International Journal of Development Mathematics Vol 1 Issue 2, Page No. 179 - 190

INTERNATIONAL JOURNAL OF DEVELOPMENT MATHEMATICS

ISSN: 3026-8656 (Print) | 3026-8699 (Online)

journal homepage: https://ijdm.org.ng/index.php/Journals

On Expectation-Maximization Algorithm with Split and Merge in R

Hassan Ahmed[a,c*], Ahmed Abdulkadir[b], Kazeem E. Lasisi[b], and Rasheed Bello[b]

[a]*Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi, Bauchi State, Nigeria.*

[b]*Department of Mathematical Sciences, Gombe State University, Gombe, Gombe State, Nigeria.*

[c]*Department of Statistics, Modibbo Adama University, Yola.Adamawa State, Nigeria.*

ARTICLE INFO

Article history:
Received 10 December 2023
Received in revised form 16 April 2024
Accepted 10 May 2024

Keywords:
Algorithm, EM, GMM, Mixture-model, Split and merge

MSC 2020 Subject classification:
62H30, 62H10, 62P10

ABSTRACT

This paper presents a comprehensive study on the implementation of the Expectation-Maximization (EM) algorithm for a 4-component bivariate Gaussian Mixture Model (GMM) with a focus on incorporating the split and merge techniques. The bivariate GMM is widely utilized in various fields, such as pattern recognition, image processing, and data clustering, due to its flexibility in capturing complex data distributions. Our proposed implementation leverages the versatility of the R programming language to create a robust and efficient framework for modeling and optimizing the parameters of the 4-component GMM. The EM algorithm is employed as a powerful tool for iteratively estimating the model parameters, ensuring convergence to a local maximum of the likelihood function. To enhance the model's flexibility, we introduce the split and merge strategies, enabling the algorithm to adapt to diverse data structures and efficiently manage the complexity of the mixture components. The split operation allows for the subdivision of components when needed, while the merge operation facilitates the combination of components that represent similar patterns. This adaptive approach contributes to the model's ability to capture intricate data patterns and improve convergence during the optimization process. The implementation is validated through extensive experimentation on synthetic, demonstrating its effectiveness in accurately estimating the parameters of the 4-component bivariate GMM. The proposed methodology proves to be a valuable addition to the existing tools available for GMM based modeling, providing researchers and practitioners with a flexible and powerful framework for analyzing complex data structures in diverse applications.

1. Introduction

The increasing prominence of finite mixture models in statistical data analysis is evident through a growing number of articles addressing both theoretical and practical aspects of these models in scientific literature. This surge in interest is attributed to the widespread adoption of finite mixtures of distributions as computationally efficient representations for modelling complex data distributions related to random phenomena. Mixture models have proven successful across diverse fields, including agriculture, astronomy, bioinformatics, biology, economics, engineering, genetics, imaging, marketing, medicine, neuroscience, psychiatry, psychology, and various other disciplines spanning biological, physical, and social sciences (McLachlan *et al*., 2019).

Finite mixture models are commonly employed to study data from populations suspected to consist of homogeneous subpopulations. For instance, when a disease has a simple genetic cause, the population may be divided into two or three homogeneous groups. In the initial stages of these investigations, having a sensitive test for the number of subpopulations (denoted as k) included in the data is crucial. However, constructing such a test is often more challenging than expected due to finite mixture models belonging to a class of non-regular models, leading to the inapplicability of many classical asymptotic results.

Finite mixtures of distributions, particularly normal distributions, have been extensively used as models in practical situations where data arises from two or more populations mixed in varying proportions (Kayal *et al*., 2023). Recent applications include a spatially constrained skew Student's-t mixture model for brain MR image segmentation and bias field correction (Cheng *et al*., 2022). Normal mixture models also play a role in developing robust estimators,

* Corresponding author. Tel.: +2348032417537
E-mail address: hassanahmed.official@gmail.com (Hassan A.)

https://doi.org/10.62054/ijdm/0102.14

addressing challenges such as sensitivity to outliers under mixtures with light-tailed distributions like the normal distribution (Sugasawa *et al*., 2022).

Cluster analysis, especially in the field of mixture likelihood approach, witnesses a growing use of mixtures of distributions, normal or otherwise. The approach assumes that observations on entities to be clustered are from a mixture of a specified number of populations or groups in varying proportions. By adopting a parametric form for the density function in each group, a likelihood is formed, and unknown parameters are estimated through this likelihood. Probabilistic clustering is then obtained based on estimated posterior probabilities of group membership, and entities can be assigned to groups by allocating them to the group with the highest estimated posterior probability of belonging (Ganesalingam and McLachlan, 1979).

