y_j)\}, \tag{2.5}$$

  where observation y_i is in cluster A, observation y_j is in cluster B, and $d(y_i, y_j)$ is the Euclidean distance between y_i and y_j.

  In other words, the distance between two clusters is given by the value of the shortest length between the clusters. At each step, the clusters A and B with the minimum value of $D(A, B)$ are merged.

  The results of the hierarchical clustering method can be described using a diagram called *dendrogram*. This diagram simply shows which two clusters are merged at each step. We illustrate the *dendrogram* using the following example.

  We are interested in the returns of various sorts of US investments for the years 1984 to 1995. The return is expressed as the value at the end of the year of one dollar invested at the beginning of the year, after adjusting for inflation. (For example, if \$1.00 invested at the beginning of a year produces \$1.43 at the end of the year, but inflation during that year was 10%, the number recorded in the file would be 1.43/1.10 = 1.30.)

The variables are as follows:

year Year, from 1949 to 1995

gbonds Return for long-term government bonds

gbills Return for short-term government bills

cbonds Return for long-term corporate bonds

indust Return for industrial stocks

transp Return for transportation stocks

util Return for utility stocks

finance Return for finance stocks

The data are given in the following table:

year	gbonds	gbills	cbonds	indust	transp	util	finance
1984	1.2825	1.0371	1.3139	1.2577	1.3093	1.2992	1.3837
1985	1.2636	1.0496	1.2709	1.1734	1.0592	1.2704	1.0689
1986	0.9102	1.0139	0.884	1.0414	0.9523	0.9246	0.8023
1987	1.0429	1.0242	1.1069	1.1105	1.168	1.1362	1.1273
1988	1.139	1.0363	1.1329	1.2361	1.182	1.4013	1.2458
1989	0.997	1.0152	1.0172	0.9343	0.8257	0.9365	0.7688
1990	1.131	1.0233	1.1639	1.2681	1.4356	1.1024	1.4348
1991	1.062	1.0058	1.0946	1.0275	1.0697	1.0547	1.2042
1992	1.155	1.0029	1.1761	1.0618	1.1544	1.1334	1.0845
1993	0.906	1.0166	0.8778	1.0114	0.8053	0.8882	0.9359
1994	1.2422	1.0298	1.3107	1.312	1.3471	1.2377	1.4958
1995	0.9735	1.0175	0.9743	1.1898	1.1232	1.0173	1.3073

Table 2.1: US Investment Table (1984 - 1995)

The dendrogram by using single linkage clustering is shown as follows:

- **Average Linkage Clustering**

Assume there are two clusters, cluster A and B. There are N_A number of observations in cluster A, and N_B number of observations in cluster B. Using the average linkage

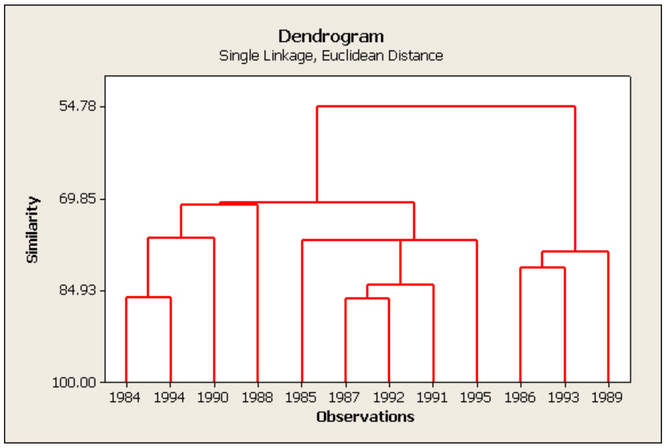

Figure 2.1: Dendrogram for Single Linkage Clustering

method, the distance between A and B is defined as:

$$D(A, B) = \frac{1}{N_A N_B} \sum_{i=1}^{N_A} \sum_{j=1}^{N_B} d(y_i, y_j), \tag{2.6}$$

where observation y_i is in cluster A, observation y_j is in cluster B, and $d(y_i, y_j)$ is the Euclidean distance between y_i and y_j.

At each step, the clusters A and B with the minimum value of $D(A, B)$ are merged.

Using the previous example, the dendrogram shows the steps in this method:

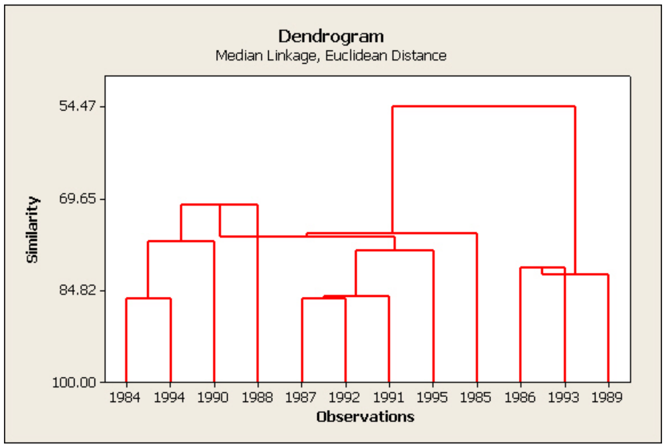

Figure 2.2: Dendrogram for Average Linkage Clustering

- **Complete Linkage Clustering**

  Complete linkage method is also called the *farthest neighbour* method. Assume we have two clusters A and B. The distance between them is defined as the maximum distance between an observation in A and an observation in B as follows:

$$D(A,B) = \max\{d(y_i, y_j)\}, \tag{2.7}$$

where observation y_i is in cluster A, observation y_j is cluster B, and $d(y_i, y_j)$ is the Euclidean distance between y_i and y_j. At each step, the clusters A and B that have the minimum value of $D(A,B)$ are merged. Using the previous example, the dendrogram shows the steps in this method:

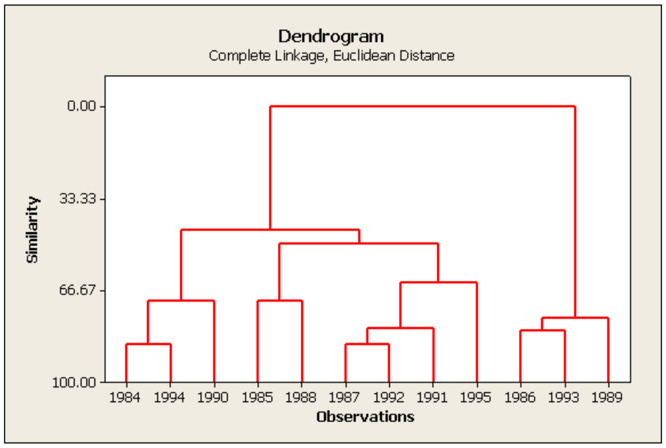

Figure 2.3: Dendrogram for Complete Linkage Clustering.

- **Centroid Clustering**

Assume there are two clusters, cluster A and B. There are N_A number of observations in cluster A, and N_B number of observations in cluster B. Let $\overline{y}_A$ and $\overline{y}_B$ be the means for the observations in A and B, respectively. In other words, $\overline{y}_A = \frac{\sum_{i=1}^{N_A} y_i}{N_A}$, and $\overline{y}_B = \frac{\sum_{j=1}^{N_B} y_j}{N_B}$. Then the distance between A and B is defined as:

$$D(A,B) = d(\overline{y}_A, \overline{y}_B) \tag{2.8}$$

That is, the Euclidean distance between $\overline{y}_A$ and $\overline{y}_B$. Using the previous example, the dendrogram shows the steps in this method:

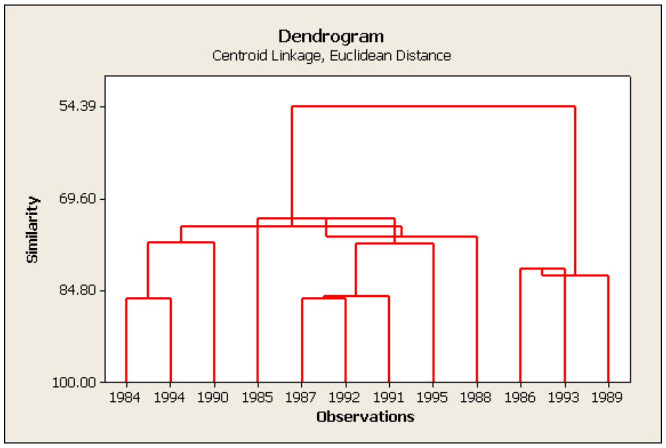

Figure 2.4: Dendrogram for Centroid Clustering

- **Ward's Method**

Ward proposed a clustering procedure seeking to partition the data using the sum of squares criterion. Assume there are two clusters, cluster A and B. There are N_A number of observations in cluster A, and N_B number of observations in cluster B. In Ward's method, the within cluster sum of squares and between cluster sum of squares are calculated as follows:

$$\text{Sum of squares within cluster } A\text{: } \mathrm{D}_A = \sum_{i=1}^{N_A} (y_i - \overline{y}_A)'(y_i - \overline{y}_A),$$

$$\text{Sum of squares within cluster } B\text{: } \mathrm{D}_B = \sum_{j=1}^{N_B} (y_j - \overline{y}_B)'(y_j - \overline{y}_B),$$

$$\text{Sum of squares between cluster } A \text{ and } B\text{: } \mathrm{D}_{A\cup B} = \sum_{i=1}^{N_A+N_B} (y_i - \overline{y}_{AB})'(y_i - \overline{y}_{AB}),$$

where $\overline{y}_A$ is the means for the observations in A, $\overline{y}_B$ is the means for the observations in B, and $\overline{y}_{AB} = (N_A\overline{y}_A + N_B\overline{y}_B)/(N_A + N_B)$.

The goal of Ward's method is to minimize the increase in sum of squares, which is equal to $\mathrm{D}_{A\cup B} - \mathrm{D}_A - \mathrm{D}_B$. Let $\mathrm{D}_{A,B} = \mathrm{D}_{A\cup B} - \mathrm{D}_A - \mathrm{D}_B$, then

$$\begin{aligned}\mathrm{D}_{A,B} &= \mathrm{D}_{A\cup B} - \mathrm{D}_A - \mathrm{D}_B \qquad (2.9)\\ &= \frac{N_A N_B}{N_A + N_B}(\overline{y}_A - \overline{y}_B)'(\overline{y}_A - \overline{y}_B) \qquad (2.10)\\ &= \frac{N_A N_B}{N_A + N_B}\mathrm{D}_{centroid}(\overline{y}_A, \overline{y}_B) \qquad (2.11)\end{aligned}$$

Ward's method is similar to the centroid method in Section (2.2). The only difference is the coefficient $\frac{N_A N_B}{N_A+N_B}$ in the Ward's method. If N_A or N_B increases, $\frac{N_A N_B}{N_A+N_B}$ increases as well. As a result, Ward's method is more likely to merge small clusters first compared to the centroid method.

Using the previous example, the dendrogram shows the steps in this method:

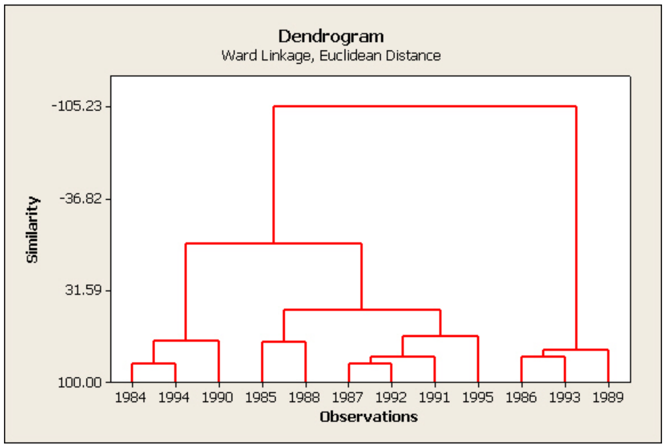

Figure 2.5: Dendrogram for Ward Clustering

Chapter 3
Model-based Clustering Methods

In this chapter, I review classic clustering methods based on finite mixture models. I will review finite mixture models, EM algorithm for fitting mixture models, and provide some examples. The limitation of classic clustering methods based on finite mixture models is that we must provide a number as to how many mixture components we have, which is often unavailable in practice. I will then review a recently proposed method called X-Means, which can determine the number of components automatically.

3.1 General Description of Finite Mixture Models

In model based clustering, the data we observed are assumed to be a mixture of random samples from different populations. Assume $\mathbf{X}$ is the sample data set, $\mathbf{N}$ is the sample data size, $\mathbf{x}^{(i)}$ is a single data point in the sample data set, which can have one or multiple variables, for $i = 1, \ldots, N$, K is the number of clusters, and $z^{(i)}$ is a component label for $\mathbf{x}^{(i)}$, which is assumed to take values from $1, \ldots, K$, Z is a vector that contains all values of $z^{(i)}$, for $i = 1, \ldots, N$. $P(z^{(i)} = k) = \pi_k$ is the probability that data point $\mathbf{x}^{(i)}$ comes from cluster k , for $k = 1, \ldots, K$, $\boldsymbol{\pi}$ is a vector that contains all values of π_k, $\boldsymbol{\theta}_k$ is the parameter(s) for cluster k, for $k = 1, \ldots, K$, and $\boldsymbol{\theta}$ is a vector that contains all values of $\boldsymbol{\theta}_k$.

Given $z^{(i)} = k$, let $P(\mathbf{x}^{(i)}|\pi_k, \boldsymbol{\theta}_k)$ be the density function of an observation $\mathbf{x}^{(i)}$ from the kth cluster, then

$$P(\mathbf{x}^{(i)}|\pi_k, \boldsymbol{\theta}_k) = P(\mathbf{x}^{(i)}|z^{(i)} = \mathrm{k}) \tag{3.1}$$

The marginal distribution of $\mathbf{x}^{(i)}$ is

$$\begin{aligned} P(\mathbf{x}^{(i)}) &= \sum_{k=1}^{K} P(\mathbf{x}^{(i)}|z^{(i)} = \mathrm{k}) P(z^{(i)} = \mathrm{k}) && (3.2) \\ &= \sum_{k=1}^{K} P(\mathbf{x}^{(i)}|\pi_k, \boldsymbol{\theta}_k)\pi_k && (3.3) \end{aligned}$$

Suppose we are given a data set for patients who have cancer. We are also given the information that the stage of cancer that each patient is at. Apply the clustering method over this data set. Let

- K: Number of clusters obtained.
- y: An indicator variable that indicates whether the patient is in early stage of cancer or critical stage of cancer. Assume $y = 0$ means the patient is in early stage of cancer, while $y = 1$ means the patient is in critical stage of cancer
- $\mathbf{P}_k^{(y=1)}$: Probability of $y = 1$ for all data points inside Cluster k, for $k = 1, \ldots, K$.

- $\mathbf{X}_N$: Data set for a new patient.
- π_k^N: Probability that this new patient belongs to Cluster k, for $k = 1, \ldots, K$.
- C_N: Cluster for this new patient.
- $\mathbf{P}_N^{(y=1)}$: Probability that this new patient is in critical stage of cancer.

We can predict the probability that this new patient is in critical stage of cancer by

$$\begin{aligned}
\mathbf{P}_N^{(y=1)} &= P(y = 1|\mathbf{X}_N) && (3.4)\\
&= \sum_{k=1}^{K} P(y = 1|\mathbf{X}_N, C_N = k)P(C_N = k|\mathbf{X}_N) && (3.5)\\
&= \sum_{k=1}^{K} \mathbf{P}_k^{(y=1)}\pi_k^N && (3.6)
\end{aligned}$$

3.2 EM Algorithm for Fitting Finite Mixture Models

An *expectation-maximization (EM) algorithm* is a general approach to find maximum likelihood estimates of parameters in probability models, where the data for the models are incomplete ([18]). Given the probability model stated in Section (3.1), the *complete* data are considered to be $(\mathbf{x}, \mathbf{z})$. We can observe $\mathbf{x}$, but we can't observe $\mathbf{z}$ since they are incomplete. EM algorithm can help us find reasonable estimates for $\mathbf{z}$. Given the estimates of $\mathbf{z}$, we can find the estimate of the distribution $\mathbf{x}$.

