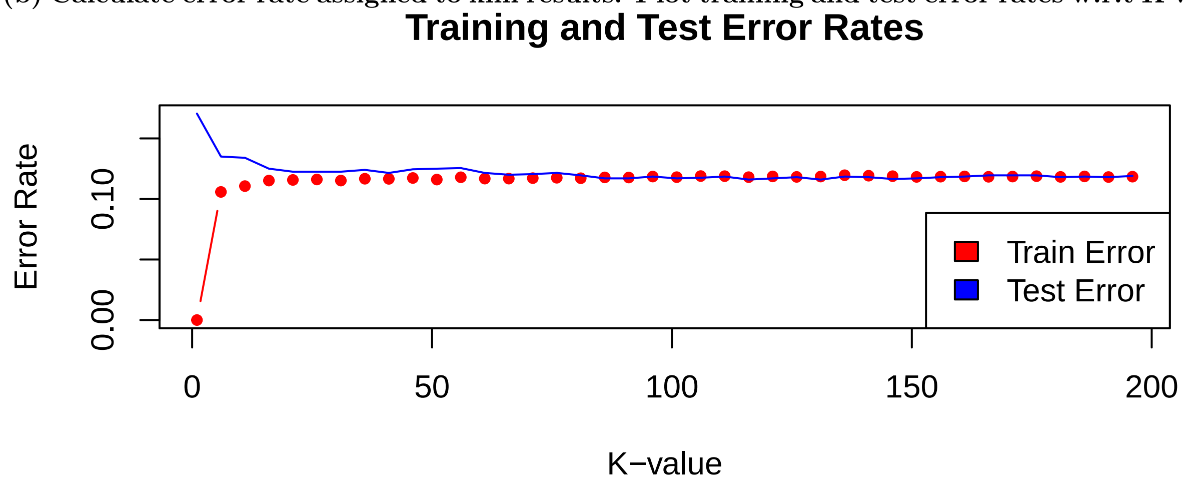
MiniProj1

Wenxiong Lu

1

(a) Firstly, fit a Knn with K = 1,6,. . . ,196. (for constant output, we set seed=1)

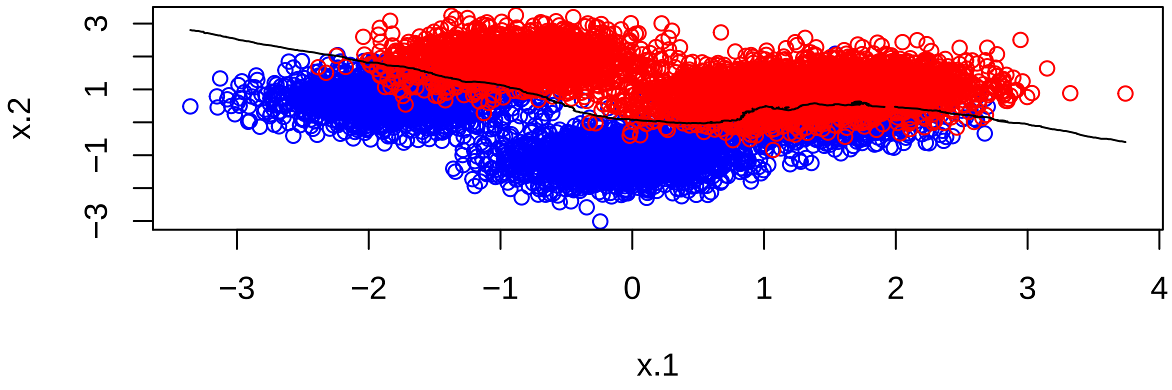
(b) Calculate error rate assigned to knn results. Plot training and test error rates w.r.t K value.



Comment: As show in the figure, the Training Error Rate increase when K is about 1~6 and keep increasing as K goes up. On the other hand, the Test Error Rates are decreasing in the big picture. (There are some fluctuation) Over all it is consistent with the observation on the course.

(c) For the Training Error Rate knn is optimal at k=6, the associated error is 0. For the Test Error Rate k = 116 is the optimal k value with Test Error Rate = 0.116 .

