Homework 2 (Problem 1)

MATH 6333 Statistical Learning

Md Salman Rahman

Instructor: Dr. Tamer Oraby

Problem 1: Write a detailed report to explain every step in the file Lab 1 - R.R that was covered in the third week.

Solution:

Code line 84 and 85: ##################Part 2##################

K-NN

Explanation: # means comments in R. This indicates the comments of the code.

Code line 86: library(class)

Explanation: it is a library, and it is called so that we can use knn from this library.

Code line 87 to 88: KNN<-knn(DF_mix_data[,1:2],DF_mix_data[,1:2],DF_mix_data[,3],

k=15, prob=TRUE,use.all=TRUE)

Explanation: knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE) this is the basic structure of knn. Here first DF_mix_data[,1:2] is training data, and second DF_mix_data[,1:2], is testing data, and c1= DF_mix_data[,3] is factor of true classifications of training set, and k=15 is 15 closest neighbour is considered here, prob =TRUE means the proportion of the votes for the winning class are returned as attribute prob, use.all means means controls handling of ties and use.all=TRUE means controls handling of ties all distances equal to the kth largest are included. If false, a random selection of distances equal to the kth is chosen to use exactly k neighbours

Code line 90: mean(KNN != DF_mix_data[,3]) #misclassificiation error

Explanation: This is for finding misclassification error. KNN! = DF_mix_data[,3] it finds how many of DF_mix_data[,3] data frame are not equal to KNN. And we find the mean misclassification error by using the mean function.

Code line 92 to 93: KNN2<-knn(DF_mix_data[,1:2],DF_M,DF_mix_data[,3],

k=15, prob=TRUE,use.all=TRUE)

Explanation: We want to color whole mesh and we know the mesh is given by DF_M. We change the middle DF_mix_data[,1:2] to DF_M. It will make prediction for each mesh point. This will use for coloring.

Code line 94: new_M_KNN<-cbind.data.frame(DF_M,KNN2)

Explanation: cbind.data.frame(DF_M,KNN2) will put together the input of the mesh DF_M and its prediction KNN2.

Code line 96 to 97: p6=p4+geom_point(data=new_M_KNN[,1:2],aes(x=cx,y=cy),

colour= ifelse(new_M_KNN[, 3]==1,"orange","blue"), alpha=.01)

Explanation: making plot using ggplot library

We define p4 earlier and we will combine p4 with geometric point of new_M_KNN data;

geom_point(): this is for plotting geometric plot of new_M_KNN data ;

aes: this is for axis, and here on the x axis it will be cx, and on the y axis it will be cy;

colour= ifelse(new_M_KNN[, 3]==1,"orange","blue"): the colouring is done by if else statement; here if the new_M_KNN[,3]==1 (if the third column is equal to 1) condition is true then it will colour as "orange", and if the statement is false then colour will be "blue".

Here, orange and blue may overlap so that we use alpha =0.01 which will make the color lighter.

Code line 99: p6

Explanation: calling the p6 variable.

Code line 101: pr1<-attr(KNN2,"prob")

Explanation: pr1 indicate probability 1. attr stands for attribute and "prob" is asking for attribute of probability.

Code line 102: pr2<-ifelse(KNN2=="1",1-pr1,pr1)

Explanation: we want to switch the probability, here if KNN2="1" then it will switch to 1- pr1, otherwise pr1.

Code line 103: DF_M_KNN<-cbind(DF_M,pr2)

Explanation: cbind(DF_M,pr2) this will bind DF_M and probability_2 and give a matrix.

Code line 105 to 107: p7=p6+

geom_contour(data=DF_M_KNN,aes(x=cx,y=cy,z=pr2),

binwidth=0.51,lwd=.5,colour="black")

Explanation: making plot using ggplot library

We define p6 earlier and we will combine p6 with contour of DF_M_KNN data;

geom_contour(): this is for plotting contour of data DF_M_KNN data ;

aes: this is for axis, and here on the x axis it will be cx, and on the y axis it will be cy, and z will made by pr2;

binwidth=0.51; binwidth give the contour bins. we setup the binwidth just to get the exact number of contour. Split up as 0.51 is used because , below 0.5 will go to blue.

lwd=.5, line width is 0.5

colour="black"; colour of contour is black.

Code line 108: p7

Explanation: calling the p7 variable.

Code line 110: ggsave("p7.png",plot=p7)

Explanation: saving the p7 plot as png using ggsave

Code line 112: library(png)

Explanation: it is a library, and it is used to open any png image.

Code line 113: img<-readPNG("p7.png")

Explanation: this will read p7.png figure using readPNG

Code line 114: grid::grid.raster(img)

Explanation: this will make better figure. rendering a raster object (bitmap image) at the given location, size, and orientation.

Code line 116 &118: ############ Cross-Validation

Monte-Carlo CV for K-NN

Explanation: # means comments in R. This indicates the comments of the code.

