EM Algorithm for Finite Mixture Regression

HW 5 of STAT 5361 Statistical Computing

Biju Wang[*]

10/09/2018

1 E- and M-Steps Derivations

1.1 E-Step Derivation

$$
\begin{aligned}
Q(\Psi|\Psi^{(k)}) &= \sum_Z \left[p(Z|\mathbf{y}, X, \Psi^{(k)}) \log p(\mathbf{y}, Z|X, \Psi)\right] && (1)\\
&= \sum_Z \left[p(Z|\mathbf{y}, X, \Psi^{(k)}) \log \prod_{i=1}^n p(y_i, \mathbf{z}_i|\mathbf{x}_i, \Psi)\right] && (2)\\
&= \sum_{i=1}^n \sum_Z \left[p(Z|\mathbf{y}, X, \Psi^{(k)}) \log p(y_i, \mathbf{z}_i|\mathbf{x}_i, \Psi)\right] && (3)\\
&= \sum_{i=1}^n \sum_{\mathbf{z}_i} \left[p(\mathbf{z}_i|\mathbf{y}, X, \Psi^{(k)}) \log p(y_i, \mathbf{z}_i|\mathbf{x}_i, \Psi)\right] && (4)\\
&= \sum_{i=1}^n \sum_{\mathbf{z}_i} \left[p(\mathbf{z}_i|y_i, \mathbf{x}_i, \Psi^{(k)}) \log p(y_i, \mathbf{z}_i|\mathbf{x}_i, \Psi)\right] && (5)\\
&= \sum_{i=1}^n \sum_{j=1}^m \left[p(\mathbf{z}_i = (0, \cdots, 1, \cdots, 0)'|y_i, \mathbf{x}_i, \Psi^{(k)}) \log p(y_i, \mathbf{z}_i = (0, \cdots, 1, \cdots, 0)'|\mathbf{x}_i, \Psi)\right] && (6)\\
&= \sum_{i=1}^n \sum_{j=1}^m \left[p(z_{ij} = 1|y_i, \mathbf{x}_i, \Psi^{(k)}) \log p(y_i, \mathbf{z}_i = (0, \cdots, 1, \cdots, 0)'|\mathbf{x}_i, \Psi)\right] && (7)\\
&= \sum_{i=1}^n \sum_{j=1}^m \left[E(z_{ij}|y_i, \mathbf{x}_i, \Psi^{(k)})\{\log \pi_j + \log \varphi(y_i - \mathbf{x}_i^T \boldsymbol{\beta}_j, 0, \sigma^2)\}\right] && (8)\\
&= \sum_{i=1}^n \sum_{j=1}^m p_{ij}^{(k)}\{\log \pi_j + \log \varphi(y_i - \mathbf{x}_i^T \boldsymbol{\beta}_j, 0, \sigma^2)\} && (9)
\end{aligned}
$$

where

$$\mathbf{y} = (y_1, \cdots, y_n)'$$

$$
Z = \begin{pmatrix} \mathbf{z}_1' \\ \vdots \\ \mathbf{z}_n' \end{pmatrix} = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nm} \end{pmatrix} \quad X = \begin{pmatrix} \mathbf{x}_1' \\ \vdots \\ \mathbf{x}_n' \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}
$$

$$
p_{ij}^{(k)} = E(z_{ij}|y_i, \mathbf{x}_i, \Psi^{(k)}) = \frac{\pi_j^{(k)} \varphi(y_i - \mathbf{x}_i^T \boldsymbol{\beta}_j^{(k)}, 0, \sigma^{2^{(k)}})}{\sum_{j=1}^m \pi_j^{(k)} \varphi(y_i - \mathbf{x}_i^T \boldsymbol{\beta}_j^{(k)}, 0, \sigma^{2^{(k)}})}
$$

The elaboration of the above steps are

- Step1→Step2: Use independence among $(y_i, \mathbf{z}_i)$

[*] bijuwang@uconn.edu

- Step3→Step4: Marginal density of $\mathbf{z}_i$
- Step4→Step5: Use the fact $\mathbf{z}_i \perp (y_1, \cdots, y_{i-1}, y_{i+1}, \cdots, y_n)|y_i$, we can get rid of $(y_1, \cdots, y_{i-1}, y_{i+1}, \cdots, y_n)$
- Step6→Step7: Easy to see conditional joint density is equal to condition marginal density

