# 第四次作业参考答案

## 第一种：



# Homework4

*于颖 统计1602 1303160223*

*2019年5月7日*

作业：编程实现 EM 算法，并用如下数据和初始值估计一个 two-component Gaussian mixture model. 使用 contour plot 展示估计的正态分布。

```
##create database

library(MASS)
set.seed(123)
n=1000
mu1=c(0,4)
mu2=c(-2,0)
Sigma1=matrix(c(3,0,0,0.5),nr=2,nc=2)
Sigma2=matrix(c(1,0,0,2),nr=2,nc=2)
phi=c(0.6,0.4)
X=matrix(0,nr=2,nc=1000)

for(i in 1:n){
  if(runif(1)<phi[1]){
    X[,i]=mvrnorm(1,mu=mu1,Sigma = Sigma1)
  }else{
    X[,i]=mvrnorm(1,mu=mu2,Sigma = Sigma2)
  }
}

##inital guess for parameters
x=t(X)
mu10=runif(2)
mu20=runif(2)
Sigma10=diag(2)
Sigma20=diag(2)
phi0=runif(2)
phi=phi0/sum(phi0)
mu=cbind(mu10,mu20)
Sigma=list(Sigma10,Sigma20)
```

20

```
gmm <- function(x, mu, Sigma ,phi){

  ##set initial value

  K=length(phi)
  n=nrow(x)
  epsilon=1e-5
  goal=0

  ##the goal and intermediate variables

  p=matrix(0,nr=nrow(x),nc=ncol(x))
  w=matrix(0,nr=nrow(x),nc=ncol(x))
  sp=matrix(0,nr=nrow(x),nc=ncol(x))
  wx=matrix(0,nr=n,nc=2*K)
  xwx=array(0,dim = c(K,K,n,K))
  w_phi=matrix(0,nr=n,nc=K)
  s=numeric(n)
  sum_w=numeric(2)

  while(TRUE){

    goal0=goal

    ##calculate the probablity of multi normal distribution

    for (i in 1:n){
      for (j in 1:K){
        p[i,j]=1/(sqrt((2*pi)^2)*det(Sigma[[j]]))*exp(-0.5*t(x[i,]-mu[,j])%
*%solve(Sigma[[j]])%*%(x[i,]-mu[,j]))
      }
    }

    for (i in 1:n){
      for(j in 1:K){
        sp[i,j]=p[i,j]*phi[j]
      }
    }

    for(i in 1:n){
      s[i]=sum(sp[i,])
    }


    ##calculate w-ij


    for (i in 1:n){
      for (j in 1:K){
        w[i,j]=sp[i,j]/s[i]
      }
    }
```

```
 for(j in 1:K){
   sum_w[j]=sum(w[,j])
 }

 ##calculate some varible to simply the symbol;


 for (i in 1:n){
   for(j in 1:K){
     wx[i,c((2*j-1),2*j)]=w[i,j]*x[i,]
     w_phi[i,j]=w[i,j]*log(phi[j])
   }
 }

 ##estimate the parameters mu and phi of the next time

 for(j in 1:K){
   phi[j]=1/n*sum(w[,j])
   mu[,j]=(c(sum(wx[,(2*j-1)]),sum(wx[,(2*j)])))/sum_w[j]
 }

 ##estimate the parameter of the next time

 for(i in 1:n){
   for(j in 1:K){
     xwx[,,i,j]=w[i,j]*((x[i,]-mu[,j])%*%(t(x[i,]-mu[,j])))
   }
 }

 for( j in 1:K){
   S=matrix(0,2,2)
   for(i in 1:n){
     S=S+xwx[,,i,j]
   }
   Sigma[[j]]=S/sum_w[j]
 }


 goal=sum(w_phi)

 ##the condition to terminate

 if(abs(goal-goal0)<epsilon)
 {
   break
 }

}
return (list(mu = mu,Sigma=Sigma, Phi = phi))
return(p[i,j])
}
```

```
##apply the function of EM algorim to estimate parameters
result=gmm(x,mu,Sigma,phi)
```

