Optional Tasks

Andreas Stasinakis

November 17, 2018

Special task 1

1.1

```
library(readxl)

#import the data from excel file
my_data <- read_excel("..\\spambase.xlsx")

# Spliting the data into training and test data.
n=dim(my_data)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.5))
data = my_data[id,]
newdata = my_data[-id,]



knearest <- function(data,K,newdata){
  X = data[, -ncol(data)]
  Y = newdata[, -ncol(newdata)]

  # Calculate the distance Matrix
  X_hat <- X / sqrt(rowSums(X^2))
  Y_hat <- Y /sqrt(rowSums(Y^2))
  C <- as.matrix(X_hat) %*% as.matrix(t(Y_hat))
  D <- 1 - C
  predictions = c()

  # Find the k - lowest distances
  for (i in 1:ncol(D)) {
    temp_indexes <- order(D[,i])
    indexes <- temp_indexes[1:K]
    predict <- mean(data$Spam[indexes])
    predictions <- c(predictions, predict)
  }

  return(predictions)
}
```

1.2

Table 1: Misclassification rates for cosine similarity

misclassification.rate.train.	0.2627737
misclassification.rate.test.	0.3094891

Misclassification rates for question 1.4

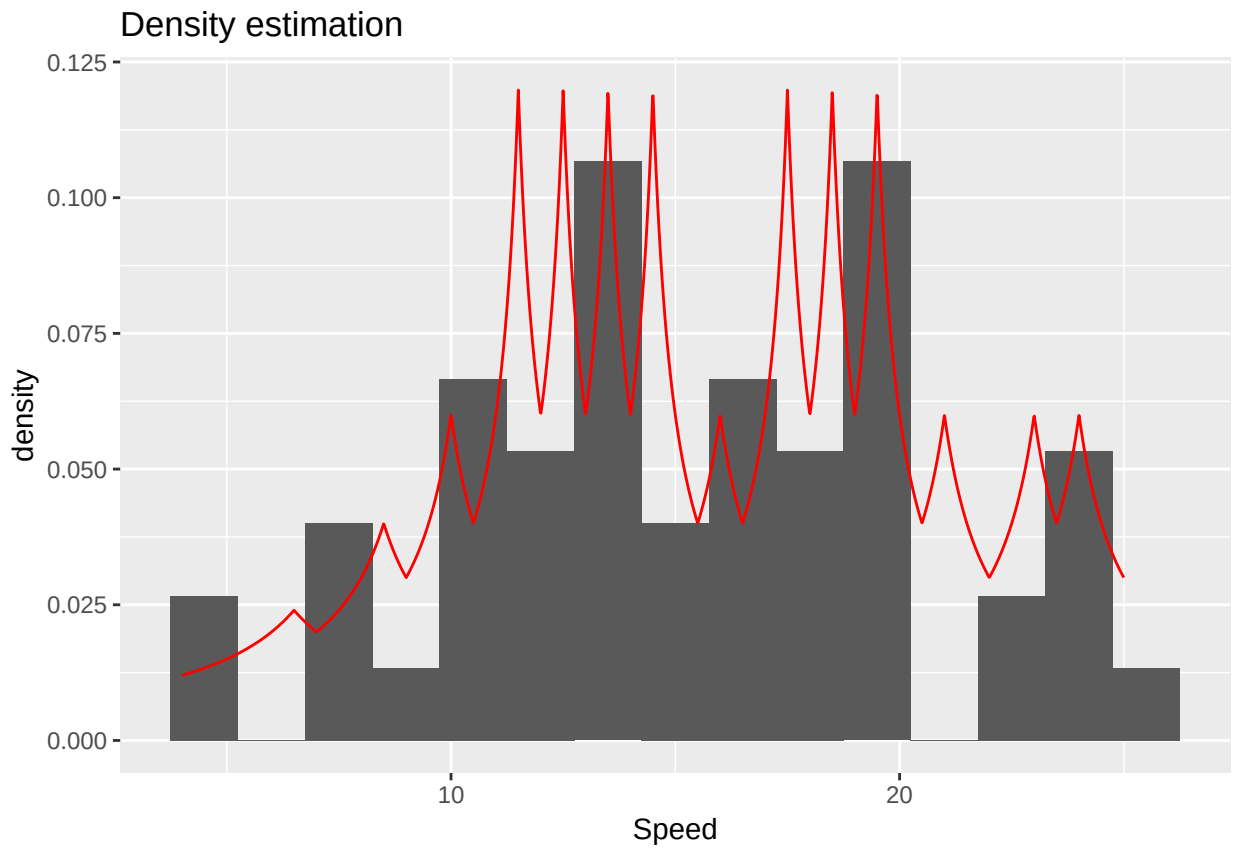
Table 9: misclassification rates

misclassification.rate.train.	0.1722628
misclassification.rate.test.	0.3299270

Analysis

For the cosine similarity, the misclassification rates between the training data(0.2627737) and test data(0.3094891) are not that different. In contrast to question 1.4, where the misclassification rate for test data is almost double than for the training data. Therefore it can be said that for the cosine similarity the model has higher bias than the one in question 1.4 *but* when we check it for the test data, as mentioned before, the misclassification rate for 1.4 increases gradually. So, the variance in 1.4 model is higher than in the cosine similarity, which means that the model may not make accurate predictions and as a result it is overfitting. In general, when we have to choose between different model, we are more interested how accurate they predict in different data sets. Therefore, if i have to choose one of them, i believe that the model with the cosine similarity will make more accurate predictions.

