Solution Lesson03: KNN

Machine Learning using R

Exercise 1: Crabs data set

The `crabs` data set describes different morphological measurements of two different crab-species (see https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/crabs.html for a detailed description of the data). We want to train a KNN-classifier on the crabs data set.

a) The data set is contained in the r package `MASS`. Load the package in order to access the data frame `crabs`.

```
#install.packages("MASS")
library(MASS)
head(crabs)
```

```
##   sp sex index   FL  RW   CL   CW  BD
## 1  B   M     1  8.1 6.7 16.1 19.0 7.0
## 2  B   M     2  8.8 7.7 18.1 20.8 7.4
## 3  B   M     3  9.2 7.8 19.0 22.4 7.7
## 4  B   M     4  9.6 7.9 20.1 23.1 8.2
## 5  B   M     5  9.8 8.0 20.3 23.0 8.2
## 6  B   M     6 10.8 9.0 23.0 26.5 9.8
```

b) Get a first impression of the data and make sure that everything is coded correctly.

```
dim(crabs)
```

```
## [1] 200   8
```

```
str(crabs)
```

```
## 'data.frame':    200 obs. of  8 variables:
##  $ sp   : Factor w/ 2 levels "B","O": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sex  : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ index: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ FL   : num  8.1 8.8 9.2 9.6 9.8 10.8 11.1 11.6 11.8 11.8 ...
##  $ RW   : num  6.7 7.7 7.8 7.9 8 9 9.9 9.1 9.6 10.5 ...
##  $ CL   : num  16.1 18.1 19 20.1 20.3 23 23.8 24.5 24.2 25.2 ...
##  $ CW   : num  19 20.8 22.4 23.1 23 26.5 27.1 28.4 27.8 29.3 ...
##  $ BD   : num  7 7.4 7.7 8.2 8.2 9.8 9.8 10.4 9.7 10.3 ...
```

```
summary(crabs)
```

```
##  sp      sex         index            FL              RW              CL
##  B:100   F:100   Min.   : 1.0   Min.   : 7.20   Min.   : 6.50   Min.   :14.70
##  O:100   M:100   1st Qu.:13.0   1st Qu.:12.90   1st Qu.:11.00   1st Qu.:27.27
##                  Median :25.5   Median :15.55   Median :12.80   Median :32.10
```

```
##                   Mean   :25.5   Mean   :15.58   Mean   :12.74   Mean   :32.11
##                   3rd Qu.:38.0   3rd Qu.:18.05   3rd Qu.:14.30   3rd Qu.:37.23
##                   Max.   :50.0   Max.   :23.10   Max.   :20.20   Max.   :47.60
##        CW              BD
##  Min.   :17.10   Min.   : 6.10
##  1st Qu.:31.50   1st Qu.:11.40
##  Median :36.80   Median :13.90
##  Mean   :36.41   Mean   :14.03
##  3rd Qu.:42.00   3rd Qu.:16.60
##  Max.   :54.60   Max.   :21.60
```

Everything looks okay.

c) Fit a knn classifier to the data using k=3. We try to model the species (`sp`) of the crabs based on the five morphological measurements (if unsure, check the data description to make sure you pick the correct columns). (**Hint**: `knn()`, you need to load the package `class` for the `knn()` function)

```
morph <- crabs[,4:8]
library(class)
knn.pred <- knn(train = morph,
                test = morph,
                cl = crabs$sp,
                k = 3)
knn.pred
```

```
##   [1] B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
##  [38] B B B B B B B B B B B O B B B B B B B B B B B B B B B B B B B B B B B B B
##  [75] B B B B B B B B B B B B B B B B B B B B B O B B B B O O O O O O O O O O O
## [112] O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O
## [149] O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O O
## [186] O O O O O O O O O O O O O O O
## Levels: B O
```

d) Display the confusion matrix of the predictions (on the training-data itself) and calculate the training error (see slides).

```
# Confusion matrix:
confT <- table(knn.pred, crabs$sp)
confT
```

```
##
## knn.pred  B  O
##        B 98  1
##        O  2 99
```

```
# Training error:
missCount <- sum(confT[row(confT)!=col(confT)])
trainErr <- missCount/nrow(crabs)
trainErr
```

```
## [1] 0.015
```

Exercise 2: NYC taxi trip records

The taxi trip data set is a collection of information regarding taxi trips in New York City. The data set includes information regarding (amongst others) the pick-up and drop-off location, trip distances, fare prices and passenger count. See https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page for more information. We work with a reduced version of the data set, which only includes a sample of the total records.