While decomposing a finite mixture of distributions is a challenging problem, the advent of high-speed computers has shifted attention to likelihood estimation of parameters in mixture distributions. The Expectation-Maximization (EM) algorithm has become a widely applicable approach for iterative computation of maximum likelihood estimates in incomplete data problems, where traditional methods may be more complicated (Dempster *et al*., 1977). The EM algorithm, with its Expectation step (E-step) and Maximization step (M-step), has proven valuable in various problems involving incomplete data, offering an intuitive and natural approach to estimation even before its formalization in the seminal DLR paper (Dempster *et al*., 1977).

The Expectation–Maximization (EM) algorithm and its applications span from theoretical studies on convergence, such as the analysis of EM and its variant deterministic annealing EM (DA-EM)(Yu and Yang, 2018), to diverse modifications tailored for specific purposes, including image matching (Ma *et al*., 2019), parameter estimation (Liu *et al.,* 2019; Du and Gui, 2019), malaria diagnoses (Pag`es-Zamora *et al*., 2019), mixture simplification (Yu and Chan, 2018), and audio-visual scene analysis (Gebru *et al.*, 2016).

The surge in EM's popularity is largely attributed to its effectiveness in estimating parameters of mixture models (MM) (McLachlan, 2000; McLachlan and Krishnan, 2007). Mixture models refer to probabilistic models where the modeled population consists of several sub-populations. Each sub-population is assumed to follow a simple parametric distribution, referred to as the component of the MM. The type of the MM is identified by the distribution family of its components, with Gaussian Mixture Models (GMM) being a notable example where components follow a Gaussian distribution. Once estimated, MM parameters find applications in density estimation (Yu *et al.*, 2018; B¨acklin *et al*, 2018) and clustering tasks (Pag`es-Zamora *et al.,* 2019; Yang *et al*., 2012; Celeux and Govaert, 1995). In the context of MM parameter estimation, the EM algorithm serves as a clustering algorithm that maximizes the missing data log-likelihood function, acting as a maximum-likelihood estimator. EM possesses favorable properties like monotonic convergence and probabilistic constraints, making it particularly appealing in the context of MM parameter estimation.

However, the EM algorithm comes with documented drawbacks (Baudry and Celeux, 2015). It necessitates initial parameters for the first iteration, known as initialization, which can be challenging for MMs with numerous parameters. Moreover, EM is highly sensitive to initialization due to the multi-modality of the likelihood function for the MM. As a strictly hill climbing algorithm with a monotonic convergence property, EM can get trapped in local optima. Consequently, the emphasis on the initialization of the EM algorithm is significant, leading to extensive research in this area. Effective initialization not only enhances the results of the estimation problem but can also expedite the estimation process. Despite its advantages, the EM algorithm is noted for its slow linear convergence rate (Baudry and Celeux, 2015; Ng and McLachlan, 2004), yet, with good initial parameters close to a saddle point (local optima) of the likelihood function, fewer computationally expensive EM iterations are needed to complete the process.

In their paper (Zhang *et al*., 2004), a novel Competitive EM (CEM) algorithm designed for finite mixture models was introduced aiming to address the primary limitations of the EM algorithm, namely, susceptibility to local maxima entrapment and occasional convergence to the boundary of the parameter space. The proposed algorithm exhibits the ability to autonomously determine the clustering number and efficiently execute "split" or "merge" operations, leveraging a new competitive mechanism. Notably, it demonstrates insensitivity to the initial configuration of the mixture component number and model parameters.

The aim of this paper is to implement the EM algorithm for a multivariate Gaussian mixture model with "split" or "merge" operations using R-platform (R-core Team, 2010).

2. Methods

For convenience, we provide some necessary and useful definitions and some mathematical derivations on finite mixture model, EM algorithm and the Competitive EM algorithm which we need throughout the research work.

2.1 Learning finite mixture models

It is said a d-dimensional random variable $x = [x_1,x_2,....,x_d]^T$ follows a k-component finite mixture distribution, if its probability density function can be written as

$$p(x/\theta) = \sum_{m=1}^{\theta} \pi_m \, p(x/\theta_m) \tag{1}$$

where π_m is the prior probability of the mth component and satisfies

$$\pi_m \geq 0, \sum_{m=1}^{k} \pi_m = 1 \tag{2}$$

where θ_m is the parameter of the mth density model and

$$\theta = \{(\pi_m, \theta_m), m = 1, \dots, k\} \tag{3}$$

is the parameter of the mixture models. For our study we used a 4component 2-dimensional Gaussian mixture model.

