Exercises Support Vector Machines

Marjolein Fokkema

Fit SVMs for predicting biodegradability of molecules

Read in data and separate the data in a training and test set:

```r
qsar <- readRDS("qsar.Rda")
set.seed(42)
train <- sample(1:nrow(qsar), size = 700)
test <- which(!(1:nrow(qsar) %in% train))
library("e1071")
```

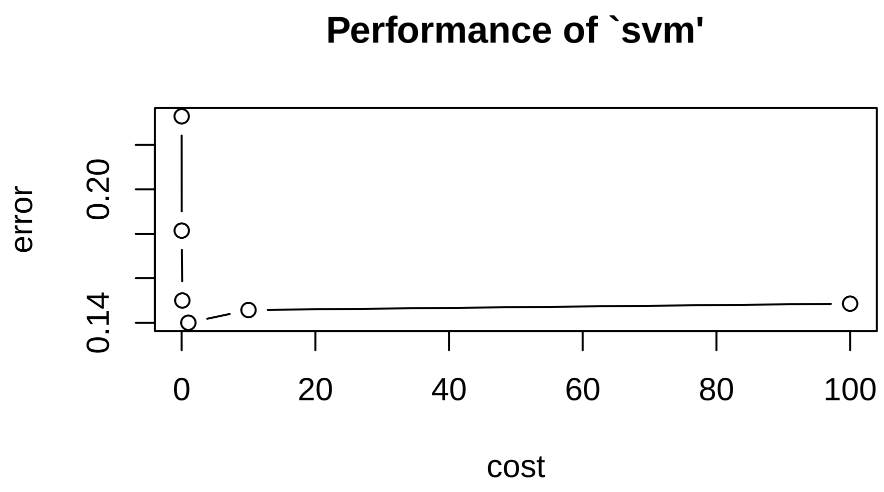
Part 1: Linear kernel

Tune and fit the SVM with linear kernel:

```r
set.seed(42)
cost <- c(.001, .01, .1, 1, 10, 100)
tune.out <- tune(svm, class ~ ., data = qsar[train, ], kernel = "linear",
                 ranges = list(cost = cost))
```

a)

```r
plot(tune.out)
```



```r
tune.out$performances
```

```
##    cost     error dispersion
## 1 1e-03 0.2328571 0.07679817
## 2 1e-02 0.1814286 0.03929654
## 3 1e-01 0.1500000 0.03030458
## 4 1e+00 0.1400000 0.02590756
## 5 1e+01 0.1457143 0.02409354
## 6 1e+02 0.1485714 0.02352207
```

