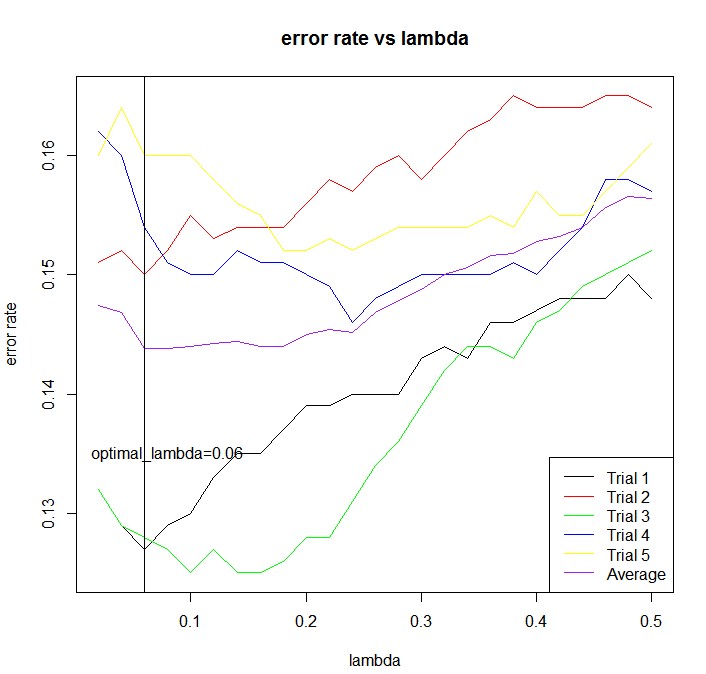
# LDA classification of the handwritten digits:

First of all, we should determine an optimal lambda. Here we choose lambda from 0.02to 0.5 and the interval is 0.02.

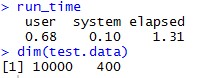
The error rate plots after running the algorithm 5 times with lambda ranging from 0.02 to o.5 are in the following:



From the above we can see that when lambda is 0.04 , the error rate is the minimum.

Then we use the lambda to train the 5000 training data and test on the testing data.

When test the testing data, the total time cost is in the following:



Here user time is how many seconds the computer spent doing your calculations. System time is how much time the operating system spent responding to your program's requests. Elapsed time is the sum of those two, plus whatever "waiting around" your program and/or the OS had to do.

Therefore, I use the elapsed time to denote the time cost. Meaningfully, it is better to use the average time which indicates the time cost of one data stream.

We also record the error rate.

C:\Users\Christina\AppData\Roaming\Tencent\Users\919834852\QQ\WinTemp\RichOle\@]CBL(GEZ4V_[0S@V@P%4O2.jpg

These kind of information are in the following grid:

|  |  |  |
| --- | --- | --- |
|  | Average time of one data stream(s) | The error rate of LDA method |
|  | 1.31xe(-4) | 0.1451 |

# Code:

# use the LDA method to classify the handwritten digits

data=load("C:/Users/Christina/Desktop/digitsdata.RData")

image(t(1 - training.data[3,1,,])[,20:1],col=gray(seq(0, 1, length.out=256)),axes=FALSE, asp=1)

num.class <- dim(training.data)[1] # Number of classes

num.training <- dim(training.data)[2] # Number of training data per class

d <- prod(dim(training.data)[3:4]) # Dimension of each training image (rowsxcolumns)

num.test <- dim(test.data)[2] # Number of test data

dim(training.data) <- c(num.class \* num.training, d) # Reshape training data to 2-dim matrix

dim(test.data) <- c(num.class \* num.test, d) # Same for test.

training.label <- rep(0:9, num.training) # Labels of training data.

test.label <- rep(0:9, num.test) # Labels of test data

# for the conveniece of dealing with data, we try to set training label as the rownames of the

#training data

rownames(training.data)=training.label

#return the ax+b

lda=function (mean,sigma,x){

Isigma=solve(sigma )

b=-(0.5\*((t(mean)%\*%Isigma)%\*%mean))[1,1]-1

a=t(mean)%\*%Isigma

return (x%\*%t(a)+b)

}

#divide the training set per class into 2 parts:400 training examples and 100 remaing ones to test

#compute the corresponding covariance matrix with different lambda:get the value from 0 to 1

