Support Vector Machines Exercises

Simone Collier

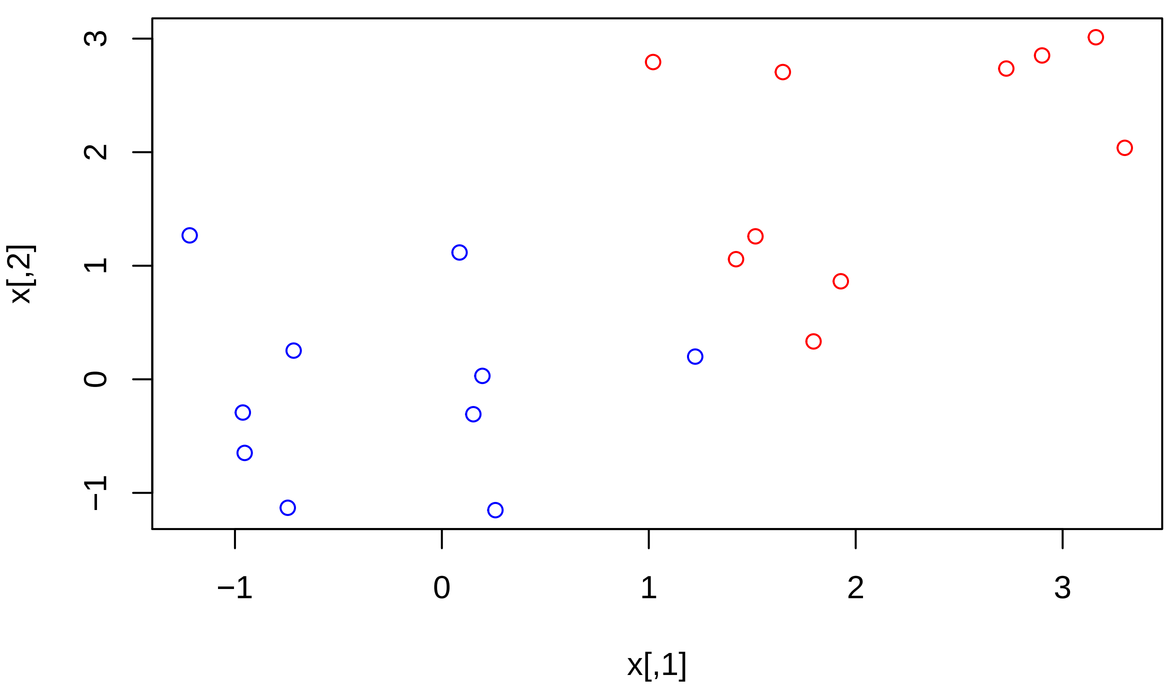
Maximal Margin Classifier

Start by loading the packages we need. If you need to install the packages first then run `install.packages("PACKAGENAME")` in your console before running the code chunk.

```
library(e1071)
library(ISLR2)
```

We want to generate a data set with two dimensional observations X. Each observation belongs to one of two classes $\{-1, 1\}$.

```
set.seed(3)
# Create a matrix of 20 observations where each row is an observation.
x <- matrix(c(rnorm(20, mean = 0), rnorm(20, mean = 2)), ncol = 2, byrow = TRUE)
# Create a vector of the classes of the observations
y <- c(rep(-1, 10), rep(1, 10))
plot(x, col = (3 - y))
```

