Handwritten Digit Clustering Using Principal Components Gaussian Mixture Model and EM Algorithm (with R code)

Xiaochen Zhang
Georgia Tech 2015 Spring

In this project, we cluster the handwritten digits data using the EM algorithm with a principle components step within each maximization. The test dataset comes from "Semeion Handwritten Digit Data Set".

Data Description:

- A subset of the 1593 handwritten digits were used. stretched in a rectangular box 16x16 in a gray scale of 256 values. Then each pixel of each image was scaled into a Boolean (1/0) value using a fixed threshold.
- Each person wrote on a paper all the digits from 0 to 9, twice. The commitment was to write the digit the first time in the normal way (trying to write each digit accurately) and the second time in a fast way (with no accuracy).

The whole process consists of 5 steps:

- Step 1: Cluster the raw data using K-means method.
- Step 2: Perform EM algorithm and visualize the log likelihood through iteration.
- Step 3: Choose the number of principle components using AIC.
- Step 4: Visualize the clustering results.
- Step 5: Model Evaluation and Statistics.

Step 1: Initialization

Use R's kmeans function with several random starts to build a preliminary cluster. Set γ_{ik}= 1 if observation i is assigned to cluster k and $\gamma_{ik} = 0$ otherwise. By running the K-means algorithm 20 times, each with a maximum iteration number of 100, I get 10 clusters. Figure 1 shows the cluster mean of the 10 clusters.

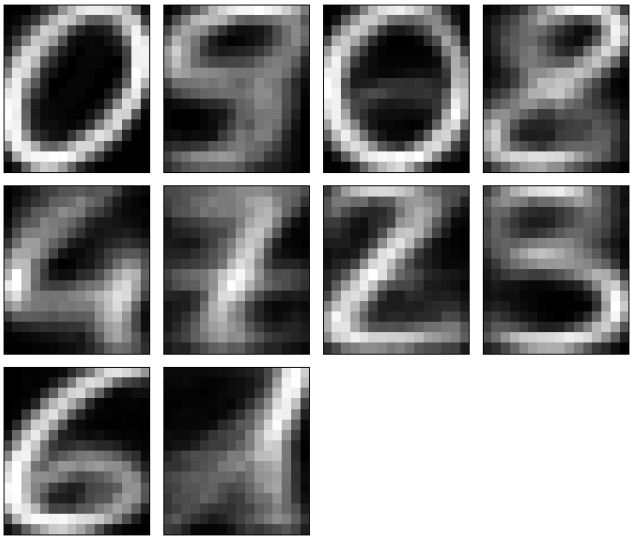


Figure 1 Kmeans result

The mis-categorization rates for the 10 digits are shown in Figure 2. The overall mis-categorization rate is 41.87%.

