Report

Chenwei Yue

903243179

**1.Initialization**

At first I use kmeans() function to assign the 1593 data points to 10 clusters using k-means algorithm:



where the object “pre\_cluster” records the information of the preliminary clustering.

Here I set the iteration number of k-means to 15 because this initial clustering might be too “good” to be improved greatly by E-M algorithm if the k-means iteration number is greater than 20.

After that, I calculated the initial value of Nk, , and according to the formulas provided in (3) and (4).

**2. Convergence**

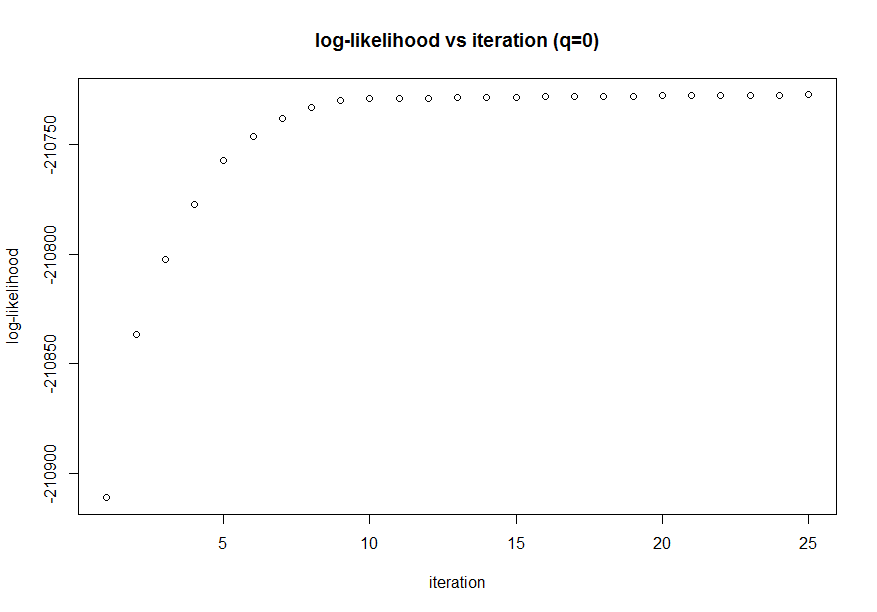
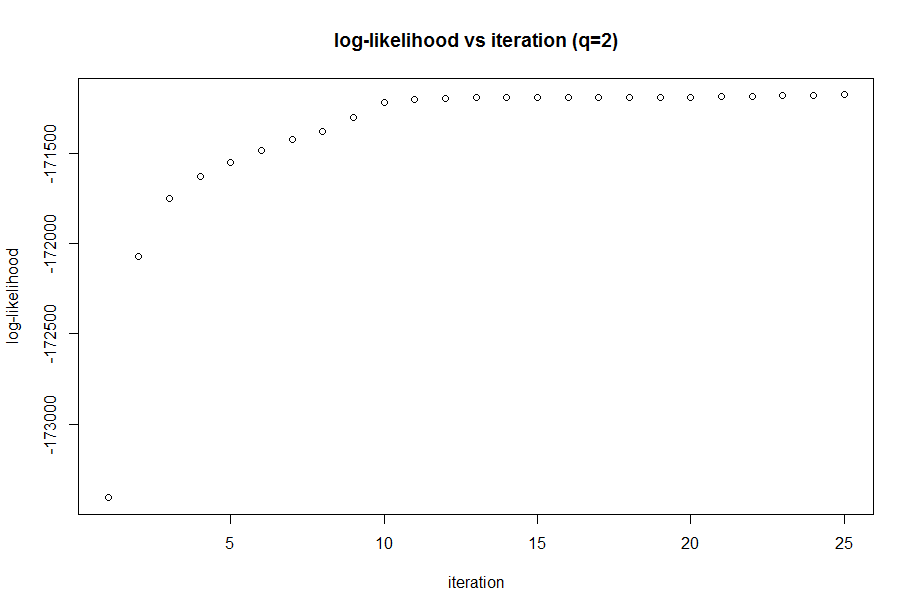
I set the iteration number of E-M algorithm to 25 since the log-likelihood usually come to convergence after the 23th iteration.