1.2 M-Step Derivation

Since we have

$$\begin{aligned}
Q(\Psi|\Psi^{(k)}) &= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)}\{\log \pi_j + \log \varphi(y_i - \mathbf{x}_i^T\boldsymbol{\beta}_j, 0, \sigma^2)\} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)} \log \pi_j - \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)} \log \sqrt{2\pi}\sigma - \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)} \frac{(y_i - \mathbf{x}_i^T\boldsymbol{\beta}_j)^2}{2\sigma^2} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)} \log \frac{\pi_j}{\sqrt{2\pi}} - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)} \log \sigma^2 - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)} \frac{(y_i - \mathbf{x}_i^T\boldsymbol{\beta}_j)^2}{\sigma^2} \\
&= I_1 - \frac{1}{2}I_2 - \frac{1}{2}I_3
\end{aligned}$$

From the above, we can see only I_3 contains $\boldsymbol{\beta}_j$ and

$$I_3 = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)} \frac{(y_i - \mathbf{x}_i^T\boldsymbol{\beta}_j)^2}{\sigma^2} = \sum_{j=1}^{m}\sum_{i=1}^{n} p_{ij}^{(k)} \frac{(y_i - \mathbf{x}_i^T\boldsymbol{\beta}_j)^2}{\sigma^2}$$

To minimize I_3, we only need to fix j and optimize with regard to $\boldsymbol{\beta}_j$. We can directly use the formula from generazied least square method and obtain

$$\boldsymbol{\beta}_j^{(k+1)} = (X'V^{-1}X)^{-1}X'V^{-1}\mathbf{y} = \left(\sum_{i=1}^{n} \mathbf{x}_i\mathbf{x}_i^T p_{ij}^{(k)}\right)^{-1} \left(\sum_{i=1}^{n} \mathbf{x}_i p_{ij}^{(k)} y_i\right) \quad j = 1, \cdots, m$$

where

$$V^{-1} = diag(p_{1j}^{(k)}, \cdots, p_{nj}^{(k)})$$

Only I_2 and I_3 contains σ^2, since

$$I_2 + I_3 = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)} \log \sigma^2 + \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)} \frac{(y_i - \mathbf{x}_i^T\boldsymbol{\beta}_j)^2}{\sigma^2}$$

We minimize it with regard to σ^2 given $\boldsymbol{\beta}_j = \boldsymbol{\beta}_j^{(k+1)}$ and obtain

$$\sigma^{2^{(k+1)}} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)}(y_i - \mathbf{x}_i^T\boldsymbol{\beta}_j^{(k+1)})^2}{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)}} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)}(y_i - \mathbf{x}_i^T\boldsymbol{\beta}_j^{(k+1)})^2}{n}$$

Only I_1 contains π_j and

$$I_1 = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)} \log \frac{\pi_j}{\sqrt{2\pi}} = -\frac{1}{2}\log(2\pi)\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k)} + \sum_{j=1}^{m}\left(\sum_{i=1}^{n} p_{ij}^{(k)}\right)\log \pi_j$$

In order to maximize I_1 under constraint $\pi_1 + \cdots \pi_m = 1$. We use Lagrange multiplier method

$$L(\pi_1, \cdots, \pi_m) = \sum_{j=1}^{m}\left(\sum_{i=1}^{n} p_{ij}^{(k)}\right)\log \pi_j - \lambda\left(\sum_{j=1}^{m} \pi_j - 1\right)$$

with λ a Lagrange multiplier. We can obtain

$$\pi_j^{(k+1)} = \frac{\sum_{i=1}^{n} p_{ij}^{(k)}}{\sum_{j=1}^{m}\sum_{i=1}^{n} p_{ij}^{(k)}} = \frac{\sum_{i=1}^{n} p_{ij}^{(k)}}{n} \quad j = 1, \cdots, m$$