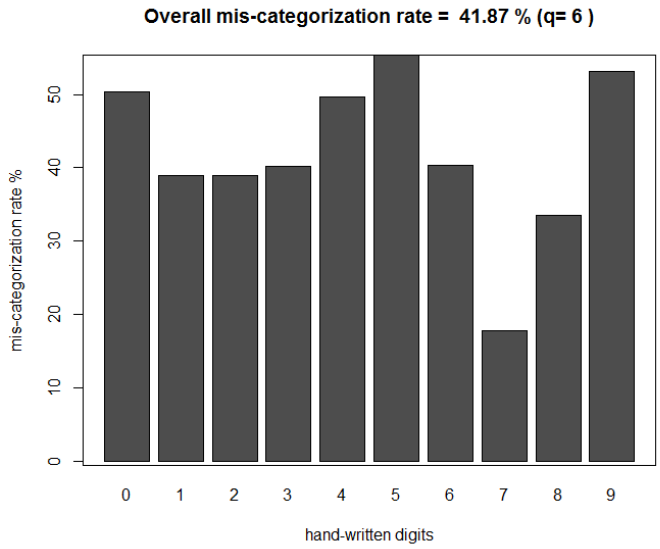
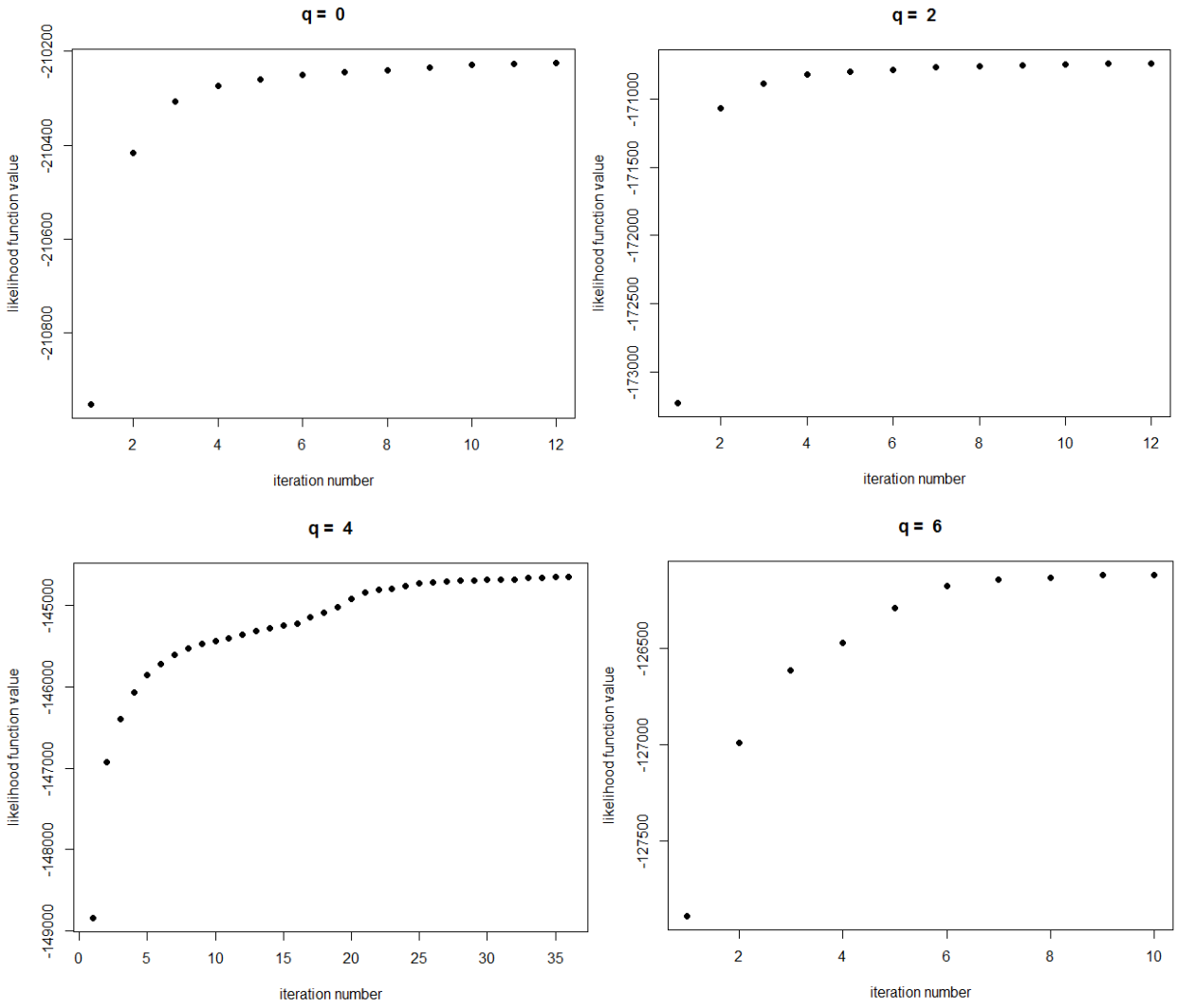


Figure 2 Mis-categorization rate

Step 2. Check Convergence:

The log-likelihood Vs iteration number plots are shown as follows, where the convergence criterion is set as the log-likelihood value improvement is less than 0.001%. We generate a plot of the

observed data log-likelihood vs. iteration number (4 plots, 1 for a different q, where q is the number of principle components).



