ASSIGNMENT 9 Exercise 15: Introduction to Machine Learning

Abhijit Mandal

2020-10-31

## Assignment

**In this problem, you will use the nearest neighbors algorithm to fit a model on two simplified datasets. The first dataset (found in binary-classifier-data.csv) contains three variables; label, x, and y. The label variable is either 0 or 1 and is the output we want to predict using the x and y variables. The second dataset (found in trinary-classifier-data.csv) is similar to the first dataset except that the label variable can be 0, 1, or 2. Note that in real-world datasets, your labels are usually not numbers, but text-based descriptions of the categories (e.g. spam or ham). In practice, you will encode categorical variables into numeric values.**

## Question A:

**Plot the data from each dataset using a scatter plot.**

## Answer for A

## Code

## Set the working directory to the root of your DSC 520 directory  
  
setwd("~/Documents/GitHub/dsc520")  
  
## Load the `binary-classifier-data.csv`  
classifierdata.binary <- read.csv("data/binary-classifier-data.csv")  
head(classifierdata.binary)

## label x y  
## 1 0 70.88469 83.17702  
## 2 0 74.97176 87.92922  
## 3 0 73.78333 92.20325  
## 4 0 66.40747 81.10617  
## 5 0 69.07399 84.53739  
## 6 0 72.23616 86.38403

summary(classifierdata.binary)

## label x y   
## Min. :0.000 Min. : -5.20 Min. : -4.019   
## 1st Qu.:0.000 1st Qu.: 19.77 1st Qu.: 21.207   
## Median :0.000 Median : 41.76 Median : 44.632   
## Mean :0.488 Mean : 45.07 Mean : 45.011   
## 3rd Qu.:1.000 3rd Qu.: 66.39 3rd Qu.: 68.698   
## Max. :1.000 Max. :104.58 Max. :106.896

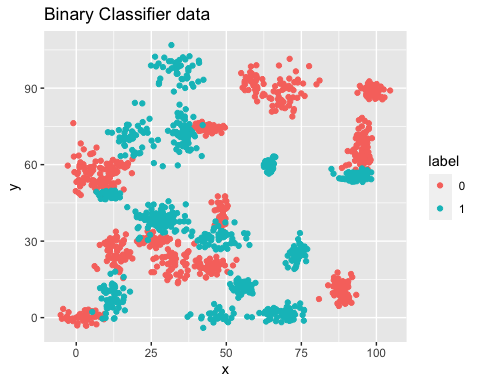
## Load the `trinary-classifier-data.csv`  
classifierdata.trinary <- read.csv("data/trinary-classifier-data.csv")  
head(classifierdata.trinary)

## label x y  
## 1 0 30.08387 39.63094  
## 2 0 31.27613 51.77511  
## 3 0 34.12138 49.27575  
## 4 0 32.58222 41.23300  
## 5 0 34.65069 45.47956  
## 6 0 33.80513 44.24656

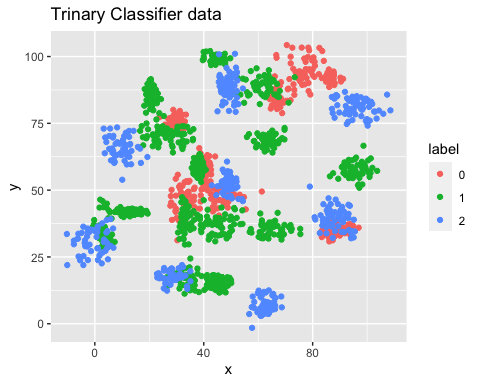
summary(classifierdata.trinary)

## label x y   
## Min. :0.000 Min. :-10.26 Min. : -1.541   
## 1st Qu.:0.000 1st Qu.: 31.15 1st Qu.: 35.906   
## Median :1.000 Median : 45.59 Median : 55.073   
## Mean :1.037 Mean : 48.86 Mean : 55.282   
## 3rd Qu.:2.000 3rd Qu.: 66.27 3rd Qu.: 77.403   
## Max. :2.000 Max. :108.56 Max. :104.293

classifierdata.binary$label <- as.factor(classifierdata.binary$label)  
classifierdata.trinary$label <- as.factor(classifierdata.trinary$label)  
  
library(ggplot2)  
ggplot(data = classifierdata.binary, aes(y = y, x = x, color = label)) +   
geom\_point() + ggtitle("Binary Classifier data")



ggplot(data = classifierdata.trinary, aes(y = y, x = x, color = label)) +   
geom\_point() + ggtitle("Trinary Classifier data")



## Question B.

**Fit a k nearest neighbors model for each dataset for k=3, k=5, k=10, k=15, k=20, and k=25. Compute the accuracy of the resulting models for each value of k. Plot the results in a graph where the x-axis is the different values of k and the y-axis is the accuracy of the model.**

