Simple Review of Mixtures of Linear Mixed Models

Daeyoung Lim[*]
Department of Statistics
Korea University

January 6, 2016

1 Mixture models

1.1 Overview

By *mixture models*, we refer to a class of statistical models that assume the presence of subpopulations. The data that we observe is a "*pooled*" data (or "grouped data" if we follow the terminology of the paper) having multiple subpopulations, or *mixture components*. With the pooled data, we seek to find a way of learning how many mixture components there are alongside obtaining the parameter estimates of each component.

1.2 Goals of MM

- Split the data into components. This problem is similar to classification, or clustering.
  $\Rightarrow$ achieved through VGA
- Each component has a structure of a regression model, whether it be a linear mixed model, or generalized linear model etc. Fit each model with an appropriate regression model.
  $\Rightarrow$ achieved through VA

This paper suggests a method of attaining both goals at the same time through variational approximation and variational greedy algorithm which uses the lower bound (or the *free energy*) value for splitting.

1.3 Mixtures of Linear Mixed Models

A linear mixed model is a regression model that imports random effects as well as the fixed effects that already exist in generalized linear models. (The linear mixed model postulated in this model differs from the one that we considered last time. The current model is an extremely simplified version of the generalized model from last paper.)

The generalized linear model was

$$\begin{aligned}\boldsymbol{y}_i|\boldsymbol{u}_i &\overset{ind.}{\sim} \exp\left\{\boldsymbol{y}_i^T\left(\boldsymbol{X}_i\boldsymbol{\beta}+\boldsymbol{Z}_i\boldsymbol{u}_i\right)-\mathbf{1}_i^T b\left(\boldsymbol{X}_i\boldsymbol{\beta}+\boldsymbol{Z}_i\boldsymbol{u}_i\right)+\mathbf{1}_i^T c\left(\boldsymbol{y}_i\right)\right\}\\ \boldsymbol{u}_i &\overset{ind.}{\sim} \mathcal{N}\left(\mathbf{0},\boldsymbol{\Sigma}\right)\end{aligned}$$

[*]Prof. Taeryon Choi

where $\boldsymbol{u}_i$ are the random effects vectors. The model that we will be considering is, in one way, much simpler than this one-parameter exponential family linear mixed models but at the same time displays more complexity in that it has a dependency on to which mixture each $\boldsymbol{y}_i$ belongs. This dependency is modeled hierarchically and efficiently contained within the Bayesian framework. Thus, the model becomes

$$\boldsymbol{y}_i | z_i = j, \theta = \boldsymbol{X}_i \boldsymbol{\beta}_j + \boldsymbol{W}_i \boldsymbol{a}_i + \boldsymbol{V}_i \boldsymbol{b}_j + \boldsymbol{\varepsilon}_i$$

where $z_i = j$ indicates that the i^{th} cluster corresponds to j^{th} mixture component.

Before starting the splitting procedure, we initially have only 1 cluster. We will, as in every statistical classification model, divide cluster(s) according to a properly assigned probability. In this paper, the "*properly assigned*" probability is calculated through softmax function:

$$\text{softmax}\left(j, z_1, z_2, \ldots, z_N\right) = \frac{e^{z_j}}{\sum_{i=1}^{N} e^{z_i}}.$$