a) Load the data with the command `read.csv('taxi.csv', stringsAsFactors=TRUE)` and get a first impression of the data. There should only be one factor in the data which is the colour of the taxi (there are green and yellow cabs in NY). How many rows does the data contain? How many observations are there for green and yellow taxi trips? (**Note**: the pick-up and drop-off location IDs shouldn't ideally be coded as numbers, however, for this exercises we will assume that location IDs with similar numbers are also closer to each other and will therefore keep the two variables as numericals)

```r
taxi <- read.csv('taxi.csv', stringsAsFactors = TRUE)
str(taxi)
```

```
## 'data.frame':    10000 obs. of  7 variables:
##  $ PULocationID   : int  97 26 95 217 42 255 66 66 145 74 ...
##  $ DOLocationID   : int  225 11 160 97 116 37 61 33 145 239 ...
##  $ passenger_count: int  1 1 1 1 1 2 1 3 5 1 ...
##  $ trip_distance  : num  2.62 3.32 2.01 3.03 0.64 2.83 3.88 1.17 0.6 3.37 ...
##  $ fare_amount    : num  13.5 13 11 13 6 13 17 7.5 4 14.5 ...
##  $ tip_amount     : num  2.96 2 3.84 0 0 3.45 3.66 1.86 0.96 3.82 ...
##  $ col            : Factor w/ 2 levels "green","yellow": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
table(taxi$col)
```

```
##
##  green yellow
##   5000   5000
```

There are 5000 observations each for yellow and green taxis.

b) We want to compare the training and test error when using a k-nearest neighbor classification. The goal is to predict the colour of the taxi based on the trip information. Remove a random sample from the taxi data (size=1000), we will use this sample as a test data set. The remaining data will be used as training data. Calculate the training and test error of the KNN algorithm with k=4. (**Hint**: `sample()`, fix the random seed with `set.seed()`)

```r
set.seed(111)
s=1000
test.ind <- sample(1:nrow(taxi), size = s)
taxi.test <- taxi[test.ind,]
taxi.train <- taxi[-test.ind,]
# Apply KNN:
knn.pred.tr <- knn(train = taxi.train[,-7], # To training data (we have to remove
                                            # the target variabel (column 7))
                test = taxi.train[,-7],
                cl = taxi.train$col,
                k = 4)
```

```r
knn.pred.te <- knn(train = taxi.train[,-7], # To test data
                test = taxi.test[,-7],
                cl = taxi.train$col,
                k = 4)
# Confusion matrix (training data):
confT <- table(knn.pred.tr, taxi.train$col)
confT
```

```
##
## knn.pred.tr green yellow
##      green   4145    447
##      yellow   346   4062
```

```r
# Training error:
missCount <- sum(confT[row(confT)!=col(confT)])
trainErr <- missCount/nrow(taxi.train)
trainErr
```

```
## [1] 0.08811111
```

```r
# Confusion matrix (test data):
confT.te <- table(knn.pred.te, taxi.test$col)
confT.te
```

```
##
## knn.pred.te green yellow
##      green    455     71
##      yellow    54    420
```

```r
# Test error:
missCount.te <- sum(confT.te[row(confT.te)!=col(confT.te)])
testErr <- missCount.te/nrow(taxi.test)
testErr
```

```
## [1] 0.125
```

The test error (0.125) is higher than the training error (0.088), which is not further surprising. Because we drew a random test sample from the original data your results can vary from the ones here. In order to reproduce these results one would have to set the same seed.

c) **Extra:** We now want to compare the training and test error for different k. Write a loop in which you fit a knn classifier to the training data (from previous exercise) with k ranging from 1 to 30. Collect for each k the training error and the test error in a table. Also: Standardize the data (train and test set) before applying the knn classifier.

