SAJSET
ISSN: 2992-3581

Savannah Journal of Science and Engineering Technology
Volume: 02; Issue: 03; Pages: 094-102; 2024.
www.sajsetjournal.com.ng

DOI: xx.xxxx.x/sajset-02-2024-0003

SAJSET
ISSN: 2992-3581

A logistic function for Expectation-Maximization Algorithm with split and merge

[1,2]Ahmed, H., [1]Abdulkadir, A., [1]Lasisi, K. E. and [3]Hassan, H.
[1]*Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi, Bauchi State, Nigeria*
[2]*Department of Statistics, Modibbo Adama University, Yola. Adamawa State, Nigeria*
[3]*Department of Mathematics, Nigerian Army College of Education, Illorin, Kwara State, Nigeria*

Paper History
Received: 01st May, 2024
Accepted: 23rd May, 2024
Published: May, 2024

Corresponding author
Ahmed, H.
hassanahmed.official@gmail.com

Abstract:
The Expectation-Maximization (EM) algorithm provides a framework for finding maximum likelihood estimates of parameters in statistical models. But the algorithm has challenges which include converging to local optima, initialization sensitivity, convergence speed, etc. To seek solutions to these challenges, this paper reviews 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 aim of this paper is to implement a logistic function in the split and merge components of the EM algorithm. This extension was then applied to both simulated and real data. The simulated data is generated from the Gaussian mixture while real data is on 553 diabetic patients. The extended algorithm selected a 4component Gaussian mixture with good estimates of the parameter. It also selected a 4-component Gaussian mixture for real data set with 95.1% rate of convergence with 80 iterations. This shows that, the application of the extended algorithm to both datasets indicate an improvement in the algorithm.

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

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 modeling 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 [1].

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 [2]. Recent applications include a spatially constrained skew Student's-t mixture model for brain MR image segmentation and bias field correction [3]. Normal mixture models also play a role in developing robust estimators, addressing challenges such as sensitivity to outliers under mixtures with light-tailed distributions like the normal distribution [4].

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 [5].

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 [6]. 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 [6].

The Expectation–Maximization (EM) algorithm and its applications span from theoretical studies on convergence, such as the analysis of EM and its variant DA-EM[7], to diverse modifications tailored for specific purposes, including image matching [8], parameter estimation [9, 10], malaria diagnoses [11], mixture simplification [12], and audio-visual scene analysis [13].

The surge in EM's popularity is largely attributed to its effectiveness in estimating parameters of mixture models (MM) [14, 15]. 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 [12, 16] and clustering tasks [11, 17, 18]. 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.

In their paper [19], 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 a logistic function in the split and merge components of the EM algorithm.

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 $x = [x_1, x_2, \ldots, x_d]^T$ follows a k-component finite mixture distribution, if its probability density function can be written as

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

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

$$\pi_m \geq 0, \sum_{m=1}^{k} \pi_m = 1 \quad (2)$$

Where θ_m is the parameter of the mth density model and

$$\theta = \{(\pi_m, \theta_m), m = 1, \ldots, k\} \quad (3)$$

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.,

$$\underline{X} \sim N\left(\underline{X}_j; \underline{\mu}_z, \underline{\Sigma}'_z\right) \quad (4)$$

Then,

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

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

2.2.1 E-Step

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

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

The unnormalized responsibilities is given by,

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

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

Therefore, normalised responsibility is given by,

$$\rho_d^{[i]} = \frac{\widetilde{\rho_d^{[i]}}}{\sum_{c=0}^{D-1} \widetilde{\rho_c^{[i]}}} \quad (10)$$

2.2.2 M-Step

The conditional expectation that lead to the equation 10 can be solved as:

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

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

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

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

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

2.2.3 Maximising with respect to π

Solving equation 12 with respect to the mixing probabilities,

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

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

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

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

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

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

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

NOTE:

1. $$\frac{\partial}{\partial \widehat{\pi_e}}\sum_{d=0}^{D-1} I(d=c)\log \pi_d = \frac{\partial}{\partial \widehat{\pi_e}} I(d=c)\log \pi_d = \frac{\partial}{\partial \widehat{\pi_e}}\log \Pi_d$$
$$= \frac{\partial \log \pi_d}{\partial \pi_d}\frac{\partial \pi_d}{\partial \pi_e}$$
$$= \frac{1}{\pi_d}\partial_{de} \quad (14)$$

