SCHW6

John Rothen

10/9/2020

4.8.1

Part 1

$E[ln(L(t|Y)|x,t')] = E[sum_i(sum_j(z_{ij}log(pi_j c)))]$ where c is the normal density function of $y_i - x_i^T B_j; 0, s^2$. This is equal to $E[sum(p(z|x,t')ln(p(x,z|t)))]$, where $p(z|x,t') = E(z_i j|y_i x_i) = p(ij)/sum(p(ij)) = pi_j * c^k/sum(pi_j * c^k)$, which we will abbreviate as a. With this we can finish the E step with $Q = sum(sum(a * [log(pi_j) + log(c)]))$. The M-step is then to maximize this equation Q for pi_j^{k+1}, B_j^{k+1} and $s^{2(k+1)}$.

Part 2

```
regmix_em <- function(y,xmat,pi.init, beta.init, sigma.init, control){
  #note, iteration max was used as the loop breaker, as I ran into issues accurately calculating the Lo

  #set up the constants
  n <- 400
  m <- 3
  #initialize empty set for p
  p <- as.data.frame(cbind(rep(0,n),rep(0,n),rep(0,n)))
  #make sure everything is matrix multiplication ready
  x <- as.matrix(xmat)
  beta <- as.matrix(beta.init)
  #more constants
  maxit <- control$maxit
  tol <- control$tol
  iter=1

  #adjust for loop access
  pi.new <- pi.init
  b.new <- beta.init
  si <- sigma.init

  #prepare empty matrixes/sets for storing things mid-calculation
  b1 <- as.matrix(cbind(rep(0,n),rep(0,n),rep(0,n)))
  b2<- as.matrix(cbind(rep(0,n),rep(0,n),rep(0,n)))
  btemp <- rep(0,m)
  btemp2 <- rep(0,m)
  b.cool <- rep(0,m)
  temp2 <- as.matrix(cbind(rep(0,n),rep(0,n),rep(0,n)))
  temp3 <- rep(0,m)

#Actual loop process (while loop was giving me issues so I used repeat instead)
```

```
repeat{
  for (i in 1:n) {
    for(j in 1:m){
    p[i,j] <- pi.new[j]*dnorm(y[i]- x[i,]%*%b.new[,j],0,si^2)
    p[i,j] <- p[i,j]/finitesum(p[,j]) #find the new p values
    pi.new[j] <- finitesum(p[,j])/ n # calculate new pi values
    }}
  iter <- iter+1

  #set maxiter to 10 just for ease of use, this ofcourse would be maxit, but was too time consuming d
  if(iter==10){break}
}

#reset iter
iter=1

#loop to prepare betas
  repeat{
   for (i in 1:n) {
    for(j in 1:m){
    p[i,j] <- pi.new[j]*dnorm(y[i]- x[i,]%*%b.new[,j],0,si^2)
    p[i,j] <- p[i,j]/finitesum(p[,j])#the improved p
    b.new <-b.new[-1,] # beta's matrix dimensions confused me, so I created a process that creates th
    b1[i,j] <- (x[i,]%*%t(x)[,i]%*%p[i,j])
    btemp[j] <- finitesum(b1[,j])^-1
    b2[i,j]<- (sum(as.matrix(x[i,],2,1)%*% p[i,j] %*%y[i]))
    btemp2[j] <- finitesum(b2[,j])
    b.cool[j] <- btemp[j]%*%btemp2[j]
    b.new <- rbind(b.new, b.cool)
    row.names(b.new) <- c("Beta", "Beta")

    }}
    iter<- iter+1
  if(iter==10){break}}
iter=1
#sigma estimate
  repeat{
  for (i in 1:n) {
    for(j in 1:m){
    temp2[i,j] <-(p[i,j]*(y[i] -x[i,]%*%b.new[,j]))
    temp3[j] <- finitesum(temp2[,j])
    si <- finitesum(temp3)/n
    }
  }
  iter <- iter+1
  if(iter==10){break}}

#collect the results
templist <- list("pi estimates" = pi.new, "beta estimate" = b.new, "sigma" = si, "iteration" = iter )
 #print it!
return(templist)
}
```

Part 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))
    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(rnorm(6), 2, 3),
          sigma.init = sig / sig,
          control = list(maxit = 500, tol = 1e-5))
```

```
## $`pi estimates`
## [1] 0.0003973958 0.0004522488 0.0004622270
##
## $`beta estimate`
##           [,1]      [,2]       [,3]
## Beta 0.0331845 0.0335472 0.02996839
## Beta 0.0331845 0.0335472 0.02996856
##
## $sigma
## [1] -5.525753e-05
##
## $iteration
## [1] 10
```

4.8.2

Part 1

```
library(SQUAREM)
regmix_em1step <-function(y,xmat,pi.init, beta.init, sigma.init, control){
  n <- 400
  m <- 3
  p <- as.data.frame(cbind(rep(0,n),rep(0,n),rep(0,n)))
  x <- as.matrix(xmat)
  beta <- as.matrix(beta.init)
  q <- rep(0, control$maxit)
```

