Homework 3

陈敬贤 15338013 统计学

May 5, 2018

1 Monte Carlo average of log-likelihood

The original form of complete data likelihood function as:

$$L(\Omega|Y_{i,j},U_i,Z_{1,i},Z_{2,i})=\prod_{i=1}^{n}\prod_{c=1}^{2}\left\{\pi_c f_c(Z_{c,i})\left[\prod_{j=1}^{T}f_c(Y_{i,j}|Z_{c,i})\right]\right\}^{w_{ic}}$$

where $f_c(Z_{c,i})$is the density function of Normal distribution, $f_c(Y_{i,j}|Z_{c,i})=P_{ij}^{Y_{ij}}(1-P_{ij})^{1-Y_{ij}}$
$w_{ic}$is the dummy variable of U_i, i.e:

$$w_{ic}=1$$

if subject i belong to cluster c;

$$w_{ic}=0$$

otherwise.

To set up the EM algorithm, consider the random effects, U and Z, to be the missing data. The complete data, Q, is then $Q=(Y,U,Z)$, and the complete-data log-likelihood using the Monte Carlo average to estimate is given by:

$$\begin{aligned}Q(\Omega,\Omega^{(m)})=&\frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{n}w_{i1}\left\{log(\pi_1)-\frac{1}{2}log(2\pi\sigma_1^2)-\frac{Z_{1,i}^{(k)}}{2\sigma_1^2}+\sum_{j=1}^{10}\left[Y_{ij}(\beta_1X_{1,ij}+Z_{1,i}^{(k)})-log\left[1+exp(\beta_1X_{1,ij}+Z_{1,i}^{(k)})\right]\right]\right\}\\&+\frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{n}(1-w_{i1})\left\{log(1-\pi_1)-\frac{1}{2}log(2\pi\sigma_2^2)-\frac{Z_{2,i}^{(k)}}{2\sigma_2^2}+\sum_{j=1}^{10}\left[Y_{ij}(\beta_2X_{2,ij}+Z_{2,i}^{(k)})-log\left[1+exp(\beta_2X_{2,ij}+Z_{2,i}^{(k)})\right]\right]\right\}\end{aligned}$$

2 Details of the MCEM algorithm steps:

Incorporating the Metropolis-Hastings step and Gibbs-sampling into the EM algorithm gives an MCEM algorithm as follows:

2.1 Initial Parameters

(1) Choose starting values $\Omega^{(0)}=\left\{\beta_1^{(0)}=0.9,\beta_2^{(0)}=4,\sigma_1^{(0)}=1.5,\sigma_2^{(0)}=9.5,\pi_1^{(0)}=0.5\right\}$. Set m=1.

2.2 Gibbs-sampling and Metropolis-Hastings Algorithms

(2) Using Gibbs-sampling to update (z, u), steps are as follows:

2.2.1 Generate Z (using Metropolis-Hastings algorithm)

a. Generate $Z^{(m)}$ from the conditional probability $P(Z|Y, \Omega^{(m-1)}, U^{(m-1)})$, in details:

$$P(Z|Y, \Omega^{(m-1)}, U^{(m-1)}) = \frac{P(Z|\Omega^{(m-1)}, U^{(m-1)}) * P(\Omega^{(m-1)}, U^{(m-1)}) * P(Y|Z, \Omega^{(m-1)}, U^{(m-1)})}{P(\Omega^{(m-1)}, U^{(m-1)}, Y)}$$

where

$$P(Z|\Omega^{(m-1)}, U^{(m-1)}) = dnorm(Z, 0, \sigma) \tag{1}$$

$$P(\Omega^{(m-1)}, U^{(m-1)}) = \pi^{(m-1)}(U) \tag{2}$$

$$\begin{aligned} P(Y|Z, \Omega^{(m-1)}, U^{(m-1)}) &= P_{ij}^{Y_{ij}}(1 - P_{ij})^{1-Y_{ij}} \\ &= \frac{exp^{Y_{ij}}(\beta_c X_{c,ij} + Z_{c,i})}{1 + exp(\beta_c X_{c,ij} + Z_{c,i})} \\ &\approx \frac{exp(\beta_c \bar{X}_{c,i} + Z_{c,i})}{1 + exp(\beta_c \bar{X}_{c,i} + Z_{c,i})} \end{aligned} \tag{3}$$