2.2 Formulation and Derivations of the EM algorithm.

To estimate the parameters of the finite mixture model defined above, we employ the EM algorithm. Suppose we have a random variable X from a D-dimensional Gaussian mixture model, i.e.,

$$\underline{X} \sim N\left(\underline{X}_j; \underline{\mu}_z, \underline{\Sigma}'_z\right)$$

Then,

$$p(Z, \underline{X}) = p(Z)p(\underline{X}|Z) = cat(Z: \prod 1)N\left(\underline{X}; \underline{\mu}_z, \underline{\Sigma}'_z\right) \tag{5}$$

$$= \prod_{d=0}^{D-1} \prod_{d}^{I(z=d)} \frac{1}{\sqrt{(2\Pi)^k det(\Sigma'_z)}} \exp\left\{\{-\frac{1}{2}\left(\underline{X} - \underline{\mu}_z\right)^{T\underline{\Sigma}'^{-1}_z}\left(\underline{X} - \underline{\mu}_z\right)\right\} \tag{6}$$

2.2.1 E-Step

We apply Baye's rule since the posterior is yet unknown

$$p(Z, \underline{X}) = \frac{p(Z)p(\underline{X}|Z)}{p(\underline{X})} \sim p(Z)p(\underline{X}|Z) = p(Z, \underline{X}) \tag{7}$$

The unnormalized responsibilities is given by,

$$\widetilde{\rho_d^{[i]}} = p(Z = d, \underline{X} = \underline{X}^{[i]}; \underline{\theta}^{[K]}) \tag{8}$$

$$= \prod_d \frac{1}{\sqrt{(2\Pi)^k det(\Sigma_z')}} \exp\left\{-\frac{1}{2}\left(\underline{X}^{[i]} - \underline{\mu}_d\right)^T \underline{\Sigma}_z'^{-1}\left(\underline{X}^{[i]} - \underline{\mu}_d\right)^T\right\} \quad (9)$$

Therefore, normalised responsibility is given by,

$$\rho_d^{[i]} = \frac{\widetilde{\rho_d^{[i]}}}{\sum_{c=o}^{D-1} \widetilde{\rho_c^{[i]}}} \quad (10)$$

2.2.2M-Step

$$\boldsymbol{Q}\left(\underline{\boldsymbol{\theta}}; \boldsymbol{D}; \underline{\boldsymbol{\theta}}^{[k]}\right)$$

$$= \sum_{i=0}^{N-1}\sum_{c=0}^{D-1} \rho_c^{[i]} \log p\left(Z = d, \underline{X} = \underline{X}^{[i]}; \underline{\theta}\right) \quad (11)$$

$$= \sum_{i=0}^{N-1}\sum_{c=0}^{D-1} \rho_c^{[i]} \left(\sum_{d=0}^{D-1} I(d=c) log \underline{\prod}_d - \frac{K}{2}\log 2\Pi - \frac{1}{2}\log\det\left(\underline{\Sigma}_d'\right) - \frac{1}{2}\left(\underline{X}^{[i]} - \underline{\mu}_d\right)^T \underline{\Sigma}_z'^{-1}\left(\underline{X}^{[i]} - \underline{\mu}_d\right)\right)$$

To derive the unconstrained optimization, we build a Lagrangian, that is,

$$\hat{Q}\left(\underline{\theta}, \lambda\right) = Q\left(\underline{\theta}\right) + \lambda(\lambda - \textstyle\sum_{c=o}^{D-1} \pi_c) \quad (12)$$

2.2.3 Maximising with respect to π

$$\frac{\partial \widehat{\boldsymbol{Q}}}{\partial \widehat{\boldsymbol{\Pi}_e}} = \left(\sum_{i=0}^{N-1}\sum_{c=0}^{D-1} \boldsymbol{\rho_c^{[i]}} \left(\frac{\partial}{\partial \widehat{\boldsymbol{\Pi}_e}} \sum_{d=0}^{D-1} \mathrm{I(d=c)} \log \boldsymbol{\Pi_d}\right)\right) - \boldsymbol{\lambda} \frac{\partial}{\partial \widehat{\boldsymbol{\Pi}_e}} \sum_{c=0}^{D-1} \boldsymbol{\Pi_c}$$

$$= \left(\sum_{i=0}^{N-1}\sum_{c=0}^{D-1} \boldsymbol{\rho_c^{[i]}} \sum_{d=0}^{D-1} \mathrm{I(d=c)} \frac{1}{\boldsymbol{\Pi_d}} \boldsymbol{\partial_{de}}\right) - \boldsymbol{\lambda} \sum_{c=0}^{D-1} \boldsymbol{\partial_{ce}}$$

