EM algorithm HW5

Yichu Li.

10/8/2018

Abstract

The project is about to deriving the updating rules for EM algorithm, and then implement it in r with given data and estimate the parameters.

4.8.1

Construction of an EM algorithm

For $P_{ij}^{(k+1)}$, based on the lecture notes, we have the following:

$$Q(\Psi|\Psi(k)) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(Z_{ij}=1|y_i,\Psi(k)) \times lnP(Z_{ij}=1,y_i|\Psi(k))$$

denote $p_{ij} = P(Z_{ij}=1|y_i,\Psi(k))$

$$p_{ij}^{(k+1)} = \frac{P(Z_{ij}=1,y_i|\Psi(k))}{P(y_i|\Psi(k))} = \frac{P(Z_{ij}=1,y_i|\Psi(k))}{\sum_{s=1}^{m} P(Z_{is}=1,y_i|\Psi(k))}$$

$$P(Z_{ij}=1,y_i|\Psi(k)) = P(Z_{ij}=1|\Psi(k)) \times P(y_i|Z_{ij}=1,\Psi(k)) \tag{1}$$

$$= \pi_j^{(k)}\phi(y_i - \vec{x}_i^{\,T}\vec{\beta}_j^{\,(k)};0,\sigma^2) \tag{2}$$

$$p_{ij}^{(k+1)} = \frac{\pi_j^{(k)}\phi(y_i - \vec{x}_i^{\,T}\vec{\beta}_j^{\,(k)};0,\sigma^{2k})}{\sum_{j=1}^{m}\pi_j^{(k)}\phi(y_i - \vec{x}_i^{\,T}\vec{\beta}_j^{\,(k)};0,\sigma^{2k})}$$

From (1), we could get

$$Q(\Psi|\Psi(k)) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} \times lnP(Z_{ij}=1,y_i|\Psi(k))$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} \times ln[\pi_j^{(k)}\phi(y_i - \vec{x}_i^{\,T}\vec{\beta}_j^{\,(k)};0,\sigma^{2k})]$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} ln\pi_j^{(k)} + \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} ln(\frac{1}{\sqrt{2\pi}\sigma}) + \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}(-\frac{(y_i - \vec{x}_i^{\,T}\vec{\beta}_j^{\,(k)})^2}{2\sigma^2})$$

Only the third part contains $\vec{\beta}_j^{\,(k)}$, so in order to maximize this part, we can minimize $\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}(y_i - \vec{x}_i^{\,T}\vec{\beta}_j^{\,(k)})^2$, from the property of the sample mean, $\vec{x}_i^{\,T}\vec{\beta}_j^{\,(k)}$ must be the mean of the weighted sample $y_i(i=1,2,\dots,n)$, each y_i has weight $p_{ij}^{(k+1)}$. So

$$\vec{x_i}\vec{x_i}^T\vec{\beta_j}^{(k+1)} = \frac{(\sum_{i=1}^{n}\vec{x_i}p_{ij}^{(k+1)}y_i)}{\sum_{i=1}^{n}p_{ij}^{(k+1)}}$$

$$\vec{\beta_j}^{(k+1)} = (\sum_{i=1}^{n}\vec{x_i}\vec{x_i}^T p_{ij}^{(k+1)})^{-1} \times (\sum_{i=1}^{n}\vec{x_i}p_{ij}^{(k+1)}y_i), y = 1, 2, \ldots, m$$

Recall that in $Q(\Psi|\Psi(k))$ only the 2nd and 3rd parts, which we call I_2, I_3 contain σ^2, and if we denote

$$I_2^{(\star)} = \sum_{i=1}^{n}\sum_{j=1}^{m}p_{ij}^{(k+1)}ln(\sigma^2)$$

$$I_3^{(\star)} = \sum_{i=1}^{n}\sum_{j=1}^{m}p_{ij}^{(k+1)}(y_i - \vec{x_i}^T\vec{\beta_j}^{(k)})^2/\sigma^2$$

then $I_2^{(\star)} + I_3^{(\star)}$ is the sum of m terms of the following form,

$$S_j = \sum_{i=1}^{n}p_{ij}^{(k+1)}ln(\sigma^2) + \sum_{i=1}^{n}p_{ij}^{(k+1)}(y_i - \vec{x_i}^T\vec{\beta_j}^{(k)})^2/\sigma^2$$

