STAT 489 HW1

Nicholas Thompson

9/6/2021

#Question 1  
euclid.dist= function(u,v){  
 n=length(u)  
 N=1:n  
 sqrt(sum((u[N]-v[N])^2))  
}  
  
euclid.dist(u=c(0,0,0,1), v=c(2,5,2,4))

## [1] 6.480741

sqrt(42) #value to compare

## [1] 6.480741

#Question 2  
set.seed(42)  
attach(iris)  
#create a function to find k nearest neighbors  
neighbors=function(data,t,k){  
 #one parameter of the distance function will be t  
 #Call Euclidean distance function  
 dist=apply(data, 1,euclid.dist, t)  
   
 #Find closest neighbors  
 distances=sort(dist)  
 kdist=distances[1:k]  
 neighbor.index=which(dist %in% kdist)  
 kclosest=list(neighbor.index)  
   
 return(kclosest)  
}  
  
#Run a test on function to see if function works. Run test to find nearest neighbors of the last row of iris, set input k to 4  
trial.train=sample(1:150,110)  
trial.data=iris[trial.train,]  
trial.vector=iris[-trial.train,]  
  
q2.test= neighbors(trial.data[,1:4],trial.vector[3,1:4],4)[[1]] #we'll use 3rd row of test data  
q2.test

## [1] 20 32 37 47

as.matrix(trial.data[q2.test,])

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species   
## 27 "5.0" "3.4" "1.6" "0.4" "setosa"  
## 30 "4.7" "3.2" "1.6" "0.2" "setosa"  
## 8 "5.0" "3.4" "1.5" "0.2" "setosa"  
## 50 "5.0" "3.3" "1.4" "0.2" "setosa"

#compare to  
trial.vector[3,1:4] #excluding fifth column for prediction reasons

## Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 12 4.8 3.4 1.6 0.2

#Question 3  
  
#Prediction class function  
KNN= function(train,y){  
 temp.table=table(train[,y])  
 pred.class=temp.table[temp.table==max(temp.table)]  
 pred.class=unlist(attributes(pred.class), use.names = FALSE) #remove the name from max, leaving just a character  
 return(pred.class)  
}  
q3.test=KNN(trial.data[q2.test,], 'Species')  
q3.test

## [1] "setosa"

trial.vector[3,5] #compare to the actual test species

## [1] setosa  
## Levels: setosa versicolor virginica

#Question 4  
set.seed(888)  
train.inx=sample(1:150,120)  
train.iris=iris[train.inx,] #80% training data  
test=iris[-train.inx,]  
  
#After consideration, I created a function combining all 3 previous, as well as use it to address a potential situation  
knn.alg=function(train,t,k,y){  
 predictions=c()  
 for(i in 1:nrow(t)){  
 kneighbors=neighbors(train[,1:4],t[i,1:4],k)[[1]]  
 pred=KNN(train[kneighbors,],y)  
 if (length(pred)>1){ #This is to prevent a case where 2 or more factors have the maximum  
 pred=knn.alg(train, t[i,1:4],k=k+1,y)  
 }  
   
 predictions[i]=pred  
 }  
   
 return(predictions)  
   
}  
q4=knn.alg(train.iris,test, 3,'Species')#take k=3  
q4

## [1] "setosa" "setosa" "setosa" "versicolor" "versicolor"  
## [6] "versicolor" "versicolor" "versicolor" "versicolor" "versicolor"  
## [11] "versicolor" "versicolor" "virginica" "versicolor" "versicolor"  
## [16] "versicolor" "versicolor" "virginica" "virginica" "virginica"   
## [21] "virginica" "virginica" "virginica" "virginica" "virginica"   
## [26] "virginica" "virginica" "virginica" "virginica" "virginica"

mean(Species[30]!=q4)

## [1] 0.9

#Testing error rate is 0.9  
#Create a confusion matrix using the caret package to assess the model accuracy  
realq4data=factor(q4,levels = levels(test$Species))  
confusionMatrix(as.factor(realq4data), test$Species)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction setosa versicolor virginica  
## setosa 3 0 0  
## versicolor 0 13 0  
## virginica 0 1 13  
##   
## Overall Statistics  
##   
## Accuracy : 0.9667   
## 95% CI : (0.8278, 0.9992)  
## No Information Rate : 0.4667   
## P-Value [Acc > NIR] : 4.148e-09   
##   
## Kappa : 0.9431   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: setosa Class: versicolor Class: virginica  
## Sensitivity 1.0 0.9286 1.0000  
## Specificity 1.0 1.0000 0.9412  
## Pos Pred Value 1.0 1.0000 0.9286  
## Neg Pred Value 1.0 0.9412 1.0000  
## Prevalence 0.1 0.4667 0.4333  
## Detection Rate 0.1 0.4333 0.4333  
## Detection Prevalence 0.1 0.4333 0.4667  
## Balanced Accuracy 1.0 0.9643 0.9706

#Model accuracy is 96.67%

#Question 5  
#Let's consider 2 different k values  
q5=knn.alg(train.iris,test,6,'Species')#take k=6  
str(q5)

## chr [1:30] "setosa" "setosa" "setosa" "versicolor" "versicolor" ...

mean(Species[30]!=q5)

## [1] 0.9

#testing error is the same as when k=3  
  
#Take k=6 for prediction of the training labels  
q5.train=knn.alg(train.iris,train.iris,6,'Species')  
str(q5.train)

## chr [1:120] "virginica" "setosa" "versicolor" "setosa" "virginica" ...

mean(Species[120] !=q5.train) #training error rate

## [1] 0.6833333

#0.683333  
  
#Let's try a bigger k, say k=20  
q20=knn.alg(train.iris,test,20,'Species')  
mean(Species[30] !=q20)

## [1] 0.9

q20.train=knn.alg(train.iris,test,20,'Species')  
mean(Species[120] !=q20.train)

## [1] 0.5333333

#0.5333  
  
#Perhaps, lowering the k to 1  
lowerk=knn.alg(train.iris,test,1,'Species')  
mean(Species[30] !=lowerk)

## [1] 0.9

lowerk.train=knn.alg(train.iris,train.iris,1,'Species')  
mean(Species[120] !=lowerk.train)

## [1] 0.6916667

#Training error of 0.691667  
  
#So, no matter what k we choose, we achieve the same test error rate. But the training error decreases when k increases.