HW2

2022-09-06

Question 3.1 (a) For this question, I decided to use the train.kknn function in R to train the data set. This method uses n-fold cross-validation and n represents the number of data points. I used ??train.kknn to figure out the useage.

```
# clear the environment for this assignment and use the correct libaray
rm(list = ls())
library(kknn)

data <- read.table("/Users/xiaofanjiao/Desktop/credit_card_data.txt",
header= F, stringsAsFactors = F)

# view the first few data
head(data)
```

```
##   V1    V2    V3   V4 V5 V6 V7 V8  V9 V10 V11
## 1  1 30.83 0.000 1.25  1  0  1  1 202   0   1
## 2  0 58.67 4.460 3.04  1  0  6  1  43 560   1
## 3  0 24.50 0.500 1.50  1  1  0  1 280 824   1
## 4  1 27.83 1.540 3.75  1  0  5  0 100   3   1
## 5  1 20.17 5.625 1.71  1  1  0  1 120   0   1
## 6  1 32.08 4.000 2.50  1  1  0  0 360   0   1
```

```
# set the random number generator
set.seed(1)


# use train.kknn for leave-one-out cross-validation where I decided to set maximum value of k (number o
kmax = 50

model <- train.kknn(V11~V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10, data,kmax=kmax,scale=TRUE)

# create array of 0 for predictions
accuracy <- rep(0,kmax)

# calculate prediction qualities
for (k in 1:kmax) {
    predicted <- as.integer(fitted(model)[[k]][1:nrow(data)] + 0.5) # round off to 0 or 1
    #calculate the correct predictions
    accuracy[k] <- sum(predicted == data$V11)
}

# show results
accuracy
```

```
##  [1] 533 533 533 533 557 553 554 555 554 557 557 558 557 557 558 558 558 557 556
## [20] 556 555 554 552 553 553 552 550 548 549 550 548 547 547 545 544 544 543 544
## [39] 544 544 544 547 547 547 549 550 548 549 550 549
```

Question 3.1 (b) For this question, we need to split the data into training, validation, and test data sets with a ratio of 70, 20, 10

```
# clear the environment for this assignment and use the correct libaray
rm(list = ls())
library(kknn)
library(kernlab)

data <- read.table("/Users/xiaofanjiao/Desktop/credit_card_data.txt",
header= F, stringsAsFactors = F)

# view the first few data
head(data)
```

```
##   V1    V2    V3   V4 V5 V6 V7 V8  V9 V10 V11
## 1  1 30.83 0.000 1.25  1  0  1  1 202   0   1
## 2  0 58.67 4.460 3.04  1  0  6  1  43 560   1
## 3  0 24.50 0.500 1.50  1  1  0  1 280 824   1
## 4  1 27.83 1.540 3.75  1  0  5  0 100   3   1
## 5  1 20.17 5.625 1.71  1  1  0  1 120   0   1
## 6  1 32.08 4.000 2.50  1  1  0  0 360   0   1
```

```
# set the random number generator
set.seed(1)

## copied from Question 3.1(a)

# ---- Split data ----

# 70% for training
m_train = sample(nrow(data), size = floor(nrow(data) * 0.7))
cred_train = data[m_train,] # training data set

# Using the remaining data for test and validation split 30%
remaining = data[-m_train, ]

# Split the data again to 20% 10%
m_val = sample(nrow(remaining), size = floor(nrow(remaining)/3*2))

cred_val = remaining[m_val,]  # validation data set 20%
cred_test = remaining[-m_val, ] # test data set 10%

acc <- rep(0,30)

# ---- Train SVM models ----

# values of C to test
amounts <- c(0.001, 0.01, 0.1, 1, 10, 100, 1000)
```

```
for (i in 1:10) {

    # fit model using training set
     model_scaled <- ksvm(as.matrix(cred_train[,1:10]),
               as.factor(cred_train[,11]),
             type = "C-svc", # Use C-classification method
               kernel = "vanilladot", # Use simple linear kernel
               C = amounts[i],
             scaled=TRUE)

    #  compare models using validation set

                pred <- predict(model_scaled,cred_val[,1:10])
                acc[i] = sum(pred == cred_val$V11) / nrow(cred_val)
}
```

```
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
```

```
acc[1:10]
```

```
##  [1] 0.7786260 0.8702290 0.8702290 0.8702290 0.8702290 0.8702290 0.8702290
##  [8] 0.6412214 0.6412214 0.6412214
```

```
# Best c value
amounts[which.max(acc[1:10])]
```

```
## [1] 0.01
```

```
# best validation set
max(acc[1:10])
```

```
## [1] 0.870229
```

Question 4.1 I watch Netflix a lot and I think Netflix can use clustering to analyze users depending on their characteristics. The company can look at: 1. Consumer Demographic information (age, gender. . .), 2. What time of the day do they watch Netflix the most, 3. What type of video is watched the most (movie, TV show, documentary . . .), 4. What genera is most popular, 5. Geographic location.

Question 4.2 I understood the concept for this question is for us to find the best number of clusters for this data set. We need to run multiple times with different initial cluster centers. However, I am very lost in how to perform this in R as I am still pretty new to R. I tried using the ??kmeans for reference and this is what I got for this question.

```r
# Clear environment
rm(list = ls())

data <- read.table("/Users/xiaofanjiao/Desktop/iris.txt", header = TRUE)

head(data)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
set.seed(1)

# Use the kmeans() function to run the clustering algorithm, here I decided to try k from 2 to 5 to see

irisClusterALL2 <- kmeans(data[,1:4], 2, nstart = 20)
irisClusterALL3 <- kmeans(data[,1:4], 3, nstart = 20)
irisClusterALL4 <- kmeans(data[,1:4], 4, nstart = 20)
irisClusterALL5 <- kmeans(data[,1:4], 5, nstart = 20)
```

and then I'm just kinda lost on how to code and how to perform the following activities.