Homework 5

Travis Nestor

4.8.1.1) Verify the E & M steps

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

E-step: take conditional expetation of $l_n^c(\Psi)$ Note: Let $Y = (x_i, z_i, y_i)$ where z_i is the missing data

Let

$$Q(\Psi; \Psi^{(k)}) = E[l_n^c(\Psi)] = E[log(L_n^c(\Psi; Y); x_i, y_i, \Psi^{(k)}]$$

$$= \sum p(z_i; x_i, y_i, \Psi^{(k)}) ln(p(x, y, z; \Psi))$$

Reduces as Z is discrete

$$\Rightarrow Q(\Psi, \Psi^{(k)}) = \sum p(z_i; z_i, y_i, \Psi^{(k)}) logp(x, y, z; \Psi)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{m} p(z_i = k; y_i, x_i, \Psi^{(k)}) logp(z_i = k, y_i, x_i; \Psi)$$

Use Bayes Theorem to replace $p(z_i = k; y_i, x_i, \Psi^{(k)})$ Let $P_{ij}^{k+1} = p(z_i = k; y_i, x_i, \Psi^{(k)})$

$$= p(z_i = k, y_i, x_i; \Psi^{(k)}) / p(y_i, x_i; \Psi^{(k)})$$

$$= p(z_i = k, y_i, x_i, ; \Psi^{(k)}) / \sum_{s=1}^{m} p(z_i = s, x_i, y_i; \Psi^{(k)})$$

and

$$p(z_i = k, y_i, x_i, ; \Psi^{(k)}) = \pi_j^{(k)}$$
$$\phi(y_i - x_i^T\beta_j^{(k)}; 0, \sigma^{2(k)}\}$$

$$\Rightarrow P_{ij}^{k+1} = \frac{(\pi_j^{(k)}\phi(y_i - x_i^T\beta_j^{(k)}; 0, \sigma^{2(k)})}{\sum_{j=1}^{m}\phi(y_i - x_i^T\beta_j^{(k)}; 0, \sigma^{2(k)})}$$

Therefore,

$$Q(\Psi; \Psi^{(k)}) = \sum_{i=1}^{n}\sum_{j=1}^{m} P_{ij}^{k+1} logp(z_i = k, y_i, x_i; \Psi)$$

Reduce further by the same process as above:

$$logp(z_i = k, y_i, x_i; \Psi) = log\{\pi_j \phi(y_i - x_i^T\beta_j; 0, \sigma^2)\}$$

By logarithm properties:

$$= \log \pi_j + log\phi(y_i - x_i^T\beta_j; 0, \sigma^2)$$

M-step, Maximize $Q(\Psi;\Psi^{(k)})$

Maximazation of $\pi_j^{(k)}$ Since $\sum_{j=1}^{n}\pi_j^{(k)}=1$ then

$$\delta\mathcal{L}(\pi_1,...,\pi_j)/\delta\pi_j=0$$

where

$$\mathcal{L}(\pi_1,...,\pi_j)=\sum_{j=1}^{k}\sum_{i=1}^{(k)}log(\pi_j)-\lambda\{\sum_{j=1}^{(k)}\pi_j-1\}$$

with λ a Lagrange multiplier.

$$\Rightarrow\pi_j=(P_{ij}^{k+1})/\sum_{j=1}^{(k+1)}P_{ij}^{(k+1)}$$

Since for each j,

$$\sum_{j=1}^{(k+1)}P_{ij}^{(k+1)}=1$$

,

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

Calcuation of $\beta_j^{(k+1)}$

By properties of sample mean $\mu=\beta_j^{(k+1)}*x_i^T$, must be the mean of a weighted sample, Therefore:

$$\beta_j^{(k+1)}x_i^T=(\sum_{i=1}^{n}P_{ij}^{(k+1)}y_i)/(\sum_{i=1}^{n}P_{ij}^{(k+1)})$$

Divide both sides by x_i^T and multiply the right hand side by x_i and $(x_i)^{-1}$ (as $x_i*x_i^{-1}=1$)

$$\Rightarrow\beta_j^{(k+1)}=(\sum_{i=1}^{n}x_ix_i^TP_{ij}^{(k+1)})^{-1}(\sum_{i=1}^{n}x_iy_iP_{ij}^{(k+1)})$$

Similarly, $\sigma^{2(k+1)}$ is the sample variance of the weighted sample

$$\Rightarrow\sigma^{2(k+1)}=(\sum_{i=1}^{n}P_{ij}^{(k+1)}(y_i-\beta_j^{(k+1)}x_i^T)(y_i-\beta_j^{(k+1)}*x_i^T)')/(\sum_{i=1}^{n}P_{ij}^{(k+1)})$$

Use same equivalency we used to calculate $\pi_j^{(k+1)}$ where:

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

So by taking the summation the top and bottom of the right hand side and replacing with $\pi_j^{(k+1)}$, we get:

$$\sum_{j=1}^{m}(y_i-\beta_j^{(k+1)}x_i^T)(y_i-\beta_j^{(k+1)}*x_i^T)')$$

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

4.8.1.2)

```
regmix_em <- function(y, xmat, pi.0, beta.0, sigma.0, control){
                  control = list(maxit = 500, tol = 1e-5)
    xmat  <- as.matrix(xmat)
    k     <- length(pi.0)
    p     <- ncol(xmat)
    n     <- nrow(xmat)
    pi    <- pi.0
    beta  <- beta.0
    sigma <- sigma.0
    maxit <- control$maxit

    em.mat <- matrix(data = NA, nrow = n, ncol = k)
    beta.1 <- matrix(data = NA, nrow = p, ncol = k)



    for (i in 1:maxit)    {
      for (j in 1:n) {

        em.mat[j,] <- pi * dnorm(y[j] - xmat[j,] %*% beta, mean = 0, sigma) / sum(pi * dnorm(y[j] - xmat
      }

    pi.1 <- colMeans(em.mat)

    for (j in 1:k) {
      beta.1[,j] <- solve(t(xmat) %*% diag(em.mat[ ,j]) %*% xmat) %*% t(xmat) %*%           diag(em.mat
    }

    sigma.1 <- sqrt(sum(em.mat * (y %*% t(rep(1, k)) - xmat %*% beta.1) ^2) / n )

    conv <- sum(abs(pi.1 - pi)) + sum(abs(beta.1 - beta)) + abs(sigma.1 - sigma)
    if (conv < tol) break

    pi <- pi.1
    beta <- beta.1
    sigma <- sigma.1
}
  if (i == maxiter)
  print("reached maxiter")

  list(pi = pi.1, beta = beta.1, sigma = sigma.1, conv = conv, iter = i)

}
```

4.8.1.3)

```
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))
```

```r
    y <- rowSums(ymat * ind)
    data.frame(y, xmat)
}
```

Given initial data

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

```
## $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
```

By changing the beta.0 value from beta * 0 to beta * 1, we can get a more accurate approximation.

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

```
## $pi
## [1] 0.3858218 0.2687873 0.3453909
##
## $beta
##           [,1]       [,2]       [,3]
## [1,] 0.8796608  0.9911852 -0.9136977
## [2,] 0.9341964 -1.2424569 -1.1990372
##
## $sigma
```

```
## [1] 1.023598
##
## $conv
## [1] 8.597268e-06
##
## $iter
## [1] 49
```