Untitled

Michael Ahn

1/23/2019

```
library(kknn)
```

Homework 2

Question 3.1

```
# Read data and randomly reorder
df.cc <- read.table("credit_card_data.txt", header=FALSE, stringsAsFactors=FALSE)
df.cc <- df.cc[sample(nrow(df.cc)),]

# Split train and test data
N <- nrow(df.cc)
n <- ceiling(N*0.8)
df.cc.train <- df.cc[1:n,]
df.cc.test <- df.cc[(n+1):N,]

k <- 1:25
```

The data was split into training and testing sets.
The training set will be used for cross-validation/ validation while searching for optimal parameters to our model (kknn).
The testing set will be used to give a final evaluation of our model's accuracy.

a

```
# empty list of accuracy scores
scores <- c()

# train and validate models using each k-value
for (i in 1:length(k)) {
    # Rescramble the training set each iteration
    df.cc.train[sample(nrow(df.cc.train)),]
    # Split training and cross-validation sets
    nsample <- n*0.75
    train <- df.cc.train[1:nsample,]
    cv <- df.cc.train[(nsample+1):n,]

    # Build model using k
    kknn <- kknn(V11~.,
                 train,cv,
                 k=k[i],
                 kernel="optimal",
                 distance=2,
                 scale=TRUE)
```

```
    # Use predictions to get accuracy score on cv set
    pred <- round(fitted.values(kknn))
    scores[i] <- sum(pred == cv[,11]) / length(pred)
}
# return highest scoring k-value
# chooses lowest k-value if multiple, following "simplest model" philosophy
best_k <- k[which(scores==max(scores))[1]]
best_k
```

```
## [1] 14
```

```
max(scores)
```

```
## [1] 0.8473282
```

After randomly splitting training and validating sets for each k-value, we know which k-value produced the best score.
Now, it is time to build the model based on this optimal parameter, "best_k".

```
  # Build model using best_k value
  # Test on df.cc.test set aside from the very start
kknn_best <- kknn(V11~.,
                  df.cc.train,
                  df.cc.test,
                  k=best_k,
                  kernel="optimal",
                  distance=2,
                  scale=TRUE)
y <- round(fitted.values(kknn_best)) == df.cc.test[,11]
yscore <- sum(y)/ length(y)
```

```
## [1] "The kknn model predicts with 0.838462 accuracy using k = 14"
```

b.

```
# empty list of accuracy scores
scores <- c()
models <- c()

kknn <- train.kknn(V11~.,
                   df.cc.train,
                   kmax=25,
                   distance=2,
                   kernel="optimal",
                   scale=TRUE)
pred <- round(predict(kknn, df.cc.test)) == df.cc.test[,11]
score <- sum(pred)/length(pred)
```

Validates and finds the best model using train.knn function.
The best model is then used on the testing set reserved to find the actual representation of model accuracy.

Question 4.1

An example of clustering's usefulness is in a business trying to target consumers. While the more intuitive approach might be to send marketing messages to customers vs not customers, clustering might further help by organizing potential customers into categories that were not understood bbefore - for example, should this demographic be put into "loyal customers", "not loyal", or something else entirely?
Predictors may include previous transaction records - frequency, which items, how much money, etc., or frequency of website visitation.