I am surprise to find out that it took 36 iterations for q=4 to converge. According to the figure, it seems that the cluster algorithm managed to jumps out of a local optimal around iteration 15 before it converges.
The final log-likelihood function values are listed as follows:

q value	Log-likelihood	Iteration number
0	-210225	12
2	-170740	12
4	-144646	36
6	-126119	10

As we can see from the table, the log-likelihood value increases as q increase. We can generally predict that the q=6 should yield the best clustering result later.

Step 3. Choice of Number of Principle Components Through AIC:
The Akaike information criterion (AIC) is a measure of the relative quality of a statistical model for a given set of data. A lower AIC value means less information is lost given the specific model. By computing the AIC value at convergence, the AIC value is minimized at q=6.

q value	AIC value
0	420451
2	342502
4	291328
6	255280

Step 4. Visualization of Clusters:
We visualized the cluster mean and drew 5 samples from each cluster-specific distribution, where q is set to be 6. The clustering algorithm works pretty good at digits: 3, 4, 7, 8, but performs poorly at digits 0, 1, 5, 9. From the visualization, we can see that although the EM algorithm provides a better clustering results compared with Kmeans, it is still not good enough for practice application. For example, the third cluster has a "theta" shape, which does not belong to any digit.

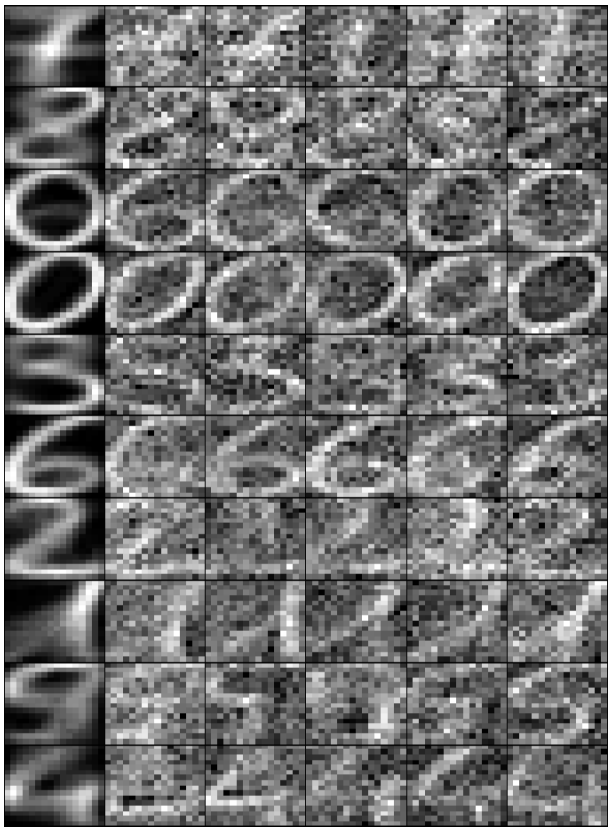
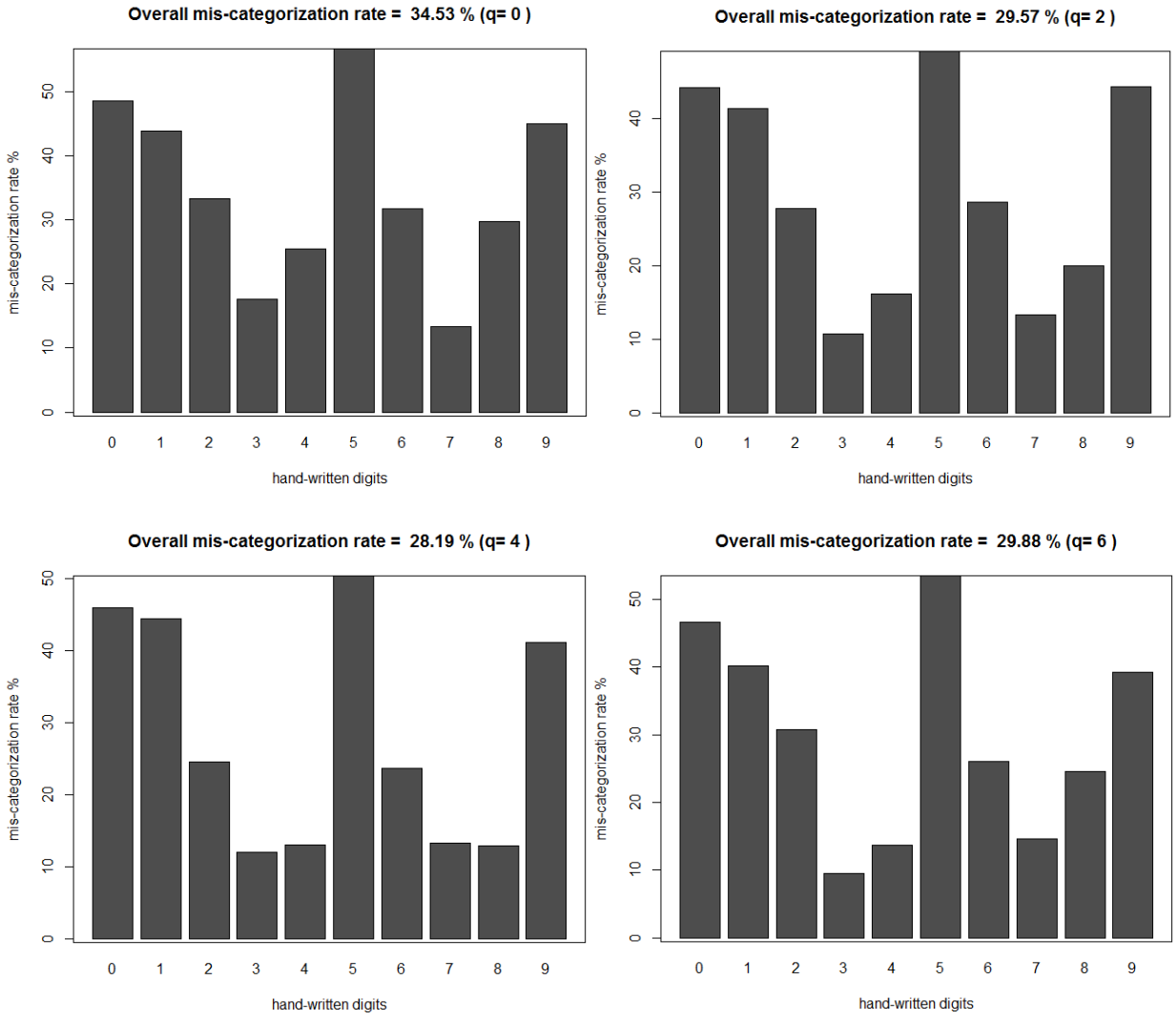


