---
title: "week1_HW"
author: "Cyntia Wonsch"
date: "5/18/2019"
output: pdf_document

Question 2.1
It would be interesting to know if a student would be able to qualify to get certified by the end of the ISYE6501 class by just answering a few questions before paying for the verified track.
Predictors:
Does the student has strong mathematical background? yes/no
Does the student has previous R knowledge? yes/no
Does the student has preveious programming knowledge in any other language? yes/no
How many hours the student has available per week to study?
Is the students goal to pursue the OMSA program? yes/no

Question 2.2
1. Based on the analysis below, the better classifier for this data set is the SVM using the vanilladot kernel. Eventhough the accuracy while training the dataset was higher using rbfdot kernel, when we ran the predictions on the test data set the classifier used with the vanilladot kernel predicted better than the classifier using rbfdot.Vanilladot got accuracy of 84% against 81% for rbfdot

Equation for the classifier:
-4.393778e-05A1 -6.329270e-04A2 -4.438598e-04A3 + 6.706653e-04A8 + 1.008353e+00A9 -4.443065e-04A10 + 6.554160e-05A11 + 2.164979e-04A12 -3.887049e-04A14 + 1.149383e-01A15 + 0.0827413 = 0

2. I wanted to see how a linear kernel(vanilladot) would perform against a non-linear kernel (rbfdot)

3. Good value of k=12 with 85% accuracy

Question 3.1
knn cross-validation had a 84% accuracy on the test validation set

```{r svm for credit card data}
# find out the working directory to be used below to read the data from correct folder
# getwd()
```

```
# set working directory to get desired file
setwd("C:/Users/cynti/Desktop/GTx/ISYE6501/week_1_data-summer")

# read file assigning it to variable named 'data'
data <- read.table("./data 2.2/credit_card_data-headers.txt", header = TRUE)
# head(data)

# Just to have a general idea, I plotted the predictors against each other. Identifying 0 as red
and 1 as blue.

#data$Color = "black"
#data$Color[data$R1==0]="red"
#data$Color[data$R1==1]="blue"
#plot(data[,1:10], col=data$Color)
```
Split data into train and test data (test data to be used to validate)
```{r splitting data}
set.seed(13)
train_data <- sample(1:nrow(data), 0.8*nrow(data))
test_data <- setdiff(1:nrow(data), train_data)

X_train <- data[train_data, -15]
y_train <- data[train_data, 'R1']

X_test <- data[test_data, -15]
y_test <- data[test_data, 'R1']

```
Testing different kernels
Vanilladot
```{r traing vanilla}
library(kernlab)
# note: ksvm requires data matrix and data factor

vanilla_model <- ksvm(as.matrix(X_train[,1:10]), as.factor(y_train),
type="C-svc",kernel="vanilladot", C=1, scaled=TRUE)
# tried C in various ranges. From 0.002 to 999 it presented the highest accuracy, left it at 0.002
because the code runs faster

pred_vanilla <- predict(vanilla_model,X_train[,1:10])
# pred_vanilla
# vanilla_model
```

```
# weighted
a_vanilla <- colSums(vanilla_model@xmatrix[[1]]*vanilla_model@coef[[1]])
a_vanilla

# intercept
a0_vanilla <- -vanilla_model@b
a0_vanilla
```

```{r find best value for C}
#commenting it out to "setting default kernel doesnt print 200 times on my markdown file
vanilla_C <- rep(0,200)
for (C in 1:200) {
  model1 <- ksvm(as.matrix(X_train[,1:10]), as.factor(y_train),  type="C-svc",kernel="vanilladot",
C=C, scaled=TRUE)
 model_pred1 <- predict(model1, X_train[,1:10])

 vanilla_C[C] <- sum(model_pred1 == y_train) / nrow(X_train)
 }
```
Best C=1 for vanilladot
```{r vanilla_C results}
# The value for C is the same from the range I looked into (1 to 200), I will use C=1 just for the
sake of speed
max(vanilla_C)
which.max(vanilla_C)
# vanilla_C
```
rbfdot
```{r training rbfdot kernel}
rbf_model <- ksvm(as.matrix(X_train[,1:10]), as.factor(y_train),
              type="C-svc", kernel="rbfdot", C=158, scaled=TRUE)
rbf_model

pred_rbf <- predict(rbf_model, X_train[,1:10])

# checking for prediction accuracy
sum(pred_rbf == y_train) / nrow(X_train)

# Calculating the weight of each predictor (coeficients)
a_rbfdot <- colSums(rbf_model@xmatrix[[1]]*rbf_model@coef[[1]])
a_rbfdot
```

```
# Calculate the constant (negative intercept)
a0_rbfdot <- -rbf_model@b
a0_rbfdot
```
Best C= for rbfdot
```{r find best C value for rbfdot}
rbf_C <- rep(0,200)
for (C in 1:200) {
  model2 <- ksvm(as.matrix(X_train[,1:10]), as.factor(y_train), type="C-svc",kernel="rbfdot",
C=C, scaled=TRUE)
  model_pred2 <- predict(model2, X_train[,1:10])
  rbf_C[C] <- sum(model_pred2 == y_train) / nrow(X_train)
}
```
```{r rbf_C results}
# The value for C is the same from the range I looked into (1 to 200), I will use C=1 just for the
sake of speed
max(rbf_C)
which.max(rbf_C)
# rbf_C
```