Code line 120: library(class)

Explanation: it is a library, and it is called so that we can use knn from this library.

Code line 122 to 132: Find_K<-function(K,M=1000){

```
 MSE<-0
 for(i in 1:M){
  train<-sample(200,200-40) # for 5-fold CV
  KNN<-knn(DF_mix_data[train,1:2], DF_mix_data[-train,1:2], DF_mix_data[train,3],
       k=K,prob=TRUE,use.all=FALSE)
  MSE<-MSE+mean(KNN != DF_mix_data[-train,3])
 }
 CVK<-MSE/M
 return(CVK)
}
```

Explanation:

functions provide the base mechanisms for defining new functions in the R language. Here we want to build a function that search for best k. We give the function name as Find_K.

function(K,M=1000){

}: the function will take input of K and check the statement inside the curly braces, function can take more than one input for example M is another default input.

MSE<-0: We assign 0 to MSE and it will updated using for loop.

for(i in 1:M){

}: the for loop will run from 1 to M.

train<-sample(200,200-40) # for 5-fold CV: training data is randomly sampled from 200 points

KNN<-knn(DF_mix_data[train,1:2], DF_mix_data[-train,1:2], DF_mix_data[train,3],

k=K,prob=TRUE,use.all=FALSE)

MSE<-MSE+mean(KNN != DF_mix_data[-train,3])

: we run knn algorithm here (we discuss this in earlier also), Here first DF_mix_data[train,1:2] is training data, and second DF_mix_data[-train,1:2], is testing data and -train means non trained data, and

DF_mix_data[train,3] is factor of true classifications of training set, and k=K is K closest neighbour is considered here, prob =TRUE means the proportion of the votes for the winning class are returned as attribute prob, use.all means means controls handling of ties and use.all=TRUE means controls handling of ties all distances equal to the kth largest are included. If false, a random selection of distances equal to the kth is chosen to use exactly k neighbours.

mean(KNN != DF_mix_data[-train,3]): This is for finding misclassification error. KNN! = DF_mix_data[,3] it finds how many of DF_mix_data[-train,3] data frame are not equal to KNN. And we find the mean misclassification error by using the mean function.

MSE<-MSE+mean(KNN != DF_mix_data[-train,3]): it will update the MSE with misclassification error

CVK<-MSE/M: CVK is cross validation is sum of MSE over how many runs for example here M=1000

return(CVK): It will return CVK

Code line 134: Find_K(15)

Explanation: Finding K=15 using the above defined function

Code line 136: library("purrr")

Explanation: it is a functional Programming tools library

Code line 138: LK<-1:30

Explanation: LK stand for list for K and the list is from 1 to 30.

Code line 139 to 140: VCVK<-LK %>%

map(function(K) Find_K(K))

Explanation: LK %>% apply maping function in this (function(K) Find_K(K))

Code line 142: VCVK<-unlist(vCVK)

Explanation: unlist provides a flatten Lists, and here unlist is applied in vCVK.

Code line 143: VCVK<-as.data.frame(vCVK)

Explanation: converting vCVK into a data frame.

Code line 144: colnames(vCVK)<-c("CVK")

Explanation: We assigned the column names as “CVK”

Code line 146 to 149: ggplot()+

geom_point(data=vCVK,aes(x=LK,y=CVK))+

xlab("K")+

ylab("CV")

Explanation: making plot using ggplot library

geom_point(): this is for plotting geometric plot of vCVK ;

aes: this is for axis, and here on the x axis it will be LK, and on the y axis it will be CVK;

xlab("K"): label of x axis is defined as K

ylab("CV"): label of y axis is defined as CVK

Code line 151: LK[which.min(vCVK$CVK)]

Explanation: finding minimum k value

Code line 153: ############ LOOCV and Kfold-CV for K-NN

Explanation: We will check leave one out cross-validation

Code line 155: CV<-
knn.cv(train=DF_mix_data[,1:2],as.factor(DF_mix_data[,3]),k=11,prob=FALSE,use.all=TRUE)

Explanation: Here knn.cv() represent k-Nearest Neighbour cross-validatory classification and the training data is DF_mix_data[,1:2],

as.factor(DF_mix_data[,3]): transforming 1 and 0 into factors and the level will be 1 and 0

and k=11 is 11 closest neighbour is considered here,

prob =TRUE means the proportion of the votes for the winning class are returned as attribute prob,
use.all means means controls handling of ties

use.all=TRUE means controls handling of ties all distances equal to the kth largest are included. If false, a random selection of distances equal to the kth is chosen to use exactly k neighbours

Code line 157: mean(as.numeric(CV)-DF_mix_data[,3])

Explanation: as.numeric(CV) will transform into numeric value and finding the mean of mean(as.numeric(CV)-DF_mix_data[,3])

##################End of file##################