Support Vector Machine Assignment

Michael J. Jones

06/05/2015

Problem 1

Suppose that the following are a set of points in two classes:

`Class 1:  (1,1) (1,2) (2,1)`

`Class 2:  (0,0) (1,0) (0,1)`

Plot them and find the optimal separating line. What are the support vectors, and what is the margin?

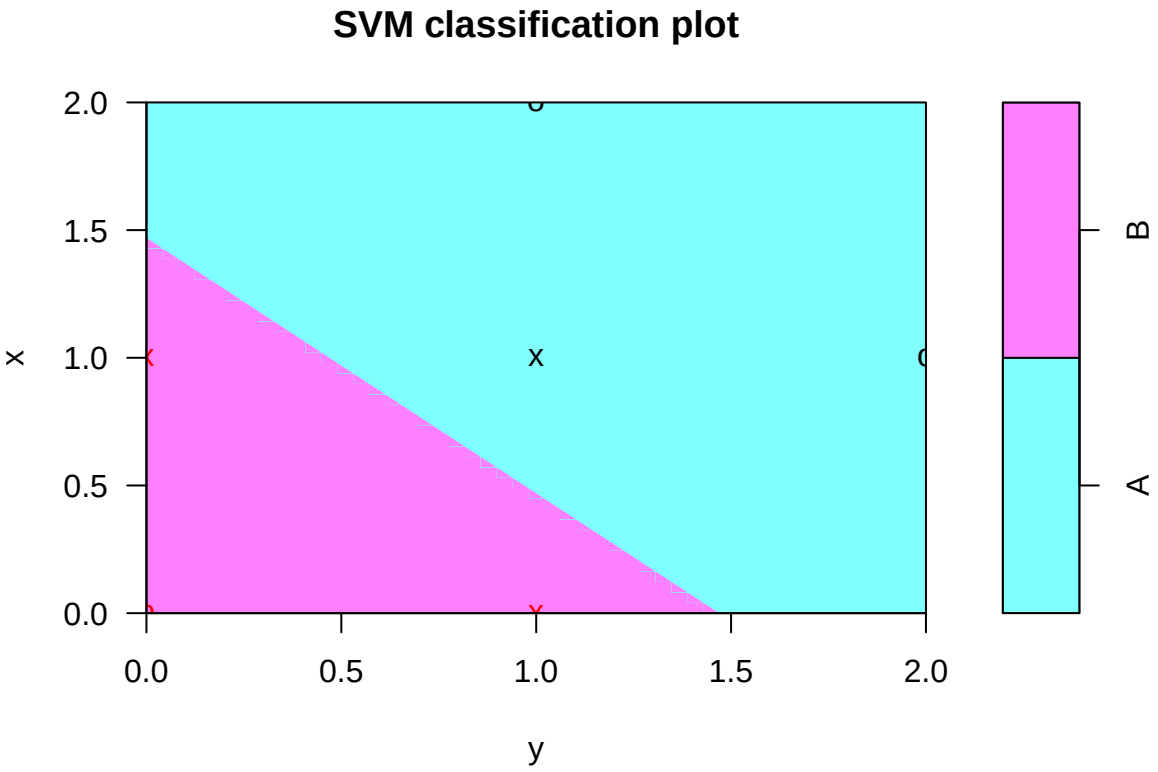
```
library(knitr)
library("e1071")
ABpoints <- read.table("ABpoints.tab", header=T)

model <- e1071::svm(group~., data=ABpoints, type="C-classification",
                    kernel="linear", scale = F, cost=10)
```

The reason we use `scale = F` is because we want to find the suppor vector values without them having been scaled:

```
support.vectors <- model$SV
knitr::kable(model$SV, format="markdown")
```

	x	y.1
1	1	1
5	1	0
6	0	1

```
plot(model, ABpoints)
```



We can calculate the margin to our hyperplane by first calculating the weight vector, **w**

```
w <- t(model$coefs) %*% model$SV;
kable(w, format="markdown")
```

x	y.1
1.999023	1.999023

Knowing the margin is equal to $\frac{2}{||w||}$, we can calculate the margin as

```
2 / norm(w)
```

```
## [1] 1.000489
```

Problem 2

Using the SVM R package "e1071", Apply SVM classification to the AML vs MDS gene expression datasets.