The steps are as follows:

- Step 1: Call the current stage as Stage t (t starts with 1). Let $\hat{\boldsymbol{\pi}}^{(t)} = (\hat{\pi}_1^{(t)}, \ldots, \hat{\pi}_K^{(t)})$ and $\hat{\boldsymbol{\theta}}^{(t)} = (\hat{\boldsymbol{\theta}}_1^{(t)}, \ldots, \hat{\boldsymbol{\theta}}_K^{(t)})$ be the initial estimates for $\boldsymbol{\pi}$ and $\boldsymbol{\theta}$ at Stage t.
- Step 2: Repeat the following until the differences between $\hat{\boldsymbol{\pi}}^{(t)}$ and $\hat{\boldsymbol{\pi}}^{(t+1)}$, and $\hat{\boldsymbol{\theta}}^{(t)}$ and $\hat{\boldsymbol{\theta}}^{(t+1)}$ are less than 0.00001.
  - E-step (Expectation step):

    In this step, we need to compute the value of $P(z^{(i)} = k|\mathbf{x}^{(i)}, \hat{\boldsymbol{\pi}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)})$ using the values of $\hat{\boldsymbol{\pi}}^{(t)}$ and $\hat{\boldsymbol{\theta}}^{(t)}$.

Given $\hat{\boldsymbol{\pi}}^{(t)}$ and $\hat{\boldsymbol{\theta}}^{(t)}$, compute $P(z^i = k|\mathbf{x}^i, \hat{\boldsymbol{\pi}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)})$ for $i = 1, \ldots, N$, and $k = 1, \ldots, K$.

By **Bayes' Rule**,

$$P(z^{(i)} = k|\mathbf{x}^{(i)}, \hat{\boldsymbol{\pi}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)}) = \frac{P(\mathbf{x}^{(i)}|z^{(i)} = k, \hat{\boldsymbol{\pi}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)})P(z^{(i)} = k|\hat{\boldsymbol{\pi}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)})}{\sum_{k=1}^{K} P(\mathbf{x}^{(i)}|z^{(i)} = k, \hat{\boldsymbol{\pi}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)})P(z^{(i)} = k|\hat{\boldsymbol{\pi}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)})} \tag{3.7}$$

$$\propto P(\mathbf{x}^{(i)}|z^{(i)} = k, \hat{\boldsymbol{\pi}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)})\hat{\pi}_k^{(t)} \tag{3.8}$$

Let's denote $P(z^{(i)} = k|\mathbf{x}^{(i)}, \hat{\boldsymbol{\pi}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)})$ by $\hat{\pi}_k^{(i)}$, which is called the *updated class allocation probability*, for $i = 1, \ldots, N$, and $k = 1, \ldots, K$.

– M-step (Maximization step):

In this step, we need to compute the new estimates for $\boldsymbol{\pi}^{(t)}$ and $\boldsymbol{\theta}^{(t)}$, denoted by $\hat{\boldsymbol{\pi}}^{(t+1)}$ and $\hat{\boldsymbol{\theta}}^{(t+1)}$, that maximize the expected log likelihood function found in E-step. Then, we need to replace the values of $\hat{\boldsymbol{\pi}}^{(t)}$ and $\hat{\boldsymbol{\theta}}^{(t)}$ with the values of $\hat{\boldsymbol{\pi}}^{(t+1)}$ and $\hat{\boldsymbol{\theta}}^{(t+1)}$.

Let's denote the expected log likelihood function as $\hat{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\pi}; \mathbf{x}, \mathbf{z}, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\pi}}^{(t)})$. Then

$$\hat{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\pi}; \mathbf{x}, \mathbf{z}, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\pi}}^{(t)}) = E_{\mathbf{z}|\mathbf{x}, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\pi}}^{(t)}} \left(\log(P(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}, \boldsymbol{\pi}))\right) \tag{3.9}$$

$$= \sum_{i=1}^{N} E_{z^{(i)}|\mathbf{x}^{(\mathbf{i})}, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\pi}}^{(t)}} \log\left(P(\mathbf{x}^{(\mathbf{i})}, z^{(i)}|\boldsymbol{\theta}, \boldsymbol{\pi})\right) \tag{3.10}$$

Since

$$E_{z^{(i)}|\mathbf{x}^{(\mathbf{i})}, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\pi}}^{(t)}}(\log(P(\mathbf{x}^{(\mathbf{i})}, z^{(i)}|\boldsymbol{\theta}, \boldsymbol{\pi}))) = E_{z^{(i)}|\mathbf{x}^{(\mathbf{i})}\hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\pi}}^{(t)}} \left(\log(P(\mathbf{x}^{(\mathbf{i})}|z^{(i)}, \boldsymbol{\theta})P(z^{(i)}|\boldsymbol{\pi}))\right) \tag{3.11}$$

$$= \sum_{k=1}^{K} \left(\log(P(\mathbf{x}^{(\mathbf{i})}|z^{(i)} = k, \boldsymbol{\theta}_k) + \log(\pi_k)\right) \hat{\pi}_k^{(i)} \tag{3.12}$$

Then we have

$$\hat{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\pi}; \mathbf{x}, \mathbf{z}, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\pi}}^{(t)}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \big(\log(P(\mathbf{x}^{(\mathbf{i})}|z^{(i)} = k, \boldsymbol{\theta}_k) + \log(\pi_k)\big)\, \hat{\pi}_k^{(i)} \tag{3.13}$$

$$= \sum_{k=1}^{K} \log(\pi_k) \sum_{i=1}^{N} \hat{\pi}_k^{(i)} + \sum_{i=1}^{N} \log\big(P(\mathbf{x}^{(\mathbf{i})}|z^{(i)} = k, \boldsymbol{\theta}_k)\big)\, \hat{\pi}_k^{(i)} \tag{3.14}$$

Therefore,

$$\hat{\boldsymbol{\pi}}^{(t+1)} = \underset{\pi}{\operatorname{argmax}} \sum_{k=1}^{K} \left(\log(\pi_k)^{(t)} \sum_{i=1}^{N} \hat{\pi}_k^{(i)} \right) \tag{3.15}$$

$$= \left(\sum_{i=1}^{N} \hat{\pi}_1^{(i)}/N, \ldots, \sum_{i=1}^{N} \hat{\pi}_K^{(i)}/N \right) \tag{3.16}$$

and

$$\hat{\boldsymbol{\theta}}^{(t+1)} = \underset{\boldsymbol{\theta}_k}{\operatorname{argmax}} \sum_{i=1}^{N} \log\big(P(\mathbf{x}^{(\mathbf{i})}|z^{(i)} = k, \boldsymbol{\theta}_k)\big)\, \hat{\pi}_k^{(i)} \tag{3.17}$$

$$= \underset{\boldsymbol{\theta}_k}{\operatorname{argmax}} \hat{\mathcal{L}}(\boldsymbol{\theta}_k) \tag{3.18}$$

3.2.1 EM Algorithm for Fitting Mixture of Normal Distributions

Given the probability model stated in Section (3.1), assume $P(\mathbf{x}^{(\mathbf{i})}|\boldsymbol{\theta}_k)$ forms independent normal distributions, with $\boldsymbol{\theta}_k = (\boldsymbol{\mu_k}, \boldsymbol{\sigma_k^2})$, where $\boldsymbol{\mu_k} = (\mu_{k1}, \ldots, \mu_{kp})$, $\boldsymbol{\sigma_k^2} = (\sigma_{k1}^2, \ldots, \sigma_{kp}^2)$, and $k = 1, \ldots, K$. That is, $P(\mathbf{x}^{(\mathbf{i})}|\boldsymbol{\theta}_k)$ is assumed as

$$P(\mathbf{x}^{(\mathbf{i})}|\boldsymbol{\theta}_k) = \prod_{j=1}^{P} N(x_j^{(i)}|\mu_{kj}, \sigma_{kj}^2) \tag{3.19}$$

$$= \prod_{j=1}^{P} \frac{1}{\sqrt{2\pi}} \sigma_{kj}^{-1} \exp\left(-\frac{(x_j^{(i)} - \mu_{kj})^2}{2\sigma_{kj}^2} \right) \tag{3.20}$$

Therefore,

$$\hat{\mathcal{L}}(\boldsymbol{\theta}_k) = \sum_{i=1}^{N} \log\left(P(\mathbf{x}^{(\mathbf{i})}|z^{(i)} = k, \boldsymbol{\theta}_k)\right)\hat{\pi}_k^{(i)} \tag{3.21}$$

$$= \sum_{i=1}^{N} \hat{\pi}_k^{(i)} \sum_{j=1}^{P} \log\left(P(x_j^{(i)}|z^{(i)} = k, \boldsymbol{\theta}_{kj})\right) \tag{3.22}$$

$$= \sum_{j=1}^{P} \sum_{i=1}^{N} \hat{\pi}_k^{(i)} \log\left(P(x_j^{(i)}|z^{(i)} = k, \boldsymbol{\theta}_{kj})\right) \tag{3.23}$$

$$= \sum_{j=1}^{P} \hat{\mathcal{L}}\left(\mu_{kj}, \sigma_{kj}^2\right) \tag{3.24}$$

Therefore, maximizing $\hat{\mathcal{L}}(\boldsymbol{\theta}_k)$ is the same as maximizing each $\sum_{j=1}^{P} \hat{\mathcal{L}}\left(\mu_{kj}, \sigma_{kj}^2\right)$. Since

$$\sum_{j=1}^{P} \hat{\mathcal{L}}\left(\mu_{kj}, \sigma_{kj}^2\right) = -\frac{1}{2} \sum_{i=1}^{N} \hat{\pi}_k^{(i)} \left(\log(2\pi) + \log(\sigma_{kj}^2) + \frac{(x_j^{(i)} - \mu_{kj})^2}{\sigma_{kj}^2}\right) \tag{3.25}$$

$$= -\frac{1}{2}\left(\log(\sigma_{kj}^2) \sum_{i=1}^{N} \hat{\pi}_k^{(i)} + \sum_{i=1}^{N} \hat{\pi}_k^{(i)} \frac{(x_j^{(i)} - \mu_{kj})^2}{\sigma_{kj}^2} + \log(2\pi) \sum_{i=1}^{N} \hat{\pi}_k^{(i)}\right) \tag{3.26}$$

Set $\frac{\partial \sum_{j=1}^{P} \hat{\mathcal{L}}\left(\mu_{kj}, \sigma_{kj}^2\right)}{\partial \mu_{kj}} = \sum_{i=1}^{N} \hat{\pi}_k^{(i)} \frac{2\left(x_j^{(i)} - \mu_{kj}\right)}{\sigma_{kj}^2}$ to 0, and solve for μ_{kj}, we obtain at Stage t + 1:

$$\mu_{kj}^{(t+1)} = \frac{\sum_{i=1}^{N} \hat{\pi}_k^{(i)} x_j^{(i)}}{\sum_{i=1}^{N} \hat{\pi}_k^{(i)}} \tag{3.27}$$

Set $\dfrac{\partial \sum_{j=1}^{P} \hat{\mathcal{L}}(\mu_{kj},\sigma_{kj}^2)}{\partial \sigma_{kj}^2} = -\frac{1}{2}\left(\frac{1}{\sigma_{kj}^2} \sum_{i=1}^{N} \hat{\pi}_k^{(i)} - \sum_{i=1}^{N} \hat{\pi}_k^{(i)} \frac{\left(x_j^{(i)} - \mu_{kj}\right)^2}{(\sigma_{kj}^2)^2} \right)$ to 0, and solve for σ_{kj}^2, we have

$$\sigma_{kj}^{(t+1)} = \sqrt{\frac{\sum_{i=1}^{N} \hat{\pi}_k^{(i)} \left(x_j^{(i)} - \mu_{kj}^{(t+1)}\right)^2}{\sum_{i=1}^{N} \hat{\pi}_k^{(i)}}} \tag{3.28}$$

3.2.2 A Demonstration of EM Algorithm

The detailed steps of EM algorithm for clustering can be demonstrated by the following example:

- **Step 1.** Given a data set, select a value for the number of clusters K ($K = 3$ in this example), and also select initial values for the unknown parameters, i.e, the values to define the positions of cluster centroids. For each data point, calculate the probability that this data point belongs to each cluster, and assign this data point to the cluster that it has highest probability of coming from. See Figure (3.1).

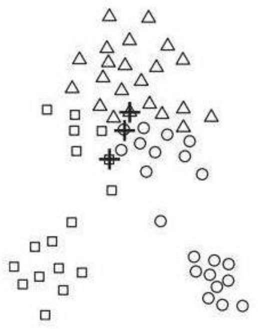

Figure 3.1: Initialize number of clusters and cluster centroids.

- **Step 2.** Run EM algorithm among the entire data set. The EM algorithm will stop when the position of cluster centroids never change again. At each iteration, the positions of cluster centroids are updated, and the data points are re-assigned to corresponding cluster using the same method stated in Step 1. Figure (3.2) show the changes at each iteration.

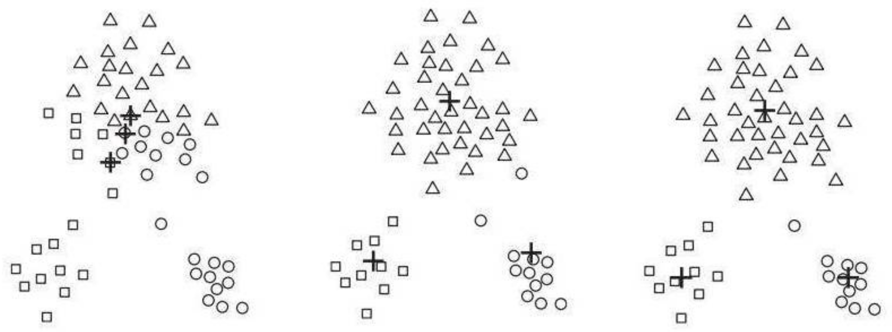
(a) Iteration 1 (b) Iteration 2 (c) Iteration 3

Figure 3.2: Each iteration of EM algorithm

3.2.3 Limitations Of EM Algorithm For Clustering

Although EM algorithm for clustering is one of the simplest unsupervised learning algorithms that solves the well known clustering problem, it suffers the following major shortcomings:

- Number of clusters K has to be supplied by the user, and there is no general theoretical solution to find the optimal value of K for any given data set.
- Clustering result is very sensitive to the initial choice of parameter $\boldsymbol{\theta}$. A bad choice of initial $\boldsymbol{\theta}$ can have a huge impact on the clustering performance.

3.3 X-Means Algorithm

Since EM algorithm has several drawbacks, a new clustering method, called *X-MEANS*, was developed by [21]. The biggest advantage of X-Means algorithm over EM algorithm is that X-Means can search the cluster locations and the number of clusters K by optimizing the Bayesian Information Criterion (BIC).

3.3.1 Bayesian Information Criterion

Given a data set, we can use a series of models to represent this data set, where different models have different numbers of parameters. Among these models, which one is the best to represent the data set? Here the concept of Bayesian Information Criterion kicks in. In Statistics, *Bayesian Information Criterion (BIC)* is a criterion for model selection among a series of parametric models that have different numbers of parameters. If we use EM algorithm to find the maximum mixture likelihood to estimate model parameters, the fit of a mixture model for a given data set can be improved by adding additional parameters, and likelihood can also be increased by adding more parameters, which may create the problem of over-fitting. The BIC resolves this problem by adding a penalty term to the log likelihood to penalize the complexity of the model.

Mathematically, if we let x be the sample data, n be the number of data points in x (the sample data size), k be the number of free parameters in the model that need to be estimated, θ be the parameter(s) in the model, and L be the maximized value of the likelihood function for the estimated model, then the BIC formula is

$$BIC = \log\left(L(x|\theta)\right) - \frac{1}{2}k\log(n) \tag{3.29}$$

The number of explanatory variables and unexplained variation in the dependent variable decreases the value of BIC. Therefore, a model with higher BIC value means this model better fits the data, or it contains less explanatory variables. Hence, in terms of model selection, the model with highest BIC value is the one we prefer.

