Optimisation of Likelihood for Bernoulli Mixture Models

by

Faezeh Frouzesh

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Master of Science
in Statistics and Operations Research

Victoria University of Wellington
2012

Abstract

The use of mixture models in statistical analysis is increasing for datasets with heterogeneity and/or redundancy in the data. They are likelihood based models, and maximum likelihood estimates of parameters are obtained by the use of the expectation maximization (EM) algorithm. Multi-modality of the likelihood surface means that the EM algorithm is highly dependent on starting points and poorly chosen initial points for the optimization may lead to only a local maximum, not the global maximum. In this thesis, different methods of choosing initialising points in the EM algorithm will be evaluated and two procedures which make intelligent choices of possible starting points and fast evaluations of their usefulness will be presented. Furthermore, several approaches to measure the best model to fit from a set of models for a given dataset, will be investigated and some lemmas and theorems are presented to illustrate the information criterion.

This work introduces two novel and heuristic methods to choose the best starting points for the EM algorithm that are named Combined method and Hybrid PSO (Particle Swarm Optimisation). Combined method is based on a combination of two clustering methods that leads to finding the best starting points in the EM algorithm in comparison with the different initialisation point methods. Hybrid PSO is a hybrid method of Particle Swarm Optimization (PSO) as a global optimization approach and the EM algorithm as a local search to overcome the EM algorithm's problem that makes it independent to starting points. Finally it will be compared with different methods of choosing starting points in the EM algorithm.

Produced Publications and Presentation

1. Faezeh Frouzesh, Shirley Pledger and Yuichi Hirose, "A Combined Method for Finding Best Starting Points for Optimisation in Bernoulli Mixture Models", in *Proceedings of the 21st International Conference on Pattern Recognition, Japan*. IEEE Press, 2012 (in press).

2. Faezeh Frouzesh, Yuichi Hirose, Shirley Pledger and Mahdi Setayesh, "A Hybrid Particle Swarm Optimization Approach to Bernoulli Mixture Models," in *Proceedings of the Ninth International Conference on Simulated Evolution And Learning (SEAL'2012), Vietnam*. ACM Press, 2012 (in press).

3. Faezeh Frouzesh, "Evaluation of Starting Points for Optimization in Mixture Models", NZMASP 2011 Conference, 21-24 November 2011, Silverstream Retreat, New Zealand.

Acknowledgments

I have finally come to the end of my journey in completion of my thesis and my graduate studies for master of science in statistics and operations research at Victoria University of Wellington. It has been an honor to work with so many bright professionals in this field to fulfill the requirements to grant this great success. I would like to express my thanks to my first supervisor, Dr. Yuichi Hirose. He played an important role in completion of my thesis. He was a great encourager and a great friend throughout this process. I also extend my highest appreciation to my second supervisor, Professor Shirley Pledger. I greatly appreciate her patience and hard work in the past year while I was finishing my thesis and also her magnificent support as my supervisor during my Honors project.

This has been a great experience in my life and I have made very precious memories during this time. I was so privileged to be next to my husband during this time. His help has been very monumental to complete my work. He has been helping me morally and intellectually through this process and has been the cornerstone of my graduate studies success.

I would also like to offer my special thanks to my parents and my sister and my brother-in-law who supported me in every step of the way to reach to this excellence.

Contents

Chapter 1

Introduction

The determination of groups in clustered data is the main goal of cluster analysis when the observed value is the only available information. The use of clustering methods is increasing due to their applications in new domains in, for example, physics, astronomy, biology and social science. Most clustering techniques are based on heuristic or distance-based methods, such as hierarchical clustering and iterative methods. Their intuitive construction and reasonable computational time are two major advantages of these methods. Due to the lack of statistical basis to determine the answer to classical questions such as the number of clusters, heuristic methods have been less used for clustering in recent years. Statistical-based clustering techniques are a principal alternative to heuristic-based methods. In this area the data come from a mixture of probability distributions representing different clusters. Finite mixture models have also been widely applied to a variety of problems such as image analysis, survival analysis and discriminant analysis. From both theoretical and practical points of view, finite mixture distributions have attracted researchers. Although most researchers have focused on mixtures models for continuous data, binary or discrete mixture models have a better performance in many pattern recognition problems. This thesis concentrates on multivariate Bernoulli mixture models for binary data sets.

Bernoulli mixture models are determined by the parameters which can be optimised based on likelihood by optimisation methods, such as the EM algorithm.

The Akaike Information criterion (AIC) is an information-theoretic criterion that measures the goodness of fit of the models. Actually, it is a model evaluation criterion which shows a merit or a performance measure for competing models.

The EM algorithm is a standard approach to finding maximum likelihood in a variety of problems where there is incomplete information or missing data. This algorithm was proposed by Dempster et al. in 1977 [9]. The EM algorithm suffers from choosing the initialisation points. The main goal of this thesis is to find a good solution for the initial value problem in the EM algorithm through techniques in computer science (machine learning).

1.1 Motivation

Bernoulli mixture models are frequently used to classify binary data. They are likelihood based models, and the maximum likelihood estimates of parameters are often obtained using the expectation maximization (EM) algorithm. However, multimodality of the likelihood surface means that poorly chosen starting points for optimisation may lead to only a local maximum, not a global maximum. Beside all the advantages of the EM algorithm, such as its simplicity, monotonic convergence and its natural statistical interpretation, it has some general drawbacks which have been noted in the literature [16]. The EM algorithm is an iterative hill-climbing method whose performance is highly dependent on its initial points, especially in the multivariate context because of the multi-modality of the likelihood function. Therefore, its sensitivity to the initial points and trapping in local optima are major disadvantages of the EM algorithm.

1.2 Research Goals

The overall goal of this thesis is to improve the EM algorithm's performance in optimising the likelihoods of finite mixture models. This thesis concentrates on using the EM algorithm to estimate the parameters of the Bernouli mixture models for binary datasets. To achieve the overall goal of the thesis, this thesis focuses on

two major sub-goals:

1. Reducing sensitivity of the EM Algorithm to initialisation points

   To achieve this sub-goal, we investigate the performance of the EM algorithm when it is initialised through different clustering techniques, proposing a novel combined clustering method and comparing its performance with other techniques;

2. Combing the EM algorithm as a local search with the Particle Swarm Optimisation (PSO) algorithm as a global search method that is more likely to avoid trapping in local optima.

1.3 Thesis Contribution

The major contributions of this thesis are:

- Investigating influence of using different clustering techniques for the initialisation of EM Algorithm
- Proposing a combined clustering technique for intelligently choosing initial points
- Adopting likelihood as a fitness function for PSO algorithm and proposing a hybrid PSO method which combines PSO as a global optimisation method with the EM algorithm as a local search method.

1.4 Thesis Outline

The outline of the thesis is as follows:

- Chapter 2: The EM Algorithm for Finite Mixture Models
- Chapter 3: The Maximum Likelihood Method and Information Criterion

- Chapter 4: A Combined Clustering Method for Initialization of EM Algorithm
- Chapter 5: Application of PSO for Optimization of Likelihood
- Chapter 6: Conclusions

Chapter 2

The EM Algorithm for Finite Mixture Models

Finite mixture models provide a natural representation for great flexibility in fitting models for continuous or discrete outcomes that are observed from populations including a finite number of homogeneous subpopulations [16]. The mixture models also provide a convenient formal setting for model-based clustering. There are a lot of applications of finite mixture models in the social and behavioral sciences, biological, physical and environmental sciences, engineering, finance, medicine and psychiatry among many other fields [8]. The most important advantage of mixture models is that the model can have quite a complex distribution through choosing its components to have an accurate local area in order to fit the best distribution to the model. As a result, finite mixture models can be used in situations where a single parametric family is unable to provide an adequate model for local variations in the observed data [8].

To estimate a mixture distribution, a variety of approaches have been used over the years like graphical methods, method of moments, minimum-distance methods, maximum likelihood and Bayesian approaches. The maximum likelihood method is a well-known approach over the last 30 years due to the existence of an associated statistical theory and we concentrate on it over this thesis.

Finding estimation of parameters in finite mixture models is of practical im-

portance in pattern recognition and other related fields. These parameters are estimated by maximizing the likelihood. The EM algorithm is a standard approach to finding maximum likelihood in a variety of problems where there is incomplete information or missing data. This algorithm was proposed by Dempster et al. in 1977 [9]. Beside all the advantages of the EM algorithm, such as its simplicity, monotonic convergence and its natural statistical interpretation, it has some general drawbacks which have been noted in the literature [16]. The EM algorithm is an iterative hill-climbing method whose performance is highly dependent on its initial points, especially in the multivariate context because of the multi-modality of the likelihood function. Therefore, its sensitivity to the initial points and trapping in local optima are the main disadvantages of the EM algorithm.

2.1 Notations

This section list all notations which will be used throughout the thesis. All data sets used in this thesis are from an $n \times p$ matrix $\mathbf{Y}$ of binary data. The value of Y_{ij} is 1 if there is "success "on a Bernoulli trial with parameter θ_{ij}, and 0 if there is failure. In the binary data, the rows may represent species while the columns or variables are the presence or absence over p quadrants or samples. We are interested in knowing if the rows can be grouped (clustered) into two or more relatively homogeneous groups. It is important to know if the species fall into natural groups, so that species are more similar (in their pattern of occurrence) within groups and dissimilar between different groups. The notation that will be used throughout this thesis is as follows:

L = Likelihood
l = log likelihood
l_c = log likelihood under complete knowledge
θ_{ij} = Bernoulli parameter for $Y_{ij}, i = 1, ...n, j = 1, ...p$
z_{ir} = Indicator of row i being in row-group r
π_r = A priori membership probabilities from row-group r $(r = 1, ...R)$
τ_{ir} = A posteriori probability row i from row-group r

2.2 Parametric formulation of mixture model

Let $\mathbf{Y} = \{Y_1, ..., Y_n\}^T$ denote a random sample of size n where Y_i is a p-dimensional random vector of $\mathbb{R}^r$, y_i its realization or the observed value of the random vector Y_i and $f(y_i)$ is its probability density function, where the superscript T denotes vector transpose. We use $\mathbf{Y}$ to represent the entire sample which is an n-tuple of points in $\mathbb{R}^r$. Though we consider the feature vector Y_i as a continuous random vector here, we can still take $f(y_i)$ as a probability function in the case where Y_i is discrete by adopting the count measure. The density of Y_i is assumed to be a mixture of R densities in the mixture model context, such that:

$$f(y_i) = \sum_{r=1}^{R} \pi_r f_r(y_i) \tag{2.1}$$

where π_r is a nonnegative quantity that can be viewed as the weight of the r-th component of the mixture with the following constraint,

$$\sum_{r=1}^{R} \pi_r = 1 \qquad\qquad 0 \leq \pi_r \leq 1. \tag{2.2}$$

In equation (2.1), $f_r(y_i)$ is a density function and called a *component density* of the mixture. Although the number of components R is considered fixed in equations (2.1) and (2.2), in many applications, the value of R is unknown and it has to be inferred from the given data set, along with the mixture of proportions and the parameters in the specified forms for the component densities. In many applications, $f_r(y_i)$ are designated to belong to some parametric family. Then, the component densities $f_r(y_i)$ are indicated as $f(y_i; \theta_r)$, where θ_r denotes the vector of unknown parameters of the model for the r^{th} component density in the mixture. Therefore, the mixture density $f(y_i)$ can be rewritten as

$$f(y_i; \psi) = \sum_{r=1}^{R} \pi_r f(y_i; \theta_r), \tag{2.3}$$

with $\psi = (\theta_{11}, ..., \theta_{1p}, \theta_{21}, ..., \theta_{2p}, ..., \theta_{R1}, ...\theta_{Rp}, \pi_1, ..., \pi_R)$ containing all the unknown parameters in the mixture model. In the case of incomplete data problems,

since the assignment of the observed data is unknown, mixture models are reformulated. If we consider the complete data vector by $X_i = \{Y_i, Z_i\}$ whose only component being observed is Y_i, then its density function is,

$$g(x_i; \psi) = \prod_{r=1}^{R} [\pi_r f(y_i; \theta_r)]^{z_{ir}}$$

where

$$z_{ir} = \begin{cases} 1 & \text{if } i^{th} \text{ subject is in group } r \\ 0 & \text{otherwise} \end{cases} .$$

2.3 Optimizing mixture models via the EM algorithm

This section is an adaptation of Picard [18] that was published in 2007. Expectation-Maximization (EM) is a method used in point estimation. The main goal of the EM algorithm is to estimate the maximum likelihood through an iterative method when the observation is incomplete. The EM algorithm optimizes the observed data likelihood using the simpler MLE computation of the complete data likelihood. According to equation (2.3), the likelihood function for n independent observations from a mixture is:

$$L(\psi; \mathbf{Y}) = \prod_{i=1}^{n} \left\{ \sum_{r=1}^{R} \pi_r f(y_i; \theta_r) \right\}.$$

Since the maximization of the above likelihood with respect to ψ requires an iterative procedure and is not straightforward, the EM algorithm is the best choice for estimating the parameters of a mixture model due to its candid formulation. In this section, we give a general presentation of the EM algorithm followed by its formulation for mixture models.

2.3.1 General EM algorithm

Let χ indicates the complete data sample space from which x arises, Υ represent the observed sample space and Z is the hidden sample space. Let $\chi = \Upsilon \times Z$ and

$x = (y, z)$. We consider the density of the observed data in χ as:

$$g(x; \psi) = f(y; \psi)k(z|y; \psi),$$

where $f(y; \psi)$ indicates the density of the observed given the data in Υ and $k(z|y; \psi)$ shows the conditional density of the missing observation given the data. Now we need to define different likelihoods:$L(y, \psi)$ is the observed/incomplete-data likelihood and $L^c(x; \psi)$ is the unobserved/complete-data likelihood. The likelihood $\log L^c(x; \psi)$ is a function of $\log L(y, \psi)$ and $\log k(z|y; \psi)$ such that:

$$\log L^c(x; \psi) = \log L(y, \psi) + \log k(z|y; \psi),$$

where

$$\log L^c(x; \psi) = \sum_{t=1}^{n} \log g(x_t, \psi),$$

and

$$\log k(z|y; \psi) = \sum_{i=1}^{n} \sum_{r=1}^{R} z_{ip} \log \mathbb{E}\left\{Z_{ir}|Y_i = y_i\right\}.$$

Define

$$\hat{\tau}_{ir} = Pr_{\hat{\psi}}\{Z_{ir} = 1|Y_i = y_i\} = \frac{\hat{\pi}_r f(y_i; \hat{\theta}_r)}{\sum_{l=1}^{R} \hat{\pi}_l f(y_i; \hat{\theta}_l)}.$$

$\Pr\{Z_{ir} = 1\} = \pi_r$ shows the probability of belonging to population r for each data point which is the only available information about the data. The population's weights π_r are considered as *prior* probabilities of assigning to a given population and $\hat{\tau}_{ir}$ are interpreted as their *posterior* probabilities of constituent membership. Briefly, the estimated probability of data point y_t belonging to the first, second, ..., the mixture probabilities are represented by $\hat{\tau}_{i1}, ..., \hat{\tau}_{iR}$.

The EM algorithm is composed of an iterative method to optimize the incomplete-data likelihood indirectly by the use of the current fit for ψ. If we indicate the value of the parameter at iteration h with $\psi^{(h)}$, and define:

$$\begin{aligned} Q(\psi; \psi^{(h)}) &= \mathbb{E}_{\psi^{(h)}}\{\log L^c(X; \psi)|\mathbf{Y}\}, \\ H(\psi; \psi^{(h)}) &= \mathbb{E}_{\psi^{(h)}}\{\log k(Z|Y; \psi)|\mathbf{Y}\}, \end{aligned}$$

where $\mathbb{E}_{\psi^{(h)}}\{.\}$ marks the expectation operator by taking the current fit $\psi^{(h)}$ for ψ, it follows that:

$$\log L(\mathbf{y};\psi) = Q(\psi;\psi^{(h)}) - H(\psi;\psi^{(h)}).$$

The EM algorithm has two steps:

- *E*-step: compute $Q(\psi;\psi^{(h)})$,
- *M*-step: select $\psi^{(h+1)} = \text{Argmax}\{Q(\psi;\psi^{(h)})\}$.

These steps are repeated alternately until the difference between $\psi^{(h+1)}$ and $\psi^{(h)}$ changes by an arbitrarily small amount. There are several stopping rule for the EM algorithm. The difference of log likelihood between two steps could make a stopping rule. Note that, since parameter $\psi^{(h+1)}$ keeps changing, this difference can be fixed if the log-likelihood stuck on a plateau with respect to ψ. One of the most important properties of the EM algorithm which was introduced by Dempster et al. [9] is that after each iteration of the algorithm, the incomplete data log-likelihood increases.

2.4 The EM algorithm's formulation for mixture models

The log-likelihoods are written as the following when they are applied to the special case of mixture models:

$$\begin{array}{lcccl} \log L(y;\psi) & = & \sum_{i=1}^{n} \log f(y_i;\psi) & = & \sum_{i=1}^{n} \log \left\{ \sum_{r=1}^{R} \pi_r f(y_i;\theta_r) \right\} \\ \log L^c(x;\psi) & = & \sum_{i=1}^{n} \log g(x_i;\psi) & = & \sum_{i=1}^{n} \sum_{r=1}^{R} z_{ir} \log\{\pi_r f(y_i;\theta_r)\} \end{array}$$

The *E*-step only needs to compute the conditional expectation of the missing information given the observed data y_i because the complete data log-likelihood is linear in the unobservable data z_{ip} by the use of the current fit $\psi^{(h)}$ for ψ. It leads to

$$Q(\psi;\psi^{(h)}) = \sum_{i=1}^{n} \sum_{r=1}^{R} \mathbb{E}_{\psi^{(h)}}\{Z_{ir}|Y_i = y_i\} \log \pi_r f(y_i;\theta_r),$$

with

$$\mathbb{E}_{\psi^{(h)}}\{Z_{ir}|Y_i = y_i\} = Pr\{Z_{ir} = 1|Y_i = y_i\} = \tau_{ir}^{(h)},$$

and also

$$\tau_{ir}^{(h)} = \frac{\pi^{(h-1)} f(y_i; \theta^{(h-1)})}{\sum_{l=1}^{r} \pi^{(h-1)} f(y_i; \theta^{(h-1)})}.$$

Therefore

$$Q(\psi; \psi^{(h)}) = \sum_{i=1}^{n} \sum_{r=1}^{R} \tau_{ir}^{(h)} \log\{\pi_r f(y_i; \theta_r)\}.$$

The *M*-step needs to compute the global maximization of $Q(\psi; \psi^{(h)})$ with respect to ψ to obtain an updated estimate $\psi^{(h+1)}$. By constrained maximization of the incomplete-data log-likelihood, for finite mixture models we get the following result:

$$\hat{\pi}_r^{(h+1)} = \frac{\sum_{i=1}^{n} \tau_{ti}^{(h)}}{n}.$$

2.5 EM Algorithm for Bernoulli Mixture Models

Given a data set and any statistical model, the ML method (because of its desirable properties) is the most commonly used method to estimate the parameters of the models. For this purpose, we need to know the probability of observing the data set. The E-step or the expectation step computes the expectation of the log-likelihood evaluated using the current estimate for the latent variables. For an $n \times p$ matrix of data from a discrete distribution, the overall likelihood is the product of the likelihoods for the individual rows:

$$L(\theta, \pi|\mathbf{y}) = \prod_{i=1}^{n} \left[\sum_{r=1}^{R} \pi_r \prod_{j=1}^{p} \theta_{rj}^{y_{ij}} (1 - \theta_{rj})^{1-y_{ij}} \right] \tag{2.4}$$

where $\sum_r \pi_r = 1$ and so the log likelihood is:

$$l(\theta, \pi|\mathbf{y}) = \sum_{i=1}^{n} \log \left[\sum_{r=1}^{R} \pi_r \prod_{j=1}^{p} \theta_{rj}^{y_{ij}} (1 - \theta_{rj})^{1-y_{ij}} \right]. \tag{2.5}$$

This assumes independence of rows, and conditional independence of columns within each row (local independence). Consider the missing information as an $n \times R$ group membership matrix, $\mathbf{Z}$, where $z_{ir} = 1$ if case i is in group r otherwise 0. With complete knowledge,

$$l_c(\theta, \pi|\mathbf{y},\mathbf{z}) = \sum_{i=1}^{n}\sum_{r=1}^{R} z_{ir} \sum_{j=1}^{p} \Big[y_{ij} \log(\theta_{rj}) \\ +(1 - y_{ij}) \log(1 - \theta_{rj}) \Big] + \sum_{i=1}^{n}\sum_{r=1}^{R} z_{ir} \log \pi_r. \tag{2.6}$$

The maximisation step or M-step uses the complete data through maximizing using estimation of the missing data from the E-step in order to find parameters. The partial derivative with respect to θ_{rj} is found, and equated to zero, giving an exact mathematical solution,

$$\left[\frac{\partial l_c}{\partial \theta_{rj}}\right] = \sum_{i=1}^{n} z_{ir} \left[\frac{y_{ij}}{\theta_{rj}} - \frac{1 - y_{ij}}{1 - \theta_{rj}}\right] = 0, \ \ \forall r, \forall j \tag{2.7}$$

So,

$$\hat{\theta}_{rj} = \frac{\sum_i z_{ir} y_{ij}}{\sum_i z_{ir}}. \tag{2.8}$$

Also, with complete data, the maximum likelihood estimates of $\pi_r, r = 1, ..., R$ are simply

$$\hat{\pi}_r = \frac{\sum_i z_{ir}}{n} \tag{2.9}$$

i.e. the proportion of the n cases in group r. The other calculation will be in the E-step, where we use current parameter estimates, $\hat{\theta}_{rj}$ and $\hat{\pi}_r$ to find the expected value of z_{ir}. The expected value (mean) of Bernoulli distribution is θ the probability of success. For case i

$$\hat{z}_{ir} = \frac{\pi_r \prod_{j=1}^{p} \theta_{rj}^{y_{ij}} (1 - \theta_{rj})^{1-y_{ij}}}{\sum_{r=1}^{R} \pi_r \prod_{j=1}^{p} \theta_{rj}^{y_{ij}} (1 - \theta_{rj})^{1-y_{ij}}}. \tag{2.10}$$

2.6 Summary

In this chapter, we introduced notations which will be used over this thesis and described the EM algorithm along with its formulations for mixture models and the Bernoulli Mixture Model. In the next chapter, basics of the maximum likelihood estimator and derivation of information criteria such as AIC will be investigated.

Chapter 3

The Maximum Likelihood Method and Information Criterion

There are several ways to measure the relative goodness of a fitted statistical model. In this chapter, different methods are investigated to measure the best model from a set of models for a given data set followed by some Lemmas for more illustration. The Akaike Information Criterion (AIC) and (AICc) are two most well-known approaches which are used in the next chapters to measure the goodness of the fitted model. These methods are based on the minimization of Kullback-Leibler distance between the model and the truth. Since the value of AIC and AICc shows the amount of lost information, the model which loses a minimum amount of information is the best candidate to fit. Therefore, a model with the minimum AIC or AICc is the best model to choose.

3.1 Theoretical background

3.1.1 Maximum Likelihood Estimation

Suppose that we have a random sample of data and an appropriate theoretical model with parameters. Maximum likelihood, ML is the most commonly used method to estimate the parameters of the model to fit the best model to the dataset.

Hence, it is also called Maximum likelihood estimators or MLE. Briefly, the likelihood is measuring some kind of plausibility for possible value of the unknown parameter(s). Let $X_1, X_2, ..., X_n$ be a sequence of random variables. Assume the random variables are independent and identically distributed **(iid)** with common pdf $f(x;\theta)$ and we wish to estimate θ. The likelihood of the parameter for a given data is defined as the following:

$$L(\theta) = \prod_{i=1}^{n} f(x_i;\theta). \tag{3.1}$$

Let $\hat{\theta}$ be a maximum likelihood estimator (MLE):

$$\hat{\theta} = \text{Argmax}_\theta L(\theta) \tag{3.2}$$

where Argmax denotes that $L(\theta)$ has its maximum value at $\hat{\theta}$. The type of distribution of X (e.g. normal, binomial, Poisson, Exponential and etc.) will be specified by our model. However, we do not know (all) the parameters.