#run the algorithm five times on a random subset of 400 training examples per class

#evaluate the error on the remaining 100 examples per class

#obtain the optimal lamda that minimize the average error of the 5 times

error\_rate=matrix(NA,nrow=5,ncol=25)

start\_time= proc.time()

for(i in 1:5){

# this is because we want to run the whole data 5 times

# we sample rows aaccording to their labels and get the corresponding traing data and test data.

#then we get the mean and cov

#class 0

class0=training.data[rownames(training.data)=="0",]

train\_0=sample((nrow(class0)),size=400,replace = FALSE, prob = NULL)

train0=class0[train\_0, ]

test0=class0[-train\_0, ]

cov0=cov(train0)

#class 1

class1=training.data[rownames(training.data)=="1",]

train\_1=sample(nrow(class1),size=400,replace = FALSE, prob = NULL)

train1=class1[train\_1, ]

test1=class1[-train\_1, ]

cov1=cov(train1)

mean1=colMeans(train1)

#class 2

class2=training.data[rownames(training.data)=="2",]

train\_2=sample(nrow(class2), size=400,replace = FALSE, prob = NULL)

train2=class2[train\_2, ]

test2=class2[-train\_2, ]

cov2=cov(train2)

mean2=colMeans(train2)

# class 3

class3=training.data[rownames(training.data)=="3",]

train\_3=sample(nrow(class3), size=400,replace = FALSE, prob = NULL)

train3=class3[train\_3, ]

test3=class3[-train\_3, ]

cov3=cov(train3)

mean3=colMeans(train3)

#class 4

class4=training.data[rownames(training.data)=="4",]

train\_4=sample(nrow(class4), size=400,replace = FALSE, prob = NULL)

train4=class4[train\_4, ]

test4=class4[-train\_4, ]

cov4=cov(train4)

mean4=colMeans(train4)

#class 5

class5=training.data[rownames(training.data)=="5",]

train\_5=sample(nrow(class5), size=400,replace = FALSE, prob = NULL)

train5=class5[train\_5, ]

test5=class5[-train\_5, ]

cov5=cov(train5)

mean5=colMeans(train5)

#class 6

class6=training.data[rownames(training.data)=="6",]

train\_6=sample(nrow(class6), size=400,replace = FALSE, prob = NULL)

train6=class6[train\_6, ]

test6=class6[-train\_6, ]

cov6=cov(train6)

mean6=colMeans(train6)

#class 7

class7=training.data[rownames(training.data)=="7",]

train\_7=sample(nrow(class7), size=400,replace = FALSE, prob = NULL)

train7=class7[train\_7, ]

test7=class7[-train\_7, ]

cov7=cov(train7)

mean7=colMeans(train7)

# class 8

class8=training.data[rownames(training.data)=="8",]

train\_8=sample(nrow(class8), size=400,replace = FALSE, prob = NULL)

train8=class8[train\_8, ]

test8=class8[-train\_8, ]

cov8=cov(train8)

mean8=colMeans(train8)

#class 9

class9=training.data[rownames(training.data)=="9",]

train\_9=sample(nrow(class9), size=400,replace = FALSE, prob = NULL)

train9=class9[train\_9, ]

test9=class9[-train\_9, ]

cov9=cov(train9)

mean9=colMeans(train9)

# we use the covariance mean as the whole same covarice

cov\_mean=(cov0+cov1+cov2+cov3+cov4+cov5+cov6+cov7+cov8+cov9)/10

test=rbind(test0,test1,test2,test3,test4,test5,test6,test7,test8,test9)

#

for(j in 1:25){

m=seq(0.02,0.5,by=0.02)

lamda=m[j]

new\_cov=(1-lamda)\*cov\_mean+lamda\*diag(1/4,nrow=400, ncol=400)

#predict the test data for every mean

pre\_test0=lda(mean0,new\_cov,test)

pre\_test1=lda(mean1,new\_cov,test)

pre\_test2=lda(mean2,new\_cov,test)

pre\_test3=lda(mean3,new\_cov,test)

pre\_test4=lda(mean4,new\_cov,test)

pre\_test5=lda(mean5,new\_cov,test)

pre\_test6=lda(mean6,new\_cov,test)

pre\_test7=lda(mean7,new\_cov,test)

pre\_test8=lda(mean8,new\_cov,test)

pre\_test9=lda(mean9,new\_cov,test)