2 Paper Review

2.1 Variational approximation for MLMM

Model Specifications	
Likelihood	
$y_i \vert \theta$	$\mathcal{N}\left(\mathbf{0}, X_i \Sigma_{\beta_i} X_i^T + \sigma_{a_j}^2 W_i W_i^T + \sigma_{b_j}^2 V_i V_i^T + \Sigma_{ij}\right)$
Priors	
β_j	$\mathcal{N}\left(\mathbf{0}, \Sigma_{\beta_j}\right)$
$a_i \vert \sigma_{a_j}^2$	$\mathcal{N}\left(\mathbf{0}, \sigma_{a_j}^2 \boldsymbol{I}_{s_1}\right)$
$b_j \vert \sigma_{b_j}^2$	$\mathcal{N}\left(\mathbf{0}, \sigma_{b_j}^2 \boldsymbol{I}_{s_2}\right)$
$z_i = j \vert \delta_j$	$\text{softmax}\left(j, u_i^T \delta_1, u_i^T \delta_2, \ldots, u_i^T \delta_k\right)$
Hyperpriors	
$\sigma_{a_j}^2$	$\text{IG}\left(\alpha_{a_j}, \lambda_{a_j}\right)$
$\sigma_{b_j}^2$	$\text{IG}\left(\alpha_{b_j}, \lambda_{b_j}\right)$
σ_{jl}^2	$\text{IG}\left(\alpha_{jl}, \lambda_{jl}\right)$
δ	$\mathcal{N}\left(\mathbf{0}, \Sigma_\delta\right)$

- $\theta = \begin{bmatrix} \beta^T & a^T & b^T & \sigma_a^{2\,T} & \sigma_b^{2\,T} & \sigma^{2\,T} & \delta^T & z^T \end{bmatrix}^T.$
- $\beta = \begin{bmatrix} \beta_1^T & \cdots & \beta_k^T \end{bmatrix}^T.$
- $a = \begin{bmatrix} a_1^T & \cdots & a_n^T \end{bmatrix}^T.$

- $\sigma_a^2 = \begin{bmatrix}\sigma_{a_1}^2 & \cdots & \sigma_{a_k}^2\end{bmatrix}^T$.
- $\sigma_b^2 = \begin{bmatrix}\sigma_{b_1}^2 & \cdots & \sigma_{b_k}^2\end{bmatrix}^T$.
- $\sigma_j^2 = \begin{bmatrix}\sigma_{j1}^2 & \cdots & \sigma_{jg}^2\end{bmatrix}^T$.
- $\delta = \begin{bmatrix}\delta_2^T & \cdots & \delta_k^T\end{bmatrix}^T$.
- $z = \begin{bmatrix}z_1 & \cdots & z_n\end{bmatrix}^T$.
- $\text{softmax}\left(j, u_i^T\delta_1, u_i^T\delta_2, \ldots, u_i^T\delta_k\right) = \frac{\exp\left(u_i^T\delta_j\right)}{\sum_{\ell=1}^k \exp\left(u_i^T\delta_\ell\right)}$.
- $\Sigma_{ij} = \text{blockdiag}\left(\sigma_{j1}^2 \boldsymbol{I}_{\kappa_{i1}}, \ldots, \sigma_{jg}^2 \boldsymbol{I}_{\kappa_{ig}}\right)$ where $\sum_{\ell=1}^g \kappa_{i\ell} = n_i$.

Under *mean-field* assumption, the lower bound should be obtained through $\mathrm{E}\left\{\log p(y,\theta)\right\} - \mathrm{E}\left(\log q(\theta)\right)$.

$$\begin{aligned} p(y,\theta) = &\prod_{i=1}^n \prod_{j=1}^k \left\{p\left(y_i | z_i = j, \beta_j, a_i, b_j, \Sigma_{ij}\right) \cdot p\left(a_i | \sigma_{a_j}^2\right) p\left(z_i = j\right)\right\}^{\zeta_{ij}} \\ &\times p(\delta) \prod_{j=1}^k \left\{p(\beta_j) p\left(b_j | \sigma_{b_j}^2\right) p\left(\sigma_{a_j}^2\right) p\left(\sigma_{b_j}^2\right) \prod_{\ell=1}^g p\left(\sigma_{j\ell}^2\right)\right\} \end{aligned}$$