Dataset 207

File "set207-MDS84.tab" contains the expression values of 25 key genes of 84 MDS patients

File "set207-AML93.tab" contains the expression values of 25 key genes of 93 AML patients.

Dataset 228

Files for this datasets contain similar data as indicated by their file names.

In your analysis, use one dataset as the training set, and then apply the trained model to make prediction on the other dataset. Write a brief report on your analysis of the datasets: eg., Explain your choices of parameters and what is the accuracy of your prediction, etc.

Given our datasets, let's see how many patients are in each set. We take away the first two columns which are the `uniqueID` and `tables_count` columns.

```
num.patients <- sapply(X=dir(pattern = "set*"), FUN=function(X) {
  ncol(read.table(X)[,c(-1,-2)])
})
knitr::kable(as.data.frame(num.patients), format="markdown")
```

	num.patients
set207-AML93.tab	93
set207-MDS84.tab	84
set228-AML109.tab	109
set228-MDS80.tab	80

We will use **Dataset 207** as our training data set. In order to use the `e1071::svm` command with our data, we first need to transform our data into a data frame. The data frame will contain a row for each patient with columns for each of the probe IDs and a further column indicating whether the patient is labeled as having *AML* and *MDA*. The same will be done for loading the test data.

Loading Dataset 207 into a dataframe

First, let's load the `set207-AML93` data set into a data.frame

```
set207.AML93 <- read.table(file="set207-AML93.tab", header=T)
knitr::kable(head(set207.AML93)[,1:3], format="markdown")
```

uniqueID	tables_count	GSM376053.CEL.gz.tab_GSM376053.CEL.gz
200998_s_at	93	7.3336326
201362_at	93	6.4068203
203765_at	93	8.0434053
203936_s_at	93	7.9024176
204636_at	93	-0.0516147
205040_at	93	3.8135818

Next, we remove `uniqueID` and `tables_count` columns, but first saving the uniqueIDs (the probeset IDs) in a separate vector.

```
probe.ids <- as.vector(set207.AML93$uniqueID)
set207.AML93 <- set207.AML93[,-c(1,2)]
```

Having done this, we can transpose the table using the `t()` command

```
set207.AML93 <- t(set207.AML93)
```

We then add an extra column to show that each of the patients is labeled as being part of the group "AML" and finally adding the probeset IDs and "group" as column names to the resulting dataframe.

```
set207.AML93 <- data.frame("AML", set207.AML93)
colnames(set207.AML93) <- c("group", probe.ids)

knitr::kable(head(set207.AML93)[,1:4], format="markdown")
```

	group	200998_s_at	201362_at	203765_at
GSM376053.CEL.gz.tab_GSM376053.CEL.gz	AML	7.333633	6.406820	8.043405
GSM376055.CEL.gz.tab_GSM376055.CEL.gz	AML	4.968282	6.523884	7.036591
GSM376056.CEL.gz.tab_GSM376056.CEL.gz	AML	6.655827	6.672962	7.489620
GSM376060.CEL.gz.tab_GSM376060.CEL.gz	AML	5.847922	6.453568	8.315975
GSM376061.CEL.gz.tab_GSM376061.CEL.gz	AML	7.180296	7.217161	8.549044
GSM376063.CEL.gz.tab_GSM376063.CEL.gz	AML	7.031247	6.788623	9.189426