Remark:
Because the dimension of Z is $K * n$, which is not equal to the dimension of X and Y ($K * n * T$), in order to let their dimensions be equal, in the equation we replace $X_{c,ij}$ with $\bar{X}_{c,i}$ (the row mean of $X_{c,ij}$) and disregard Y_{ij} term.

Also, because the denominator of $P(Z|\Omega^{(m-1)}, U^{(m-1)})$ is hard to calculate, here we use **Metropolis-Hastings method** to generate Z:

1. Specify the proposal distribution $g(z)$ as the Normal distribution with mean $Z[i-1]$ and variance σ_c^2.
2. Then the accept probability is

$$\alpha(Z, Z^*) = min\left\{1, \frac{P(Z^*|\Omega^{(m-1)}, U^{(m-1)})}{P(Z|\Omega^{(m-1)}, U^{(m-1)})}\right\} \tag{4}$$

where Z^* is the new value generated.
and the second term in braces in (4) simplifies to

$$\begin{aligned} \frac{P(Z^*|\Omega^{(m-1)}, U^{(m-1)})}{P(Z|\Omega^{(m-1)}, U^{(m-1)})} &= \frac{P(Z^*|\Omega^{(m-1)}, U^{(m-1)}) * P(\Omega^{(m-1)}, U^{(m-1)}) * P(Y|Z^*, \Omega^{(m-1)}, U^{(m-1)})}{P(Z|\Omega^{(m-1)}, U^{(m-1)}) * P(\Omega^{(m-1)}, U^{(m-1)}) * P(Y|Z, \Omega^{(m-1)}, U^{(m-1)})} \\ &= \frac{dnorm(Z^*, 0, \sigma_c) * \frac{exp(\beta_c \bar{X}_{c,i} + Z^*_{c,i})}{1 + exp(\beta_c \bar{X}_{c,i} + Z^*_{c,i})}}{dnorm(Z, 0, \sigma_c) * \frac{exp(\beta_c \bar{X}_{c,i} + Z_{c,i})}{1 + exp(\beta_c \bar{X}_{c,i} + Z_{c,i})}} \end{aligned} \tag{5}$$

relevant R coding:

```
# F1 is the numerator of the simplified conditional probability of Z,
# which is used to compute the accept probability
F1 <- function(pai,sigma,z,x,beta){
  return(pai*dnorm(z,0,sigma)*(exp(beta*x+z)/(1+exp(beta*x+z))))
}

# MH is the function according to the Metropolis-Hastings algorithm
MH<-function(x,beta,sigma,pai){
     z<-numeric(1000)           # length of the chain
     z[1]<-rnorm(1,0,sigma)
     for(i in 2:1000){
       zt<-z[i-1]
       Z<-rnorm(1,zt,sigma)      # proposal distribution
       num<-F1(pai,sigma,Z,x,beta)
       den<-F1(pai,sigma,zt,x,beta)
       u<-runif(1)               # for accept/reject step
       ifelse(u<(num/den),z[i]<-Z,z[i]<-zt)
     }
     z1<-sample(z[-c(1:100)],1) # to randomly select a value from the chain
     return(z1)
  }


# generate #500*100 z from two chain
Z1 <- matrix(0,nrow = K,ncol=n)
Z2 <- matrix(0,nrow = K,ncol=n)

# to alter the dimension of X
Xmean <- apply(X, 1, mean)

for(k in 1:K){
 for(i in 1:n){

      Z1[k,i] <- MH(Xmean[i],beta1[m-1],sigma1[m-1],pu[m-1])
      Z2[k,i] <- MH(Xmean[i],beta2[m-1],sigma2[m-1],pu[m-1])


 }
}
```