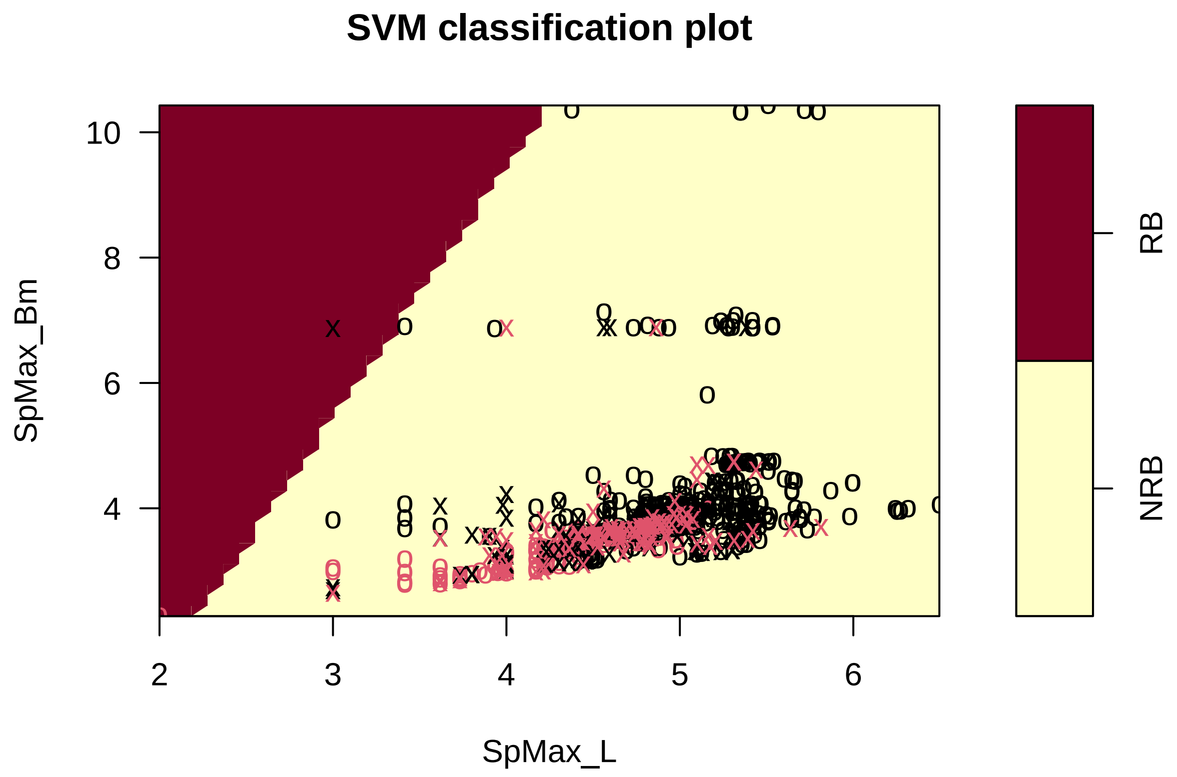
```r
tune.out$best.parameters
```

```
##   cost
## 4    1
```

We obtained a convex curve, so we need not expand the range of the cost parameter. Perhaps we might want to try a finer grid around the optimal value of the cost parameter, but here we do not aim for perfect optimization of performance.

b)

```r
svmfit <- svm(class ~ ., data = qsar[train, ], kernel = "linear",
              cost = tune.out$best.parameters)
slice <- as.list(sapply(qsar[ , - which(names(qsar) == "class")], mean))
plot(svmfit, qsar[train, ], SpMax_Bm ~ SpMax_L, slice = slice)
```



In the plot, x is a support vector, o are for other data points. Colors represent observed classes (black for non-biodegradable, red for biodegradable). The yellow area indicates where observations would be classified as non-biodegradable, the red area indicates where observations would be classified as biodegradable.

We see that a linear decision boundary was applied for classifying the observations, because the red and yellow areas are separated by a straight line. Note that the ruggedness of the line is a plotting artifact, it is not really present in the fitted predictive model. Note also that in practice, one would likely not make these plots, and treat the SVM simply like a black-box prediction machine. The plots can be interesting and helpful to understand how SVMs work, that's why we inspect the plots in this exercise.

c)

```r
tab <- table(true = qsar[test, "class"],
             pred = predict(svmfit, newdata = qsar[test, ]))
tab
```

```
##      pred
## true  NRB  RB
##   NRB 231  16
##   RB   18  90
```

```r
prop.table(tab)
```

```
##      pred
## true         NRB         RB
##   NRB 0.65070423 0.04507042
##   RB  0.05070423 0.25352113
```

```r
1 - sum(diag(prop.table(tab))) ## MCR (correct classifications are on diagonal)
```

```
## [1] 0.09577465
```

The correct classification rate (CCR) on test data is 0.9042254, the MCR on test data is 0.0957746.