$$= \left(\sum_{i=0}^{N-1}\sum_{c=0}^{D-1} \boldsymbol{\rho_c^{[i]}} \boldsymbol{I(e=c)} \frac{1}{\boldsymbol{\pi_e}}\right) - \boldsymbol{\lambda}$$

$$= \left(\sum_{i=0}^{N-1} \boldsymbol{\rho_e^{[i]}} \frac{1}{\boldsymbol{\pi_e}}\right) - \boldsymbol{\lambda}$$

$$= \frac{1}{\boldsymbol{\pi_e}} \sum_{i=0}^{N-1} \boldsymbol{\rho_e^{[i]}} = \boldsymbol{\lambda}$$

$$= \frac{1}{\boldsymbol{\pi_e}} \boldsymbol{\gamma_e} = \boldsymbol{\lambda}$$

$$\therefore \widehat{\boldsymbol{\pi_e}} = \frac{\boldsymbol{\gamma_e}}{\boldsymbol{\lambda}} \quad (13)$$

NOTE:

1. $\frac{\partial}{\partial \widehat{\pi_e}} \sum_{d=0}^{D-1} \mathrm{I(d=c)} \log \pi_d$

$$= \frac{\partial}{\partial \widehat{\pi_e}} \mathrm{I(d=c)} \log \pi_d = \frac{\partial}{\partial \widehat{\pi_e}} \log \Pi_d$$

$$= \frac{\partial \log \pi_d}{\partial \pi_d} \frac{\partial \pi_d}{\partial \pi_e}$$

$$= \frac{1}{\pi_d} \partial_{de} \quad (14)$$

2. $\frac{\partial}{\partial \widehat{\pi_e}} \pi_c = \partial_{ce}$ (15)

Maximising with respect to λ

$$\frac{\partial \hat{Q}}{\partial \lambda} = \lambda - \sum_{c=0}^{D-1} \pi_c = 0$$

$$\sum_{c=0}^{D-1} \frac{\gamma_c}{\lambda} = 1$$

$$\lambda = \sum_{c=0}^{D-1} \gamma_c = N \tag{16}$$

2.2.4 Maximising with respect to μ

$$\frac{\partial \hat{Q}}{\partial \underline{\mu}_e} = \sum_{i=0}^{N-1} \sum_{c=0}^{D-1} \rho_c^{[i]} \left(\frac{\partial}{\partial \mu_e} \left(-\frac{1}{2} \left(\underline{X}^{[i]} - \underline{\mu}_c \right)^{T \underline{\Sigma}_c'} \left(\underline{X}^{[i]} - \underline{\mu}_c \right) \right) \right) = \underline{0}$$

$$= \sum_{i=0}^{N-1} \sum_{c=0}^{D-1} \rho_c^{[i]} \left(-\underline{\Sigma}_c' \left(\underline{X}^{[i]} - \underline{\mu}_c \right) \right) \partial_{ce} = \underline{0}$$

$$= \sum_{i=0}^{N-1} \rho_e^{[i] \underline{\Sigma}_e'} \left(\underline{X}^{[i]} - \underline{\mu}_e \right) = \underline{0}$$

$$= \sum_{i=0}^{N-1} \rho_e^{[i]} \left(\underline{X}^{[i]} - \underline{\mu}_e \right) = \underline{0}$$

$$= \sum_{i=0}^{N-1} \rho_e^{[i]} = \gamma_{e \underline{\mu}_e}$$

$$\therefore \underline{\mu}_e = \frac{\left(\sum_{i=0}^{N-1} \rho_e^{[i]} \underline{X}^{[i]} \right)}{\gamma_e} \tag{17}$$

2.2.5 Maximising with respect to Σ_e'

$$\frac{\partial \hat{Q}}{\partial \underline{\Sigma}_e'} = \sum_{i=0}^{N-1} \sum_{c=0}^{D-1} \rho_c^{[i]} \left(\frac{\partial}{\partial \underline{\Sigma}_c'} \left(\frac{-1}{2} \log \underline{\Sigma}_c \right) + \frac{\partial}{\partial \underline{\Sigma}_c'} \left(-\frac{1}{2} \left(\underline{X}^{[i]} - \underline{\mu}_c \right)^{T \underline{\Sigma}_c'} \left(\underline{X}^{[i]} - \underline{\mu}_c \right) \right) \right)$$

$$= \sum_{i=0}^{N-1} \sum_{c=0}^{D-1} \rho_c^{[i]} \left(\left(\frac{\partial}{\partial \underline{\Sigma}_c'} \left(\frac{-1}{2} \right) \underline{\Sigma}_c \right) \partial_{ce} + \left(\frac{-1}{2} \right) \underline{\Sigma}_c \left(\underline{X}^{[i]} - \underline{\mu}_c \right) \left(\underline{X}^{[i]} - \underline{\mu}_c \right)^T \frac{\partial}{\partial_{ce}} \right) = \underline{0}$$

