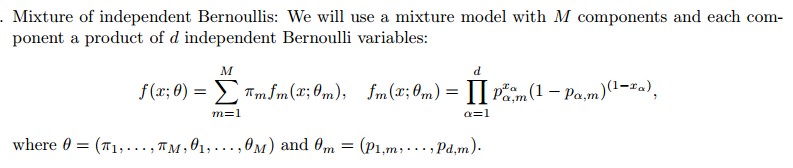
# EM Classification of the Handwritten Digits

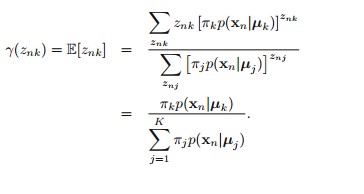
## Theory:



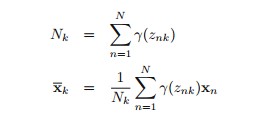
C:\Users\Christina\AppData\Roaming\Tencent\Users\919834852\QQ\WinTemp\RichOle\%9FQ}@VOUSC}OPM`0KLEFKH.jpg

C:\Users\Christina\AppData\Roaming\Tencent\Users\919834852\QQ\WinTemp\RichOle\LXI6J1QM]J%TR8{2LH5{K89.jpg

E STEP:



M STEP:



C:\Users\Christina\AppData\Roaming\Tencent\Users\919834852\QQ\WinTemp\RichOle\[2{R]5PL~S]ZY$8)1G~PFU6.jpg

C:\Users\Christina\AppData\Roaming\Tencent\Users\919834852\QQ\WinTemp\RichOle\)7P}JQ3CZU_OV{L{WKJ@G%U.jpg

#compute initial values of p and pi

p=matrix(0,d,num.class)

for(i in 1:num.class){

p[,i]=(colSums(training.data[i,,])+1)/(nrow(training.data[i,,])+2)

}

pi=matrix(0,1,num.class)

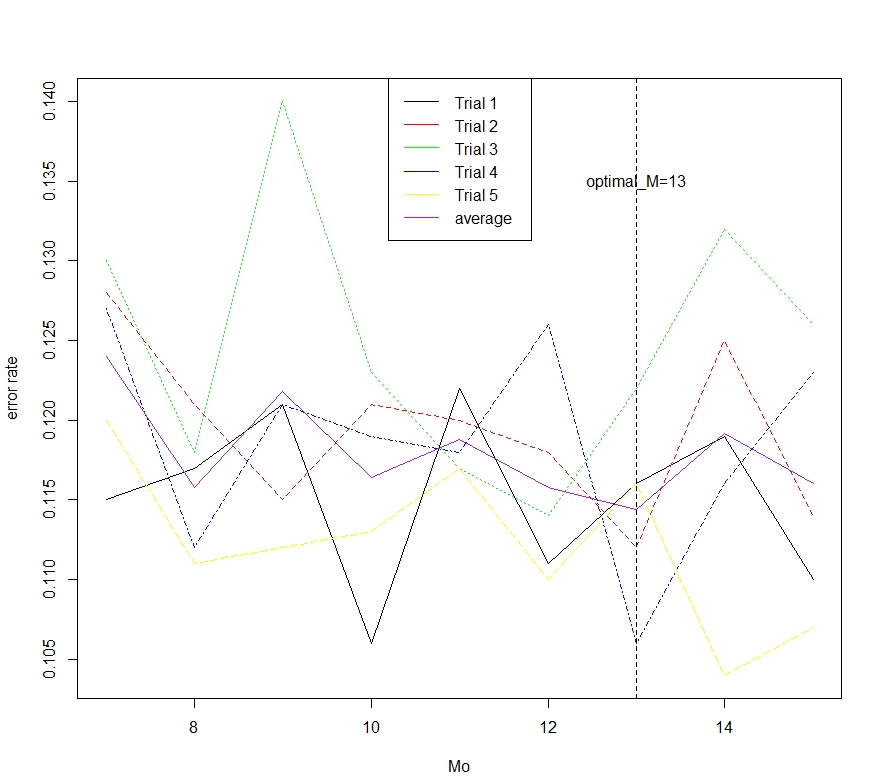
for(i in 1:num.class){

pi[,i]=(nrow(training.data[i,,])+1)/(num.class \* num.training+10)

}

C:\Users\Christina\AppData\Roaming\Tencent\Users\919834852\QQ\WinTemp\RichOle\7_(PVR2QCL1GLHR$ND}7Q30.jpgC:\Users\Christina\AppData\Roaming\Tencent\Users\919834852\QQ\WinTemp\RichOle\LU60%J24W9]ZQJL0C_7XYB0.jpg

