Homework 5

Xiaokang Liu

16 October 2018

Contents

1 Finite mixture regression

1.1 Follow the lecture notes to verify the validity of the provided E-step and M-step.

Based on the lecture notes, we have

$$Q(\Psi|\Psi^{k+1}) = E(l_n^c(\Psi)|x, y, \Psi^{k+1})$$

Since the complete log-likelihood can be written as

$$l_n^c(\Psi) = \sum_{i=1}^{n}\sum_{j=1}^{m} z_{ij}\log\{\pi_j\phi(y_i - x_i'\beta_j; 0, \sigma^2\},$$

and notice that, only z_{ij} is the unknown part in the complete data, the expectation with respect to the complete data will affect z_{ij} only. So we have

$$\begin{aligned} Q(\Psi|\Psi^{k+1}) &= \sum_{i=1}^{n}\sum_{j=1}^{m} E(z_{ij}|x_i, y_i, \Psi^k)\log\{\pi_j\phi(y_i - x_i'\beta_j; 0, \sigma^2\} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}\log\{\pi_j\phi(y_i - x_i'\beta_j; 0, \sigma^2\}. \end{aligned}$$

Then we consider $p_{ij}^{(k+1)}$, since z_{ij} only takes two values, 1 and 0, its conditional expectation is just the conditional probability of taking value 1. Then based on Bayes rule, we have

$$\begin{aligned} p_{ij}^{(k+1)} &= E(x_{ij}|x_i, y_i, \Psi^k) \\ &= p(x_{ij}|x_i, y_i, \Psi^k) \\ &= \frac{\pi_j^{(k)}\phi(y_i - x_i'\beta_j; 0, \sigma^{2(k)})}{\sum_{j=1}^{m}\pi_j^{(k)}\phi(y_i - x_i'\beta_j; 0, \sigma^{2(k)})}. \end{aligned}$$

Then by taking first order derivative of funtion $Q(\Psi|\Psi^{k+1})$ with respect to $\{\pi_j\}$, $\beta_j, j = 1, \ldots, m$ and σ^2 seperately, and set them to be 0, under $\sum_{j=1}^{m} \pi_j = 1$ we can solve the equations to get the MLE. Based on the form of $Q(\Psi|\Psi^{k+1})$ and the density function of normal distribution, we have

$$Q(\Psi|\Psi^{k+1}) = \sum_i \sum_j p_{ij}^{(k+1)} \log \pi_j - \frac{1}{2} \sum_i \sum_j p_{ij}^{(k+1)} \ln \sigma^2 - \frac{1}{2\sigma^2} \sum_i \sum_j p_{ij}^{(k+1)} (y_i - x_i' \beta_j)^2 + C,$$

where C is a constant. For the estimation of π_j's, since we have the constraint $\sum_j \pi_j = 1$, we can use lagrange multiplier method to solve it. Moreover, only the first term of Q function contains π_j's, it suffices to maximize $\sum_i \sum_j p_{ij}^{(k+1)} \log \pi_j$ under the linear constant on π_j's, i.e., we want to solve

$$\sum_j \sum_i p_{ij}^{(k+1)} \log \pi_j - \lambda(\sum_j \pi_j - 1),$$

where λ is the Lagrange multiplier. We take the first derivative to the above function with respect to each π_j and λ, respectively. We have

$$\sum_j \pi_j = 1 \text{ also } \frac{\sum_i p_{ij}^{(k+1)}}{\pi_j} - \lambda = 0,\ \ j = 1, \ldots, m.$$

Then we have $\lambda = \sum_i \sum_j p_{ij}^{(k+1)} = n$, and $\pi_j = \sum_i p_{ij}^{(k+1)}/\lambda = \sum_i p_{ij}^{(k+1)}/n$.

