Exercise:
K Nearest Neighbor

Introduction to Machine Learning with R

SOLUTION

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. Create a pairs plot to get a visual impression.

```
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. In the pairs plot (not shown) we can see that most of the morphological variables are strongly correlated with each other.

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 (`FL`, `RW`, `CL`, `CW`, `BD`). (**Hint**: `knn()`, you need to load the package `class` for the `knn()` function)

```
morph <- crabs[,4:8]
library(class)
knn_pred <- knn(train = morph,
                cl = crabs$sp,
                test = morph,
                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
##  [38] B 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
##  [75] 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
## [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
## [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
## [186] 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.

```
# Confusion matrix:
confT <- table(knn_pred, crabs$sp)
confT
```

```
##
## knn_pred  B  O
##        B 98  1
##        O  2 99
```

```
# Training error:
missT <- confT
diag(missT) <- 0
trainErr <- sum(missT)/sum(confT)
trainErr
```

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

Exercise 2: Cardiotocography data set

The Cardiotocography data set is a collection of fetal cardiotocograms and the associated diagnostic features. Each cardiotocogram was also classified by three expert obstetricians with regard to the fetal state (normal, suspect or pathologic). See https://archive.ics.uci.edu/ml/datasets/Cardiotocography for more information. We work with a reduced version of the data set, which only includes a sample of the total records.

a) This data set is supplied as a .csv file. Read in the data with the command `fet <- read.csv('Cardiotocography.csv', stringsAsFactors=TRUE)` and get a first impression of the data. There should only be one factor in the data which is the status of the fetus. How many rows does the data contain? How many observations are there with `normal`, `suspect` and `pathologic` status, respectively?

```
fet <- read.csv('Cardiotocography.csv', stringsAsFactors = TRUE)
str(fet)
```

```
## 'data.frame':    671 obs. of  9 variables:
##  $ NSP     : Factor w/ 3 levels "normal","pathologic",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ DP      : num  -0.227 -0.216 -0.638 -0.274 -0.315 ...
##  $ AC      : num  -0.162 0.696 -0.416 4.528 1.966 ...
##  $ UC      : num  1.398 -0.942 1.083 1.326 1.979 ...
##  $ DL      : num  2.478 -0.515 -0.538 -0.584 3.61 ...
##  $ Variance: num  -0.221 -0.414 -0.514 1.007 2.04 ...
##  $ FM      : num  -0.188 -0.164 -0.217 -0.212 -0.212 ...
##  $ MLTV    : num  -0.5794 2.0466 0.0667 4.2832 -1.3662 ...
##  $ DS      : num  0.0307 -0.7643 1.0959 0.7222 -0.0534 ...
```

```
table(fet$NSP)
```

```
##
##     normal pathologic    suspect
##        200        176        295
```

There are 200 observations for the status normal, 176 observations for the status pathologic and 295 observations for the status suspect.

b) We want to compare the training and test error when using k-nearest neighbor classification on the cardiotocography data. The goal is to predict the status of a fetus based on the available measurements. Remove a random sample from the data (size=200), we will use this subsample as a test data set. To take a random sample from a data frame, we can make use of the `sample` function. The `sample` function can be given a vector as an argument, and it will take a random sample of a specified size from this vector:

```
sample(1:10, size = 4)
```

```
## [1] 4 2 6 1
```

Try to use the `sample` function to extract 200 random rows from our data. The remaining data will be used as training data. Apply KNN to the training data with k=4 and calculate the training and test error.

```
set.seed(1121)
s <- 200
test_ind <- sample(1:nrow(fet), size = s)
```

```
fet_test <- fet[test_ind,]
fet_train <- fet[-test_ind,]
# Apply KNN:
# For training error:
knn_pred_tr <- knn(train = fet_train[,-1], # We have to remove the target variable (column 1)
                   cl = fet_train$NSP,
                   test = fet_train[,-1],
                   k = 4)
# For test error:
knn_pred_te <- knn(train = fet_train[,-1],
                   cl = fet_train$NSP,
                   test = fet_test[,-1],
                   k = 4)
# Confusion matrix (training data):
confT <- table(knn_pred_tr, fet_train$NSP)
confT
```

```
##
## knn_pred_tr  normal pathologic suspect
##   normal        110          2      11
##   pathologic      3         99      16
##   suspect        29         18     183
```

```
# Training error:
missT <- confT
diag(missT) <- 0
trainErr <- sum(missT)/sum(confT)
trainErr
```

```
## [1] 0.1677282
```

```
# Confusion matrix (test data):
confT_te <- table(knn_pred_te, fet_test$NSP)
confT_te
```

```
##
## knn_pred_te  normal pathologic suspect
##   normal         40          0       4
##   pathologic      1         35       7
##   suspect        17         22      74
```

```
# Test error:
missT <- confT_te
diag(missT) <- 0
testErr <- sum(missT)/sum(confT_te)
testErr
```

```
## [1] 0.255
```

The test error (0.255) is higher than the training error (0.168), which is not further surprising. Because we drew a random test sample from the original data your results can vary from the ones here (to reproduce the results the same seed has to be set).

c) 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 with k ranging from 1 to 30. Collect for each k the training error and the test error in a table.