2.2.2 Generate U (using Bayesian Formula)

b. Generate $U^{(m)}$ from the Bernoulli distribution with success probability $P(U=1|Y,Z^{(m)},\Omega^{(m-1)})$, here we use Bayesian Formula to change the form, in details:

$$
\begin{aligned}
P(U=1|Y,Z^{(m)},\Omega^{(m-1)}) &= \frac{P(Y|U=1,Z^{(m)},\Omega^{(m-1)}) * \pi(U)}{P(Y|U=1,Z^{(m)},\Omega^{(m-1)}) * \pi(U) + P(Y|U=2,Z^{(m)},\Omega^{(m-1)}) * (1-\pi(U))} \\
&= \frac{\frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{T} w_{i1} * P(Y_{ij}|U=1,Z_{1,i}^{(k,m)},\Omega^{(m-1)}) * \pi(U)}{\frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{T}(w_{i1}P(Y_{ij}|U=1,Z_{1,i}^{(k,m)},\Omega^{(m-1)})\pi(U) + w_{i2}P(Y_{ij}|U=2,Z_{2,i}^{(k,m)},\Omega^{(m-1)})(1-\pi(U)))}
\end{aligned} \tag{6}
$$

where

$\pi(U)$ here refers to the prior distribution of U, that is $P(U=1|Y,Z^{(m-1)},\Omega^{(m-2)})$;
$P(Y_{ij}|U=c,Z_{c,i}^{(k,m)},\Omega^{(m)}) = \frac{exp^{Y_{ij}}(\beta_c X_{c,ij}+Z_{c,i}^{(k,m)})}{1+exp(\beta_c X_{c,ij}+Z_{c,i}^{(k,m)})}$;
$w_{ic}=1$, if subject i belong to cluster c;
$w_{ic}=0$, otherwise.

relevant R coding:

```
#generate u(update pu)
num <- 0    # to record the conditional probability of Y with U=1
den <- 0    # to record the conditional probability of Y with U=2
for(k in 1:K){
 for(i in 1:n){
  for(j in 1:T){
      num <- num+exp(beta1[m-1]*X[i,j]+Z1[k,i])^Y[i,j]/
             (1+exp(beta1[m-1]*X[i,j]+Z1[k,i]))
      den <- den+exp(beta2[m-1]*X[i,j]+Z2[k,i])^Y[i,j]/
             (1+exp(beta2[m-1]*X[i,j]+Z2[k,i]))

  }
 }
}
Num <- num*pu[m-1]    # numerator of the result
Den <- num*pu[m-1]+den*(1-pu[m-1]) # denominator of the result
pu[m] <- Num/Den

# to generate u according to the updated pu
U <- sample(c(1,2),size = n,replace = T,prob = c(pu[m],1-pu[m]))
wi1[m,U==1] <- 1
```

3 Update Parameters & Accelerate Method

(3) After generate K values, $Z^{(1)}, Z^{(2)}, ..., Z^{(K)}$, and the corresponding U from $f(U,Z|Y,\Omega)$ using the Gibbs-sampling incorporating Metropolis-Hastings algorithm described previously:

1 Choose $\sigma_c^{(m)}$ and $\beta_c^{(m)}$ to maximize a Monte Carlo estimate $Q(\Omega,\Omega^{(m)})$
2 Set $m=m+1$.

3.1 Update σ_c

In this process, with respect to σ_c, we only need to calculate the derivative term $\frac{\partial Q(\Omega,\Omega^{(m)})}{\partial \sigma_c}$ then:

$\because$

$$\frac{\partial Q(\Omega,\Omega^{(m)})}{\partial \sigma_c^{(m+1)}} = 0$$

$\therefore$

$$\sigma_c^{(m+1)} = \sqrt{\frac{\frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{n} w_{ic} * Z_{c,i}^{2\ (m,k)}}{\sum_{i=1}^{n} w_{ic}}}$$

relevant R coding:

```
# update sigma 1 & 2
# num1 is the numerator of sigma1
# num2 is the numerator of sigma2
num1 <- num2 <- 0
for(k in 1:K){
  for(i in 1:n){
      num1 <- num1+wi1[m-1,i]*Z1[k,i]^2
      num2 <- num2+(1-wi1[m-1,i])*Z2[k,i]^2
  }
}
sigma1[m] <- sqrt(1/K*num1/sum(wi1[m-1,]))
sigma2[m] <- sqrt(1/K*num2/sum(1-wi1[m-1,]))
```

3.2 Update β_c & Newton-Raphson Method

As the first-order derivative term of β_c is as follows:

$$\frac{\partial Q(\Omega,\Omega^{(m)})}{\partial \beta_c} = \frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{n} w_{ic} * \sum_{j=1}^{T}(Y_{ij} * X_{c,ij} - X_{c,ij} * \frac{exp(\beta_c * X_{c,ij} + Z_{c,i}^{(k)})}{1 + exp(\beta_c * X_{c,ij} + Z_{c,i}^{(k)})}$$

We can not get the explicit solution of β_c from the equation above;

So, here we are going to implement the Newton-Raphson method to update β_c; To achieve our goal, we need to calculate the second-order derivative term of β_c, that is:

$$\frac{\partial^2 Q(\Omega,\Omega^{(m)})}{\partial \beta_c^2} = \frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{n} -w_{ic} * \sum_{j=1}^{T} \frac{X_{c,ij}^2 * exp(\beta_c * X_{c,ij} + Z_{c,i}^{(k)})}{(1 + exp(\beta_c * X_{c,ij} + Z_{c,i}^{(k)}))^2}$$

We expand $\frac{\partial Q(\Omega,\Omega^{(m)})}{\partial \beta_c}$ as a function of β_c around the value $\beta_c^{(m)}$ gives:

$$\frac{\partial Q(\Omega,\Omega^{(m)})}{\partial \beta_c^{(m+1)}} \cong \frac{\partial Q(\Omega,\Omega^{(m)})}{\partial \beta_c}|_{\beta_c=\beta_c^{(m)}} + \frac{\partial^2 Q(\Omega,\Omega^{(m)})}{\partial \beta_c^2}|_{\beta_c=\beta_c^{(m)}}(\beta_c^{(m+1)} - \beta_c^{(m)}) = 0$$

$\therefore$

$$\beta_c^{(m+1)} = \beta_c^{(m)} - \frac{\frac{\partial Q(\Omega,\Omega^{(m)})}{\partial \beta_c^{(m)}}}{\frac{\partial^2 Q(\Omega,\Omega^{(m)})}{\partial \beta_c^{(m)2}}}$$

relevant R coding:

```
#update beta 1&2
# s1 is the first-order derivative of beta1
# s2 is the second-order derivative of beta1
# s3 is the first-order derivative of beta2
# s4 is the second-order derivative of beta2
s1<-s2<-s3<-s4<-0
for(k in 1:K){
  for(i in 1:n){
    for(j in 1:T){
      s1 <- s1+wi1[m-1,i]*(Y[i,j]*X[i,j]-X[i,j]*exp(beta1[m-1]*X[i,j]+Z1[k,i])
            /(1+exp(beta1[m-1]*X[i,j]+Z1[k,i])))
      s2 <- s2-wi1[m-1,i]*(X[i,j]^2*exp(beta1[m-1]*X[i,j]+Z1[k,i])
            /(1+exp(beta1[m-1]*X[i,j]+Z1[k,i]))^2)
      s3 <- s3+(1-wi1[m-1,i])*(Y[i,j]*X[i,j]-X[i,j]*exp(beta2[m-1]*X[i,j]
+Z2[k,i])/(1+exp(beta2[m-1]*X[i,j]+Z2[k,i])))
      s4 <- s4+(wi1[m-1,i]-1)*(X[i,j]^2*exp(beta2[m-1]*X[i,j]+Z2[k,i])
            /(1+exp(beta2[m-1]*X[i,j]+Z2[k,i]))^2)
    }
  }
}
beta1[m] <- beta1[m-1]-s1/s2
beta2[m] <- beta2[m-1]-s3/s4
```