#For every row, return the largest column index that has largest a\*x+b

pre\_test00=cbind(pre\_test0,pre\_test1,pre\_test2,pre\_test3,pre\_test4,pre\_test5,pre\_test6,pre\_test7,pre\_test8,pre\_test9)

pre\_test=apply(pre\_test00,1,which.max)

error\_rate[i,j]=sum((pre\_test-1)!=rownames(test))/nrow(test)

}

}

end\_time= proc.time()

run\_time=end\_time-start\_time

#draw a picture to find the optimal lambda

mean\_error\_rate=as.matrix(colMeans(error\_rate))

lamda\_optimal=0.02\*(apply(mean\_error\_rate,2,which.min))

m=as.matrix(m)

matplot(m,cbind(t(error\_rate),mean\_error\_rate),type='l',col=c('black','red','green','blue','yellow','purple'),ylab='error rate',xlab='lambda',lty =1,cex=2)

title(main='error rate vs lambda')

legend(legend=c('Trial 1','Trial 2','Trial 3','Trial 4','Trial 5','Average'),col=c('black','red','green','blue','yellow','purple'),'bottomright',lty=1)

abline(v=lamda\_optimal)

text(x=0.08,y=0.135,label='optimal\_lambda=0.06')

# the optimal lambda we get is 0.06 and the error rate we get of the training data is 0.125

#######################################################

# we use theoptimal lambda we get to train the whole data and do test on the test data

rownames(test.data)=test.label

#get the variance and mean

Cov02=cov(training.data[rownames(training.data)=="0",])

Cov12=cov(training.data[rownames(training.data)=="1",])

Cov22=cov(training.data[rownames(training.data)=="2",])

Cov32=cov(training.data[rownames(training.data)=="3",])

Cov42=cov(training.data[rownames(training.data)=="4",])

Cov52=cov(training.data[rownames(training.data)=="5",])

Cov62=cov(training.data[rownames(training.data)=="6",])

Cov72=cov(training.data[rownames(training.data)=="7",])

Cov82=cov(training.data[rownames(training.data)=="8",])

Cov92=cov(training.data[rownames(training.data)=="9",])

cov.training=(Cov02+Cov12+Cov22+Cov32+Cov42+Cov52+Cov62+Cov72+Cov82+Cov92)/10

cov\_new=(1-lamda\_optimal)\*cov.training+lamda\_optimal\*diag(1/4,nrow=400, ncol=400)

mean02=colMeans(training.data[rownames(training.data)=="0",])

mean12=colMeans(training.data[rownames(training.data)=="1",])

mean22=colMeans(training.data[rownames(training.data)=="2",])

mean32=colMeans(training.data[rownames(training.data)=="3",])

mean42=colMeans(training.data[rownames(training.data)=="4",])

mean52=colMeans(training.data[rownames(training.data)=="5",])

mean62=colMeans(training.data[rownames(training.data)=="6",])

mean72=colMeans(training.data[rownames(training.data)=="7",])

mean82=colMeans(training.data[rownames(training.data)=="8",])

mean92=colMeans(training.data[rownames(training.data)=="9",])

# use the similar method we calculate the errorrate

start\_time= proc.time()

pre\_test02=lda(mean02,cov\_new,test.data)

pre\_test12=lda(mean12,cov\_new,test.data)

pre\_test22=lda(mean22,cov\_new,test.data)

pre\_test32=lda(mean32,cov\_new,test.data)

pre\_test42=lda(mean42,cov\_new,test.data)

pre\_test52=lda(mean52,cov\_new,test.data)

pre\_test62=lda(mean62,cov\_new,test.data)

pre\_test72=lda(mean72,cov\_new,test.data)

pre\_test82=lda(mean82,cov\_new,test.data)

pre\_test92=lda(mean92,cov\_new,test.data)

#For every row, return the largest column index that has largest a\*x+b

pre\_test=apply(cbind(pre\_test02,pre\_test12,pre\_test22,pre\_test32,pre\_test42,pre\_test52,

pre\_test62,pre\_test72,pre\_test82,pre\_test92),1,which.max)

end\_time= proc.time()

run\_time=end\_time-start\_time

error\_rate=sum((pre\_test-1)!=rownames(test.data))/nrow(test.data)

#the errorrate we get is 0.1451