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

2.2.4 Maximising with respect to λ

Solving equation 12 with respect to the Lagrange built,

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

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

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

2.2.5 Maximising with respect to μ

Solving equation 12 again with respect to the means,

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

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

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

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

$$= \sum_{i=0}^{N-1}\rho_e^{[i]} = \gamma_e\underline{\mu_e}$$

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

2.2.6 Maximising with respect to Σ_e'

Solving equation 12 with respect to the variance-covariance matrix,

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

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

$$= \sum_{i=0}^{N-1}\rho_e^{[i]}\left(\underline{\Sigma_e'}\underline{\Sigma_c'}\left(\underline{X}^{[i]} - \underline{\mu_e}\right)\left(\underline{X}^{[i]} - \underline{\mu_c}\right)^T\underline{\Sigma_e'}\right)$$

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

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

$$\therefore \underline{\Sigma_e'} = \frac{1}{\gamma_e}\sum_{i=0}^{N-1}\rho_e^{[i]}\left(\underline{X}^{[i]} - \underline{\mu_e}\right)\left(\underline{X}^{[i]} - \underline{\mu_e}\right)^T \quad (18)$$

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 over fits 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. This formulation is due to Zhang[23]. Define

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

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' $j(m';\theta'$ to), 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.

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

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

Where

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

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:

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

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

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

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.

The process for computing Split probability and Merge probability as shown above can be completely replaced by the logistic function, which is a standard sigmoid function. The function is given below.

$$p(x) = \frac{1}{1+e^{-x}} \quad (27)$$

Since the logistic function is non-negative and often show a return value (y axis) in the range 0 to 1, for extended CEM, if $p(x) < 0.5$ the algorithm splits and it merges, otherwise.

Equation 26 above calculates the acceptance of the candidates for either operation. But with this extension, candidates are accepted with certainty.

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.

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:

$$\pi_1 = \pi_2 = \pi_3 = 0.3,\ \ \pi_4 = 0.1, \quad (28)$$

$$\mu_1 = \mu_2 = [-4,-4]^T \quad (29)$$

$$\mu_3 = [2,2]^T,\ \ \mu_4 = [-1,-6]^T \quad (30)$$

3.1 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 version: 2.0.0[20], mclust version 6.0.1 [21] and gratis version 1.0.5 [22].

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.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.

