# REM manuscript

author

May 2022

## 1 Algorithm

### 1.1 EM algorithm

Assume that a data set $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$ is sampled from a mixture of $K$ multidimensional Gaussian distribution. The mixture model is provided below

$$f(\boldsymbol{x}_i; \boldsymbol{\tau}, \boldsymbol{\Sigma}, \boldsymbol{\mu}) = \sum_{k=1}^{K} \tau_k \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k) \tag{1}$$

$\tau_k > 0$ denotes the mixing proportions that sums up to 1, and $\mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k)$ denotes the density of the data belonging to the $k$th multivariate Gaussian cluster $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. We also let $\boldsymbol{z} = (z_1, z_2, ..., z_n)$ be the unobserved latent variable where $z_i \in \{1, 2, ..., K\}$. If the $i$th observation belongs to the $k$th cluster then we will set $z_i = k$. The log-likelihood of the complete data $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n, z_1, z_2, ..., z_n)$ can be written as

$$\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\tau}, \boldsymbol{\Sigma}, \boldsymbol{\mu}) = \sum_{i=1}^{n} \sum_{k=1}^{K} \boldsymbol{T}_{ki} \ln\left[\tau_k \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k)\right] \tag{2}$$

where $\boldsymbol{T}_{ki}$ is a indicator that takes value 1 when $z_i = k$. The aim of EM algorithm is to maximize the log-likelihood of the incomplete data with $\boldsymbol{x}$, which is the log-likelihood function of the complete data marginalized over $\boldsymbol{z}$. Since the latent variable $\boldsymbol{z}$ is unknown, direct maximizing the marginalized log-likelihood is difficult. Instead of solving for the log-likelihood of the incomplete data directly, EM algorithm first introduces a distribution $q(\boldsymbol{z})$ over the latent variables. Then it transforms the original maximization question to maximizing a lower bound for the log-likelihood of the complete data with $q(\boldsymbol{z})$ based on Jensen's inequality. Then EM algorithm takes on an iterative approach that consists of E-step (establishing the lower bound) and M-step to determine the maximizing arguments of $\boldsymbol{x}, \boldsymbol{z};, \boldsymbol{\tau}, \boldsymbol{\Sigma}, \boldsymbol{\mu}$ until convergence.

**E-step:** Since the latent variable $\boldsymbol{T}_{ki}$ is unknown, we replace $\boldsymbol{T}_{ki}$ with the conditional expectation $\mathrm{E}(\boldsymbol{T}_{ki}|\boldsymbol{x}_i; \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k)$. By solving for the expectation, we find

$$\hat{\boldsymbol{T}}_{k,i}^{(t+1)} = \frac{\tau_k^{(t)} \mathcal{N}(\boldsymbol{x}_i | \hat{\boldsymbol{\mu}}_k^{(t)}, \hat{\boldsymbol{\Sigma}}_k^{(t)})}{\sum\limits_k \tau_k^{(t)} \mathcal{N}(\boldsymbol{x}_i | \hat{\boldsymbol{\mu}}_k^{(t)}, \hat{\boldsymbol{\Sigma}}_k^{(t)})} \tag{3}$$

Here, $t$ stands for the current index of iterations. $\hat{\boldsymbol{T}}_{k,i}^{(t+1)}$ denotes the estimated probability of $z_i$ taking value $k$ after $t$ iterations of E-step and M-step

**M-step:** Next, we update $\hat{\tau}_k$, $\hat{\boldsymbol{\mu}}_k$, and $\hat{\boldsymbol{\Sigma}}_k$ to their new maximum likelihood estimations with the updated $\hat{\boldsymbol{T}}_{k,i}^{(t+1)}$. The updated parameters are listed as follows

$$\hat{\tau}_k^{(t+1)} = \frac{\sum_{i=1}^n \hat{\boldsymbol{T}}_{k,i}^{(t+1)}}{n} \tag{4}$$

$$\hat{\boldsymbol{\mu}}_k^{(t+1)} = \frac{\sum_{i=1}^n \hat{\boldsymbol{T}}_{k,i}^{(t+1)} \boldsymbol{x}_i}{\sum_{i=1}^n \hat{\boldsymbol{T}}_{k,i}^{(t+1)}} \tag{5}$$

$$\hat{\boldsymbol{\Sigma}}_k^{(t+1)} = \frac{\sum_{i=1}^n \hat{\boldsymbol{T}}_{k,i}^{(t+1)} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k^{(t+1)})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k^{(t+1)})^T}{\sum_{i=1}^n \hat{\boldsymbol{T}}_{k,i}^{(t+1)}} \tag{6}$$

EM will repeat the above E-step and M-step alternatively until a specific termination condition is met. In our implementation, we terminate the iteration when $\|\hat{\boldsymbol{T}}^{(t+1)} - \hat{\boldsymbol{T}}^{(t)}\|$ is smaller than a preset value.