Since $\log$ is a monotone increasing function, we use $\log(L(\theta))$ instead of $L(\theta)$. A maximum for $L(\theta)$ is also a maximum for

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log f(x_i;\theta). \tag{3.3}$$

If $\hat{\theta}$ is MLE, it should satisfy the following equation which we often label as an estimating equation or EE:

$$\frac{\partial}{\partial\theta} l(\theta) = \frac{\partial}{\partial\theta} \log L(\theta) = \sum_{i=1}^{n} \frac{\partial}{\partial\theta} \log f(x_i;\theta) = 0. \tag{3.4}$$

3.1.2 Assumption: Regularity Conditions

- Regularity (R1): The pdfs are distinct; i.e., $\theta \neq \theta' \Rightarrow f(x;\theta) \neq f(x;\theta')$.

- (R2): The true value θ_0 is an interior point in the parameter space Ω.

Additional regularity Conditions

- (R3): We can differentiate the pdf function $f(x;\theta)$ with respect to θ twice.
- (R4): $\int f(x;\theta)dx$ is twice differentiable under the integral sign as a function of θ.

3.1.3 Asymptotic Properties of the Maximum Likelihood Estimators

In this section, we discuss the asymptotic properties of the maximum likelihood estimator of a continuous parametric model $\{f(x;\theta); \theta \in \Theta \subset \mathbb{R}^p\}$ with p-dimensional parameter vector θ.

Asymptotic normality: Suppose that $g(x)$ is a true pdf and $G(x)$ is a true cdf respectively and $X = \{X_1, ..., X_n\}$ are independent and identically distributed (i.i.d.) with $g(x)$ and also the density function $f(x;\theta)$ is two times differentiable and

$$J(\theta) = -\int \frac{\partial^2}{\partial\theta\partial\theta^T} \log f(x;\theta) f(x;\theta) dx,$$

and

$$I(\theta) = \int \frac{\partial}{\partial\theta} \log f(x;\theta) \frac{\partial}{\partial\theta^T} \log f(x;\theta) f(x;\theta) dx.$$

Under the above conditions, we have the following properties:

(a) Let $\mathbf{X} = \{x_1, x_2, ..., x_n\}$ be a collection of i.i.d. random variables which have the same probability density function $f(x;\theta_0)$ ($f(x;\theta_0)$ is the true model). Here, θ_0 is the solution of

$$\int f(x;\theta) \frac{\partial f(x;\theta)}{\partial\theta} dx = 0.$$

Furthermore, if $\hat{\theta}_n$ is the maximum likelihood estimator based on n observations, the following properties can be derived:

(i) When $n \to \infty$, the maximum likelihood estimator $\hat{\theta}_n$ converges in probability to θ_0.

(ii) The MLE $\hat{\theta}_n$ has an asymptotic normality: the distribution of $\sqrt{n}(\hat{\theta}_n - \theta_0)$ converges in law to the p-dimensional normal distribution with the zero vector mean and the variance covariance matrix $I(\theta_0)^{-1}$ which $I(\theta_0)$ is not singular as follows:

$$\sqrt{n}(\hat{\theta} - \theta_0) \xrightarrow{d} N[0, I^{-1}(\theta_0)]$$

(b) Assume that $g(x) \neq f(x;\theta)$ for all θ and $\mathbf{X} = \{x_1, x_2, ..., x_n\}$ are observations according to the distribution $g(x)$. Let θ_0 be the solution of

$$\int g(x) \frac{\partial f(x;\theta)}{\partial \theta} dx = 0.$$

In this case, we can state the following properties with respect to the MLE $\hat{\theta}_n$:

(i) The MLE $\hat{\theta}_n$ converges in probability to θ_0 when $n \to \infty$.

(ii) The distribution of $\sqrt{n}(\hat{\theta} - \theta_0)$ with respect to the MLE $\hat{\theta}$ converges in law to the p-dimensional normal distribution so that its mean vector is zero and the variance covariance matrix is $J^{-1}(\theta_0)I(\theta_0)J^{-1}(\theta_0)$ when $n \to \infty$, i.e.

$$\sqrt{n}(\hat{\theta} - \theta_0) \xrightarrow{d} N[0, J^{-1}(\theta_0)I(\theta_0)J^{-1}(\theta_0)].$$

PROOF. [(a),(b)] : By Taylor expansion, there is a θ^* that lies in between $\hat{\theta}$ and θ_0. The first derivative of the maximum log-likelihood $l(\hat{\theta}_n) = \sum_{i=1}^{n} \log f(x_i; \hat{\theta}_n)$ around θ_0, then

$$0 = \frac{\partial l(\hat{\theta}_n)}{\partial \theta} = \frac{\partial l(\theta_0)}{\partial \theta} + \frac{\partial^2 l(\theta^*)}{\partial \theta \partial \theta^T}(\hat{\theta}_n - \theta_0). \tag{3.5}$$

A score function can be defined as:

$$S(x;\theta) = \frac{\partial}{\partial \theta} \log f(x;\theta). \tag{3.6}$$

If we apply Taylor expansion for $S(x;\hat{\theta})$, we have

$$S(x;\hat{\theta}) = S(x;\hat{\theta_0}) + S'(x;\hat{\theta^*})(\hat{\theta} - \theta_0) \tag{3.7}$$

where

$$S'(x;\theta) = \frac{\partial S(x;\theta)}{\partial \theta^T} = \frac{\partial^2 \log f(x;\theta)}{\partial \theta \partial \theta^T}. \tag{3.8}$$

The equation (3.7) can be changed to

$$0 = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} S(x_i;\hat{\theta}) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} S(x_i;\theta_0) + \frac{1}{\sqrt{n}} \sum_{i=1}^{n} S'(x_i;\theta^*)(\hat{\theta} - \theta_0). \tag{3.9}$$

By multiplying and dividing the second term by $\sqrt{n}$ we get

$$0 = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} S(x_i;\theta_0) + \frac{1}{n} \sum_{i=1}^{n} S'(x_i;\theta^*)\sqrt{n}(\hat{\theta} - \theta_0). \tag{3.10}$$

Since $\theta^* \xrightarrow{p} \theta_0$ and based on Uniform Weak Law of Large Numbers (**WLLN**),

$$\frac{1}{n} \sum_{i=1}^{n} S'(x_i;\theta^*) \xrightarrow{p} -J(\theta_0) \tag{3.11}$$

where $J(\theta_0) = -E[S'(X;\theta_0)]$. So

$$\frac{1}{n} \sum_{i=1}^{n} S'(x_i;\theta^*) = -J(\theta_0) + o_P(1). \tag{3.12}$$

We can now rearrange equation (3.10) as the following:

$$\begin{aligned} 0 &= \frac{1}{\sqrt{n}} \sum_{i=1}^{n} S(x_i;\theta_0) + [-J(\theta_0) + o_P(1)]\sqrt{n}(\hat{\theta} - \theta_0) \\ &= \frac{1}{\sqrt{n}} \sum_{i=1}^{n} S(x_i;\theta_0) + (-J(\theta_0))\sqrt{n}(\hat{\theta} - \theta_0) + o_P(1)\sqrt{n}(\hat{\theta} - \theta_0) \end{aligned} \tag{3.13}$$

As we know $o_P(1)\sqrt{n}(\hat{\theta} - \theta_0) = o_P(1)$, so

$$J(\theta_0)\sqrt{n}(\hat{\theta} - \theta_0) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} S(x_i;\theta_0) + o_P(1)$$

Multiply both sides by $J^{-1}(\theta_0)$ giving

$$\sqrt{n}(\hat{\theta} - \theta_0) = J^{-1}(\theta_0)\frac{1}{\sqrt{n}}\sum_{i=1}^{n} S(X_i;\theta_0). + o_P(1) \tag{3.14}$$

Since $E[S(X;\theta_0)] = 0$, we have

$$\begin{aligned} \text{Var}\,[S(X;\theta_0)] &= E[S(X;\theta_0)S(X;\theta_0)^T] - E[S(X;\theta_0)]E[S(X;\theta_0)] \\ &= E[S(X;\theta_0)S(X;\theta_0)^T] \\ &= I(\theta_0). \end{aligned} \tag{3.15}$$

By using the Central Limit Theorem (CLT), $\frac{1}{\sqrt{n}}\sum_{i=1}^{n} S(X_i;\theta_0) \xrightarrow{d} N(0, I(\theta_0))$. Therefore,

$$\sqrt{n}(\hat{\theta} - \theta_0) \xrightarrow{d} N[0, J^{-1}(\theta_0)I(\theta_0)J^{-1}(\theta_0)].$$

If $g(x) = f(x;\theta_0)$ according to the Lemma 1 that is followed by this proof,

$$J^{-1}(\theta_0)I(\theta_0)J^{-1}(\theta_0) = I^{-1}(\theta_0)$$

hence

$$\sqrt{n}(\hat{\theta} - \theta_0) \xrightarrow{d} N[0, I^{-1}(\theta_0)].$$

LEMMA 1.

Assume $g(x) = f(x;\theta_0)$, then $J(\theta_0) = I(\theta_0)$ where

$$J(\theta) = -\int \frac{\partial^2}{\partial\theta\partial\theta^T}\log f(x;\theta)f(x;\theta)dx,$$

and

$$I(\theta) = \int \frac{\partial}{\partial\theta}\log f(x;\theta)\frac{\partial}{\partial\theta^T}\log f(x;\theta)f(x;\theta)dx.$$

PROOF. It is obvious that for all θ we have $\int f(x;\theta)dx = 1$. If we differentiate it, $\int \frac{\partial}{\partial\theta}f(x;\theta)dx = 0$ for all θ. We can multiply $\frac{f(x;\theta)}{f(x;\theta)}$ both sides, so

$$\int \frac{\frac{\partial}{\partial\theta} f(x;\theta)}{f(x;\theta)} f(x;\theta)dx = 0.$$

In view of the fact that $\frac{\partial}{\partial\theta}\log f(x;\theta) = \frac{\frac{\partial}{\partial\theta} f(x;\theta)}{f(x;\theta)}$ for all θ we can reform above equation such that

$$\int \left(\frac{\partial}{\partial\theta}\log f(x;\theta)\right) f(x;\theta)dx = 0.$$

By differentiating the above equation and using product rule, for all θ,

$$\int \left[\frac{\partial^2}{\partial\theta\partial\theta^T}\log f(x;\theta)\right] f(x;\theta)dx + \int \left[\frac{\partial}{\partial\theta}\log f(x;\theta)\right]\left[\frac{\partial}{\partial\theta^T} f(x;\theta)\right] dx = 0.$$

If we multiply the second term by $\frac{f(x;\theta)}{f(x;\theta)}$,

$$\int \left[\frac{\partial^2}{\partial\theta\partial\theta^T}\log f(x;\theta)\right] f(x;\theta)dx + \int \left[\frac{\partial}{\partial\theta}\log f(x;\theta)\right]\left[\frac{\frac{\partial}{\partial\theta^T} f(x;\theta)}{f(x;\theta)}\right] f(x;\theta)dx = 0.$$

Again we can substitute $\left[\frac{\frac{\partial}{\partial\theta^T} f(x;\theta)}{f(x;\theta)}\right]$ by $\left[\frac{\partial}{\partial\theta}\log f(x;\theta)\right]^T$. So

$$\int \left[\frac{\partial^2}{\partial\theta\partial\theta^T}\log f(x;\theta)\right] f(x;\theta)dx + \int \left[\frac{\partial}{\partial\theta}\log f(x;\theta)\right]\left[\frac{\partial}{\partial\theta}\log f(x;\theta)\right]^T f(x;\theta)dx = 0.$$

Therefore, the desired result can be obtained such that at $\theta = \theta_0$, $I(\theta_0) = J(\theta_0)$.

3.1.4 Derivation of Bias for (K-L) distance

This section covers information criteria which are called Takeuchi information criteria. The Kullback-Leibler (K-L) distance between g and $f(.;\hat{\theta})$ is defined by

$$I(g(.), f(.;\theta)) = \int \log g(x)dG(x) - \int \log f(x;\hat{\theta})dG(x).$$

Since the first term on the right hand side is constant, the greater the integral $\int \log f(x;\hat{\theta})dG(x)$ the smaller the K-L distance. Since G is the unknown probability distribution, we replace it with an empirical distribution function G_n. So, $\int \log f(x,\hat{\theta})dG_n(x)$ is an estimator of the integral $\int \log f(x,\hat{\theta})dG(x)$. Hence the bias of this estimator is

$$\begin{aligned} \text{Bias} &= E\{\int \log f(x;\hat{\theta})dG_n(x) - \int \log f(x;\hat{\theta})dG(x)\} \\ &= E\left[\int \log f(x;\hat{\theta})d(G_n - G)(x)\right]. \end{aligned} \tag{3.16}$$

By applying Taylor's expansion,

$$\log f(x;\hat{\theta}) = \log f(x;\theta_0) + S(x;\theta)(\hat{\theta}-\theta_0) + \frac{1}{2}(\hat{\theta}-\theta_0)^T S'(x;\theta^*)(\hat{\theta}-\theta_0) \tag{3.17}$$

where θ^* is located between $\hat{\theta}$ and θ_0.

When $\hat{\theta} \xrightarrow{p} \theta_0$, $\theta^* \xrightarrow{p} \theta_0$. Thus $S'(x;\theta^*) \xrightarrow{p} S'(x;\theta_0)$.

Thus the equation (3.17) can be rewritten as follows:

$$\begin{aligned} \log f(x;\hat{\theta}) &= \log f(x;\theta_0) + S(x;\theta_0)(\hat{\theta}-\theta_0) + \frac{1}{2}(\hat{\theta}-\theta_0)^T \left(S'(x;\theta_0) + o_p(1)\right)(\hat{\theta}-\theta_0) \\ &= \log f(x;\theta_0) + S(x;\theta_0)(\hat{\theta}-\theta_0) + \frac{1}{2}(\hat{\theta}-\theta_0)^T S'(x;\theta_0)(\hat{\theta}-\theta_0) + o_p(1)(\hat{\theta}-\theta_0)^2 \end{aligned} \tag{3.18}$$

By the use of equation (3.18), equation (3.16) can be written as:

$$\begin{aligned} \text{Bias} &= E\left[\int \log f(x;\theta_0)d(G_n - G)(x)\right] \\ &+ E\left[\int S(x;\theta_0)d(G_n - G)(x)(\hat{\theta}-\theta_0)\right] \\ &+ \frac{1}{2}E\left[(\hat{\theta}-\theta_0)^T \int S'(x;\theta_0)d(G_n - G)(x)(\hat{\theta}-\theta_0)\right] \\ &+ E\left[o_p(1)(\hat{\theta}-\theta_0)^2\right]. \end{aligned}$$

Let

$$\begin{aligned} A &= E\left[\int \log f(x;\theta_0) d(G_n - G)(x)\right]; \\ B &= E\left[\int S(x;\theta_0) d(G_n - G)(x)(\hat{\theta} - \theta_0)\right]; \\ C &= \frac{1}{2n} E\left[\sqrt{n}(\hat{\theta} - \theta_0)^T \int S'(x;\theta_0) d(G_n - G)(x) \sqrt{n}(\hat{\theta} - \theta_0)\right]; \\ D &= E\left[o_p(1)(\hat{\theta} - \theta_0)^2\right]. \end{aligned}$$

Therefore, Bias $= A + B + C + D$. Firstly, we start from A:

$$\begin{aligned} A &= E\left[\int \log f(x;\theta_0) dG_n(x) - \int \log f(x;\theta_0) dG(x)\right] \\ &= E\left[\frac{1}{n}\sum_{i=1}^{n} \log f(X_i;\theta_0)\right] - \int \log f(x;\theta_0) dG(x) \\ &= \frac{1}{n}\sum_{i=1}^{n} \int \log f(x;\theta_0) dG(x) - \int \log f(x;\theta_0) dG(x) = 0. \quad (3.19) \end{aligned}$$

Secondly, we compute C. According to the **WLLN**,

$$\int S'(x;\theta_0) dG_n(x) \xrightarrow{p} \int S'(x;\theta_0) dG(x).$$

Hence,

$$\int S'(x;\theta_0) d(G_n - G)(x) = \int S'(x;\theta_0) dG_n(x) - \int S'(x;\theta_0) dG(x) = o_p(1),$$

Then

$$\begin{aligned} C &= \frac{1}{2n} E\left[\sqrt{n}(\hat{\theta} - \theta_0)^T \int S'(x;\theta_0) d(G_n - G)(x) \sqrt{n}(\hat{\theta} - \theta_0)\right] \\ &= \frac{1}{2n} E\left[\sqrt{n}(\hat{\theta} - \theta_0)^T o_p(1) \sqrt{n}(\hat{\theta} - \theta_0)\right] \\ &= \frac{1}{n} o_p(1). \end{aligned}$$

Thirdly, we compute B:

Since $\int S(x;\theta_0)dG(x) = 0$,

$$\begin{aligned} B &= \frac{1}{n}E\left[\sqrt{n}\int S(x;\theta_0)d(G_n - G)(x)\sqrt{n}(\hat{\theta}-\theta_0)\right] \\ &= \frac{1}{n}E\left[\sqrt{n}\int S(x;\theta_0)dG_n(x)\sqrt{n}(\hat{\theta}-\theta_0)\right]. \end{aligned}$$

According to the equation (3.14),

$$\sqrt{n}(\hat{\theta}-\theta_0) = J^{-1}(\theta_0)\frac{1}{\sqrt{n}}\sum_{i=1}^{n} S(X_i;\theta_0) + o_P(1).$$

Since $\frac{1}{\sqrt{n}} = \frac{\sqrt{n}}{n}$,

$$\sqrt{n}(\hat{\theta}-\theta_0) = J^{-1}(\theta_0)\sqrt{n}\int S(x;\theta_0)d(G_n)(x) + o_p(1).$$

Then

$$\begin{aligned} B &= \frac{1}{n}E\left[\sqrt{n}\int S(x;\theta_0)dG_n(x)J^{-1}(\theta_0)\sqrt{n}\int S(x;\theta_0)dG_n(x) + o_P(1)\right] \\ &= \frac{1}{n}\text{tr}\left\{J^{-1}(\theta_0)E\left[\sqrt{n}\int S(x;\theta_0)dG_n(x)\int S(x;\theta_0)dG_n(x)\right]\right\} + \frac{1}{n}o_p(1). \end{aligned}$$

Since $n(\hat{\theta}-\theta_0)^2 = o_p(1)$, $D = \frac{1}{n}o_p(1)$ and Var $(\sqrt{n}\int S(x;\theta_0)dG_n(x)) = I(\theta_0)$,

$$B = \frac{1}{n}\text{tr}\left[J^{-1}I(\theta_0)\right] + \frac{1}{n}o_p(1). \tag{3.20}$$

Altogether the bias of the estimator is given by (3.20).

3.1.5 Takeuchi Information Criterion (TIC)

TIC [26] is defined by

$$TIC = -2\log L(\hat{\theta}) + 2\text{tr}[J^{-1}(\theta_0)I(\theta_0)].$$

According to LEMMA 1, if $g(x) = f(x;\theta_0)$, then $J(\theta_0) = I(\theta_0)$. Therefore,

$$\text{tr}[J^{-1}(\theta_0)I(\theta_0)] = \text{tr}(I_K) = K$$

where K is the dimension of θ and TIC becomes AIC as:

$$AIC = -2\log L(\hat{\theta}) + 2K.$$

The bias in AIC evaluation is derived under the assumption that the true distribution $g(x)$ is contained in the specified parametric model. But TIC is an asymptotically unbiased estimate of expected K-L that is not based on the true model in dataset. Since the bias adjustment term in TIC involves the estimation of the elements of two $K \times K$ matrices $I(\theta_0)$ and $J^{-1}(\theta_0)$ in first and second partial derivatives, it is more complicated to compute than AIC.

AIC with the second-order correction for small sample size n with respect to K is called AICc which is calculated as:

$$AICc = -2\log L(\hat{\beta}) + 2K(n/(n-K-1)). \tag{3.21}$$

3.2 Estimation with nuisance parameter

The proofs of all Theorems and Lemmas in this chapter are based on Dr.Yuichi Hirose's notes. Suppose that we have a model with a partitioned parameter $\theta = (\beta, \eta)$ as:

$$\{f(x;\beta,\eta) : \beta \in \Theta_\beta \subset \mathbb{R}^m, \eta \in \Theta_\eta \subset \mathbb{R}^k\},$$

where β is called a parameter of interest and η is called as a nuisance parameter.

Let $L_n(\beta,\eta) = \sum_{i=1}^n \log f(X_i;\beta,\eta)$. The maximizer of $L_n(\beta,\eta)$ is denoted by $(\hat{\beta},\hat{\eta})$. For any fixed β, consider $\hat{\eta}(\beta)$ as the maximizer of $L_n(\beta,\eta)$: $\hat{\eta}(\beta) = \text{Argmax}L_n(\beta_n)$. So $L_n(\beta,\hat{\eta}(\beta)) = \max_\eta L_n(\beta,\eta)$ where $L_n(\beta,\hat{\eta}(\beta))$ is called a **profile likelihood** for β. We denote the maximizer of $L_n(\beta,\hat{\eta}(\beta))$ by $\hat{\beta}_{PL}$. Thus

$$L_n\left(\hat{\beta}_{PL}, \hat{\eta}(\hat{\beta}_{PL})\right) = L_n(\hat{\beta},\hat{\eta}).$$

We consider the log-likelihood as a function of the emprical cdf function:

$$\begin{aligned}\frac{1}{n}L_n(\beta,\eta) &= \frac{1}{n}\sum_{i=1}^n \log f(x;\beta,\eta) \\ &= \int \log f(x;\beta,\eta) dG_n(x).\end{aligned}$$

Since for any fixed β, the maximizer of $\int \log[f(x;\beta,\eta)]dG_n(x)$ is $\hat{\eta}(\beta)$ and also $\hat{\eta}$ is a function of G_n which is defined by

$$\hat{\eta}(\beta) = \hat{\eta}(\beta, G_n) = \text{Argmax}_n \int \log[f(x;\beta,\eta)]dG_n(x). \tag{3.22}$$

Let

$$\tilde{S}(x;\beta,G) = \frac{\partial}{\partial\beta}\log f(x;\beta,\hat{\eta}(\beta,G)),$$

and

$$\tilde{S}'(x;\beta,G) = \frac{\partial^2}{\partial\beta\partial\beta^T}\log f(x;\beta,\hat{\eta}(\beta,G)).$$

Then the maximizer $\hat{\beta}$ of the profile likelihood is a solution to estimating equation

$$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\tilde{S}(X_i;\hat{\beta},G_n) = 0. \tag{3.23}$$

We call $\tilde{S}(x;\beta_0,F_0)$ the efficient score function which satisfies

$$E[\tilde{S}(X;\beta_0,F_0)S_\eta^T(X;\beta_0,\eta_0)] = 0$$

with

$$S_\eta(x;\beta_0,\eta_0) = \frac{\partial}{\partial\eta}\bigg|_{\eta=\eta_0}\log f(x;\beta_0,\eta).$$

We consider a model with a partitioned parameter $\theta = (\beta,\eta)$ in this section and use the result of asymptotic linearity of $MLE(\hat{\beta})$ in derivation of information criteria in the next section.

THEOREM *1. If $g(x) = f(x;\beta_0,\eta_0)$, then the solution $\hat{\beta}$ of the estimate equation (3.23) is an asymptotic linear estimator,*

$$\sqrt{n}(\hat{\beta}-\beta_0) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\tilde{I}_0^{-1}\tilde{S}(X_i;\beta_0,G) + o_p(1),$$

which implies

$$\sqrt{n}(\hat{\beta}-\beta_0) \xrightarrow{d} N(0,\tilde{I}_0^{-1}),$$

where $\tilde{I}_0 = E\left[\tilde{S}(X;\beta_0,G)\tilde{S}^T(X;\beta_0,G)\right]$ is the efficient information matrix.