Part 2: Polynomial kernel

Tune the SVM with polynomial kernel:

```r
set.seed(42)
tune.out <- tune(svm, class ~ ., data = qsar[train, ],
                 kernel = "polynomial", ranges = list(cost = cost, degree = 2:5))
```

a) A polynomial kernel of degree 1 equals the linear kernel, so this value need not be tried anymore.

b)

```r
tune.out$best.parameters
```

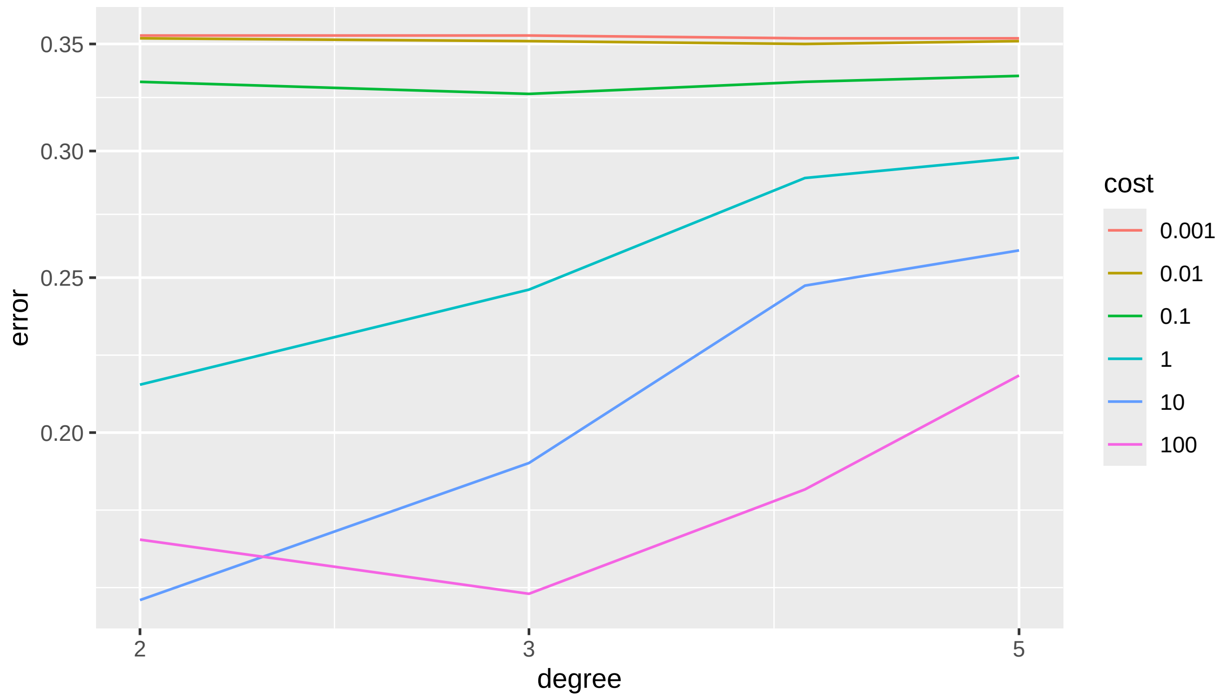
```
##   cost degree
## 5   10      2
```

```r
library("ggplot2")
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:e1071':
##
##     element
```