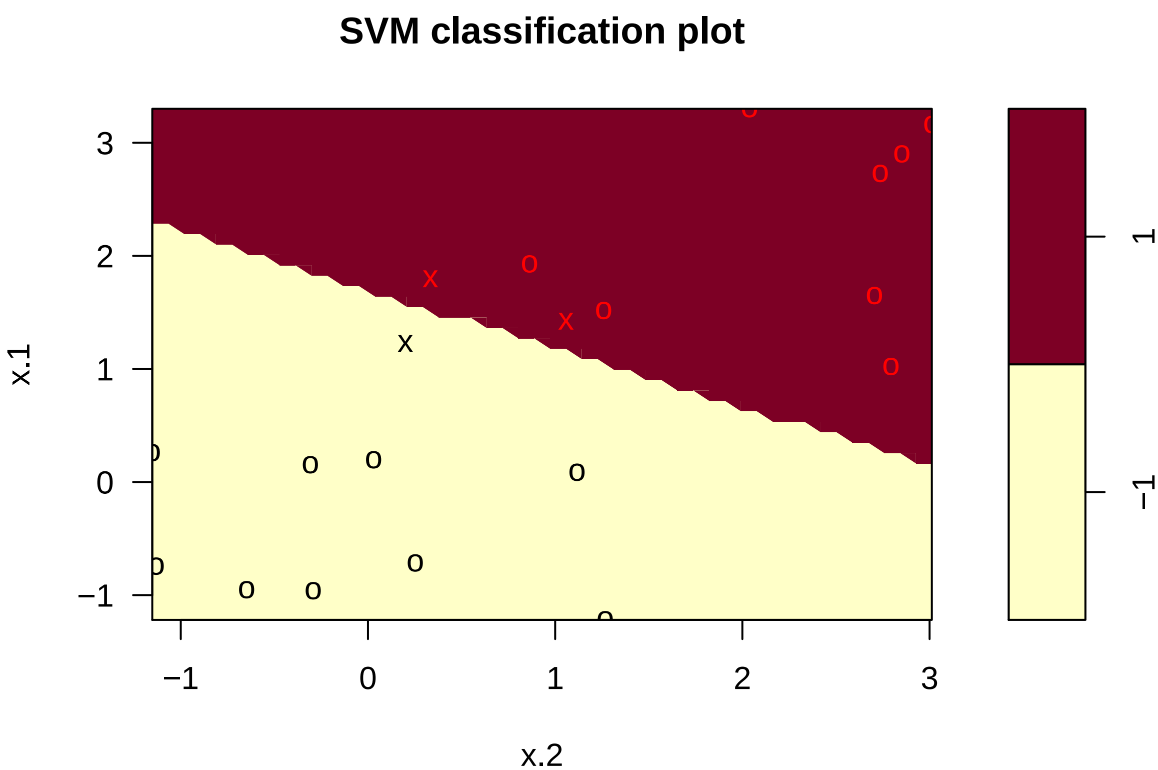


We can see that the observations from each class can be separated by a hyperplane so we can use the maximal margin classifier. To start let's put our observations in a data frame with the response coded as a factor.

```r
data <- data.frame(x = x, y = as.factor(y))
```

Now we can use the `svm()` function from the `e1071` library to fit our classifier. We choose `kernal = "linear"` set `cost` as a very large number (these parameters will be explained more later. We specify `scale = FALSE` so the function does not scale each feature to have mean zero and standard deviation one.

```r
mmcfit <- svm(y ~ ., data = data, kernel = "linear", cost = 1e10, scale = FALSE)
plot(mmcfit, data)
```



The plot makes the decision boundary look jagged but it is indeed linear. Note that `x[, 1]` and `x[, 2]` have switched axes compare to our last plot. The support vector are plotted as crosses and the remaining observations are circles.

```r
summary(mmcfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = data, kernel = "linear", cost = 1e+10,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1e+10
##
## Number of Support Vectors:  3
##
```

```
##  ( 1 2 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

This tells us that there are 3 support vectors with `( 1 2 )` indicating there are 1 from the first class and 2 from the second.

We can create a set of test observations using the same criteria as our training set.

```
set.seed(2)
xtest <- matrix(c(rnorm(20, mean = 0), rnorm(20, mean = 2)), ncol = 2, byrow = TRUE)
ytest <- c(rep(-1, 10), rep(1, 10))
testdata <- data.frame(x = xtest, y = as.factor(ytest))
```

Now we can try predicting the class labels of our test observations using our best model.