PROOF. Notation: for a function $\Phi(G)$ of G, $d_G\Phi$ denotes the Frechet derivative of Φ respect to G. According to Taylor's expansion,

$$\begin{aligned} 0 &= \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\tilde{S}(X_i;\hat{\beta},G_n) \\ &= \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\tilde{S}(X_i;\beta_0,G_n)+\frac{1}{n}\sum_{i=1}^{n}\tilde{S}'(X_i;\beta^*,G_n)\sqrt{n}(\hat{\beta}-\beta_0), \quad (3.24) \end{aligned}$$

where β^* lies in between $\hat{\beta}$ and β_0. Let the first and second part of equation be A and B :

$$\begin{aligned} A &= \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\tilde{S}(X_i;\beta_0,G_n) \\ B &= \frac{1}{n}\sum_{i=1}^{n}\tilde{S}'(X_i;\beta^*,G_n)\sqrt{n}(\hat{\beta}-\beta_0). \end{aligned}$$

We apply Taylor's expansion to A again,

$$A=\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\tilde{S}(X_i;\beta_0,G)+\frac{1}{n}\sum_{i=1}^{n}d_G\tilde{S}(X_i;\beta_0,G^*)\sqrt{n}(G_n-G)$$

where G^* is between G_n and G. Now, we need to prove that the second term in the right hand side of above equation is $o_p(1)$. Let $S_G(x;\beta,G)=d_G\log f(x;\beta,\hat{\eta}(\beta,G))$. By Lemmas 2 and 3 given below

$$\begin{aligned} &\frac{1}{n}\sum_{i=1}^{n}d_G\tilde{S}(X_i;\beta_0,G^*)\sqrt{n}(G_n-G) \\ =\ & E\left[d_G\tilde{S}(X;\beta_0,G)\right]\sqrt{n}(G_n-G)+o_p(1) \\ =\ & E\left[\tilde{S}(X;\beta_0,G)S_G(X;\beta_0,G)\right]\sqrt{n}(G_n-G)+o_p(1) \\ =\ & o_p(1). \end{aligned}$$

Therefore

$$A=\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\tilde{S}(X_i;\beta_0,G)+o_p(1).$$

Now equation (3.24) becomes

$$0 = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \tilde{S}(X_i; \beta_0, G) + o_p(1) + \frac{1}{n} \sum_{i=1}^{n} \tilde{S}'(X_i; \beta^*, G_n) \sqrt{n}(\hat{\beta} - \beta_0).$$

When $\beta^* \xrightarrow{p} \beta_0$ and $G_n \xrightarrow{p} G$, by the Uniform Weak Law of Large Numbers (**UWLLN**),

$$\frac{1}{n} \sum_{i=1}^{n} \tilde{S}'(X_i; \beta^*, G_n) \xrightarrow{p} -\tilde{I}_0.$$

By assuming that $\tilde{I}_0^{-1}$ exists,

$$\sqrt{n}(\hat{\beta} - \beta_0) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \tilde{I}_0^{-1} \tilde{S}(X_i; \beta_0, G) + o_p(1).$$

As we know

$$\text{Var} \left[\tilde{S}(X; \beta_0, G) \right] = E \left[\tilde{S}(X; \beta_0, G) \tilde{S}^T(X; \beta_0, G) \right] = \tilde{I}_0.$$

Since $\text{Var}\,(\tilde{I}_0^{-1} \tilde{S}) = \tilde{I}_0^{-1} \text{Var}\,(\tilde{S}) \tilde{I}_0^{-1} = \tilde{I}_0^{-1}$ and by the central limit theorem, we obtain our desired result such that

$$\sqrt{n}(\hat{\beta} - \beta_0) \xrightarrow{d} N(0, \tilde{I}_0^{-1}).$$

LEMMA *2. Prove that*

$$E \left[d_G \tilde{S}(X; \beta_0, G) \right] = -E \left[\tilde{S}(X; \beta_0, G) S_G(X; \beta_0, G) \right].$$

PROOF. As we know $\int f(x; \beta, \hat{\eta}(\beta, G)) dx = 1$ for all (β, G). When we differentiate this equation, we have

$$\int \frac{\partial}{\partial \beta} f(x; \beta, \hat{\eta}(\beta, G)) = 0.$$

By multiplying and dividing top and bottom of above equation by $f(x;\beta,G)$,

$$\begin{aligned} 0 &= \int \frac{\frac{\partial}{\partial\beta} f(x;\beta,\hat{\eta}(\beta,G))}{f(x;\beta,\hat{\eta}(\beta,G))} f(x;\beta,\hat{\eta}(\beta,G))dx \\ &= \int \tilde{S}(x;\beta_0,G) f(x;\beta,\hat{\eta}(\beta,G))dx, \qquad \text{for all}(\beta,G). \quad (3.25) \end{aligned}$$

If we differentiate (3.25), we have

$$\int d_G\tilde{S}(x;\beta,G) f(x;\beta,\hat{\eta}(\beta,G))dx + \int \tilde{S}(X;\beta,G) d_G f(x;\beta,\hat{\eta}(\beta,G))dx = 0 \quad \text{for all}(\beta,G).$$

By substituting $S_G(x;\beta,G) = \frac{d_G f(x;\beta,\hat{\eta}(\beta,G))}{f(x;\beta,\hat{\eta}(\beta,G))}$,

$$\int d_G\tilde{S}(x;\beta,G) f(x;\beta,\hat{\eta}(\beta,G))dx + \int \tilde{S}(x;\beta,G) S_G(x;\beta,G) f(x;\beta,\hat{\eta}(\beta,G))dx = 0.$$

LEMMA *3. We show*

$$E[\tilde{S}(X;\beta_0,G) S_G(X;\beta_0,G)] = 0 \qquad (3.26)$$

PROOF. According to the chain rule, we have

$$\begin{aligned} S_G(x;\beta,G) &= d_G \log f(x;\beta,\hat{\eta}(\beta,G)) \\ &= \frac{\partial}{\partial\eta} \log f(x;\beta,\hat{\eta}(\beta,G)) d_G\hat{\eta}(\beta,G) \\ &= S_\eta(x;\beta,\hat{\eta}(\beta,G)) d_G\hat{\eta}(\beta,G). \end{aligned}$$

and the efficient score function $\tilde{S}(X;\beta_0,G)$ has the following property,

$$-E\left[\tilde{S}(X;\beta_0,G) S_\eta(X;\beta_0,\eta_0)\right] = 0$$

and $\hat{\eta}(\beta_0;G) = \eta_0$, then

$$\begin{aligned} & E\left[\tilde{S}(X;\beta_0,G) S_G(X;\beta_0,G)\right] \\ = \; & E\left[\tilde{S}(X;\beta_0,G) S_\eta(X;\beta_0,\eta_0)\right] d_G\hat{\eta}(\beta_0,G) = 0. \end{aligned}$$

LEMMA *4. Show*

$$-E[\tilde{S}'(X;\beta_0,G)] = \tilde{I}_0. \tag{3.27}$$

PROOF. If we differentiate equation (3.25) with respect to β we get the following result for all (β, G):

$$\int \left\{ \frac{\partial}{\partial \beta^T} \tilde{S}(x;\beta,G) \right\} f(x;\beta,\hat{\eta}(\beta,G))dx + \int \tilde{S}(x;\beta,G) \left\{ \frac{\partial}{\partial \beta^T} f(x;\beta,\hat{\eta}(\beta,G)) \right\} dx = 0.$$

Consider the second integral,

$$\begin{aligned} & \int \tilde{S}(x;\beta,G) \frac{\frac{\partial}{\partial \beta^T} f(x;\beta,\hat{\eta}(\beta,G))}{f(x;\beta,\hat{\eta}(\beta,G))} f(x;\beta,\hat{\eta}(\beta,G))dx \\ = & \int \tilde{S}(x;\beta,G)\tilde{S}^T(x;\beta,G) f(x;\beta,\hat{\eta}(\beta,G))dx. \end{aligned}$$

Now for all (β, G), we have

$$-\int \left\{ \frac{\partial}{\partial \beta^T} \tilde{S}(x;\beta,G) \right\} f(x;\beta,\hat{\eta}(\beta,G))dx = \int \tilde{S}(x;\beta,G)\tilde{S}^T(x;\beta,G) f(x;\beta,\hat{\eta}(\beta,G))dx.$$

THEOREM *5. Assume that* $g(x) \neq f(x;\beta,\eta)$. *Let*

$$\tilde{S}(x;\beta,G) = \frac{\partial}{\partial \beta} \log f(x;\beta,\hat{\eta}(\beta,G)),$$

and

$$\tilde{S}'(x;\beta,G) = \frac{\partial^2}{\partial \beta \partial \beta^T} \log f(x;\beta,\hat{\eta}(\beta,G)).$$

So the solution of the profile likelihood score equation $\sum_{i=1}^n \tilde{S}(X_i,\hat{\beta}) = 0$ *is asymptotically linear estimator:*

$$\sqrt{n}(\hat{\beta} - \beta_0) = \frac{1}{n} \sum_{i=1}^n \tilde{J}_0^{-1} \tilde{S}(X_i;\beta_0) + o_p(1)$$

which results in

$$\sqrt{n}(\hat{\beta} - \beta_0) \xrightarrow{d} N\left[0, \tilde{J}_0^{-1}\tilde{I}_0\tilde{J}_0^{-1}\right]$$

where $\tilde{J}_0 = -E\left[\tilde{S}'(X;\beta_0)\right]$.

PROOF. By applying Taylor's expansion, we have

$$0 = \frac{1}{n}\sum_{i=1}^{n}\tilde{S}(X_i,\beta_0) + \frac{1}{n}\sum_{i=1}^{n}\tilde{S}'(X_i,\beta^*)(\hat{\beta} - \theta_0)$$

Since $\frac{1}{n} = \frac{1}{\sqrt{n}\sqrt{n}}$, by multiplying $\sqrt{n}$ we obtain

$$0 = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\tilde{S}(X_i,\beta_0) + \frac{1}{n}\sum_{i=1}^{n}\tilde{S}'(X_i,\beta^*)\sqrt{n}(\hat{\beta} - \theta_0) \tag{3.28}$$

Since $\hat{\beta} \xrightarrow{P} \beta_0$ then $\beta^* \xrightarrow{P} \beta_0$. By **UWLLN**, we have

$$\frac{1}{n}\sum_{i=1}^{n}\tilde{S}'(X_i,\beta^*) = -\tilde{J} + o_p(1).$$

Hence equation (3.28) is replaced by

$$0 = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\tilde{S}(X_i,\beta_0) + (-\tilde{J} + o_p(1))\sqrt{n}(\hat{\beta} - \theta_0).$$

When we expand above equation, we get the following result:

$$0 = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\tilde{S}(X_i,\beta_0) + (-\tilde{J})\sqrt{n}(\hat{\beta} - \theta_0) + o_p(1).$$

After rearranging, we have

$$\sqrt{n}(\hat{\beta} - \theta_0) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\tilde{J}^{-1}\tilde{S}(X_i,\beta_0) + o_p(1).$$

According to the CLT,

$$\sqrt{n}(\hat{\beta} - \beta_0) \xrightarrow{d} N\left[0, \tilde{J}_0^{-1}\tilde{I}_0\tilde{J}_0^{-1}\right].$$

3.2.1 Derivation of Bias for K-L distance with Nuisance Parameter

In this section, an expression of bias for K-L distance with nuisance parameter is first derived and then this result is used to develop Profile Likelihood Information Criterion (PLIC).

Let G be the true cdf and $\hat{\eta}(\beta) = \hat{\eta}(\beta, G)$ where the function $\hat{\eta}(\beta, G)$ is defined in (3.22). We now derive an information criteria based on the K-L distance between g and $f(.; \hat{\beta}, \hat{\eta}(\hat{\beta}))$

$$I\left(g(.), f(.; \hat{\beta}, \hat{\eta}(\hat{\beta}))\right) = \int \log g(x) dG(x) - \int \log f(x; \hat{\beta}, \hat{\eta}(\hat{\beta})) dG(x).$$

As you know the first term on the right hand side is constant, therefore the greater the integral $\int \log f(x; \hat{\beta}, \hat{\eta}(\hat{\beta})) dG(x)$ the smaller K-L distance. We use G_n instead of G because G is unknown. So $\int \log f(x; \hat{\beta}, \hat{\eta}(\hat{\beta})) dG_n(x)$ is as an estimator of the integral $\int \log f(x; \hat{\beta}, \hat{\eta}(\hat{\beta})) dG(x)$. The Bias of this estimator is

$$\begin{aligned} \text{Bias} &= E\left[\int \log f(x; \hat{\beta}, \hat{\eta}(\hat{\beta})) dG_n(x) - \int \log f(x; \hat{\beta}, \hat{\eta}(\hat{\beta})) dG(x)\right] \\ &= E\left[\int \log f(x; \hat{\beta}, \hat{\eta}(\hat{\beta})) d(G_n - G)(x)\right]. \end{aligned} \tag{3.29}$$

By applying Taylor's expansion,

$$\begin{aligned} \log f(x; \hat{\beta}, \hat{\eta}(\hat{\beta})) &= \log f(x; \beta_0, \hat{\eta}(\beta_0)) + \tilde{S}(x; \beta_0)(\hat{\beta} - \beta_0) \\ &+ \frac{1}{2}(\hat{\beta} - \beta_0)^T \tilde{S}'(x; \beta^*)(\hat{\beta} - \beta_0) \end{aligned} \tag{3.30}$$

where β^* is between $\hat{\beta}$ and β_0. As we already defined

$$\tilde{S}(x; \beta, G) = \frac{\partial}{\partial \beta} \log f(x; \beta, \hat{\eta}(\beta, G)),$$

and

$$\tilde{S}'(x; \beta, G) = \frac{\partial^2}{\partial \beta \partial \beta^T} \log f(x; \beta, \hat{\eta}(\beta, G)).$$

Since $\hat{\beta} \xrightarrow{p} \beta_0$, then $\beta^* \xrightarrow{p} \beta_0$. Thus

$$S'(x; \beta^*) \xrightarrow{p} S'(x; \beta_0).$$

Now equation (3.31) becomes

$$\begin{aligned}\log f(x;\hat{\beta},\hat{\eta}(\hat{\beta})) &= \log f(x;\beta_0,\hat{\eta}(\beta_0)) + \tilde{S}(x;\beta_0)(\hat{\beta}-\beta_0)\\ &+ \frac{1}{2}(\hat{\beta}-\beta_0)^T(\tilde{S}'(x;\beta_0)+o_p(1))(\hat{\beta}-\beta_0)\\ &= \log f(x;\beta_0,\hat{\eta}(\beta_0)) + \tilde{S}(x;\beta_0)(\hat{\beta}-\beta_0)\\ &+ \frac{1}{2}(\hat{\beta}-\beta_0)^T\tilde{S}'(x;\beta_0)(\hat{\beta}-\beta_0)+o_p(1)\left\|\hat{\beta}-\beta_0\right\|^2.\end{aligned} \tag{3.31}$$

By using equation (3.31) in equation (3.16),

$$\begin{aligned}\text{Bias} &= E\left[\int \log f(x;\hat{\theta})d(G_n-G)(x)\right]\\ &+ E\left[\int \tilde{S}(x;\beta_0)d(G_n-G)(x)(\hat{\beta}-\beta_0)\right]\\ &+ \frac{1}{2}E\left[(\hat{\beta}-\beta_0)^T\int \tilde{S}'(x;\beta_0)d(G_n-G)(x)(\hat{\beta}-\beta_0)\right]\\ &+ o_p(1)E\left[\left\|\hat{\beta}-\beta_0\right\|^2\right].\end{aligned}$$

Let

$$\begin{aligned}A &= E\left[\int \log f(x;\hat{\beta}_0,\hat{\eta}(\beta_0))d(G_n-G)(x)\right];\\ B &= E\left[\int \tilde{S}(x;\beta_0)d(G_n-G)(x)(\hat{\beta}-\beta_0)\right];\\ C &= \frac{1}{2}E\left[(\hat{\beta}-\beta_0)^T\int \tilde{S}'(x;\beta_0)d(G_n-G)(x)(\hat{\beta}-\beta_0)\right];\\ D &= o_p(1)E\left[\left\|\hat{\beta}-\beta_0\right\|^2\right].\end{aligned}$$

Then

$$\text{Bias} = A + B + C + D$$

Using Lemma 5

$$A = E\left[\int \log f(x;\beta_0,\hat{\eta}(\beta_0))d(G_n-G)(x)\right] = 0.$$

Using Lemma 6

$$\begin{aligned} B &= E\left[\int \tilde{S}(x;\beta_0)d(G_n - G)(x)(\hat{\beta} - \beta_0)\right] \\ &= \frac{1}{n}\mathrm{tr}[\tilde{J}_0^{-1}\tilde{I}_0] + \frac{1}{n}o_p(1). \end{aligned}$$

Using Lemma 7

$$\begin{aligned} C &= \frac{1}{2}E\left[\int (\hat{\beta} - \beta_0)^T \tilde{S}'(x;\beta_0)d(G_n - G)(x)(\hat{\beta} - \beta_0)\right] \\ &= \frac{1}{n}o_p(1). \end{aligned}$$

Therefore

$$\text{Bias} = \frac{1}{n}\mathrm{tr}[\tilde{J}_0^{-1}\tilde{I}_0] + \frac{1}{n}o_p(1).$$

LEMMA *5. Prove that*

$$E\left[\int \log f(x;\beta_0,\hat{\eta}(\beta_0))d(G_n - G)(x)\right] = 0.$$

PROOF.

$$\begin{aligned} & E\left[\int \log f(x;\beta_0,\hat{\eta}(\beta_0))d(G_n - G)(x)\right] \\ = \; & E\left[\frac{1}{n}\sum_{i=1}^{n} \log f(X_i;\beta_0,\hat{\eta}(\beta_0))\right] - \int \log f(x;\hat{\beta}_0,\hat{\eta}(\beta_0))dG(x) \\ = \; & \int \log f(x;\hat{\beta}_0,\hat{\eta}(\beta_0))dG(x) - \int \log f(x;\hat{\beta}_0,\hat{\eta}(\beta_0))dG(x) = 0. \end{aligned}$$

LEMMA *6. Prove that*

$$E\left[\int \tilde{S}(x;\beta_0)d(G_n - G)(x)(\hat{\beta} - \beta_0)\right] = \frac{1}{n}tr[\tilde{J}_0^{-1}\tilde{I}_0] + \frac{1}{n}o_p(1).$$

PROOF. As we know,

$$E\left[\int \tilde{S}(x;\beta_0)d(G_n - G)(x)(\hat{\beta} - \beta_0)\right] = \frac{1}{n}E\left[\sqrt{n}\int \tilde{S}(x;\beta_0)d(G_n - G)(x)\sqrt{n}(\hat{\beta} - \beta_0)\right].$$

Since $\int \tilde{S}(x;\beta_0)dG(x) = 0$ the equation above becomes

$$\frac{1}{n}E\left[\sqrt{n}\int \tilde{S}(x;\beta_0)dG_n(x)\sqrt{n}(\hat{\beta} - \beta_0)\right]. \tag{3.32}$$

According to theorem 1,

$$\begin{aligned}\sqrt{n}(\hat{\beta} - \beta_0) &= \frac{1}{n}\sum_{i=1}^{n}\tilde{J}_0^{-1}\tilde{S}(x;\beta_0,G) + o_p(1)\\ &= \sqrt{n}\int \tilde{J}_0^{-1}\tilde{S}(X;\beta_0,G)dG_n(x) + o_p(1).\end{aligned}$$

So equation (3.32) becomes,

$$\begin{aligned}&\frac{1}{n}E\left[\sqrt{n}\int \tilde{S}(x;\beta_0)dG_n(x)\sqrt{n}\int \tilde{J}_0^{-1}\tilde{S}(X;\beta_0,G)dG_n(x) + o_p(1)\right]\\ =\ &\frac{1}{n}\text{tr}\left\{\tilde{J}_0^{-1}E\left[\sqrt{n}\int \tilde{S}(x;\beta_0)dG_n(x)\sqrt{n}\int \tilde{S}(X;\beta_0,G)dG_n(x)\right\} + \frac{1}{n}o_p(1)\right\}\\ =\ &\frac{1}{n}\text{tr}[\tilde{J}_0^{-1}\tilde{I}_0] + \frac{1}{n}o_p(1).\end{aligned}$$

LEMMA 7. *Show*

$$\frac{1}{2}E\left[\int (\hat{\beta} - \beta_0)^T\tilde{S}'(x;\beta_0)d(G_n - G)(x)(\hat{\beta} - \beta_0)\right] = \frac{1}{n}o_p(1) \tag{3.33}$$

PROOF. According to the Weak Law of Large Numbers,

$$\int \tilde{S}'(x;\beta_0)dG_n(x) \xrightarrow{p} \int \tilde{S}'(x;\beta_0)dG(x),$$

so,

$$\int \tilde{S}'(x;\beta_0)d(G_n - G)(x) = \int \tilde{S}'(x;\beta_0)dG_n(x) - \int \tilde{S}'(x;\beta_0)dG(x) = o_p(1).$$

Hence, equation (3.33) becomes

$$\begin{aligned} & \frac{1}{n}E\left[\sqrt{n}(\hat{\beta}-\beta_0)^T\int \tilde{S}(x;\beta_0)d(G_n-G)(x)\sqrt{n}(\hat{\beta}-\beta_0)\right] \\ = \; & \frac{1}{n}E\left[\sqrt{n}(\hat{\beta}-\beta_0)^T o_p(1)\sqrt{n}(\hat{\beta}-\beta_0)\right] \\ = \; & \frac{1}{n}o_p(1). \end{aligned} \tag{3.34}$$

3.2.2 Information Criteria based on profile likelihood (PLIC)

The general formula for PLIC is given by,

$$\text{PLIC} = -2\log L(\hat{\beta}) + 2\text{tr}[\tilde{J}_0^{-1}\tilde{I}_0]$$

When $g(x) = f(x;\beta_0,\eta_0)$, for all β, η, then $\tilde{J}_0 = \tilde{I}_0$. Therefore $\text{tr}[\tilde{J}_0^{-1}\tilde{I}_0] = p$ where p is the dimension of matrix β. In this case, PLIC becomes profile AIC (pAIC).

$$pAIC = AIC = -2\log L(\hat{\beta}) + 2p.$$

3.3 Summary

Since Akaike's Information Criterion (AIC) is the most commonly used powerful method for the selection of a good model and we use as a measure criterion in this thesis, we derived an expression for AIC in this chapter. We also showed the asymptotic linearity of MLE($\hat{\theta}$) and used the results for derivation of bias for Kullback-Leibler (K-L) distance and then derived Takeuchi Information Criterion (TIC) and also AIC based on (K-L) for the parametric models with or without nuisance parameters.

Chapter 4

A Combined Clustering Method for Initialization of EM Algorithm

In most applications, the parameters of a mixture model are estimated by maximizing the likelihood (ML). The EM algorithm proposed by Dempster et al. [9] in 1977 is a standard method for finding maximum likelihood or maximum posteriori in a variety of problems where there is incomplete information or missing data. It is an iterative method entailing 2 steps: E-step and M-step. These two steps are performed alternately until the parameter estimates are converged.

The most important advantages of the EM algorithm are its simplicity in comparison to other similar methods, its monotonic convergence and its natural statistical interpretation [16]. The general drawbacks of the EM algorithm are its slow convergence, trapping in local optima and sensitivity to its starting points. The solution of the EM algorithm is highly dependent on its starting position, especially in a multivariate context. Many methods have been proposed in the literature in order to overcome the problem of choosing initial values such that the EM algorithm can more likely lead to a global maximum. In this chapter, in the context of clustering by finite mixtures, we aim to investigate the performance of the initialising methods such as random starts, K-means and divisive methods. Then we introduce a novel heuristic procedure to intelligently choose the starting points in order to increase the performance of the EM algorithm in binary datasets.

4.1 Related Work for Starting Points

For the first time in 1978, Laird [13] suggested an approach based on a grid search for setting the initial values. Woodward et al. [27] used an ad hoc quasi-clustering technique to obtain starting values for the EM algorithm. Böhning [6] suggested an initial classification of the data by maximizing the within sum of squares criterion. Seidel et al. [24] used a random starts method in the EM algorithm for finite exponential mixtures. Their research showed that starting with several random initial values was preferable in order to ensure that the global maximum is obtained. In 2006, Sara et al. [22] used the K-mean algorithm to initialise the EM algorithm and the results were promising. In 2009, Bessadok et al. [5] suggested an initiative method via variable neighborhood search to overcome the problem of choosing starting points in the EM algorithm which often gets trapped at local maxima. To the authors' knowledge, the K-mean algorithm is the only clustering technique whose centroids have been used as starting points in the EM algorithm. Since there are various clustering methods with different performances, we briefly review hierarchical and partitional clustering techniques followed by the random start method in this section.

4.1.1 Clustering techniques

A number of objects of the same kind or having a set of similar patterns are called cluster, group or class. Regardless of the formal definition of these words, they are used in an intuitive manner. The need for categorizing similar things has been felt since early humans, for instance, being able to recognize many individual objects that shared some characteristics like being alive or lifeless, poisonous or edible and so on. The classification of a two-dimensional mixture of three normal components are shown in Figure 4.1. There are many applications of the classification of elements which is fundamental to most branches of science such as ecology and astronomy. It plays a significant role in zoology and biology especially as a basis for Darwin's theory. Since data clustering is a good technique to make a large data set more understandable and accessible to recall information, it is

used to organise massive data sets. By summarizing a data set through making a small number of groups, several concise pattern descriptions of similarities and differences are provided via group labels. Clustering is a main feature of pattern recognition and machine learning in computer science [12] and works as a main process in Artificial Intelligence. Many clustering methods have been proposed through the years. They are divided to two most popular categories:$Hierarchical\ clustering$ and $Partitional\ clustering$. A brief overview of these methods has been provided in the next sections.

