Finite mixture regression

Yuance He

10/12/2018

1. E-step

$$
\begin{aligned}
Q(\Psi|\Psi^{(k)}) &= E[l_n^c(\Psi)|(x,y)] \\
&= \sum_z P(z|x,y,\Psi^{(k)}) l_n^c(\Psi) \\
&= \sum_z P(z|x,y,\Psi^{(k)}) \sum_{i=1}^n \sum_{j=1}^m z_{ij}\{ln\pi_j + ln\varphi(y_i - x_i^T\beta_j; 0, \sigma^2)\} \\
&= \sum_{i=1}^n \sum_{j=1}^m [\sum_z z_{ij} P(z|(x,y),\Psi^{(k)})]\{ln\pi_j + ln\varphi(y_i - x_i^T\beta_j; 0, \sigma^2)\} \\
&= \sum_{i=1}^n \sum_{j=1}^m \{ln\pi_j + ln\varphi(y_i - x_i^T\beta_j; 0, \sigma^2)\} E(z_{ij}|(x,y),\Psi^{(k)})
\end{aligned}
$$

Since we have $p_{ij}^{(k+1)} = \sum_{z_{ij}} p(z_{ij}|x_i, y_i, \Psi^{(k)}) z_{ij} = E[z_{ij}|x_i, y_i, \Psi^{(k)}] = \frac{\pi_j^{(k)}\phi(y_i - x_i^T\beta_j^{(k)}; 0, \sigma^{2(k)})}{\sum_{j=1}^m \pi_j^{(k)}\phi(y_i - x_i^T\beta_j^{(k)}; 0, \sigma^{2(k)})}$,

$$
Q(\Psi|\Psi^{(k)}) = \sum_{i=1}^n \sum_{j=1}^m p_{ij}^{(k+1)}\{ln\pi_j + ln\varphi(y_i - x_i^T\beta_j; 0, \sigma^2)\}
$$

2. M-step

Now we need to maxmize $Q(\Psi|\Psi^{(k)})$.

$$
Q(\Psi|\Psi^{(k)}) = \sum_{i=1}^n \sum_{j=1}^m p_{ij}^{(k+1)}\{ln\pi_j + ln\varphi(y_i - x_i^T\beta_j; 0, \sigma^2)\}
$$

$$
Q(\Psi|\Psi^{(k)}) = \sum_{i=1}^n \sum_{j=1}^m p_{ij}^{(k+1)} ln\pi_j^{(k)} + \sum_{i=1}^n \sum_{j=1}^m p_{ij}^{(k+1)} ln(\frac{1}{\sigma\sqrt{2\pi}}) + \sum_{i=1}^n \sum_{j=1}^m p_{ij}^{(k+1)}(-\frac{(y_i - x_i^T\beta_j^k)^2}{2\sigma^2})
$$

(1) π_j^{k+1}

$$
\begin{aligned}
\frac{\partial Q}{\partial \pi_j} &= \frac{\partial}{\partial \pi_j} \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}^{(k+1)} \ln \pi_j \\
&= \ln \pi_1 \sum_{i=1}^{n} p_{i1}^{(k+1)} + ... + \ln \pi_j \sum_{i=1}^{n} p_{ij}^{(k+1)} + ... \ln \pi_m \sum_{i=1}^{n} p_{im}^{(k+1)} \\
&= \frac{\sum_{i=1}^{n} p_{ij}}{\pi_j} - \frac{\sum_{i=1}^{n} p_{im}}{\pi_m} \\
&= 0
\end{aligned}
$$

Let $\frac{\sum_{i=1}^{n} p_{ij}^{k+1}}{\pi_j} = n$, then $\pi_j^{k+1} = \frac{\sum_{i=1}^{n} p_{ij}^{k+1}}{n}$

(2) β_j^{k+1}

$$
\begin{aligned}
\frac{\partial Q}{\partial \beta_j} &= \frac{\partial}{\partial \beta_j} \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}^{k+1} \frac{(y_i - x_i^T \beta_j)^2}{2\sigma^2} \\
&= -\frac{1}{2\sigma^2} \sum_{i=1}^{n} p_{ij}^{(k+1)} (-2)(y_i - x_i^T \beta_j^{(k+1)}) \frac{\partial (x_i^T \beta_j^{(k+1)})}{\partial \beta_j} \\
&= 0 \\
\beta_j^{(k+1)} &= (\sum_{i=1}^{n} x_i x_i^T p_{ij}^{(k+1)})^{-1} (\sum_{i=1}^{n} x_i p_{ij}^{(k+1)} y_i)
\end{aligned}
$$

(3) $\sigma^{2(k+1)}$

$$
\begin{aligned}
\frac{\partial Q}{\partial \sigma^{2(k+1)}} &= \frac{\partial}{\partial \sigma^{2(k+1)}} (\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}^{k+1} \log(2\sigma^2 \pi)) - (\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}^{k+1} \frac{(y_i - x_i^T \beta_j)^2}{2\sigma^2}) \\
&= 0 \\
n &= \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}^{(k+1)} (y_i - x_i^T \beta_j^{(k+1)})^2}{\sigma^{2(k+1)}} \\
\sigma^{2(k+1)} &= \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}^{k+1} (y_i - x_i^T \beta_j^{k})^2}{n}
\end{aligned}
$$

3. Implementation of the Algorithm

```r
regmix_em <- function(y, xmat, pi.init, beta.init, sigma.init,
                      control=list(maxit = 500, tol =1e-5)){
  r <- nrow(xmat)
  c <- ncol(xmat)
  l <- length(pi.init)
  max <- control$maxit
  xmat <- as.matrix(xmat)
  tol <- control$tol
  pi <- pi.init
  beta <- beta.init
  sigma <- sigma.init
  p <- matrix(0,r,l)
  beta.new=matrix(0,c,l)
  for (k in 1:max) {
    for (i in 1:n) {
      p[i,] <- pi * dnorm(y[i] - xmat[i,] %*% beta, mean=0, sigma)/
           sum(pi * dnorm(y[i] - xmat[i,] %*% beta, mean=0, sigma))
    }
    pi.new <- colMeans(p)

    for (j in 1:l) {
      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, l)) - xmat %*% beta.new)^2)/n)

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

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

  }

  if(k == max)
  print("Maximum Iteration")

  list(pi = pi.new, beta = beta.new, sigma = sigma.new, dv = dv, iter = k)
}
```

4. Generating Data

```r
regmix_sim <- function(n, pi, beta, sigma) {
    K <- ncol(beta)
```

```r
    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)
bet <- matrix(c( 1,  1,  1,
                -1, -1, -1), 2, 3)
sig <- 1
set.seed(1205)
dat <- regmix_sim(n, pi, bet, sig)
result<- regmix_em(y = dat[,1], xmat = dat[,-1],
             pi.init = pi / pi / length(pi),
             beta.init = bet * 0,
             sigma.init = sig / sig,
             control = list(maxit = 500, tol = 1e-5))
result
```

```
## $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
##
## $dv
## [1] 0
##
## $iter
## [1] 2
```