ST793 Project: A Blogpost on "Doubly Enhanced EM Algorithm for Model-Based Tensor Clustering" by Mai et al

Ayumi Mutoh, Jisu Oh, Shih-Ni Prim

December 2, 2023

1 Introduction

In recent decades, tensor data have gained popularity in modern science, their high-dimensional structures often pose challenges for statistical analysis, specifically in model-based clustering. Model-based clustering is a statistical approach to data clustering, where observed data is considered to have been created from a finite combination of component models, such as the Gaussian mixture model (GMM). Since the formalization of the expected-maximization (EM) algorithm by Dempster et al. (1977), the EM algorithm has been widely employed in the majority of model-based clustering applications. While the GMMs can be readily extended to higher-order tensors using the standard EM algorithm, their performance can be further enhanced by integrating the DEEM algorithm, as proposed by Qing Mai and Deng (2022). Mai et al. consider a tensor normal mixture model (TNMM) that incorporates tensor correlation structure and variable selection for clustering and parameter estimation. They developed the DEEM algorithm which enables DEEM to excel in high-dimensional tensor data analysis. Similar to the EM algorithm, DEEM carries out an enhanced E-step and an enhanced M-step.

In this blogpost, we first introduce the DEEM methods with intermediate steps for the theoretical explanation. The objective is to break down the steps, making the derivation more accessible for our readers to follow. Subsequently, we will conduct a simulation study to evaluate the performance of DEEM.

2 Theoretical Derivation

2.1 EM Algorithm

Before delving into DEEM, we would like to review the EM algorithm and its functioning in clustering.

As we have learned in class, the EM algorithm is an iterative approach that cycles between two steps for maximum likelihood estimation in the presence of latent variables. The observed

data Y is incomplete and data Z is missing. The first step is to write down the joint likelihood, $L_c(\theta|Y,Z)$, of the "complete" data (Y,Z). The "E" step of the EM algorithm is to compute the conditional expectation of log-likelihood, $\log L_c(\theta|Y,Z)$, given Y assuming the true parameter value is $\theta^{(\nu)}$

$$Q(\theta,\theta^{(\nu)},Y) = E_{\theta^{(\nu)}}(\log L_c(\theta|Y,Z)|Y).$$

In the "M" step, we maximize $Q(\theta,\theta^{(\nu)},Y)$ with respect to θ with $\theta^{(\nu)}$ fixed.

The EM algorithm is well-known for use in unsupervised learning problems such as clustering with a mixture model. The process goes as follows:

1. Identify the number of clusters.
2. Define each cluster by generating a Gaussian model.
3. For every observation, calculate the probability that it belongs to each cluster (Ex. observation 12 has 40% probability of belonging to Cluster A and 60% probability of belonging to Cluster B.)
4. Using the above probabilities, recalculate the Gaussian models.
5. Repeat until observations "converge" on their assignments.

Let's consider a simple example. Suppose we have data X_i as shown in Figure 1, which comes from two distinct classes. We use this data to build a Gaussian model for each class. Since we don't know which class each observation belongs to, there is no straightforward way to construct two Gaussian models to partition the data. Therefore, we begin with a random guess of our Gaussian model parameters: $\mu_1, \sigma_1^2, \mu_2, \sigma_2^2$.

We have 'missing' data points X_i that we believe belong to either of the two distributions. After initializing two random Gaussian models, we compute the likelihood of each observation, X_i, being expressed in both of the Gaussian models. The next is the E-step, where we compute the probability that each X_i can belong to any of two distributions. Now we have a probability of belonging to either distribution for each point.

In the M-step, we update the parameters, $\mu_1, \sigma_1^2, \mu_2, \sigma_2^2$, of the model to their most likely values. For the new μ_1, we take a weighted average of all the points, weighted by the probability that they belong to the first distribution. Denoting p_i is the probability that X_i belongs to the first distribution.

$$\mu_1 = \frac{p_1X_1 + p_2X_2 + \cdots + p_nX_n}{p_1 + p_2 + \cdots + p_n}$$

The new σ_1^2 can be updated similarly.

$$\sigma_1^2 = \frac{p_1(X_1-\mu_1)^2 + p_2(X_2-\mu_1)^2 + \cdots + p_n(X_n-\mu_1)^2}{p_1 + p_2 + \cdots + p_n}$$