Question 4.2

```
df.iris <- read.table('iris.txt', header=TRUE)
```

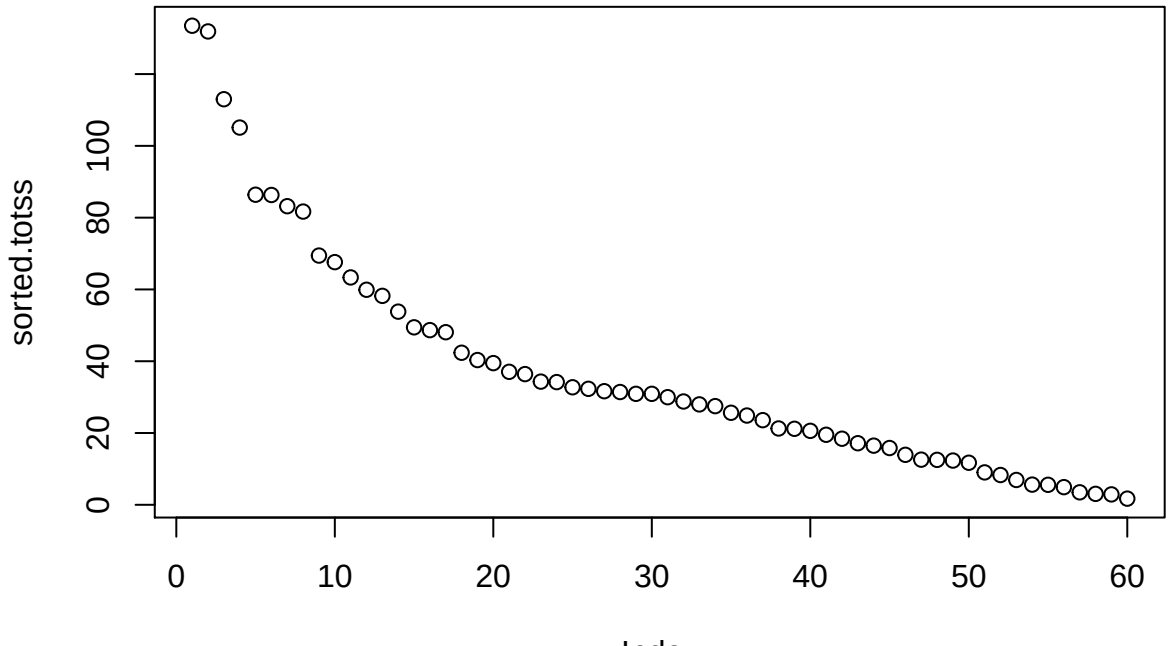
```
#install.packages("combn")
#install.packages("sets")

comb <- c(2,3,4,5)
a <- combn(comb,1)
b <- combn(comb,2)
c <- combn(comb,3)
d <- combn(comb,4)
totss <- c()
models <- c()

x <- 1
for (j in 1:length(a)){
  for (i in 2:5) {
    kmeans <- kmeans(df.iris[,a[j]],
                     centers=i)
    totss[x] <- kmeans$tot.withinss
    models[[x]] <- kmeans
    x <- x+1
  }
}
for (j in 1:ncol(b)){
  for (i in 2:5) {
    kmeans <- kmeans(df.iris[,b[,j]],
                     centers=i)
    totss[x] <- kmeans$tot.withinss
    models[[x]] <- kmeans
    x <- x+1
  }
}
for (j in 1:ncol(c)){
  for (i in 2:5) {
    kmeans <- kmeans(df.iris[,c[,j]],
                     centers=i)
    totss[x] <- kmeans$tot.withinss
    models[[x]] <- kmeans
    x <- x+1
  }
}
for (j in 1:1){
```

```
  for (i in 2:5) {
    kmeans <- kmeans(df.iris[,d[j]],
                     centers=i)
    totss[x] <- kmeans$tot.withinss
    models[[x]] <- kmeans
    x <- x+1
  }
}


sorted.totss <- abs(sort(-totss))
plot(sorted.totss)
```



In this iteration, it looks like the 9th indexed data point is the "elbow" when compared across all combinations and clusters.

```
value.totss <- sorted.totss[9]
index.totss <- which(totss==value.totss)
index.totss
```

```
## [1] 42
```

The value is 42, the second iteration of the third combination i.e. c(2,3,5) with k=3 in this case. Interestingly, since these are randomly built, I have had multiple iterations point toward the same combination and cluster value, which supports the idea that this is the correct one.

```
best_kmeans <- models[[42]]
ans <- best_kmeans$cluster
table(ans, df.iris[,6])
```

```
##
## ans setosa versicolor virginica
##   1      0         45        13
##   2      0          5        37
##   3     50          0         0
```

Given that setosa has 50, verisocolor is split 45/5, and virginica is split 37/13, the MAX accuracy possibble is (150-18)/150 which leads to 88%.