The value of log-likelihood after each iteration is show below:

**Table 2.1 log-likelihood after each iteration(q=0,2,4,6)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | q=0 | q=2 | q=4 | q=6 |
| 1 | -211347.6 | -173816.6 | -149157.9 | -130167.7 |
| 2 | -211275.2 | -172375.1 | -147409.3 | -128040.1 |
| 3 | -211239.6 | -171976.0 | -146902.9 | -127501.0 |
| 4 | -211222.1 | -171824.7 | -146699.9 | -127315.2 |
| 5 | -211214.1 | -171733.9 | -146569.5 | -127262.3 |
| 6 | -211207.5 | -171676.2 | -146515.3 | -127207.4 |
| 7 | -211203.5 | -171645.9 | -146497.4 | -127121.0 |
| 8 | -211201.1 | -171628.5 | -146467.1 | -127062.6 |
| 9 | -211198.6 | -171607.9 | -146450.6 | -127006.0 |
| 10 | -211196.0 | -171596.1 | -146441.8 | -126986.3 |
| 11 | -211194.5 | -171559.6 | -146407.4 | -126953.3 |
| 12 | -211193.6 | -171536.1 | -146231.6 | -126888.3 |
| 13 | -211193.0 | -171514.6 | -146155.9 | -126881.8 |
| 14 | -211192.8 | -171473.4 | -146109.0 | -126870.2 |
| 15 | -211192.5 | -171431.3 | -146077.0 | -126844.8 |
| 16 | -211192.2 | -171404.8 | -146016.5 | -126837.8 |
| 17 | -211191.8 | -171393.3 | -146000.4 | -126801.2 |
| 18 | -211191.4 | -171389.0 | -145986.3 | -126742.1 |
| 19 | -211190.9 | -171388.5 | -145979.2 | -126724.9 |
| 20 | -211190.3 | -171388.1 | -145979.2 | -126713.9 |
| 21 | -211190.2 | -171386.1 | -145979.2 | -126698.5 |
| 22 | -211190.0 | -171385.5 | -145979.2 | -126690.2 |
| 23 | -211189.9 | -171385.5 | -145979.2 | -126688.2 |
| 24 | -211189.9 | -171385.5 | -145979.2 | -126687.6 |
| 25 | -211189.9 | -171385.5 | -145979.2 | -126687.4 |

The plots of the log-likelihood vs. iteration number when q=0, 2, 4, 6 respectively are shown below. We can see from the plots that the log-likelihood come to convergence at the last iteration. The gradually increased log-likelihood also proves that E-M algorithm can improve the accuracy of model.