3.3.2 Principal Component Analysis

Principle Component Analysis (PCA) is a mathematical operation that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, i.e., the most significant relationship between the data dimensions, and each succeeding component accounts for as much of the remaining variability as possible. Since data patterns are hard to visualize in high dimension, PCA is a powerful tool for analysing data in high dimensions. Once we find the data patterns, we can reduce the number of dimensions for the data set without losing much of the information inside the data set. More information about PCA can be found from [14].

The steps to apply PCA on a data set can be described as follows:

- **Step 1.** Given a data set with high dimensions, subtract the empirical mean from each data dimension, i.e., center the data. A new data set with mean zero for each dimension is produced. Let us call this data set as adjusted data set.

- **Step 2.** Calculate the covariance matrix for the adjusted data set.

- **Step 3.** Calculate the eigenvalues and eigenvectors for the covariance matrix.

- **Step 4.** Order the eigenvectors by the values of corresponding eigenvalues, from greatest to smallest. The eigenvector with the greatest eigenvalue is the first principle component of the data set. The eigenvector with second greatest eigenvalue is the second principle component of the data set, and so on. Here we can reduce the dimensions of the data set by ignoring the principle components that are corresponded to smaller value of eigenvalues, i.e., ignore the components with less significant relationship between the data dimensions. We will lose some information about the data set if we ignore these components, but if the eigenvalues are small enough, we do not lose much of the information.

- **Step 5.** After we obtain the eigenvectors, we can derive the transformed new data set. Construct a matrix whose columns contain the eigenvectors. Let us call the matrix as rotation matrix. Then the transformed data set is:

$$\text{Transformed data} = (\text{rotation matrix}^t * \text{adjusted data set}^t)^t \tag{3.30}$$

Where $(\text{rotation matrix})^t$ is the transpose of the rotation matrix.

3.3.3 X-Means Algorithm

In general, X-Means algorithm initially splits the data into a certain number of clusters, which is called the lower bound of the number of clusters using the EM algorithm clustering. It recursively splits each centroid into two children by running a *local* EM algorithm clustering (with $K = 2$) until the splitting results in a lower BIC value or reaches the upper bound of the number of clusters.

- **Step 1.** Run EM algorithm clustering with three centroids.
- **Step 2.** Split each centroid into two children using Principal Component Analysis (PCA), which will be discussed in Section (3.3.2).
- **Step 3.** Inside each parent area, run a *local* EM algorithm clustering with number of clusters $K = 2$ for each pair of children. Here *local* means only the data points inside the given parent area are considered while running EM algorithm clustering. The data points outside of the parent area are ignored.
- **Step 4.** Inside each parent area, a new model with two centroids (a pair of children) is created, and model selection is performed for all pairs of children. If the model with two children has a higher value of BIC than the original parent model, we will replace the original parent model with the new children model. Otherwise, we will keep the original parent model.
- **Step 5.** Repeat Step 2, 3, and 4 until no more splitting is accepted or the upper bound of the number of clusters is reached.

The EM algorithm requires the user to supply the number of clusters K, and also the initial positions for the cluster centroids. X-Means algorithm has the ability to estimate K. It's a huge improvement over EM algorithm, but it does not mean X-Means algorithm is perfect. It still suffers from several problems. X-Means algorithm splits each two centroids into two children. In each parent region, a local EM algorithm is performed for each pair of children. Only points inside the parent region are considered while running the EM algorithm. Points outside of the parent region are ignored. As a result, if mistakes are made in the previous clustering, for example, assign the data point into the wrong cluster, or split a group of data points which should be grouped together into two clusters, X-Means algorithm does not have the ability to go back and correct these mistakes in the following clustering steps. Once a mistake is made, it stays there forever. Therefore, we developed a new clustering algorithm, aiming to improve EM algorithm and X-Means algorithm, which will be introduced in Chapter (4).

Chapter 4

New Non-MCMC DPM Clustering Algorithm

X-Means algorithm is an improvement over EM algorithm, it still has some drawbacks. First of all, X-Means algorithm uses BIC value as model selection criterion. Since BIC is a rough estimation, inappropriate model maybe selected during the clustering process. Secondly, it runs a local EM algorithm by only consider the observations inside each individual group; therefore, it does not have the ability to go back to correct previous incorrect partitions. We developed a Non-MCMC DPM clustering algorithm for Fitting Dirichlet Process Mixture Models, with the aim to solve the problems raised by X-Means algorithm.

4.1 Dirichlet Process Ctiterion for Selecting Number of Mixture Components

We introduce the important model selection criterion, called *Dirichlet process criterion*, shortened by DPC, throughout this thesis, which is used for measuring the goodness of different clusterings of the data points (aka data objects). Our clustering algorithm will terminate when we find a clustering of cases that seemingly optimizes Dirichlet process criterion. Briefly speaking, Dirichlet process criterion for a clustering of n data points is just the posterior of the corresponding partition of indice $\{1, \ldots, n\}$ of n data points (eg, 1,4,5/2,3,6 if $n = 6$), based on Dirichlet process prior for all possible partitions, and the data information. Mixture models based on a Dirichlet process prior for partitions of data points are often called Dirichlet process mixture (DPM) models [see eg 19, 8, 16, 20, 22, 5, 12, 11, and the references therein]. DPM models become popular in recently developed clustering methods probably because of the automatic learning of the number of mixture components from the data.

4.1.1 Dirichlet Process Mixture (DPM) Models

Let $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ denote n data points, where each data point $\boldsymbol{y}_i$ is a vector of values of a number of variables for the object or subject indexed by i. As discussed by [19, 20, 9], a Dirichlet process mixture model can be regarded as the limiting model of the ordinary Bayesian mixture model with a fixed number, K, of clusters (also called mixture components, sub-populations, groups, etc.) as $K \to \infty$. Bayesian mixture models with K (fixed) clusters are described in abstract form by the following hierarchy:

$$\begin{aligned}
\boldsymbol{y}_i | c_i, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_K &\sim F(\boldsymbol{\phi}_{c_i}), \quad \text{for } i = 1, \ldots, n, && (4.1)\\
\boldsymbol{\phi}_k &\sim G_0, \quad \text{for } k = 1, \ldots, K && (4.2)\\
P(c_1, \ldots, c_n | \pi_1, \ldots, \pi_K) &= \prod_{k=1}^{K} \pi_k^{n_k}, && (4.3)\\
\pi_1, \ldots, \pi_K &\sim \text{Dirichlet}(\alpha/K, \ldots, \alpha/K), && (4.4)
\end{aligned}$$

where $F(\boldsymbol{\phi})$ is the distribution of $\boldsymbol{y}_i$ given parameter $\boldsymbol{\phi}$ and cluster label, such as $N(\mu, \sigma^2)$, G_0 is the prior distribution for the parameter $\boldsymbol{\phi}_k$ of each cluster; $c_1, \ldots, c_n$ are latent cluster labels, taking integers from 1 to K, and $n_k = \sum_{i=1}^{n} I(c_i = k)$, the number of data points belonging to cluster k, where $I(\cdot)$ is the indicator function, which is equal to 1 if the condition in bracket is true, 0 otherwise; $\pi_1, \ldots, \pi_K$ are unknown cluster proportions, indicating relative cluster size. The probability density function of the prior for the cluster proportions of (4.4) is

$$f(\pi_1, \ldots, \pi_K) = \frac{\Gamma(\alpha)}{\Gamma(\alpha/K) \cdots \Gamma(\alpha/K)} \pi_1^{\alpha/K - 1} \cdots \pi_K^{\alpha/K-1}, \text{ if } \sum_{k=1}^{K} \pi_k = 1, \tag{4.5}$$

and 0 otherwise. This model is also called multinomial allocation model by [9].

Dirichlet prior for $\pi_1, \ldots, \pi_K$ given by (4.4) is conjugate to the multinomial allocation distribution for the latent cluster labels $c_1, \ldots, c_n$ (equation (4.3)), therefore we can integrate $\pi_1, \ldots, \pi_K$ out, and obtain a marginalized prior for the latent cluster labels:

$$P(c_1, \ldots, c_n) = \frac{\Gamma(\alpha)}{\Gamma(\alpha/K) \cdots \Gamma(\alpha/K)} \times \frac{\Gamma(\alpha/K + n_1) \cdots \Gamma(\alpha/K + n_K)}{\Gamma(\alpha + n)}, \tag{4.6}$$

where $n_k = \sum_{i=1}^{n} I(c_i = k)$, the number of data points in the kth cluster; for understanding the above integral, note that the right expression of (4.5) integrates to 1. Using the fact that $\Gamma(x) = (x-1)\Gamma(x-1)$, we can rewrite (4.6) more elementarily as

$$P(c_1, \ldots, c_n) = \left(\frac{\alpha}{K}\right)^d \times \frac{\prod_{k \in \{1,\ldots,K\} \text{ with } n_k \geq 2} [(\alpha/K+1)\cdots(\alpha/K+n_k-1)]}{\alpha(\alpha+1)\cdots(\alpha+n-1)}, \tag{4.7}$$

where d is the number of non-empty clusters, ie, $d = \sum_{k=1}^{K} I(n_k \geq 1)$. Note that there are no more than d factors in the product above the line, also that we assume $n > 1$ in (4.7).

With α and other values in (4.7) fixed, if we let $K \to \infty$, the marginal probability for latent cluster labels $c_1, \ldots, c_n$ will converge to 0, which means that the chance of seeing a particular set of $c_1, \ldots, c_n$ will approach to 0. However, this is because when K is large there are many labellings of $c_1, \ldots, c_n$ that imply the same partition of data points. For example, if $K \geq 200$ and $n = 4$, the labelling with $c_1 = 1, c_2 = 2, c_3 = 1, c_4 = 1$ and the labeling with $c_1 = 200, c_2 = 3, c_3 = 200, c_4 = 200$ imply the same partition of $n = 4$ data points; more detailed discussions are given in [17]. The total number of different labellings that imply the same partition of n data points, denoted by $\mathcal{P}$, with d clusters is $K!/(K-d)!$ — the number of ways of choosing d integers with order mattering from $1, \ldots, K$ to represent the d distinct clusters in $\mathcal{P}$. Clearly, our data does not have information for distinguishing these $K!/(K-d)!$ different labellings, and the multinomial allocation prior [equation (4.7)] also gives them the same preference. We therefore consider the prior for all possible partitions of n data points induced by the multinomial allocation prior. The prior probability for a partition $\mathcal{P}$ is the sum of the probabilities for all these labellings, which is $K!/(K-d)!$ multiples of the probability for one such labelling (since they are all the same):

$$P(\mathcal{P}) \quad = \quad \frac{K!}{(K-d)!}\left(\frac{\alpha}{K}\right)^d \times \frac{\prod_{k \in \{1,\ldots,K\} \text{ with } n_k \geq 2} (\alpha/K+1)\cdots(\alpha/K+n_k-1)}{\alpha(\alpha+1)\cdots(\alpha+n-1)}. \tag{4.8}$$

When $K \to \infty$, the factor before $\times$ in equation (4.8) converges to α^d, and the numerator in the factor after $\times$ converges to $\prod_{k \in \{1,\ldots,K\} \text{ with } n_k \geq 1} (n_k - 1)!$, it follows that the prior distribution for a partition $\mathcal{P}$ given by (4.8) converges to the so-called Dirichlet Process prior:

$$P_{dp}(\mathcal{P}) = \alpha^d \frac{\prod_{j=1}^{d}(n_j - 1)!}{\alpha(\alpha+1)\cdots(\alpha+n-1)}, \tag{4.9}$$

where d is the number of clusters implied by the partition $\mathcal{P}$, and $n_1, \ldots, n_d$ (all ≥ 1) are the numbers of data points of the d clusters (not the numbers of data points in the jth cluster as in (4.8)).

It is convenient to represent a partition $\mathcal{P}$ with n labels $c_1, \ldots, c_n$ — the data points of the same cluster are assigned with the same label, with arbitrarily chosen d distinct label values for representing d different clusters. With a partition of n data points represented by labels $c_1, \ldots, c_n$, we can describe a Dirichlet process mixture model as follows:

$$\begin{aligned}
\boldsymbol{y}_i | c_i, \boldsymbol{\phi}_1, \ldots &\sim F(\boldsymbol{\phi}_{c_i}), \quad \text{for } i = 1, \ldots, n, && (4.10)\\
\boldsymbol{\phi}_k &\sim G_0, \quad \text{for } k = 1, 2, \ldots, && (4.11)\\
P_{dp}(c_1, \ldots, c_n) &= \alpha^d \frac{\prod_{j=1}^{d}(n_j - 1)!}{\alpha(\alpha+1)\cdots(\alpha+n-1)}, && (4.12)
\end{aligned}$$

where d is the number of distinct labels in $c_1, \ldots, c_n$, ie, the number of clusters, $n_1, \ldots, n_d$ are the numbers of data points in each cluster.

The joint distribution P_{dp} for labels $c_1, \ldots, c_n$ in (4.12) can also be written as the product of n successive conditional distributions (or a stochastic process generating labels $c_1, \ldots, c_n$):

$$P(c_i = c | c_1, \ldots, c_{i-1}) = \begin{cases} \dfrac{n_c^{(i)}}{\alpha + i - 1}, & \text{for } c = \text{some } c_k, k \leq i-1, \\ \dfrac{\alpha}{\alpha + i - 1}, & \text{otherwise,} \end{cases} \tag{4.13}$$

where $n_c^{(i)} = \sum_{j=1}^{i-1} I(c_j = c)$. This method for describing Dirichlet process prior is used by [20]. Regarded as a stochastic process for generating labels $c_1, \ldots, c_n$ (with c_1 given an arbitrary value), the conditional probability in equation (4.13) means that given $c_1, \ldots, c_{i-1}$, the next label c_i is randomly set to one of the labels that have appeared in $c_1, \ldots, c_{i-1}$, with probabilities proportional to the times of appearing, or is assigned with a new label with probability $\alpha/(\alpha + i - 1)$. There are also other methods that describe Dirichlet processor prior on n parameters $\boldsymbol{\theta}_i = \boldsymbol{\phi}_{c_i}$, in which the parameters for the data points in the same cluster are identical, see eg, [20, 8].

4.1.2 Dirichlet Process Criterion (DPC)

To obtain the posterior of the partitions, which is represented by labels $c_1, \ldots, c_n$, we will marginalize the cluster-specific model parameters $\boldsymbol{\phi}_k$ out. For simplicity of notation, let's assume the d cluster labels in $c_1, \ldots, c_n$ are represented by integers $1, \ldots, d$, by Bayes' rule, the posterior distribution of $c_1, \ldots, c_n$ is given by:

$$
\begin{aligned}
& P(c_1, \ldots, c_n | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n) \\
& \quad \propto \; P(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n | c_1, \ldots, c_n) \times P_{dp}(c_1, \ldots, c_n) \qquad (4.14) \\
& \quad = \; \prod_{j=1}^{d} \int \prod_{i=1,\ldots,n,\text{with } c_i=j} P(\boldsymbol{y}_i|\boldsymbol{\phi}_j) g_0(\boldsymbol{\phi}_j) d\boldsymbol{\phi}_j \times \alpha^d \frac{\prod_{j=1}^{d}(n_j-1)!}{\alpha(\alpha+1)\cdots(\alpha+n-1)}, \qquad (4.15) \\
& \quad \equiv \; \mathrm{DPC}(\mathcal{P}) \qquad (4.16)
\end{aligned}
$$

where the conditional distribution $P(\boldsymbol{y}_i|\boldsymbol{\phi})$ is specified by $F(\boldsymbol{\phi})$, $g_0(\boldsymbol{\phi})$ is the probability density function of the prior G_0, for example a normal distribution. Equation (4.15) is the *Dirichlet Process Criterion* we will use to measure the goodness of partition, $\mathcal{P}$, of data points in light of data information $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$. The factor before $\times$ in (4.15) is the product of d marginalized likelihoods (with parameters averaged out with respect to prior G_0) of data points in each cluster. Ratio of marginalized likelihoods for two possible models, called Bayes factor, has been long used for selecting models, because it has an appealing property that it automatically favours a simpler model if a more complex model doesn't fit the data substantially well, see eg [6]. Note that, however, marginalized likelihood is also highly controversial, see eg [15, 2].

