Computer lab 1 block 3 (732A99 Machine Learning)

Lennart Schilling (lensc874)

17 December 2018

Contents

Assignment 1: Kernel methods **1**

Assignment 2: Support Vector Machines **1**

Appendix **3**

Assignment 1: Kernel methods

Assignment 2: Support Vector Machines

In this assignment, a Support Vector Machine (SVM), a supervised learning model, will be used to classify spam dataset that is included within the *kernlab*-package which will be used for this assignment.

In the first step, the package will be loaded, attached and the *spam*-data frame will be read.

```
# loading/attaching kernlab library
library(kernlab)
# importing data
data(spam)
# printing nrow & ncol
dim(spam)
```

```
## [1] 4601   58
```

The *spam* data consists of 4601 observations emails described by in total 58 features. The *type*-feature classifies the mails as either *spam* or *nonspam*.

The model selection will be based on the *holdout method* which uses training data to fit the model and test data to evaluate it. To make sure that enough observations are integrated within both data sets, a relation of 70:30 will be chosen.

```
# dividing data into train and test set
n = dim(spam)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.7))
train = spam[id,]
test = spam[-id,]
```

Using the training data and the *radial basis function (RBF) kernel* with a width of 0.05, three different SVM-models with a different C-parameter (0.5, 1 and 5) will be fitted. Within every iteration, the test misclassification error respectively will be calculated. If it will be identified as the lowest error, the model will be saved as *bestModel*.

```
# setting up minimum misclassification rate to 1 for following loop
minMisclassificationRate = 1
# fitting svm-models for different parameter C
for (C in c(0.5, 1, 5)) {
```

```
  svmModel = ksvm(type ~ .,
                  data = spam,
                  kernel = "rbfdot",
                  kpar = list(sigma = 0.05),
                  C = C)
  classification = predict(svmModel , test[, -which(colnames(train) == "type")])
  misclassificationRate = mean(test$type != classification)
  print(paste0("Test misclassification error for C = ", C, ": ", round(misclassificationRate, 3)))
  if (misclassificationRate < minMisclassificationRate) {
    minMisclassificationRate = misclassificationRate
    bestModel = svmModel
  }
}
```

```
## [1] "Test misclassification error for C = 0.5: 0.043"
## [1] "Test misclassification error for C = 1: 0.036"
## [1] "Test misclassification error for C = 5: 0.023"
```

Based on the lowest test error, the model using $C = 5$ will be identified as the most optimal classifier. In the following, it is summarised:

```
# returning parameter C and erros of best model
knitr::kable(
  x = as.data.frame(
    cbind(C = bestModel@param,
          trainError = round(bestModel@error, 3),
          testError = round(minMisclassificationRate, 3))),
  caption = "Summary of best identified svm model",
  row.names = FALSE)
```

Table 1: Summary of best identified svm model

C	trainError	testError
5	0.022	0.023

```
# returning confusion matrix for best model
knitr::kable(
  table(y = test$type, yFit = predict(bestModel, test[, -which(colnames(train) == "type")])),
  caption = "Confusion matrix for best identified svm model")
```

Table 2: Confusion matrix for best identified svm model

	nonspam	spam
nonspam	825	10
spam	22	524

Comparing the train and test error, it can be clearly seen that both values are very similar which indicates that neither too much overfitting nor underfitting seem to occur.

The parameter C decides about if the classifier accepts a few more misclassifications for a bigger margin hyperplane (in case of a small C) or if it aims for better classification of all the training observations choosing a smaller margin hyperplane (in case of a large C). The higher C the more misclassifications will be penalised.

Appendix

```r
# loading/attaching kernlab library
library(kernlab)
# importing data
data(spam)
# printing nrow & ncol
dim(spam)
# dividing data into train and test set
n = dim(spam)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.7))
train = spam[id,]
test = spam[-id,]
# setting up minimum misclassification rate to 1 for following loop
minMisclassificationRate = 1
# fitting svm-models for different parameter C
for (C in c(0.5, 1, 5)) {
  svmModel = ksvm(type ~ .,
                  data = spam,
                  kernel = "rbfdot",
                  kpar = list(sigma = 0.05),
                  C = C)
  classification = predict(svmModel , test[, -which(colnames(train) == "type")])
  misclassificationRate = mean(test$type != classification)
  print(paste0("Test misclassification error for C = ", C, ": ", round(misclassificationRate, 3)))
  if (misclassificationRate < minMisclassificationRate) {
    minMisclassificationRate = misclassificationRate
    bestModel = svmModel
  }
}
# returning parameter C and erros of best model
knitr::kable(
  x = as.data.frame(
    cbind(C = bestModel@param,
          trainError = round(bestModel@error, 3),
          testError = round(minMisclassificationRate, 3))),
  caption = "Summary of best identified svm model",
  row.names = FALSE)
# returning confusion matrix for best model
knitr::kable(
  table(y = test$type, yFit = predict(bestModel, test[, -which(colnames(train) == "type")])),
  caption = "Confusion matrix for best identified svm model")
```