$$= \sum_{i=0}^{N-1} \rho_e^{[i]} \left(\underline{\Sigma}_e' \underline{\Sigma}_c' \left(\underline{X}^{[i]} - \underline{\mu}_e \right) \left(\underline{X}^{[i]} - \underline{\mu}_c \right)^T \underline{\Sigma}_e' \right)$$

$$= \left(\sum_{i=0}^{N-1} \rho_e^{[i]} \right) \underline{\Sigma}_e' - \sum_{i=0}^{N-1} \underline{\Sigma}_e' \rho_e^{[i]} \left(\underline{X}^{[i]} - \underline{\mu}_e \right) \left(\underline{X}^{[i]} - \underline{\mu}_e \right)^T \underline{\Sigma}_e' = \underline{0}$$

$$= \underline{\Sigma}_e' \gamma_e - \sum_{i=0}^{N-1} \rho_e^{[i]} \left(\underline{X}^{[i]} - \underline{\mu}_e \right) \left(\underline{X}^{[i]} - \underline{\mu}_e \right)^T = \underline{0}$$

$$\therefore \underline{\Sigma}_e' = \frac{1}{\gamma_e} \sum_{i=0}^{N-1} \rho_e^{[i]} \left(\underline{X}^{[i]} - \underline{\mu}_e \right) \left(\underline{X}^{[i]} - \underline{\mu}_e \right)^T \tag{18}$$

The E and M step is performed iteratively until convergence.

2.3 The Competitive EM(CEM)

When EM encounters local maxima, the components usually overpopulate in some regions, i.e. the model

overfits the data, but under-populate in other regions. The difficulty of passing through some low likelihood regions prevents them from getting to the expected positions. To overcome this problem, CEM do the split or merge operation when EM has converged to a local maximum, thus the components' distribution can self-adapt, and split will take place where the components are too few and merge will take place where the components are too many.

2.3.1 Formulation of Split and Merge

CEM use local Kullback divergence to measure the distance between the local data density $f_m(x)$ and model density $p_m(x)$ of the *mth* component. Define

$$j(m;\theta) = \int f_m(x) \frac{f_m(x)}{p_m(x)}\, dx \tag{19}$$

Split probability of the *mth* component is in direct proportion to $j(m;\theta)$ and the merge probability of the *mth* component is in inverse proportion
$j(m';\theta')$, where m and θ denote, respectively, the new index of merged component and new parameters of mixture models if the *mth* and *lth* components merge.

$$p_{split}(m;\theta) = \frac{j(m;\theta)}{Z(\theta)} \tag{20}$$

$$p_{merge}(m,l;\theta) = \frac{\beta / j(m';\theta')}{Z(\theta)} \tag{21}$$

where

$$Z(\theta) = \sum_{m=1}^{k} p_{split}(m;\theta) + \sum_{m=1}^{k} \sum_{l=m+1}^{k} p_{merge}(m,l;\theta) = 1 \tag{22}$$

where β is a constant.

2.3.2 Operations of Split and Merge

The operation type and the candidates of split or merge components are
sampled by $p_{split}(m;\theta)$ and $p_{merge}(m,l;\theta)$

Split Operation: Suppose that the *mth* component is chosen to split, CEM will generate two new components from the current samples in the *mth* component. We initialize the two new components parameters by random initialization method and optimize them by the basic EM algorithm.

Merge operation: Suppose that the *mth* and *lth* components are selected to merge, the parameters of the merged component can be calculated directly from the original *mth* and *lth* components parameters as follows:

$$\pi_m^{(M)} = \pi_m + \pi_l \tag{23}$$

$$\mu^{(M)} = (\pi_m \mu_m + \pi_l \mu_l)/\pi^{(M)} \tag{24}$$

$$\Sigma_m^{(M)} = \frac{\pi_m \left[\Sigma_m + \left(\mu_m - \mu_m^{(M)}\right)\left(\mu_m - \mu_m^{(M)}\right)^T\right] + \pi_l \left[\Sigma_l + \left(\mu_l - \mu_m^{(M)}\right)\left(\mu_l - \mu_m^{(M)}\right)^T\right]}{\pi_m^{(M)}} \tag{25}$$

2.4 Probability of Acceptance

After the operation type and operation candidates are chosen by sampling according to the split and merge probabilities, the acceptance probability is calculated to prevent poor operation. It is calculated by:

$$P_a = min\left(\exp\left(\frac{L(\theta(t+1),\boldsymbol{X})-L(\theta(t),\boldsymbol{X})}{\gamma}\right),1\right) \quad (26)$$

where γ is a constant.