(d) Plot traning data with decision boundary for optimal K = 6.



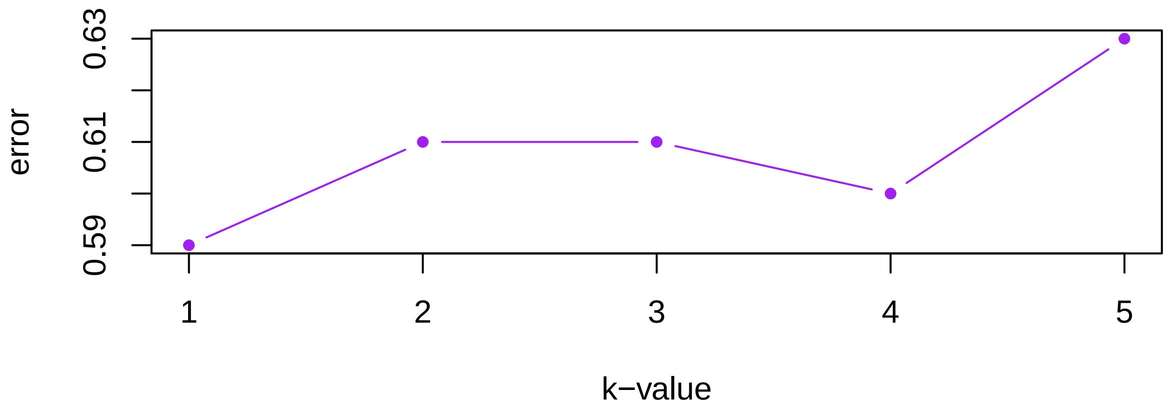
#####Comment: The decision boundary is the black line. As one can see the decision boundary is quite sensible to differ the two category of red and blue dots(which are results 'yes' and 'no').

####2

#####(a) Firstly, fit KNN with K = 50, 100, 200, 300, 400. Exam the test error rate.

```
## [1] 0.59 0.61 0.61 0.60 0.63
```

#####(b) The best value of K is k=1 here. For k=1, the confusion matrix is following:



```
## [1] 50
```

```
##       result
## y.test  0  1  2  3  4  5  6  7  8  9
##      0  3  0  1  0  0  0  1  0  1  0
##      1  1  1  1  0  2  0  1  0  1  1
##      2  1  0  2  0  1  1  0  0  0  0
##      3  0  1  2  2  5  1  0  0  0  0
##      4  1  0  3  0  7  0  1  0  0  0
##      5  0  0  2  0  1  2  2  0  1  1
##      6  0  0  1  0  5  0  6  0  0  0
##      7  0  0  1  1  2  1  0  1  1  0
##      8  0  0  1  0  2  1  0  0 14  1
##      9  1  0  2  0  4  0  0  0  1  3
```

Comment: The confusion matrix is a 10X10 matrix because there are 10 categories of classification objects. It is a complicated matrix. #####(c) The using knn classification in this case is not good idea. Since there are 10 categories of objects, the data dimension is 10, which require large computational power of bayes probabilities for each data (Remember each data point is consisted by three layers color-data and a positional data).