```
  q[1] = 0
  maxit <- control$maxit
  tol <- control$tol
  q[2]<- -1661.683
  iter=1
  pi.new <- pi.init
  b.new <- beta.init
  si <- sigma.init
  b1 <- as.matrix(cbind(rep(0,n),rep(0,n),rep(0,n)))
  b2<- as.matrix(cbind(rep(0,n),rep(0,n),rep(0,n)))
  btemp <- rep(0,m)
  btemp2 <- rep(0,m)
  b.cool <- rep(0,m)
  temp2 <- as.matrix(cbind(rep(0,n),rep(0,n),rep(0,n)))
  temp3 <- rep(0,m)
  #function is the same save for the repeat segments
  for (i in 1:n) {
      for(j in 1:m){
      p[i,j] <- pi.new[j]*dnorm(y[i]- x[i,]%*%b.new[,j],0,si^2)
      p[i,j] <- p[i,j]/finitesum(p[,j])
      pi.new[j] <- finitesum(p[,j])/ n
      }}
  for (i in 1:n) {
      for(j in 1:m){
      b.new <-b.new[-1,]
      b1[i,j] <- (x[i,]%*%t(x)[,i]%*%p[i,j])
      btemp[j] <- finitesum(b1[,j])^-1
      b2[i,j]<- (sum(as.matrix(x[i,],2,1)%*% p[i,j] %*%y[i]))
      btemp2[j] <- finitesum(b2[,j])
      b.cool[j] <- btemp[j]%*%btemp2[j]
      b.new <- rbind(b.new, b.cool)
      row.names(b.new) <- c("Beta", "Beta")
      }}
  for (i in 1:n) {
      for(j in 1:m){
      temp2[i,j] <-(p[i,j]*(y[i] -x[i,]%*%b.new[,j]))
      temp3[j] <- finitesum(temp2[,j])
      si <- finitesum(temp3)/n
      }
  }
   templist <- list("pi estimates" = pi.new, "beta estimate" = b.new, "sigma" = si)
   return(templist)
}
```

Part 2

```
regmix_em_new<- function(y,xmat,pi.init, beta.init, sigma.init, control){
    n <- 400
  m <- 3
  p <- as.data.frame(cbind(rep(0,n),rep(0,n),rep(0,n)))
  x <- as.matrix(xmat)
  beta <- as.matrix(beta.init)
  q <- rep(0, control$maxit)
```

```
  q[1] = 0
  maxit <- control$maxit
  tol <- control$tol
  q[2]<- -1661.683
  iter=1
  pi.new <- pi.init
  b.new <- beta.init
  si <- sigma.init
  b1 <- as.matrix(cbind(rep(0,n),rep(0,n),rep(0,n)))
  b2<- as.matrix(cbind(rep(0,n),rep(0,n),rep(0,n)))
  btemp <- rep(0,m)
  btemp2 <- rep(0,m)
  b.cool <- rep(0,m)
  temp2 <- as.matrix(cbind(rep(0,n),rep(0,n),rep(0,n)))
  temp3 <- rep(0,m)

  #simply repeats the single step function
    iter = 1
    pi.new <- pi.init
    maxit <- control$maxit
    repeat{
      regmix_em1step(y,xmat,pi.init, beta.init, sigma.init, control)
      iter = iter+1
      if(iter >= maxit){break}
    }
  templist <- list("pi estimates" = pi.new, "beta estimate" = b.new, "sigma" = si)
   return(templist)
}

#record start time
ptm <- proc.time()
regmix_em_new(y = dat[,1], xmat = dat[,-1],
          pi.init = pi / pi / length(pi),
          beta.init = matrix(rnorm(6), 2, 3),
          sigma.init = sig / sig,
          control = list(maxit = 500, tol = 1e-5))
```

```
## $`pi estimates`
## [1] 0.3333333 0.3333333 0.3333333
##
## $`beta estimate`
##             [,1]       [,2]       [,3]
## [1,] -0.04951689 -0.4742768 -0.4968567
## [2,] -0.59119700  1.7971784 -0.2408548
##
## $sigma
## [1] 1
```

```
#record time to compute
proc.time() - ptm
```

```
##    user  system elapsed
##   88.96    1.70   91.87
```

Part 3

```
zfunc <- function(i,j){
  z<-ifelse(i==j,1,0)
}

loglik.regmix <- function(p=p,pie=pi.new,y=dat[,1],x=dat[,-1],b=b.new,s=si){
  for (i in 1:n) {
      for(j in 1:m){
        dn[i,j] <- dnorm(y[i] -x[i,]%*%b[,j])
  l <-finitesum(zfunc(i,j)*log(pie[j])*dn[i,j])
}}}

#pf2 <- squarem(par=l, y=y, fixptfn=regmix_em1step, objfn=loglik.regmix, control=list(tol=tol))


#takes around 60 seconds, but is now giving an error
```

Part 4

The time to compute via the step-based algorithm was around ~90seconds on my home pc. This is considerably longer compared to the squarem package (~60 seconds or lower).