4.1.2 Hierarchical clustering

Hierarchical clustering as a widely used data analysis tool generates a hierarchy of clusters or a cluster tree through merging or splitting methods. A cluster tree means "showing a sequence of clustering with each clustering being a partition of the data set "[14]. Hierarchical clustering is divided into $agglomerative$ ("Bottom-up ") methods and $divisive$ ("top-down ") methods. These techniques can be symbolized by a two-dimensional diagram which is called $dendrogram$. Agglomerative method is a popular method to merge n individuals to groups and algorithms that use dividing to generate groups are called divisive methods. Once divisions are made by hierarchical methods, they are unchangeable. That is to say when an agglomerative method joins two individuals, the individuals cannot be separated and when a divisive method makes a split, it cannot be reversed. Figure 4.2 shows an example of such a dendrogram or cluster tree.

Agglomerative methods

Algorithms that use merging to create the cluster tree are called *Agglomerative hierarchical clustering methods*. Agglomerative methods treat each object as a singleton cluster and begin with as many clusters as objects. These objects are successfully merged until only one cluster is obtained. In other words, in agglomerative approach which is also called a bottom up approach, each of the n elements is assumed as a cluster at the first step and then they are successively merged into

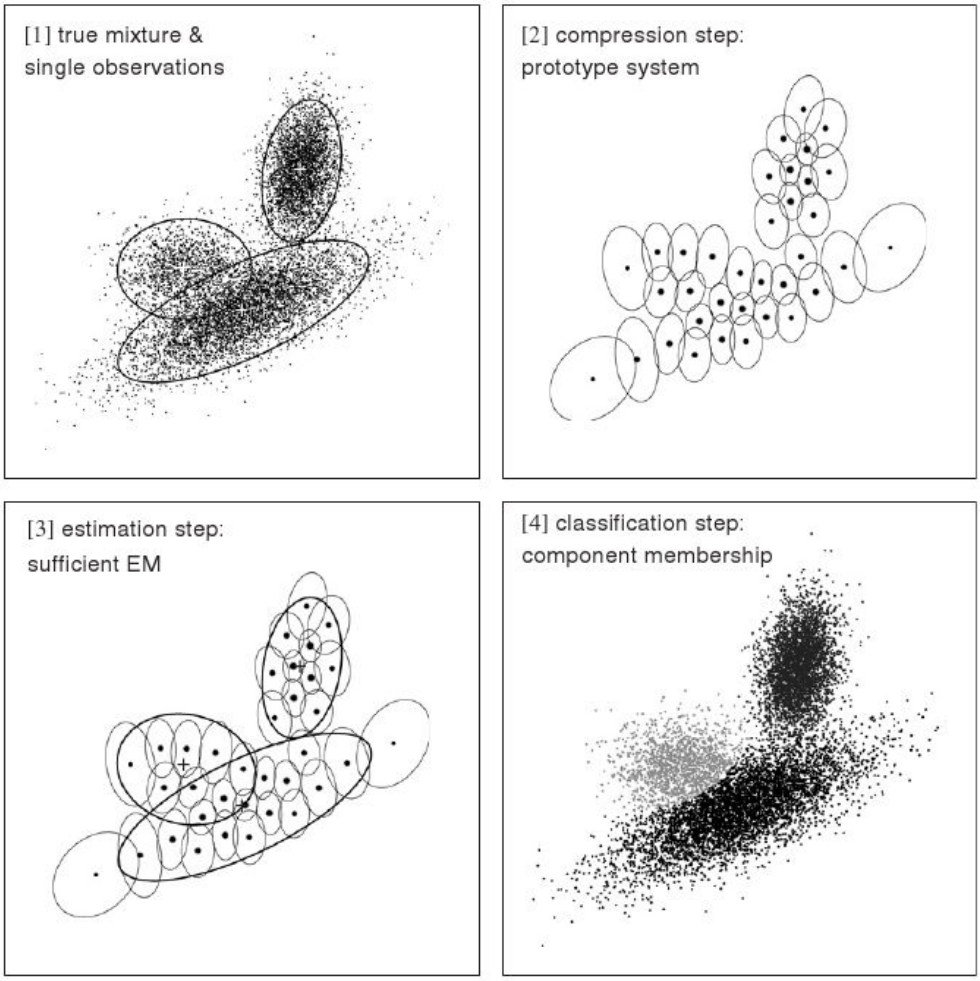

Figure 4.1: Unsupervised classification by estimating a mixture model via sufficient EM on compressed data (these figures are adopted from [25]).

larger clusters until the last partition contains all n elements in a single cluster. This fusion procedure is step by step and at each stage two clusters which are most similar are merged. Based on the definition of this similarity between clusters, several agglomerative methods have been proposed. The most widely used agglomerative clustering methods are: Single linkage clustering, complete linkage clustering, average linkage clustering and average group linkage.

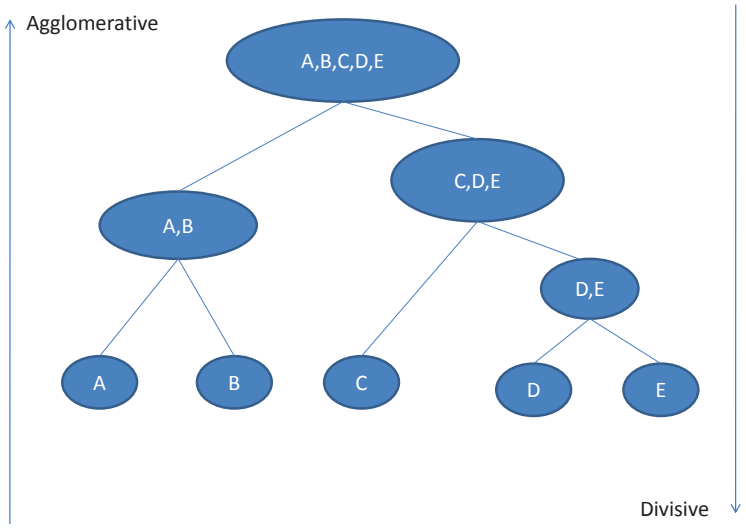

Figure 4.2: Example of a dendrogram or a hierarchical tree structure

Divisive methods

The divisive methods of hierarchical cluster analysis proceed in an opposite way to agglomerative methods. The divisive methods begin with all objects in one cluster and the clusters achieved at the previous step are gradually subdivided into smaller clusters until there are as many clusters as objects. This process is seen in Figure 4.2 as a dendrogram. In other words, at the beginning of such methods, the entire data set containing N objects belongs to a cluster and after $N-1$ steps, there will be N clusters, each including a single object. There are $2^{N-1} - 1$ possible divisions into two sub-clusters at each stage for a cluster containing N objects. The divisive hierarchical clustering method's criterion is based on the distance between two new clusters which is defined by the chosen linkage function. For binary data there are some *monothetic divisive methods* which work based on the presence and absence of each of N variable.

4.1.3 Partitional clustering

Partitional clustering algorithms typically decompose a dataset into a number of disjoint clusters at once. These divisions are based on some predefined objective functions which are optimal. The goal of partitional algorithms as an optimization methods is to minimise certain criteria such as a squared error function. These algorithms usually start with a random partitioning and then refine it iteratively through clustering techniques such as K-means and model based clustering. Partitional clustering attempts to minimize the criterion function which emphasizes the global or local structure of the data by the allocation of clusters to peak in the probability density function. In many researches, it has been shown that partitional clustering is very well-suited to cluster large datasets because of their low computational requirements. Hence, partitional algorithms are more commonly used than hierarchical clustering methods in pattern recognition.

K-means algorithm

K-means is a special case of a general procedure known as the Expectation Maximization (EM) algorithm. This commonly used iterative method, proposed by Forgy [11] in 1965, generates K disjoint flat (non-hierarchical) clusters from n observations. Each cluster is associated with a centroid (center point) and each point is assigned to the cluster with the closest centroid. The number of clusters, K, must be predetermined. Algorithm 1 shows the basic simple K-means algorithm and Figure 4.3 is an example of K-means clustering where $K = 2$.

Advantages and disadvantages of K-means algorithm:

The main advantage of the K-Means Cluster Analysis procedure is that it is much faster than the Hierarchical Cluster Analysis procedure. On the other hand, the hierarchical procedure allows much more flexibility in a cluster analysis: any of a number of distance or similarity measures including options for binary and count data, can be used and the number of clusters is not required to be specified in advance. Once groups are identified, a model can be built which is useful for iden-

Algorithm 1 K-means algorithm

1: Select K initial centroid points randomly (the mean of the points in the cluster is typically chosen as the centroid)
2: **repeat**
3: Create K clusters by assigning all data points to the closest centroid (The distance function should be specified)
4: Recalculate the centroid of each cluster
5: **until** the centroids don't change or relatively few points change clusters

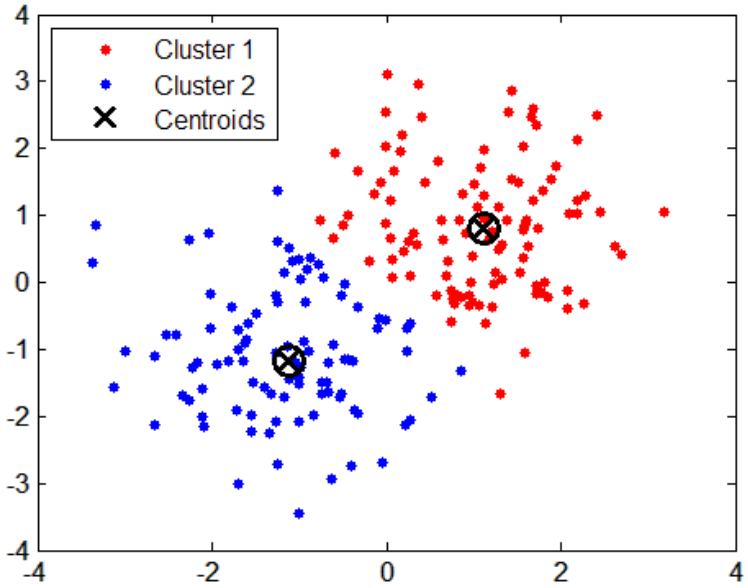

Figure 4.3: An example of K-means clustering where K=2 (this figure is adopted from [23]).

tifying new cases using a discriminant procedure. The saved cluster membership information can be used to explore other relationships in subsequent analyses, such as Crosstabs (process that summarises categorical data to create a contingency table) or GLM (the generalized linear model) Univariate.

Advantages to using K-means clustering techniques are as follows:

- Its implementation is very easy;
- With a large number of variables, it is much faster than the Hierarchical Cluster Analysis procedure;
- The clusters which are produced by K-means can be tighter than the produced clusters by hierarchical clustering, especially if the clusters are globular.

Drawbacks to using K-means:

- This algorithm does not work well with non-globular clusters;
- It can be highly dependent on the initial conditions, which may cause the algorithm to converge to local optima;
- With a fixed number of clusters, the prediction for what K should be, is more difficult.

4.2 New Method

Since the advantages of partitional algorithms are the disadvantages of hierarchical algorithms and vice versa, finding a combination of these algorithms is desirable. Hybrid clustering methods usually have a higher accuracy than other methods. For example, Hammerly and Elkan [12] presented a hybrid method which outperformed other clustering methods, such as the K-means and the Fuzzy C-means algorithms. These methods are linear time algorithms which make them suitable for very large data sets. To find a better method to choose starting points with a higher maximum likelihood, we propose a combined method in this section. The centroids found by a combination of a partitional clustering and divisive method are used in the novel method as the initial points of the EM algorithm. The proposed method has 5 steps:

1. Use the K-means algorithm based on the definition of K centroids and create G (in the first step $G = K$) row clusters because a partial clustering algorithm divides the data set into some specified clusters at once with low time consumption. This gives new pseudo-rows (new row containing of combined rows) which are regarded as a single entity. G is smaller than the final number of the required groups;
2. Apply the divisive method for each group to divide it into two new clusters. A pseudo-row will not be separated when specifying starting points;
3. Consider the clusters resulting from the division of each group along with other groups as G+1 row clusters. We consider all possible sets of G+1 groups and choose the G+1 group, which has the least Akaike information criterion (AIC as will be described in section 4.3.2), as a new pseudo row;
4. Increase G and repeat steps 2 and 3 until the number of required clusters is reached;
5. Set the initial points of the EM algorithm at the centroids of the clusters found at step 4.

4.3 Experiment design

4.3.1 Data Set

To evaluate the initial points found by the K-means and the divisive method and validate the performance of the novel method, we use a dataset from [15] which is the presence/ absence measure of the 25 most abundant plant species ($n = 25$) on 17 plots ($p = 17$) from a grazed meadow in Steneryd Nature Reserve in Sweden. This is a challenging binary data set, as partitioning the rows into (say) two groups has 2^{n-1} possible groupings. It is a substantial data set that is a representative of a range of matrix sizes.

Species	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1.*Festuca.ovina*	38	43	43	30	10	11	20	0	0	5	4	1	1	0	0	0	0
2.*Anemone.nemorosa*	0	0	0	4	10	7	21	14	13	19	20	19	6	10	12	14	21
3.*Stallaria.holostea*	0	0	0	0	0	6	8	21	39	31	7	12	0	16	11	6	9
4.*Agrostis.tenuis*	10	12	19	15	16	9	0	9	28	8	0	4	0	0	0	0	0
5.*Ranunculus.ficaria*	0	0	0	0	0	0	0	0	0	0	13	0	0	21	20	21	37
6.*Mercurialis.perennis*	0	0	0	0	0	0	0	0	0	0	1	0	0	0	11	45	45
7.*Poa.pratenis*	1	0	5	6	2	8	10	15	12	15	4	5	6	7	0	0	0
8.*Rumex.acetosa*	0	7	0	10	9	9	3	9	8	9	2	5	5	1	7	0	0
9.*Veronica.chamaedris*	0	0	1	4	6	9	9	9	11	11	6	5	4	1	7	0	0
10.*Dactylis.glomerata*	0	0	0	0	0	8	0	14	2	14	3	9	8	7	7	2	1
11.*Fraxinus.excelsior.juv*	0	0	0	0	0	8	0	0	6	5	4	7	9	8	8	7	6
12.*Saxifraga.granulata*	0	5	3	9	12	9	0	1	7	4	5	1	1	1	3	0	0
13.*Deschampsia.flexuosa*	0	0	0	0	0	0	30	0	14	3	8	0	3	3	0	0	0
14.*Luzula.campestris*	4	10	10	9	7	6	9	0	0	2	1	0	2	0	1	0	0
15.*Plantago.lanceolata*	2	9	7	15	13	8	0	0	0	0	0	0	0	0	0	0	0
16.*Festuca.rubra*	0	0	0	0	15	6	0	18	1	9	0	0	2	0	0	0	0
17.*Hieracium.pilosella*	12	7	16	8	1	6	0	0	0	0	0	0	0	0	0	0	0
18*Geum.urbanum*	0	0	0	0	0	7	0	2	2	1	0	7	9	2	3	8	7
19.*Lathyrus.montanus*	0	0	0	0	0	7	9	2	12	6	3	8	0	0	0	0	0
20.*Campanula.persicifolia*	0	0	0	0	2	6	3	0	6	5	3	9	3	2	7	0	0
21.*Viola.riviniana*	0	0	0	0	0	4	1	4	2	9	6	8	4	1	6	0	0
22.*Hepatica.nobilis*	0	0	0	0	0	8	0	4	0	6	2	10	6	0	2	7	0
23.*Achillea.millifolium*	1	9	16	9	5	2	0	0	0	0	0	0	0	0	0	0	0
24.*Allium.sp*	0	0	0	0	2	7	0	1	0	3	1	6	8	2	0	7	4
25.*Trifolium.repens*	0	0	6	14	19	2	0	0	0	0	0	0	0	0	0	0	0

Table 4.1: Abundance measures for 25 plant species on 17 plots from a grazed meadow in Steneryd Nature Reserve, Sweden

Species	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1.*Festuca.ovina*	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0
2.*Anemone.nemorosa*	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3.*Stallaria.holostea*	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1
4.*Agrostis.tenuis*	1	1	1	1	1	1	0	1	1	1	0	1	0	0	0	0	0
5.*Ranunculus.ficaria*	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1
6.*Mercurialis.perennis*	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1
7.*Poa.pratenis*	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
8.*Rumex.acetosa*	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0
9.*Veronica.chamaedris*	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
10.*Dactylis.glomerata*	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1
11.*Fraxinus.excelsior.juv*	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1
12.*Saxifraga.granulata*	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	0
13.*Deschampsia.flexuosa*	0	0	0	0	0	0	1	0	1	1	1	0	1	1	0	0	0
14.*Luzula.campestris*	1	1	1	1	1	1	1	0	0	1	1	0	1	0	1	0	0
15.*Plantago.lanceolata*	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
16.*Festuca.rubra*	0	0	0	0	1	1	0	1	1	1	0	0	1	0	0	0	0
17.*Hieracium.pilosella*	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
18*Geum.urbanum*	0	0	0	0	0	1	0	1	1	1	0	1	1	1	1	1	1
19.*Lathyrus.montanus*	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
20.*Campanula.persicifolia*	0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	0	0
21.*Viola.riviniana*	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0
22.*Hepatica.nobilis*	0	0	0	0	0	1	0	1	0	1	1	1	1	0	1	1	0
23.*Achillea.millifolium*	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
24.*Allium.sp*	0	0	0	0	1	1	0	1	0	1	1	1	1	1	0	1	1
25.*Trifolium.repens*	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Table 4.2: Presence/absence of 25 plant species on 17 plots from a grazed meadow in Steneryd Nature Reserve, Sweden

4.3.2 Performance Index

We use AIC as a measure of the goodness of fitted statistical models based on the statistical likelihood function. When we have several candidate models for a given data set, we can rank them according to their AIC, and the model with the minimum AIC is the best model for fitting. The value of AIC shows the amount of lost information when we use a model to estimate the true model. The model which loses the minimum amount of information is the best model to fit. AIC is calculated as:

$$AIC = 2K - 2\log(L) \tag{4.1}$$

where L is the maximized likelihood of a fitted statistical model and K is the number of parameters in the model [2]. The AIC with the second-order correction for small sample size n with respect to K is called AICc which is calculated as:

$$AICc = -2\log(L) + 2K(n/(n-K-1)) \tag{4.2}$$

If AIC or AICc starts to rise through increasing the number of clusters, the clustering process should be stopped, as any more groups cannot be justified statistically.

In the finite mixture context, AIC provides a valid comparison between any two models with the same number of groups. However, the failure of regularity conditions at the boundary of the parameter space makes comparison between models with different numbers of groups more of a problem. Burnham and Anderson [7] has done simulations showing that if two models with different numbers of groups fit in the interior of the parameter space, the use of AIC to compare them is justified [7]. For the fit, to be interior in the space, we require $0 < \hat{\pi}_r < 1$ for each row-group $r = 1, 2, ..., R$, and also that no two groups have identical parameters' estimates; i.e. the model estimates R different and non-empty groups. The question of deciding the number of groups using information criteria is an ongoing research area, with ICL-BIC offering good possibilities (McLachlan and Peel [16], chapter 6).

4.4 Results and Discussion

4.4.1 Evaluation of Random Start as a Starting Point Strategy for EM algorithm

The random start is a common strategy to find the best starting point for the EM algorithm. In this strategy, the EM algorithm is executed several times with different random starts to obtain the best solution. In this strategy, the parameter of Bernoulli distribution, θ is randomly initialised and then the EM algorithm is executed. After the execution of the EM algorithm, the optimised parameter for the Bernoulli mixture model is obtained. This process is repeated n times and the best optimised parameter is chosen as a final result. Since this method is based on a random way, the EM algorithm can more likely avoid trapping in local optima and accordingly it can perform better than other methods in some problems.

To evaluate the effect of the number of the iterations in the random start strategy, we set the number of random starting points at 10. We initially work with the model with three row clusters. We change the number of the iterations from 50 to 550 with the step of 10. Figure 4.4 shows the maximum likelihood resulted from the EM algorithm with different number of iterations. As can be seen in this figure, the EM algorithm converges within a few cycles for the chosen dataset. This implies that there is no advantage in increasing the number of the cycles and the algorithm is still getting locked into a local (not global) maximum likelihood in a small percentage of the starts. To compute the probability of finding the maximum likelihood in the random start method for the binary data, we first need to find the maximum likelihood from 50 to 270 iterations. This investigation leads to the graph plotted in Figure 4.5 which shows the logarithm of the maximum likelihood is -180.871. We do 20 trials with 270 iterations and record the number of the trials whose maximum likelihoods are equal or more than -180.871. By this way, the probability of the highest maximum likelihood is $\frac{3}{20}$ for this dataset. This implies that this probability is very low for the random start method.

In the random start method, there can be more than 10 random starting points. If the number of the points is increased, there is a very gradual increase in the

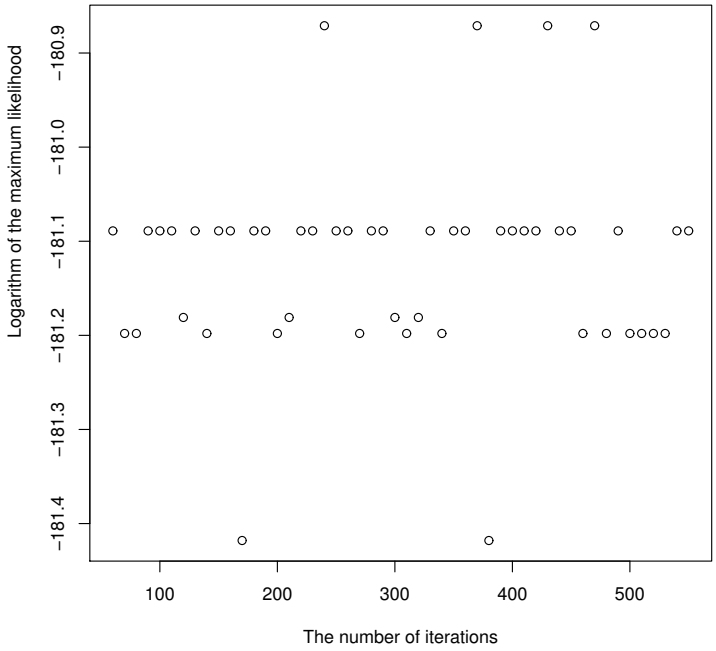

Figure 4.4: Plot of the maximum likelihood found with three row clusters when we increase the iterations

number of times that the global maximum is obtained (as can be seen in Figure 4.6).

For the evaluation purpose, it would be good to switch to using the binomial distribution. If we consider a large number (e.g. 1000 or perhaps 10000) of separate trials and record the maximum likelihood for each trial, we can then classify the outcomes by whether they reach the global maximum ("success ") or only reach a local but not global maximum ("failure "). This gives us an estimate of: p = *probability of reaching global maximum from a random start.* For this purpose, we examined the random start method with 10, 20, 30 and 40 random starts and for each random starts, we executed the algorithm 1000 times. Table 4.3 shows that for more than 30 random starts, we have the probability 0.95 of finding the

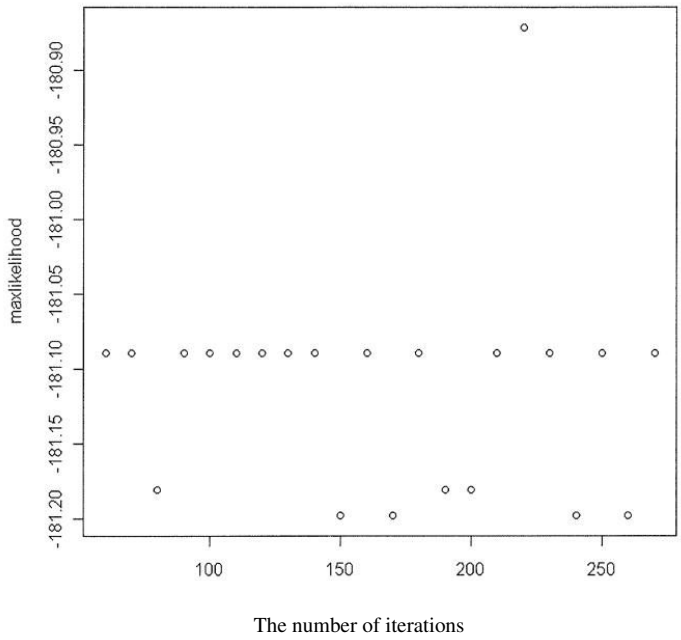

Figure 4.5: Plot of the maximum likelihood found with three row clusters to find the first maximum likelihood when we increase the iterations

global maximum. If we draw a graph of $y = 1 - (1 - P)^x$ where y is the probability of reaching the global maximum and x is the number of random starts, the graph in Figure 4.7 is resulted. This graph shows that we may have probability 0.95 after 25 random starts.