Figure 3 Clustering results (q=6)

Step 5. Accuracy Assessment:

Compare the class labels to the clusters. For each class label, we calculate the mis-categorization rate (define an observation as mis-categorized if it is not in the most common categorization for the particular class and the overall mis-categorization rate.

The mis-categorization rates are visualized by q = 0, 2, 4, and 6. Although q=6 yeilds the lowest AIC value, we are supervised to find out the lowest mis-categorization rate is achieved when q=4.



Appendix: R code

```
#####################################################################
# Read in the data
#####################################################################
library(mvtnorm)
myData=read.csv("semeion.csv",header=FALSE)
# Build data matrix with (thresholded) pixel and label data
myX=data.matrix(myData[,1:256])
myLabel=apply(myData[,257:266],1,function(xx){
 return(which(xx=="1")-1)
})

#####################################################################
# Find out initial clusters using Kmeans
#####################################################################
K = 10
N = dim(myX)[1]
D = dim(myX)[2]
myKmeans = kmeans(myX, K, iter.max = 100,nstart=20)

clusterLabel = myKmeans$cluster
clusterMean = myKmeans$centers

# plot the clusterMean
dev.new(width=8, height=6)
par(mai=c(0.1,0.1,0.1,0.1),cex=0.8,mfrow=c(3,4))
for(ii in 1:K){
   image(t(matrix(clusterMean[ii,],byrow=TRUE,16,16)[16:1,]),col=gray(0:128/128),axes=FALSE)
   box()
}
# accuracy analysis
# calculate new Labels
KMLabel = matrix(0,N,K)
for(ii in 1:N){
 KMLabel[ii,clusterLabel[ii]] = 1
}
# Accuracy Assessment
misRate = matrix(1,K,1)
temp1 = 0
for (ii in 1:K){
 temp = apply(KMLabel[myLabel==(ii-1),],2,function(xx){
  return(sum(xx))
 })
 misRate[ii,] = 1-max(temp)/sum(temp)
 temp1 = temp1+max(temp)
}
OverAllMisRate = 1-temp1/N
misRate = matrix(misRate,K,1)
# plot accuracy Assessment result
misRate = misRate*100
OverAllMisRate = OverAllMisRate*100
barplot(t(misRate),names.arg = c("0", "1", "2","3", "4", "5","6", "7", "8","9"),xlab="hand-written digits", ylab="mis-
categorization rate %", main=paste("Overall mis-categorization rate = ", round(OverAllMisRate, digits = 2), "% (q=",q,")"))
box()
```

```
#####################################################################
# EM algorithm
#####################################################################

q = 6
# Initialize alfa=mean, beta=variance, and pi=class membership possibility.

myZ = matrix(0,N,K) # Initialize myZ
temp = cbind(1:N,clusterLabel)
myZ[temp] = 1;

Nk = apply(myZ,2,function(xx){
  return(sum(xx))
}) # Initialize membership probability
Pi = Nk/N

myMean = clusterMean # Initialize clustermean

mySigma = array(0,dim=c(D,D,K)) # Initialize variance
myVar = array(0,dim=c(D,D,K))
for(ii in 1:K){
  temp = matrix(0,D,D)
  for(jj in 1:N){
    temp = temp+(myX[jj,]-myMean[ii,])%*%t(myX[jj,]-myMean[ii,])*myZ[jj,ii]
  }
  myVar[,,ii] = temp/Nk[ii]
  mySVD = svd(myVar[,,ii])
  SQ = 1/(D-q)*sum(mySVD$d[(q+1):D])
  WQ = mySVD$v[,1:q]%*%diag(apply(as.matrix(mySVD$d[1:q]),1,function(xx){
    return(sqrt(xx-SQ))
  }),q,q)
  mySigma[,,ii] = WQ%*%t(WQ)+SQ*diag(1,D,D)
#  cat("=")
}

# calculate the likelihood
tempP = matrix(0,N,K)
for(ii in 1:K){
  tempP[,ii] = dmvnorm(myX, mean = myMean[ii,], sigma = mySigma[,,ii])