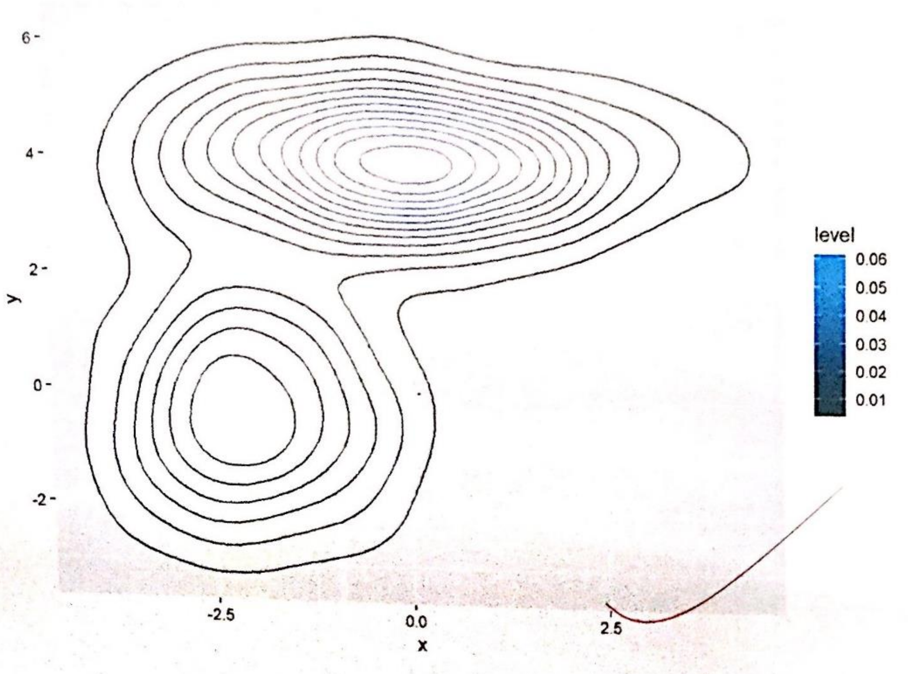
估计得到的two-component Gaussian mixture model

```
result
```

```
## $mu
##              mu10        mu20
## [1,] -2.0446225 -0.02619483
## [2,] -0.2037778  4.01843819
##
## $Sigma
## $Sigma[[1]]
##            [,1]       [,2]
## [1,] 1.01367706 0.02804665
## [2,] 0.02804665 1.72003584
##
## $Sigma[[2]]
##            [,1]       [,2]
## [1,] 2.97606586 0.03478678
## [2,] 0.03478678 0.47973608
##
##
## $Phi
## [1] 0.4050054 0.5949946
```

使用 contour plot 展示估计的正态分布。

```
library(MASS)
library(ggplot2)
library(mvtnorm)
mu1<-c(-2.0446225,-0.2037778)
sigma1<-matrix(c(1.01367706,0.02804665,0.02804665,1.72003584),nrow=2,ncol=2)
mu2<-c(-0.02619483,4.01843819)
sigma2<-matrix(c(2.97606586,0.03478678,0.03478678,0.47973608),nrow=2,ncol=2)
N = 1000
U = runif(N)
x = matrix(NA,nrow=N,ncol=2)
for(i in 1:N){
  if(U[i]<.4)
   {x[i,] = mvrnorm(1,mu1,sigma1)}
  else
   {x[i,] = mvrnorm(1,mu2,sigma2)}
}
x<-data.frame(x=x[,1],y=x[,2])
ggplot(x,aes(x=x,y=y))+stat_density2d(aes(colour = ..level..))
```


6
4
2
0
-2
y
-2.5
0.0
2.5
x
level
0.06
0.05
0.04
0.03
0.02
0.01

**第二种：**

1006160117

单顺衡

# Homework-4

*Ssh*

*2019/5/1*

## EM exercise

About this question of estimation of GMM, the algorithem of EM has given the function of iteration, below:

$w_{ij} = \frac{p(x_i|\mu_j^{(t)},\Sigma_j^{(t)})\phi_j^{(t)}}{\sum_{k=1}^{K} p(x_i|\mu_k^{(t)},\Sigma_k^{(t)})\phi_k^{(t)}}, j = 1, ùùù, K; i = 1, ùùù, n.$

$\phi_j^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} w_{ij}\ , j = 1, ùùù, K.$

$\mu_j^{(t+1)} = \frac{\sum_{i=1}^{n} w_{ij}x_i}{\sum_{i=1}^{n} w_{ij}}, j = 1, ùùù, K.$