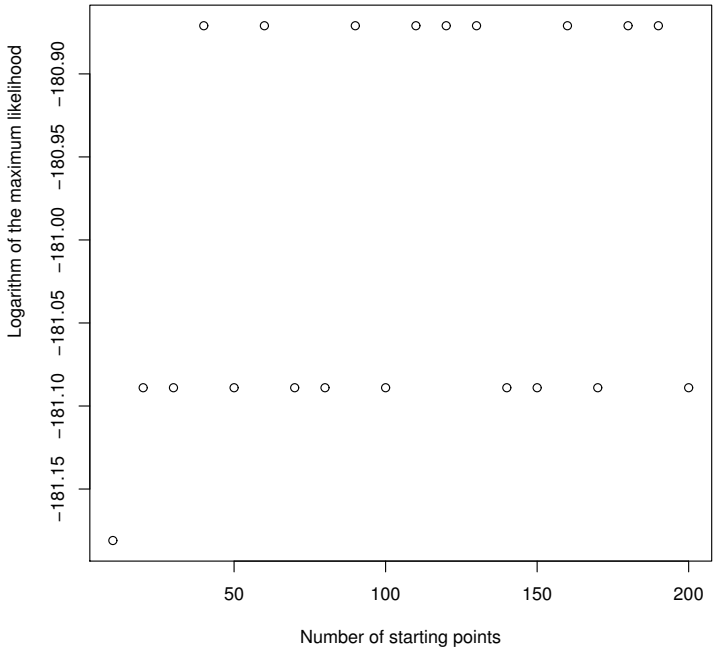

Figure 4.6: Plot of the maximum likelihood found with three row clusters to evaluate the maximum likelihood when we are increasing the number of random starts

4.4.2 Comparison of New Method with Traditional Clustering Methods

We first utilize the new method to choose the initial points of the EM algorithm. Table 4.4 shows the results when we have 3, 4, ... and 8 clusters. This table demonstrates logarithm of the maximum likelihood (Max.ll), residual standard deviation, number of parameters, AIC and AICc of the models. As can be seen in this table, AIC decreases when the number of clusters (column '# of C') is changed from 3 to 4 but it increases when the number of clusters is rising above 4. This implies that the model fitted for 4 row clusters is the best fitted model for this binary data. Since biologists are often interested to have two to six row groups, for

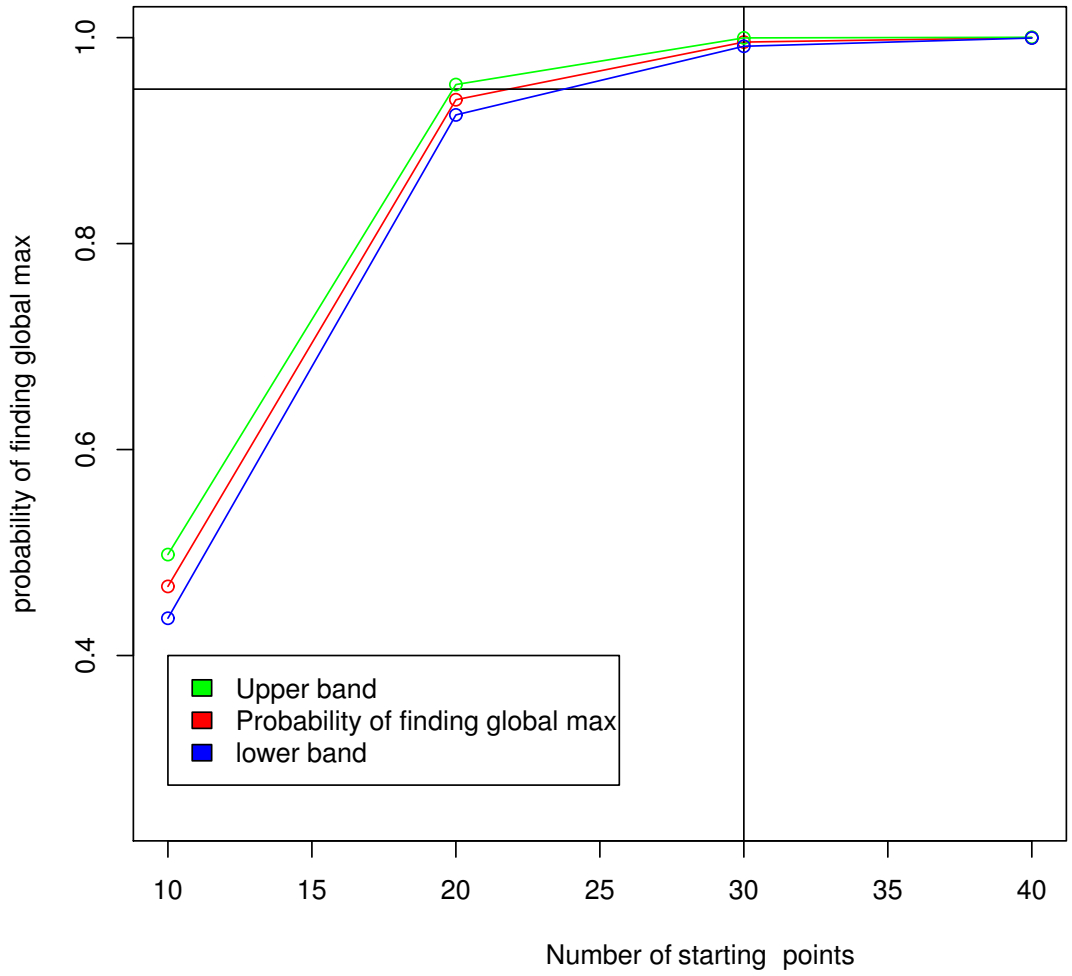

Figure 4.7: 95% Confidence band for p (*probability of reaching global maximum from a random start.*)

# of random starts	probability of finding global maximum	95 % Confidence interval for p	# of runs
10	$P_{10} = 0.4670$	0.4670 ± 0.0309	1000
20	$P_{20} = 0.9396$	0.9396 ± 0.0147	1000
30	$P_{30} = 0.9956$	0.9956 ± 0.0040	1000
40	$P_{40} = 0.9999$	0.9999 ± 0.0003	1000

Table 4.3: Summary of evaluation of random starts analysis and 95% confidence interval

# of C	Max.ll	Res.Dev.	npar	AIC	AICc
3	-180.871	361.742	53	467.742	483.796
4	-160.301	320.602	71	462.602	492.466
5	-145.875	291.750	89	469.750	518.791
6	-133.545	267.091	107	479.744	555.597
7	-125.333	250.665	125	500.665	608.061
8	-117.321	234.642	143	520.642	669.785

Table 4.4: Summary of the results of the Combined method

interpretable results, we compare our new method with K-means, Random starts and Divisive methods when we have 3, 4, 5 and 6 clusters. Table 4.5 shows the results for all methods when the number of clusters is 3 and 4 correspondingly. As can be seen in this table, AIC and AICc for the model, found by the EM algorithm when the combined method is used to select the initial points, are lower than the other methods. This implies that this model which is found by the combined method is better than the others. These methods can be ranked in the order of best performance in 3 and 4 row clusters as the Combined method, Random starts, K-means and Divisive methods. For the case with 5 clusters, the results in Table 4.5 shows that the performances of the Combined and Random starts are better than that of the two other methods. For 6 row clusters, the combined and K-means method perform better than the Random Starts and Divisive methods.

Table 4.6 shows the execution time of the four methods. The combined method's

Method	Max.ll	Res.Dev.	npar	AIC	AICc	R
Random Starts	-181.089	362.178	53	468.178	484.232	3
K-means	-182.664	365.329	53	471.329	487.383	3
Divisive	-196.913	393.825	53	499.825	515.879	3
Combined	-180.871	361.742	53	**467.742**	**483.796**	3
Random Starts	-161.050	322.100	71	464.100	493.963	4
K-means	-162.216	324.432	71	466.432	496.296	4
Divisive	-182.642	365.283	71	507.283	537.147	4
Combined	-160.301	320.602	71	**462.602**	**492.466**	4
Random Starts	-145.875	291.750	89	**469.750**	**518.791**	5
K-means	-146.785	293.571	89	471.571	520.613	5
Divisive	-175.716	351.433	89	529.433	578.475	5
Combined	-145.875	291.750	89	**469.750**	**518.791**	5
Random Starts	-133.498	266.995	107	480.995	555.502	6
K-means	-132.872	265.744	107	**479.744**	**554.250**	6
Divisive	-169.844	339.688	107	553.688	628.194	6
Combined	-132.872	265.744	107	**479.744**	**554.250**	6

Table 4.5: *Comparison of combined method with the traditional methods when we have* $R = 3, 4, 5$ *and 6 row clusters. The best AIC is shown in boldface.*

run time is 140.83 seconds for making 4 row clusters and 440.76 seconds for 6 row clusters. These run times are longer than the other methods' run times; However, the combined method's *AIC or AICc* are better (less) than the other methods.

4.5 Summary

Mixture models are frequently used to classify data. They are likelihood based models, and the maximum likelihood estimates of parameters are often obtained using the expectation maximization (EM) algorithm. However, multimodality of the likelihood surface means that poorly chosen starting points for optimisation may lead to only a local maximum, not a global maximum. In this chapter, different methods of choosing starting points were evaluated and compared, mainly

Method	Run time in seconds for 4 row clusters	6 row clusters
Random starts	23.29	39.49
K-means	2.36	4.13
Divisive/Split off	101.15	254.69
Combined	140.83	440.76

Table 4.6: *Summary of evaluation of run time for four different methods when the number of random starts is 10 and we have 4 and 6 row clusters.*

via simulated data and then we introduced a procedure which made intelligent choices of possible starting points and fast evaluations of their usefulness.

Chapter 5

Application of PSO for Optimization of Likelihood

In this chapter, we aim to present a novel method to overcome the problem of choosing initial point in the EM algorithm. Particle Swarm Optimisation (PSO) is a meta-heuristic method that has successfully been used to solve global optimisation problems. The ease of implementation, fewer operations, limited memory for each particle and high speed of convergence are the main advantages of PSO in comparison with other heuristic methods, such as genetic algorithms [1]. PSO has been used for the optimisation of parameters in Gaussian mixture models [3][21], but it has never been used for optimisation in Bernoulli mixture models up to now. The rows of the matrix of Bernoulli random variables are regarded as particles in the PSO method. The main goals of this paper are as follows:

- investigate the performance of the use of different most well-known clustering techniques to choose the initial points for the EM algorithm in Bernoulli mixture models for binary datasets;
- develop a new PSO-based algorithm as a global search method combined with the EM algorithm as a local search method in order to more likely to cover all the potential solutions;
- develop a new fitness function for the Hybrid PSO-based algorithm along

with a simple encoding scheme;

- compare the performance of the new proposed method to the EM algorithm which is initialised by different clustering methods in terms of the goodness of fitted statistical models.

5.1 PSO as a Global Search

Particle Swarm Optimization (PSO) is a population based stochastic optimization technique. It is a computational method that iteratively optimizes a problem by improving candidate solutions regarding their fitnesses. The population of candidate solutions is called particles and we can move these particles through multi-dimensional search space according to simple mathematical formulae over the particle's position and velocity. Each particle moves based on its personal best position and the best position of the neighborhoods of those particles. Moving the swarm toward the best solutions is expected. Suppose that, there is a population of m particles in PSO. The following information is maintained through each particle:

- x_i: The current position of the particle;
- V_i: The current velocity of the particle.

The position of ith particle is represented as the vector $X_i(t) = (x_{i1}(t), x_{i2}(t), ..., x_{in}(t))$ in an n-dimensional search space at time t. The position of the particle is changed by its own experience (memory influence and particle) and that of its neighbors. A particle's position $X_i(t)$ is updated at each iteration of PSO by adding a velocity $V_i(t)$.

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{5.1}$$

The velocity is changed according to three components: Current motion influence, particle memory influence, and swarm influence:

$$V_i(t+1) = wV_i(t) + C_1r_1(X_{pbest_i} - X_i(t)) + C_2r_2(X_{leader} - X_i(t)) \tag{5.2}$$

where w denotes the inertia weight to control the impact of the previous velocity; C_1 and C_2 are the self and swarm confidence learning factors which control how far a particle will go in a single iteration. In other words, they represent the attraction of a particle toward its best previous position and the best particle of the population; r_1 and r_2 are uniform random variables between 0 and 1; X_{pbest_i} denotes the personal best position of ith particle so far and X_{leader} represents the position of a particle to guide and lead other particles toward better regions of the search space. The basic PSO algorithm can be represented as the following:

```
For each particle P on population do
    Initialize the position and velocity of particle
    Repeat
        For each particle do
            Calculate fitness value
            Update best particle if fitness value is better than best fitness value pbest,
            and set current position as pbest
        For each particle do
            Find local best particle from neighborhood
            Compute particle velocity (5.2)
            Update particle position (5.1)
    Until stopping criterion
```

5.2 Neighborhood Topologies

Topology is an important characteristic of PSO which shows how particles are connected to each other as an information sharing mechanism [20]. The social structure among a swarm's particles is defined through a topology. The topology determines the leader of each particle according to the type of neighborhood graph. Several typical neighborhood topologies have been proposed in the literature as follows:

- **The von Neumann topology (VNT):** In this topology, each particle is surrounded with four neighbors within its neighborhood and exchanges the information with them [20]. These particles are usually situated in four different directions of its position. Figure 5.1(a) shows an instance of this kind of topology in 2-dimensional search space.

- **Tree-based graph(TBG):** This topology is a hierarchical topology which has a root particle at the top level of the tree and all particles in the second level are joined to the root. You can see an example of this hierarchical topology in Figure 5.1(b). The root of each particle plays the role of parent for the child particles in the second level and parents are the leader of each particle in the tree. Whenever the solution found by each child particle is better than the best particle bound through its parent, the personal information is exchanged between the child and parent particles. In this case, $leader = pbest_{parent}$ in equation (5.2).

- **Star graph(SG):** This topology has a star shape, as shown in Figure 5.1(c). In this case, one particle in the center which is named the focal particle, is just connected to all other particles [20]. In this topology particles are separate from each other and they communicate through the focal particle. Sometimes this topology is called the wheel topology. In this topology, $leader = focal$. The focal particle is chose randomly before starting PSO iterations.

- **Ring topology:** Each particle has just two immediate neighbors as shown in Figure 5.1(d) in this case. This topology is based on the local best topology in which $k = 2$. Hence, each particle has a local best particle among two neighbor particles. In this topology, each particle is influenced through a leader in its neighborhood and its own past experience(pbest). The leader is nominated the local best(lbest) particle. Low speed of information sharing among particles and high exploration ability are the main features of the ring topology.

- **Fully connected graph (FCG)**: In this case, the opposite of the empty topology, each particle is fully connected to the other particles and is influenced by the best particle of the entire swarm (gbest) as well as its own past experience(pbest). In this case, $leader = gbest$ in equation (5.2) and an example of this topology is shown in Figure 5.1(e).

In some papers (e.g.,[17]), it has been represented that there is a strong relation between the kind of topology which is chosen for the PSO algorithm and its robustness to premature convergence to optimize some benchmark fitness function. In this section the fully connected topology is used for the PSO algorithm because of its simplicity and speed of convergence and also it is more commonly used topology.

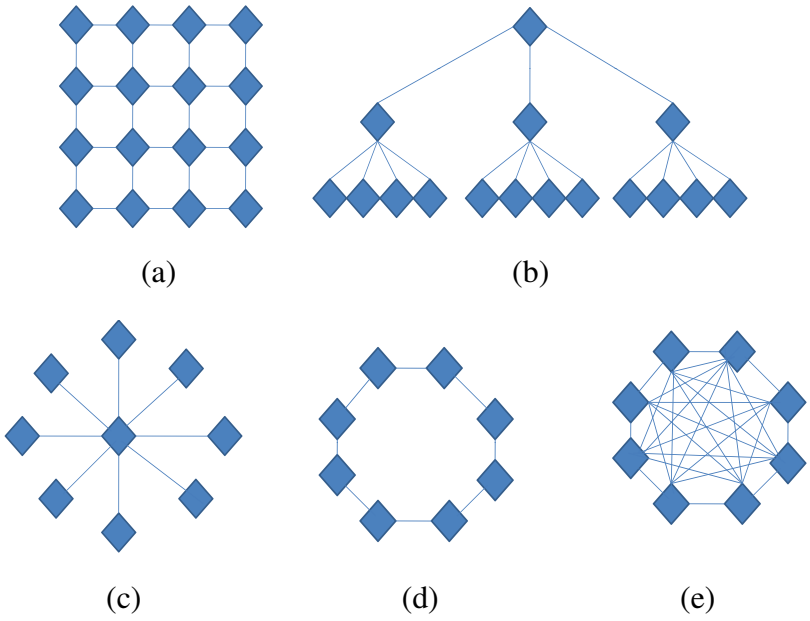

Figure 5.1: Some important topologies used in PSO:(a) The von Neumann topology (VNT), (b) Tree-based graph(TBG), (c) Star graph(SG), (d) Ring topology(RT) and (e) Fully connected graph (FCG)

5.3 PSO for Optimisation of Parameters of Bernoulli Mixture Model

Since the parameters of the Bernoulli mixture model (BMM) are in Euclidean space, we expect that PSO as a global optimisation method in continuous search spaces would be a good candidate to optimise these parameters. In this section, we first introduce the fitness function used in our PSO algorithm and then describe the particle encoding representing the parameters of BMM.

5.3.1 Fitness Function for PSO

We first merge two steps of the EM algorithm in order to develop a proper fitness function to be optimised by PSO. In order to get a proper fitness function, we first substitute the posterior probability $\hat{z}_{ir}$ (see Equation 2.10) in the equation of likelihood under complete knowledge (see Equation 2.6) as follows:

$$l_c(\theta, \pi|y, z) = \sum_{i=1}^{n}\sum_{r=1}^{R}\hat{z}_{ir}\sum_{j=1}^{p}\Big[y_{ij}\log(\theta_{rj}) \\ +(1-y_{ij})\log(1-\theta_{rj})\Big] + \sum_{i=1}^{n}\sum_{r=1}^{R}\hat{z}_{ir}\log\pi_r \tag{5.3}$$

where $\hat{z}_{ir}$ is given by Equation 2.10. Since the parameter values which optimise the likelihood under complete knowledge (Equation 2.6) are the same as the parameters which optimise the log likelihood in Equation 2.5, the maximum log likelihood is computed through substitution of these parameters in Equation 2.5. As described earlier, the parameter π in the complete likelihood should satisfy the constraint $\sum_{r=1}^{R}\pi_r = 1$. Hence, it is necessary to use a Lagrange multiplier [4] to satisfy this constraint. So, let $Q = l_c + \lambda(\sum_{r=1}^{R}\pi_r - 1)$. The partial derivatives of Q with respect to π_r are found and equated to zero as follows:

$$\frac{\partial Q}{\partial \pi_r} = \frac{\partial l_c}{\partial \pi_r} + \lambda = \sum_{i=1}^{n} z_{ir}\frac{1}{\pi_r} + \lambda = 0. \tag{5.4}$$

Based on the constraint, the following equation is obtained:

$$1 = \sum_r \pi_r = \sum_r (-\frac{\sum_{i=1}^n z_{ir}}{\lambda}) = -\frac{\sum_i \sum_r z_{ir}}{\lambda} = -\frac{\sum_i 1}{\lambda} = \frac{-n}{\lambda}. \quad (5.5)$$

Therefore, the function Q which should be optimised by PSO is as follows,

$$Q = l_c - n(\sum_{r=1}^{R} \pi_r - 1) \quad (5.6)$$

5.3.2 Particle Encoding

Since PSO needs a particle encoding to represent the parameters which should be optimized, we develop a very simple encoding to represent all required parameters in the likelihood for Bernoulli mixture models. The parameters are $\theta = (\theta_{11}, ..., \theta_{1p}, \theta_{21}, ..., \theta_{2p}, ..., \theta_{R1}, ...\theta_{Rp})$ and $\pi = (\pi_1, ..., \pi_R)$ with $0 < \theta_{rj} < 1$ and $0 < \pi_r < 1$ where $r = 1, ..., R$ and $j = 1, ..., p$. We should have a vector of parameters like $\psi = (\theta_{11}, ..., \theta_{1p}, \theta_{21}, ..., \theta_{2p}, ..., \theta_{R1}, ...\theta_{Rp}, \pi_1, ..., \pi_R)$ which is represented by each particle.

θ_{11}	...	θ_{1p}	θ_{21}	...	θ_{2p}	...	θ_{R1}	...	θ_{Rp}	π_1	...	π_R

Figure 5.2: Particle encoding for representing the parameters of Bernoulli mixture models

5.3.3 PSO with EM algorithm

Since the EM algorithm is a well-known approach as a local search, finding a hybrid method which can lead to a global search is worthwhile. Hence, PSO as a global search is a good candidate method to mix with the EM algorithm. The Hybrid PSO and EM algorithm can be represented as follows:

Algorithm 2 Hybrid PSO algorithm as a heuristic method to overcome the drawbacks of the EM algorithm

```
Initialize the position and velocity of particle
repeat
    for all particle P in population do
        Calculate fitness value of P
        Update its personal best if fitness value is better than best fitness
        value pbest, and set current position as pbest
    end for
    for all particle P in population do
        Find local best particle from neighbourhood
        Compute particle velocity (equation (5.2))
        Update particle position (equation (5.1))
        Initialize EM algorithm with the position of current particle
        Improve the position of current particle through applying EM algorithm
    end for
until stopping criterion
```

5.4 Experiment Design

To assess the starting points for the EM algorithm that are found by the Random starts, K-means and Split off method, and compare the performance the proposed method, we consider 3 challenging datasets. The first one is from [15] which is the presence/absence measure of the 25 most abundant plant species ($n = 25$) on 17 plots ($p = 17$) from a grazed meadow in Steneryd Nature Reserve in Sweden. The second data come from the R package "bipartite"which can be downloaded from [10]. It is a plant-pollinator network from Memmott (1999). The rows are insect species ($n = 79$) and the columns are plant species ($p = 25$). A one is recorded if that insect species was recorded pollinating the flowers of that plant species. It started as a count data matrix (i.e. the number of times that type of insect was seen pollinating that type of flower), but this is the binary version. The third one is from the book [19] which is nine rodent species ($p = 9$) in 28 urban canyons ($n = 28$) in Southern California. These are challenging and substantial binary datasets which are representative of a range of matrix sizes.

5.5 Results and Discussion

We first applied Random starts, K-means, Split off and Hybrid PSO in dataset 1, 2 and 3. Since these methods are indeterministic, we run these algorithms 30 times for each dataset when we have three to six row clusters and then calculate 95% confidence intervals for the maximum likelihood, AIC and $AICc$ for each dataset. Biologists are often interested to have three to six row groups. Hence, we compare the Hybrid PSO with K-means, random starts and divisive methods when we have three to six clusters. Tables 5.1, 5.2 and 5.3 demonstrate the logarithm of the maximum likelihood, P-value based on student's t-test to compare the maximum log likelihood resulting from the proposed method with other methods, the number of parameters in the Bernouli mixture models, AIC, $AICc$ and the number of clusters (column R) for datasets 1, 2 and 3. We arranged a student's t-test to compare the performance of the proposed method with other methods. As

Method	Max.ll	P-value	npar	AIC	$AICc$	R
Random starts	-182.20 $\pm$ 0.00	2.24E-63	53	470.41 $\pm$ 0.00	486.46 $\pm$ 0.00	3
K-means	-182.66 $\pm$ 0.00	1.25E-67	53	471.32 $\pm$ 0.00	487.38 $\pm$ 0.00	3
Split off	-193.14 $\pm$ 10.83	0.03	53	492.28 $\pm$ 43.33	508.34 $\pm$ 43.33	3
Hybrid PSO	-181.06 $\pm$ 0.00	–	53	**468.12 $\pm$ 0.01**	484.18 $\pm$ 0.01	3
Random starts	-160.65 $\pm$ 0.07	3.1E-10	71	463.31 $\pm$ 0.29	493.17 $\pm$ 0.29	4
K-means	-162.21 $\pm$ 0.00	–	71	466.43 $\pm$ 0.00	496.29 $\pm$ 0.00	4
Split off	-178.94 $\pm$ 10.14	0.0011	71	499.89 $\pm$ 40.56	529.75 $\pm$ 40.57	4
Hybrid PSO	-160.30 $\pm$ 0.00	–	71	**462.60 $\pm$ 0.00**	492.46 $\pm$ 0.00	4
Random starts	-146.17 $\pm$ 0.10	7.1E-06	89	470.34 $\pm$ 0.42	519.38 $\pm$ 0.42	5
K-means	-145.90 $\pm$ 0.00	1.35E-06	89	469.81 $\pm$ 0.03	518.85 $\pm$ 0.03	5
Split off	-167.96 $\pm$ 9.26	6.27E-05	89	513.93 $\pm$ 37.06	562.97 $\pm$ 37.06	5
Hybrid PSO	-145.87 $\pm$ 0.00	–	89	**469.75 $\pm$ 0.00**	518.79 $\pm$ 0.00	5
Random starts	-133.49 $\pm$ 0.27	0.90	107	480.09 $\pm$ 1.09	555.49 $\pm$ 1.09	6
K-means	-133.02 $\pm$ 0.11	0.91	107	480.04 $\pm$ 0.46	554.55 $\pm$ 0.46	6
Split off	-158.98 $\pm$ 8.56	0.19	107	531.96 $\pm$ 34.25	606.46 $\pm$ 34.25	6
Hybrid PSO	-130.93 $\pm$ 0.00	–	107	**472.00 $\pm$ 0.00**	544.87 $\pm$ 0.00	6

Table 5.1: Comparison of Hybrid PSO method with the traditional methods when we have three to six row clusters in dataset 1.

can be seen in Table 5.1 and 5.5, the maximum likelihood (ML), AIC and $AICc$ for the model with three, four and five row clusters, which are found by the Hybrid PSO method, are statistically better than those values found by the EM algorithm utilising other methods for initialising starting points. For the model with six row clusters, our experiments show that the model found by our proposed method is statistically equivalent with the models found by other methods for dataset one.