}
likelihood_old = sum(log(tempP%*%Pi))
myLikelihood = likelihood_old

# EM algorithm
continueLoop = TRUE
iter = 0
while(continueLoop){
  # E step
  tempZ = matrix(0,N,K)
  for(ii in 1:K){
    tempZ[,ii] = dmvnorm(myX, mean = myMean[ii,], sigma = mySigma[,,ii])*Pi[ii]
  }
  myZ = t(apply(tempZ,1,function(xx){
    return(xx/sum(xx))
  }))
  # M step
  # update Pi
```

```
 Nk = apply(myZ,2,function(xx){
  return(sum(xx))
 }) # Nk is a k by 1 matrix consisting of Nk
 Pi = t(Nk)/N
 # update myMean
 myMean = t(myZ)%*%myX/Nk
 # update mySigma
 myVar = array(0,dim=c(D,D,K))
 for(ii in 1:K){
  temp = matrix(0,D,D)
  for(jj in 1:N){
   temp = temp+(myX[jj,]-myMean[ii,])%*%t(myX[jj,]-myMean[ii,])*myZ[jj,ii]
  }
  myVar[,,ii] = temp/Nk[ii]
  mySVD = svd(myVar[,,ii])
  SQ = 1/(D-q)*sum(mySVD$d[(q+1):D])
  WQ = mySVD$v[,1:q]%*%diag(apply(as.matrix(mySVD$d[1:q]),1,function(xx){
   return(sqrt(xx-SQ))
  }),q,q)
  mySigma[,,ii] = WQ%*%t(WQ)+SQ*diag(1,D,D)
  #  cat("=")
 }

 # calculate the likelihood
 tempP = matrix(0,N,K)
 for(ii in 1:K){
  tempP[,ii] = dmvnorm(myX, mean = myMean[ii,], sigma = mySigma[,,ii])
 }
 likelihood_new = sum(log(tempP%*%t(Pi)))
 # check variable convergence or likelihood function values
 if(abs((likelihood_new-likelihood_old)/likelihood_new)<0.00001){
  continueLoop = FALSE
 }
 likelihood_old = likelihood_new
 myLikelihood = cbind(myLikelihood,likelihood_old)
 iter=iter+1
 cat("\n","iteration number=",iter,"   ","likelihood=",likelihood_new)
}

# plot likelihood Vs. iter
dev.new(width=6, height=4)
par(mar=c(0.5,0.45,0.35.0.05),cex=0.8)
plot(1:(iter+1),myLikelihood, pch=19,axes=TRUE, xlab="iteration number",ylab="likelihood function value", main=paste("q = ",
q))

# compute AIC
AIC = -2*likelihood_old+2*(D*q-q*(q-1)/2)

# calculate new Labels
EMLabel = matrix(0,N,K)
for(ii in 1:N){
 EMLabel[ii,which.max(myZ[ii,])] = 1
}

# Accuracy Assessment
misRate = matrix(1,K,1)
temp1 = 0
for (ii in 1:K){
```

```
 temp = apply(EMLabel[myLabel==(ii-1),],2,function(xx){
  return(sum(xx))
 })
 misRate[ii,] = 1-max(temp)/sum(temp)
 temp1 = temp1+max(temp)
}
OverAllMisRate = 1-temp1/N

# plot accuracy Assessment result
misRate = misRate*100
OverAllMisRate = OverAllMisRate*100
barplot(t(misRate),names.arg = c("0", "1", "2","3", "4", "5","6", "7", "8","9"),xlab="hand-written digits", ylab="mis-
categorization rate %", main=paste("Overall mis-categorization rate = ", round(OverAllMisRate, digits = 2), "% (q=",q,")"))
box()

# visualizatoin
dev.new(width=6, height=10)
par(mai=c(0,0,0,0),cex=0.8,mfrow=c(10,6))
myDraw = array(0,dim=c(6,D,K))
clusterMean = myMean
for(ii in 1:K){
 myDraw[1,,ii] = clusterMean[ii,]
 myDraw[2:6,,ii] = rmvnorm(n=5,mean=myMean[ii,],sigma=mySigma[,,ii])
}

for(ii in 1:K){
 for(jj in 1:6){
  image(t(matrix(myDraw[jj,,ii],byrow=TRUE,16,16)[16:1,]),col=gray(0:128/128),axes=FALSE)
  box()
 }
}

#####################################################################
# EM algorithm q=0
#####################################################################
q = 0
# Initialize alfa=mean, beta=variance, and pi=class membership possibility.
myZ = matrix(0,N,K) # Initialize myZ
temp = cbind(1:N,clusterLabel)
myZ[temp] = 1;
Nk = apply(myZ,2,function(xx){
 return(sum(xx))
}) # Initialize membership probability