## Answer for B:

## Binary Classifier - Select data points, normalize and Split 80-20 into train and test datasets.

#normalize function  
normalize<-function(x){  
 return (  
 (x - min(x))/max(x)-min(x)  
 )  
}  
  
classifierdata.normalizedbinary<-as.data.frame(lapply(classifierdata.binary[,c(2:3)], normalize))  
head(classifierdata.normalizedbinary)

## x y  
## 1 5.928057 4.835151  
## 2 5.967139 4.879608  
## 3 5.955775 4.919591  
## 4 5.885243 4.815779  
## 5 5.910742 4.847877  
## 6 5.940980 4.865152

str(classifierdata.normalizedbinary)

## 'data.frame': 1498 obs. of 2 variables:  
## $ x: num 5.93 5.97 5.96 5.89 5.91 ...  
## $ y: num 4.84 4.88 4.92 4.82 4.85 ...

summary(classifierdata.normalizedbinary)

## x y   
## Min. :5.200 Min. :4.019   
## 1st Qu.:5.439 1st Qu.:4.255   
## Median :5.650 Median :4.475   
## Mean :5.681 Mean :4.478   
## 3rd Qu.:5.885 3rd Qu.:4.700   
## Max. :6.250 Max. :5.057

#Select train and test data  
  
data.d = sample(1:nrow(classifierdata.normalizedbinary), size= nrow(classifierdata.normalizedbinary) \*0.8, replace = FALSE)  
traindata.binary <- classifierdata.binary[data.d,] #80% training data  
testdata\_binary <- classifierdata.binary[-data.d,] #20% training data  
  
#Create separate dataframes for train and test data for label   
traindata\_binary\_df<-classifierdata.binary[data.d,1]  
testdata\_binary\_df<-classifierdata.binary[-data.d,1]  
  
k\_valuebinary<-round(sqrt(nrow(classifierdata.binary)))  
  
library(class)  
  
# find no of observations  
NROW(traindata\_binary\_df)

## [1] 1198

#define result data frames  
k\_binarydata <- c()  
accurary\_binarydata <- c()  
  
# Calculate KNN for K=3  
knn.3 <- knn(train = traindata.binary , test = testdata\_binary, cl = traindata\_binary\_df, k = 3)  
# Calculate Accuracy  
acc.3 <- 100\* sum(testdata\_binary\_df == knn.3)/NROW(testdata\_binary\_df)  
acc.3

## [1] 97.33333

k\_binarydata <- c(k\_binarydata, 3)  
accurary\_binarydata<- c(accurary\_binarydata, acc.3)  
  
# Calculate KNN for K=5  
knn.5 <- knn(train = traindata.binary , test = testdata\_binary, cl = traindata\_binary\_df, k = 5)  
# Calculate Accuracy  
acc.5 <- 100\* sum(testdata\_binary\_df == knn.5)/NROW(testdata\_binary\_df)  
acc.5

## [1] 97.33333

k\_binarydata <- c(k\_binarydata, 5)  
accurary\_binarydata<- c(accurary\_binarydata, acc.5)  
  
# Calculate KNN for K=10  
knn.10 <- knn(train = traindata.binary , test = testdata\_binary, cl = traindata\_binary\_df, k = 10)  
# Calculate Accuracy  
acc.10 <- 100\* sum(testdata\_binary\_df == knn.10)/NROW(testdata\_binary\_df)  
acc.10

## [1] 97.33333

k\_binarydata <- c(k\_binarydata, 10)  
accurary\_binarydata<- c(accurary\_binarydata, acc.10)  
  
# Calculate KNN for K=15  
knn.15 <- knn(train = traindata.binary , test = testdata\_binary, cl = traindata\_binary\_df, k = 15)  
# Calculate Accuracy  
acc.15 <- 100\* sum(testdata\_binary\_df == knn.15)/NROW(testdata\_binary\_df)  
acc.15

## [1] 97

k\_binarydata <- c(k\_binarydata, 15)  
accurary\_binarydata<- c(accurary\_binarydata, acc.15)  
  
# Calculate KNN for K=20  
knn.20 <- knn(train = traindata.binary , test = testdata\_binary, cl = traindata\_binary\_df, k = 20)  
# Calculate Accuracy  
acc.20 <- 100\* sum(testdata\_binary\_df == knn.20)/NROW(testdata\_binary\_df)  
acc.20

## [1] 97

k\_binarydata <- c(k\_binarydata, 20)  
accurary\_binarydata<- c(accurary\_binarydata, acc.20)  
  
# Calculate KNN for K=25  
knn.25 <- knn(train = traindata.binary , test = testdata\_binary, cl = traindata\_binary\_df, k = 25)  
# Calculate Accuracy  
acc.25 <- 100\* sum(testdata\_binary\_df == knn.25)/NROW(testdata\_binary\_df)  
acc.25