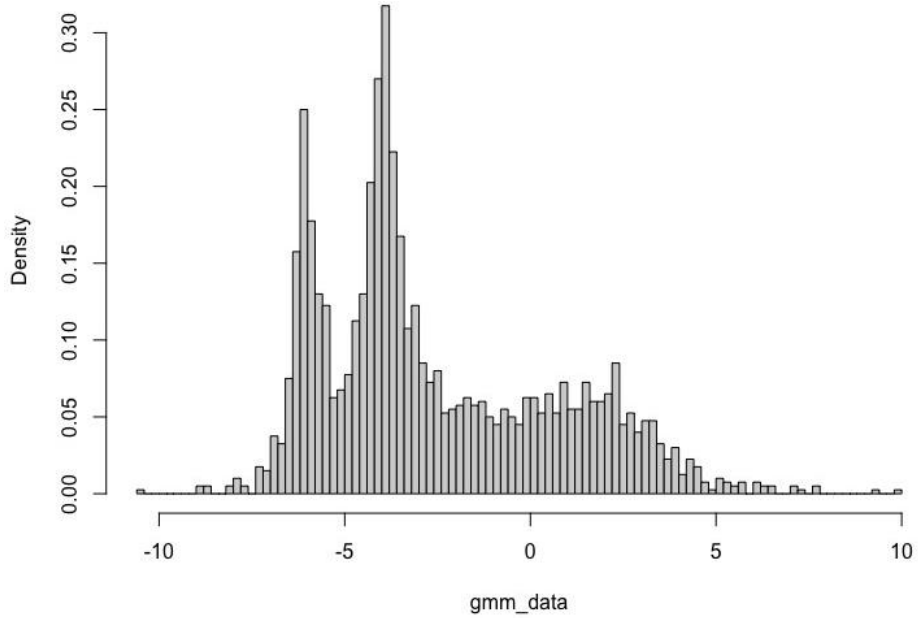

Figure 1: Bivariate two-Component Gaussian Mixture Model

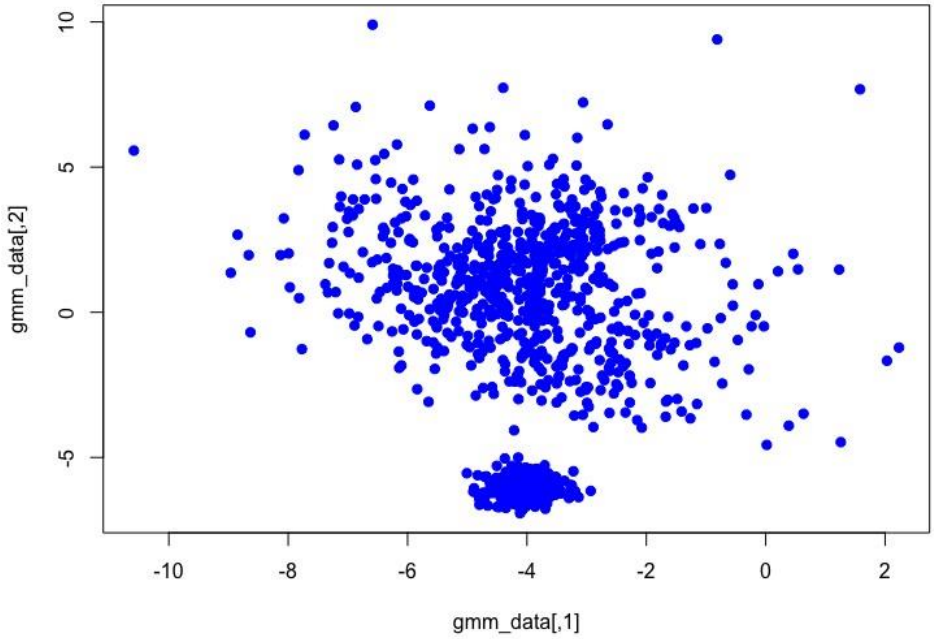

Figure 2: Plot of simulated data showing clustering

4. Synthetic data

Table 1 below shows the results and estimates of the fitted 4 - component mixture model Using Split and Merge.

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

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, replace = FALSE initialmeans = data[randomndices, ]
initialcovariances = array(diag(2), dim = c(2, 2, numcomponents)) initialweights = rep(1/numcomponents,
numcomponents)
list(means = initialmeans, covariances = initialcovariances, 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] * dmvnormdata, 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, maxiterations, splitthreshold, mergethreshold) {
parameters = initializeparameters(data, numcomponents)
for (iteration in 1:maxiterations) {
responsibilities = computeresponsibilities(data, parameters) newparameters = updateparameters(data,
responsibilities)
splitoOp = splitOperation(data, iteration); mergeOp = mergerOperation(data, newparameters, responsibilities);
      acceptanceOp = acceptanceOp() parameters = newparameters
}
return(parameters)
}