C:\Users\Christina\AppData\Roaming\Tencent\Users\919834852\QQ\WinTemp\RichOle\NQFC5)O@[%DNIA_0NWM[L2L.jpg



> run\_time

user system elapsed

1.32 0.18 4.32

> testerror

[1] 0.1007

# Code appendix:

# EM 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

#Define Posterior function

pos=function(Pi,P,x,M)

{

N=dim(x)[1]

W=x%\*%t(log(P)) + (1-x)%\*%t(log(1-P)) + matrix(log(Pi), N, M, byrow=TRUE)

W=W-matrix(apply(W,1,max), N, M)

return(exp(W)/rowSums(exp(W)))

}

#Define function to calculate parameters for each of the ten classes

para=function(x,M)

{

N=dim(x)[1]

for (n in 1:N)

{

comp=sample(1:M,size=N,replace=TRUE)

#Initialize

P <- matrix(0,M,d)

Pi <- numeric(M)

for (m in 1:M)

{

rows <- which(comp==m)

Pi[m] <- (length(rows)+1)/(N+M)

P[m,] <- (1+colSums(x[rows,]))/(length(rows)+2)

}

}

#EM iteration

for (t in 1:1000)

{

W=pos(Pi,P,x,M) #calculate weights (P{y=m|x(n)})

S=colSums(W)

Pi\_new=(S+1)/(N+M)

P\_new=(t(W)%\*%x+1)/(S+2)

diff=norm(P\_new-P)

if (diff < 1e-6)

{

output=list(P=P\_new, Pi=Pi\_new)

break

}

else

{

P=P\_new

Pi=Pi\_new

}

}

return(output)

}

# Define predict function

fit=function(x,para,len)

{

temp = matrix(0,dim(x)[1],10)

for (k in 1:10){

P = para[[k]]$P

Pi = para[[k]]$Pi

ma = x%\*%t(log(P)) + (1-x)%\*%t(log(1-P)) + matrix(log(Pi), dim(x)[1], len, byrow=TRUE)

temp[,k]=log(rowSums(exp(ma)))

}

output = apply(temp,1,which.max)-1

return(output)

}

# Try M at 7,8,...,15

Mo=seq(7,15,1)

erate=matrix(0,5,length(Mo))

#Calculate error rates for each Mo for each of the 5 trials.

for (tt in 1:5)

{

index=NULL

ind<-vector("list", 10)

for (i in 1:10) #for each of the ten classes

{

for (j in sample(1:500,400))

{

ind[[i]]=c(ind[[i]],i+(j-1)\*10)

}

index=c(index,ind[[i]])

}

train2=training.data[-index,]

trainlab2=training.label[-index]

for (v in 1:length(Mo))

{

coe=vector("list", 10)

for (k in 1:10)

{

coe[[k]]=para(x=training.data[ind[[k]],],M=Mo[v])

}

result=(trainlab2!=fit(x=train2,para=coe,len=Mo[v]))

erate[tt,v]=sum(result)/length(result)

}

}

#Summarize the error rate.

ave.error=as.matrix(apply(erate,MARGIN=2,FUN=mean))

min(ave.error) #find the minimum of average error rate

M.min=Mo[which.min(ave.error)] #find the M that can minimize the average error rate.

matplot(Mo,cbind(t(erate),ave.error),type='l',col=c('black','red','green','blue','yellow','purple'),ylab='error rate')

legend(legend=c('Trial 1','Trial 2','Trial 3','Trial 4','Trial 5','average'),col=c('black','red','green','blue','yellow','purple'),x="top",lty=1)

abline(v=M.min,lty=2)

text(x=13,y=0.135,label='optimal\_M=13')

#Run em algorithm again at the optimal M and make predictions on testdata to compute error rate.

ind<-vector("list", 10)

for (i in 1:10) #for each of the ten classes

{

for (j in sample(1:500,500))

{

ind[[i]]=c(ind[[i]],i+(j-1)\*10)

}

}

M\_opt=13

coe=vector("list", 10)

for (k in 1:10)

{

coe[[k]]=para(x=training.data[ind[[k]],],M\_opt)

}

start\_time= proc.time()

result=(test.label!=fit(x=test.data,para=coe,len=M\_opt))

testerror=sum(result)/length(result)

end\_time= proc.time()

run\_time=end\_time-start\_time

> run\_time

user system elapsed

1.32 0.18 4.32

> testerror

[1] 0.1007