Section 2 R code

```
getwd()
setwd("/Users/wenxionglu/Documents/GraduateCourse/6340 Stats/proj1")
test=read.csv("1-test_data.csv",header=T)
train = read.csv("1-training_data.csv",header=T)
library(class)
#drop unnecessary columns
test.X = test[,-3]
test.Y = test[,3]
```

```r
train.X = train[,-3]
train.Y = train[,3]
##test.Y<-sapply(test.Y,function(x) if(x=='yes') x<-1 else x<-0)
##train.Y<-sapply(train.Y,function(x) if(x=='yes') x<-1 else x<-0)

Kset=seq(1,200,by=5)
Kset
#Tuning knn k parameter
                          #Training Error
Nset=rep(0,200)                   # declaration to initiate for loop
Nset
set.seed(1)
for (i in Kset){
  knn.mod <-  knn(train.X, train.X, cl=train.Y, k=i)
  Nset[i] <- sum(train.Y == knn.mod)/NROW(train.Y)
  k=i
  cat(k,'=',Nset[k],'\n')       # to print % accuracy
}
#The optimal K and its Error rate
1-max(Nset)
match(max(Nset[-1]),Nset)
#Plot max Error Rate
plot(Kset,1-Nset[Kset], xlab="K- Value",ylab="Error Rate")  # to plot % accuracy with respect to k-valu
table(knn.mod,train.Y)
mean(train.Y!=knn.mod)
                          #Test Error
set.seed(1)
Mset=rep(0,200)
for(i in Kset){
  knn.mod2 <- knn(train.X,test.X,cl=train.Y, k = i)
  Mset[i] <- sum(test.Y == knn.mod2)/NROW(test.Y)
  k=i
  cat(k, '=', Mset[k],'\n')
}
#The optimal K and its Error rate
1-max(Mset)
match(max(Mset[-1]),Mset)
#Plot Test Error Rate
plot(Kset,1-Mset[Kset], xlab="K- Value",ylab="Error Rate")
table(knn.mod2,test.Y)
mean(test.Y!=knn.mod2)


#Plot both error rates
plot(Kset, 1-Nset[Kset],
     main = 'Training and Test Error Rates',
     ylab = 'Error Rate', xlab= 'K-value',
     type = 'b', col = 'red',pch = 20,ylim = range(c(1-Nset[Kset],1-Mset[Kset])))
lines( Kset,1-Mset[Kset], col= 'blue',pch=20)
legend('bottomright',
      c("Train Error","Test Error"),
      fill=c("red","blue")
      )
```

```r
#decision boundary
n.grid<-500  #set up grid
x1.grid<-seq(f=min(train.X[,1]),t=max(train.X[,1]),l=n.grid)
x2.grid<-seq(f=min(train.X[,2]),t=max(train.X[,2]),l=n.grid)
grid<-expand.grid(x1.grid,x2.grid)

k.opt<-match(max(Mset[-1]),Mset)  #k=116
set.seed(1)
mod.opt<-knn(train.X,grid,train.Y,k=k.opt,prob=T)
prob<-attr(mod.opt,'prob')
prob<-ifelse(mod.opt=='yes',prob,1-prob) #assign probability for each grid pixel
prob<-matrix(prob,n.grid,n.grid)

#draw the picture and decision boundary
plot(train.X, col=ifelse(train.Y == 'yes','blue','red'))
contour(x1.grid,x2.grid,prob,levels=0.5,labels ='', xlab='',ylab='',main='',add=T)




#####
#(2)

library(keras)
cifar <- dataset_cifar10()
str(cifar)
x.train <- cifar$train$x
y.train <- cifar$train$y
x.test <- cifar$test$x
y.test <- cifar$test$y
# reshape the images as vectors (column-wise)
# (aka flatten or convert into wide format)
# (for row-wise reshaping, see ?array_reshape)
dim(x.train) <- c(nrow(x.train), 32*32*3) # 50000 x 3072
dim(x.test) <- c(nrow(x.test), 32*32*3) # 50000 x 3072
# rescale the x to lie between 0 and 1
x.train <- x.train/255
x.test <- x.test/255
# categorize the response
y.train <- as.factor(y.train)
y.test <- as.factor(y.test)
# randomly sample 1/100 of test data to reduce computing time
set.seed(2021)
id.test <- sample(1:10000, 100)
x.test <- x.test[id.test,]
y.test <- y.test[id.test]

#fit knn model
knum <- c(50,100,200,300,400)
knn.pic <- sapply(knum, function (k) knn(x.train, x.test, y.train, k))
test.err.rate<-sapply(c(1:5),function(c) mean(knn.pic[,c]!=y.test)) #calculate the error rate for 5 dif
test.err.rate
```

```r
#find best k value
plot(test.err.rate,ylab='error',xlab='k-value',pch=20,type='b',col='purple')
x<-match(min(test.err.rate),test.err.rate)
#the optimal k value is:
knum[x]
result<-knn.pic[,x]
t<-table(y.test,result)
sum(diag(t))
```