splitOperation <- function(data, iteration){ #implementSplit 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)
}
```

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 0.4300285	5.339230
	-0.003298045	-1.627323	0.4300285	-1.673955
	-0.003298045	-1.627323	1.0806868	-1.673955
	0.113375020	3.056676		6.592091

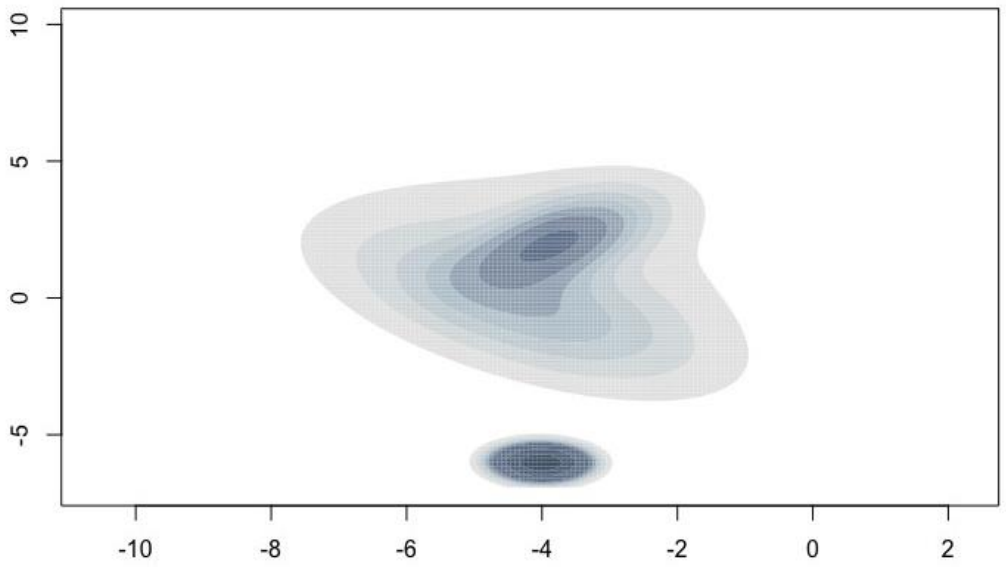

Figure 3: Plot of fitted model

4.1 Real Data

The data used was obtained from Ahmed [21]. The data set consists of 553 cases of diabetes mellitus that were collected at Federal Medical Center, Yola. The variables measured: Age(years), Mass of a patient(kg/meters), glucose level (plasma glucose concentration, a 2-hour in an oral glucose tolerance test), pressure (Diastolic blood pressure mmHg), insulin (2-hour serum insulin mu U/ml) and class variable (0 or 1) treating 0 as false or negative and 1 treated as true or positive test for diabetes. Figure 3 below shows the plot of the variables against the dependent variable class. It clearly shows the presence of mixture in all the variables. Figure 5 shows the histogram of the data.

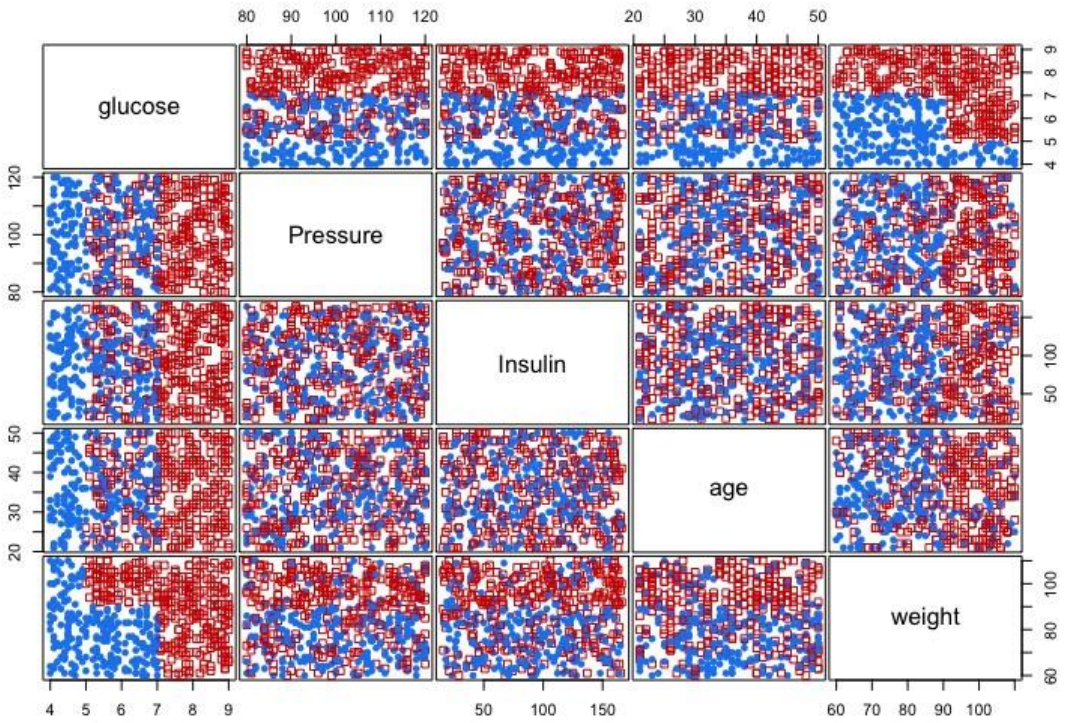

Figure 4: Visualization of data. Dependent variable against all independent variables. Both axes represent the class variable for each diabetic case.

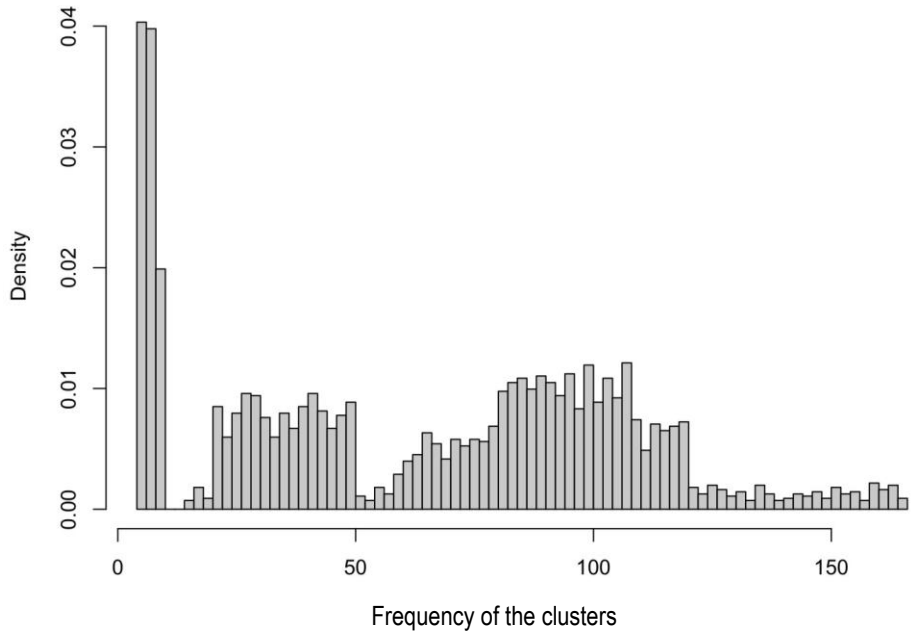

Figure 5: Plot of the data against the density for each cluster.

The updated algorithm is put through real-data sets. The results obtained for estimates of parameters of the selected model is shown in Table 2. The code shown above is slightly modified to have the results in this section.

Table 2: Estimates of 4 component Gaussian mixture model for real data.

Component	**1**	**2**	**3**	**4**
π	0.1234	0.2213	0.44355	0.443322
glucose	5.112	4.8876	5.8876	7.7766
Pressure	201.33	66.87	32.144	5.3343
Insulin	8.4434	50.8877	170.9988	320.778
age	45.3243	43.7788	29.3322	454.333
weight	88.7876	76.6655	90.4433	32.1223
BIC :	-6192.06			
Log-likelihood:	-3214.72			

It can be observed from Table 2 that, the algorithm has chosen 4-component model with a loglikelihood of -3214.72 and BIC of -6192.06.

Table 3: Performance of the algorithm

No of Splits	No. of Merge	Number of Iterations	*Tavg*	P_s(%)
75	22	80	94.6	95.1

The following are observed and recorded:

a. The number of times the program performed split operations,
b. Number of times it performs merge operations,
c. Total number of iterations before convergence,
d. Average number of iterations carried out before the first split or merge, and
e. Percentage of optimal iterations.