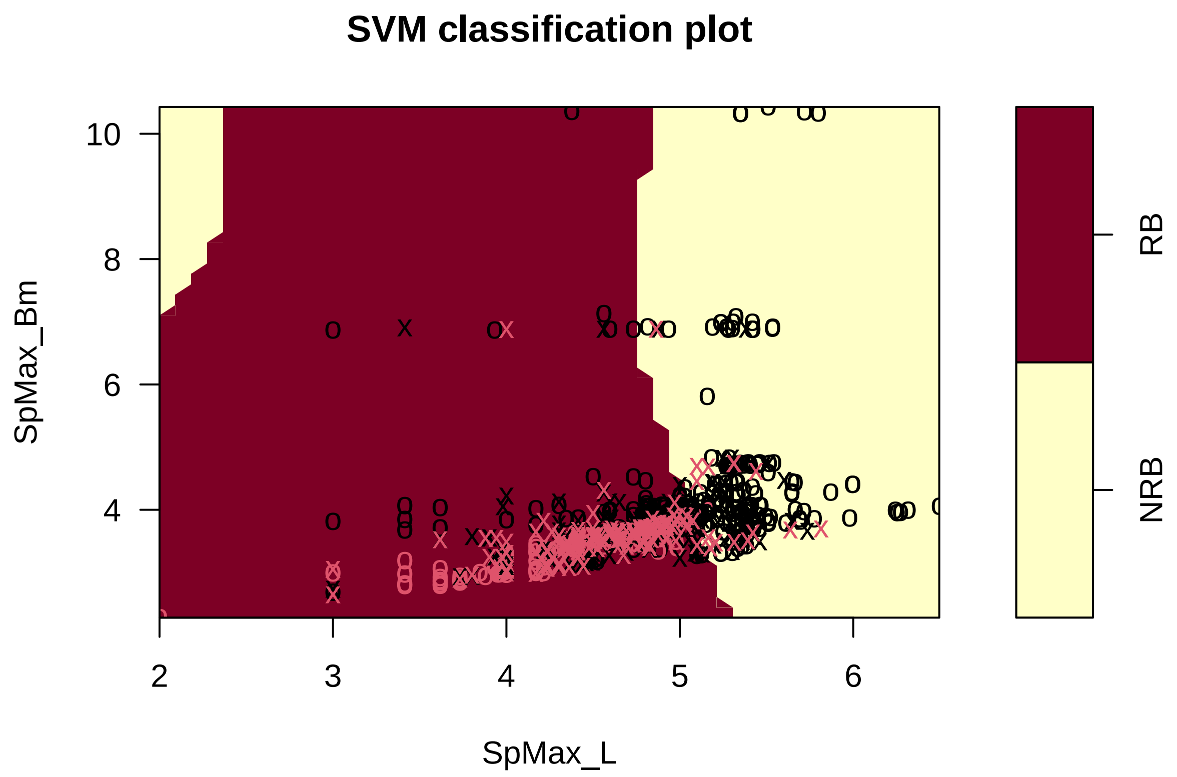
```r
perf <- tune.out$performances
perf$cost <- factor(perf$cost)
ggplot(perf) + geom_line(aes(degree, error, group=cost, color=cost)) +
  scale_x_log10() + scale_y_log10()
```



The best-performing degree is 2. Although the plot suggests a degree of 1 might perform better, we need not expand the range for this parameter because we have already obtained that model in Part 1.

c) Fit the SVM with polynomial kernel with optimal `cost` and `degree` and plot the decision boundary:

```r
svmfit <- svm(class ~ ., data = qsar[train, ], kernel = "polynomial",
              order = tune.out$best.parameters$degree,
              cost = tune.out$best.parameters$cost)
slice <- as.list(sapply(qsar[ , - which(names(qsar) == "class")], mean)-.2)
plot(svmfit, qsar[train, ], SpMax_Bm ~ SpMax_L, slice = slice)
```



The decision boundary is no longer linear.

Compute test MCR:

```
tab <- table(true = qsar[test, "class"],
             pred = predict(svmfit, newdata = qsar[test, ]))
tab
```

```
##      pred
## true  NRB  RB
##   NRB 224  23
##   RB   33  75
```

```
1 - sum(diag(prop.table(tab))) ## MCR
```