In the recent literature, there have been many MCMC based methods for fitting a Dirichlet process mixture models, see eg [19, 8, 16, 20, 22, 5, 12, 11]. MCMC methods have the advantage of automatically producing an uncertainty measure for the estimated parameters. In many cluster analysis problems, however, the primary interest is to find an appropriate partition (clustering) of data points such that the data points in a cluster are similar. For this purpose, MCMC based methods have difficulty in dealing with the multiple local modes problems. There might be multiple modes corresponding to the same partition of data points but using different labellings $c_1, \dots, c_n$, as known as the well studied label switching problems [see eg 23, 25]. There might be also multiple modes corresponding to different partitions, possibly with different numbers of clusters, which the data information cannot distinguish clearly. Therefore, MCMC samples of the cluster labels cannot be used directly for partitioning (clustering) data points. In our method, we will use an adaptively splitting method to search a partition that has reached a local mode of Dirichlet process criterion.

When a prior G_0 for $\boldsymbol{\phi}_j$ is conjugate to the distribution $F(\boldsymbol{\phi}_j)$, the integral in (4.15) can be found in a closed form. For most problems, a conjugate prior may not be realistic, and a non-conjugate prior is more appropriate. For such situations, one can use Laplace method to approximate the marginalized likelihood. For simplicity of notation, let $l_j(\boldsymbol{\phi}_j)$ denote the log un-normalized likelihood of $\boldsymbol{\phi}_j$ given the data points $\boldsymbol{y}_i$ in cluster j:

$$l_j(\boldsymbol{\phi}_j) = \log\left(\prod_{i=1,\dots,n,\text{with } c_i=j} P(\boldsymbol{y}_i|\boldsymbol{\phi}_j) g_0(\boldsymbol{\phi}_j))\right). \tag{4.17}$$

Let $\hat{\boldsymbol{\phi}}_j$ denote the maximizer of $l_j(\boldsymbol{\phi}_j)$, ie, the MLE of $\boldsymbol{\phi}_j$ based on the data points in cluster j. Let $H_j(\boldsymbol{\phi}_j)$ denote the Hessian (2nd order partial derivatives) matrix of $-l_j(\boldsymbol{\phi}_j)$ at parameters $\boldsymbol{\phi}_j$. With these notations, the marginalized likelihood can be approximated as follows:

$$\int \exp(l_j(\boldsymbol{\phi}_j)\, d\boldsymbol{\phi}_j \approx (2\pi)^{p/2} \det(H_j(\hat{\boldsymbol{\phi}}_j))^{-1/2} \times \exp\left(l_j(\hat{\boldsymbol{\phi}}_j)\right), \tag{4.18}$$

where p is the number of parameters in $\boldsymbol{\phi}_j$. In summary, our clustering method will find a partition $\mathcal{P}$ that maximizes the following approximate Dirichlet Process Criterion:

$$\widehat{\text{DPC}}(\mathcal{P}) = \prod_{j=1}^{d} \left[(2\pi)^{p/2} \det(H_j(\hat{\boldsymbol{\phi}}_j))^{-1/2} \times \exp\left(l_j(\hat{\boldsymbol{\phi}}_j)\right)\right] \times \alpha^d \frac{\prod_{j=1}^{d}(n_j-1)!}{\alpha(\alpha+1)\cdots(\alpha+n-1)} \tag{4.19}$$

4.2 Application to Mixture of Student's t-Distributions

4.2.1 Mixture of Student's t-Distributions

Given a data set $\mathbf{X}$ with N data points in a p-dimensional space, that is, $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, and $\mathbf{x}_j = \{x_{j1}, \dots, x_{jp}\}$, for $j = 1, \dots, N$. Suppose we want to cluster $\mathbf{X}$ into K partitions, denoted as $G_1, \dots, G_K$. Let z_j be a component label for $\mathbf{x}_j$, which is assumed to take values from $1, \dots, K$. Then $P(z_j = k) = \pi_k$ is the probability that data point $\mathbf{x}_j$ comes from partition G_k, for $k = 1, \dots, K$. Given a partition G_k, for all $\mathbf{x}_j$ belong to this partition, assume each dimension of such $\mathbf{x}_j$ is i.i.d, and follows a Student's t-distribution with mean μ_{ki}, standard deviation σ_{ki}, for $i = 1, \dots, p$, and specific degrees of freedom. Then for each $\mathbf{x}_j$ belonging to partition G_k, it forms a mixture of Student's t-distribution with mean $\boldsymbol{\mu}_k = \{\mu_{k1}, \dots, \mu_{kp}\}$, standard deviation $\boldsymbol{\sigma}_k = \{\sigma_{k1}, \dots, \sigma_{kp}\}$, and degrees of freedom ν. If we define $\boldsymbol{\mu}$ as a vector of all values of $\boldsymbol{\mu}_k$, $\boldsymbol{\sigma}$ as a vector of $\boldsymbol{\sigma}_k$, and $\boldsymbol{\pi}$ as a vector of all values of π_k, then the distribution of such $\mathbf{x}_j$ can be described as:

$$\begin{aligned} P(\mathbf{x}_j|z_j = k, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma}) &= \prod_{i=1}^{p} P(x_{ji}|z_j = k, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma}) && (4.20)\\ &= \prod_{i=1}^{p} \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})}(1 + \frac{(x_{ji} - \mu_{ki})^2}{\nu\sigma_{ki}^2})^{-(\frac{\nu+1}{2})} && (4.21)\\ &\propto \prod_{i=1}^{p}(1 + \frac{(x_{ji} - \mu_{ki})^2}{\nu\sigma_{ki}^2})^{-(\frac{\nu+1}{2})} && (4.22) \end{aligned}$$

Since we need to fit this mixture model of Student's t-Distributions into a Dirichlet Process Mixture Model, we also need to consider the distributions for Dirichlet process priors, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ in this case. Assume each μ_{ki} follows a normal distribution with mean m_μ and standard deviation w_μ, and each $\log(\sigma_{ki})$ follows a normal distribution with mean $m\_\log(\sigma)$ and standard deviation $w\_\log(\sigma)$, for $k = 1, \dots, K$, and $i = 1, \dots, p$.

4.2.2 Winner-Get-All EM Algorithm for Mixture of Student's t-Distributions

Using the data set $\mathbf{X}$ and the Mixture Models of Student's t-Distributions described in section (4.2.1), apply a winner-get-all EM algorithm to calculate the maximum likelihood estimates for $\boldsymbol{\mu}, \boldsymbol{\sigma}\}$ and $\boldsymbol{\pi}$. Let $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\sigma}}$, and $\hat{\boldsymbol{\pi}}$ be the initial estimates for $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$. Also set the values for m_μ, w_μ, $m\_\log(\sigma)$, and $w\_\log(\sigma)$. The winner-get-all EM algorithm is described as follows:

- E-Step:

  Given $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\sigma}}$, and $\hat{\boldsymbol{\pi}}$, compute $P(z_j = k|\mathbf{x}_j, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\theta}})$ for $i = 1, \ldots, N$, and $k = 1, \ldots, K$.

  By **Bayes' Rule**,

$$P(z_j = k|\mathbf{x}_j, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}) = \frac{P(\mathbf{x}_j|z_j = k, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}})P(z_j = k|\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}})}{\sum_{k=1}^{K} P(\mathbf{x}_j|z_j = k, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}})P(z_j = k|\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}})} \tag{4.23}$$

$$\propto P(\mathbf{x}_j|z_j = k, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}})\hat{\pi}_k \tag{4.24}$$

  Where $P(\mathbf{x}_j|z_j = k, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}})$ is expressed in (4.22).

  Denote $P(z_j = k|\mathbf{x}_j, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}})$ by $\hat{\pi}_k^{(j)}$ for $j = 1, \ldots, N$, and $k = 1, \ldots, K$. $\hat{\pi}_k^{(j)}$ is the probability that a given data point $\mathbf{x}_j$ comes from partition G_k, for $k = 1, \ldots, K$.

- M-Step:

  Assign each data point to the partition that it has the highest probability of coming from. That is, assign each $\mathbf{x}_j$ to partition G_k that produces the highest value of $\hat{\pi}_k^{(j)}$, for $k = 1, \ldots, K$. Inside each partition G_k, compute the maximum likelihood estimates for $\boldsymbol{\mu}_k$ and $\boldsymbol{\sigma}_k$, denoted by $\ddot{\boldsymbol{\mu}}_k$ and $\ddot{\boldsymbol{\sigma}}_k$, such that maximize the expected log likelihood function found in E-step.

  The log likelihood for a single $\mathbf{x}_j$ assigned to partition G_k can be shown as

$$\log\left(P(\mathbf{x}_j|z_j=k,\hat{\boldsymbol{\pi}},\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\sigma}})\right) = \log\left(\prod_{i=1}^{p} P(\mathbf{x}_{ji}|z_j=k,\hat{\boldsymbol{\pi}},\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\sigma}})\right) \tag{4.25}$$

$$= \sum_{i=1}^{p} \log\left(P(x_{ji}|z_j=k,\hat{\boldsymbol{\pi}},\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\sigma}})\right) \tag{4.26}$$

$$\propto -(\frac{\nu+1}{2})\sum_{i=1}^{p}\log\left(\frac{(x_{ji}-\mu_{ki})^2}{\sigma_{ki}^2}+\nu\right) - \sum_{i=1}^{p}\log(\sigma_{ji}) \tag{4.27}$$

Then the log likelihood for all $\mathbf{x}_j$ assigned to partition G_k can be shown as

$$\hat{\mathcal{L}}(\boldsymbol{\mu}_k,\boldsymbol{\sigma}_k) = \log\left(\prod_{\mathbf{x}_j\in G_k} P(\mathbf{x}_j|z_j=k,\hat{\boldsymbol{\pi}},\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\sigma}})\right) + \log\left(\prod_{i=1}^{p} P(\mu_{ki},\log(\sigma_{ki})\right) \tag{4.28}$$

$$= \sum_{j=1}^{N}\log\left(P(\mathbf{x}_j|z_j=k,\hat{\boldsymbol{\pi}},\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\sigma}})\right) + \sum_{i=1}^{p}\log\left(P(\mu_{ki})\right) + \sum_{i=1}^{p}\log\left(P(\log(\sigma_{ki}))\right) \tag{4.29}$$

Where the result of $\log(P(\mathbf{x}_j|z_j=k,\hat{\boldsymbol{\pi}},\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\sigma}}))$ is shown in (4.27), $\log(P(\mu_{ki}))$ is a normal distribution with mean m_μ and standard deviation $w\_\log(\sigma)$, and $\log(P(\log(\sigma_{ki})))$ is a normal distribution with mean $m\_\log(\sigma)$ and standard deviation $w\_\log(\sigma)$ for $k = 1,\dots,K$, and $i = 1,\dots,p$.

$\ddot{\mu}_k$ and $\ddot{\boldsymbol{\sigma}}_k$ are obtained by calculating the maximum likelihood estimates for function (4.29). Repeat this process for $k = 1,\dots,K$, we can obtain the values of $\ddot{\boldsymbol{\mu}}$ and $\ddot{\boldsymbol{\sigma}}$. Replace the values of $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\sigma}}$ with the values of $\ddot{\boldsymbol{\mu}}$ and $\ddot{\boldsymbol{\sigma}}$.

- Repeat the above two steps until the values of $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\sigma}}$, and $\hat{\boldsymbol{\pi}}$ converge.

4.2.3 Outline of Non-MCMC DPM Algorithm

Apply the Splitting Procedure on the data set $\mathbf{X}$ described in section (4.2.1). The detailed steps are shown as follows:

- **Step 1.** Our initial model is constructed by considering all of the data points inside X coming from one cluster, that is, set $K = 1$. Let $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\pi}}$ be the initial estimates for $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$. Also set the values for m_μ, w_μ, $m\_\log(\sigma)$, and $w\_\log(\sigma)$. Apply EM algorithm to estimate $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$.

  After obtaining $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\pi}}$, we need to calculate the value of Dirichlet Process Criterion (DPC) for the current partition. The DPC value can be estimate using Laplace method:

$$DPC(\mathcal{P}) \approx \prod_{k=1}^{K} \exp(-h(\boldsymbol{\theta}_k))(2\pi)^{\frac{p}{2}}|\nabla^2 h(\boldsymbol{\theta}_k)|^{-\frac{1}{2}} \frac{\alpha^K \prod_{k=1}^{K}(n_k - 1)!}{\alpha(\alpha+1)\dots(\alpha+n-1)} \tag{4.30}$$

  Where $-h(\boldsymbol{\theta}_k)$ is the minimum value of $\hat{\mathcal{L}}(\boldsymbol{\theta}_k)$. Here we calculate DPC value by setting $K = 1$ since there is only one partition in this case.

- **Step 2.** Repeat the following two sub-steps iteratively until no more new partition can be formed.

  Now we have a partition $\mathcal{P} = \{G_1, \dots, G_K\}$ with DPC value $DPC(\mathcal{P})$ and K clusters. The centroids for each cluster are defined by $\boldsymbol{\mu}_k$ and $\boldsymbol{\theta}_k$, for $k = 1, \dots, K$. Perform the following steps for each k, for $k = 1, \dots, K$.

  1. Randomly choose two data points within cluster k. These two points form the centroids for the two new clusters, and instead of having K clusters, now we have $K + 1$ clusters. Apply winner-get-all EM algorithm to computer the Maximum Likelihood Estimates $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\pi}}$. Also for each data point $\mathbf{x}_j$, compute $\hat{\pi}_k^{(j)}$ for $j = 1, \dots, N$, and $k = 1, \dots, K + 1$. Recall $\hat{\pi}_k^{(j)}$ stands for the probability that a given data point $\mathbf{x}_j$ comes from cluster k. Update the partition by assigning each $\mathbf{x}_j$ to the cluster that it has the highest probability of coming from. Add the newly formed cluster to the end of the original partition $\mathcal{P}$, and we have a brand new partition, call it $\mathcal{P}_{\text{new}}$, with parameter values $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\pi}}$. Approximate $DPC(\mathcal{P}_{\text{new}})$ using formula (4.30) with the parameters obtained from EM algorithm.

2. Compare the values of $DPC(\mathcal{P}_{\text{new}})$ and $DPC(\mathcal{P})$.

   – If $DPC(\mathcal{P}_{\text{new}}) \geq DPC(\mathcal{P})$, we will accept $\mathcal{P}_{\text{new}}$. Set $DPC(\mathcal{P})$ to be $\mathcal{P}_{\text{new}}$. The value of K is increased by 1. Again the newly formed cluster is placed at the end of the partition.

   – If $DPC(\mathcal{P}_{\text{new}}) < DPC(\mathcal{P})$, we will reject $\mathcal{P}_{\text{new}}$, and retain $\mathcal{P}$.

Recall X-Means algorithm runs a local EM algorithm inside each parent region. It is local because only points inside each parent region are considered, no others. Problems arise because of this local EM algorithm. If some data points are assigned to wrong data partitions in the previous step, they stay in the wrong partitions forever. In contrast to X-Means algorithm, our algorithm applies EM algorithm to the entire clusters every time when a split is performed, and the cluster assignment is updated every time after running EM algorithm. Therefore, if some data points are assigned to wrong clusters in the previous steps, we are able to re-assign them to the right clusters in the following steps.

4.3 Simulation Studies

4.3.1 Demonstration of Non-MCMC DPM Clustering Algorithm

Dirichlet Process Criterion is the key factor in the Non-MCMC DPM clustering algorithm since it is the standard we use to decide whether or not to accept the new partition after splitting the original model. Figure (4.1) and Figure (4.2) display two scenarios that use DPC value to select the best partition.

- **Case 1.** Model formed by only 1 cluster has a lower DPC value than the same model formed by 2 clusters.

  The partition with 1 cluster shown in the first graph of Figure (4.1) has a DPC value of -431, while the partition with 2 clusters shown in the second graph of Figure (4.1) has a DPC value of -21. Therefore, the partition with 2 clusters is more preferred than the partition with 1 cluster.