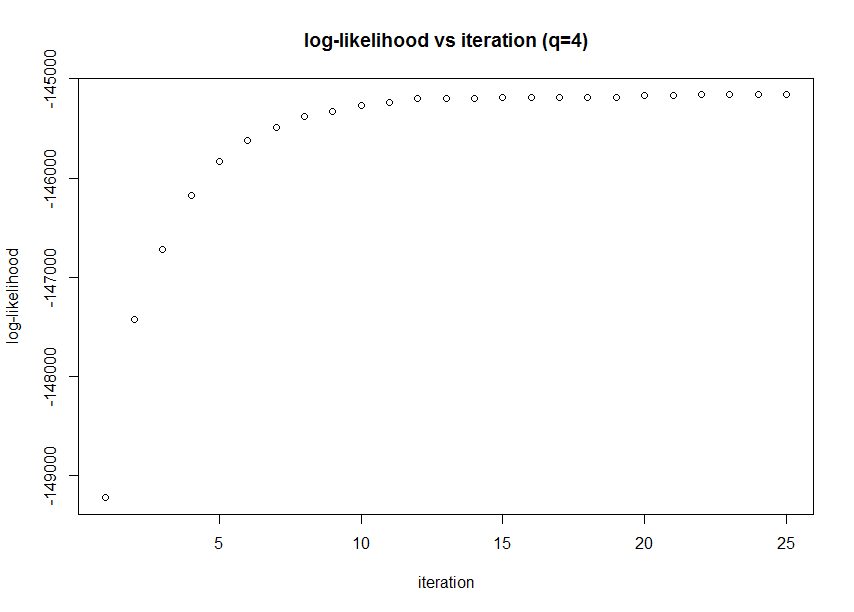
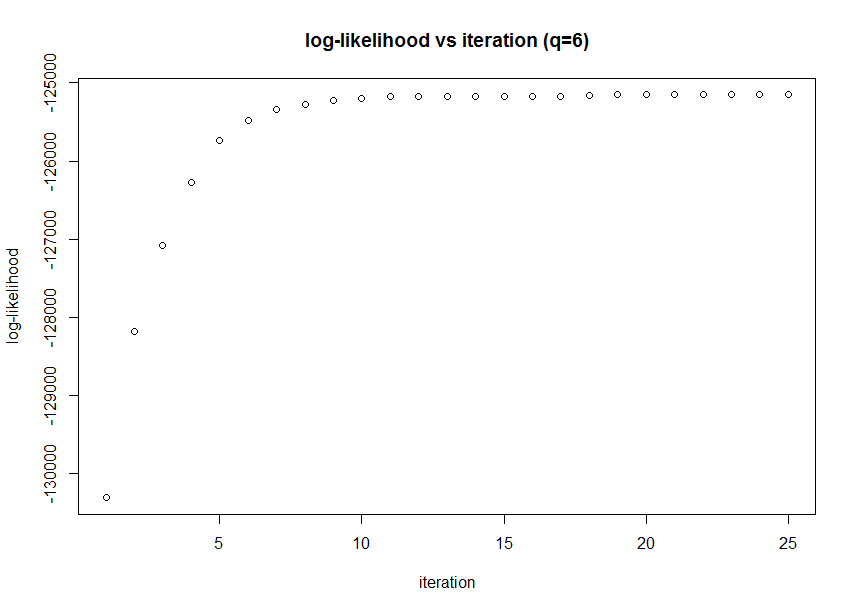
 

Fig 2.1 plots of log-likelihood vs. iteration number when q=0, 2, 4, 6

**3. Choose Number of Principle Components**

The AIC’s at convergence for each q are shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| q | 0 | 2 | 4 | 6 |
| AIC | 422019.5 | 343374.1 | 292360.3 | 253350.1 |

We can see from this table that q=6 minimizes AIC, which implies q=6 is the optimal choice among this four numbers. This is probably because much important information of the original data will be lost if we project original data into a very low dimension, especially when the original dimension is relatively high (256), leading to the decreased model accuracy.

**4. Visualization of Clusters**

I used q=6 as the number of principle components to visualize clusters. The plots for cluster means and 5 random draws for each value of q are presented below, where ith row represents ith cluster.

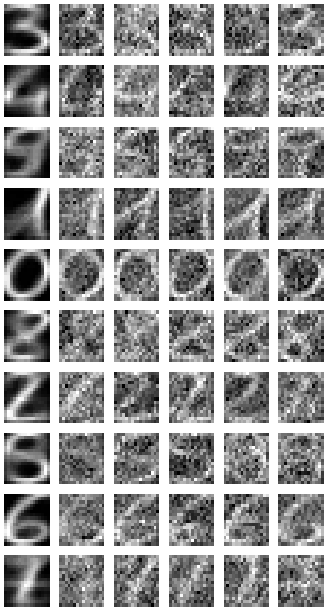
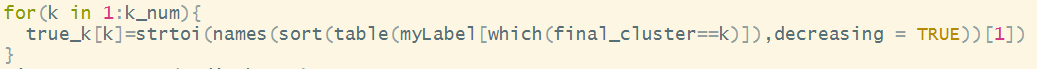


Fig 4.1 Visualization of Clusters

We can see from Fig 4.1 that the true digital value of this 10 clusters (from the 1st cluster to the 10th cluster) are: 3, 4, 9, 1, 0, 8, 2, 5, 6, 7. The randomly generated images based on the cluster-specific distribution (column 2 to 6) are kind of difficult to recognize, because the clusters have some misplaced data points and the regression models are not accurate enough. Besides, the coordinates of the randomly generated images follow normal distribution and thus the randomness of normal distribution also makes the images blurred and non-distinct. By contrast, the images of cluster means are relatively clear, but sometimes it’s still hard to distinguish between 3, 5 and 8. It is due to the fact that the original handwritten numbers are slightly different from each other even if they represent the same number, increasing the error of clustering and regression model building.