After implementing Equation 51, however, Table 3 shows a high performance CEM with 95.1% of all iterations reaching optimal solutions at just 80 iterations. This remarkably improves the convergence speed and the computational load.

5. Conclusion

The aim of this work is to implement a logistic function in the spit and merge operations of the Competitive EM algorithm. This is done with the intention of improving it with a speedier and an accurate algorithm. A logistic function was successfully implemented on the split and merge components. Real and simulated data sets were used to test the algorithm. For the real data set, a 4-component mixture was chosen with 95.1% assurance of most iterations reaching convergence (out of the 80 iterations used). For the simulated data set, a 4-component mixture was also selected and the estimate of parameters are highly accurate due to the estimates of the parameters are close to the actual values. This indicates that a logistic function replacing the local Kullback divergence is highly effective.

This study can be improved by testing other squared distances like the f-divergence, or other distances like the

Mahalanobis distance on the split and merge operations in the CEM.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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- 392.

[10] Pages-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.

[11] 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.

[12] 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.

[13] McLachlan, G., Peel., D., (2000). *Finite Mixture Models,* John Wiley & Son, Hoboken, New Jersey.

[14] McLachlan G. J. and Krishnan, T., (2007). *The EM algorithm and extensions*. John Wiley & Son, Hoboken, New Jersey.

[15] B¨acklin, 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.

[16] 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.

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

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

[19] 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. https://www.jstatsoft.org/v32/i06/

[20] 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. https://mclust-org.github.io/book/

[21] 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.

[22] Ahmed, H., Mohammed, M. B. and Baba, I. A., (2021). On Comparing Multi-Layer Perceptron and Logistic Regression for Classification of Diabetic Patients in Federal Medical Center Yola, Adamawa State, *International Journal of Engineering Technologies and Management research*, 8(6), 24-40.

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