Further to this, we do exactly the same for `set207.MDS84` dataset. (Note we *can* use the `probe.ids` vector from before to generate to name the columns.)

```
set207.MDS84 <- read.table(file="set207-MDS84.tab", header=T)
set207.MDS84 <- set207.MDS84[,-c(1,2)]

set207.MDS84 <- t(set207.MDS84)

set207.MDS84 <- data.frame("MDS", set207.MDS84)
colnames(set207.MDS84) <- c("group", probe.ids)

knitr::kable(head(set207.MDS84)[,1:4], format="markdown")
```

	group	200998_s_at	201362_at	203765_at
GSM376264.CEL.gz.tab_GSM376264.CEL.gz	MDS	9.250412	10.269487	12.14560
GSM376265.CEL.gz.tab_GSM376265.CEL.gz	MDS	9.387417	8.385525	11.19981
GSM376267.CEL.gz.tab_GSM376267.CEL.gz	MDS	8.156499	9.251335	10.50709
GSM376269.CEL.gz.tab_GSM376269.CEL.gz	MDS	9.206459	9.799664	11.41043
GSM376270.CEL.gz.tab_GSM376270.CEL.gz	MDS	9.457402	9.210845	11.21325
GSM376275.CEL.gz.tab_GSM376275.CEL.gz	MDS	7.264435	8.661589	10.77786

All that remains to load *Dataset 207* is to bind the `set207.AML93` dataset with `set207.MDS84`.

```
set207 <- rbind(set207.AML93, set207.MDS84)
```

Loading Dataset 228 into a dataframe

`set207` will be used to train our Support Vector Machine (SVM) such that we can give it unlabeled data points that the machine will classify as either being part of the `AML` or `MDS` group. Before we can do this, we need to use a testing set of data which is also labeled data that we can run through the algorithm and check whethe they have been classified correctly.

set228 will be used as the testing set of data. Let's first load *Dataset 228* into a dataframe like we created above for Dataset 207.

```
set228.AML109 <- read.table(file="set228-AML109.tab", header=T)
set228.AML109 <- set228.AML109[,-c(1,2)]

set228.AML109 <- t(set228.AML109)

set228.AML109 <- data.frame("AML", set228.AML109)
colnames(set228.AML109) <- c("group", probe.ids)

# TODO same for set228.MDS80...
set228.MDS80 <- read.table(file="set228-MDS80.tab", header=T)
set228.MDS80 <- set228.MDS80[,-c(1,2)]

set228.MDS80 <- t(set228.MDS80)

set228.MDS80 <- data.frame("MDS", set228.MDS80)
colnames(set228.MDS80) <- c("group", probe.ids)

set228 <- rbind(set228.AML109, set228.MDS80)
```

Training the SVM

As aformentioned we will use set207 as our training set and set228 as our test data set.

```
training.set <- set207
test.set <- set228
```

Here, we train our svm with the training set, assuming group is our dependent variable and all other variables are explanatory variables.

Next we train our SVM using our training set of data:

```
svm.model <- svm(group ~ ., data = training.set, kernel="sigmoid")
```

And then, using our test data, we predict whether our test data belongs to either the AML or MDS class. (Note that we exclude column 1 of the test.set which is the column that labels whether these records are AML or MDS)

```
set207.pred <- predict(svm.model, test.set[,-1])
```

We can construct a table to demonstrate the performance of our SVM showing the counts of true predictions against false predicitons of the classification of our data:

```
prediction.tab <- table(pred = set207.pred, true = test.set[,1])
prediction.tab
```