In order to let the computer “see” the true digital number of each cluster, I used the following code to find the true digital value of each cluster, according to the most common true value of elements in each cluster.

 where the vector “final\_cluster” records the assignment of all data points after the E-M iterations. The array “true\_k” stores the corresponding digital value of all 10 clusters.

**5. Accuracy**

The vector “mis\_rate” records the mis-categorization rate of each cluster. The results are shown in Table 5.1.

**Table 5.1 mis-categorization rate of each cluster**

|  |  |  |
| --- | --- | --- |
| Cluster | True number value | mis-categorization rate |
| 1 | 3 | 0.446448087 |
| 2 | 4 | 0.071428571 |
| 3 | 9 | 0.464056225 |
| 4 | 1 | 0.101851852 |
| 5 | 0 | 0.000000000 |
| 6 | 8 | 0.416000000 |
| 7 | 2 | 0.343023256 |
| 8 | 5 | 0.429245283 |
| 9 | 6 | 0.006535948 |
| 10 | 7 | 0.312195122 |
| overall |  | 0.3183114 |

According to this table, this E-M algorithm has a good performance when recognizing 0, 1, 4, 6 since the shape of this four numbers are very different from the other numbers. However, the mis-categorization rate is quite large (greater than 0.4) when this program attempts to distinguish between 3, 5 and 8. The overall mis-categorization rate is about 31.8%. In order to prove the accuracy of this algorithm, we can collect large-sized original dataset and increase the number of data dimensions (e.g. from 16\*16 to 32\*32).

**Appendix: R code**

##Read in data:

library(mvtnorm)

myData=read.csv("semeion.csv",header=FALSE)

myX=data.matrix(myData[,1:256])

# "myLabel" records the true value of each data point.

myLabel=data.matrix(apply(myData[,257:266],1,function(xx){

return(which(xx=="1")-1)

}))

d=256

data\_number=1593

Iteration=25

likelihood=matrix(0,Iteration,4)

AICs=array(0,dim=4)

final\_gamma=array(0,dim=c(data\_number,k\_num,4))

final\_mean=array(0,dim=c(k\_num,d,4))

final\_sigma=array(0,dim=c(d,d,k\_num,4))

final\_cluster=array(0,dim=data\_number)

##1.Initialize 10 clusters

k\_num=10

pre\_cluster=kmeans(myX, k\_num, nstart = 10, iter.max =15)

ini\_label=pre\_cluster$cluster #initial assignment of data points

#2.Convergence

for(q in c(0,2,4,6)){

#Initialize gamma, calculate the value of mean, pi,N

gamma=matrix(0,data\_number,10)

for (i in 1:data\_number){

gamma[i,ini\_label[i]]=1

}

N=apply(gamma,2,sum)

mean=t(gamma)%\*%myX

for(i in 1:k\_num){

mean[i,]=mean[i,]/N[i]

}

Pi=N/data\_number

#calculate sigma

sigma=array(0,dim=c(d,d,k\_num))

for(k in 1:k\_num){

Dec=matrix(0,d,d)

for (i in 1:data\_number){

Dec=Dec+gamma[i,k]\*(myX[i,]-mean[k,])%\*%(t(myX[i,]-mean[k,]))

}

Dec=Dec/N[k]

myEig=eigen(Dec,symmetric = TRUE)

Eig\_val=myEig$value

var=sum(Eig\_val[(q+1):d])/(d-q)

if(q==0){

sigma[,,k]=var\*diag(d)

}else{

Vq=myEig$vectors[,1:q]

diag\_matrix=diag(q)

for (i in 1:q){

diag\_matrix[i,i]=sqrt(Eig\_val[i]-var)

}

Wq=Vq%\*%diag\_matrix

sigma[,,k]=Wq%\*%t(Wq)+var\*diag(d)

}

}

#iterations start

for (iter\_count in 1:Iteration){

#E step: update gamma matrix

prob=matrix(0,data\_number,k\_num)

for (k in 1:k\_num){

prob[,k]=dmvnorm(myX,mean[k,],sigma[,,k],log = FALSE)\*Pi[k]