So the solution and code are below:

```
library('mvtnorm')#this package has function 'dmvnorm'
# create dataset
library(MASS)
set.seed(123)
n=1000
mu1 = c(0,4)
mu2 = c(-2,0)
Sigma1 = matrix(c(3,0,0,0.5),nr=2,nc=2)
Sigma2 = matrix(c(1,0,0,2),nr=2,nc=2)
phi = c(0.6,0.4)
X = matrix(0,nr=2,nc=n)

for (i in 1:n){
  if (runif(1)<=phi[1]){
    X[,i] = mvrnorm(1,mu=mu1,Sigma=Sigma1) }
  else{ X[,i] = mvrnorm(1,mu=mu2,Sigma=Sigma2) }
}

mu1=runif(2)
mu2=runif(2)
Sigma1 = diag(2)
Sigma2 = diag(2)
phi = runif(2)
phi = phi/sum(phi)
#above are the code of teacher,now this is mine.


w=matrix(nr=n,nc=2) #the conditional distribution of z_i

#below are for the storage of some variable
storage_mu1=data.frame(x1=mu1[1],x2=mu1[2])
storage_mu2=data.frame(x1=mu2[1],x2=mu2[2])
storage_sigma1=list(Sigma1)
storage_sigma2=list(Sigma2)
```

```
storage_phi=data.frame(x1=phi[1],x2=phi[2])

epi=10^(-5)#the precision of iteration



#begin loop
t=2#the record of number of loop,2 is a good number.
repeat{

  for (i in 1:n) {
    w[i,1]=dmvnorm(X[,i],mean=mu1,sigma=Sigma1)*phi[1]
    w[i,2]=dmvnorm(X[,i],mean=mu2,sigma=Sigma2)*phi[2]
  }
  w=w/apply(w, 1, sum) #the caucalation of w


  phi[1]=sum(w[,1])/n
  phi[2]=sum(w[,2])/n #the caucalation of phi


  mu1=X%*%w[,1]/sum(w[,1])
  mu2=X%*%w[,2]/sum(w[,2])#the caucalation of mu


  tem=(X-as.vector(mu1))%*%diag(sqrt(w[,1]))
  Sigma1=tem%*%t(tem)/sum(w[,1])
  tem=(X-as.vector(mu2))%*%diag(sqrt(w[,2]))
  Sigma2=tem%*%t(tem)/sum(w[,2])  #the caucalation of sigma


  #the storage
  storage_phi[t,]=phi
  storage_mu1[t,]=mu1
  storage_mu2[t,]=mu2
  storage_sigma1[[t]]=Sigma1
  storage_sigma2[[t]]=Sigma2

#prepare something to judge weather to stop iteration
tem1=dmvnorm(t(X),mean=as.matrix(storage_mu1[t,]),sigma=storage_sigma1[[t]])*storage_phi[t,1]
tem2=dmvnorm(t(X),mean=as.matrix(storage_mu2[t,]),sigma=storage_sigma2[[t]])*storage_phi[t,2]
tem3=log(tem1+tem2)
now=sum(tem3)#now is the likelyhood at the time of t


tem1=dmvnorm(t(X),mean=as.matrix(storage_mu1[t-1,]),sigma=storage_sigma1[[t-1]])*storage_phi[t-1,1]
tem2=dmvnorm(t(X),mean=as.matrix(storage_mu2[t-1,]),sigma=storage_sigma2[[t-1]])*storage_phi[t-1,2]
tem3=log(tem1+tem2)
past=sum(tem3)#past is the likelyhood at the time of t-1


#judge wather to stop
```

```
  if (now-past<epi ) {
    break
  }

#the increase of t
t=t+1
}
```

the results are below:

t=40,so we iterate 39 times all

the estimation of $\phi$,$\mu$,$\Sigma$ is:

$\phi = (0.4069856, 0.5930144)'$

$\mu_1 = (-2.042287, -0.1894091)'$

$\mu_2 = (-0.0210572, 4.02267566)'$

$$\Sigma_1 = \begin{bmatrix} 1.0163586 & 0.0339475 \\ 0.0339475 & 1.75587892 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 2.973612 & 0.0289267 \\ 0.0289267 & 0.4745805 \end{bmatrix}$$

the real parameter is:

$\phi = (0.4, 0.6)'$

$\mu_1 = (-2, 0)'$

$\mu_2 = (0, 4)'$

$$\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

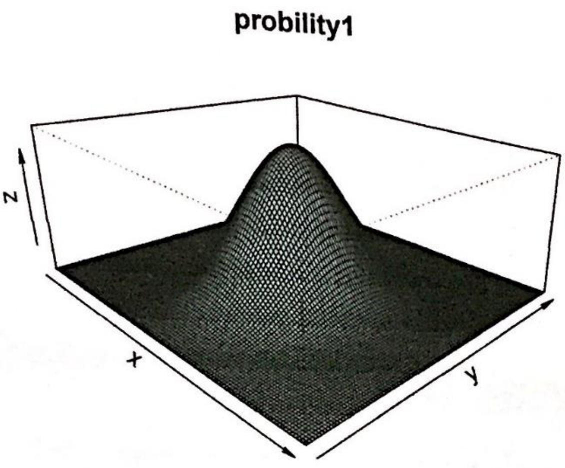
$$\Sigma_2 = \begin{bmatrix} 3 & 0 \\ 0 & 0.5 \end{bmatrix}$$

we can see the goodness of estimation of EM

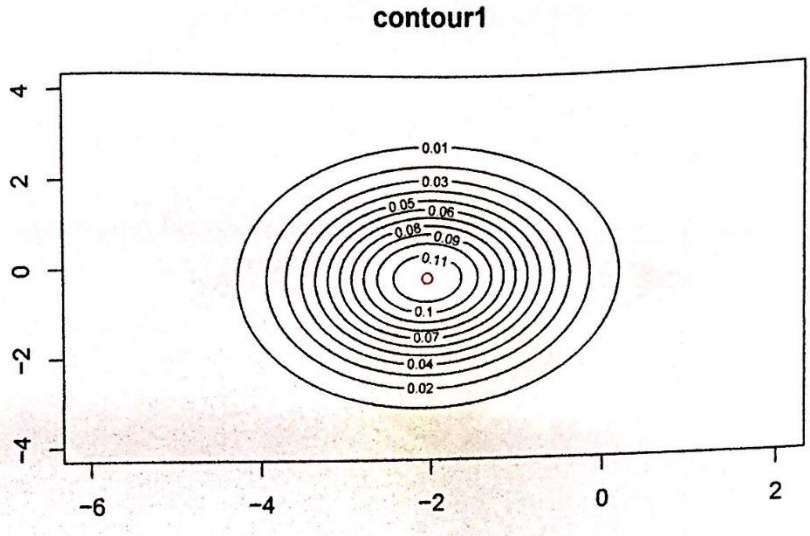
Now ,we decided to draw a contour plot of our estimation of Normal distribution

below are code:

```
mu1=c(-2.0422879,-0.1894091)#the parameter
Sigma1=matrix(c(1.01635869,0.03394757,0.03394757,1.75587892),nr=2)#the parameter


x=seq(-6,2,length=111)#x
y=seq(-4,4,length=111)#y


f<-function(x,y){  dmvnorm(matrix(c(x,y),nr=length(x)),mean=mu1,sigma=Sigma1 ) }
z<-outer(x,y,FUN = f)#z

myfigure1=persp(x,y,z,,theta =45,phi =20,expand = 0.4,col = "lightblue",main='probility1')#the 3-D prob
```

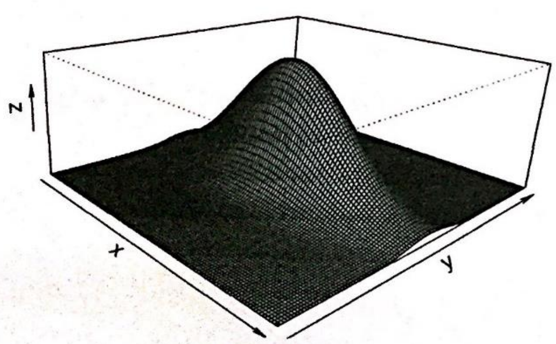



```
contour(x,y,z,main='contour1')#the contour  of normal
points(-2.0422879,-0.1894091,col = "red")
```

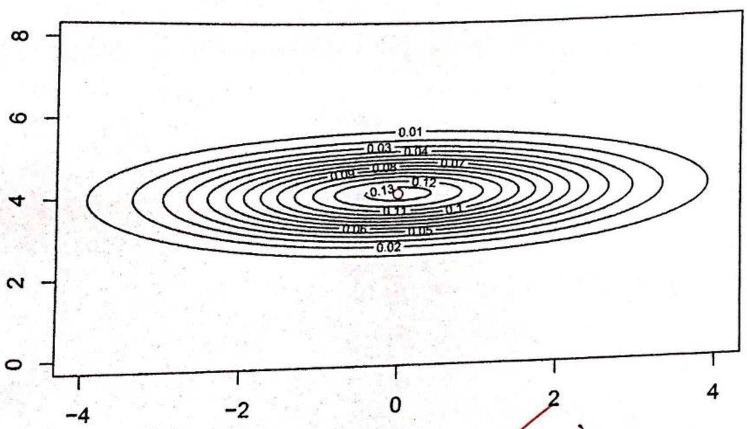



similarly,for the second normal distribution

**probility2**




**contour2**




They are handsome,do you think so?

