Assignment-1

Dhruv Patel and Danish Nawaz

08/09/2019

Question-1

library(mvtnorm)  
#mean of class A  
set.seed(500)  
 m\_A=c(1,0)  
#mean of class B  
 m\_B=c(0,1)  
#covariance matrix  
 cv=matrix(c(1, 0, 0, 1), nrow =2, byrow =TRUE)  
#generating points for multivariate distribution for class A and classs B  
 Cluster\_means\_A=rmvnorm(10, mean = m\_A, sigma = cv)  
 Cluster\_means\_B=rmvnorm(10, mean = m\_B, sigma = cv)  
#Generating train data using above means  
 Data\_A=data.frame()  
for (i in1:100) {  
 p=sample(1:10,1, replace =TRUE)  
 temp\_mean=as.vector(Cluster\_means\_A[p,])  
 new\_point=as.data.frame(rmvnorm(1, mean = temp\_mean, sigma = cv/5))  
 Data\_A=rbind(Data\_A, new\_point)  
 }  
 Data\_B=data.frame()  
for (i in1:100) {  
 p=sample(1:10,1, replace =TRUE)  
 temp\_mean=as.vector(Cluster\_means\_B[p,])  
 new\_point=as.data.frame(rmvnorm(1, mean = temp\_mean, sigma = cv/5))  
 Data\_B=rbind(Data\_B, new\_point)  
 }  
  
#labeling the data  
 Y=rep(1,100)  
 Data\_A=cbind(Data\_A,Y)   
 Y=rep(2,100)  
 Data\_B=cbind(Data\_B,Y)  
#binding A and B classes for training purpose  
 train\_data=rbind(Data\_A,Data\_B)  
#shuffling the data  
 train\_data<-train\_data[sample(nrow(train\_data)),]

Question-2

#Question 2  
 test\_Data\_A=data.frame()  
for (i in1:5000) {  
 p=sample(1:10,1, replace =TRUE)  
 temp\_mean=as.vector(Cluster\_means\_A[p,])  
 new\_point=as.data.frame(rmvnorm(1, mean = temp\_mean, sigma = cv/5))  
 test\_Data\_A=rbind(test\_Data\_A, new\_point)  
 }  
  
  
 test\_Data\_B=data.frame()  
for (i in1:5000) {  
 p=sample(1:10,1, replace =TRUE)  
 temp\_mean=as.vector(Cluster\_means\_B[p,])  
 new\_point=as.data.frame(rmvnorm(1, mean = temp\_mean, sigma = cv/5))  
 test\_Data\_B=rbind(test\_Data\_B, new\_point)  
 }  
  
  
 Y=rep(1,5000)  
 test\_Data\_A=cbind(test\_Data\_A,Y)   
 Y=rep(2,5000)  
 test\_Data\_B=cbind(test\_Data\_B,Y)  
#binding A and B classes for training purpose  
 test\_data=rbind(test\_Data\_A,test\_Data\_B)  
#shuffling the data  
 test\_data<-test\_data[sample(nrow(test\_data)),]

Question-3

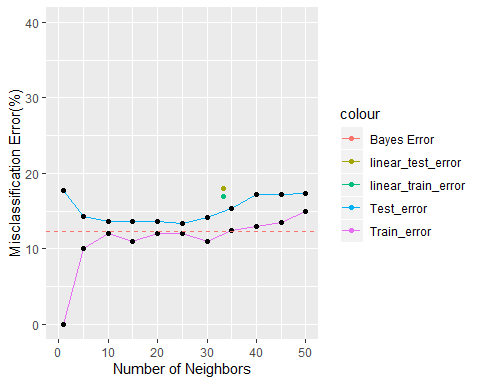
#Question 3  
  
#Knn Model  
library(class)  
  
#number of neighbours to try  
 neighbors=c(1,5,10,15,20,25,30,35,40,45,50)  
 test\_missclassification\_error=c()#(in percentage)  
 train\_missclassification\_error=c()#(in percentage)  
 accuracy <-function(x){sum(diag(x)/(sum(rowSums(x)))) \*100}  
#building model for each value of K and evaluating misclassification error  
for (i in neighbors) {  
 prd <-knn(train=train\_data[,c("V1","V2")],test=test\_data[,c("V1","V2")],cl=train\_data[,c("Y")],k=i)  
 tab <-table(prd,test\_data[,"Y"])  
 test\_missclassification\_error=append(test\_missclassification\_error,c(100-accuracy(tab)))  
 prd <-knn(train=train\_data[,c("V1","V2")],test=train\_data[,c("V1","V2")],cl=train\_data[,c("Y")],k=i)  
 tab <-table(prd,train\_data[,"Y"])  
 train\_missclassification\_error=append(train\_missclassification\_error,c(100-accuracy(tab)))  
 }  
  