The results for dataset 2 can be seen in Table 5.2 and also 5.5. In this dataset, the comparison of ML, AIC and $AICc$ for the models with three, four and five row clusters found by the different methods show that the performance of hybrid method is better than the old methods. Although, for six row clusters, the performance of the Hybrid PSO method is better than Split off method, its performance is equivalent with Random starts and K-means.

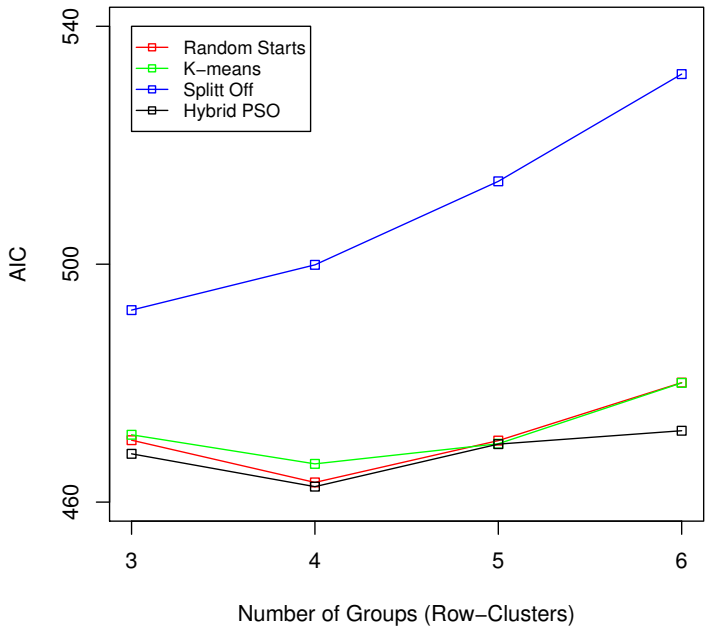

Figure 5.3: Comparison of AIC for different methods in dataset 1.

Method	Max.ll	P-value	npar	AIC	$AICc$	R
Random starts	-578.67 $\pm$ 2.26	3.16E-13	77	1311.35 $\pm$ 9.07	1317.85 $\pm$ 9.07	3
K-means	-582.42 $\pm$ 0.10	5.12E-50	77	1318.84 $\pm$ 59.94	1325.34 $\pm$ 59.94	3
Split off	-586.08 $\pm$ 0.00	1.1E-56	77	1326.15 $\pm$ 0.00	1332.65 $\pm$ 0.00	3
Hybrid PSO	-564.21 $\pm$ 0.10	–	77	**1282.42 $\pm$ 0.42**	1288.92 $\pm$ 0.42	3
Random starts	-564.02 $\pm$ 0.45	6E-09	103	1282.04 $\pm$ 1.81	1288.54 $\pm$ 1.81	4
K-means	-545.84 $\pm$ 3.05	0.9739	103	1279.67 $\pm$ 12.21	1309.35 $\pm$ 12.21	4
Split off	-572.86 $\pm$ 0.00	7.35E-13	103	1351.72 $\pm$ 0.00	1363.40 $\pm$ 0.00	4
Hybrid PSO	-545.75 $\pm$ 4.39	–	103	**1245.50 $\pm$ 17.57**	1309.18 $\pm$ 17.57	4
Random starts	-564.28 $\pm$ 0.21	4.55E-13	129	1386.56 $\pm$ 0.85	1289.06 $\pm$ 0.85	5
K-means	-528.56 $\pm$ 4.03	0.3818	129	1315.12 $\pm$ 16.14	1333.59 $\pm$ 16.14	5
Split off	-557.15 $\pm$ 0.00	5.3E-11	129	1372.30 $\pm$ 0.00	1390.77 $\pm$ 0.00	5
Hybrid PSO	-525.21 $\pm$ 6.20	–	129	**1308.42 $\pm$ 24.82**	1326.89 $\pm$ 24.82	5
Random starts	-505.77 $\pm$ 0.12	0.9462	155	1321.54 $\pm$ 0.49	1289.30 $\pm$ 0.49	6
K-means	-505.77 $\pm$ 2.80	0.9482	155	1321.53 $\pm$ 11.18	1348.48 $\pm$ 11.18	6
Split off	-542.40 $\pm$ 0.00	6.59E-08	155	1394.81 $\pm$ 0.00	1421.75 $\pm$ 0.00	6
Hybrid PSO	-505.42 $\pm$10.09	–	155	**1320.84 $\pm$ 38.40**	1347.49 $\pm$ 44.18	6

Table 5.2: Comparison of Hybrid PSO method with the traditional methods when we have three to six row clusters in dataset 2.

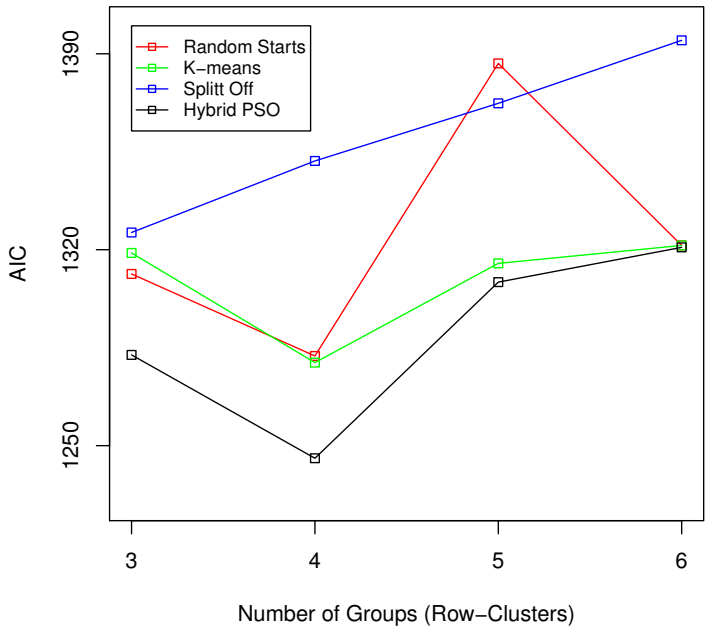

Figure 5.4: Comparison of AIC for different methods in dataset 2.

Table 5.3 and 5.5 show the results for dataset 3. In this dataset, for three row clusters, the performance of the Hybrid method is better than Random starts and Split off methods, however it is equivalent to the K-means method. The model found by Hybrid PSO in dataset 3 with four row clusters, has a better ML, AIC and $AICc$ than K-means and Split off methods and this model is approximately equivalent to Random starts. The performance of Random starts and K-means in dataset 3 with five and six row clusters, are the same as proposed method, but proposed method works better than Split off method.

Method	Max.ll	P-value	npar	AIC	$AICc$	R
Random starts	-87.8420 $\pm$ 0.00	–	29	233.6840 $\pm$ 0.00	242.1000 $\pm$ 0.00	3
K-means	-85.0119 $\pm$ 0.08	0.9983	29	228.0230 $\pm$ 0.35	236.4390 $\pm$ 0.35	3
Split off	-87.4620 $\pm$ 0.00	–	29	232.9230 $\pm$ 0.00	241.3400 $\pm$ 0.00	3
Hybrid PSO	-84.9210 $\pm$ 0.00	–	29	**227.8410 $\pm$ 0.00**	236.2570 $\pm$ 0.00	3
Random starts	-78.5822 $\pm$ 0.19	0.7039	39	235.1637 $\pm$ 4.81	250.7087 $\pm$ 4.81	4
K-means	-79.5077 $\pm$ 0.52	0.0010	39	237.0149 $\pm$ 2.10	252.5599 $\pm$ 2.10	4
Split off	-82.9460 $\pm$ 0.00	–	39	243.8930 $\pm$ 0.00	2259.4380$\pm$ 0.00	4
Hybrid PSO	-78.5450 $\pm$ 0.00	–	39	**235.0890 $\pm$ 0.00**	250.6340 $\pm$ 0.00	4
Random starts	-74.9245 $\pm$ 0.06	0.3981	49	247.8491 $\pm$ 0.24	273.2227 $\pm$ 0.02	5
K-means	-74.9465 $\pm$ 1.17	0.4084	49	247.8933 $\pm$ 4.71	273.2663 $\pm$ 4.71	5
Split off	-79.0220 $\pm$ 0.00	0.012	49	256.0440 $\pm$ 0.00	281.4170 $\pm$ 0.00	5
Hybrid PSO	-72.9772 $\pm$ 4.45	–	49	**243.9555 $\pm$ 1.76**	269.3285 $\pm$ 1.76	5
Random starts	-71.2060 $\pm$ 4.77	0.9185	59	260.4122 $\pm$ 1.40	298.7371 $\pm$ 0.07	6
K-means	-71.3434 $\pm$ 0.01	0.1231	59	260.6861 $\pm$ 0.06	299.0111 $\pm$ 0.06	6
Split off	-74.7390 $\pm$ 0.00	1.21E-28	59	267.4790 $\pm$ 0.00	305.8030 $\pm$ 0.00	6
Hybrid PSO	-71.4571 $\pm$ 0.14	–	59	**260.2467 $\pm$ 2.01**	298.1400 $\pm$ 8.02	6

Table 5.3: Comparison of Hybrid PSO method with the traditional methods when we have three to six row clusters in dataset 3.

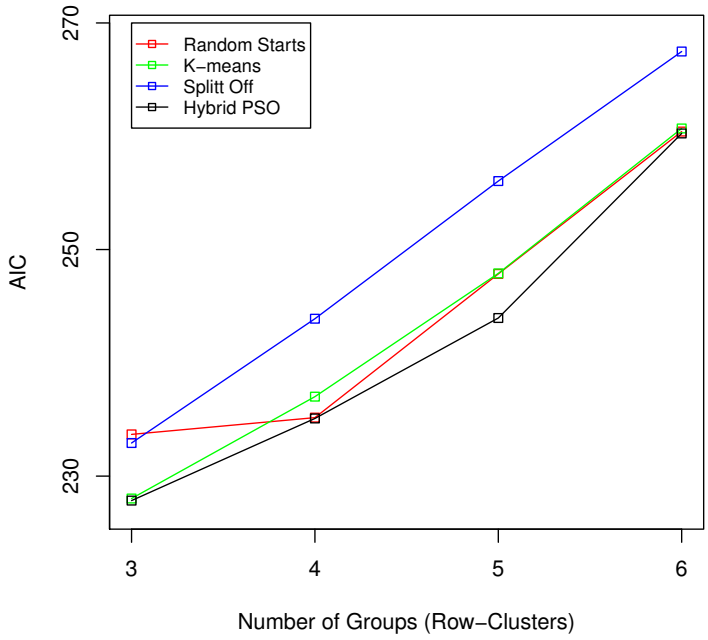

Figure 5.5: Comparison of AIC for different methods in dataset 3.

5.6 Summary

This chapter presented a new Hybrid PSO algorithm with a new fitness function and a new encoding scheme for Bernoulli mixture models which makes the EM algorithm independent of the initial points. Different methods (Random starts, K-means, Split off and Hybrid PSO methods) for choosing the best starting points for the EM algorithm were evaluated and a new heuristic algorithm which combines PSO and the EM algorithm was introduced. Since the EM algorithm is applied as a local search and its result is highly dependent on initial points, a Hybrid PSO approach could improve the EM algorithm to be independent of initial points. The results showed that in most cases, the performance of the Hybrid PSO is statistically better than the other methods and in a few cases, it is equivalent to the other methods. Our result showed that the proposed method performs better than other methods in 23 cases out of 36 and in the remaining cases as well as other methods. Since PSO has been used as a clustering method in literature, as a future work, the performance of using PSO as a clustering method can be investigated to choose the initialization points in EM algorithm.

Chapter 6

Conclusions

The main goal of this thesis was to improve the performance of the EM algorithm as an optimisation method in terms of the Bernoulli mixture models. In the first part of this research, we strived to reduce the sensitivity of the EM algorithm to the initialisation points through clustering techniques. In the second part, we combined the PSO algorithm as a global search method in order to be more likely to avoid the EM algorithm trapping in a local optimum.

In the first part, we evaluated different methods (random starts, K-means, divisive and combined methods) for choosing the best starting points for the EM algorithm and we developed an innovative way which combines K-means and divisive methods. The results showed that the random starts method has the lower *AIC* and *AICc* than the K-means and divisive methods but its run time is more than the K-means and less than the other methods. The divisive methods take a long time to run in comparison with the random starts and K-means methods and also their *AIC* and *AICc* are more than the other methods' *AIC* and *AICc*. So, the divisive methods are not a good choice for starting points. The combined method had the best *AIC* and *AICc* in comparison with the other methods, although its execution time was longer than the other methods. Our experiments showed that this method gives a substantial advantage over the random starts methods used in many optimisations over a multimodal likelihood surface in high dimensions. On the grounds that the new proposed method has the best result from a statistical

point of view, a simulation study would now be desirable for further evaluation of these starting point methods on more data sets.

In the second part, we presented a new hybrid PSO algorithm with a new fitness function and a new encoding scheme for the optimisation of parameters of Bernoulli mixture models which makes the EM algorithm independent of the initial points. We compared the new hybrid method with the EM algorithm methods which were initialised by random starts, K-means, split off and divisive methods. The results showed that in most cases, the performance of the hybrid PSO algorithm is statistically better than, and in a few cases, it is equivalent to the other methods. Our results also showed that the proposed method performs better than other methods in 23 cases out of 36 and in the remaining cases, it is equivalent to the other methods. Since PSO has been used as a clustering method in the literature, as future work, the performance of using PSO as a clustering method can be investigated to choose the initialization points in the EM algorithm.