```
ypred <- predict(mmcfit, testdata)
table(predict = ypred, truth = testdata$y)
```

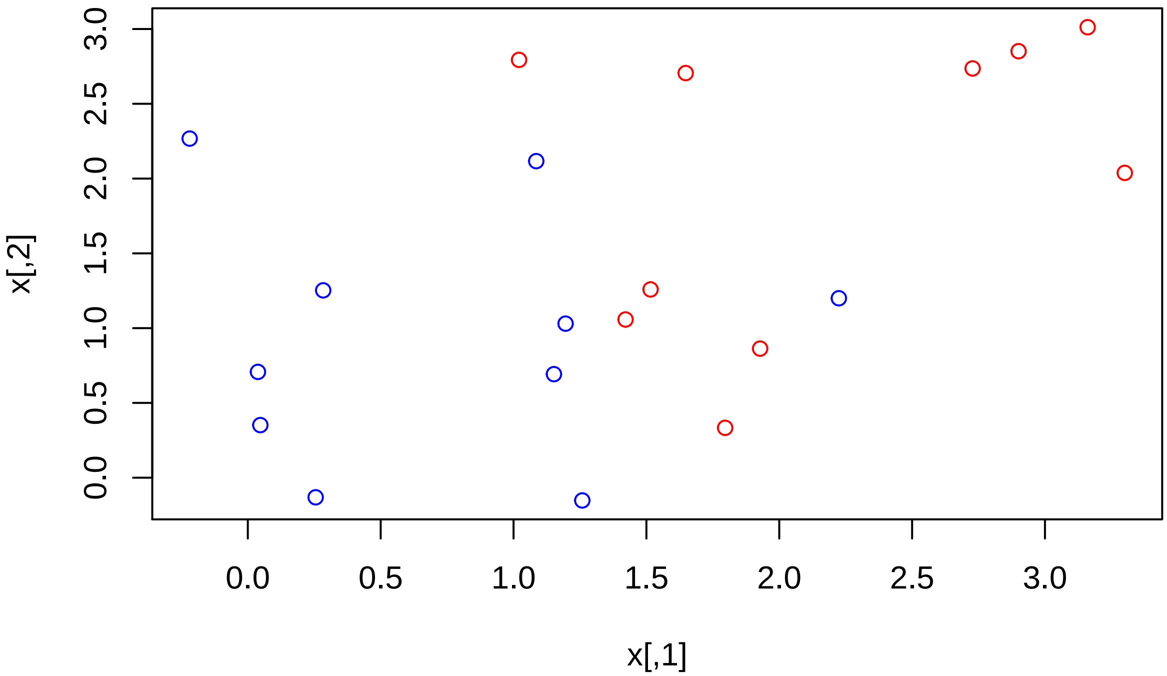
```
##        truth
## predict -1 1
##      -1  9 1
##      1   1 9
```

So all but 2 of the test observations are classified correctly.

Support Vector Classifier

We will generate a data set with two dimensional observations X. Each observation belongs to one of two classes $\{-1, 1\}$. This time we want a data set that is not seperable by a hyperplane.