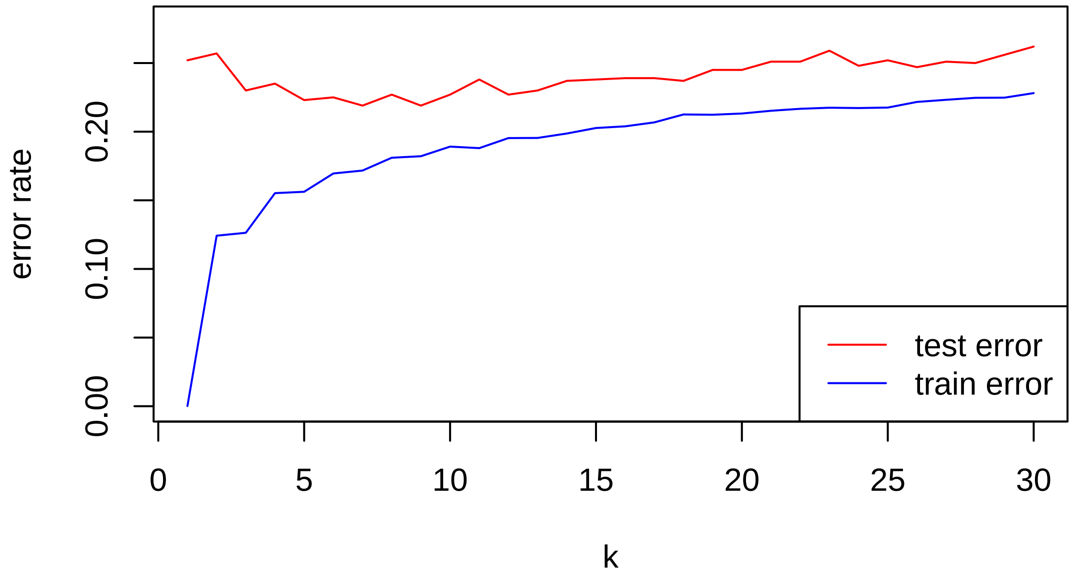
```r
ks <- 1:30
train.err <- NA
test.err <- NA
taxi.tr.sc <- scale(taxi.train[,-7])
taxi.te.sc <- scale(taxi.test[,-7])
set.seed(262)
for (i in 1:length(ks)){
  # Test error
  knn.pred.te <- knn(train = taxi.tr.sc,
                test = taxi.te.sc,
```

```r
                cl = taxi.train$col,
                k = ks[i])
  confT.te <- table(knn.pred.te, taxi.test$col)
  missCount.te <- sum(confT.te[row(confT.te)!=col(confT.te)])
  test.err[i] <- missCount.te/nrow(taxi.test)
  # Training error
  knn.pred.tr <- knn(train = taxi.tr.sc,
                test = taxi.tr.sc,
                cl = taxi.train$col,
                k = ks[i])
  confT.tr <- table(knn.pred.tr, taxi.train$col)
  missCount.tr <- sum(confT.tr[row(confT.tr)!=col(confT.tr)])
  train.err[i] <- missCount.tr/nrow(taxi.train)
  #print(paste('done with k =', ks[i]))  # to show which iteration is completed
}
# Put everything together:
knn.res <- data.frame(k=ks, train.err, test.err)
knn.res
```

```
##     k    train.err test.err
## 1   1 0.0001111111    0.252
## 2   2 0.1242222222    0.257
## 3   3 0.1263333333    0.230
## 4   4 0.1552222222    0.235
## 5   5 0.1562222222    0.223
## 6   6 0.1695555556    0.225
## 7   7 0.1716666667    0.219
## 8   8 0.1810000000    0.227
## 9   9 0.1821111111    0.219
## 10 10 0.1891111111    0.227
## 11 11 0.1880000000    0.238
## 12 12 0.1953333333    0.227
## 13 13 0.1954444444    0.230
## 14 14 0.1986666667    0.237
## 15 15 0.2026666667    0.238
## 16 16 0.2038888889    0.239
## 17 17 0.2067777778    0.239
## 18 18 0.2125555556    0.237
## 19 19 0.2123333333    0.245
## 20 20 0.2132222222    0.245
## 21 21 0.2152222222    0.251
## 22 22 0.2166666667    0.251
## 23 23 0.2174444444    0.259
## 24 24 0.2172222222    0.248
## 25 25 0.2175555556    0.252
## 26 26 0.2216666667    0.247
## 27 27 0.2232222222    0.251
## 28 28 0.2246666667    0.250
## 29 29 0.2247777778    0.256
## 30 30 0.2281111111    0.262
```

d) **Extra:** Plot the test and training error rates against k. Which k should be chosen based on this quick analysis? Why is this the best k?

```
plot(knn.res$k, knn.res$test.err, type = 'l',
     ylim = c(0, 0.28), col='red', xlab='k', ylab='error rate')
lines(knn.res$train.err, col='blue', type='l')
legend('bottomright', legend = c('test error','train error'),
       lty=c(1,1), col=c('red','blue'))
```




```
knn.res[which.min(knn.res$test.err),]
```

```
##   k train.err test.err
## 7 7 0.1716667    0.219
```

Based on this assessment k=7 is the best choice because it showed the lowest test error (0.219). Your results can differ slightly because of possible ties when classifying test data with the knn. In case of ties the `knn()` function in R uses a probability to select the winner. Depending on how you implemented the loop in the previous exercise this can influence the test error rates.