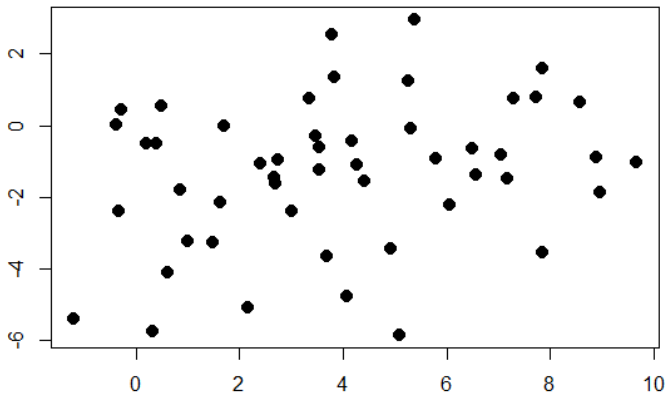

Figure 1: Mixture of two Gaussian Distributions

We repeat this process for μ_2 and σ_2^2 and update our distributions. We iterate through the E-step and M-step until convergence, obtaining two clusters as shown in Figure 2.

2.2 Tensor

While the term "tensor" might sound unfamiliar to some, tensors are simply multi-way arrays. Data is often structured as matrices, and they are in fact second-order tensors. When we use the term "tensor," we usually mean tensors of third-order and higher. You can think of a third-order tensor as a cube. There are then three dimensions from which to look at the tensor. These three dimensions are called "modes." If you are interested in knowing more about tensors, a very popular and highly cited paper by Kolda and Bader (2009) has lots of great details. So check it out! A figure from this paper is included in our Figure 3 to give you an idea of the different ways to look at a tensor. For this blogpost, we will focus on the method DEEM by Qing Mai and Deng (2022), so we will not delve into more details about tensors and will focus on how this method is an upgrade of the classical EM algorithm.

2.3 Doubly Enhanced EM Algorithm

3 Simulation Study

3.1 Data Generation

For our simulation studies, we follow the framework used in Qing Mai and Deng (2022). For each setting, K denotes the number of mixture groups, and noise is generated as a M^{th}-order

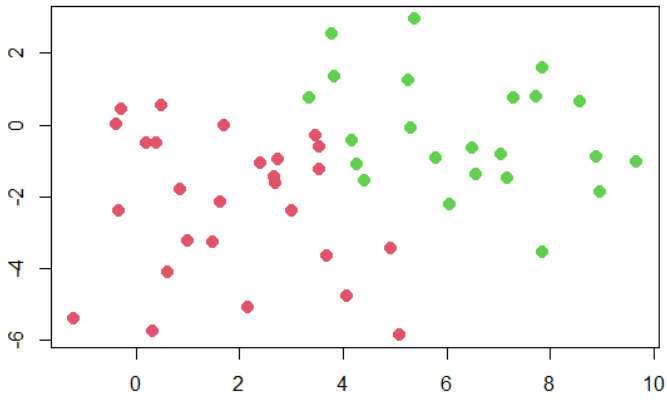

Figure 2: Clusters Found by EM algorithm

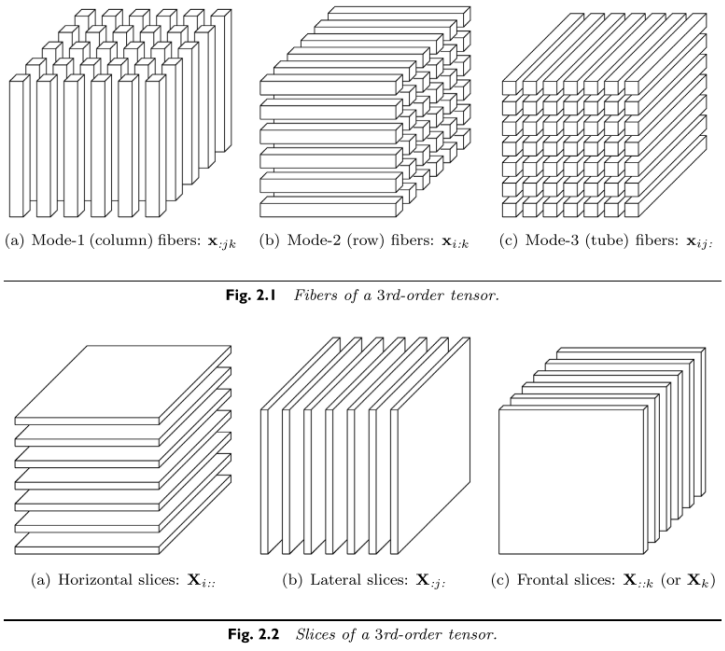

Figure 3: Dimensions and Terminology of a Tensor (from Kolda and Bader (2009))

