Review on Regression Density Estimation with Mixtures of Heteroscedastic Experts

Daeyoung Lim[*]
Department of Statistics
Korea University

January 14, 2016

1 Finite Gaussian Mixture Model

Gaussian mixture model is a way of representing a probability density function as a sum of weighted Gaussian component densities.

$$p(y|x) = \sum_{j=1}^{k} \pi_j \mathcal{N}\left(y|\mu_j, \sigma_j^2\right)$$

where $\mathcal{N}\left(y|\mu_j, \sigma_j^2\right)$ is the density function of the i^{th} component Gaussian and $\sum_{j=1}^{k} \pi_j = 1, \forall \pi_j \geq 0$. Normally, the mean and variance of each component are fixed values but in a model that allows heteroscedasticity, the mean and variance are functions of the covariates: $\mu_j\left(x\right), \sigma_j^2\left(x\right)$. This naturally yields the following representation of heteroscedastic Gaussian mixtures model:

$$p(y|x) = \sum_{j=1}^{k} \pi_j\left(x\right) \mathcal{N}\left(y|\mu_j\left(x\right), \sigma_j^2\left(x\right)\right).$$

In this paper, the author assumes that the heteroscedasticy implies a linear structure within the means and variances.

As with any mixtures model, RDE-MHN(Regression density estimation-mixtures of heteroscedastic normals) has 2 goals:

- Select the number of components, k.
- Select the number of covariates so as to avoid overfitting or underfitting.

Other than these 2 goals, there is another important issue in mixtures model.

- Overcome the local maxima problem and make sure the optimization converges to the global maximum.

We use the variational approximation for fitting and use the estimated likelihood and lower bound to determine the number of components and variables to be included. The local maxima problem was effectively dealt with via *split-and-merge* algorithm. In fitting the model, we need to keep in mind that we have three equations to fit: *mean*, *variance*, and *gate*.

[*]Prof. Taeryon Choi

2 Paper Review

2.1 Model Specifications

2.1.1 Likelihood

Likelihood	
$y_i \mid \delta_i = j, \boldsymbol{x}_i, \boldsymbol{\beta}, \boldsymbol{\alpha} \sim$	$\mathcal{N}\left(\boldsymbol{v}_i' \boldsymbol{\beta}_j, \exp\left(\boldsymbol{w}_i' \boldsymbol{\alpha}_j\right)\right) \quad , i = 1, \ldots, n$

- $(y_i, \boldsymbol{x}_i)\,, \quad i = 1, \ldots, n$: n observations where y_i are univariate responses and $\boldsymbol{x}_i$ is i^{th} row of the design matrix X, or i^{th} *covariate vector.*
- δ_i: a latent variable indicating from which component y_i comes. Since there are k components, $\delta_i \in \{1, 2, \ldots, k\}$.
- The latent variable δ_i follows the softmax function:

$$p\left(\delta_i = j \mid \boldsymbol{\gamma}, \boldsymbol{x}_i\right) = \frac{\exp\left(z_i' \gamma_j\right)}{\sum_{\ell=1}^{k} \exp\left(z_i' \gamma_\ell\right)} \quad j = 1, \ldots, k; \; i = 1, \ldots, n,$$

  where $\boldsymbol{\gamma}_j = (\gamma_{j1}, \ldots, \gamma_{jr})'$ is a vector of unknown parameters in the gating model of the j^{th} component and $\boldsymbol{z}_i$ is a sub-vector of $\boldsymbol{x}_i$ containing the covariates used to model the mixxing probabilities.
- $\boldsymbol{v}_i, \; \boldsymbol{w}_i$: the covariate vectors in the mean and variance models respectively (which are sub-vectors of $\boldsymbol{x}_i$).

2.1.2 Priors

Priors	
$\boldsymbol{\beta}_j$	$\mathcal{N}\left(\boldsymbol{\mu}^0_{\beta_j}, \Sigma^0_{\beta_j}\right), \quad j = 1, \ldots, k$
$\boldsymbol{\alpha}_j$	$\mathcal{N}\left(\boldsymbol{\mu}^0_{\alpha_j}, \Sigma^0_{\alpha_j}\right), \quad j = 1, \ldots, k$
$\boldsymbol{\gamma}$	$\mathcal{N}\left(\boldsymbol{\mu}^0_{\gamma}, \Sigma^0_{\gamma}\right)$
$\delta_i = j \mid \boldsymbol{\gamma} \; (= p_{ij})$	$\text{softmax}\left(j, z_i' \gamma_1, \ldots, z_i' \gamma_k\right)$

- $\boldsymbol{\beta}_j$: unknown parameters for the mean model.
- $\boldsymbol{\alpha}_j$: unknown parameters for the variance model.
- δ_i, γ: unknown parameters for the gating model.

2.2 Fitting the model

Parameter estimation does not diverge from the variational approximation framework. Compute the lower bound and maximize it! The maximization scheme is not a fixed object. It could either be a gradient-based method or a Hessian-based method or the EM-algorithm could be used as well. In this paper, the author uses the EM algorithm to optimize the lower bound.