}

for(i in 1:data\_number){

for(k in 1:k\_num){

gamma[i,k]=prob[i,k]/sum(prob[i,])

}

#Calculate log-likelihood

likelihood[iter\_count,q/2+1]=likelihood[iter\_count,q/2+1]+log(sum(prob[i,]))

}

#M step

N=apply(gamma,2,sum)

mean=t(gamma)%\*%myX

for(i in 1:k\_num){

mean[i,]=mean[i,]/N[i]

}

Pi=N/data\_number

#calculate sigma

sigma=array(0,dim=c(d,d,k\_num))

for(k in 1:k\_num){

Dec=matrix(0,d,d)

for (i in 1:data\_number){

Dec=Dec+gamma[i,k]\*(myX[i,]-mean[k,])%\*%(t(myX[i,]-mean[k,]))

}

Dec=Dec/N[k]

myEig=eigen(Dec,symmetric = TRUE)

Eig\_val=myEig$value

var=sum(Eig\_val[(q+1):d])/(d-q)

if(q==0){

sigma[,,k]=var\*diag(d)

}else{

Vq=myEig$vectors[,1:q]

diag\_matrix=diag(q)

for (i in 1:q){

diag\_matrix[i,i]=sqrt(Eig\_val[i]-var)

}

Wq=Vq%\*%diag\_matrix

sigma[,,k]=Wq%\*%t(Wq)+var\*diag(d)

}

}

}

#calculate AIC

AICs[q/2+1]=-2\*likelihood[Iteration,q/2+1]+2\*(d\*q+1-q\*(q-1)/2)

#record the final value of mean and sigma matrix

final\_mean[,,q/2+1]=mean

final\_sigma[,,,q/2+1]=sigma

final\_gamma[,,q/2+1]=gamma

}

# 3.choose optimial number of principle components q:

q\_op=2\*(which.min(AICs)-1)

#4.Visualization of Clusters

#Visualizaiton of Clusters

dev.new(width=7,height=3.5)

par(mai=c(0.05,0.05,0.05,0.05),mfrow=c(10,6))

for (k in 1:k\_num){

image(t(matrix(final\_mean[k,,(q\_op/2+1)],byrow=TRUE,16,16)[16:1,]),col=gray(seq(0,1,length=d)),axes=FALSE)

for(j in 1:5){

image(t(matrix(rmvnorm(1,mean=final\_mean[k,,(q\_op/2+1)],sigma=final\_sigma[,,k,(q\_op/2+1)]),byrow=TRUE,16,16)[16:1,]),col=gray(seq(0,1,length=d)),axes=FALSE)

}

}

#Accuracy

accuracy=0

for (i in 1:data\_number){

final\_cluster[i]=which.max(final\_gamma[i,,q\_op/2+1])

}

#find the true diginal number of each cluster

true\_k=array(-1,dim=k\_num)

for(k in 1:k\_num){

true\_k[k]=strtoi(names(sort(table(myLabel[which(final\_cluster==k)]),decreasing = TRUE))[1])

}

mis\_count=array(0,dim=k\_num)

mis\_rate=array(0,dim=k\_num) #records the mis-categorization rate of each cluster

for(k in 1:k\_num){

mis\_count[k]=length(which(myLabel[which(final\_cluster==k)]!=true\_k[k]))

mis\_rate[k]=mis\_count[k]/length(myLabel[which(final\_cluster==k)])

}

overall\_mis=sum(mis\_count)/data\_number

#draw plots for log-likelihood vs iteration number

PlotPic=1:Iteration

dev.new(width=10,height=5)

plot(PlotPic,likelihood[,1],main="log-likelihood vs iteration (q=0)",xlab="iteration",ylab="log-likelihood")

dev.new(width=8,height=8)

plot(PlotPic,likelihood[,2],main="log-likelihood vs iteration (q=2)",xlab="iteration",ylab="log-likelihood")

dev.new(width=8,height=8)

plot(PlotPic,likelihood[,3],main="log-likelihood vs iteration (q=4)",xlab="iteration",ylab="log-likelihood")

dev.new(width=8,height=8)

plot(PlotPic,likelihood[,4],main="log-likelihood vs iteration (q=6)",xlab="iteration",ylab="log-likelihood")