tensor:

$$\mathbf{X}_i \sim \sum_{k=1}^{K} \pi_k^* \text{TN}(\boldsymbol{\pi}_k^*; \boldsymbol{\Sigma}_1^*, \cdots, \boldsymbol{\Sigma}_{\boldsymbol{M}}^*), i = 1, \cdots, n \tag{1}$$

For $K-1$ mixture groups, the $\mathbf{X}_i$ is given as a given $\mathbf{B}_k$ plus the noise above. For 1 mixture group, the values are simply the noise. Qing Mai and Deng (2022) designate two types of $\boldsymbol{\Sigma}_k^*$:

$$\boldsymbol{\Omega} = \begin{cases} AR(\rho): & \omega_{ij} = \rho^{|i-j|} \\ CS(\rho): & \omega_{ij} = \rho + (1-\rho)1(i=j). \end{cases}$$

For each setting, we generate 100 independent datasets, the same number of replicates as Qing Mai and Deng (2022) use, and present the mean error rate and standard deviation.

3.2 Settings

The settings are provided in Table 1. Note that, for B_k^*, the indices not included in the subscript is 0. In other words, B_k^* is a sparse tensor. We chose these four settings from the seven settings, because their settings are increasingly more computationally expensive, and we believe that these four settings demonstrate the advantage of the DEEM algorithm compared to the classical EM algorithm in terms of accuracy, as shown in Table 2.

Model	Parameters
M1	$K=2, p=10\times10\times4, \boldsymbol{\Sigma}_1^*=CS(0.3), \boldsymbol{\Sigma}_2^*=AR(0.8), \boldsymbol{\Sigma}_3^*=CS(0.3), \mathbf{B}^*_{2,[1:6,1,1]}=0.5$
M3	$K=3, p=10\times10\times4, \boldsymbol{\Sigma}_1^*=CS(0.3), \boldsymbol{\Sigma}_2^*=AR(0.8), \boldsymbol{\Sigma}_3^*=CS(0.5), \mathbf{B}^*_{2,[1:6,1,1]}=0.5, \mathbf{B}^*_{3,[1:6,1,1]}=-0.5$
M4	$K=4, p=10\times10\times4, \boldsymbol{\Sigma}_1^*=\mathbf{I}_{10}, \boldsymbol{\Sigma}_2^*=AR(0.8), \boldsymbol{\Sigma}_3^*=\mathbf{I}_4, \mathbf{B}^*_{2,[1:6,1,1]}=0.8, \mathbf{B}^*_{3,[1:6,1,1]}=-0.8$
M5	$K=6, p=10\times10\times4, \boldsymbol{\Sigma}_1^*=AR(0.9), \boldsymbol{\Sigma}_2^*=CS(0.6), \boldsymbol{\Sigma}_3^*=AR(0.9), \mathbf{B}^*_{2,[1:6,1,1]}=0.6, \mathbf{B}^*_{3,[1:6,1,1]}=1.2, \mathbf{B}^*_{4,[1:6,1,1]}=1.8, \mathbf{B}^*_{5,[1:6,1,1]}=2.4, \mathbf{B}^*_{6,[1:6,1,1]}=3$

Table 1: Simulation settings

3.3 Metrics

Note that it is as straightforward to calculate the mean error rate for a clustering problem than it is for a classification problem. Both methods return labels for the groups; however, the group labels do not matter. For example, if there are five observations and if their true group labels are $(1,1,2,2,2)$ and the methods return $(2,2,1,1,1)$, the error rate should be 0. In the paper,

the authors explain that mean clustering error rate is calculated by:

$$\min_{\prod} \frac{1}{n} \sum_{i=1}^{n} 1(\hat{Y}_i \neq \prod(Y_i)) \text{ over all possible permutations } \prod : \{1, \cdots, \} \mapsto \{1, \cdots, K\}$$

We thus created a function to permute the true labels, compare the estimated labels and the true labels, and return the lowest error rate.