**第三种：**

# Homework 4 & 5

统计 1601

1303160109

甄梦楠

## 1. Homework 4

根据给出的代码估计 EM 算法的初始值：

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.5.3
```

```
set.seed(123)
n<-1000
num<-2
mu1<-c(0,4)
mu2<-c(-2,0)
Sigma1<-matrix(c(3,0,0,0.5),nr=2,nc=2)
Sigma2<-matrix(c(1,0,0,2),nr=2,nc=2)
phi<-c(0.6,0.4)
X<-matrix(0,nr=2,nc=n)
for(i in 1:n){
  if(runif(1)<=phi[1]){
    X[,i]=mvrnorm(1,mu=mu1,Sigma=Sigma1)
  }else{
    X[,i]=mvrnorm(1,mu=mu2,Sigma=Sigma2)
  }
}
mu10=runif(2)
mu20=runif(2)
sigma10=diag(2)
sigma20=diag(2)
phi0=runif(2)
phi0=phi0/sum(phi0)
```

定义迭代次数和初始化矩阵：

```
# Initialization
library(mvtnorm)
library(mnormt)
```

```
## Warning: package 'mnormt' was built under R version 3.5.2
```

```
K<-2
niter<-100
mu<-cbind(mu10,mu20)
w<-matrix(0, num, n)
mu1f<-matrix(0, num, n)
mu2f<-matrix(0, num, n)
A<-matrix(0, 1, n)
```

EM 算法:

```
# Start iteration
for (t in 1:niter){
for (i in 1:n){
w[1,i]<-phi0[1] * dmnorm(X[,i], mu[,1], sigma10)
w[2,i]<-phi0[2] * dmnorm(X[,i], mu[,2], sigma20)
A[i]<-w[1,i]+w[2,i]
w[,i]<-w[,i]/A[i]
mu1f[,i]<-w[1,i]*X[,i]
mu2f[,i]<-w[2,i]*X[,i]
}

# update probability
phi0<-rowMeans(w)

# update mean matrix
for (j in 1:K){
mu[j,1]<-sum(mu1f[j,])/sum(w[1,])
mu[j,2]<-sum(mu2f[j,])/sum(w[2,])
}

# update var-cov matrix
sigma_s1f<-list()
sigma_s2f<-list()
sigma_s1<-matrix(0,2,2)
sigma_s2<-matrix(0,2,2)
for(i in 1:1000){
sigma_s1f[[i]]<-(X[,i]-mu[,1])%*%t((X[,i]-mu[,1]))*w[1,i]
sigma_s2f[[i]]<-(X[,i]-mu[,2])%*%t((X[,i]-mu[,2]))*w[2,i]
sigma_s1<-sigma_s1+sigma_s1f[[i]]
sigma_s2<-sigma_s2+sigma_s2f[[i]]
}
sigma10<-sigma_s1/sum(w[1,])
sigma20<-sigma_s2/sum(w[2,])
}
```

输出最优迭代结果：

```
# print results
mu

##            mu10        mu20
## [1,] -2.0423029 -0.02108517
## [2,] -0.1894915  4.02265241

sigma10

##           [,1]      [,2]
## [1,] 1.0163407 0.0339089
## [2,] 0.0339089 1.7556694

sigma20

##            [,1]       [,2]
## [1,] 2.97362316 0.02895689
## [2,] 0.02895689 0.47460838

phi0

## [1] 0.4069743 0.5930257
```

绘制二维高斯混合模型密度图

```
# check limit value for x
min(X[1,])

## [1] -5.613646

max(X[1,])

## [1] 4.982456

min(X[2,])

## [1] -4.138878

max(X[2,])

## [1] 5.804117

# contour plot for GMM
x<-matrix(seq(-6,5,0.1))
y<-matrix(seq(-5,6,0.1))
z<-matrix(0,dim(x)[1],dim(y)[1])
for (k in 1:dim(x)[1]){
for (l in 1:dim(y)[1]){
z[k,l]<-phi0[1]*dmnorm(c(x[k],y[l]),mu[,1],sigma10)+phi0[2]*dmnorm(c(x
[k],y[l]),mu[,2],sigma20)
```

```
}
}
contour(x,y,z,nlevels=20)
```