2 A Function to Implement EM Algorithm

```
regmix_em <- function(y, xmat, pi.init, beta.init, sigma.init,
                      control = list(maxiter = 100, tol = .Machine$double.eps^0.2)){

  xmat <- as.matrix(xmat)

  n <- nrow(xmat)
  p <- ncol(xmat)
  m <- length(pi.init)

  pi <- pi.init
  beta <- beta.init
  sigma <- sigma.init

  maxiter <- control$maxiter
  tol <- control$tol
  conv <- 1

  P <- matrix(NA, nrow = n, ncol = m)
  beta.new <- matrix(NA, nrow = p, ncol = m)

  for (i in 1:maxiter) {
    for (j in 1:n) {
      P[j,] <- pi * dnorm(y[j] - xmat[j,] %*% beta, 0, sigma)/
        sum(pi * dnorm(y[j] - xmat[j,] %*% beta, 0, sigma))
    }

    pi.new <- apply(P, MARGIN = 2, mean)

    for (j in 1:m) {
      beta.new[,j] <- solve(t(xmat) %*% diag(P[,j]) %*% xmat) %*% t(xmat) %*% diag(P[,j]) %*% y
    }

    sigma.new <- sqrt(sum(P * (y %*% t(rep(1, m)) - xmat %*% beta.new)^2)/n)

    conv <- sum(abs(pi.new - pi)) + sum(abs(beta.new - beta)) + abs(sigma.new - sigma)
    if(conv < tol) break

    pi <- pi.new
    beta <- beta.new
    sigma <- sigma.new

  }

  if(i == maxiter)
  message("Reached the maximum iteration!")

  list(pi = pi.new, beta = beta.new, sigma = sigma.new, conv = conv, iter = i)

}
```

3 Data Generation and Parameters Estimation

After I carried out the following code, I found parameters won't be updated after the second iteration. Tracing back to E-Step Derivation, we can see if $\boldsymbol{\beta}_1 = \cdots = \boldsymbol{\beta}_m$, then $\mathbf{p}_{\cdot j}^{(k)}$ and $\pi_j^{(k)}$ will remain the same at all times.

```
regmix_sim <- function(n, pi, beta, sigma) {
    K <- ncol(beta)
    p <- NROW(beta)
    xmat <- matrix(rnorm(n * p), n, p) # normal covaraites
    error <- matrix(rnorm(n * K, sd = sigma), n, K)
    ymat <- xmat %*% beta + error # n by K matrix
    ind <- t(rmultinom(n, size = 1, prob = pi))
    y <- rowSums(ymat * ind)
    data.frame(y, xmat)
}

n <- 400
pi <- c(.3, .4, .3)
beta <- matrix(c( 1,  1,  1,
                 -1, -1, -1), 2, 3)
sigma <- 1
set.seed(1205)
dat <- regmix_sim(n, pi, beta, sigma)

fit <- regmix_em(y = dat[,1], xmat = dat[,-1],
          pi.init = pi / pi / length(pi),
          beta.init = beta * 0,
          sigma.init = sigma / sigma,
          control = list(maxiter = 500, tol = 1e-5))
fit
```

```
## $pi
## [1] 0.3333333 0.3333333 0.3333333
##
## $beta
##             [,1]       [,2]       [,3]
## [1,]  0.3335660  0.3335660  0.3335660
## [2,] -0.4754645 -0.4754645 -0.4754645
##
## $sigma
## [1] 1.732492
##
## $conv
## [1] 0
##
## $iter
## [1] 2
```

Thus we change the initial values of $\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_m$. And we can see this time after 83 iterations, the algorithm converged and I got the following consequences.

```
fit1 <- regmix_em(y = dat[,1], xmat = dat[,-1],
          pi.init = pi / pi / length(pi),
          beta.init = matrix(1:6, 2, 3),
          sigma.init = sigma / sigma,
```

```r
          control = list(maxiter = 500, tol = 1e-5))
fit1
```

```
## $pi
## [1] 0.3454017 0.3858262 0.2687721
##
## $beta
##            [,1]      [,2]       [,3]
## [1,] -0.9136801 0.8796636  0.9912061
## [2,] -1.1990374 0.9341887 -1.2424685
##
## $sigma
## [1] 1.023598
##
## $conv
## [1] 9.786183e-06
##
## $iter
## [1] 83
```