  
  
#Linear Model  
 linear\_model=lm(data=train\_data,Y~V1+V2)  
summary(linear\_model)

##   
## Call:  
## lm(formula = Y ~ V1 + V2, data = train\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.60641 -0.26346 -0.06762 0.28224 0.94021   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.50747 0.02920 51.633 < 2e-16 \*\*\*  
## V1 -0.21434 0.01862 -11.509 < 2e-16 \*\*\*  
## V2 0.12511 0.02040 6.133 4.63e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3472 on 197 degrees of freedom  
## Multiple R-squared: 0.525, Adjusted R-squared: 0.5201   
## F-statistic: 108.9 on 2 and 197 DF, p-value: < 2.2e-16

labeling\_data =function(x) {   
 k=ifelse( x>1.5, 2, 1)  
return(k) }  
  
 test\_predict =predict(linear\_model, newdata = test\_data[,c("V1","V2")])   
 test\_predict=sapply(test\_predict, labeling\_data)  
 tab <-table(test\_predict,test\_data[,"Y"])  
 linear\_test\_accuracy=accuracy(tab)  
  
 train\_predict =predict(linear\_model, newdata = train\_data[,c("V1","V2")])   
 train\_predict=sapply(train\_predict, labeling\_data)  
 tab=table(train\_predict,train\_data[,"Y"])  
 linear\_train\_accuracy=accuracy(tab)

Question-4

#bayes error  
# Approximating using grid approximation here small grid value is taken as 0.1 squares  
# taking density fucntion from range -4 to +5 (10 values with 10/100=0.1 squares)  
set.seed(500)  
A <-matrix(rep(0,100\*100),100)  
B <-matrix(rep(0,100\*100),100)  
Range\_X <-seq(-4,5,length.out=100)  
Range\_Y<-seq(-4,5,length.out=100)  
  
for (i in1:100) {  
for (j in1:100) {  
for (k in1:10)  
 {  
 A[i,j] =A[i,j]+0.1\*dmvnorm(c(Range\_X[i],Range\_Y[j]),Cluster\_means\_A[k,],cv/5)  
 B[i,j] =B[i,j]+0.1\*dmvnorm(c(Range\_X[i],Range\_Y[j]),Cluster\_means\_B[k,],cv/5)  
 }  
 }  
}  
  
# doing the integration to obtain the bayes error rate  
dx=Range\_X[2]-Range\_X[1]  
dy=Range\_Y[2]-Range\_Y[1]  
prob=0  
  
for (i in1:100) {  
for (j in1:100) {  
 prob =prob +0.5\*max(A[i,j],B[i,j])\*dx\*dy  
 }  
  
}  
bayes\_error=(1-prob)\*100  
#plotting misclassification Error vs Number of Neighbors(k) for KNN models and also Adding errors of linear model in same graph  
library(ggplot2)  
ggplot() +geom\_line(aes(x=neighbors, y=train\_missclassification\_error, color="Train\_error")) +geom\_point(aes(x=neighbors, y=train\_missclassification\_error))+  
scale\_x\_continuous(name ="Number of Neighbors", limits =c(0, 50)) +scale\_y\_continuous(name ="Misclassification Error(%)", limits =c(0, 40))+  
geom\_line(aes(x=neighbors, y=test\_missclassification\_error, color="Test\_error"))+geom\_point(aes(x=neighbors, y=test\_missclassification\_error)) +  
geom\_point(aes(x=33.33, y=100-linear\_train\_accuracy, color="linear\_test\_error")) +geom\_point(aes(x=33.33, y=100-linear\_test\_accuracy, color="linear\_train\_error"))+geom\_hline(aes(yintercept=bayes\_error ,colour="Bayes Error"), linetype="dashed")



Question 5:

Observations:

-For k=1 (DOF= 200), the trainig error is zero as it will highly overfit the data and will take the point itself as its neighbour, thereby minimizing the error.

-Near k=20-25 (DOF=8-10), training and test errors both become optimal and this is the ideal value to use as K as infered from graph. Reason behind this can be understood intuitively as we generated the data using 10+10=20 different means, so resulting distribution should be explained by 20 clusters means.

-As this data is generated by individual means and their distributions, the KNN model gives better results then linear model. Had it been generated by two bivariate gaussian distributions, the linear model would have outperfomed the Knn model!