## 1.2 Robust EM algorithm

Unfortunately, standard EM algorithm does not guarantee to converge to the global maximum and suffer from robustness problem. The performance of EM can be affected significantly by small deviation from the assumed model together with certain amount of outliers. In this section, we consider to modify the traditional EM by filtering out outliers. First, we define an observation as the sum of a centroid location and an error term. Based on the definition, we can detect outliers depends the magnitude of the error term. We will utilize Mahalanobis distance, which is scale-invariant, to measure the deviation from an observation to a specific 'true' location. The distance is calculated as follows:

$$A_{i,k} = \sqrt{(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k) \hat{\boldsymbol{\Sigma}}_k^{-1} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^T}$$

If the Mahalanobis distance $\boldsymbol{A}_{i,k}$ between the observation and the $k$th centroid position is greater than a threshold $\lambda$, the observation will be classified as an outlier. Potential outliers will be assigned a non-zero errors, otherwise the error term will be 0.

$$\hat{\boldsymbol{e}}_i = \begin{cases} 0 & \hat{\boldsymbol{A}}_{i,k} < \lambda \\ \boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k & \hat{\boldsymbol{A}}_{i,k} \geq \lambda \end{cases}$$

Note that the centroid location is unknown to us, we will use the estimated $\hat{\boldsymbol{\mu}}_k$ as a substitute. If an observation is identified as an outliers, it will be filtered out in M-step. This will allow us to prevent those corrupted points to bring impact to the learning results.

**E-step:** Our algorithm has the same estimation step as the traditional EM Gaussian algorithm. We update the $\hat{\boldsymbol{T}}_{k,i}^{(t+1)}$ with the same conditional expectation as the traditional EM does.

**M-step:** Based on our scheme, we will revise the estimates for $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$.

$$\hat{\boldsymbol{\mu}}_k^{(t+1)} = \frac{\sum\limits_{i \in I} \hat{\boldsymbol{T}}_{k,i}^{(t+1)} \boldsymbol{x}_i}{\sum\limits_{i \in I} \hat{\boldsymbol{T}}_{k,i}^{(t+1)}} \tag{7}$$

Here set $I$ contains all the index of points that have $\hat{e}$ equals to 0. If a point is considered as outlier, it will not be used in the $\hat{\boldsymbol{\mu}}_k$ update. We will only use non-outliers to update the new parameters. The same procedure will be applied to update of $\boldsymbol{\tau}_k$ and $\boldsymbol{\Sigma}_k$

$$\hat{\boldsymbol{\tau}}_k^{(t+1)} = \frac{\sum\limits_{i \in I} \hat{\boldsymbol{T}}_{k,i}^{(t+1)}}{\sum\limits_{i \in I} 1} \tag{8}$$

$$\hat{\boldsymbol{\Sigma}}_k^{(t+1)} = \frac{\sum\limits_{i \in I} \hat{\boldsymbol{T}}_{k,i}^{(t+1)} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k^{(t+1)})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k^{(t+1)})^\top}{\sum\limits_{i \in I} \hat{\boldsymbol{T}}_{k,i}^{(t+1)}} \tag{9}$$

Once the $\tau_k$, $\Sigma_k$, and $\mu_k$ are updated with their current mle value respectively, one iteration of the Robust EM is completed. We will continue the process until the termination condition is met. We can summarize the modified EM algorithm by the following steps.

