Special Task

Sridhar(sriad858)

26 November 2018

1. Implement K-nearest neighbours

1. Function for K nearest

In this question I created a function to calculate the K-nearest neighbours using the distance function specified in the question.

2. Testing the spam database on function

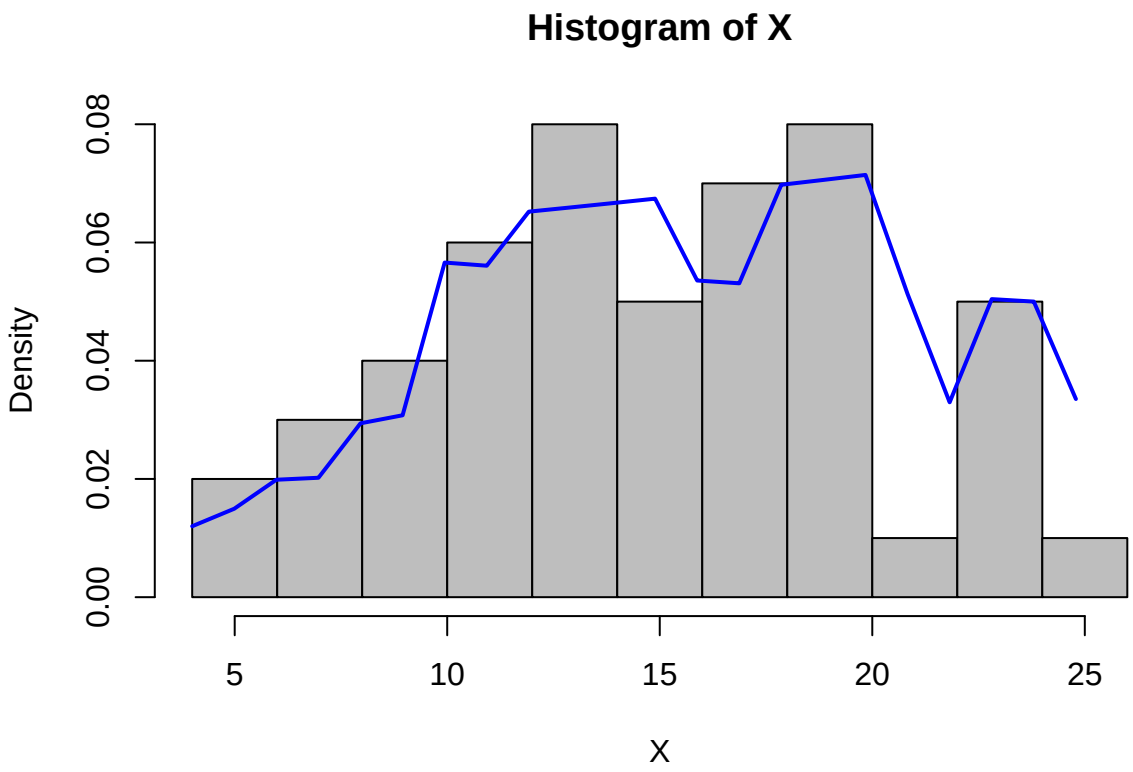
```
## [1] "Train Results -"

## [1] "Misclassification"

## [1] 0.2832117

##          not_spam spam
## not_spam      818  126
## spam          262  164

## [1] "Test Results -"

## [1] "Misclassification"

## [1] 0.2985401

##          not_spam spam
## not_spam      812  126
## spam          283  149
```

The miscalssification rate on the train using the above created function for KNN is 0.2832 and on test the misclassification is 0.2985. This shows the performance of this algorithm was almost equally good on this algorithm.

The precision of the algorithm is also very good. It is 0.8665 for train and 0.8656 for test. So it performed equally good. This performed much better than the KKNN algorithm used in the assignment 1 for the LAb 1. This had almost the same accuracy but higher precision and recall than that algorithm.

We can look for a better K using an iterative method that would fit the data better.