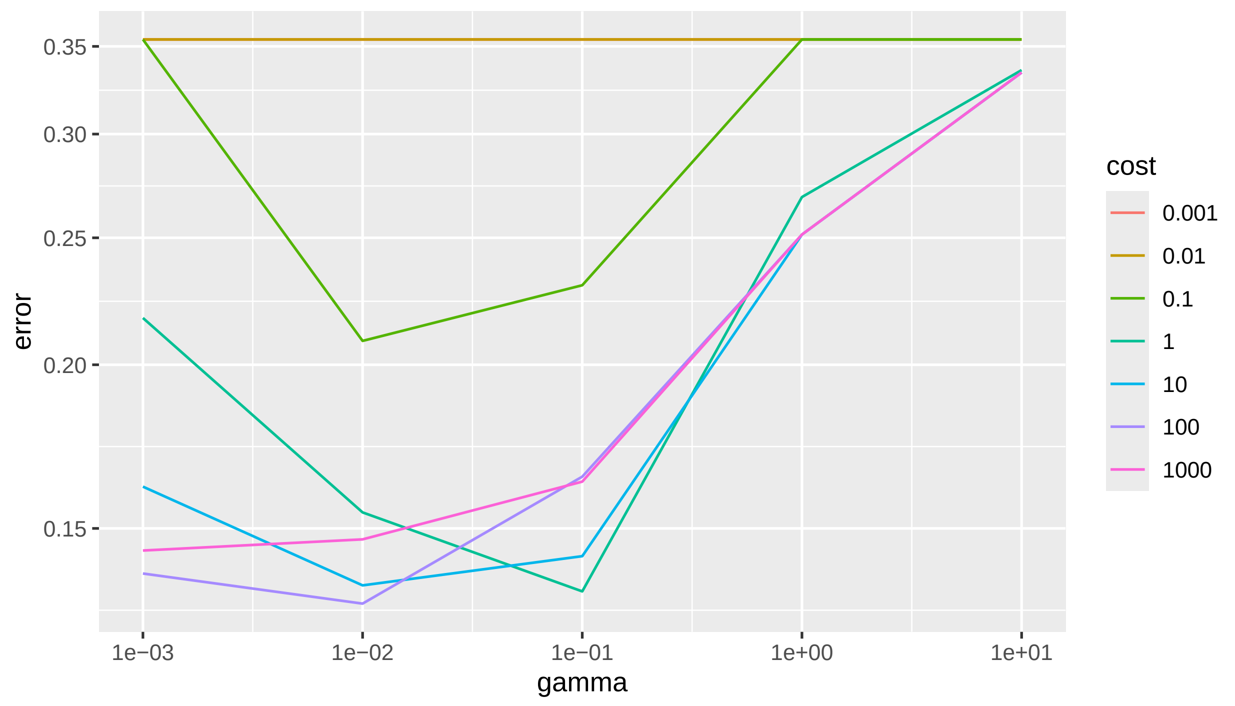
```
## [1] 0.1577465
```

d) In this data problem, the polynomial kernel SVM did not outperform the linear kernel SVM. Note that this was already suggested by the results of `tune`: the line for the cost parameter of 10 suggests lowest error is obtained at degree of 1.

Part 3: Radial basis kernel

```r
set.seed(42)
tune.out <- tune(svm, class ~ ., data = qsar[train, ], kernel = "radial",
                 ranges = list(cost = c(cost, 1000), gamma = c(.001, .01, .1, 1, 10)))
```

```r
perf <- tune.out$performances
perf$cost <- factor(perf$cost)
ggplot(perf) + geom_line(aes(gamma, error, group=cost, color=cost)) +
  scale_x_log10() + scale_y_log10()
```