**EM algorithm for Gaussian mixture:**
Step1: Fix the number of clusters $1 < K < n$, $\delta > 0$ and set s = t = 1.
Step2: Give initial value to $\hat{\tau}_k^{(t)}$, $\hat{\boldsymbol{\mu}}_k^{(t)}$, and $\hat{\boldsymbol{\Sigma}}_k^{(t)}$.
Step3: Pick $\lambda_s$ from a decreasing list $\lambda = \{\lambda_1, \lambda_2, ...\lambda_m\}$. Step4: Substitute $\hat{\boldsymbol{T}}_{k,i}^{(t+1)}$ with its conditional expectation by (3).
Step5: Update $\hat{\tau}_k^{(t+1)}$, and $\hat{\boldsymbol{\mu}}_k^{(t+1)}$ with $\hat{\boldsymbol{T}}_{k,i}^{(t+1)}$ by (4) and (7)
Step6: Update $\hat{\boldsymbol{\Sigma}}_k^{(t+1)}$ with $\hat{\boldsymbol{T}}_{k,i}^{(t+1)}$, $\hat{\tau}_k^{(t+1)}$, $\hat{\boldsymbol{\mu}}_k^{(t+1)}$ by (8)
Step7: Compare $\hat{\boldsymbol{T}}^{(t+1)}$ and $\hat{\boldsymbol{T}}^{(t)}$. If $\|\hat{\boldsymbol{T}}^{(t+1)} - \hat{\boldsymbol{T}}^{(t)}\| < \delta$, then break. Otherwise $t = t + 1$ and return to Step4.
Step8: If $s = m$, then break. Otherwise $s = s + 1$ and return to Step3.

## 1.3 Covariance Structure

Note that in the above M-step, updating $\hat{\boldsymbol{\Sigma}}_k^{(t+1)}$ has no restriction on its shape. In high dimension, such a flexible covariance term may lead to computational inefficiency and instability. To aid this issue, we can introduce structures to the updated covariance matrix term.

For example, if the covariance matrix is forced to be a constant structure, the updated $\hat{\boldsymbol{\Sigma}}_k^{(t+1)}$ will be:

$$\hat{\boldsymbol{\Sigma}}_k^{(t+1)} = \frac{\sum_{i \in I} \hat{\boldsymbol{T}}_{k,i}^{(t+1)} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k^{(t+1)})^T (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k^{(t+1)})}{d \sum_{i \in I} \hat{\boldsymbol{T}}_{k,i}^{(t+1)}} \times \mathbf{I}_{d \times d}$$

In this case, we consider that every dimension contributes equally to the covariance matrix. Therefore, we use the weighted average variance among all $d$ dimensions. The derivation of the following can be found in cov_struct pdf file.

If the covariance matrix is forced to be diagonal structure, the $j$th diagonal element of the updated $\hat{\boldsymbol{\Sigma}_k}$ will be:

$$\hat{\boldsymbol{\Sigma}}_{j,k}^{(t+1)} = \frac{\sum_{i \in I} \hat{\boldsymbol{T}}_{k,i}^{(t+1)} (\boldsymbol{x}_{i,j} - \hat{\boldsymbol{\mu}}_k^{(t+1)})^2}{\sum_{i \in I} \hat{\boldsymbol{T}}_{k,i}^{(t+1)}}$$

Here $\boldsymbol{x}_{i,j}$ denotes the $j$ the dimensional element of the $i$th observation.

We also introduce an innovative way to bridge the EEE and VVV model with a smooth transition. However, we can only solve for this case by numerical solution currently.

## 1.4 Threshold $\lambda$ setting

The threshold for Mahalanobis distance is a crucial factor in this algorithm; selecting a proper threshold value can improve the clustering result to a great extent. The probability of any given point lying outside a given threshold varies across different dimensions. In the high dimension, we need to the increase the $\lambda$ to cater to the increasing probability of incorrectly identifying an outlier. With an improperly set $\lambda$, the algorithm may identify all the data as outliers. During the RobustEM algorithm, the threshold $\lambda$ will gradually decrease exponentially from infinity to the preset threshold value. So, we can gradually screen out outliers without losing too many observations.