```
set.seed(3)
# Create a matrix of 20 observations where each row is an observation.
x <- matrix(c(rnorm(20, mean = 1), rnorm(20, mean = 2)), ncol = 2, byrow = TRUE)
# Create a vector of the classes of the observations
y <- c(rep(-1, 10), rep(1, 10))
plot(x, col = (3 - y))
```

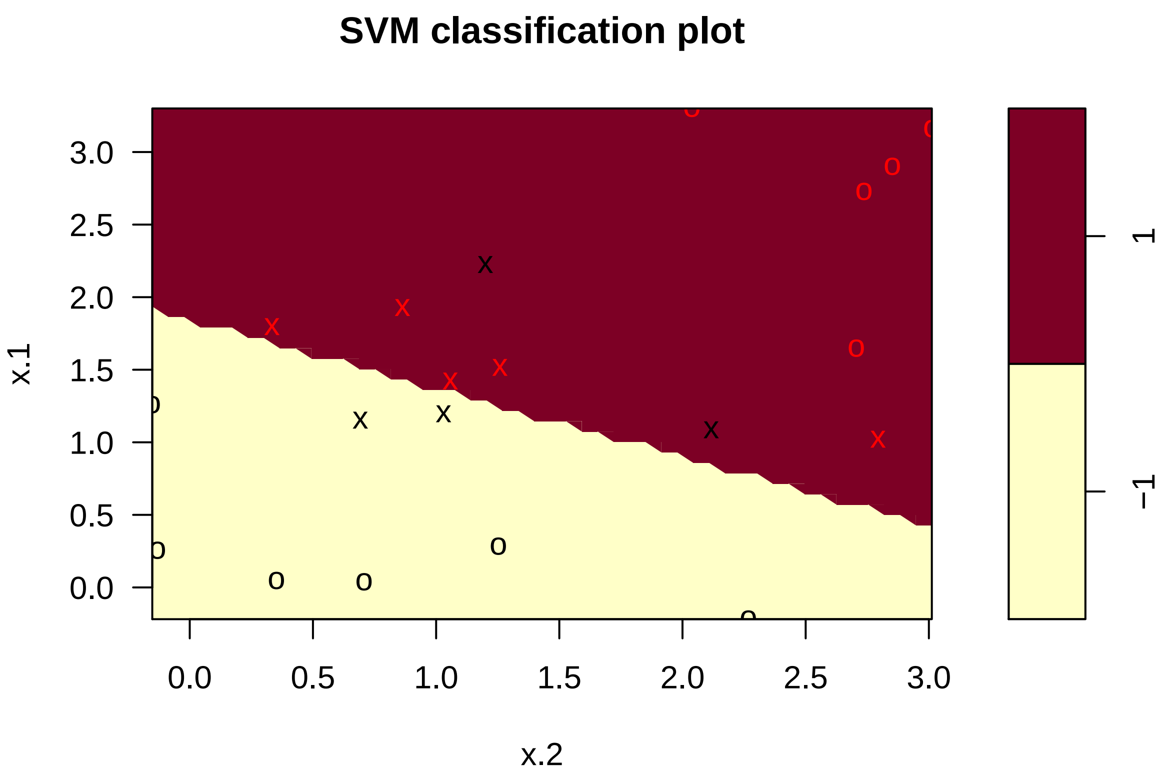


We can see that the observations from each class cannot be separated by a hyperplane so we can't use the maximal margin classifier. We will fit a support vector classifier instead. To start let's put our observations in a data frame with the response coded as a factor.

```
data <- data.frame(x = x, y = as.factor(y))
```

Now we can use the `svm()` function to fit our classifier. We choose `kernal = "linear"` to specify we want to fit a support vector classifier and set `cost = 8`.

```
svcfit <- svm(y ~ ., data = data, kernel = "linear", cost = 8, scale = FALSE)
plot(svcfit, data)
```




```
summary(svcfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = data, kernel = "linear", cost = 8, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  8
##
## Number of Support Vectors:  9
##
##  ( 4 5 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

There are 9 support vectors with 4 from the first class and 5 from the second.

Try fitting a support vector classifier with a smaller value for cost. Compare the results including the number of support vectors to the above two classifiers. From this information explain the relationship between the cost parameter and C as we have described in the slides.

We will use the `tune()` function from the `e1071` library to perform cross-validation to choose the best value for `cost`.

```r
set.seed(1)
tune.out <- tune(svm, y ~ ., data = data, kernel = "linear", ranges = list(cost = c(0.001, 0.01, 0.1, 1
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.15
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.55  0.4972145
## 2 1e-02  0.55  0.4972145
## 3 1e-01  0.35  0.3374743
## 4 1e+00  0.20  0.2581989
## 5 5e+00  0.20  0.2581989
## 6 1e+01  0.15  0.2415229
## 7 1e+02  0.15  0.2415229
```

This tells us that `cost = 10` results in the lowest cross-validation error rate. The best model found by `tune()` can be accessed using:

```r
bestsvc <- tune.out$best.model
summary(bestsvc)
```

```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = data, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  9
##
##  ( 4 5 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

We can create a set of test observations using the same criteria as our training set.