4 Conclusion & Convergence Rule

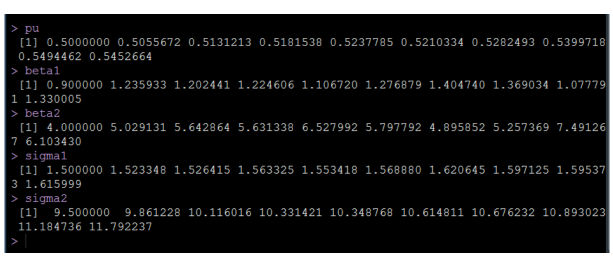
(4) If convergence is achieved, then declare $\beta_c^{(m)}$, $\sigma_c^{(m)}$, $\pi^{(m)}$ to be MLE's; otherwise, return to Step (2).

Here I define the convergence rule as $|\Omega^{(m)} - \Omega^{(m-1)}| \leq 0.1$

5 Computational Results by R

Because it will cost such a long time to set K=500 and iterate many times, here I obtain the result with K=50, iteration times=10:

5.1 result



```
> pu
 [1] 0.5000000 0.5055672 0.5131213 0.5181538 0.5237785 0.5210334 0.5282493 0.5399718
 0.5494462 0.5452664
> beta1
 [1] 0.900000 1.235933 1.202441 1.224606 1.106720 1.276879 1.404740 1.369034 1.07779
1 1.330005
> beta2
 [1] 4.000000 5.029131 5.642864 5.631338 6.527992 5.797792 4.895852 5.257369 7.49126
7 6.103430
> sigma1
 [1] 1.500000 1.523348 1.526415 1.563325 1.553418 1.568880 1.620645 1.597125 1.59537
3 1.615999
> sigma2
 [1]  9.500000  9.861228 10.116016 10.331421 10.348768 10.614811 10.676232 10.893023
 11.184736 11.792237
>
```