## 1.5 Initiation

The EM algorithm is sensitive to the initial condition, and a robust EM algorithm could have a greater dependence on the initial value selection. We initiate the robust EM algorithm in two different ways. One way is to utilize model-based Gaussian hierarchical agglomeration. This approach chooses a maximum-likelihood pair of clusters for merging at each stage. However, this method is not robust to outliers. mclust provides functions hc for model-based hierarchical agglomeration and hclass for determining the resulting classifications. Function hc implements fast procedures based on the multivariate normal classification likelihood. At this stage, our algorithm relies on hc function to create the initial value. Another way we generate the initialized value is to use the clustering result from MCLUST with the denoise option enabled. We will run Mclust funtion with a randomized noise component, so it would allow Mclust to detect potential outliers automatically. Check initial.cond() method for details.

### 1.6 Computation Patches

Due to the nature of the modified EM that it can filter out outliers, it is possible for RobustEM to run into cases such as too many points are considered as outliers. This situation can cause two main computational error. First, it may cause a cluster contains no point. This will make it possible for EM algorithm to continue M-step; $\hat{\boldsymbol{T}}_{\cdot,i}$ value for the $i$th observation are all close to 0. To prevent the first case, we will reinitialize the problematic cluster's $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ with an arbitrary value. To prevent the second case, we will stop updating the problematic points's $\hat{\boldsymbol{T}}$.

## 2 Simulation

This is a simple simulation to demonstrate the current performance of current robust EM build. The main goal of the RobustEM algorithm is to group multivariate observations without being affected by outliers. Therefore, it is necessary to evaluate the performance by outlier detection rate and classification accuracy. To evaluate the above two metrics, we chose to compare the Rand Index, adjusted Rand Index, and Faulty classification rate for cluster assignment and outlier detection. Rand Index measures the magnitude of two partitions that are in agreement with each other. A high Rand Index value stands for increased accuracy, and this metric is capped by 1, which stands for perfect classification. The adjusted Rand index is the corrected-for-chance version of the Rand index. Such a correction for chance establishes a baseline by using the expected similarity of all pair-wise comparisons between clustering specified by a random model. The faulty classification rate calculates the discrepancy between the two classifications. Algorithms with lower FCR values have better accuracy than those with high FCR values.

Data-set used in the simulation study is generated with a sample size n = 378. These data sets are a collection of 7 clusters generated randomly from a 2-dimensional t distribution. The parameters for these clusters are given as follows:

$$X_i \sim t(\mu_k, \Sigma_k, v_k)$$

Here $v_k$ stands for the degree of freedom for the $k$th t-distribution.

$$\mu_k \sim Unif(0, 100)$$

The covariance matrix is designed to be diagonal. Its $i$th diagonal term is:

$$\Sigma_{k,i,i} \sim Unif(1, C)$$

Here, C is an arbitrary constant. In addition, 42 independent noises are added to the data set as outliers. These outliers are generated by Cauchy distribution. Each outlier corresponds to a cluster in the mixture model. For example, the $i$th outlier that is generated based on the $k$th cluster will be drawn from

$$O_{k,i} \sim Cauchy(\mu_k, \Sigma_k)$$

To prevent cases where clusters are overlapped, or outliers fall into the cluster's center, we introduce parameters such as *separation* in the simulation function to give more control to the simulated data-set. We can guarantee that the squared Mahalanobis distance between each outlier to its nearest cluster center is greater than $\chi^2_{2,0.001}$, and the minimum Euclidean distance among the centers of the clusters is at least 25. An example data-set is shown in figure 1.