2 K-nearest neighbor density estimation



This is the histogram plot of speeds column in cars dataset divided into 11 bins. We then overlaid it with our calculated density plot using the Knearest density estimation method and it closely fits the data. I am using the sqeuence of values between the min and max values of speed with step size of 0.99 to calculate the corresponding probabilities using KNN density estimation and am storing it into a matrix. I am then using the matrix to create this density plot.

This density plot is a good estimation to the data. In the case where we dont know the distribution of the data this is a good method to find an estimation of the PDF for the distribution. Like in this case the PDF had multiple modes in thw PDF, the distribution for such data can not be identified easily, so using knn density estimation is a great method to determine the probability density finction.

So, I conclude that K-nearest neighbours density estimation is a great method for finding and estimate of the probability density function of an unknown distribution.

Appendix

```
library(readxl)
k_near = function(x, k){
  ind = sort(x, decreasing = F, index.return=T)$ix[1:k]
  return(ind)
}

knearest = function(train, k, test){
  X_train = train[, 1:(ncol(train)-1)]
  Y_train = train$Spam
  X_test = test[, 1:(ncol(test)-1)]
```

```
  Y_test = test$Spam
  a = as.matrix(X_train)
  b = as.matrix(X_test)

  a_div = rowSums(a^2)^(0.5)
  a_cap = a / a_div

  b_div = rowSums(b^2)^(0.5)
  b_cap = b / b_div

  c = a_cap %*% t(b_cap)

  d = 1-c

  res = matrix(0, nrow = k, ncol = 0)
  for( i in 1:ncol(d)){
    ind = k_near(d[, i], k)
    res = cbind(res, Y_train[ind])
  }
  res = t(res)
  pred = rowSums(res)/k
  # res_df = data.frame(res[2:nrow(res),])
  return(pred)
}
spam_data = read_xlsx("spambase.xlsx", sheet = "spambase_data")

## 50% of the sample size
smp_size <- floor(0.50 * nrow(spam_data))

## set the seed to make your partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(spam_data)), size = smp_size)

train <- spam_data[train_ind, ]
test <- spam_data[-train_ind, ]

#Train
pred = knearest(train, 30, train)
y_pred = ifelse(pred>0.5, 1, 0)
acc = sum(y_pred==train$Spam)
print("Train Results -")
print("Misclassification")
print(1 - (acc/nrow(train)))

pos = which(train$Spam == 1)
tn = sum(y_pred[pos])
fn = length(pos) - tn
neg = which(train$Spam == 0)
fp = sum(y_pred[neg])
tp = length(neg) - fp
tbl1 = data.frame('not_spam'= c(tp, fn), 'spam'=c(fp, tn), row.names = c('not_spam', 'spam'))
print(tbl1)
```

```r
#Test
pred = knearest(train, 30, test)
y_pred = ifelse(pred>0.5, 1, 0)
acc = sum(y_pred==test$Spam)
print("Test Results -")
print("Misclassification")
print(1 - (acc/nrow(test)))

pos = which(test$Spam == 1)
tn = sum(y_pred[pos])
fn = length(pos) - tn
neg = which(test$Spam == 0)
fp = sum(y_pred[neg])
tp = length(neg) - fp
tbl2 = data.frame('not_spam'= c(tp, fn), 'spam'=c(fp, tn), row.names = c('not_spam', 'spam'))
print(tbl2)
data = cars
kth_near = function(X, val, k){
  d = abs(X - val)
  dis = sort(d, decreasing = F, index.return=T)$x[k]
  return(dis)
}

v_cal = function(d){
  v = (pi^(d/2))/(factorial(d/2))
  return(v)
}

pcap_knn = function(X, k, d, val){
  p_knn = k/(length(X)*v_cal(d)*(kth_near(X, val, k))^d)
  return(p_knn)
}

X = cars$speed
k=6
d=1
a = matrix(0, nrow =0, ncol = 2)
for(i in seq(min(X),max(X),0.99)){
  p = pcap_knn(X, k, d, i)
  a = rbind(a, c(i, p))
}
a = data.frame(a)
colnames(a) = c('X', 'PDF')

hist(X, prob=TRUE, col="grey", breaks = 10)
lines(a, col="blue", lwd=2)
#lines(a, lty="dotted", col="darkgreen", lwd=2)
```