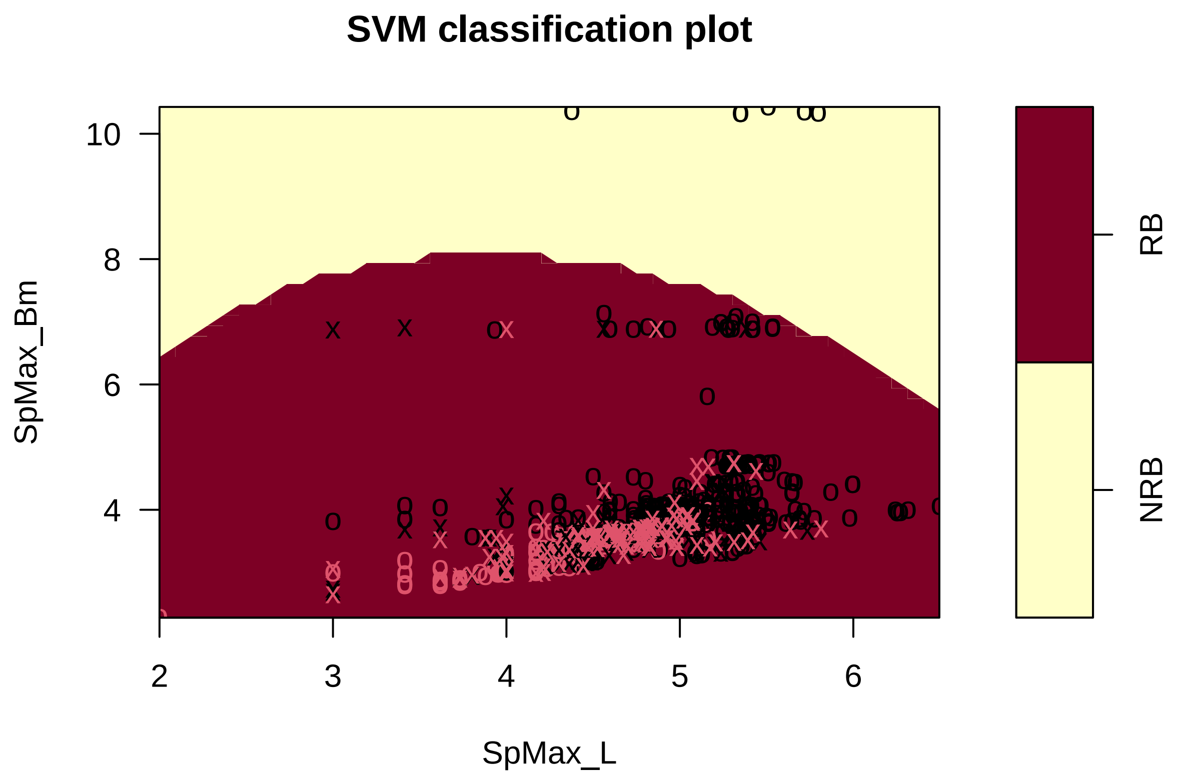
```r
tune.out$best.parameters
```

```
##    cost gamma
## 13  100  0.01
```

We obtained a convex curve for almost each value of the cost parameter. For all values of the cost parameter, gamma values of .01 or .001 seem to perform best. Te test whether the cost parameter should take a value higher than 100, I already expanded the grid with a value of 1,000 above. The cost value of 100 is also optimal.

b) Fit the radial-basis kernel SVM with optimal parameters and again plot the decision boundary as a function of `SpMax_L` and `SpMax_Bm`:

```r
svmfit <- svm(class ~ ., data = qsar[train, ], kernel = "radial",
              gamma = tune.out$best.parameters$gamma,
              cost = tune.out$best.parameters$cost)
slice <- as.list(sapply(qsar[ , - which(names(qsar) == "class")], mean)-.25)
plot(svmfit, qsar[train, ], SpMax_Bm ~ SpMax_L, slice = slice)
```



Again we obtained a non-linear decision boundary. Note that the ruggedness of the decision boundary we see is due to the limitations of the plotting method, the decision boundary is actually smooth.

c) Assess training and test error:

```
tab <- table(true = qsar[test, "class"],
             pred = predict(svmfit, newdata = qsar[test, ]))
tab
```

```
##      pred
## true  NRB  RB
##   NRB 223  24
##   RB   15  93
```

```
1 - sum(diag(prop.table(tab))) ## MCR
```

```
## [1] 0.1098592
```

The radial basis kernel also did not outperform the linear kernel.