Appendix A

Source Code of Algorithms in R Language

Listing A.1: All R Codes developed in this project

```
#The following R codes were written by Pledger S. & Frouzesh F. .


# Functions for binary data.


#------------------- LOGIT FUNCTION ---------------------------------

# Function to take logit of a vector. If a value in the vector is
# really close to 0 or 1, set logit to some max or min.

logit <- function(pvec)
   {
   # Replace values if too near 0 or 1:
   pvec <- apply(cbind(pvec,rep(0.00001,length(pvec))),1,max)
   pvec <- apply(cbind(pvec,rep(0.99999,length(pvec))),1,min)
   # Output vector:
   log(pvec/(1-pvec))
   }

#------------------- EXPIT FUNCTION ---------------------------------

# Inverse of logit function, acts on a vector.

expit <- function(lvec)
```

```
    1/(1+exp(-lvec))


# ------------ MODEL FITTING FUNCTIONS ------------------------------


# ------------ rR1cC1B ----------------------------------------------

# Null model, same Bernoulli probability for each cell in the matrix.

rR1cC1B.fn <- function()
    {
    # Estimation:
    theta <- sum(y.mat)/n/p
    max.ll <- sum(y.mat)*log(theta) + (n*p - sum(y.mat))*log(1-theta)
    # Save results:
    res.dev <- -2*max.ll
    npar <- 1
    aic  <- res.dev + 2*npar
    aicc <- aic + (2*(npar+1)*(npar+2))/(n*p - npar - 2)
    out1 <- round(c(n,p,max.ll,res.dev,npar,aic,aicc),3)
    names(out1) <- c("n","p","Max.ll","Res.Dev","npar","AIC","AICc")
    list("info"=out1,"theta"=round(theta,3))
    }

# ------------ rRcpB ------------------------------------------------

# Model with rows clustered, all columns different.

# Version 1: random starts.

rRcpB2.fn <- function(RG=2,iterations=50,nstarts=30)
    {
    # Estimation using various random starts:


    for (thisstart in 1:nstarts)
        {
        pi.v    <- rep(1/RG,RG)
        pp.m    <- matrix(pi.v,n,RG,byrow=T)  # Posterior probs
        theta.m <- matrix(
            runif(RG*p,theta0*0.5,0.5+0.5*theta0),RG,p)
        # Run the EM cycle:
        for (iter in 1:iterations)
            {
            # E-step - Update posterior probabilities:
            for (i in 1:n)
```

```
        {
      # Do numerators for row i:
      num.vect <- pi.v
      for (rg in 1:RG) for (j in 1:p)
         num.vect[rg] <- num.vect[rg]*(
            y.mat[i,j]*theta.m[rg,j] +
            (1-y.mat[i,j])*(1-theta.m[rg,j]))
      # Divide by denominator and put into y.mat:
      pp.m[i,] <- num.vect/sum(num.vect)
        }
   pi.v  <- apply(pp.m,2,mean)
   # M-step - Maximise log Lc to update theta estimates:
   theta.num <- t(pp.m) %*% y.mat
     # R*p matrix of sum(over i)(pp[ir]*x[ij])
   theta.denom <- n*matrix(pi.v,RG,p)
   theta.m <- theta.num/theta.denom
   }
# Calculate the full log likelihood:
logl <- 0
for (i in 1:n)
   {
   term <- pi.v
   for (rg in 1:RG)
      term[rg] <- term[rg]*prod(y.mat[i,]*theta.m[rg,] +
                                (1-y.mat[i,])*(1-theta.m[rg,]))
   logl <- logl + log(sum(term))
     }

this.ll <- logl
print(c(thisstart,this.ll))
# If first start, save this as best:
if (thisstart==1)
   {
   best.ll    <- this.ll
   best.theta <- theta.m
   best.pi    <- pi.v
   best.pp    <- pp.m
   }
# Otherwise, replace if better:
if ((thisstart>1)&(this.ll>best.ll))
   {
   best.ll    <- this.ll
   best.theta <- theta.m
   best.pi    <- pi.v
   best.pp    <- pp.m
   }
}
```

```
    # Find cluster grouping:
    clus <- vector("list",RG)
    for (ii in 1:RG) clus[[ii]] <-
       (1:n)[best.pp[,ii]==apply(best.pp,1,max)]
    # Save results:
    res.dev <- -2*best.ll
    npar <- RG*p + (RG-1)
    aic  <- res.dev + 2*npar
    aicc <- aic + (2*(npar+1)*(npar+2))/(n*p - npar - 2)
    out1 <- round(c(n,p,best.ll,res.dev,npar,aic,aicc,RG),3)
    names(out1) <- c("n","p","Max.ll","Res.Dev.","npar","AIC","AICc","R")
    list("info"=out1,
         "pi"=round(best.pi,3),
         "theta"=round(best.theta,3),
         "post.probs"=round(best.pp,3),
         "Row clusters"=clus)
    }



# Single chosen start at ppr.start:

rRcpB2.fn1 <- function(RG=2,iterations=50)
    {
    # Estimation using given start:
    pp.m <- ppr.start
    pi.v  <- apply(pp.m,2,mean)
    theta.num <- t(pp.m) %*% y.mat
    theta.denom <- n*matrix(pi.v,RG,p)
    theta.m <- theta.num/theta.denom
    # Run the EM cycle:
    for (iter in 1:iterations)
       {
       # E-step - Update posterior probabilities:
       for (i in 1:n)
          {
          # Do numerators for row i:
          num.vect <- pi.v
          for (rg in 1:RG) for (j in 1:p)
             num.vect[rg] <- num.vect[rg]*(
                y.mat[i,j]*theta.m[rg,j] +
                (1-y.mat[i,j])*(1-theta.m[rg,j]))
          # Divide by denominator and put into y.mat:
          pp.m[i,] <- num.vect/sum(num.vect)
          }
       pi.v  <- apply(pp.m,2,mean)
       # M-step - Maximise log Lc to update theta estimates:
```

```
        theta.num <- t(pp.m) %*% y.mat
        # R*p matrix of sum(over i)(pp[ir]*x[ij])
        theta.denom <- n*matrix(pi.v,RG,p)
        theta.m <- theta.num/theta.denom
        }
    # Calculate the full log likelihood:
    logl <- 0
    for (i in 1:n)
        {
        term <- pi.v
        for (rg in 1:RG)
            term[rg] <- term[rg]*prod(y.mat[i,]*theta.m[rg,] +
                                      (1-y.mat[i,])*(1-theta.m[rg,]))
        logl <- logl + log(sum(term))
        }
    this.ll <- logl
    # Find cluster grouping:
    clus <- vector("list",RG)
    for (ii in 1:RG) clus[[ii]] <-
        (1:n)[pp.m[,ii]==apply(pp.m,1,max)]
    # Save results:
    res.dev <- -2*this.ll
    npar <- RG*p + (RG-1)
    aic  <- res.dev + 2*npar
    aicc <- aic + (2*(npar+1)*(npar+2))/(n*p - npar - 2)
    out1 <- round(c(n,p,this.ll,res.dev,npar,aic,aicc,RG),3)
    names(out1) <- c("n","p","Max.ll","Res.Dev.","npar","AIC","AICc","R")
    list("info"=out1,
         "pi"=round(pi.v,3),
         "theta"=round(theta.m,3),
         "post.probs"=round(pp.m,3),
         "Row clusters"=clus)
    }

# Version with a start from kmeans clustering:

rRcpB2km.fn1 <- function(RG=3,iterations=50)
    {
    # Use kmeans clustering to find a starting allocation:
    temp <- kmeans(y.mat,RG,nstart=10)$cluster
    pp.m <- matrix(0,n,RG)
    for (rg in 1:RG) for (i in 1:n)
    if (temp[i]==rg) pp.m[i,rg] <- 1
    # Find starting parameter estimates (as in M step):
    pi.v        <- apply(pp.m,2,mean)
    theta.num   <- t(pp.m) %*% y.mat
    theta.denom <- n*matrix(pi.v,RG,p)
```

```
theta.m     <- theta.num/theta.denom
# Run the EM cycle:
for (iter in 1:iterations)
   {
   # E-step - Update posterior probabilities:
   for (i in 1:n)
      {
      # Do numerators for row i:
      num.vect <- pi.v
      for (rg in 1:RG) for (j in 1:p)
         num.vect[rg] <- num.vect[rg]*(
            y.mat[i,j]*theta.m[rg,j] +
            (1-y.mat[i,j])*(1-theta.m[rg,j]))
      # Divide by denominator and put into y.mat:
      pp.m[i,] <- num.vect/sum(num.vect)
      }
   pi.v  <- apply(pp.m,2,mean)
   # M-step - Maximise log Lc to update theta estimates:
   theta.num <- t(pp.m) %*% y.mat
   # R*p matrix of sum(over i)(pp[ir]*x[ij])
   theta.denom <- n*matrix(pi.v,RG,p)
   theta.m <- theta.num/theta.denom
   }
# Calculate the full log likelihood:
logl <- 0
for (i in 1:n)
   {
   term <- pi.v
   for (rg in 1:RG)
      term[rg] <- term[rg]*prod(y.mat[i,]*theta.m[rg,] +
                                (1-y.mat[i,])*(1-theta.m[rg,]))
   logl <- logl + log(sum(term))
   }
this.ll <- logl
# Find cluster grouping:
clus <- vector("list",RG)
for (ii in 1:RG) clus[[ii]] <-
   (1:n)[pp.m[,ii]==apply(pp.m,1,max)]
# Save results:
res.dev <- -2*this.ll
npar <- RG*p + (RG-1)
aic  <- res.dev + 2*npar
aicc <- aic + (2*(npar+1)*(npar+2))/(n*p - npar - 2)
out1 <- round(c(n,p,this.ll,res.dev,npar,aic,aicc,RG),3)
names(out1) <- c("n","p","Max.ll","Res.Dev.","npar","AIC","AICc","R")
list("info"=out1,
     "pi"=round(pi.v,3),
```

```
        "theta"=round(theta.m,3),
        "post.probs"=round(pp.m,3),
        "Row clusters"=clus)
    }


*****************************************************************************
# runB2.R

# Run the Steneryd data set.

# B = binary data

# Run models using functions from   funB.txt

# Need to have defined
# y.mat, the input matrix of binary data.


# Read in the Liphook Forest toadstool data:

source("readdata.R")

y.mat <- stenerydB.mat        # 25 by 17


# ---------- Setting up ----------------------------------

n      <- nrow(y.mat)
p      <- ncol(y.mat)
theta0 <- sum(y.mat)/n/p  # Overall proportion of 1's.


# ---------- Read in the Functions -----------------------

source("funB2.R")


# -------------------------------------------------------
#                    Fit the Basic Models
# -------------------------------------------------------


# -------------------------------------------------------
#                Null model, (rR1,cC1)
# -------------------------------------------------------
```

```
rR1cC1B.out <- rR1cC1B.fn()    # Stores rR1cC1B.out

print(rR1cC1B.out)


# ------------------------------------------------------------
#                  Saturated model, (rn,cp)
# ------------------------------------------------------------

rncpB.out <- c(n=n,p=p,Max.ll=0,Res.Dev=0,npar=n*p,AIC=2*n*p,
    AICc=NA)

print(rncpB.out)



# ------------------------------------------------------------
#      Model rR.cp  Rows clustered, p column groups
# ------------------------------------------------------------

# Default, two row groups, 10 random starts.

# Two row groups, random starts.
# ---------------------------------


rR2cpB.out <- rRcpB2.fn(nstart=30)

print(rR2cpB.out)


# Three row groups
# ------------------

# Method 1: random starts.

elapsed.vect <- proc.time()[3]

rR3cpB.out <- rRcpB2.fn(RG=3)  # Default is 10 random starts

elapsed.vect <- c(elapsed.vect,proc.time()[3])

print(diff(elapsed.vect))


print(rR3cpB.out)
```

```
# Method 2: single start from k-means clustering:
# ------------------------------------------------

rR3cpBkm.out <- rRcpB2km.fn1(RG=3)

print(rR3cpBkm.out)

# Method 3: Start from two-group output, split off one row
# --------------------------------------------------------

# For each row in turn, provided it is not the sole member
# of its cluster, start a new cluster with this row.


pp.temp <- rR2cpB.out$post.probs

for (bb in 1:n)
   {
   ppr.start <- cbind(pp.temp,rep(0,n))
   ppr.start[bb,] <- c(0,0,1)
   temp.out <- rRcpB2.fn1(RG=3)
   if (bb==1) best.out <- temp.out
   if (temp.out[[1]][3] > best.out[[1]][3])
      best.out <- temp.out
   print(c(bb,temp.out[[1]][3]))
   }

rR2cpBsplit.out <- best.out

print(rR2cpBsplit.out)

*******************************************************************************
# Evaluation of random starts when the number of iteration is increasing

maxlikelihood<-rep(0,50)
aiter<-rep(0,50)
iter=50
for(i in 1:50)
{
        rR3cpB.out<-rRcpB2.fn(RG=3,iterations=iter,nstarts=10)
        maxlikelihood[i]<-rR3cpB.out$info[3]
        aiter[i]=iter
        iter<-iter+10
}
plot(aiter,maxlikelihood)
```

```
# Finding the most maxlikelihood
max(maxlikelihood)
#-180.871

# Finding the probability of the highest maxlikelihood in 20 trial
Number<-0
for(i in 0:20)
{       rR3cpB.out<-rRcpB2.fn(RG=3,iterations=220,nstarts=10)
        maxlikelihood<-rR3cpB.out$info[3]
        if(maxlikelihood>= -180.871)
        {
        Number=Number+1
        }
}
 print(Number)
# 3
# Hence, the probability of the highest maxlikelihood is 3/20 in 20 trial or
# according to the Geometric distributon p=1/x=1/220 such that x= no. of the
# trials to the first succes.

# Evaluation of random starts when the number of random starts is increasing
maxlikelihood<-rep(0,20)
aiter<-rep(0,20)
number=10
for(i in 1:20)
{
        rR3cpB.out<-rRcpB2.fn(RG=3,nstarts=number)
        maxlikelihood[i]<-rR3cpB.out$info[3]
        aiter[i]=iter
        number<-number+10
}
plot(aiter,maxlikelihood,xlab="nstart")

*****************************************************************************************
#Evaluation of random starts when the number of random starts are increasing and
#run it 1000 times each time and finding a probability of reaching global maximum.
maxlikelihood<-0
aiter<-rep(0,20)
Ys<-rep(0,20)
number=10
p1=0

mm = 0

#load("RandomStartOutput.txt")

#mm = mm + 1
```

```
for(i in mm:20)
{
        max = -1000
        c <- 0
        for(d in 1:1000)
        {
                rR3cpB.out<-rRcpB2.fn(RG=3, nstarts=number)
                maxlikelihood<-rR3cpB.out$info[3]

                if(max<maxlikelihood)
        {
                        max=maxlikelihood
                        c=0
                }
                if(max==maxlikelihood)
                {
                        c=c+1
                }
        }

        aiter[i]=number
        p1=c/1000.0
        Y=1-(1-p1)^number
        Ys[i]=Y
        mm = i

        save( mm, aiter, Ys, file="RandomStartOutput.txt" )

        number=number+10
}


# Ploting the confidence band for probability of finding global max.

plot.new()

load( "RandomStartOutput.txt" )

success.fn<-function(x)
{
        success.fn<-1-(1-0.15)^x
}

# I made the color of curve white.
curve(success.fn, 10, 40, col="white",xlab="No.starts",
ylab="Probability␣of␣finding␣global␣max")
```

```
# Probability of finding global max
points(aiter[1:4],Ys[1:4],col="red")
lines (aiter[1:4],Ys[1:4],col="red")
abline(v=30, h=0.95)

#To find the confidence band
YsU <- Ys + 1.96 * sqrt( Ys * ( 1 - Ys ) / 1000 )

YsD <- Ys - 1.96 * sqrt( Ys * ( 1 - Ys ) / 1000 )
# Make upper bound
points(aiter[1:4],YsU[1:4],col="green")
lines (aiter[1:4],YsU[1:4],col="green")


# Make lower band
points(aiter[1:4],YsD[1:4],col="blue")
lines (aiter[1:4],YsD[1:4],col="blue")

legend(x=10,y=0.4, c("Upper␣band", "Probability␣of␣finding␣global␣max",
 "lower␣band"), fill=c("green", "red","blue"))

*****************************************************************************
# Combined method: k-mean and divisive methods
        y.mat <- yy.mat
        krg <- 2
        erg <- 3
        n <- nrow(y.mat)
        rR3cpBkm.out <- rRcpB2km.fn1(RG=krg)
        Row <- matrix( 0, nrow=erg, ncol= nrow(y.mat) + 1 )
        detectedRow <- krg
        for( r in 1:krg )
        {
                Row[ r,1] <- length(rR3cpBkm.out$Row[[r]])
                for( c in 1:length(rR3cpBkm.out$Row[[r]]) )
                        Row[ r,c+1 ] <- rR3cpBkm.out$Row[[r]][ c ]
        }
        post.probs <- rR3cpBkm.out$post.probs
        ppost.probs <- post.probs
        yy.mat <- y.mat
for( rg in krg:(erg-1) )
{
        for( r in 1:rg)
        {
                if( Row[ r,1 ] == 1 )
                        continue
                y.mat <- matrix( 0,nrow=Row[r,1] , ncol=ncol(yy.mat) )
                pp.temp <- rep(1, Row[r,1] )
```

```
for( i in 1:Row[r,1] )
{
        for( j in 1:ncol(yy.mat) )
        {
                y.mat[ i, j ] <- yy.mat[ Row[r, i+1 ], j ]
        }
}
n      <- nrow(y.mat)
p      <- ncol(y.mat)
theta0 <- sum(y.mat)/n/p
for (bb in 1:n)
{
        ppr.start <- cbind(pp.temp,rep(0,n))
        ppr.start[bb,] <- c(0,1)
        temp.out <- rRcpB2.fn1(RG=2)
        if (bb==1) best.out <- temp.out

        if (temp.out[[1]][3] > best.out[[1]][3])
        best.out <- temp.out
        print(c(bb,temp.out[[1]][3]))
}
ppr.start <- matrix( 0, nrow=nrow(yy.mat), ncol=rg+1 )
for( rr in 1:rg)
{
        if( rr == r )
        {
                for( c in 1:length( best.out$Row[[1]] ) )
                {
                        ppr.start[ Row[ r,best.out$Row[[1]][c]
                                        +1],r] <- 1
                }
                for( c in 1:length( best.out$Row[[2]] ) )
                {
                        ppr.start[ Row[ r,best.out$Row[[2]][c]
                                       +1],rg+1] <- 1
                }
        }

        for( c in 1:Row[ rr,1 ] )
        {
                ppr.start[ Row[ rr,c+1], rr ] <- 1
        }
}
y.mat <- yy.mat
n      <- nrow(y.mat)
p      <- ncol(y.mat)
theta0 <- sum(y.mat)/n/p
```

```
                temp.out <- rRcpB2.fn(RG=rg+1,iterations=50,nstarts=30)
                if (r==1) bestt.out <- temp.out
                if (temp.out[[1]][3] > bestt.out[[1]][3])
                        bestt.out <- temp.out
                }
        Row <- matrix( 0, nrow=rg+1, ncol= nrow(y.mat) + 1 )
        for( i in 1:(rg+1) )
        {
                Row[ i,1] <- length(bestt.out$Row[[i]])
                for( c in 1:length(bestt.out$Row[[i]]) )
                        Row[ i,c+1 ] <- bestt.out$Row[[i]][ c ]
        }
}
print(bestt.out)
```

```
#The result of combined method when the # of random stars is 10(nstart=10)

save(bestt.out, file="combined2to4firstrun10.data")
load("combined2to4firstrun10.data")
bestt.out

save(bestt.out, file="combined2to4secondrun10.data")
load("combined2to4secondrun10.data")
bestt.out

save(bestt.out, file="combined2to4thirdrun10.data")
load("combined2to4thirdrun10.data")
bestt.out
```

```
#The result of combined method when the # of random stars is 20(nstart=20)
save(bestt.out, file="combined2to4firstrun20.data")
load("combined2to4firstrun20.data")
bestt.out

save(bestt.out, file="combined2to4secondrun20.data")
load("combined2to4secondrun20.data")
bestt.out
```

```
#The result of combined method when the # of random stars is 30(nstart=30)
save(bestt.out, file="combined2to5.data")
load("combined2to5.data")
bestt.out

save(bestt.out, file="combined3to5.data")
load("combined3to5.data")
bestt.out
```

```
save(bestt.out, file="combined2to6.data")
load("combined2to6.data")
bestt.out

save(bestt.out, file="combined2to4.data")
load("combined2to4.data")
bestt.out

save(bestt.out, file="combined2to4secondrun.data")
load("combined2to4secondrun.data")
bestt.out

save(bestt.out, file="combined2to4thirdrun.data")
load("combined2to4thirdrun.data")
bestt.out

save(bestt.out, file="combined2to4forthrun.data")
load("combined2to4forthrun.data")
bestt.out


save(bestt.out, file="combined2to3.data")
load("combined2to3.data")
bestt.out

save(bestt.out, file="combined2to3secondrun.data")
load("combined2to3secondrun.data")
bestt.out

save(bestt.out, file="combined2to3thirdrun.data")
load("combined2to3thirdrun.data")
bestt.out
```

```
# The result of combined method when we want to have 6 row clusters(erg=6)
#First run
save(bestt.out, file="firstrun.data")
bestt.out=0
load("firstrun.data")
bestt.out

$info
       n          p    Max.ll  Res.Dev.       npar        AIC       AICc          R
  25.000     17.000  -133.545   267.091    107.000    481.091    555.597      6.000

$pi
[1] 0.12 0.08 0.16 0.28 0.20 0.16
```

```
$`Row clusters`
$`Row clusters`[[1]]
[1]  1  4 14

$`Row clusters`[[2]]
[1] 5 6

$`Row clusters`[[3]]
[1]  7  8  9 12

$`Row clusters`[[4]]
[1]  2  3 10 11 18 22 24

$`Row clusters`[[5]]
[1] 13 16 19 20 21

$`Row clusters`[[6]]
[1] 15 17 23 25
```

```
# The result of combined method when we want to have 6 row clusters(erg=6)
#second run
$`Row clusters`
$`Row clusters`[[1]]
[1]  7  8  9 12 20 21

$`Row clusters`[[2]]
[1] 13 16 19

$`Row clusters`[[3]]
[1] 5 6

$`Row clusters`[[4]]
[1]  4 15 17 23 25

$`Row clusters`[[5]]
[1]  1 14

$`Row clusters`[[6]]
[1]  2  3 10 11 18 22 24
```

```
# The result of combined method when we want to have 6 row clusters(erg=6)
#Third run
 save(bestt.out, file="thirdrun.data")
 bestt.out=0
 load("thirdrun.data")
 bestt.out
```

```
$`Row clusters`
$`Row clusters`[[1]]
[1]  7  8  9 12

$`Row clusters`[[2]]
[1] 5 6

$`Row clusters`[[3]]
[1]  2  3 10 11 18 22 24

$`Row clusters`[[4]]
[1] 13 16 19 20 21

$`Row clusters`[[5]]
[1]  4 15 17 23 25

$`Row clusters`[[6]]
[1]  1 14
```

```
#The result of combined method after increasing the number of random starts
#(nstarts=20)
#First run
save(bestt.out, file="firstrunRS20.data")
bestt.out=0
load("firstrunRS20.data")
bestt.out

$info
       n         p    Max.ll  Res.Dev.      npar       AIC      AICc         R
  25.000    17.000  -132.872   265.744   107.000   479.744   554.250     6.000

$pi
[1] 0.28 0.16 0.20 0.08 0.20 0.08

$`Row clusters`
$`Row clusters`[[1]]
[1]  2  3 10 11 18 22 24

$`Row clusters`[[2]]
[1]  7  8  9 12

$`Row clusters`[[3]]
[1]  4 15 17 23 25

$`Row clusters`[[4]]
[1]  1 14
```

```
$`Row clusters `[[5]]
[1] 13 16 19 20 21

$`Row clusters `[[6]]
[1] 5 6
```

```
#The result of combined method after increasing the number of random starts
#(nstarts=20)
#Second run
save(bestt.out, file="secondrunRS20.data")
bestt.out=0
load("secondrunRS20.data")
bestt.out

$`Row clusters `
$`Row clusters `[[1]]
[1]  7  8  9 12

$`Row clusters `[[2]]
[1]  4 15 17 23 25

$`Row clusters `[[3]]
[1]  2  3 10 11 18 22 24

$`Row clusters `[[4]]
[1]  1 14

$`Row clusters `[[5]]
[1] 13 16 19 20 21

$`Row clusters `[[6]]
[1] 5 6
```

```
#The result of combined method after increasing the number of random starts
#(nstarts=20)
#Third run
save(bestt.out, file="thirdrunRS20.data")
bestt.out=0
load("thirdrunRS20.data")
bestt.out

$`Row clusters `
$`Row clusters `[[1]]
[1] 13 16 19 20 21

$`Row clusters `[[2]]
[1]  4 15 17 23 25
```

```
$`Row clusters`[[3]]
[1]  1 14

$`Row clusters`[[4]]
[1] 5 6

$`Row clusters`[[5]]
[1]  7  8  9 12

$`Row clusters`[[6]]
[1]  2  3 10 11 18 22 24
```

```
#The result of combined method after increasing the number of random starts
#(nstarts=20)
#Fourth run
save(bestt.out, file="fourthrunRS20.data")
bestt.out=0
load("fourthrunRS20.data")
bestt.out

$`Row clusters`
$`Row clusters`[[1]]
[1]  7  8  9 12

$`Row clusters`[[2]]
[1]  4 15 17 23 25

$`Row clusters`[[3]]
[1]  1 14

$`Row clusters`[[4]]
[1]  2  3 10 11 18 22 24

$`Row clusters`[[5]]
[1] 13 16 19 20 21

$`Row clusters`[[6]]
[1] 5 6
```

```
#The result of combined method after increasing the number of random starts
#(nstarts=30)
#First run
save(bestt.out, file="firstrunRS30.data")
bestt.out=0
load("firstrunRS30.data")
```

```
bestt.out

$info
       n          p    Max.ll  Res.Dev.       npar        AIC       AICc          R
  25.000     17.000 -132.872   265.744    107.000    479.744    554.250      6.000

$pi
[1] 0.16 0.08 0.08 0.20 0.28 0.20

$`Row clusters`
$`Row clusters`[[1]]
[1]  7  8  9 12

$`Row clusters`[[2]]
[1] 5 6

$`Row clusters`[[3]]
[1]  1 14

$`Row clusters`[[4]]
[1]  4 15 17 23 25

$`Row clusters`[[5]]
[1]  2  3 10 11 18 22 24

$`Row clusters`[[6]]
[1] 13 16 19 20 21
```

```
#The result of combined method after increasing the number of random starts
#(nstarts=30)
#Second run
save(bestt.out, file="secondrunRS30.data")
bestt.out=0
load("secondrunRS30.data")
bestt.out
$`Row clusters`
$`Row clusters`[[1]]
[1]  4 15 17 23 25

$`Row clusters`[[2]]
[1] 5 6

$`Row clusters`[[3]]
[1]  1 14

$`Row clusters`[[4]]
[1]  2  3 10 11 18 22 24
```

```
$`Row clusters`[[5]]
[1] 13 16 19 20 21

$`Row clusters`[[6]]
[1]  7  8  9 12
```

```
#The result of combined method after increasing the number of random starts
#(nstarts=30)
#Third run
save(bestt.out, file="thirdrunRS30.data")
bestt.out=0
load("thirdrunRS30.data")
bestt.out

$`Row clusters`
$`Row clusters`[[1]]
[1]  1 14

$`Row clusters`[[2]]
[1] 13 16 19 20 21

$`Row clusters`[[3]]
[1] 5 6

$`Row clusters`[[4]]
[1]  2  3 10 11 18 22 24

$`Row clusters`[[5]]
[1]  4 15 17 23 25

$`Row clusters`[[6]]
[1]  7  8  9 12
```

```
#The result of combined method after increasing the number of random starts
#(nstarts=30)
#Fourth run
save(bestt.out, file="fourthrunRS30.data")
bestt.out=0
load("fourthrunRS30.data")
bestt.out

$`Row clusters`
$`Row clusters`[[1]]
[1] 5 6

$`Row clusters`[[2]]
```

```
[1] 13 16 19 20 21

$`Row clusters `[[3]]
[1]  1  4 14

$`Row clusters `[[4]]
[1]  7  8  9 12

$`Row clusters `[[5]]
[1] 15 17 23 25

$`Row clusters `[[6]]
[1]  2  3 10 11 18 22 24
```

```
#The resualt of combined method when (erg=7)
save(bestt.out, file="erg7.data")
bestt.out=0
load("erg7.data")
bestt.out

$info
      n         p    Max.ll  Res.Dev.      npar       AIC      AICc         R
 25.000    17.000  -125.333   250.665   125.000   500.665   608.061     7.000

$pi
[1] 0.04 0.08 0.16 0.16 0.28 0.08 0.20
```

```
#The resualt of combined method when (erg=8)
save(bestt.out, file="erg8.data")
bestt.out=0
load("erg8.data")
bestt.out

$info
      n         p    Max.ll  Res.Dev.      npar       AIC      AICc         R
 25.000    17.000  -117.321   234.642   143.000   520.642   669.785     8.000

$pi
[1] 0.04 0.04 0.28 0.16 0.08 0.16 0.08 0.16

#There is not any reduction in the AICc value. It is increasing and there is
#no need to go any further.
x=c(6,7,8)
y=c(555.597,608.061,669.785)
plot(x,y,xlab="erg",ylab="AICc")
```

```
*****************************************************************************
# To find the run time for different methods
# Four row clusters(groups)
# ------------------

# Method 1: random starts.

elapsed.vect <- proc.time()[3]

rR3cpB.out <- rRcpB2.fn(RG=4)  # Default is 10 random starts

elapsed.vect <- c(elapsed.vect,proc.time()[3])

print(diff(elapsed.vect))

#elapsed
#23.29

# Method 2: single start from k-means clustering:
# ---------------------------------------------------

elapsed.vect <- proc.time()[3]

rR3cpBkm.out <- rRcpB2km.fn1(RG=4)

elapsed.vect <- c(elapsed.vect,proc.time()[3])
print(rR3cpBkm.out)

#$`Row clusters`[[1]]
#[1]  7  8  9 12 13 16 19 20 21

#$`Row clusters`[[2]]
#[1]  1  4 14 15 17 23 25
#
#$`Row clusters`[[3]]
#[1]  2  3 10 11 18 22 24
#
#$`Row clusters`[[4]]
#[1] 5 6

print(diff(elapsed.vect))
#elapsed
# 2.36

# Method 3: Start from two-group output, split off one row
# ---------------------------------------------------------
```

```
# For each row in turn, provided it is not the sole member
# of its cluster, start a new cluster with this row.

rR2cpB.out <- rRcpB2.fn()
#Finding three row groups

elapsed.vect <- proc.time()[3]
 pp.temp <- rR2cpB.out$post.probs
for (bb in 1:n)
   {
   ppr.start <- cbind(pp.temp,rep(0,n))
   ppr.start[bb,] <- c(0,0,1)
   temp.out <- rRcpB2.fn1(RG=3)
   if (bb==1) best.out <- temp.out
   if (temp.out[[1]][3] > best.out[[1]][3])
      best.out <- temp.out
   print(c(bb,temp.out[[1]][3]))
   }

rR2cpBsplit.out <- best.out
print(rR2cpBsplit.out)

#$`Row clusters`
#$`Row clusters`[[1]]
# [1]  2  3  8  9 10 11 12 13 16 18 19 20 21 22 24
#
#$`Row clusters`[[2]]
#[1]  1  4  7 14 15 17 23 25
#
#$`Row clusters`[[3]]
#[1] 5 6

elapsed.vect <- c(elapsed.vect,proc.time()[3])
print(diff(elapsed.vect))
#elapsed
#45.62

# Finding four row groups

rR2cpB.out<-rR2cpBsplit.out
elapsed.vect <- proc.time()[3]
 pp.temp <- rR2cpB.out$post.probs
for (bb in 1:n)
   {
   ppr.start <- cbind(pp.temp,rep(0,n))
   ppr.start[bb,] <- c(0,0,0,1)
   temp.out <- rRcpB2.fn1(RG=4)
```

```
    if (bb==1) best.out <- temp.out
    if (temp.out[[1]][3] > best.out[[1]][3])
       best.out <- temp.out
    print(c(bb,temp.out[[1]][3]))
    }
rR2cpBsplit.out <- best.out
print(rR2cpBsplit.out)
#$`Row clusters`
#$`Row clusters`[[1]]
# [1]  2  3  8 10 11 13 16 18 19 20 21 22 24
#
#$`Row clusters`[[2]]
#[1]  1  4  7  9 12 14 25
#
#$`Row clusters`[[3]]
#[1] 5 6
#
#$`Row clusters`[[4]]
#$[1] 15 17 23

elapsed.vect <- c(elapsed.vect,proc.time()[3])
print(diff(elapsed.vect))
#elapsed
# 55.53

# Finding five row groups

rR2cpB.out<-rR2cpBsplit.out
elapsed.vect <- proc.time()[3]
 pp.temp <- rR2cpB.out$post.probs
for (bb in 1:n)
    {
    ppr.start <- cbind(pp.temp,rep(0,n))
    ppr.start[bb,] <- c(0,0,0,0,1)
    temp.out <- rRcpB2.fn1(RG=5)
    if (bb==1) best.out <- temp.out
    if (temp.out[[1]][3] > best.out[[1]][3])
       best.out <- temp.out
    print(c(bb,temp.out[[1]][3]))
    }
rR2cpBsplit.out <- best.out
print(rR2cpBsplit.out)
#$`Row clusters`
#$`Row clusters`[[1]]
# [1]  2  3  8 10 11 13 16 18 19 20 21 22 24
#
#$`Row clusters`[[2]]
```

```
#[1]  1  4  7  9 12 14
#
#$`Row clusters`[[3]]
#[1] 5 6
#
#$`Row clusters`[[4]]
#[1] 15 17 23
#
#$`Row clusters`[[5]]
#[1] 25

elapsed.vect <- c(elapsed.vect,proc.time()[3])
print(diff(elapsed.vect))
#elapsed
#  68.95

# Finding six row groups

rR2cpB.out<-rR2cpBsplit.out
elapsed.vect <- proc.time()[3]
 pp.temp <- rR2cpB.out$post.probs
for (bb in 1:n)
   {
   ppr.start <- cbind(pp.temp,rep(0,n))
   ppr.start[bb,] <- c(0,0,0,0,0,1)
   temp.out <- rRcpB2.fn1(RG=6)
   if (bb==1) best.out <- temp.out
   if (temp.out[[1]][3] > best.out[[1]][3])
      best.out <- temp.out
   print(c(bb,temp.out[[1]][3]))
   }
rR2cpBsplit.out <- best.out
print(rR2cpBsplit.out)
#$`Row clusters`
#$`Row clusters`[[1]]
# [1]  2  3  8 10 11 16 18 19 20 21 22 24
#
#$`Row clusters`[[2]]
#[1]  1  4  7  9 12 14
#
#$`Row clusters`[[3]]
#[1] 5 6
#
#$`Row clusters`[[4]]
#[1] 15 17 23
#
#$`Row clusters`[[5]]
```

```
#[1] 25
#
#$`Row clusters `[[6]]
#[1] 13

elapsed.vect <- c(elapsed.vect,proc.time()[3])
print(diff(elapsed.vect))
#elapsed
#  84.59

# Combined method: Combining k-mean and divisive methods
#-------------------------------------------------------------------------------
elapsed.vect <- proc.time()[3]
y.mat <- yy.mat
        krg <- 2
        erg <- 4
        n <- nrow(y.mat)
        rR3cpBkm.out <- rRcpB2km.fn1(RG=krg)
        Row <- matrix( 0, nrow=erg, ncol= nrow(y.mat) + 1 )
        detectedRow <- krg
        for( r in 1:krg )
        {
                Row[ r,1] <- length(rR3cpBkm.out$Row[[r]])
                for( c in 1:length(rR3cpBkm.out$Row[[r]]) )
                        Row[ r,c+1 ] <- rR3cpBkm.out$Row[[r]][ c ]
        }
        post.probs <- rR3cpBkm.out$post.probs
        ppost.probs <- post.probs
        yy.mat <- y.mat
for( rg in krg:(erg-1) )
{
        for( r in 1:rg)
        {
                if( Row[ r,1 ] == 1 )
                        continue
                y.mat <- matrix( 0,nrow=Row[r,1] , ncol=ncol(yy.mat) )
                pp.temp <- rep(1, Row[r,1] )
                for( i in 1:Row[r,1] )
                {
                        for( j in 1:ncol(yy.mat) )
                        {
                                y.mat[ i, j ] <- yy.mat[ Row[r, i+1 ], j ]
                        }
                }
                n       <- nrow(y.mat)
                p       <- ncol(y.mat)
                theta0 <- sum(y.mat)/n/p
```

```
        for (bb in 1:n)
        {
                ppr.start <- cbind(pp.temp,rep(0,n))
                ppr.start[bb,] <- c(0,1)
                temp.out <- rRcpB2.fn1(RG=2)
                if (bb==1) best.out <- temp.out
                if (temp.out[[1]][3] > best.out[[1]][3])
                best.out <- temp.out
                print(c(bb,temp.out[[1]][3]))
        }
        ppr.start <- matrix( 0, nrow=nrow(yy.mat), ncol=rg+1 )
        for( rr in 1:rg)
        {
                if( rr == r )
                {
                        for( c in 1:length( best.out$Row[[1]] ) )
                        {
                                ppr.start[ Row[ r,best.out$Row[[1]][c]
                                        +1],r] <- 1
                        }
                        for( c in 1:length( best.out$Row[[2]] ) )
                        {
                                ppr.start[ Row[ r,best.out$Row[[2]][c]
                                        +1],rg+1] <- 1
                        }
                }
                for( c in 1:Row[ rr,1 ] )
                {
                        ppr.start[ Row[ rr,c+1], rr ] <- 1
                }

        }
        y.mat <- yy.mat
        n       <- nrow(y.mat)
        p       <- ncol(y.mat)
        theta0 <- sum(y.mat)/n/p
        temp.out <- rRcpB2.fn(RG=rg+1,iterations=50,nstarts=10)
        if (r==1) bestt.out <- temp.out
        if (temp.out[[1]][3] > bestt.out[[1]][3])
                bestt.out <- temp.out
        }
Row <- matrix( 0, nrow=rg+1, ncol= nrow(y.mat) + 1 )
for( i in 1:(rg+1) )
{
        Row[ i,1] <- length(bestt.out$Row[[i]])
        for( c in 1:length(bestt.out$Row[[i]]) )
                Row[ i,c+1 ] <- bestt.out$Row[[i]][ c ]
```

```
        }
}

print(bestt.out)
#$`Row clusters`
#$`Row clusters`[[1]]
#[1] 5 6
#
#$`Row clusters`[[2]]
#[1]  2  3 10 11 13 16 18 19 20 21 22 24
#
#$`Row clusters`[[3]]
#[1]  4 15 17 23 25
#
#$`Row clusters`[[4]]
#[1]  1  7  8  9 12 14


elapsed.vect <- c(elapsed.vect,proc.time()[3])

print(diff(elapsed.vect))
#elapsed
#140.83
#-------------------------------------------------------------------------------
#An algorithm to get the parameter Theta and Pi to make a vector and vice versa.
GetThetaAndPi <- function( params )
{
        pi.v    <- rep(0 ,RG )
        theta.m <- matrix(0,RG,p)
        for(i in 1:RG)
        {
                pi.v[i]<-params[RG*p+i]
        for(j in 1:p)
                {
                        theta.m[i,j]<-params[(i-1)*p+j]
                }
        }
        theta.m[ theta.m > 1 ] <- 1
        theta.m[ theta.m < 0 ] <- 0
        if( sum( pi.v ) == 0 )
        {
                print ( "ERROR in GetTheta" )
                pi.v = rep( 1/RG ,RG )
        }
        pi.v <- pi.v / sum( pi.v )
        o <- list( "pi.v" = pi.v, "theta.m" = theta.m )
        return (o)
```

```
}

GetParams <- function( theta.m, pi.v )
{
        params<-rep(0,RG*p+RG)
        for(i in 1:RG)
        {
                params[RG*p+i] <- pi.v[i]
        for(j in 1:p)
                {
                        params[(i-1)*p+j] <- theta.m[i,j]
                }
        }
        return (params)
}
#-------------------------------------------------------------------------------
#PSO algorithm. The pattern for the following PSO R code was taken from
http://www.r-project.org/ website.

psoptim <- function (par, fn, gr = NULL, ..., lower=0, upper=1,
                     control = list(), Hybrid = TRUE, MaxIteration=1000,
                          Stagnation=Inf, SwarmScale=1) {
  fn1 <- function(par) fn(par, ...)/p.fnscale
  mrunif <- function(n,m,lower,upper) {
    return(matrix(runif(n*m,0,1),nrow=n,ncol=m)*(upper-lower)+lower)
  }
  norm <- function(x) sqrt(sum(x*x))
  npar <- length(par)
  lower <- as.double(rep(lower, ,npar))
  upper <- as.double(rep(upper, ,npar))
  con <- list(trace = 0, fnscale = 1, maxit = MaxIteration, maxf = Inf,
              abstol = -Inf, reltol = 0, REPORT = 10,
              s = NA, k = 3, p = NA, w = 1/(2*log(2)),
              c.p = .5+log(2), c.g = .5+log(2), d = NA,
              v.max = NA, rand.order = TRUE, max.restart=Inf,
              maxit.stagnate = Stagnation,
              vectorize=FALSE, hybrid = Hybrid , hybrid.control = NULL)
  nmsC <- names(con)
  con[(namc <- names(control))] <- control
  if (length(noNms <- namc[!namc %in% nmsC]))
    warning("unknown␣names␣in␣control:␣", paste(noNms, collapse = ",␣"))
  ## Argument error checks
  if (any(upper==Inf | lower==-Inf))
    stop("fixed␣bounds␣must␣be␣provided")

  p.trace <- con[["trace"]]>0L # provide output on progress?
  p.fnscale <- con[["fnscale"]] # provide output on progress?
```

```
  p.maxit <- con[["maxit"]] # maximal number of iterations
  p.maxf <- con[["maxf"]] # maximal number of function evaluations
  p.abstol <- con[["abstol"]] # absolute tolerance for convergence
  p.reltol <- con[["reltol"]] # relative minimal tolerance for restarting
  p.report <- as.integer(con[["REPORT"]]) # output every REPORT iterations
# swarm size
  p.s <- ifelse(is.na(con[["s"]]),floor(10+2*sqrt(npar)),con[["s"]])
  p.s <- p.s * SwarmScale
# average % of informants
  p.p <- ifelse(is.na(con[["p"]]),1-(1-1/p.s)^con[["k"]],con[["p"]])
  p.w0 <- con[["w"]] # exploitation constant
  if (length(p.w0)>1) {
    p.w1 <- p.w0[2]
    p.w0 <- p.w0[1]
  } else {
    p.w1 <- p.w0
  }
  p.c.p <- con[["c.p"]] # local exploration constant
  p.c.g <- con[["c.g"]] # global exploration constant
# domain diameter
  p.d <- ifelse(is.na(con[["d"]]),norm(upper-lower),con[["d"]])
  p.vmax <- con[["v.max"]]*p.d # maximal velocity
# process particles in random order?
  p.randorder <- as.logical(con[["rand.order"]])
  p.maxrestart <- con[["max.restart"]] # maximal number of restarts
# maximal number of iterations without improvement
  p.maxstagnate <- con[["maxit.stagnate"]]
  p.vectorize <- as.logical(con[["vectorize"]]) # vectorize?
  p.hybrid <- as.logical(con[["hybrid"]]) # use local BFGS search
  p.hcontrol <- con[["hybrid.control"]] # control parameters for hybrid optim
  if ("fnscale" %in% names(p.hcontrol)){
    p.hcontrol["fnscale"] <- p.hcontrol["fnscale"]*p.fnscale
  }else{
    p.hcontrol["fnscale"] <- p.fnscale
  }
  if (p.trace) {
    message("S=",p.s,",␣K=",con[["k"]],",␣p=",signif(p.p,4),",␣w0=",
            signif(p.w0,4),",␣w1=",
            signif(p.w1,4),",␣c.p=",signif(p.c.p,4),
            ",␣c.g=",signif(p.c.g,4))
    message("v.max=",signif(con[["v.max"]],4),
            ",␣d=",signif(p.d,4),",␣vectorize=",p.vectorize,
            ",␣hybrid=",p.hybrid)
  }
  ## Initialization
  if (p.reltol!=0) p.reltol <- p.reltol*p.d
  if (p.vectorize) {
```

```
  lowerM <- matrix(lower,nrow=npar,ncol=p.s)
  upperM <- matrix(upper,nrow=npar,ncol=p.s)
}
X <- mrunif(npar,p.s,lower,upper)
if (!any(is.na(par)) && all(par>=lower) && all(par<=upper)) X[,1] <- par
V <- (mrunif(npar,p.s,lower,upper)-X)/2
if (!is.na(p.vmax)) { # scale to maximal velocity
  temp <- apply(V,2,norm)
  temp <- pmin.int(temp,p.vmax)/temp
  V <- V%*%diag(temp)
}



f.x <- apply(X,2,fn1) # first evaluations
stats.feval <- p.s
P <- X
f.p <- f.x
P.improved <- rep(FALSE,p.s)
i.best <- which.min(f.p)
error <- f.p[i.best]

init.links <- TRUE
if (p.trace && p.report==1)
  message("It␣1:␣fitness=",signif(error,4))
## Iterations
stats.iter <- 1
stats.restart <- 0
stats.stagnate <- 0

while (stats.iter<p.maxit && stats.feval<p.maxf && error>p.abstol &&
       stats.restart<p.maxrestart && stats.stagnate<p.maxstagnate) {
  print("Iteration")
  print(stats.iter)

  stats.iter <- stats.iter+1
  if (p.p!=1 && init.links) {
    links <- matrix(runif(p.s*p.s,0,1)<=p.p,p.s,p.s)
    diag(links) <- TRUE
  }
  ## The swarm moves
  if (!p.vectorize) {
    if (p.randorder) {
      index <- sample(p.s)
    } else {
      index <- 1:p.s
    }
```

```
    for (i in index) {
      if (p.p==1)
        j <- i.best
      else
        j <- which(links[,i])[which.min(f.p[links[,i]])] # best informant
      temp <- (p.w0+(p.w1-p.w0)*max(stats.iter/p.maxit, stats.feval/p.maxf))
      V[,i] <- temp*V[,i] # exploration tendency
      V[,i] <- V[,i]+runif(npar,0,p.c.p)*(P[,i]-X[,i]) # exploitation
      if (i!=j) V[,i] <- V[,i]+runif(npar,0,p.c.g)*(P[,j]-X[,i])
      if (!is.na(p.vmax)) {
        temp <- norm(V[,i])
        if (temp>p.vmax) V[,i] <- (p.vmax/temp)*V[,i]
      }
      X[,i] <- X[,i]+V[,i]
      ## Check bounds
      temp <- X[,i]<lower
      if (any(temp)) {
        X[temp,i] <- lower[temp]
        V[temp,i] <- 0
      }
      temp <- X[,i]>upper
      if (any(temp)) {
        X[temp,i] <- upper[temp]
        V[temp,i] <- 0
      }
      ## Evaluate function
      if (p.hybrid) {
#       temp <- optim(X[,i],fn,gr,...,method="L-BFGS-B",lower=lower,
#                     upper=upper,control=p.hcontrol)
        temp <- emoptim( X[,i] )
        V[,i] <- V[,i]+temp$par-X[,i] # disregards any v.max imposed
        X[,i] <- temp$par
        f.x[i] <- temp$value
        stats.feval <- stats.feval+as.integer(temp$counts[1])
      } else {
        f.x[i] <- fn1(X[,i])
        stats.feval <- stats.feval+1
      }

      if (f.x[i]<f.p[i]) { # improvement
        P[,i] <- X[,i]
        f.p[i] <- f.x[i]
        if (f.p[i]<f.p[i.best]) {
              print( "--------------------")
              print( "NewLike" )
              print( f.p[ i ] )
              print( "LogLike" )
```

```
            print( LogLike( X[,i] ) )
          i.best <- i
        }
      }
      if (stats.feval>=p.maxf) break
    }
  } else {
    if (p.p==1)
      j <- rep(i.best,p.s)
    else # best informant
      j <- sapply(1:p.s,function(i)
                  which(links[,i])[which.min(f.p[links[,i]])])
    temp <- (p.w0+(p.w1-p.w0)*max(stats.iter/p.maxit,stats.feval/p.maxf))
    V <- temp*V # exploration tendency
    V <- V+mrunif(npar,p.s,0,p.c.p)*(P-X) # exploitation
    temp <- j!=(1:p.s)
    V[,temp] <- V[,temp]+mrunif(npar,sum(temp),0,p.c.p)*(P[,j[temp]]-X[,temp])
    if (!is.na(p.vmax)) {
      temp <- apply(V,2,norm)
      temp <- pmin.int(temp,p.vmax)/temp
      V <- V%*%diag(temp)
    }
    X <- X+V
    ## Check bounds
    temp <- X<lowerM
    if (any(temp)) {
      X[temp] <- lowerM[temp]
      V[temp] <- 0
    }
    temp <- X>upperM
    if (any(temp)) {
      X[temp] <- upperM[temp]
      V[temp] <- 0
    }
    ## Evaluate function
    if (p.hybrid) { # not really vectorizing
      for (i in 1:p.s) {
        temp <- optim(X[,i],fn,gr,...,method="L-BFGS-B",lower=lower,
                      upper=upper,control=p.hcontrol)
#       temp <- emoptim( X[,i] )
        V[,i] <- V[,i]+temp$par-X[,i] # disregards any v.max imposed
        X[,i] <- temp$par
        f.x[i] <- temp$value
        stats.feval <- stats.feval+as.integer(temp$counts[1])
      }
    } else {
      f.x <- apply(X,2,fn1)
```

```
      stats.feval <- stats.feval+p.s
    }
    temp <- f.x<f.p
    if (any(temp)) { # improvement
      P[,temp] <- X[,temp]
      f.p[temp] <- f.x[temp]
      i.best <- which.min(f.p)
    }
    if (stats.feval>=p.maxf) break
  }
  if (p.reltol!=0) {
    d <- X-P[,i.best]
    d <- sqrt(max(colSums(d*d)))
    if (d<p.reltol) {
      X <- mrunif(npar,p.s,lower,upper)
      V <- (mrunif(npar,p.s,lower,upper)-X)/2
      if (!is.na(p.vmax)) {
        temp <- apply(V,2,norm)
        temp <- pmin.int(temp,p.vmax)/temp
        V <- V%*%diag(temp)
      }
      stats.restart <- stats.restart+1
      if (p.trace) message("It ",stats.iter,": restarting")
    }
  }
  init.links <- f.p[i.best]==error # if no overall improvement
  stats.stagnate <- ifelse(init.links,stats.stagnate+1,0)
  error <- f.p[i.best]
  if (p.trace && stats.iter%%p.report==0) {
    if (p.reltol!=0)
      message("It ",stats.iter,": fitness=",signif(error,4),
              ", swarm diam.=",signif(d,4))
    else
      message("It ",stats.iter,": fitness=",signif(error,4))
  }
}
if (error<=p.abstol) {
  msg <- "Converged"
  msgcode <- 0
} else if (stats.feval>=p.maxf) {
  msg <- "Maximal number of function evaluations reached"
  msgcode <- 1
} else if (stats.iter>=p.maxit) {
  msg <- "Maximal number of iterations reached"
  msgcode <- 2
} else if (stats.restart>=p.maxrestart) {
  msg <- "Maximal number of restarts reached"
```

```
    msgcode <- 3
  } else {
    msg <- "Maximal number of iterations without improvement reached"
    msgcode <- 4
  }
  if (p.trace) message(msg)
  o <- list(par=P[,i.best],value=f.p[i.best],
            counts=c("function"=stats.feval,"iteration"=stats.iter,
              "restarts"=stats.restart),
            convergence=msgcode,message=msg)
  return(o)
}
#-------------------------------------------------------------------------------
# Hybrid PSO and EM algorithm funcion.

PSOEM<-function(RG=3)
{
        pi.v    <- rep(1/RG ,RG )
        theta.m <- matrix(
        runif(RG*p,theta0*0.5,0.5+0.5*theta0),RG,p)
        params<- GetParams( theta.m, pi.v )
        # OptimisedParams <- optim(params,NewLike,method = "L-BFGS-B",lower = 0,
         upper = 1, hessian = T )
        #OptimisedParams <- emoptim(params,iterations=50 )
        OptimisedParams <- psoptim(params,NewLike,gr = NULL,lower = 0,
        upper = 1,Hybrid=TRUE ,MaxIteration=4,Stagnation=2,SwarmScale=3)
        #complete likelihood
        OptimisedParams$value
        #converting params to theta and pi.
        params<-OptimisedParams$par
        print( LogLike(params) )
        ThetaAndPi <- GetThetaAndPi( params )
        pi.v <- ThetaAndPi$pi.v
      theta.m <- ThetaAndPi$theta.m
        # Save results:
        res.dev <- -2*LogLike(params)
        npar <- RG*p + (RG-1)
        aic  <- res.dev + 2*npar
        aicc <- aic + (2*(npar+1)*(npar+2))/(n*p - npar - 2)
        out1 <- round(c(n,p,LogLike(params),res.dev,npar,aic,aicc,RG),3)
        names(out1) <- c("n","p","Max.ll","Res.Dev.","npar","AIC","AICc","R")
        list("info"=out1,
                "pi"=round(pi.v,3),
                "theta"=round(theta.m,3))
}
#-------------------------------------------------------------------------------
# Run PSO with EM algorithm.
```

```
source("readdata3.R")
# ---------- Setting up ----------

y.mat <- stenerydB.mat        # 25 by 17
n     <- nrow(y.mat)
p     <- ncol(y.mat)
theta0 <- sum(y.mat)/n/p  # Overall proportion of 1's.



source("ThetaAndPiAndParamsConversion.R" )
source("LogLike.R" )
source("NewLike.R")
source("EMOptim.R")
source("pso.R")
source("PSOEM.R")

elapsed.vect <- proc.time()[3]

PSOEM.out<-PSOEM(RG)

elapsed.vect <- c(elapsed.vect,proc.time()[3])
print(diff(elapsed.vect))
```