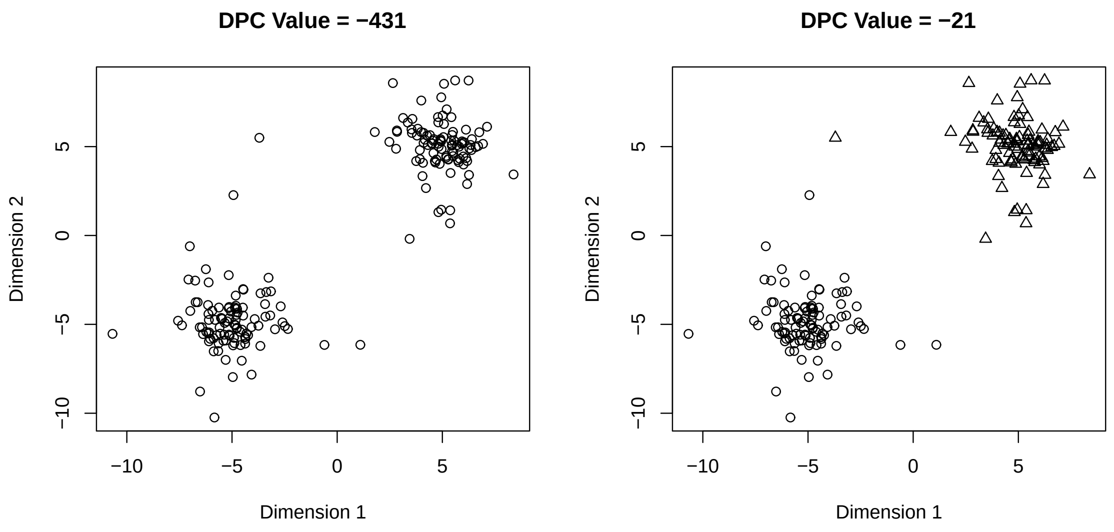

Figure 4.1: Since the model formed by 1 cluster has a lower DPC value than the same model formed by 2 clusters, the partition shown in the first graph is less preferred than the partition shown in the second graph is selected.

- **Case 2.** Model formed by only 1 cluster has a higher DPC value than the same model formed by 2 clusters.

  The partition with 1 cluster shown in the first graph of Figure (4.2) has a DPC value of 111, while the partition with 2 clusters shown in the second graph of Figure (4.2) has a DPC value of 102. Therefore, the partition with 1 clusters is more preferred than the partition with 2 clusters.

We can show the complete Non-MCMC DPM clustering algorithm in Figure (4.3) over a sample data set. A graph is recorded when a new partition is formed.

As we can see from Figure (4.3), the clustering result using the Non-MCMC DPM clustering algorithm is perfect.

4.3.2 Performance Comparison with X-Means

I assess the performance of the methodology introduced in chapter (4) and the performance of X-Means algorithm using a simulated data. I generate a data set of 500 two-dimensional observations as follows:

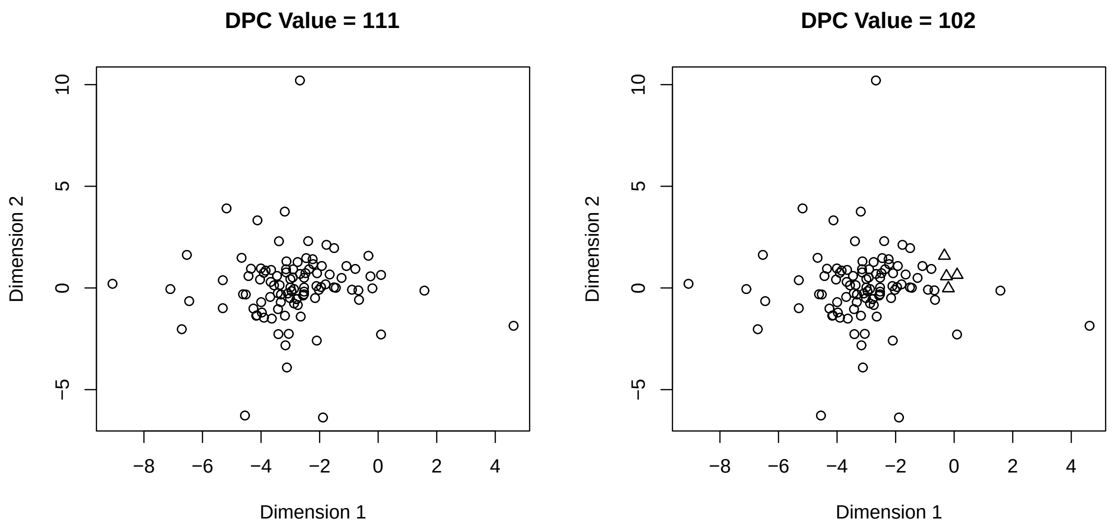

Figure 4.2: Since the model formed by 1 cluster has a higher DPC value than the same model formed by 2 clusters, the partition shown in the first graph is more preferred than the partition shown in the second graph.

- Generate a 2x100 matrix. Each element of this matrix is generated from a standard Student's t-Distribution with mean 0, standard deviation 1, and degrees of freedom 3. Generate another 4 matrices using the same method.

- Add different values to each row of these 5 matrices to make sure the elements values within each matrix will be different from others. For this data set, I add -3 to the first row of Matrix 1, and 0 to the second row of Matrix 1, denoted by (-3, 0). Then I add (3, 5) to the rows of Matrix 2, (3, -6) to the rows of Matrix 3, (3, 0) to the rows of Matrix 4, and (10, 0) to the Matrix 5.

- Form a new matrix by combining all 5 matrices by columns. Transpose the matrix.

The final matrix we obtained contains our simulated data set. Performing steps above, we intentionally assign these 500 observations into 5 groups. If the clustering algorithm is good, it should produce 5 clusters.

Figure (4.4) shows the final partitions we obtain after applying the Non-MCMC DPM clustering algorithm and X-Means algorithm over the simulated data.

The Non-MCMC DPM clustering algorithm successfully clusters the simulated data into correct 5 partitions, while X-Means algorithm only produces 3 partitions. The Non-MCMC

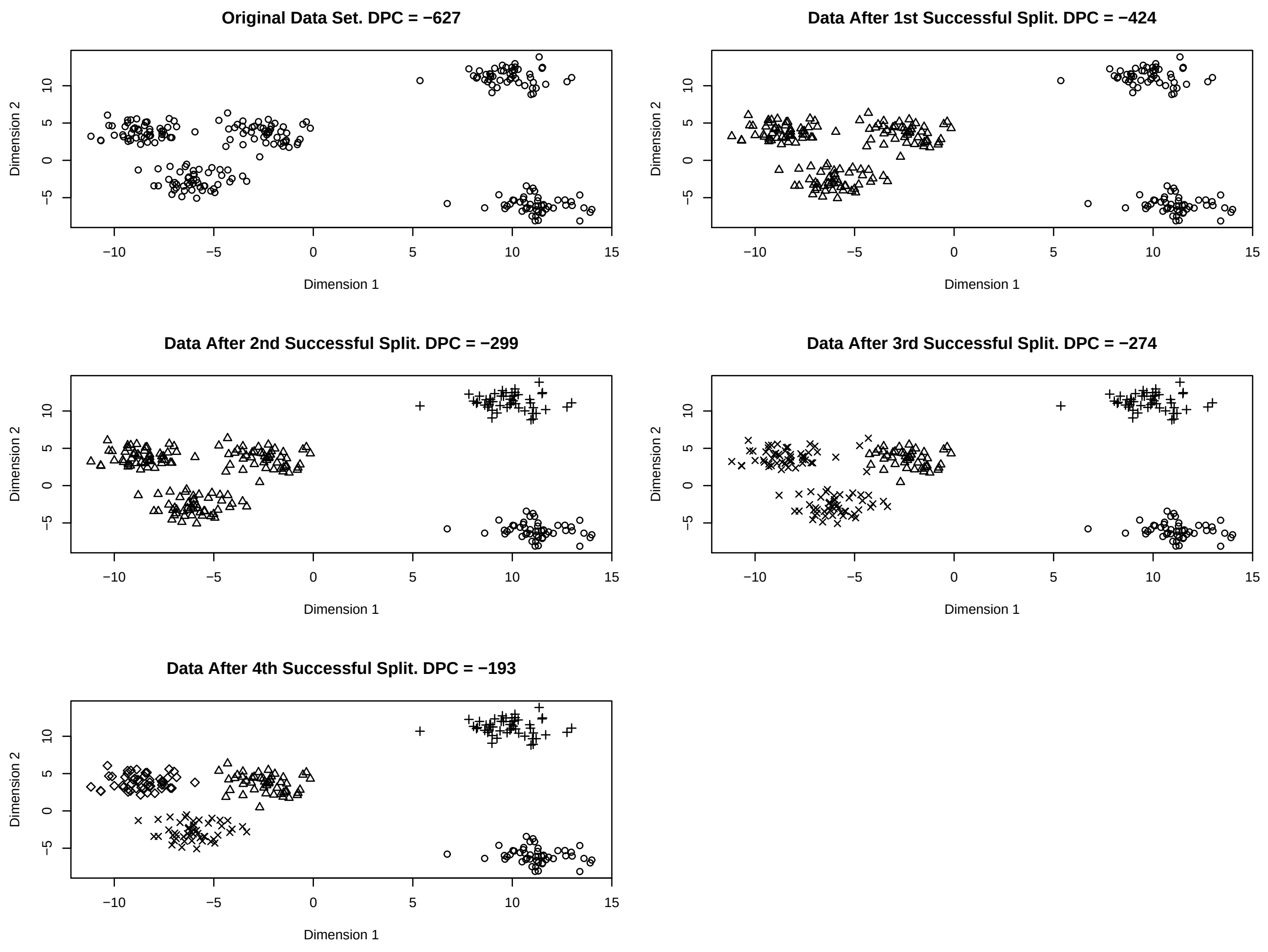

Figure 4.3: Cluster sample data set using Non-MCMC DPM clustering algorithm. Each graph is recorded when a new partition is formed.

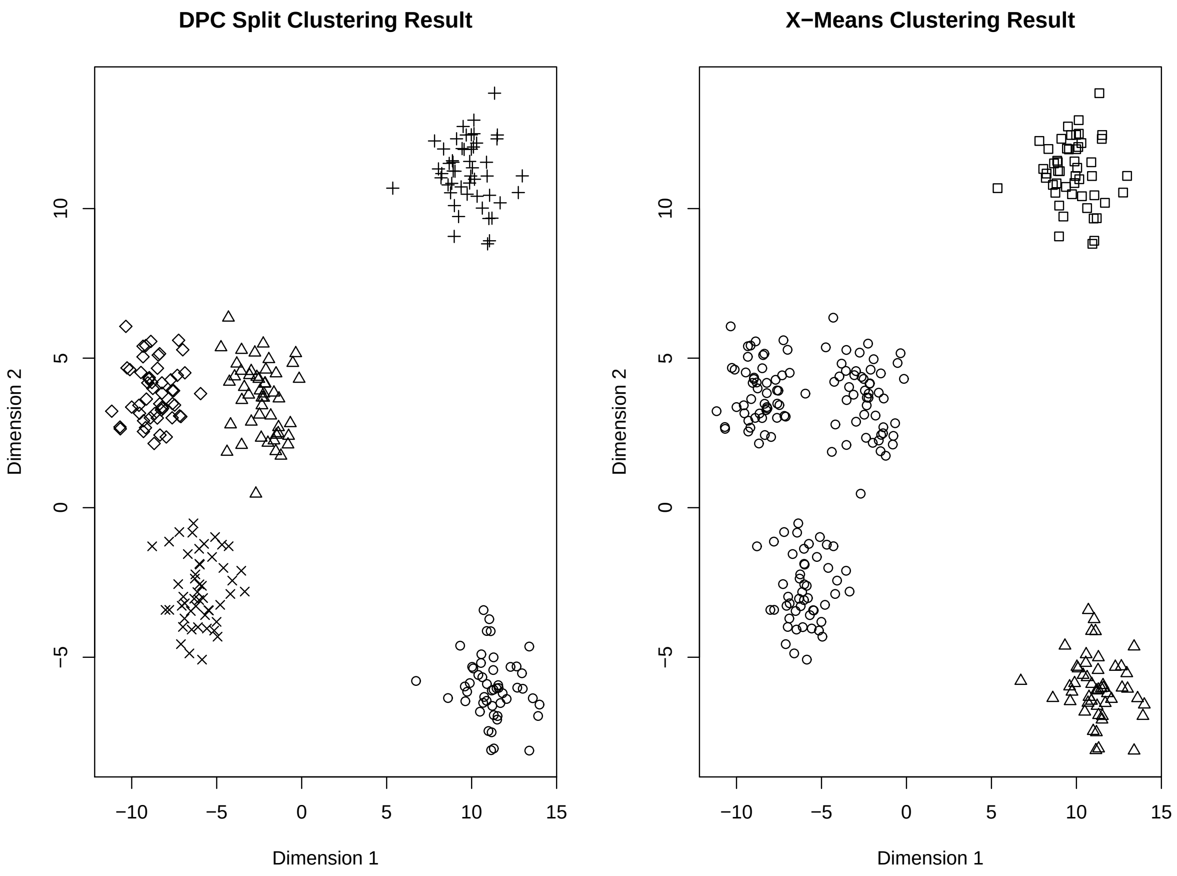

Figure 4.4: Simulate a data set of 500 two-dimensional observations. Each dimension follows a standard Student's t-distribution with degrees of freedom 3. Assign them into 5 groups. Apply Non-MCMC DPM clustering algorithm and X-Means algorithm over the data set.

DPM clustering algorithm does a much better job than X-Means algorithm.

Inside Non-MCMC DPM clustering algorithm, there is one variable that can change the final clustering result. That variable is α. Inside X-Means algorithm, there is also a variable can change the final clustering result, which is the minimum number of clusters pre-set by the user. Will the clustering result change dramatically if we change the value of α in Non-MCMC DPM clustering algorithm, and change the minimum number of clusters in X-Means algorithm? Let us test it by generating different simulated data sets. Instead of generating a data set of two-dimensional observations with each dimension follows a standard Student's t-distribution, we generate a 250 data set of six-dimensional observations as follows:

- Generate a 6x50 matrix. The elements for the first 3 rows are generated from a standard Student's t-Distribution with mean 0, standard deviation 1, and degrees of freedom 3, while the elements for the remaining 3 rows are generated from a standard normal distribution with mean 0, standrd deviation 1. Generate another 4 matrices using the same method.

- Add a randomly generated value to each row of these 5 matrices to make sure the elements' values within each matrix will be different from others. This randomly generated value follows a normal distribution $N(0, 3)$.

- Form a new matrix by combining all 5 matrices by columns. Transpose the matrix.

The final matrix we obtained contains our simulated data set. By performing steps above, we intentionally assign these 250 observations into 5 groups. If the clustering algorithm is good, it should produce 5 clusters. Generate this matrix 50 times.

Figure (4.5) shows the histogram for number of clusters obtained after applying Non-MCMC DPM clustering algorithm over the 50 simulated data sets for $\alpha = 1, 10, 100, 1000$, respectively.

Figure (4.6) shows the histogram for number of clusters obtained after applying X-Means algorithm over the 50 simulated data sets for Minimum Number of Clusters = 1, 2, 3, 5, respectively.