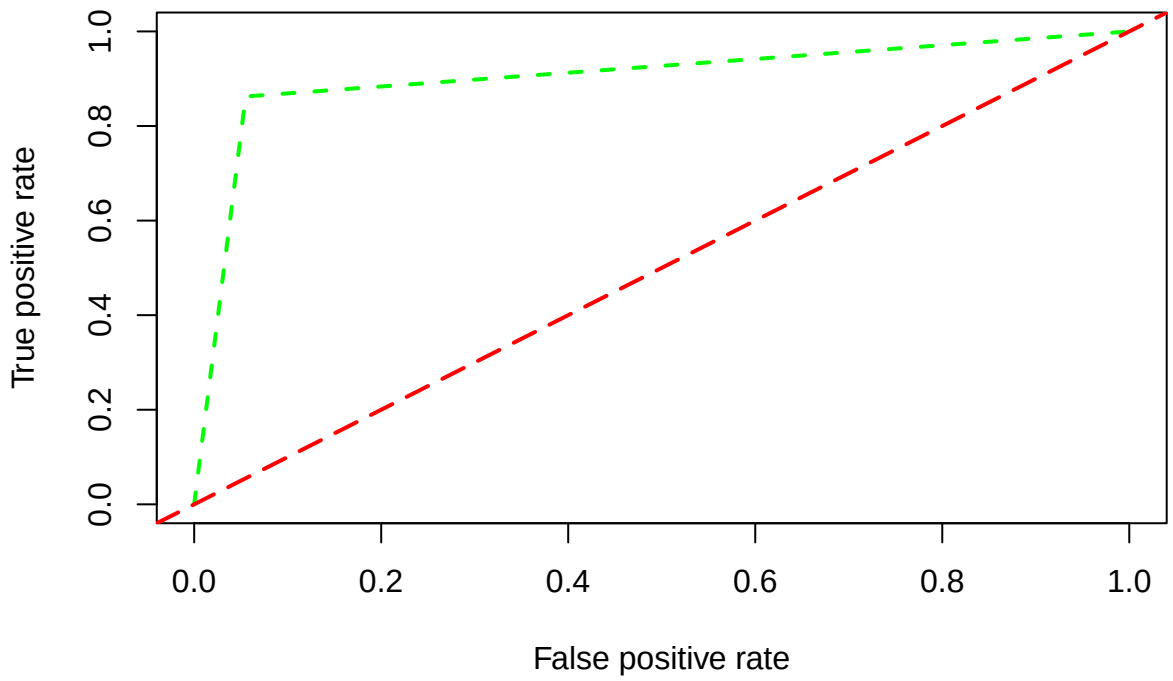
```
##      true
## pred  AML MDS
##   AML 103  11
##   MDS   6  69
```

We can observe from the table that the SVM we created performed quite well. We can visualise these results with the R package `ROCR` by drawing a ROC (Receiver Operating Characteristics) curve.

```
library(ROCR)
```

```
## Loading required package: gplots
##
## Attaching package: 'gplots'
##
## The following object is masked from 'package:stats':
##
##     lowess
```

```
pred <- ROCR::prediction(list(as.numeric(set207.pred)), labels =  list(as.numeric(test.set[,1])))
performance <- performance(pred, measure="tpr", x.measure="fpr")
plot(performance, col="green", lty=2, lwd=2)
abline(0,1, col = "red", lty=5, lwd=2)
```



The green line shows how much better the SVM is at determining the class of a case (green dotted line) as opposed to a random guess (shown by the red dotted line).

Tuning the SVM

We can look at tuning the SVM using *k-fold cross validation* which is in-built into the `e1071` implementation. K-fold cross validation involves partitioning the training data into k equally sized subsamples. The method will compute values for γ (gamma) and C (cost) and ultimately determine a best fit model for us.

```
tuned.svm <- tune.svm(group ~ ., data = training.set, sampling="cross", cross=50, kernel="sigmoid")
gamma <- tuned.svm$best.model$gamma
cost <- tuned.svm$best.model$cost
```

```
gamma
```

```
## [1] 0.04
```

```
cost
```