To compare the speed of the two methods, we also record the computation time. Table 3 provides the mean computation time and standard error (in parentheses) for each setting.

3.4 External R Packages and Functions

For the DEEM algorithm, we use the function DEEM; for the standard EM algorithm, we use the function TGMM. Both functions are from the R package TensorClustering. We use the Trnorm function from the R package Tlasso to generate tensor noise with designated covariance matrices. We use the permutations function in the gtools package to permute true labels. In short, be sure to install the three R packages: TensorClustering, Tlasso, and gtools if you would like to reproduce our simulation.

3.5 Simulation result

The error rates and computation time are shown in Tables 2 and 3. It is clear that DEEM has lower mean error rates in all four settings. The computation time tells a different story, however. As seen in Table 3, DEEM is not always the winner in terms of time. As the setting becomes more complicated and estimating the clusters becomes more challenging, it takes longer for DEEM to converge. In fact, judging from the amount of time it took to run the setting M5, it is possible that DEEM reached the maximum iterations.

Model	DEEM	EM
M1	0.41 (0.05)	0.45 (0.03)
M3	0.46 (0.09)	0.56 (0.05)
M4	0.35 (0.03)	0.57 (0.06)
M5	0.31 (0.11)	0.43 (0.06)

Table 2: Error Rates from 100 Replicates

Model	DEEM	EM
M1	0.72 (0.45)	0.93 (0.39)
M3	13.95 (7.78)	7.74 (3.99)
M4	15.8 (0.81)	21.9 (9.68)
M5	332.96 (124.38)	14.66 (5.88)