Comparing vanilladot (for linear data set) and rbfdot (for non-linear data set) with train data
set
```{r prediction accuracy on train data set}
sum(pred_vanilla == y_train) / nrow(X_train)
sum(pred_rbf == y_train) / nrow(X_train)

```
The tables bellow show us that vanilladot predicted 449 out of 523 from the training data set
and rbfdot predicted 499 out of 523
```{r prediction tables}
table(pred_vanilla == y_train)
table(pred_rbf == y_train)

```
To validate the models I tested them on the test data to see how the model would perform
on new data
```{r getting prediction on test data set}
test_vanilla <- predict(vanilla_model, X_test[,1:10])
sum(test_vanilla == y_test) / nrow(X_test)

test_rbf <- predict(rbf_model,X_test[,1:10])
```

```
sum(test_rbf == y_test) / nrow(X_test)

table(test_vanilla == y_test)
table(test_rbf == y_test)
```
Finding a good value of k for the entire data set
```{r knn best value of k}
library(kknn)
# X is the function variable for the possible k-values
kknn_accurary = function(X) {
  # set vector with empty rows to store predictions, in this case it will be 654
  pred_kknn <- rep(0,(nrow(data)))

  for (item in 1:nrow(data)) {
    model_kknn <- kknn(R1~., data[-item,], data[item,],
                       k=X, scale = TRUE)

    # mode(model_kknn)

    # each prediction will be added to a row from the empty vector                above
    pred_kknn[item] <-  as.integer(fitted(unlist(model_kknn)) + 0.5)
    # pred_kknn

  }
  # factor of accuracy
  acc = sum(pred_kknn == data[,11]) / nrow(data)

  # at each iteration I want it to return the factor of accuracy for each      possible tested k-values
  return(acc)
}
```
```{r call function}
# set empty vector list with 20 empty spaces
test_k <- rep(0,100)

# check accuracy for k-values between 1 and 20
for (X in 1:100) {
  # append results into empty vector list created above
  test_k[X] <- kknn_accurary(X)
}

```
Best value for k=12 giving a accurary factor of 0.853211

```{r finding best k-value}
max(test_k)
which.max(test_k)
```
3.1
```{r use cross-validation}
# by the beginning of this markdown I had already split the data into 80% train/ 20% test data naming them train_data and test_data respectively

library(kknn)
length(cv_pred)
length(data[train_data,11])
model_cv <- train.kknn(R1~., data[train_data,], scale=TRUE)
cv_pred <- predict(model_cv, data[test_data,])

rounded <- round(cv_pred)

acc_cv <- table(rounded, data[test_data,11])
acc_cv

sum(rounded == data[test_data,11])/length(data[test_data,11])
```