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

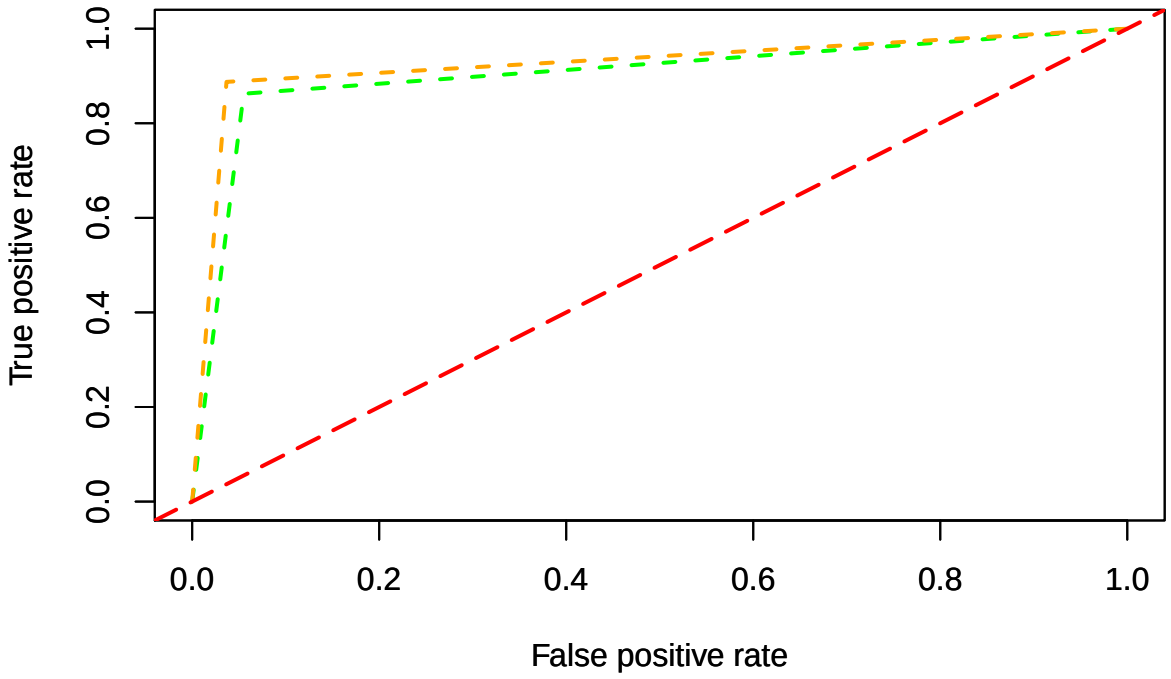
Due to the k-fold cross validation then, the values converged upon for gamma and cost were 0.04 and 1 respectively. Using these values to generate another SVM we can genrate another prediction table to compare its performance to our previous SVM.

```
best.svm <- svm(group ~ ., data = training.set, cost = cost, gamma = gamma)
best.pred <- predict(best.svm, test.set[,-1])
best.prediction.tab <- table(pred = best.pred, true = test.set[,1])
best.prediction.tab
```

```
##      true
## pred  AML MDS
##   AML 105   9
##   MDS   4  71
```

The tuned SVM using k-fold cross validation performs marginally better than the default. We can also compare the performance of our k-fold cross validated svm by plotting it against our original SVM on a ROC graph.

```
pred.best <- ROCR::prediction(list(as.numeric(best.pred)), labels =  list(as.numeric(test.set[,1])))
performance.best <- performance(pred.best, measure="tpr", x.measure="fpr")
plot(performance, col="green", lty=2, lwd=2)
par(new=T)
plot(performance.best, col="orange", lty=2, lwd=2)

abline(0,1, col = "red", lty=5, lwd=2)
```



Running the code in this document

This document was compiled using the `knitr` extension to R. Knitr is an implementation of the *Literate Programming* paradigm conceived of by Donald Knuth. This allows an author to *weave* code and text together into one document.

To run the code from this document, follow these instructions.

1. Open the zipped directory containing this pdf
2. (Alternatively, download the whole project from http://github.com/hiraethus/svm-assignment)
3. Open RStudio and load this project by choosing `File > Open Project` and navigating to the file `svm_assignment_report.Rproj`.
4. From the file browser in RStudio, choose `svm_assignment_report.Rmd`
5. Use the `chunks` menu located on the text editor opened in R to run each of the chunks in the file
6. Run all the code chunks in RStudio by using typing `Ctrl + Alt + R`.