2.3 Split-and-Merge algorithm

Aside from fitting the parameters, we should also split the components into several parts and also select a subset of covariates that prevents overfitting. In pursuit of such a goal, we employ the *split-and-merge* algorithm where we go back and forth between splitting and merging the most seemingly plausible two components whereby we gradually approach the most desirable number of components and also explore the most appropriate subset of all features that has the most explaining power and does not overfit the model.

2.3.1 Merge criterion

To start with the merging scheme,

- there should be a *metric* that measures how close two components are to decide whether or not the components should be merged.
- there should be a refined way of resetting the parameter estimates once the components are merged.

Here the *metric* is the one that we discuss in mathematical analysis courses that satisfies 3 conditions:

1. $d(p_i, p_j) \geq 0$ and $d(p_i, p_j) = 0$ iff $p_i = p_j$. (Nonnegativity)
2. $d(p_i, p_j) = d(p_j, p_i)$. (Symmetry)
3. $d(p_i, p_j) \leq d(p_i, p_r) + d(p_r, p_j)$. (Triangle inequality)

In this paper, the author suggests the *symmetrized KL divergence* or in short *KL distance.* If we denote KL distance $\mathrm{KL}\,(P, Q)$ and KL divergence $\mathrm{KL}\,(P||Q)$, then the relation is defined as follows:

$$\mathrm{KL}\,(P, Q) := \frac{1}{2}\left(\mathrm{KL}\,(P||Q) + \mathrm{KL}\,(Q||P)\right).$$

By doing so, the KL distance recovers the symmetry property and triangle inequality property of a metric. Considering the responses follow normal distribution, the average KL distance becomes

$$\mathrm{KL}\,(j_1, j_2) = \frac{1}{4n}\sum_{i=1}^{n}\left(\frac{\left(\boldsymbol{v}_i'\boldsymbol{\mu}^q_{\beta_{j_1}} - \boldsymbol{v}_i'\boldsymbol{\mu}^q_{\beta_{j_2}}\right)^2 + \exp\left(\boldsymbol{w}_i'\boldsymbol{\mu}^q_{\alpha_{j_1}}\right)}{\exp\left(\boldsymbol{w}_i'\boldsymbol{\mu}^q_{\alpha_{j_2}}\right)} + \frac{\left(\boldsymbol{v}_i'\boldsymbol{\mu}^q_{\beta_{j_1}} - \boldsymbol{v}_i'\boldsymbol{\mu}^q_{\beta_{j_2}}\right)^2 + \exp\left(\boldsymbol{w}_i'\boldsymbol{\mu}^q_{\alpha_{j_2}}\right)}{\exp\left(\boldsymbol{w}_i'\boldsymbol{\mu}^q_{\alpha_{j_1}}\right)} - 2\right).$$

If the KL distance turns out small, it suggests that the two components being compared are not that far away from each other. In other words, they are *most likely to be merged.* The merging procedure is constructed as follows:

1. Compute the pair (j_1, j_2) whose KL distance is the smallest.

2. Confirm if the merge improves the lower bound.

3. Check if the number of merging operations exceeded the maximum,$C^{\max}_{\text{merge}}$, set in advance.

Once two components are merged, there has to be a rule that incorporates the information that two components had in advance and produces efficient initial values for parameters of the newly created component. Recal that there are three parts: mean, variance and gating models. The following is how to set the initial values.

$$\begin{aligned}
\boldsymbol{\mu}^q_{\beta_{j'}} &= \frac{\overline{q}_{.j_1}\boldsymbol{\mu}^q_{\beta_{j_1}} + \overline{q}_{.j_2}\boldsymbol{\mu}^q_{\beta_{j_2}}}{\overline{q}_{.j_1} + \overline{q}_{.j_2}}, \quad \Sigma^q_{\beta_{j'}} = \frac{\overline{q}_{.j_1}\Sigma^q_{\beta_{j_1}} + \overline{q}_{.j_2}\Sigma^q_{\beta_{j_2}}}{\overline{q}_{.j_1} + \overline{q}_{.j_2}}, \\
\boldsymbol{\mu}^q_{\alpha_{j'}} &= \frac{\overline{q}_{.j_1}\boldsymbol{\mu}^q_{\alpha_{j_1}} + \overline{q}_{.j_2}\boldsymbol{\mu}^2_{\alpha_{j_2}}}{\overline{q}_{.j_1} + \overline{q}_{.j_2}}, \quad \Sigma^q_{\alpha_{j'}} = \frac{\overline{q}_{.j_1}\Sigma^q_{\alpha_{j_1}} + \overline{q}_{.j_2}\Sigma^q_{\alpha_{j_2}}}{\overline{q}_{.j_1} + \overline{q}_{.j_2}},
\end{aligned}$$

where $\overline{q}_{.j} = \frac{1}{n}\sum_{i=1}^{n} q_{ij}$ and $q_{ij'} = q_{ij_1} + q_{ij_2}$. Other parameters should be fixed at current values.