special task 2



Analysis

In this task, we implement *k-nearest neighbor density estimation* in order to calculate densities for a sample by using the above formula :

$$p(x) = K/N * \Delta$$

where K is the number of neighbors we want, N is the number of observations of the original data and Δ is the range of the internal which contains the k nearest neighbors for the sample data we choose. For the sample, we divide the interval of the original data *cars* in 2000 values in order to make the density function more understandable. We use this amount of values because for values less than 2000, *Delta* is large and the density is not clear enough, while for values greater than 2000 the density does not change. In this case, we use a histogram and the density estimation to compare the data distribution to a theoretical model and, if possible, to understand what is the distribution of our original data. But we should also mention that the data set we use for the histogram in this question is quite small, only 50 observations, so we can not be 100% for the results.

From the above plot, it is obvious that the density estimation has many peaks (7) at 0.12, while in the histogram the peaks are only two. It can be also said that the predictions of the density follow the real densities of the histogram. But for $K = 6$, we can not have a clear conclusion for the distribution of the original data, so one solution may be to increase K in order to have a more general idea for the data.

Appendix

```
knitr::opts_chunk$set(echo = FALSE)
library(readxl)

#import the data from excel file
my_data <- read_excel("..\\spambase.xlsx")

# Spliting the data into training and test data.
n=dim(my_data)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.5))
data = my_data[id,]
newdata = my_data[-id,]



knearest <- function(data,K,newdata){
  X = data[, -ncol(data)]
  Y = newdata[, -ncol(newdata)]

  # Calculate the distance Matrix
  X_hat <- X / sqrt(rowSums(X^2))
  Y_hat <- Y /sqrt(rowSums(Y^2))
  C <- as.matrix(X_hat) %*% as.matrix(t(Y_hat))
  D <- 1 - C
  predictions = c()

  # Find the k - lowest distances
  for (i in 1:ncol(D)) {
    temp_indexes <- order(D[,i])
    indexes <- temp_indexes[1:K]
    predict <- mean(data$Spam[indexes])
    predictions <- c(predictions, predict)
  }

  return(predictions)
}


#change from probabilites to 0,1 for training data
predicted_train <- knearest(data = data ,30, newdata = data)
predicted_train[which(predicted_train > 0.5)] = 1
predicted_train[which(predicted_train <= 0.5)] = 0

#Change from probabilities to 0,1 for test data
predicted_test <- knearest(data = data ,30, newdata = newdata)
predicted_test[which(predicted_test > 0.5)] = 1
predicted_test[which(predicted_test <= 0.5)] = 0


conf_matrix_train <- table( data$Spam, predicted_train)
rownames(conf_matrix_train) <- c("real(no-spam)","real(spam)")
```

```r
colnames(conf_matrix_train) <- c("predicted(no-spam)", "predicted(spam)")

#reporting the confusion matrix for test data
conf_matrix_test <- table( newdata$Spam, predicted_test)
rownames(conf_matrix_test) <- c("real(no-spam)","real(spam)")
colnames(conf_matrix_test) <- c("predicted(no-spam)", "predicted(spam)")

#print both confusion matrices
#knitr::kable(x = conf_matrix_train, caption = "confusion matrix for train data")
#knitr::kable(x = conf_matrix_test, caption = "confusion matrix for test data")

#Misclarification rates for cosine similarity
mis_rate_train <- 1-sum(diag(conf_matrix_train))/sum(conf_matrix_train)
mis_rate_test <- 1-sum(diag(conf_matrix_test))/sum(conf_matrix_test)
mis_rate <- data.frame("misclassification rate(train)" = mis_rate_train,"misclassification rate(test)" =
knitr::kable(x = t(mis_rate), caption = "Misclassification rates for cosine similarity")

library(ggplot2)

# Calculate the distance
K = 6
all_densities = c()

density_estimator = function(X,K){
data = as.data.frame(cars$speed)

  for (i in 1: nrow(data)) {
    data$distances[i] = abs(data[i,1] - X)
    distances = sort(data$distances)
    range = 2*distances[K]
    my_density =  K/(nrow(data)*range)

  }
  return(my_density)
}

# create a vector with lenght 50 as sample
X = seq(from = min(cars$speed),to = max(cars$speed), length.out = 2000)

# Loop which calculate all densities for my sample X
for (i in 1:length(X)) {
  temp_density = density_estimator(X[i], K)
  all_densities = c(all_densities, temp_density)
}

# combine the sample with each density to a data frame
density_df = data.frame("speed" = X, "density" = all_densities)

# Plot the densities and the histogram
ggplot() +
  geom_histogram(mapping = aes(x = cars$speed, y = ..density..), bins = 15) +
  geom_line(mapping = aes(x = density_df$speed, y = density_df$density),color = "red") +
  labs(title = "Density estimation", x = "Speed", y = "density")
```