## [1] 97.33333

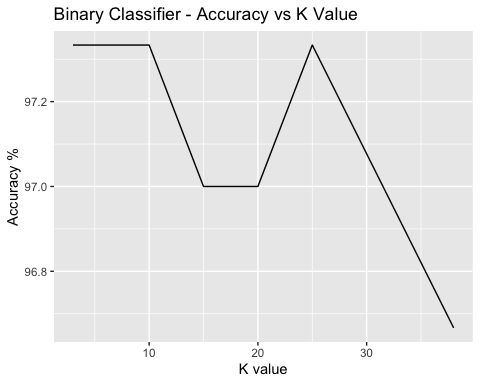
k\_binarydata <- c(k\_binarydata, 25)  
accurary\_binarydata<- c(accurary\_binarydata, acc.25)  
  
# Calculate KNN for K=38 (square root of observations)  
knn.38 <- knn(train = traindata.binary , test = testdata\_binary, cl = traindata\_binary\_df, k = 38)  
# Calculate Accuracy  
acc.38 <- 100\* sum(testdata\_binary\_df == knn.38)/NROW(testdata\_binary\_df)  
acc.38

## [1] 96.66667

k\_binarydata <- c(k\_binarydata, 38)  
accurary\_binarydata<- c(accurary\_binarydata, acc.38)  
  
accuracy\_binarydata\_df <- data.frame(k\_binarydata, accurary\_binarydata)  
accuracy\_binarydata\_df

## k\_binarydata accurary\_binarydata  
## 1 3 97.33333  
## 2 5 97.33333  
## 3 10 97.33333  
## 4 15 97.00000  
## 5 20 97.00000  
## 6 25 97.33333  
## 7 38 96.66667

## K Values vs Accuracy Plotting  
  
library(ggplot2)  
  
ggplot(data = accuracy\_binarydata\_df, aes(y = accurary\_binarydata, x = k\_binarydata)) + geom\_line() + ggtitle("Binary Classifier - Accuracy vs K Value") + ylab("Accuracy %") + xlab("K value")



## Trinary Classifier - Select data points, normalize and Split 80-20 into train and test datasets.

#normalize function  
normalize<-function(x){  
 return (  
 (x - min(x))/max(x)-min(x)  
 )  
}  
  
classifierdata.normalizedtrinary<-as.data.frame(lapply(classifierdata.trinary[,c(2:3)], normalize))  
head(classifierdata.normalizedtrinary)

## x y  
## 1 10.63355 1.935708  
## 2 10.64453 2.052150  
## 3 10.67074 2.028185  
## 4 10.65656 1.951069  
## 5 10.67562 1.991786  
## 6 10.66783 1.979964

str(classifierdata.normalizedtrinary)

## 'data.frame': 1568 obs. of 2 variables:  
## $ x: num 10.6 10.6 10.7 10.7 10.7 ...  
## $ y: num 1.94 2.05 2.03 1.95 1.99 ...

summary(classifierdata.normalizedtrinary)

## x y   
## Min. :10.26 Min. :1.541   
## 1st Qu.:10.64 1st Qu.:1.900   
## Median :10.78 Median :2.084   
## Mean :10.81 Mean :2.086   
## 3rd Qu.:10.97 3rd Qu.:2.298   
## Max. :11.36 Max. :2.556

#Select train and test data  
  
data.d = sample(1:nrow(classifierdata.normalizedtrinary), size= nrow(classifierdata.normalizedtrinary) \*0.8, replace = FALSE)  
traindata.trinary <- classifierdata.trinary[data.d,] #80% training data  
testdata\_trinary <- classifierdata.trinary[-data.d,] #20% training data  
  
#Create separate dataframes for train and test data for label   
traindata\_trinary\_df<-classifierdata.trinary[data.d,1]  
testdata\_trinary\_df<-classifierdata.trinary[-data.d,1]  
  
k\_valuetrinary<-round(sqrt(nrow(classifierdata.trinary)))  
  
library(class)  
  
# find no of observations  
NROW(traindata\_trinary\_df)

## [1] 1254

#define result data frames  
k\_trinarydata <- c()  
accurary\_trinarydata <- c()  
  
# Calculate KNN for K=3  
knn.3 <- knn(train = traindata.trinary , test = testdata\_trinary, cl = traindata\_trinary\_df, k = 3)  
# Calculate Accuracy  
acc.3 <- 100\* sum(testdata\_trinary\_df == knn.3)/NROW(testdata\_trinary\_df)  
acc.3

## [1] 92.99363

k\_trinarydata <- c(k\_trinarydata, 3)  
accurary\_trinarydata<- c(accurary\_trinarydata, acc.3)  
  