As we can see from Figure (4.5), 47 out of 50 times Non-MCMC DPM clustering algorithm produces a partition of 5 clusters, and 3 out of 50 times it produces a partition of 4 clusters,

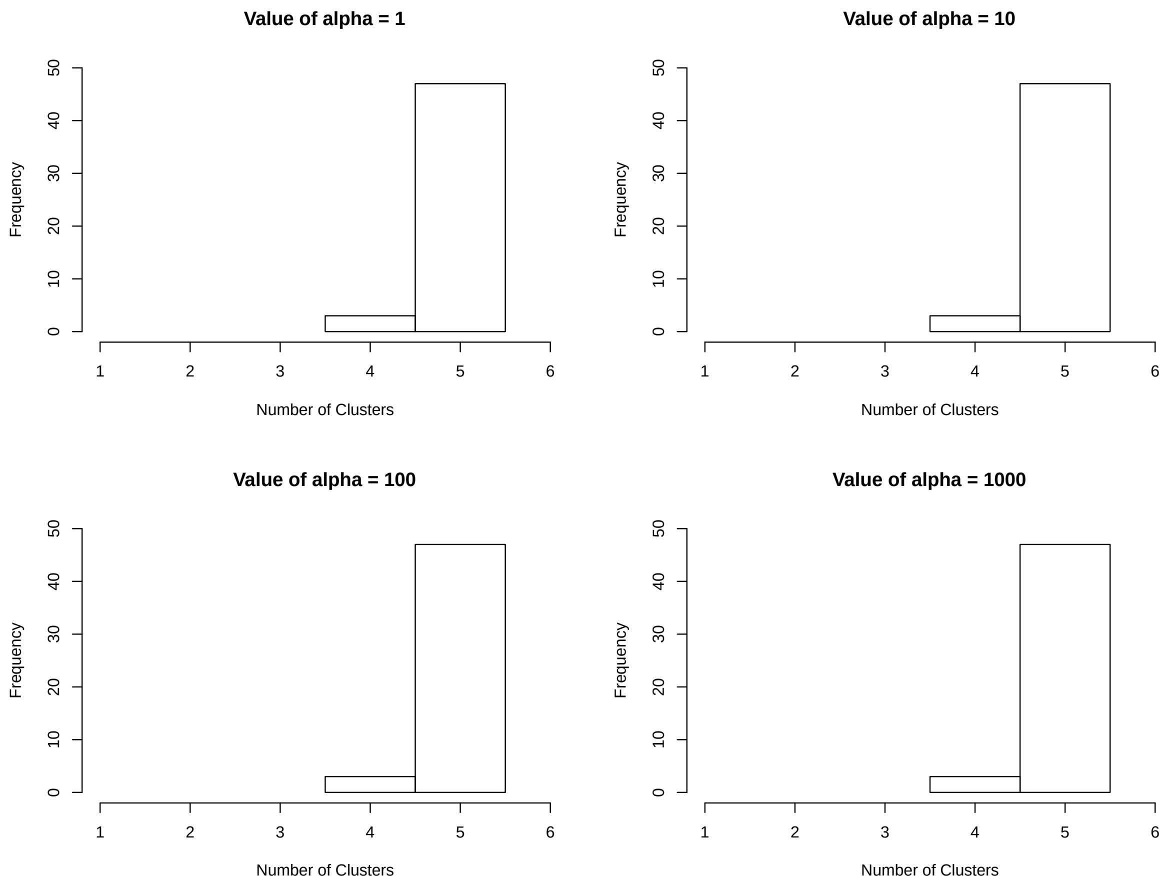

Figure 4.5: Histogram for number of clusters after applying Non-MCMC DPM clustering algorithm over the 50 simulated data sets for $\alpha = 1, 10, 100, 1000$, respectively.

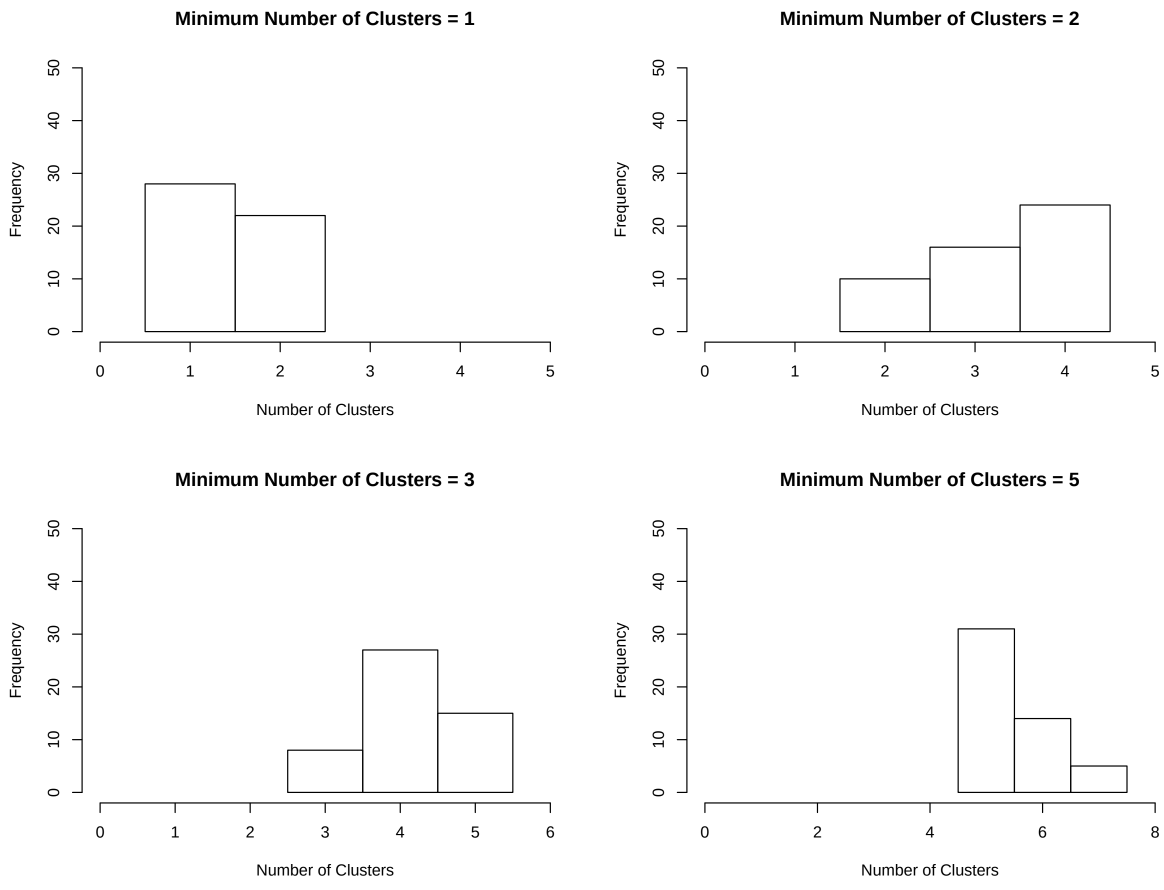

Figure 4.6: Histogram for number of clusters after applying X-Means algorithm over the 50 simulated data sets with Minimum Number of Clusters = 1, 2, 3, 5 respectively.

no matter what value of α is used. On the other hand, Figure (4.6) tells us that when X-Means algorithm is applied over the simulated data sets, number of clusters varies a lot with different minimum number of clusters. The clustering results are also very poor when the minimum number of clusters is selected to be 1 or 2. The result produced by Non-MCMC DPM clustering algorithm is very impressive.

4.4 Application to Real Data Analysis

4.4.1 Class Prediction Using Clustering Results

Data or observations that are grouped inside the same clusters usually have similar characteristics. For example, apply the clustering methods over a data set related to Cancer patients. Among the clusters obtained, one cluster will likely contain the data that indicate the patients are in early stage of Cancer, and another cluster will likely contain the data that indicate patients are in critical stage of Cancer. Therefore, if we obtain the cancer data for a new patient, we can predict whether this patient is in early stage of cancer or critical stage of cancer using our clustering results.

Suppose we are given a data set for patients who have cancer. We are also given the information that the stage of cancer that each patient is at. Apply the clustering method over this data set. Let

- K: Number of clusters obtained.
- y: An indicator variable that indicates whether the patient is in early stage of cancer or critical stage of cancer. Assume $y = 0$ means the patient is in early stage of cancer, while $y = 1$ means the patient is in critical stage of cancer
- $\mathbf{P}_k^{(y=1)}$: Probability of $y = 1$ for all data set inside Cluster k, for $k = 1, \ldots, K$.
- $\mathbf{X}_N$: Data set for a new patient.
- π_k^N: Probability that this new patient belongs to Cluster k, for $k = 1, \ldots, K$.
- C_N: Cluster for this new patient.

- $\mathbf{P}_N^{(y=1)}$: Probability that this new patient is in critical stage of cancer.

We can predict the probability that this new patient is in critical stage of cancer by

$$\mathbf{P}_N^{(y=1)} = P(y = 1|\mathbf{X}_N) \tag{4.31}$$

$$= \sum_{k=1}^{K} P(y = 1|\mathbf{X}_N, C_N = k)P(C_N = k|\mathbf{X}_N) \tag{4.32}$$

$$= \sum_{k=1}^{K} \mathbf{P}_k^{(y=1)} \pi_k^N \tag{4.33}$$

4.4.2 Application To Prostate Microarray Data

We analyzed real microarray data containing gene expression data related to the prostate cancer. This data set contains expression profiles of 6033 genes from 50 normal and 52 cancerous tissues. This data set was originally reported by Singh et al. (2002). We analyzed a data set downloaded from the website http://stat.ethz.ch/∼dettling/bagboost.html for Dettling (2004), which contains more descriptions about this data set.

We select the first four most important features for Prostate Microarry Data using Kruskal-Wallis rank sum test, which is an extension of the Wilcoxon test and can be used to test the hypothesis that a number of unpaired samples originate from the same population. Figure (4.7) displays the scatter plots for 1st and 2nd, 1st and 3rd, 1st and 4th, and 2nd and 3rd important features of Prostate Microarry Data.

We select 20 most important features of Prostate Microarray Data, and apply the Non-MCMC DPM Clustering Algorithm over the selected features. Four clusters are produced. Figure (4.8) displays the means for each features inside each cluster.

Microarray experiments are expected to contribute significantly in cancer treatment by providing a precise and early diagnosis. Each Prostate Microarray Data contains an indicator that indicates whether or not this data has prostate cancer symptom. We can use the prediction test described in Section (4.4.1) to test the accuracy of our clustering methods. I will apply a 10-Fold Cross-Validation method over the data to predict whether or not the data has Prostate cancer symptom. The steps are as follows:

- Group the data into 10 groups in random. Select 1 group as a testing data set, and the

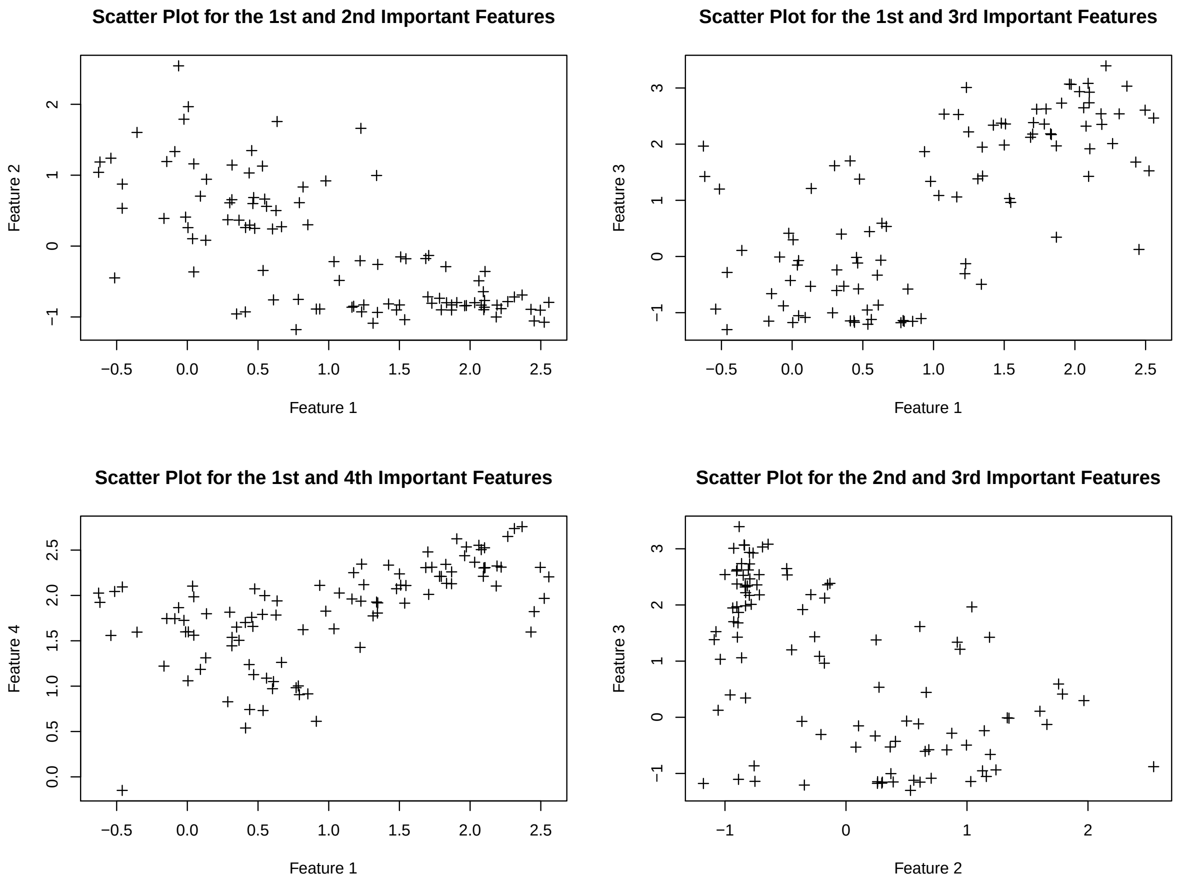

Figure 4.7: Scatter Plots for 1st and 2nd, 1st and 3rd, 1st and 4th, and 2nd and 3rd important features of Prostate Microarry Data.

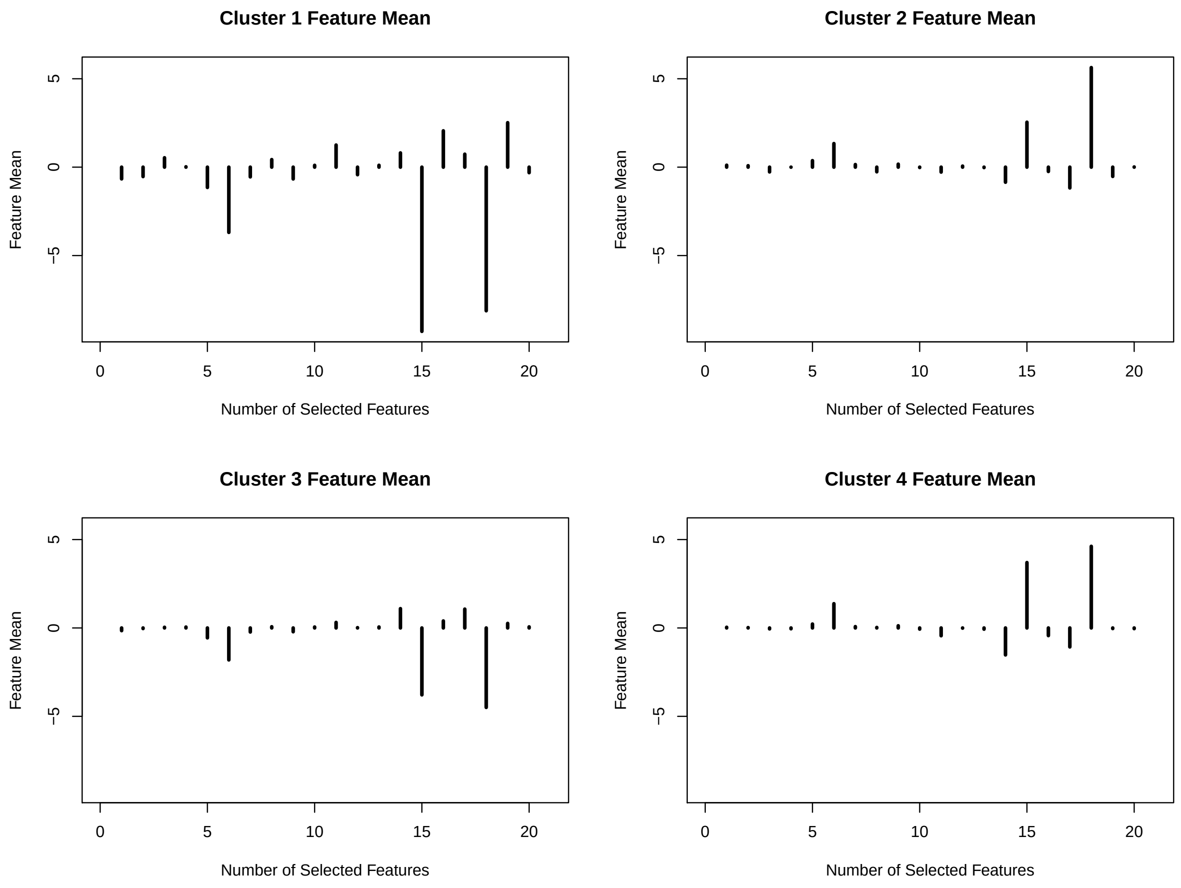

Figure 4.8: Means for each feature selected inside each cluster.

other 9 groups as a training data set. Ignore the cancer indicators for the data points in the testing data set. Use the information in training data set (including the cancer indicators for the data points in the training data set) to predict the cancer indicators for the data points in testing data set. Details about how to perform the prediction is described in Section (4.4.1).