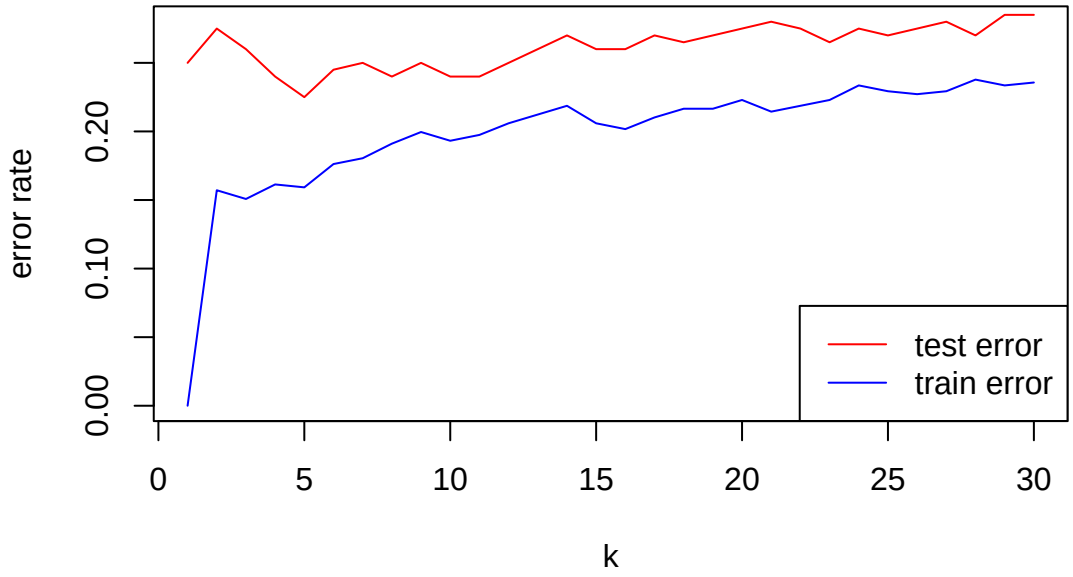
```r
ks <- 1:30
train_err <- NA
test_err <- NA
fet_tr_sc <- fet_train[,-1]
fet_te_sc <- fet_test[,-1]
set.seed(2456)
for (i in 1:length(ks)){
  # Test error
  knn_pred_te <- knn(train = fet_tr_sc,
                cl = fet_train$NSP,
                test = fet_te_sc,
                k = ks[i])
  confT_te <- table(knn_pred_te, fet_test$NSP)
  missT <- confT_te
  diag(missT) <- 0
  test_err[i] <- sum(missT)/sum(confT_te)
  # Training error
  knn_pred_tr <- knn(train = fet_tr_sc,
                cl = fet_train$NSP,
                test = fet_tr_sc,
                k = ks[i])
  confT_tr <- table(knn_pred_tr, fet_train$NSP)
  missT <- confT_tr
  diag(missT) <- 0
  train_err[i] <- sum(missT)/sum(confT_tr)
  #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 0.0000000    0.250
## 2   2 0.1571125    0.275
## 3   3 0.1507431    0.260
## 4   4 0.1613588    0.240
## 5   5 0.1592357    0.225
## 6   6 0.1762208    0.245
## 7   7 0.1804671    0.250
## 8   8 0.1910828    0.240
## 9   9 0.1995754    0.250
## 10 10 0.1932059    0.240
## 11 11 0.1974522    0.240
## 12 12 0.2059448    0.250
## 13 13 0.2123142    0.260
## 14 14 0.2186837    0.270
## 15 15 0.2059448    0.260
## 16 16 0.2016985    0.260
## 17 17 0.2101911    0.270
## 18 18 0.2165605    0.265
## 19 19 0.2165605    0.270
## 20 20 0.2229299    0.275
## 21 21 0.2144374    0.280
## 22 22 0.2186837    0.275
```

```
## 23 23 0.2229299    0.265
## 24 24 0.2335456    0.275
## 25 25 0.2292994    0.270
## 26 26 0.2271762    0.275
## 27 27 0.2292994    0.280
## 28 28 0.2377919    0.270
## 29 29 0.2335456    0.285
## 30 30 0.2356688    0.285
```

d) 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(x = knn_res$k, y = knn_res$test_err, type = 'l',
     ylim = c(0, 0.28), col='red', xlab='k', ylab='error rate')
lines(x = knn_res$k, y = knn_res$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
## 5 5 0.1592357    0.225
```

Based on this assessment k=5 is the best choice because it showed the lowest test error (0.225). 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.