# Calculate KNN for K=5  
knn.5 <- knn(train = traindata.trinary , test = testdata\_trinary, cl = traindata\_trinary\_df, k = 5)  
# Calculate Accuracy  
acc.5 <- 100\* sum(testdata\_trinary\_df == knn.5)/NROW(testdata\_trinary\_df)  
acc.5

## [1] 91.40127

k\_trinarydata <- c(k\_trinarydata, 5)  
accurary\_trinarydata<- c(accurary\_trinarydata, acc.5)  
  
# Calculate KNN for K=10  
knn.10 <- knn(train = traindata.trinary , test = testdata\_trinary, cl = traindata\_trinary\_df, k = 10)  
# Calculate Accuracy  
acc.10 <- 100\* sum(testdata\_trinary\_df == knn.10)/NROW(testdata\_trinary\_df)  
acc.10

## [1] 88.8535

k\_trinarydata <- c(k\_trinarydata, 10)  
accurary\_trinarydata<- c(accurary\_trinarydata, acc.10)  
  
# Calculate KNN for K=15  
knn.15 <- knn(train = traindata.trinary , test = testdata\_trinary, cl = traindata\_trinary\_df, k = 15)  
# Calculate Accuracy  
acc.15 <- 100\* sum(testdata\_trinary\_df == knn.15)/NROW(testdata\_trinary\_df)  
acc.15

## [1] 88.8535

k\_trinarydata <- c(k\_trinarydata, 15)  
accurary\_trinarydata<- c(accurary\_trinarydata, acc.15)  
  
# Calculate KNN for K=20  
knn.20 <- knn(train = traindata.trinary , test = testdata\_trinary, cl = traindata\_trinary\_df, k = 20)  
# Calculate Accuracy  
acc.20 <- 100\* sum(testdata\_trinary\_df == knn.20)/NROW(testdata\_trinary\_df)  
acc.20

## [1] 88.8535

k\_trinarydata <- c(k\_trinarydata, 20)  
accurary\_trinarydata<- c(accurary\_trinarydata, acc.20)  
  
# Calculate KNN for K=25  
knn.25 <- knn(train = traindata.trinary , test = testdata\_trinary, cl = traindata\_trinary\_df, k = 25)  
# Calculate Accuracy  
acc.25 <- 100\* sum(testdata\_trinary\_df == knn.25)/NROW(testdata\_trinary\_df)  
acc.25

## [1] 88.53503

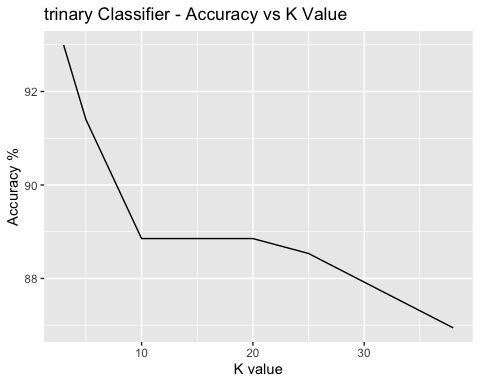
k\_trinarydata <- c(k\_trinarydata, 25)  
accurary\_trinarydata<- c(accurary\_trinarydata, acc.25)  
  
# Calculate KNN for K=38 (square root of observations)  
knn.38 <- knn(train = traindata.trinary , test = testdata\_trinary, cl = traindata\_trinary\_df, k = 38)  
# Calculate Accuracy  
acc.38 <- 100\* sum(testdata\_trinary\_df == knn.38)/NROW(testdata\_trinary\_df)  
acc.38

## [1] 86.94268

k\_trinarydata <- c(k\_trinarydata, 38)  
accurary\_trinarydata<- c(accurary\_trinarydata, acc.38)  
  
accuracy\_trinarydata\_df <- data.frame(k\_trinarydata, accurary\_trinarydata)  
accuracy\_trinarydata\_df

## k\_trinarydata accurary\_trinarydata  
## 1 3 92.99363  
## 2 5 91.40127  
## 3 10 88.85350  
## 4 15 88.85350  
## 5 20 88.85350  
## 6 25 88.53503  
## 7 38 86.94268

## K Values vs Accuracy Plotting  
  
library(ggplot2)  
  
ggplot(data = accuracy\_trinarydata\_df, aes(y = accurary\_trinarydata, x = k\_trinarydata)) + geom\_line() + ggtitle("trinary Classifier - Accuracy vs K Value") + ylab("Accuracy %") + xlab("K value")



## Question C:

**Looking back at the plots of the data, do you think a linear classifier would work well on these datasets?**

## Answer For C

Looking at the distribution of data in the plot, it is spread in different clusters so a linear classifier may not work with this data.