Figure 1: result

6 Attachment: R codes

```
# initial parameter
M<-200;T<-10;n<-100;
beta1<-beta2 <- pu <- sigma1 <- sigma2 <- numeric(M)
beta1[1]<-0.9;beta2[1]<-4;
pu[1]<-0.5;sigma1[1]<-1.5;sigma2[1]<-9.5;K<-50

# generate fixed effect X
X<-matrix(rnorm(1000,0,1),n,T)
X1<-X2<-matrix(0,n,T)
U <- sample(c(1,2),size = n,replace = T,prob = c(pu[1],1-pu[1]))
X1[U==1,] <- X[U==1,]
X2[U==2,] <- X[U==2,]
wi1 <- matrix(0,nrow=M,ncol=n)
wi1[1,U==1] <- 1




#generate Y
U_true <- sample(c(1,2),size = n,replace = T,prob = c(0.6,0.4))
X1_true <- X[U_true==1,]
X2_true <- X[U_true==2,]
Z1_true <- rnorm(length(X1_true),0,2)
Z2_true <- rnorm(length(X2_true),0,10)
Y1 <- ifelse(exp(X1_true+Z1_true)/(1+exp(X1_true+Z1_true))>0.5,1,0)
Y2 <- ifelse(exp(5*X2_true+Z2_true)/(1+exp(5*X2_true+Z2_true))>0.5,1,0)
Y<-matrix(0,nrow=100,ncol=10)
Y[U_true==1,]<-Y1
Y[U_true==2,]<-Y2


for(S in 1:M){      # M simulation
flag<-1             # for convergence decision
while(flag==1){


F1 <- function(pai,sigma,z,x,beta){
  return(pai*dnorm(z,0,sigma)*(exp(beta*x+z)/(1+exp(beta*x+z))))
}


MH<-function(x,beta,sigma,pai){
     z<-numeric(1000)
     z[1]<-rnorm(1,0,sigma)
     for(i in 2:1000){
        zt<-z[i-1]
        Z<-rnorm(1,zt,sigma)
        num<-F1(pai,sigma,Z,x,beta)
        den<-F1(pai,sigma,zt,x,beta)
```

```
      u<-runif(1)
      ifelse(u<(num/den),z[i]<-Z,z[i]<-zt)
    }
    z1<-sample(z[-c(1:100)],1)
    return(z1)
  }
# generate #500*100 z
Z1 <- matrix(0,nrow = K,ncol=n)
Z2 <- matrix(0,nrow = K,ncol=n)

Xmean <- apply(X, 1, mean)
for(k in 1:K){
 for(i in 1:n){

      Z1[k,i]<-MH(Xmean[i],beta1[m-1],sigma1[m-1],pu[m-1])
      Z2[k,i]<-MH(Xmean[i],beta2[m-1],sigma2[m-1],pu[m-1])



 }
}

#generate u(update pai)
num <- 0
den <- 0
for(k in 1:K){
 for(i in 1:n){
  for(j in 1:T){
       num <- num+exp(beta1[m-1]*X[i,j]+Z1[k,i])^Y[i,j]/(1+exp(beta1[m-1]*X[i,j]+Z1[k,i]
       den <- den+exp(beta2[m-1]*X[i,j]+Z2[k,i])^Y[i,j]/(1+exp(beta2[m-1]*X[i,j]+Z2[k,i]

  }
 }
}
Num <- num*pu[m-1]
Den <- num*pu[m-1]+den*(1-pu[m-1])
pu[m] <- Num/Den
U <- sample(c(1,2),size = n,replace = T,prob = c(pu[m],1-pu[m]))
wi1[m,U==1] <- 1


#update sigma 1 & 2
num1 <- num2 <- 0
for(k in 1:K){
  for(i in 1:n){
       num1 <- num1+wi1[m-1,i]*Z1[k,i]^2
       num2 <- num2+(1-wi1[m-1,i])*Z2[k,i]^2
  }
}
sigma1[m] <- sqrt(1/K*num1/sum(wi1[m-1,]))
sigma2[m] <- sqrt(1/K*num2/sum(1-wi1[m-1,]))
```

```
#update beta 1&2
s1<-s2<-s3<-s4<-0
for(k in 1:K){
  for(i in 1:n){
    for(j in 1:T){
      s1 <- s1+wi1[m-1,i]*(Y[i,j]*X[i,j]-X[i,j]*exp(beta1[m-1]*X[i,j]+Z1[k,i])/(1+exp(b
      s2 <- s2-wi1[m-1,i]*(X[i,j]^2*exp(beta1[m-1]*X[i,j]+Z1[k,i])/(1+exp(beta1[m-1]*X[
      s3 <- s3+(1-wi1[m-1,i])*(Y[i,j]*X[i,j]-X[i,j]*exp(beta2[m-1]*X[i,j]+Z2[k,i])/(1+e
      s4 <- s4+(wi1[m-1,i]-1)*(X[i,j]^2*exp(beta2[m-1]*X[i,j]+Z2[k,i])/(1+exp(beta2[m-1
    }
  }
}
beta1[m] <- beta1[m-1]-s1/s2
beta2[m] <- beta2[m-1]-s3/s4

#convergence rule
while(abs(beta1[m]-beta1[m-1])<=0.5 & abs(beta2[m]-beta2[m-1])<=0.5 & abs(sigma1[m]-sigm
{flag<-0}
}
}
```