3. Results and Discussion

In this section, the same synthetic data is used to determine the distribution for experiments. Results of the experiments i.e parameter estimates combined with the randomly selected value for either of the variables is presented and discussed.

3.1 Synthetic data

We use 1000 samples from 4-component GMM. In this GMM, the two components (kernels 1 and 2) share a common mean, but have different covariance matrices. The prior probability of kernel 4 is a little lower than the other kernels. The parameters of GMM are given as follows:

$$\pi_1 = \pi_2 = \pi_3 = 0.3,\ \ \pi_4 = 0.1, \quad (27)$$

$$\mu_1 = \mu_2 = [-4,-4]^T \quad (28)$$

$$\mu_3 = [2,2]^T,\ \ \mu_4 = [-1,-6]^T \quad (29)$$

3.2 Software version and Implementation in R

The models are run in R using R-studio. The used versions for this work are R version 4.3.2 (2023-10-31), Platform: x86 64-apple-darwin23.0.0 (64bit), R-studio version: RStudio/2023.09.1+494 For MAC Operating system 14.2.1.

The package used are mixtools (Benaglia, Chauveau, Hunter, and Young, 2009) version: 2.0.0, mclust (Scrucca, Fraley, Murphy, and Raftery, 2023) version 6.0.1 and gratis (Kang, Hyndman, and Li, 2020) version 1.0.5

When working analyses, make sure to check the version used and always consult the latest version of the package's reference manual/documentation. Updates and bug-fixes may eventually necessitate alterations to the codes provided below.

The r code used in generating the data is given below:

```
#install package if(!require('gratis')) install.packages('gratis') library('gratis')

# Set a seed for reproducibility set.seed(123)

# Generate some sample data with two components means <- matrix(c(-4, -4,-4,-4, 2, 2, -1, -6), ncol =
4, byrow = TRUE)

    sigmas <- array(c(1, 0.5, 0.5, 1,6, -2, -2, 6,2, -1, 1, 2,0.125, 0, 0, 0.125), c(2,2,4)) weights <- c(0.3, 0.3, 0.3, 0.4)

n <- 1000

gmmdata <- rmixnorm(n, means = means, sigma = sigmas,
weight = weights)
```

Plot plot(gmmdata, col = ’blue’, pch = 19, main = ’’) hist(gmmdata, breaks = 100, freq = FALSE)

Figure 1 clearly shows that in the simulated data there is considerable heterogeneity in both components. Note that these data are model-driven and merely used for illustration of the software functionality. Figure 2 shows a histogram of the generated data where we can see presence of mixture.

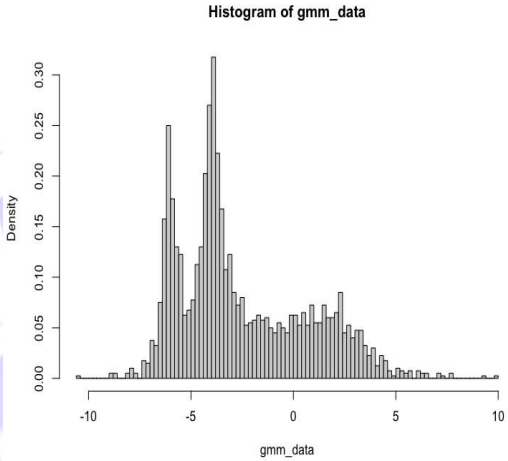

Figure 1: Bivariate two-Component Gaussian Mixture Model

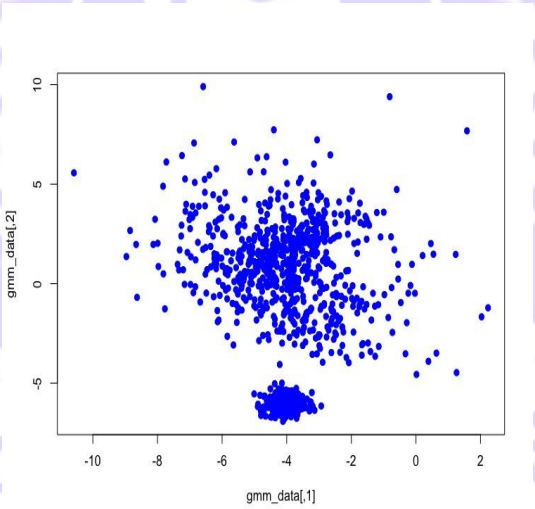

Figure 2: Plot of simulated data showing clustering

4, Results

First let’s see the R code for the general of EM algorithm with ’split’ and ’merge’