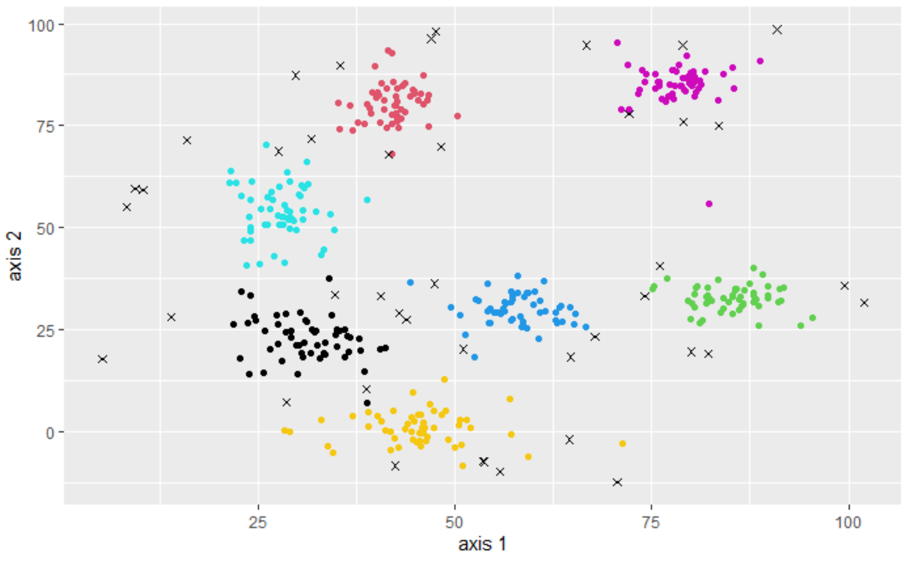


Figure 1: Random mixture sample generated by sim_mixture(cluster = 7, d = 2, separation = 25, n = 60, out_perc = 0.1, cov_scale = 1, df = 7). Cross stands for outliers, and colored points are non-outliers.

We will achieve the following result by applying Mclust and RobustEM to the previous data set. Mclust achieves 0.9703489 as the Random Index and 0.8647058 as the adjusted random index. The rate of mclust misclassifies outliers and cluster labels are 0.07619048 and 0.07142857, respectively. The total percentage of outliers mclust detects is 0.06666667, which is slightly lower than the actual value 0.1. Meanwhile, robustEM with mclust's result as the initial value achieves 0.9638595 as the Random Index and 0.8330295 as the adjusted random index. The rate of mclust misclassifies outliers and cluster labels are 0.08571429 and 0.08333333, respectively. The total percentage of outliers mclust detects is 0.1261905, which is slightly higher than the actual value 0.1.

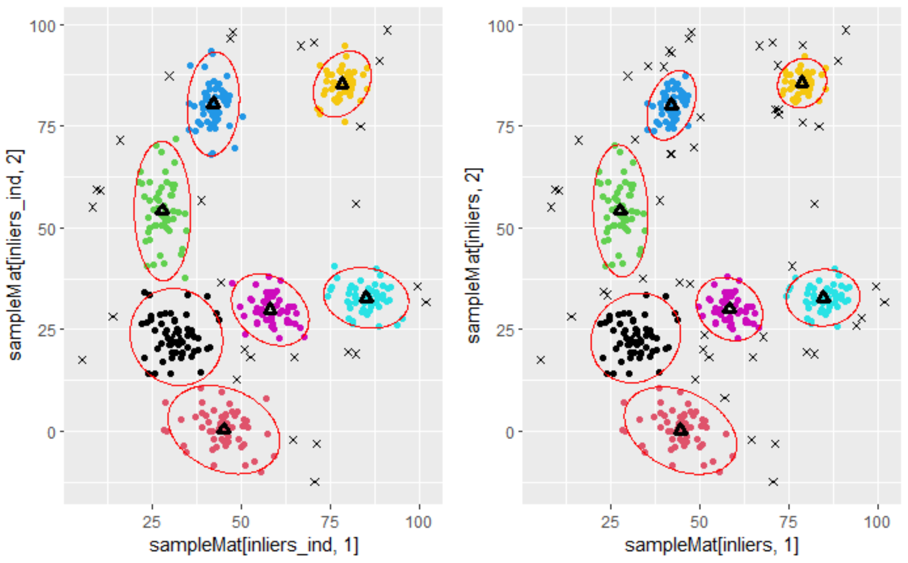
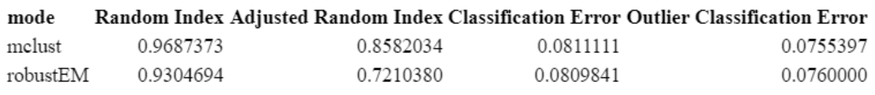


Figure 2: Clustering result of Mclust (left), RobustEM(right). Cross stands for outliers



| mode | Random Index | Adjusted Random Index | Classification Error | Outlier Classification Error |
|---|---|---|---|---|
| mclust | 0.9687373 | 0.8582034 | 0.0811111 | 0.0755397 |
| robustEM | 0.9304694 | 0.7210380 | 0.0809841 | 0.0760000 |

Figure 3: statistics of 150 rounds of simulations

We repeated the simulation process 150 times and generated a table that summarizes the performance of mclust and robustEM. mclust performs slightly better than robust EM.