k-Nearest Neighbours Machine Learning in R

Zhongjiu Lu

21 November 2017

```r
# load neccessary packages
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(class)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
library(gmodels)
```

loads data file into R

```r
wine <- data.frame(read.csv('wine-quality-white.csv', sep = ';'))
head(wine)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.0             0.27        0.36           20.7     0.045
## 2           6.3             0.30        0.34            1.6     0.049
## 3           8.1             0.28        0.40            6.9     0.050
## 4           7.2             0.23        0.32            8.5     0.058
## 5           7.2             0.23        0.32            8.5     0.058
## 6           8.1             0.28        0.40            6.9     0.050
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  45                  170  1.0010 3.00      0.45     8.8
## 2                  14                  132  0.9940 3.30      0.49     9.5
## 3                  30                   97  0.9951 3.26      0.44    10.1
## 4                  47                  186  0.9956 3.19      0.40     9.9
## 5                  47                  186  0.9956 3.19      0.40     9.9
## 6                  30                   97  0.9951 3.26      0.44    10.1
##   quality
## 1       6
## 2       6
## 3       6
## 4       6
## 5       6
## 6        6
```

construct a new binary column "good wine", which is defined by quality of 6 or higher, otherwise, it is not a "good wine"

```r
wine <- wine %>% mutate(good_wine = ifelse
                        (wine['quality'] >= 6, 'good', 'bad'))
```

splits the data set into a training data set (~40%), a validation data set (~30%) and a test data set (~30%)

firstly shuffle the records

```r
# set the seed for consistent computation
set.seed(101)

# create the shuffled index
index <- sample(seq(1, 3), size = nrow(wine),
                replace = TRUE,
                prob = c(0.4, 0.3, 0.3))
# training data set (~40%)
training <- wine[index == 1, ]
# validation data set (~30%)
validation <- wine[index == 2, ]
# test data set (~30%)
test <- wine[index == 3, ]
```

normalises the data according to the Z-score transform

```r
# training set of predictors
training_x <-
  as.data.frame(scale(select(training,-one_of(
  c('quality', 'good_wine')
  ))))

# training set of response
training_y <- training[,"good_wine"]

# obtain the mean and sd value of training data by column
train_mean <- apply((select(training, -one_of(
  c('quality', 'good_wine')
  ))), 2, mean)

train_sd <- apply((select(training, -one_of(
  c('quality', 'good_wine')
  ))), 2, sd)

# using training set's mean and sd to perform z-score normalisation in validation data set

# validation set of predictors
validation_x <- as.data.frame(scale((select(
  validation,-one_of(c('quality', 'good_wine'))
  )), train_mean, train_sd))

# validation set of response
validation_y <- validation[, "good_wine"]

# test set of predictors
test_x <- as.data.frame(scale((select(test, -one_of(
  c('quality', 'good_wine')
  ))), train_mean, train_sd))

# test set of response
test_y <- test[, "good_wine"]
```

trains the k-Nearest Neighbours classifiers for k = 1, 2,. . . , 80

For training our data, we use the "knn()" function, which calculates the 'k' nearest neighbors (in Eculidean distance). Our performance measure for this classification problem is the accuracy (1 - misspecification error) and the optimal value for 'k' is obtained when the accuracy is at its highest.

```r
# create empty list to save knn model's result
knn.train <- list()

# training knn model for k = 1, 2, ..., 80
for(k in 1:80) {
  knn.train[[k]] <- knn(training_x,
                         validation_x,
                         training_y,
                         k = k)}
```

evaluates each classifier on the validation set and selects the best classifier

In classification problems, there are different evaluation methods (e.g. accuracy, precision, recall), depending on the objectives of the model. In our case, we choose accuracy because we are concerned about if our model and classify wines correctly in general.

```r
# empty vector to store the accuracy measure
accuracy <- c()

# calculate each classifier's performance
for(k in 1:80) {
  accuracy[k] <-
  sum(knn.train[[k]] == validation_y) / length(validation_y)}
```

```r
# obtain the highest accuracy with k value
best.k <- which.max(accuracy); best.k
```

```
## [1] 15
```

predicts the generalisation error using the test data set, as well as well as a confusion matrix.

```r
knn.best <- knn(training_x, test_x, training_y, k = best.k)
confusionMatrix(knn.best, test_y, positive = 'good')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad  264  132
##       good 209  831
##
##                Accuracy : 0.7625
##                  95% CI : (0.7397, 0.7843)
##     No Information Rate : 0.6706
##     P-Value [Acc > NIR] : 1.567e-14
##
##                   Kappa : 0.4393
##  Mcnemar's Test P-Value : 3.861e-05
##
##             Sensitivity : 0.8629
##             Specificity : 0.5581
##          Pos Pred Value : 0.7990
##          Neg Pred Value : 0.6667
##              Prevalence : 0.6706
##          Detection Rate : 0.5787
##    Detection Prevalence : 0.7242
##       Balanced Accuracy : 0.7105
##
##        'Positive' Class : good
##
```

The k value ranges from 1 and 80. The accuracy (1 - misspecification error) is used as the performance measure; hence the optimal "k" is obtained when the accuracy is at a maximum (which is the same as when the misspecification error is at a minimum) k = 15 with 76.67% accurancy rate. However, we should note that accuracy itself is not always enough to judge if a classifier is well-suited for the data set or not. Considering the good wine is the category of interest, the specificity is only 55.81% which means the model does not accurately predict if the wine is bad. This might be the result from having higher number of good wine in the dataset.