initializeparameters = function(data, numcomponents) {
numsamples = nrow(data)
randomindices = sample1:numsamples, numcomponents, re-

```
place = FALSE initialmeans = data[randomndices, ]

    initialcovariances = array(diag(2), dim = c(2, 2, num-
components)) initialweights = rep(1/numcomponents, numcomponents)

    list(means = initialmeans, covariances = initialcovari-
    ances, weights = initialweights)
    }
    computeresponsibilities <- function(data, parameters)
    {
    numcomponents <- length(parameters$weights) responsibilities = matrix(0, nrow(data), numcomponents)

    for ( i in 1:numcomponents) {
    responsibilities[, i] = parameters$weights[i] * dmvnor-
    mdata, mean = parametersmeans[i, ], sigma = parameterscovariances[,, i]
    }
    responsibilities = responsibilities / rowSums(responsibilities)
    return(responsibilities)
    }
    updateparameters = function(data, responsibilities) { numcomponents = ncol(responsibilities) numsamples =

    nrow(data) newweights = colSums(responsibilities / numsamples newmeans = t(responsibilities) * data /

    colSums(responsibilities)

    newcovariances = array(0, dim = c(2, 2, numcomponents)))
    for (i in 1:numcomponents) { diff = data - newmeans[i,
    ]
    newcovariances[,, i] <- t(diff) *(diff * responsibil-
    ities[, i]) / sum(responsibilities[, i]) }
        list(means = newmeans, covariances = newcovariances, weights = newweights)
    }
    emalgorithm = function(data, numcomponents, maxitera-
tions, splitthreshold, mergethreshold) { parameters = initializeparameters(data, numcomponents)
    for (iteration in 1:maxiterations) {
    responsibilities = computeresponsibilities(data, param-
eters) newparameters = updateparameters(data, responsibilities)

    splitoOp = splitOperation(data, iteration); mergeOp = mergerOperation(data, newparameters, respon-
    sibilities);
        acceptanceOp = acceptanceOp() parameters = newparameters
    } return(parameters)
    }
    splitOperation <- function(data, iteration){ #implement
    Split Operation here }
        mergeOperation <- function(data, iteration){ #implement Merge Operation here }
    numcomponents = 4 maxiterations = 100 splitthreshold = splitProb() mergethreshold = mergeProb()
```

result = emalgorithm(data, numcomponents, maxiterations, splitthreshold, mergethreshold) print(result)

The table below shows the results and estimates of the fitted 4 - component mixture model Using Split and Merge.

Table 1: estimates of 2 component Gaussian mixture model for test data.

	Component 1	Component 2	component 3	component 4
Mixing Probability:	0.3027889	0.3343421	0.1917989	0.1710701
Means:				
Variable 1	-4.016259	-4.209846	-3.736834	-3.839645
Variable 2	-6.012891	-0.332322	2.135295	2.320582
Variances:				
	0.117010199	2.570516	0.6657319	5.339230
	-0.003298045	-1.627323	0.4300285	-1.673955
	-0.003298045	-1.627323	0.4300285	-1.673955
	0.113375020	3.056676	1.0806868	6.592091

The Figure 3 shows heat map of the fitted bivariate 4-cmponent GMM. The Figure 3 is plotted using 'mclust' package.

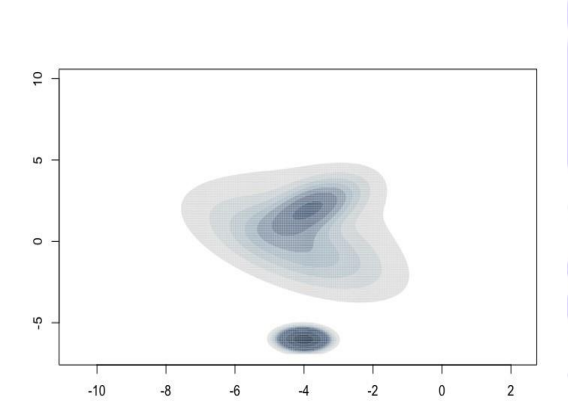

Figure 3: Plot of fitted model.

5. Conclusion

This work showed how the EM algorithm can be implemented for bivariate 4 - component GMM with 'split' and 'merge' in R. Slight modification of the same code can be applied to multivariate cases with more than 2 variables. It can be seen that the split and merge operation helped in overcoming the challenges of EM algorithm and aided the program in reaching global solution.