where $\zeta_{ij} = I\left(z_i = j\right)$. Therefore, the log-likelihood is

$$\begin{aligned} \log p(y,\theta) = &\sum_{i=1}^n \sum_{j=1}^k \zeta_{ij} \left\{\log p\left(y_i | z_i = j, \beta_j, a_i, b_j, \Sigma_{ij}\right) + \log p\left(a_i | \sigma_{a_j}^2\right) + \log p\left(z_i = j\right)\right\} + \log p\left(\delta\right) \\ &+ \sum_{j=1}^k \left\{\log p\left(\beta_j\right) + \log p\left(b_j | \sigma_{b_j}^2\right) + \log p\left(\sigma_{a_j}^2\right) + \log p\left(\sigma_{b_j}^2\right) + \sum_{\ell=1}^g \log p\left(\sigma_{j\ell}^2\right)\right\} \end{aligned}$$

Deterministically assigning parametric family for each variational parameter, it is not as hard to compute the lower bound. In fact, the paper considers 3 different models, one of which considers no hierarchical centering whereas the others consider either single layer of hierarchical centering or double.

2.2 Variational Greedy Algorithm

VGA incorporates the VA addressed in the previous section into a framework of splitting the data and optimizing the parameters simultaneously. The summary of VGA would be

1. Fit 1-component model.
2. For each component, randomly cluster the member observations and set the variational posterior parameters according to the given rule.
3. Perform VA only updating the new partitioned sets.
4. For each component, perform the steps 2 $\sim$ 3 M times (since we randomly partition the cluster, we get different lower bound each time) and choose a partition that yields the highest lower bound.

5. A “successful” split is checked via the increase in log-marginal likelihood:

$$\log\left\{\frac{\exp\left(u_i^T\mu_{\delta_j}\right)}{\sum_\ell \exp\left(u_i^T\mu_{\delta_\ell}\right)}\right\}.$$

   (Refer to p.575)

6. Reverse the parameter estimates if the split is unsuccessful.

7. After looping through all the mixture components performing from step 2 to 6, apply full VA(in comparison to partial VA in step 3.). In other words, update all parameter estimates.

8. Stop splitting if all splits are unsuccessful. (Stopping rule)

The original text explaining the variational greedy algorithm LACKS CLARITY in that it does NOT give any definition of the term log-marginal likelihood and still uses it. Furthermore, it is sensible to reverse the parameter estimates if the split is unsuccessful but the paper hardly has any description about this.

3 Code Review

3.1 Results

- The results were different from those given in the paper. For **algorithm 1**, my R returned 11 components whereas the paper gave 16. For **algorithm 2**, my result was 5 components whereas the paper's was 6. In fact, the author of the paper carried out the same algorithm multiple times to see if the resulting numbers of components were the same. For algorithm 2, out of 5 in total, 3 resulted in 6 components and 2 in 7 components.

- The inaccuracy of this algorithm rises from estimating the scale parameter of nonstandardized t-distribution. The log-likelihood function of nonstandardized t-distribution is

$$\ell(\sigma) = \log\Gamma\left(\frac{v+1}{2}\right) - \log\Gamma\left(\frac{v}{2}\right) - \frac{1}{2}\log(\pi v) - \log\sigma - \frac{v+1}{2}\log\left\{1+\left(\frac{x-\mu}{\sigma}\right)^2\right\}$$

  which is equivalent to the form of $-\log\sigma - \log\left(1+1/\sigma^2\right)$. The second derivative is

$$\frac{d^2}{d\sigma^2}\ell(\sigma) = \frac{\sigma^4-4\sigma^2-1}{\left(\sigma^3+\sigma\right)^2}.$$

  The real roots for $\sigma^4-4\sigma^2-1=0$ are $\pm\sqrt{2+\sqrt{5}}$ which indicates that $\ell(\sigma)$ is not convex. This yields the problem that the optimization is likely to end up reaching its local optimum rather than global.

- The code is terribly slow. R is not a good language to write codes with a lot of loops involved since loops have significantly high overhead in R. This could be easily rectified by rewriting the code in Fortran or C++.