- Repeat the step above by selecting a different group as a testing data set each time until the prediction is done for all groups.

We will obtain the predicted cancer indicators for all data points. Then we will compare the cancer indicators obtained with the cancer indicators given in the data to compute the error rate (rate of difference). Here I use four different techniques with number of features 2, 5, 10, 20, and 50 selected to compute the error rates:

- Class prediction method based on Non-MCMC Clustering Algorithm with Multivariate Student's t-Distribution.
- Class prediction method based on Non-MCMC Clustering Algorithm with Multivariate Normal Distribution.
- Class prediction method based on Lasso, which is an alternative regularized version of least squares. More information for Lasso can be found at [24].
- Class prediction method based on DLDA, which stands for Diagonal Linear Discriminant Analysis. More information for DLDA can be found at [7].

Figure (4.9) displays the error rates for these four different methods.

Here is a summary of information we obtain from Figure (4.9).

- Class prediction method based on Non-MCMC Clustering Algorithm with Multivariate Student's t-Distribution:
  Minimum Error Rate 5.88% obtained when Number of Features 20 is selected.
- Class prediction method based on Non-MCMC Clustering Algorithm with Multivariate Normal Distribution:
  Minimum Error Rate 6.86% obtained when Number of Features 5 is selected.

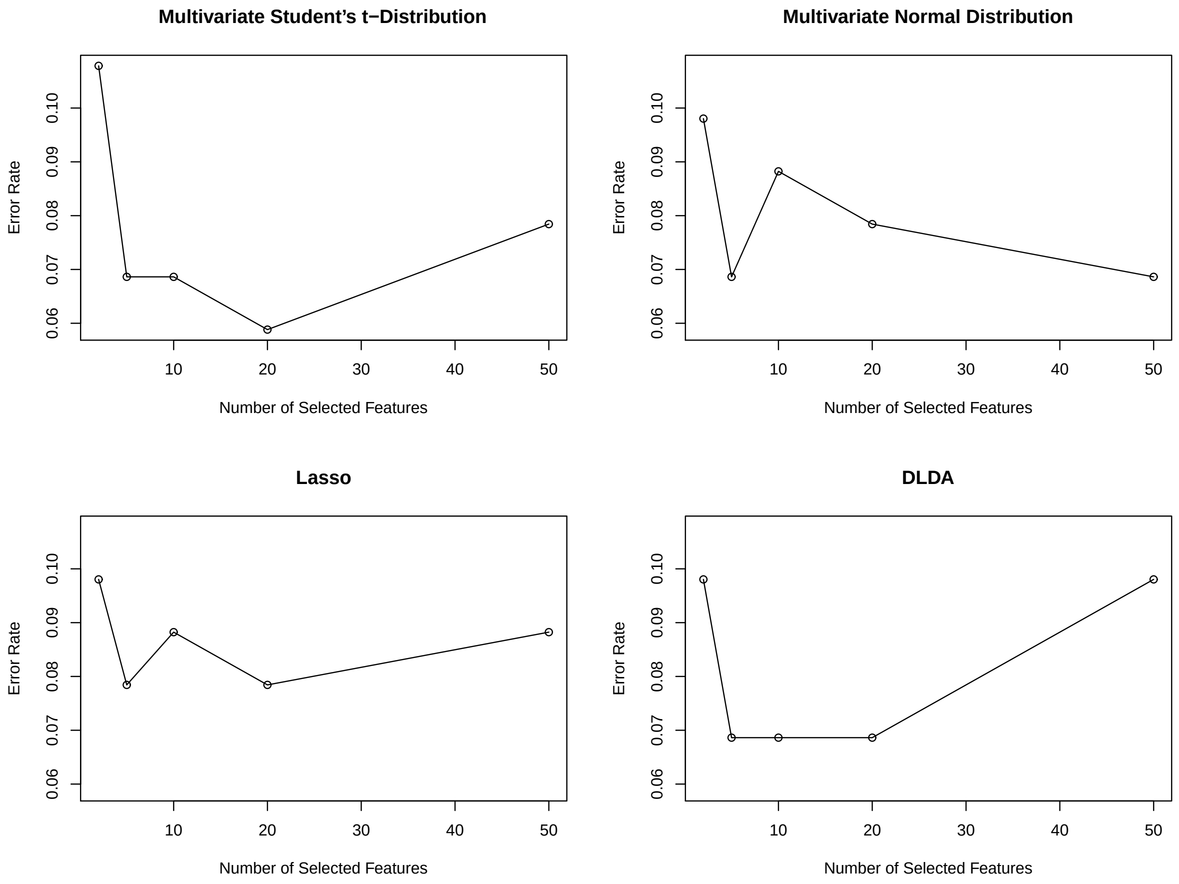

Figure 4.9: Error Rates Plot for four different clustering methods with number of features 2, 5, 10, 20, and 50 selected.

- Class prediction method based on Lasso:
  Minimum Error Rate 7.84% obtained when Number of Features 5 is selected.

- Class prediction method based on DLDA:
  Minimum Error Rate 6.86% obtained when Number of Features 5, 10, 20 is selected.

The class prediction method based on Non-MCMC Clustering Algorithm with Multivariate Student's t-Distribution performs better than the other methods in the test above.

Chapter 5
Conclusion and Discussion

The Non-MCMC DPM Clustering Algorithm we developed has several advantages over the existing methods discussed in section (1.2). First of all, it has the ability to determine number of clusters without any user's input. Secondly, if incorrect partitions are formed in the previous steps, it has the ability to fix the previous mistakes in the later stages by applying *Global* EM Algorithm over the entire data set. Last but not least, changing the parameter values in the method only has a minor effect regarding the final clustering result.

In Section (4.3.2), we simulated a data set of 500 two-dimensional observations, and assigned them into 5 groups. Non-MCMC DPM Clustering Algorithm with Multivariate Student's t-Distribution successfully partitioned the data set into 5 clusters, while X-Means Algorithm clustered the data set into 3 clusters. Then we simulated 50 data sets such that each of them contained 250 six-dimensional observations. We assigned each data set into 5 clusters as well. Both Non-MCMC DPM Clustering Algorithm with Multivariate Student's t-Distribution and X-Means Algorithm were applied to these 50 data sets with different parameter value for each method. For non-MCMC DPM Clustering Algorithm with Multivariate Student's t-Distribution, 1, 10, 100, and 1000 were used as the value of α, while for X-Means Algorithm, 1, 2, 3, and 5 were used as the value for the minimum number of clusters. 47 out of 50 times Non-MCMC DPM Clustering Algorithm with Multivariate Student's t-Distribution produced a correct partition of 5 clusters, no matter what value of α was used. In contrast, the number of clusters produced by X-Means Algorithm varied a lot with different minimum number of clusters.

In Section (4.4.2), we applied Non-MCMC DPM Clustering Algorithm with Multivariate Student's t-Distribution, Non-MCMC DPM Clustering Algorithm with Multivariate Normal Distribution, Lasso, and DLDA over the prostate microarray data set. A prediction test described in Section (4.4.1) was performed to test the accuracy of each method. Non-MCMC DPM Clustering Algorithm with Multivariate Student's t-Distribution performed better than the other methods for the given prostate microarray data set.

From our experiences, this clustering algorithm cannot separate a small number (say 1) of data points with distinct features from other data points to form a cluster, but rather merges them with other big clusters. This problem is originated to the lack of balanced allocation of data points inherent to Dirichlet process prior, as discussed by [9]. One may consider modifying the Dirichlet process in advance for allowing small cluster without resulting in an overly large number of clusters. For future development, we will compare this Non-MCMC procedure with existing MCMC procedures to identify the similarities and differences.

References

[1] J. Abonyi and B. Feil. *Cluster Analysis for Data Mining and System Identification.* Springer, Birkhuser Verlag AG, Berlin, 2007.

[2] M. Aitkin. Posterior bayes factors. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 111–142, 1991.

[3] N. Alldrin, A. Smith, and D. Turnbull. Clustering with em and k-means. 2003. available from `http://louis.ucsd.edu/ nalldrin/research/wo3/cse253/project1.pdf`.

[4] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucleic acids research*, 16:22, 1988.

[5] David B. Dahl. An improved merge-split sampler for conjugate dirichlet process mixture models. Technical Report 1086, Department of Statistics, and Department of Biostatistics and Medical Informatics, University of Wisconsin - Madison, 2003. available from `http://www.stat.wisc.edu/techreports/tr1086.pdf`.

[6] Philip Dawid and Stephen Senn. Statistical model selection. In *Simplicity, Complexity and Modelling*, pages 11–33. John Wiley and Sons, Ltd, Hoboken, New Jersey, 2011.

[7] S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457):77–87, 2002.

[8] M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the american statistical association*, pages 577–588, 1995.

[9] P. J. Green and S. Richardson. Modelling heterogeneity with and without the dirichlet process. *Scandinavian journal of statistics*, pages 355–375, 2001.

[10] J.A. Hartigan and M.A. Wong. A k-means clustering algorithm. *Royal Statistical Society*, 28:100, 1979.

[11] S. Jain and R. M. Neal. A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1):158–182, 2004.

[12] S. Jain and R. M. Neal. Splitting and merging components of a nonconjugate dirichlet process mixture model. *Bayesian Analysis*, 2(3):445–472, 2007.

[13] L. Kaufman and P.J. Rousseeuw. *Finding groups in data : an introduction to cluster analysis*. Wiley-Interscience, New Jesey, 1990.

[14] P.M Kroonenberg. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Springer Complete Collection*, 45:69, 1980.

[15] M. Lavine and M. J. Schervish. Bayes factors: What they are and what they are not. *The American Statistician*, 53(2), 1999.

[16] S. N. MacEachern and P. Mueller. Estimating mixture of dirichlet process models. *Journal of Computational and Graphical Statistics*, 7:223–238, 1998.

[17] P. McCullagh and J. Yang. How many clusters. *Bayesian Analysis*, 3(1):101–120, 2008.

[18] G.J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley-Interscience, New Jesey, 2008.

[19] R. M. Neal. Bayesian mixture modeling by monte carlo simulation. In *Maximum Entropy and Bayesian Methods:Proceedings of the 11th International Workshops on Maximum Entropy and Bayesian Methods of Statistical Analysis, Seattle*, pages 197–211. Dordrecht: Kluwer Academic Publishers, 1992.

[20] R. M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[21] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of number of clusters. *Proc. 17th Int. Conf. Machine Learning (ICML"00)*, pages 727 –734, 2000.

[22] F. A. Quintana and P. L. Iglesias. Bayesian clustering and product partition models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(2):557–574, 2003.

[23] M. Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 19:795–809, 2000.

[24] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[25] Weixin Yao and B. G. Lindsay. Bayesian mixture labeling by highest posterior density. *Journal of the American Statistical Association*, 104(486):758, 2009.