```r
set.seed(1)
xtest <- matrix(c(rnorm(20, mean = 1), rnorm(20, mean = 2)), ncol = 2, byrow = TRUE)
ytest <- c(rep(-1, 10), rep(1, 10))
testdata <- data.frame(x = xtest, y = as.factor(ytest))
```

Now we can try predicting the class labels of our test observations using our best model.

```r
ypred <- predict(bestsvc, testdata)
table(predict = ypred, truth = testdata$y)
```

```
##         truth
## predict -1 1
##      -1  4 1
##      1   6 9
```

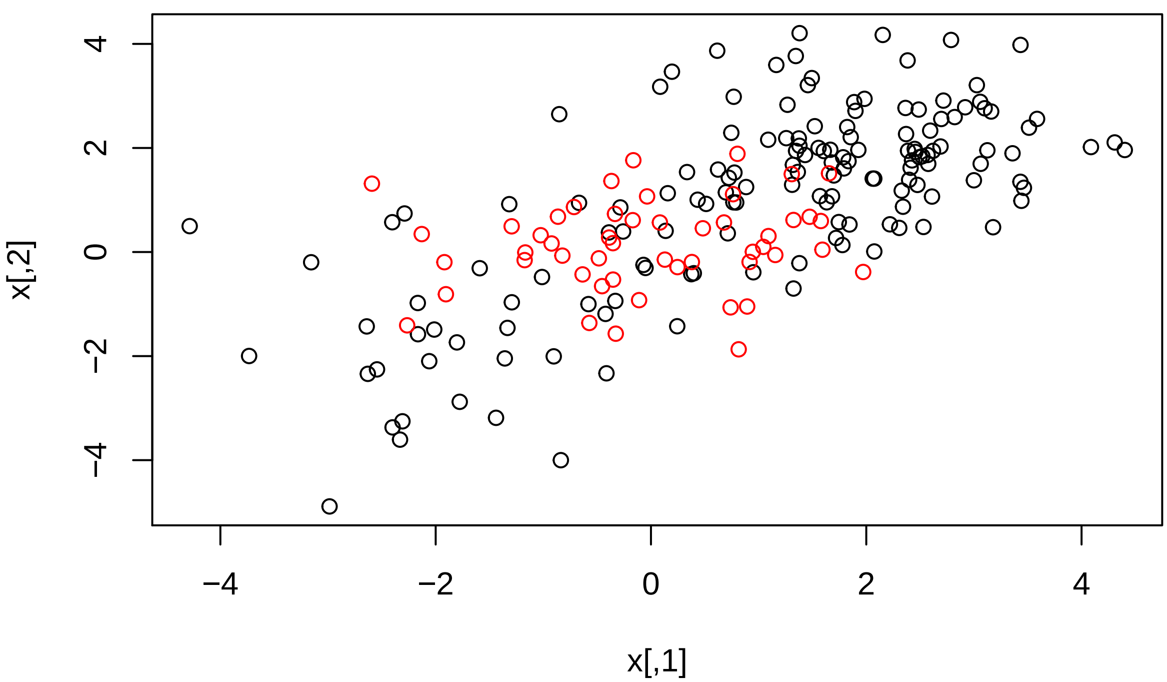
So 13 out of 20 observations are correctly classified.

Support Vector Machine

We will use the `svm()` function to fit an SVM with a non-linear kernel. We first generate some data with a non-linear class boundary.

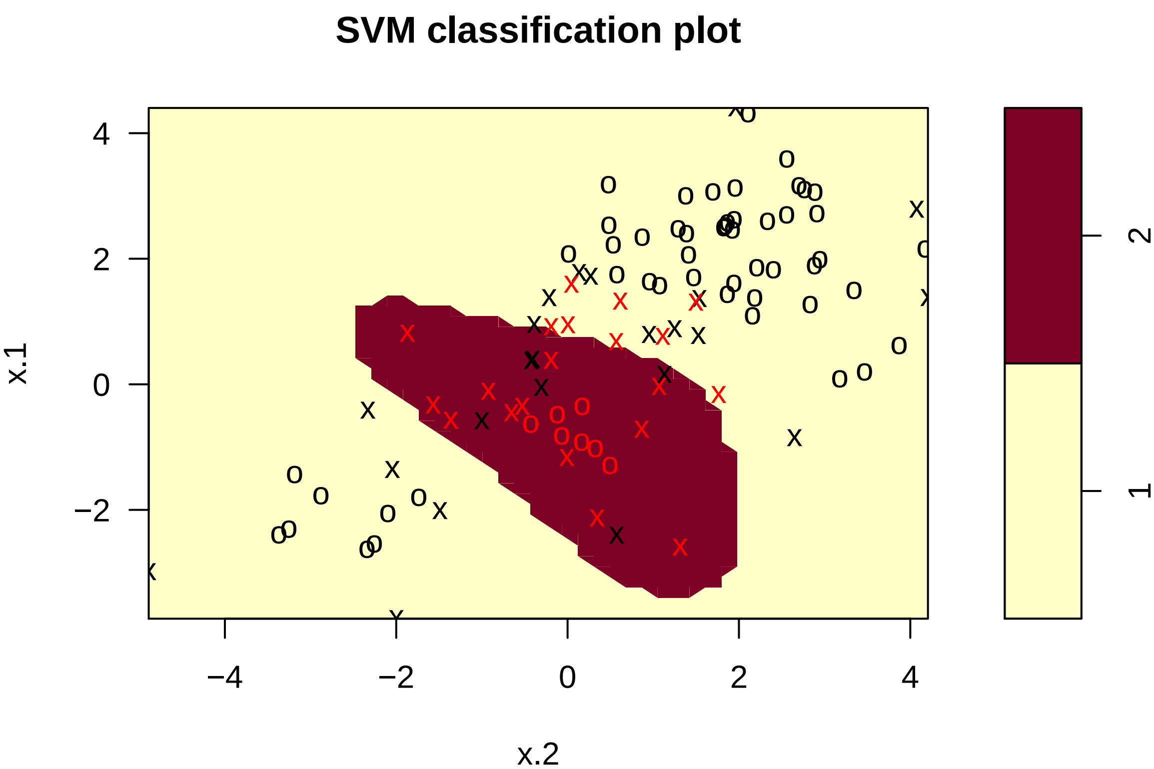
```r
set.seed(1)
x <- matrix(c(rnorm(200, mean = 2), rnorm(50, mean = -2), rnorm(150)), ncol = 2, byrow = TRUE)
y <- c(rep(1, 150), rep(2, 50))

data <- data.frame(x = x, y = as.factor(y))
plot(x, col = y)
```



We randomly split our data set into training and test sets. Then we fit the training data with a radial kernel. The argument `gamma` controls how non-linear the fit with the radial kernel can be.

```
train <- sample(200, 100)
svmfit <- svm(y ~ ., data = data[train, ], kernal = "radial", cost = 1, gamma = 1)
plot(svmfit, data[train, ])
```




```
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = data[train, ], kernal = "radial", cost = 1,
##     gamma = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  43
##
##  ( 23 20 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

Perform cross-validation using* `tune()` *to select the best choice of* `gamma` *and* `cost`*. Try the ranges* `cost = c(0.1, 1, 10, 100, 1000)` *and* `gamma = c(0.5, 1, 2, 3, 4)`*. Output the best model.

Using the best model, predict the responses for the test set. What percent of observations were misclassified by this SVM?

The `svm()` function has the capability to perform classification when there are more than two response classes. In this case the `svm()` will use the one-versus-one approach.

Support vector regression is also possible with the `svm()` function when the response is quantitative!

These exercises were adapted from : James, Gareth, et al. An Introduction to Statistical Learning: with Applications in R, 2nd ed., Springer, 2021.