SLG - SVM in R

Todd Wilson

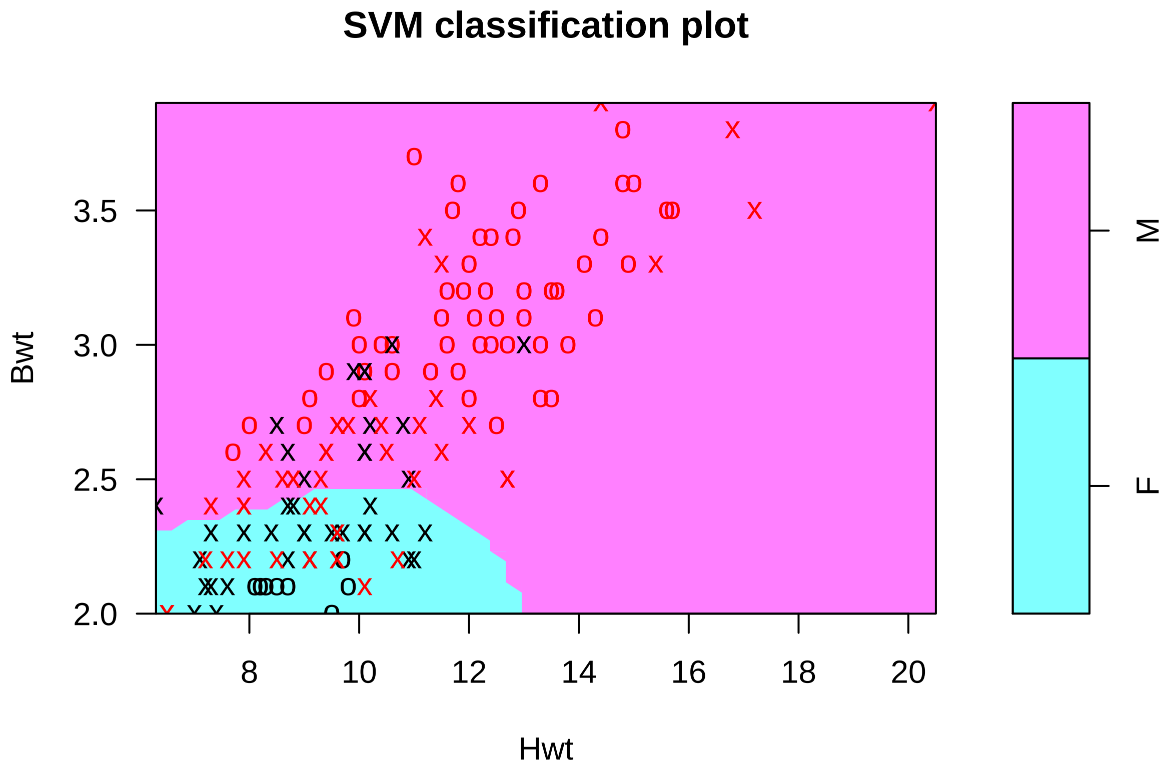
September 27, 2016

This document follows SVM code using the `e1071` package in R. The first block is open source code available in the Wikibooks article on data mining algorithms in R. The second block was written by Melvin L., an R blogger and machine learning enthusiast.

```
library(MASS)
library(e1071)

########################################################
###           SVM using cats data set in R           ###
###      Open source code available on Wikibooks     ###
########################################################

#load data set
data(cats)

#build SVM
model       <- svm(Sex~., data = cats)

#view results
summary(model)
```

```
##
## Call:
## svm(formula = Sex ~ ., data = cats)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.5
##
## Number of Support Vectors:  84
##
##  ( 39 45 )
##
##
## Number of Classes:  2
##
## Levels:
##  F M
```

```
plot(model, cats)
```



```
#divide data into training and test sets
index        <- 1:nrow(cats)
testindex    <- sample(index, trunc(length(index)/3))
testset      <- cats[testindex,]
trainset     <- cats[-testindex,]

#build SVM with training data
model        <- svm(Sex~., data = trainset)
prediction   <- predict(model, testset[,-1])

#confusion matrix
table(pred = prediction, true = testset[,1])
```

```
##     true
## pred  F  M
##    F 12  5
##    M  5 26
```

