Statistical Computing Homework 5

Jieying Jiao

05 October 2018

Abstract

This is Jieying Jiao's homework 5 for statistical computing, fall 2018.

Contents

1 Exercise 4.8.1

1.1 Verify E- and M- step formula

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

Since z_{ij} is binary random variable, so we have:

$$\begin{aligned}
E(z_{ij}|(x,y),\Psi^{(k)}) &= P(z_{ij}=1|(x,y),\Psi^{(k)}) \\
&= P(z_{ij}=1|(x_i,y_i),\Psi^{(k)}) \\
&= \frac{P(x_i,y_i,z_{ij}=1|\Psi^{(k)})}{P(x_i,y_i|\Psi^{(k)})} \\
&= \frac{\pi_j^{(k)}\varphi(y_i - x_i^T\beta_j;0,\sigma^2)}{\sum_{j=1}^m \pi_j^{(k)}\varphi(y_i - x_i^T\beta_j;0,\sigma^2)} \\
&= p_{ij}^{(k+1)}
\end{aligned}$$

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

with $p_{ij}^{(k+1)}$ defined above, and it's all known at the k-th iteration step.

Next we need to maximum $Q(\Psi|\Psi^{(k)})$. As we can easily observe:

$$\sum_{j=1}^{m} p_{ij}^{(k+1)} = 1, \sum_{j=1}^{m} \pi_j = 1$$

$$\begin{aligned}
Q(\Psi|\Psi^{(k)}) &= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}\{log\pi_j + log\varphi(y_i - x_i^T\beta_j; 0, \sigma^2)\} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} log\pi_j + (-\frac{1}{2})\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} log2\pi\sigma^2 + (-\frac{1}{2})\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} \frac{(y_i - x_i\beta_j)^2}{\sigma^2} \\
&= I_1 + I_2 + I_3
\end{aligned}$$

(1) For $\pi_j^{(k+1)}$:

$$\begin{aligned}
\frac{\partial}{\partial \pi_j} Q(\Psi|\Psi^{(k)}) &= \frac{\partial I_1}{\partial \pi_j} \\
&= \frac{\partial}{\partial \pi_j}\{\sum_{j=1}^{m-1} log\pi_j \sum_{i=1}^{n} p_{ij}^{(k+1)} + log(1 - \pi_1 - ... - \pi_{m-1})\sum_{i=1}^{n} p_{im}^{(k+1)}\} \\
&= \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)}}{\pi_j} - \frac{\sum_{i=1}^{n} p_{im}^{(k+1)}}{\pi_m} \\
&= 0
\end{aligned}$$

$$\Rightarrow \frac{\sum_{i=1}^n p_{ij}^{(k+1)}}{\pi_j} = \frac{\sum_{i=1}^n p_{im}^{(k+1)}}{\pi_m} = c$$

$$\Rightarrow \sum_{i=1}^n p_{ij}^{(k+1)} = c\pi_j$$

$$\Rightarrow \sum_{j=1} \sum_{i=1}^n p_{ij}^{(k+1)} = c \sum_{j=1}^m \pi_j$$

$$\Rightarrow \sum_{i=1}^n 1 = c$$

$$\Rightarrow c = n$$

$$\Rightarrow \pi_j^{(k+1)} = \frac{\sum_{i=1}^n p_{ij}^{(k+1)}}{n}, j = 1, 2, ..., m-1$$

$$\Rightarrow \pi_m^{(k+1)} = 1 - \sum_{j=1}^{m-1} \pi_j^{(k+1)} = \frac{\sum_{i=1}^n \sum_{j=1}^m p_{ij}^{(k+1)} - \sum_{j=1}^{m-1} \sum_{i=1}^n p_{ij}^{(k+1)}}{n} = \frac{\sum_{i=1}^n p_{im}^{(k+1)}}{n}$$

(2) For $\beta_j^{(k+1)}$:

$$\begin{aligned} \frac{\partial}{\partial \beta_j} Q(\Psi|\Psi^{(k)}) &= \frac{\partial I_3}{\partial \beta_j} \\ &= -2 \sum_{i=1}^n p_{ij}^{(k+1)} x_i (y_i - x_i^T \beta_j) \\ &= 0 \end{aligned}$$

$$\Rightarrow \sum_{i=1}^n p_{ij}^{(k+1)} (x_i y_i - x_i x_i^T \beta_j^{(k+1)}) = 0$$

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

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

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

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

It's easy to verify that the second derivitives are negative, so the above results indeed maximum the conditional expectation.

1.2 EM algorithm function in R

```
regmix_em <- function(y, xmat, pi.init, beta.init, sigma.init,
                      control=list(maxit = 100, tol = .Machine$double.eps^0.5)) {
  maxit <- control$maxit
  tol <- control$tol
  n <- nrow(xmat)
  m <- length(pi.init)
  pi <- pi.init
  beta <- beta.init
  sigma <- sigma.init
  p <- matrix(0, nrow = n, ncol = m)
  beta.new <- matrix(0, nrow = ncol(xmat), ncol = m)
  xmat <- as.matrix(xmat)
  convergence <- 0

  for (i in 1:maxit) {
    for (obs in 1:n) {
      p[obs, ] <- pi * dnorm(y[obs] - xmat[obs, ] %*% beta, mean = 0, sd = sigma) /
        sum(pi * dnorm(y[obs] - xmat[obs, ] %*% 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 == maxit) {
    print("reach maximum iteration number without convergence")
    convergence <- 1
  }
  return(list(pi = pi.new, beta = beta.new, sigma = sigma.new,
              convergence = convergence, iteration.number = i))
}
```

1.3 generate data and estimate parameters

```
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)

fit <- regmix_em(y = dat[,1], xmat = dat[,-1], pi.init = pi / pi / length(pi),
                 beta.init = matrix(c(1, 2, 3, 0, 0, 0), 2, 3),
                 sigma.init = sig / sig,
                 control = list(maxit = 500, tol = 1e-5))
fit$convergence
```

```
## [1] 0
```

```
fit$iteration.number
```

```
## [1] 59
```

```
fit$pi
```

```
## [1] 0.3858260 0.2687727 0.3454014
```

```
fit$beta
```

```
##            [,1]       [,2]       [,3]
```

```
## [1,] 0.8796635  0.9912055 -0.9136807
## [2,] 0.9341890 -1.2424681 -1.1990374
```

```
fit$sigma
```

```
## [1] 1.023598
```

The fitted value are shown above. Also it's shown that the algorithm converged successfully after 59 iterations. As we can observe, if the initial value for β is same for every group, then the value won't get updated in each iteration. So I changed the initial value instead of using all 0's.