2.3.2 Split criterion

While exploring each component, there has to be a measure on which we can depend in determining whether we should split the component into two separate ones. This is equivalent to measuring the *likelihood* of each component: how likely each component is from the perspective of the parameters. Therefore, we calculate the joint log-likelihood at each component point and see if the component is *reliable*. The measure of reliability is given as follows:

$$R(j) = \frac{1}{n}\sum_{i=1}^{n} \log \hat{p}_j(\boldsymbol{x}_i) = \frac{1}{n}\sum_{i=1}^{n}\left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\boldsymbol{w}'_i\boldsymbol{\mu}^q_{\alpha_j} - \frac{1}{2}\frac{\left(y_i - \boldsymbol{v}'_i\boldsymbol{\mu}^q_{\beta_j}\right)^2}{\exp\left(\boldsymbol{w}'_i\boldsymbol{\mu}^q_{\alpha_j}\right)}\right).$$

Therefore, the splitting procedure is given as follows:

1. Find the component that displays the smallest $R(j)$.

2. Check if the split improves the lower bound.

3. Check if the number of split exceeded the maximum, $C^{\max}_{\text{split}}$, set in advance.

Once a component is split, there has to be a rule that sets the initial values for the parameter estimates of the two newly created components. This time, the rule is significantly simpler. Set both components' parameters same as the ones from the component before splitting. Put simply, duplicate 2 of the not-yet-split component with one exception: $q_{ij_1} = q_{ij_2} = q_{ij'}/2$.

2.4 Model Selection

The previous section was devoted to discussing how we can cluster the observations into multiple separate components using certain criteria. Given a design matrix, the previous chapter could be translated as row-wise operation whereas the current chapter which will shortly be discussed is going to be about how to choose the columns. *Feature selection*, *variable selection*, *model selection*, or whatever you might call this, it is simply choosing the covariates that should be included into a certain model. Under the model that we are now discussing where heteroscedasticity is under consideration, we generated three component models: mean, variance, and gating model. Therefore, we have no choice but to come up with an efficient algorithm that explores all three models providing sufficient explaining power as to why one covariate should be included into what model.

Variational approximation lends itself easily to model selection by offering its lower bound as a tool for model selection. As such, we will look whether the lower bound increases as we include covariates one by one. The three models need not be explained separately since they only rely on the increase in lower bound and they are eventually incorporated into the full algorithm. The gating model, however, needs additional clarification.

2.4.1 Gating model selection

Gating model selection adopts a new measure called *distance correlation.* It was proven that it is the same as the *Brownian covariance.* Let $x_{\hat{\ell}}$ denote the covariate that has the largest distance correlation with y. Then, we have three cases:

1. If $\hat{\ell} \notin C_m \bigcup C_v$, then $\hat{\ell}$ is included in the gating model if it improves the lower bound.
2. If $\hat{\ell} \in C_m$ or $\hat{\ell} \in C_v$, $\hat{\ell}$ is included in the gating model if it improves the lower bound.
3. If $x_{\hat{\ell}}$ does not improve the lower bound, we consider the covariate with the next largest distance correlation until we no longer have any covariate left.

The full algorithm is given in the paper.

3 Code Review

3.1 Easy example

Since this is a simulation, the author set in advance the number of components to 3. The resultant number of components was also 3. Let the columns denote the components and the rows denote the parameters.

$$\boldsymbol{\mu_\beta} = \begin{bmatrix} -5.005175 & 4.931319 & 2.070027 \\ 3.010657 & -1.954091 & -1.954091 \\ 0.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 \\ -4.020815 & 4.150012 & 1.762546 \\ 0.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 \end{bmatrix}$$

$$\boldsymbol{\mu_\alpha} = \begin{bmatrix} -0.926359 & -2.0850771 & -1.088369 \\ 1.370972 & 1.4123877 & -3.178568 \\ 0.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 \\ -2.853680 & -0.7307139 & 3.100485 \\ 0.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 \end{bmatrix}$$

$$\boldsymbol{\mu}_\gamma = \begin{bmatrix} 0 & -0.6961027 & 0.6338794 \\ 0 & 3.4261816 & 4.1524135 \\ 0 & 0.0000000 & 0.0000000 \\ 0 & 0.0000000 & 0.0000000 \\ 0 & -2.1298831 & -5.4977586 \\ 0 & 0.0000000 & 0.0000000 \\ 0 & 0.0000000 & 0.0000000 \\ 0 & 0.0000000 & 0.0000000 \end{bmatrix}$$

Because the author has not written down the true parameters, I am not sure if my results are correct. Neither am I sure if it is a coincidence that all thre models(mean, variance, and gating) have the same covariates.

3.2 Hard example

This example involves 1,000 covariates (high-dimension), 500 observations, and 3 components. Therefore, it is basically impossible to jot down the results of the R code. However, it turns out that the algorithm produces a result where the estimate vectors are sparse with a large number of zeros. It implies that the algorithm somehow ends up reaching a degenerate model.