Bibliography

[1] M. R. AlRashidi and M. E. El-Hawary. A survey of particle swarm optimization applications in electric power systems. *IEEE,Transactions on Evolutionary Computation*, 13(4):913–918, 2009.

[2] D. Anderson. *Model Based Inference in the Life Sciences*. Springer, 2008.

[3] C. Ari and S. Aksoy. Maximum likelihood estimation of gaussian mixture models using particle swarm optimization. In *20th International Conference on Pattern Recognition (ICPR)*, pages 746 –749, 2010.

[4] D. P. Bertsekas and D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

[5] A. Bessadok, P. Hansen, and A. Rebai. EM algorithm and variable neighborhood search for fitting finite mixture model parameters. In *Proceedings of International Conference on Computer Science and Information Technology*, pages 725–733, 2009.

[6] D. Böhning. *Computer-assisted analysis of mixtures and applications: meta-analysis, disease mapping and others*. 1999.

[7] K. Burnham and D. Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer, 2nd edition, July 2002.

[8] P. Deb. Finite mixture models. Summer north american stata users' group, Stata Users Group, 2008.

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

[10] C. F. Dormann, B. Gruber, and J. Frund. Introducing the bipartite package: Analysing ecological networks. *R News*, 8(2):8–11, 2008. URL `www.nceas.ucsb.edu/interactionweb/html/memmott_1999.htm`.

[11] E. W. Forgy. Cluster analysis of multivariate data: efficiency vs interpretability of classifications. *Biometrics*, 21:768–769, 1965.

[12] G. Hammerly and C. Elkan. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the 11th international conference on information and knowledge management*, 2002.

[13] N. Laird. Nonparametric maximum likelihood estimation of a mixing distribution. *Journal of the American Statistical Association*, 73(364):pp. 805–811, 1978.

[14] Y. Leung, J.-S. Zhang, and Z.-B. Xu. Clustering by scale-space filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12): 1396–1410, 2000.

[15] B. F. J. Manly. *Multivariate Statistical Methods: a Primer*. Boca Raton, 2005,1st edition.

[16] G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics. Wiley-Interscience, 1 edition, October 2000.

[17] R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm: Simpler, may be better. In *IEEE Transactions on Evolutionary Computation 8*, pages 204–210, 2004.

[18] F. Picard. An introduction to mixture models. Research report, Statistics for Systems Biology Group, 2007. URL `http:`

//pbil.univ-lyon1.fr/members/fpicard/franckpicard_fichiers/pdf/SSB-RR-7.mixture-tutorial.pdf.

[19] G. P. Quinn and M. J. Keough. *Experimental Design and Data Analysis for Biologists*. Cambridge University Press, 2002.

[20] M. Reyes-sierra and C. A. C. Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.

[21] R. Saeidi, H. Mohammadi, T. Ganchev, and R. Rodman. Particle swarm optimization for sorted adapted gaussian mixture models. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(2):344–353, 2009.

[22] N. Sara, A. Rawan, and V. Gregory. A modified fuzzy k-means clustering using expectation maximization. In *Proceedings of IEEE World Congress on Computational Intelligence*, 2006.

[23] G. A. F. Seber. *Multivariate Distributions*, pages 17–58. John Wiley and Sons, Inc., 2008.

[24] W. Seidel, K. Mosler, and M. Alker. A cautionary note on likelihood ratio tests in mixture models. *Annals of the Institute of Statistical Mathematics*, 52:481–487, 2000.

[25] P. Steiner and M. Hudec. Classification of large data sets with mixture models via sufficient em. *Computational Statistics and Data Analysis*, 51(11): 5416 – 5428, 2007. ISSN 0167-9473. Advances in Mixture Models.

[26] K. Takeuchi. Distribution of informational statistics and a criterion of model ŕtting. *Suri-Kagaku (Mathematical Sciences)*, 1976.

[27] W. A. Woodward, W. C. Parr, W. R. Schucany, and H. Lindsey. A comparison of minimum distance and maximum likelihood estimation of a mixture proportion. *Journal of the American Statistical Association*, 79(387): pp. 590–598, 1984.