Appendix: R Code for Non-MCMC DPM Clustering Algorithm

```
## A function to calculate the value of log pdf for a single observation.
## x --- A single observation. Each dimension of x follows a Student’s
##       t-Distribution.
## mu_lsigma --- A vector contains the values of mean and log standard
##               deviation for each dimension of x.
## df --- Degrees of freedom for each dimension of x, which is assumed to
##        be the same.
## The value of log pdf for a single observation x is returned.
log_prob_t <- function(x, mu_lsigma, df)
{ p <- length(x)
  mu <- mu_lsigma[1:p]
  lsigma <- mu_lsigma[-(1:p)]
  sum(dt((x-mu)/exp(lsigma), df, log=TRUE)) - sum(lsigma)
}


## A function to caluculate the negative value of log likelihood function
## for all observations.
## data --- A data set contains all observations.
## mu_lsigma --- A vector contains the values of mean and log standard
## deviation for each dimension of the data.
## df --- Degrees of freedom for each dimension of the data, which is
##        assumed to be the same.
## m_mu --- Mean for Dirichlet Prior mean.
## w_mu --- Standard deviation for Dirichlet Prior mean.
## m_lsigma --- Mean for Dirichlet Prior log standard deviation.
## w_lsigma --- Standard deviation for Dirichlet Prior log standard
##              deviation.
## The negative value of log likelihood function for all observations is
## returned.
minus_log_like_mulsigma <-
   function(mu_lsigma,complete_data,df,m_mu,m_lsigma,w_mu,w_lsigma)
{ num_obs <- nrow(complete_data)
  p <- length(mu_lsigma)/2
  result = (-sum(apply(complete_data,1,log_prob_t,mu_lsigma = mu_lsigma,
             df=df)) - sum(dnorm(mu_lsigma,c(rep(m_mu,p),rep(m_lsigma,p)),
             c(rep(w_mu,p),rep(w_lsigma,p)),log=TRUE)))
  result
}

## A function to caluculate the MLE for mean and log standard deviation
## for each dimension of observations.
## data --- A data set contains all observations.
## mu_lsigma --- A vector contains the values of mean and log standard
##               deviation for each dimension of all observations.
## df --- Degrees of freedom for each dimension of the data, which is
##        assumed to be the same.
## m_mu --- Mean for Dirichlet Prior mu.
## w_mu --- Standard deviation for Dirichlet Prior mu.
## m_lsigma --- Mean for Dirichlet Prior log sigma.
## w_lsigma --- Standard deviation for Dirichlet Prior log sigma.
## A vector contains the MLE for mean and log standard deviation for
## each dimension of observations is returned.
find_mle_mulsigma <-
```

```
   function(ini_mu_lsigma,complete_data,df,m_mu,m_lsigma,w_mu,w_lsigma)
{  ##print(ini_mu_lsigma)
   ##print(minus_log_like_mulsigma)
   optim(par = ini_mu_lsigma, fn = minus_log_like_mulsigma,
         complete_data = complete_data, df = df,m_mu=m_mu,
         m_lsigma=m_lsigma, w_mu=w_mu,
         w_lsigma=w_lsigma, hessian=TRUE)
}

## A function to calculate the log probabilty that a given observation
## belongs to a cluster.
## x --- A single observation from the data set.
## list_c --- A list that contains:
##            1. MLE of mean for each dimension of observations that
##               belongs to Cluster c.
##            2. MLE of log standard deviation for each dimension of
##               observations that belongs to Cluster c.
##            3. Probability that a random observation belongs to
##               Cluster c.
## df --- Degrees of freedom for each dimension of x, which is assumed
##        to be the same.
## The log probabilty that a given observation belongs to a cluster is
## returned.
log_joint_prob <- function(one_data,list_c,df)
{  log(list_c$prob) +
   log_prob_t(one_data,mu_lsigma=c(list_c$mu,list_c$lsigma),df=df)
}

## For a given partition, a function to assign each observation to the
## cluster that it has the highest probability of coming from using
## winner-get-all EM algorithm. This function also calculates the DPC
## value for the given partition.
## data --- A data set contains all observations.
## prob_mu_lsigma --- A list that contains values of parameters for all
##                    clusters. Each element inside this list is also a
##                    list that contains values of parameters for a
##                    single cluster as follows:
##      1. MLE of mean for each dimension of observations
##                       that belongs to this cluster.
##                    2. MLE of log standard deviation for each dimension
##                       of observations that belongs to this cluster.
##                    3. Probability that a random observation belongs to
##                       this cluster.
## df --- Degrees of freedom for each dimension of the data, which is
## assumed to be the same.
## threshold --- A terminate factor for EM algorithm.
## max_iter --- Maximum number of iterations for EM algorithm.
## m_mu --- Mean for Dirichlet Prior mu.
## w_mu --- Standard deviation for Dirichlet Prior mu.
## m_lsigma --- Mean for Dirichlet Prior log sigma.
## w_lsigma --- Standard deviation for Dirichlet Prior log sigma.
## alpha --- A positive scaling parameter used for calculating DPC value.
## The clustering result is returned. It includes the following:
## 1. A list contains the MLE for all the parameters for the given
##    partition.
## 2. A condtional matrix with number of observations as number of rows,
##    number of clusters as number of columns, and each element represents
##    the probability that a single observation belongs to a given cluster.
## 3. A vector that contains the cluster number for each observation that
##    it has the highest probability of coming from.
## 4. DPC value for the given partition.
cls_t <- function(complete_data,prob_mu_lsigma,df,threshold,max_iter,debug,
                  m_mu,m_lsigma,w_mu,w_lsigma,alpha)
{  num_var <- ncol(complete_data)
   num_obs <- nrow(complete_data)
   num_cluster <- length(prob_mu_lsigma)
   ## nlm list for one cluster
   nlm_list = list(list())
   ## nlm list for all the clusters
   total_nlm_list = list(list(list()))
   cond_prob <- matrix(0,num_obs,num_cluster)
   log_lik <- 0
   index <- 1
   while(index < max_iter)
   { log_lik_prev <- log_lik

     ## E - step
     for(c in 1: num_cluster)
     { cond_prob[,c] <-
           apply(complete_data,1,log_joint_prob,
                 list_c=prob_mu_lsigma[[c]],df=df)
     }
     log_prob_mar <- apply(cond_prob,1,log_sum_exp)
     cond_prob <- exp(cond_prob - log_prob_mar )
     clusters <- apply(cond_prob,1,which.max)
     log_lik <- sum(log_prob_mar)

     ## M - step
     cond_prob_sum <- apply(cond_prob,2,sum)
```

```
  if(debug)
      cat("Iter =",index,",","Log likelihood =",log_lik,",",
            "\n Cluster freq =", cond_prob_sum/num_obs,"\n")
  for( c in 1:num_cluster)
  {
     ## update overall probability
     prob_mu_lsigma[[c]]$prob <-
         cond_prob_sum[c] / num_obs

     ## update mu, and lsigma,
     cases_sel = which(clusters == c)

     if(debug) cat("cluster id is ",c,"\n")

     for( i in 1:num_var)
     {   result_min <- find_mle_mulsigma(
         ini_mu_lsigma = c(prob_mu_lsigma[[c]]$mu[i],
                          prob_mu_lsigma[[c]]$lsigma[i]),
         complete_data = complete_data[cases_sel,i,drop=FALSE],
         df=df,m_mu=m_mu,m_lsigma=m_lsigma,w_mu=w_mu,w_lsigma=w_lsigma)
         nlm_list[[i]] =  result_min
         prob_mu_lsigma[[c]]$mu[i] <- result_min$par[1]
         prob_mu_lsigma[[c]]$lsigma[i] <- result_min$par[2]
     }
     total_nlm_list[[c]] = nlm_list
  }
  if(abs(log_lik-log_lik_prev) < threshold) break
  index = index + 1
}
total_cluster_det = 0
```

```
 total_cluster_mode = 0
 length_list = rep(0,num_cluster)

## Determine whether there is negative det. 0: No. 1: Yes
negative_d = 0

for (c in 1:num_cluster)
{ mode_det = cal_single_cluster_mode_det(num_var,total_nlm_list[[c]])
  if(mode_det$neg_det == TRUE)
  { negative_d = 1
    break
  }
  single_cluster_det = mode_det$single_cluster_det
  total_cluster_det = total_cluster_det + single_cluster_det
  signle_cluster_mode = mode_det$single_cluster_mode
  total_cluster_mode = total_cluster_mode + signle_cluster_mode
  cases_sel = which(clusters == c)
  length_list[c] = length(cases_sel)
}

if(negative_d == 0)
{ log_integal = -total_cluster_mode + num_var * log(2*pi) -
  0.5 * total_cluster_det
  diri_value = log_integal + num_cluster * log(alpha) +
  sum(lgamma(length_list))
} else
{ diri_value = -Inf
}

list( pars = prob_mu_lsigma, cond_prob = cond_prob,
        avg_log_lik = log_lik, df = df, clusters = clusters, diri_value
        = diri_value, length_list = length_list)
}


## A function that applys the clustering method over the data set, and
## outputs the final clustering result.
## data --- A data set contains all observations.
## df --- Degrees of freedom for each dimension of the data, which is
##        assumed to be the same.
## threshold --- A terminate factor for EM algorithm.
## max_iter --- Maximum number of iterations for EM algorithm.
## m_mu --- Mean for Dirichlet Prior mu.
## w_mu --- Standard deviation for Dirichlet Prior mu.
## m_lsigma --- Mean for Dirichlet Prior log sigma.
## w_lsigma --- Standard deviation for Dirichlet Prior log sigma.
## alpha --- A positive scaling parameter used for calculating DPC value.
## range --- A value to determine how far to split the orginal centroid.
## The final clustering result is returned. It includes the following:
## 1. A list contains the MLE for all the parameters for all clusters.
## 2. A condtional matrix with number of observations as number of rows,
##    number of clusters as number of columns, and each element represents
##    the probability that a single observation belongs to a given cluster.
## 3. A vector that contains the cluster number for each observation that
##    it has the highest probability of coming from.
## 4. DPC value for the final partition.
split_process <- function(data,df,threshold,max_iter,debug,
                       alpha,m_mu,m_lsigma,w_mu,w_lsigma,range)
{ num_var = ncol(data)
  num_obs = nrow(data)
  list = generate_first_list(data)
```

```
result = cls_t(data,list,df,threshold,max_iter,debug,m_mu,m_lsigma,
               w_mu,w_lsigma,alpha)

## Set the oiginal model to be the best model to begin with.
best_result = result
## Set the original Dirichlet value to be the best value to begin with.
best_diri_value = result$diri_value
num_cluster = length(result$pars)
updated_num_cluster = 0
split_index = c("1st", "2nd", "3rd", "4th", "5th")
trace = 1

while (updated_num_cluster != num_cluster)
{ list = result$pars
  log_like = result$avg_log_lik
  prob_mat = result$cond_prob
  clusters = result$clusters
  updated_num_cluster = length(list)
  num_cluster = updated_num_cluster

  for (c in 1:num_cluster)
  { case_select = which(clusters == c)
    if (length(case_select) == 1)
    { break
    }
    data_split = data[case_select,,drop=FALSE]
    select_cluster_mu = list[[c]]$mu
    split_mu = generate_split_list_mu(data_split,select_cluster_mu,
                                      range)
    split_list = list
```

```
l = length(list) + 1
split_list[[l]] = split_list[[c]]
split_list[[c]]$mu = split_mu[[1]]
split_list[[l]]$mu = split_mu[[2]]
split_list[[l]]$lsigma = split_list[[c]]$lsigma
mother_prob = split_list[[c]]$prob
split_list[[c]]$prob = (1/2) * mother_prob
split_list[[l]]$prob = split_list[[c]]$prob
new_result = cls_t(data,split_list,df,threshold,max_iter,debug,
                     m_mu,m_lsigma,w_mu,w_lsigma,alpha)
new_list = new_result$pars
new_clusters = new_result$clusters
new_diri_value = new_result$diri_value

## Determine if any cluster has no data points inside it.
empty_cluster = FALSE
for( ind in 1:length(new_list))
{ if(length(which(new_clusters == ind)) == 0)
  { empty_cluster = TRUE
    break
  }
}

## If empty clusters exist.
if(empty_cluster == TRUE)
{ new_diri_value = -Inf
}

cat("new diri value", new_diri_value, "\n")
cat("old diri value", best_diri_value, "\n")
```

```
      if(new_diri_value > best_diri_value)
      { trace = trace + 1
        best_result = new_result
        best_diri_value = new_diri_value
      }
     }
     result = best_result
     updated_num_cluster = length(result$pars)
   }

  list = result$pars
  log_like = result$avg_log_lik
  prob_mat = result$cond_prob
  clusters = result$clusters
  diri_value = result$diri_value
  list( pars = list, cond_prob = prob_mat, df = df, clusters = clusters,
        diri_value = diri_value)
}

## A function to calculate the log determinate for a given cluster.
## num_var --- Number of dimension for the data set.
## nlm_list --- A list obtained after applying General-purpose
##              optimization (optim) over the -log likelihood function
##              for all observations belong to a given cluster.
## The value of log determinate and the minimum value of -log likelihood
## function for a given cluster are returned.
cal_single_cluster_mode_det <- function(num_var,nlm_list)
{ log_det = 0
  neg_log_like_mode = 0
  temp = FALSE
  for(i in 1:num_var)
  {
    if (det(nlm_list[[i]]$hessian) < 0)
    { temp = TRUE
      break
    }

    temp_det = log(det(nlm_list[[i]]$hessian))
    log_det = log_det + temp_det
    temp_mode = nlm_list[[i]]$value
    neg_log_like_mode = neg_log_like_mode + temp_mode
  }
  list(single_cluster_mode = neg_log_like_mode, single_cluster_det =
       log_det, neg_det = temp)
}

## A function to calculate the value of Log Sum of Exponentials for
## components inside a vector.
## lx - A vector which contains several components.
## The value of Log Sum of Exponentials for components inside a vector
## is returned.
log_sum_exp <- function(lx)
{  mlx <- max(lx)
   log( sum( exp(lx - mlx) ) ) + mlx
}

## A function to generate initial variable list.
## data --- A data set contains all observations.
## An initial variable list is returned.
```

```
generate_first_list <- function(data)
{ num_var <- ncol(data)
  mu = apply(data,2,median)
  lsigma = rep(0,num_var)
  prob_mu_l = list(list(prob=1,mu=mu,lsigma=lsigma))
  prob_mu_l
}

## A function to split a data set into two data sets using PCA.
## data_split --- The data set that needs to be splitted.
## mu --- Centroid for the original data set.
## low_range --- The lower range for how far to split the orginal
##               centroid.
## up_range --- the upper range for how far to split the orginal
##              centroid.
## A data set is splitted into two. The centroids for tow new data
## sets are returned.
generate_split_list_mu <- function(data,mu,range)
{ pca_data = prcomp(data,center = T, scale = T)
  r_matrix = pca_data$rotation
  sd_data = pca_data$scale
  mu_1 = r_matrix[,1] * range
  mu_2 = r_matrix[,1] * -range
  mu_1 = mu_1 * sd_data
  mu_2 = mu_2 * sd_data
  mu_1 = mu_1 + mu
  mu_2 = mu_2 + mu
  temp = list(c())
  temp = rep(temp,2)
  temp[[1]] = mu_1
  temp[[2]] = mu_2
  temp
}

## A function to calculate the total number of parameters for a partition.
## num_cluster --- Number of clusters.
## num_var --- Number of parameters inside each cluster.
## Toal total number of parameters is returned.
cal_num_parameters <- function(num_cluster,num_var)
{ num_cluster * 2 * num_var + num_cluster
}

#################### a generic crossvalidation function ####################
## X --- features with rows for cases
## y --- a vector of response values
## no_fold --- number of folds in cross validation
## trpr_fn --- function for training and prediction:
##  the arguments of trpr_fn must include X_tr, y_tr, X_ts
##  the outputs of trpr_fn must include probs_pred
## ... --- other arguments needed by trpr_fn other than X_tr, y_tr,
##         X_ts
cross_vld <- function (trpr_fn, no_fold = 10, X, y, ...)
{
  if (!is.matrix(X)) stop ("'X' must be a matrix with rows for cases")

  n <- nrow(X)
  nos_g <- as.vector (tapply (rep(1,n), INDEX = y, sum))
  if (any(nos_g < 2)) stop ("less than 2 cases in some group in your
                           data")
  G <- length (nos_g)
  ## partition all cases into no_fold subsets
  ## This function partitions a set of observations into subsets of
  ## almost equal size. The result is used in crossvalidation
  epartition <- function(items, g, randomize = FALSE)
  {
    n <- length (items)
    m <- ceiling (n/g)
    index_par <- matrix ( c(1:n, rep (0, g * m - n)), nrow=g, ncol=m )
    items_par <- rep (list (""), g)

    if (randomize) items <- items [sample (1:n)]

    for(i in 1:g) items_par [[i]] <- items [index_par[i,]]

    items_par
  }

  list_testset <- epartition (1:n, no_fold)
  probs_pred <- matrix (0,n,G)

  for (i_test in 1:no_fold)
  {
      ts <- list_testset [[i_test]]
      tr <- (1:n)[- (ts)]

result_vld <- trpr_fn (
       X_tr = X[tr,, drop = FALSE], y_tr = y[tr],
       X_ts = X[ts,, drop = FALSE], ...)
probs_pred [ts,] <- result_vld $ probs_pred
```

```
  }

  ## replace the predictive probabilities of the last fold of cv  with
  ## predictive probabilities of all cases
  ## note: only the training result of the last fold of cv is returned.
  result_vld$probs_pred <- probs_pred
  result_vld$y <- y
  result_vld
}

## Given the train data X_tr and feature data y_tr for X_tr, a function
## to predict the probability that each observation inside testing data
## X_ts comes from each feature.
## X_tr --- Train data set.
## y_tr --- Feature data for X_tr.
## num_feature_select --- Number of features are selected from X_tr.
## df --- Degrees of freedom.
## threshold --- A terminate factor for EM algorithm.
## max_iter --- Maximum number of iterations for EM algorithm.
## m_mu --- Mean for Dirichlet Prior mu.
## w_mu --- Standard deviation for Dirichlet Prior mu.
## m_lsigma --- Mean for Dirichlet Prior log sigma.
## w_lsigma --- Standard deviation for Dirichlet Prior log sigma.
## alpha --- A positive scaling parameter used for calculating DPC
## value.
## low_range --- The lower range for how far to split the orginal
##               centroid.
## up_range --- The upper range for how far to split the orginal
##              centroid.
## A prediction matrix is returned with number of X_ts as number of
## rows, number of features as number of columns, and each element
## represents the probability that each X_ts belongs to each feature.
tmix_trpr <- function(X_tr,y_tr,X_ts,df,threshold,
                      max_iter,num_iter,debug,alpha,m_mu,m_lsigma,w_mu,
                      w_lsigma, max_num_cluster,NC,range)
{ result <- split_process(data=X_tr,df=df,
                                threshold=threshold,max_iter=max_iter,
                                debug=debug,alpha=alpha,
                                m_mu=m_mu,m_lsigma=m_lsigma,w_mu=w_mu,
                                w_lsigma=w_lsigma,range = range)

  print (result$clusters)
  print (tbl_cls_type (result$clusters, y_tr) )

  num_cluster = length(result$pars)

  ## computing conditional probabilities of test cases belonging to diff
  ## clusters
  cond_prob <- matrix(0,nrow(X_ts),num_cluster)
  for(c in 1: num_cluster)
  {
    ##print(X_ts_select[1,])
    cond_prob[,c] <-
    apply(X_ts,1,log_joint_prob,
          list_c=result$pars[[c]],df=df)
  }
  log_prob_mar <- apply(cond_prob,1,log_sum_exp)
  cond_prob <- exp(cond_prob - log_prob_mar)

  pred_prob_array = array(0, dim = c(nrow(X_ts), NC, num_cluster))
  for(i in 1:num_cluster)
  { cases_cluster = which(result$clusters == i)
    X_tr_select_cluster = X_tr[cases_cluster,]
    y_tr_select_cluster = y_tr[cases_cluster]
    y_prob = rep(0,NC)
    for(j in 1:NC)
    { y_prob[j] = length(which(y_tr_select_cluster == j))
    }
    y_prob = y_prob/length(cases_cluster)
    pred_prob_array[,,i] = matrix(y_prob, nrow(X_ts), NC, byrow = TRUE)
  }

  for(c in 1:num_cluster)
  {
    pred_prob_array[,,c] = pred_prob_array[,,c] * cond_prob[,c]


  }
  overall_predprob = apply (pred_prob_array, c(1,2),sum)
  list(probs_pred = overall_predprob)
}
```