References

Ba¨cklin, C. L., Andersson, C., and Gustafsson, M. G. (2018). Self-tuning density estimation based on Bayesian averaging of adaptive kernel density estimations yields state-of-the-art performance. *Pattern Recognition*, *78*, 133–143.

Baudry, J.-P., and Celeux, G. (2015). EM for mixtures: Initialization requires special care. *Statistics and computing*, *25*, 713–726.

Benaglia, T., Chauveau, D., Hunter, D. R., and Young, D. (2009). mixtools: An R Package for Analyzing Finite Mixture Models. *Journal of Statistical Software*, *32*(6), 1–29. Retrieved from https://www.jstatsoft.org/v32/i06/

Celeux, G., and Govaert, G. (1995). Gaussian parsimonious clustering models. *Pattern recognition*, *28*(5), 781–793.

Cheng, K. K., Lam, T. H., and Leung, C. C. (2022). Wearing face masks in the community during the COVID-19 pandemic: altruism and solidarity. *The Lancet*, *399*(10336), e39–e40.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society: series B (methodological)*, *39*(1), 1–22.

Du, Y., and Gui, W. (2019). Goodness of fit tests for the log-logistic distribution based on cumulative entropy under progressive type II censoring. *Mathematics*, *7*(4), 361.

Ganesalingam, S., and McLachlan, G. J. (1979). Small sample results for a linear discriminant function estimated from a mixture of normal populations. *Journal of Statistical Computation and Simulation*, *9*(2), 151–158.

Gebru, I. D., Alameda-Pineda, X., Forbes, F., and Horaud, R. (2016). EM algorithms for weighted-data clustering with application to audio-visual scene analysis. *IEEE transactions on pattern analysis and machine intelligence*, *38*(12), 2402–2415.

Kang, Y., Hyndman, R. J., and Li, F. (2020). GRATIS: GeneRAting
TIme Series with diverse and controllable characteristics. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *13*(4), 354–376.

Kayal, S., Bhakta, R., and Balakrishnan, N. (2023). Some results on stochastic comparisons of two finite mixture models with general components. *Stochastic Models*, *39*(2), 363–382.

Liu, C., Li, H.-C., Fu, K., Zhang, F., Datcu, M., and Emery, W. J. (2019). Bayesian estimation of generalized gamma mixture model based on variational em algorithm. *Pattern Recognition*, *87*, 269– 284.

Ma, J., Jiang, X., Jiang, J., and Gao, Y. (2019). Feature-guided Gaussian mixture model for image matching. *Pattern Recognition*, *92*, 231–245.

McLachlan, G. (2000). Peel., d. *Finite Mixture Models*.

McLachlan, G. J., and Krishnan, T. (2007). *The EM algorithm and extensions*. John Wiley and Sons.

McLachlan, G. J., Lee, S. X., and Rathnayake, S. I. (2019). Finite mixture models. *Annual review of statistics and its application*, *6*, 355–378.

Ng, S.-K., and McLachlan, G. J. (2004). Speeding up the EM algorithm for mixture model-based segmentation of

magnetic resonance images. *Pattern Recognition*, *37*(8), 1573–1589.

Pag`es-Zamora, A., Cabrera-Bean, M., and Diaz-Vilor, C. (2019). Unsupervised online clustering and detection algorithms using crowdsourced data for malaria diagnosis. *Pattern Recognition*, *86*, 209–223.

Scrucca, L., Fraley, C., Murphy, T. B., and Raftery, A. E. (2023). *Model-Based Clustering, Classification, and Density Estimation Using mclust in R*. Chapman and Hall/CRC. Retrieved from https://mclust-org.github.io/book/ doi: 10.1201/9781003277965

Sugasawa, S., Kim, J. K., and Morikawa, K. (2022). Semiparametric imputation using latent sparse conditional Gaussian mixtures for multivariate mixed outcomes. *arXiv preprint arXiv:2208.07535*.

Yang, M.-S., Lai, C.-Y., and Lin, C.-Y. (2012). A robust EM clustering algorithm for Gaussian mixture models. *Pattern Recognition*, *45*(11), 3950–3961.

Yu, J., Chaomurilige, C., and Yang, M.-S. (2018). On convergence and parameter selection of the EM and DA-EM algorithms for Gaussian mixtures. *Pattern Recognition*, *77*, 188–203.

Yu, L., Yang, T., and Chan, A. B. (2018). Density-preserving hierarchical EM algorithm: Simplifying Gaussian mixture models for approximate inference. *IEEE transactions on pattern analysis and machine intelligence*, *41*(6), 1323–1337.

Zhang, B., Zhang, C., and Yi, X. (2004). Competitive EM algorithm for finite mixture models. *Pattern recognition*, *37*(1), 131–144.