Table 3: Computation Time (seconds) from 100 Replicates

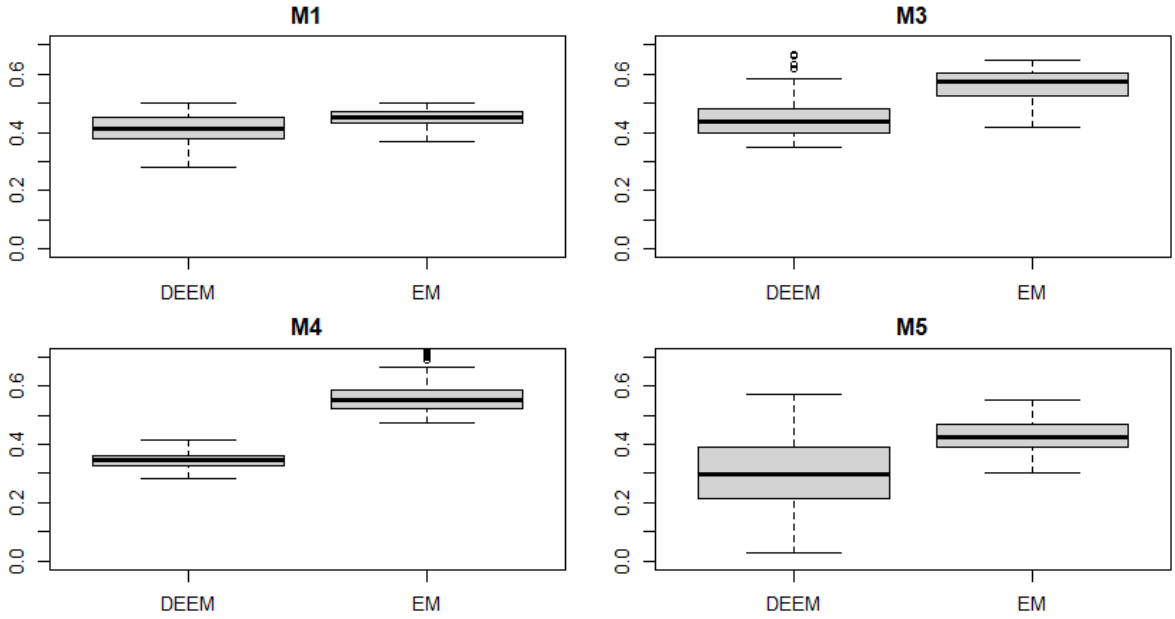

Figure 4: Boxplots of Mean Error Rates from 100 Replicates

Next we transform the values in the tables into figures, which sometimes tell clearer pictures. As shown in Figure 4, DEEM always has lower mean error rates. However, as the model becomes complicated, DEEM's error rates become more varied, even though the mean rate is still lower. In Figure 5, the story seems more complicated. (Note that we cannot make the y-axis all the same for the four plots, because the computation time for DEEM for M5 is so long, which would make some of the boxes very small and not informative.) For the two settings M1 and M4, DEEM has lower computation time. For M3, the computation time for DEEM is much more varied, and EM has overall shorter computation time. For M5, DEEM has very long computation time; in fact, the 100 replicates took almost 10 hours to run. It is unclear if the reduction in error rate is worht the much longer computation time.

4 Discussion

In this blogpost, we review a new method proposed by Qing Mai and Deng (2022), which is essentially an upgraded version of the classical EM algorithm. This new method, DEEM, tend to have lower error rates on tensor data. However, the running time could be prohibitive.

5 Conclusion

[to be added]

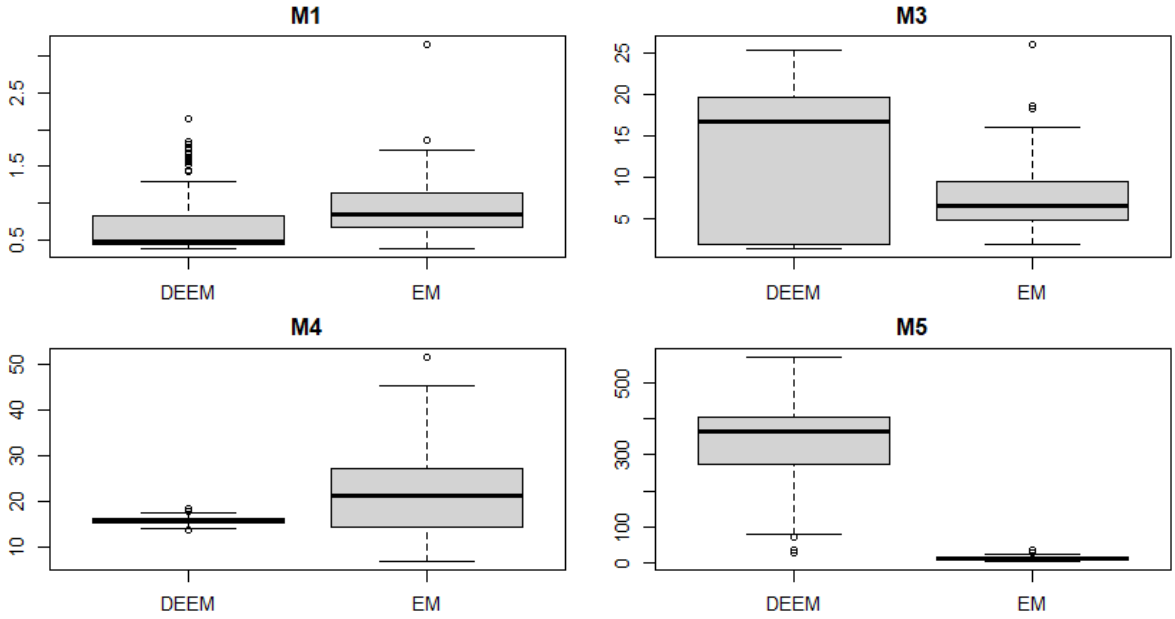

Figure 5: Boxplots of Mean Computation Time (in seconds) from 100 Replicates

References

Dempster, A., Laird, N. and Rubin, D. (1977) Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, **39**, 1–22.

Kolda, T. G. and Bader, B. W. (2009) Tensor decompositions and applications. *SIAM Review*, **51**, 455–500. URL`https://doi.org/10.1137/07070111X`.

Qing Mai, Xin Zhang, Y. P. and Deng, K. (2022) A doubly enhanced em algorithm for model-based tensor clustering. *Journal of the American Statistical Association*, **117**, 2120–2134.