Thus we only need to find σ^2 to minimize each S_j. Now that $\vec{x_i}^T\vec{\beta_j}^{(k+1)}$ is equal to the weighted mean of y_i, to minimize S_j, also from the property of sample variance, σ^2 must be the sample variance of the weighted sample $y_1, y_2, \ldots, y_n$. Therefore,

$$\sigma_j^{2(k+1)} = \frac{\sum_{i=1}^{n}p_{ij}^{(k+1)}(y_i - \vec{x_i}^T\vec{\beta_j}^{(k)})^2}{\sum_{i=1}^{n}p_{ij}^{(k+1)}}$$

According to the given condition,

$$\sigma_1^{2(k+1)} = \sigma_2^{2(k+1)} = \cdots = \sigma_m^{2(k+1)} = \sigma^{2(k+1)}$$

$$\sigma^{2(k+1)}\sum_{i=1}^{n}p_{ij}^{(k+1)} = \sum_{i=1}^{n}p_{ij}^{(k+1)}(y_i - \vec{x_i}^T\vec{\beta_j}^{(k)})^2 = \sigma^{2(k+1)} = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n}p_{ij}^{(k+1)}(y_i - \vec{x_i}^T\vec{\beta_j}^{(k+1)})^2}{n}$$

Finally, as $\sum_{j=1}^{m}\pi_j = 1$, we can construct the Lagrangian Function:

$$L(\pi_1^{(k)}, \ldots, \pi_m^{(k)}; \lambda) = Q(\Psi|\Psi(k)) - \lambda(\sum_{j=1}^{m}\pi_j^{(k)} - 1)$$

Set $\frac{\partial L}{\partial \pi_j^k} = 0$,\$(j=1, 2, . . . , m), we have

$$\sum_{i=1}^{n} p_{ij}^{(k+1)} \frac{1}{\pi_j^(k+1)} - \lambda = 0, (j = 1, 2, \ldots, m) \tag{3}$$

$$\pi_j = \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)}}{\lambda} \tag{4}$$

$$\sum_{j=1}^{m} \pi_j = \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)}}{\lambda} = \frac{n}{\lambda} = 1 \tag{5}$$

$$\tag{6}$$

Thus we have $\pi_j^{(k+1)} = \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)}}{n}$.

Implement of EM algorithm

```
regmix_em <- function(y, xmat, pi.init, beta.init, sigma.init,
  control=list(max = 100, tol = .Machine$double.eps^0.1)) {
  max <- control$max
  xmat <- as.matrix(xmat)
  tol <- control$tol
  nr <- nrow(xmat)
  nc <- ncol(xmat)
  m <- length(pi.init)
  pi <- pi.init
  beta <- beta.init
  sigma <- sigma.init
  p <- matrix(NA, nrow = nr, ncol = m)
  beta.new <- matrix(NA, nrow = nc, ncol = m)
  converg <- 1
for (i in 1:max) {
for (j in 1:nr) {
p[j, ] <- pi * dnorm(y[j] - xmat[j, ] %*% beta, mean = 0, sd = sigma) /
sum(pi * dnorm(y[j] - xmat[j, ] %*% beta, mean = 0, sd = sigma))
}
pi.new <- colMeans(p)
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)
if (sum(abs(pi-pi.new))+sum(abs(beta-beta.new))+abs(sigma-sigma.new) < tol) {
break
}
pi <- pi.new
beta <- beta.new
sigma <- sigma.new
```

```
}
if (i == max) {
print("maximum iteration")
}
return(list(pi = pi.new, beta = beta.new, sigma = sigma.new,
converg = converg, iteration = i))
}
```

```
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)
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 = matrix(c(1,1,2,2,3,3), 2, 3),
                 sigma.init = sig / sig,
                 control = list(max = 500, tol = 1e-5))
result
```

```
## $pi
## [1] 0.3454012 0.3858259 0.2687728
##
## $beta
##            [,1]      [,2]       [,3]
## [1,] -0.9136809 0.8796635  0.9912052
## [2,] -1.1990374 0.9341891 -1.2424680
##
## $sigma
## [1] 1.023598
##
## $converg
## [1] 1
##
## $iteration
## [1] 78
```