Pi = Nk/N
myMean = clusterMean # Initialize clustermean
mySigma = array(0,dim=c(D,D,K)) # Initialize variance
myVar = array(0,dim=c(D,D,K))
for(ii in 1:K){
 temp = matrix(0,D,D)
 for(jj in 1:N){
  temp = temp+(myX[jj,]-myMean[ii,])%*%t(myX[jj,]-myMean[ii,])*myZ[jj,ii]
 }
 myVar[,,ii] = temp/Nk[ii]
 mySVD = svd(myVar[,,ii])
 SQ = 1/(D-q)*sum(mySVD$d[(q+1):D])
 mySigma[,,ii] = SQ*diag(1,D,D)
 #  cat("=")
}
```

```
# calculate the likelihood
tempP = matrix(0,N,K)
for(ii in 1:K){
 tempP[,ii] = dmvnorm(myX, mean = myMean[ii,], sigma = mySigma[,,ii])
}
likelihood_old = sum(log(tempP%*%Pi))
myLikelihood = likelihood_old

# EM algorithm
continueLoop = TRUE
iter = 0
while(continueLoop){
 # E step
 tempZ = matrix(0,N,K)
 for(ii in 1:K){
  tempZ[,ii] = dmvnorm(myX, mean = myMean[ii,], sigma = mySigma[,,ii])*Pi[ii]
 }
 myZ = t(apply(tempZ,1,function(xx){
  return(xx/sum(xx))
 }))
 # M step
 # update Pi
 Nk = apply(myZ,2,function(xx){
  return(sum(xx))
 }) # Nk is a k by 1 matrix consisting of Nk
 Pi = t(Nk)/N
 # update myMean
 myMean = t(myZ)%*%myX/Nk
 # update mySigma
 myVar = array(0,dim=c(D,D,K))
 for(ii in 1:K){
  temp = matrix(0,D,D)
  for(jj in 1:N){
   temp = temp+(myX[jj,]-myMean[ii,])%*%t(myX[jj,]-myMean[ii,])*myZ[jj,ii]
  }
  myVar[,,ii] = temp/Nk[ii]
  mySVD = svd(myVar[,,ii])
  SQ = 1/(D-q)*sum(mySVD$d[(q+1):D])
  mySigma[,,ii] = SQ*diag(1,D,D)
  #  cat("=")
 }

 # calculate the likelihood
 tempP = matrix(0,N,K)
 for(ii in 1:K){
  tempP[,ii] = dmvnorm(myX, mean = myMean[ii,], sigma = mySigma[,,ii])
 }
 likelihood_new = sum(log(tempP%*%t(Pi)))
 # check variable convergence or likelihood function values
 if(abs((likelihood_new-likelihood_old)/likelihood_new)<0.00001){
  continueLoop = FALSE
 }
 likelihood_old = likelihood_new
 myLikelihood = cbind(myLikelihood,likelihood_old)
 iter=iter+1
 cat("\n","iteration number=",iter,"   ","likelihood=",likelihood_new)
}
```

```
# plot likelihood Vs. iter
dev.new(width=6, height=4)
par(mar=c(0.5,0.45,0.35.0.05),cex=0.8)
plot(1:(iter+1),myLikelihood, pch=19,axes=TRUE, xlab="iteration number",ylab="likelihood function value", main=paste("q = ",
q))

# compute AIC
AIC = -2*likelihood_old+2*(D*q-q*(q-1)/2)

# calculate new Labels
EMLabel = matrix(0,N,K)
for(ii in 1:N){
  EMLabel[ii,which.max(myZ[ii,])] = 1
}

# Accuracy Assessment
misRate = matrix(1,K,1)
temp1 = 0
for (ii in 1:K){
  temp = apply(EMLabel[myLabel==(ii-1),],2,function(xx){
    return(sum(xx))
  })
  misRate[ii,] = 1-max(temp)/sum(temp)
  temp1 = temp1+max(temp)
}
OverAllMisRate = 1-temp1/N

# plot accuracy Assessment result
misRate = misRate*100
OverAllMisRate = OverAllMisRate*100
barplot(t(misRate),names.arg = c("0", "1", "2","3", "4", "5","6", "7", "8","9"),xlab="hand-written digits", ylab="mis-
categorization rate %", main=paste("Overall mis-categorization rate = ", round(OverAllMisRate, digits = 2), "% (q=",q,")"))
box()

# visualizatoin
dev.new(width=6, height=10)
par(mai=c(0,0,0,0),cex=0.8,mfrow=c(10,6))
myDraw = array(0,dim=c(6,D,K))
clusterMean = myMean
for(ii in 1:K){
  myDraw[1,,ii] = clusterMean[ii,]
  myDraw[2:6,,ii] = rmvnorm(n=5,mean=myMean[ii,],sigma=mySigma[,,ii])
}

for(ii in 1:K){
  for(jj in 1:6){
    image(t(matrix(myDraw[jj,,ii],byrow=TRUE,16,16)[16:1,]),col=gray(0:128/128),axes=FALSE)
    box()
  }
}
```