Then we consider the other parameters. Take first derivative to Q with respect to β_j and set it to 0, we have

$$\sum_i p_{ij}^{(k+1)} (x_i y_i - x_i x_i' \beta_j) = 0,$$

thus we have $\beta_j^{(k+1)} = (\sum_{i=1}^{n} x_i x_i' p_{ij}^{(k+1)})^{-1} \sum_{i=1}^{n} x_i p_{ij}^{(k+1)} y_i, j = 1, \ldots, m$. As for σ^2, we have

$$\frac{\partial Q}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2(\sigma^2)^2} \sum_i \sum_j p_{ij}^{(k+1)} (y_i - x_i' \beta_j)^2 = 0,$$

so we have $\sigma^{2^{(k+1)}} = \frac{\sum_{j=1}^{m} \sum_{i=1}^{n} p_{ij}^{(k+1)} (y_i - x_i' \beta_j^{(k+1)})^2}{n}$. In summary, we have

$$\begin{aligned}
\pi_j^{(k+1)} &= \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)}}{n} \\
\beta_j^{(k+1)} &= (\sum_{i=1}^{n} x_i x_i' p_{ij}^{(k+1)})^{-1} \sum_{i=1}^{n} x_i p_{ij}^{(k+1)} y_i, j = 1, \ldots, m; \\
\sigma^{2^{(k+1)}} &= \frac{\sum_{j=1}^{m} \sum_{i=1}^{n} p_{ij}^{(k+1)} (y_i - x_i' \beta_j^{(k+1)})^2}{n}.
\end{aligned}$$

For each time's iteration, we at first update $p_{ij}^{(k+1)}$ based on $\Psi^{(k)}$, then use it to get $\Psi^{(k+1)}$ and compute the distance between $\Psi^{(k)}$ and $\Psi^{(k+1)}$. This procedure will continue until the distance is less than a pre-specified convergence tolerance or the iteration number attain the specified maximum iteration number.

1.2 Implement this algorithm in R

```
regmix_em <- function(y,xmat,pi.init,beta.init,sigma.init,
                      control=list(max.ite,con.tol)){
  n <- length(y)
  p <- ncol(xmat)
  k <- ncol(beta.init)
  err <- 100
  ite <- 0
  conver <- 0
  max.ite <- control[[1]]
  con.tol <- control[[2]]
  xmat <- as.matrix(xmat)
  while ((err > con.tol)&(ite < max.ite)) {
    p.mat <- matrix(nrow = n, ncol = k)
    for (i in 1:n){
      for (j in 1:k){
        p.mat[i,j] <- pi.init[j]*dnorm(y[i]-t(xmat[i,])%*%beta.init[,j],0,sigma.init)/sum(pi.in
      }
    }
    pi.ite <- apply(p.mat,2,mean)
    beta.ite <- matrix(nrow = p, ncol = k)
    for (j in 1:k){
      upp <- 0
      low <- 0
      for (i in 1:n){
        upp <- upp+xmat[i,]%*%t(xmat[i,])*p.mat[i,j]
        low <- low+xmat[i,]*y[i]*p.mat[i,j]
      }
      beta.ite[,j] <- solve(upp)%*%low
    }
    sigma2.ite <- 0
    for (j in 1:k){
      sigma2.ite <- sigma2.ite+sum((p.mat[,j]*(y-xmat%*%beta.ite[,j])^2))
    }
    sigma.ite <- sqrt(sigma2.ite/n)
    #err <- sqrt(sum((pi.ite-pi.init)^2)+sum((beta.ite-beta.init)^2)+
    #            sum((sigma.ite-sigma.init)^2))
    err <- sum(abs(pi.ite-pi.init))+sum(abs(beta.ite-beta.init))+
                sum(abs(sigma.ite-sigma.init))
    pi.init <- pi.ite
    beta.init <- beta.ite
    sigma.init <- sigma.ite
    ite <- ite+1
  }
  if (ite >= max.ite) conver <- 1
  return(list(pi.est=pi.init, beta.est=beta.init,
              sigma.est=sigma.init, converge=conver))
```

```
}
```

1.3 Generate data to test the algorithm

```
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)
regmix_em(y = dat[,1], xmat = dat[,-1],
          pi.init = pi / pi / length(pi),
          #beta.init = matrix(c( 1,  2,  3,
          #      -1, -1, -1), 2, 3),
          beta.init = bet * 0,
          sigma.init = sig / sig,
          control = list(max.ite = 500, con.tol = 1e-5))
```

```
## $pi.est
## [1] 0.3333333 0.3333333 0.3333333
##
## $beta.est
##            [,1]       [,2]       [,3]
## [1,]  0.3335660  0.3335660  0.3335660
## [2,] -0.4754645 -0.4754645 -0.4754645
##
## $sigma.est
## [1] 1.732492
##
## $converge
## [1] 0
```