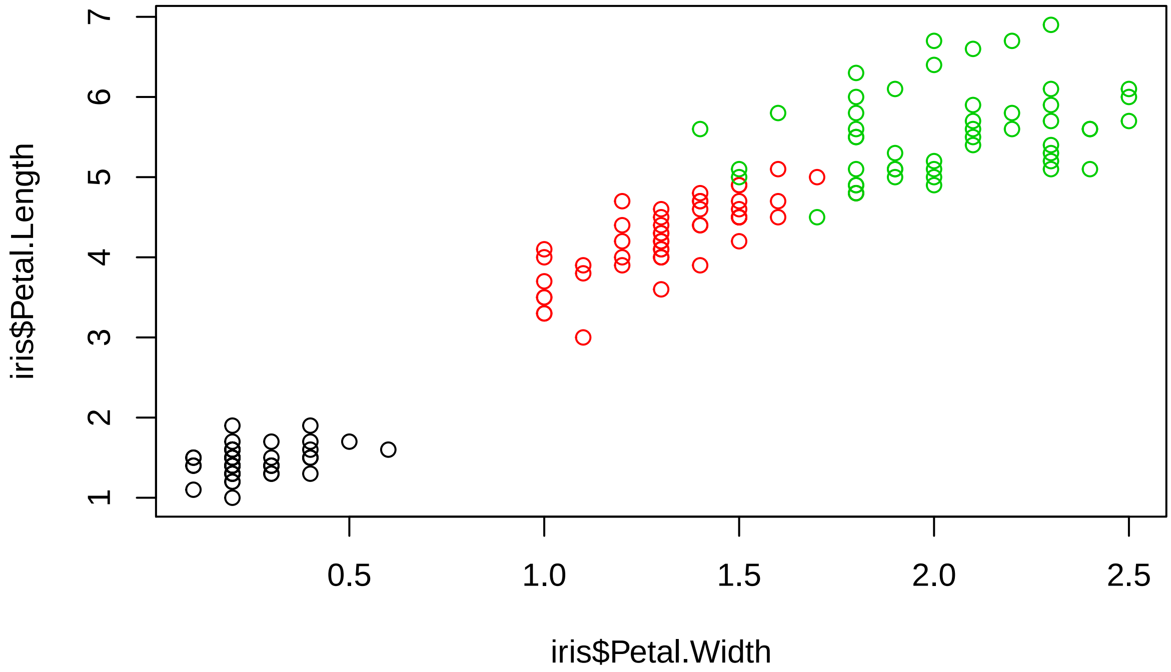
```
#tune model parameters
tuned        <- tune.svm(Sex~., data = trainset, gamma = 10^(-6:-1), cost = 10^(1:2))
summary(tuned)
```

```
##
## Parameter tuning of 'svm':
##
```

```
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##    0.1  100
##
## - best performance: 0.2088889
##
## - Detailed performance results:
##    gamma cost     error dispersion
## 1  1e-06   10 0.3133333 0.13196832
## 2  1e-05   10 0.3133333 0.13196832
## 3  1e-04   10 0.3133333 0.13196832
## 4  1e-03   10 0.3133333 0.13196832
## 5  1e-02   10 0.2200000 0.07835763
## 6  1e-01   10 0.2188889 0.08718270
## 7  1e-06  100 0.3133333 0.13196832
## 8  1e-05  100 0.3133333 0.13196832
## 9  1e-04  100 0.3133333 0.13196832
## 10 1e-03  100 0.2088889 0.08555475
## 11 1e-02  100 0.2200000 0.07835763
## 12 1e-01  100 0.2088889 0.08245213
```

We notice that the radial kernel has been selected because the data are no linearly separable. Also, when tuning the model, our initial values for the cost and gamma parameters (the latter is necessary for a non-linear kernel) are determined to be different than their optimal values.

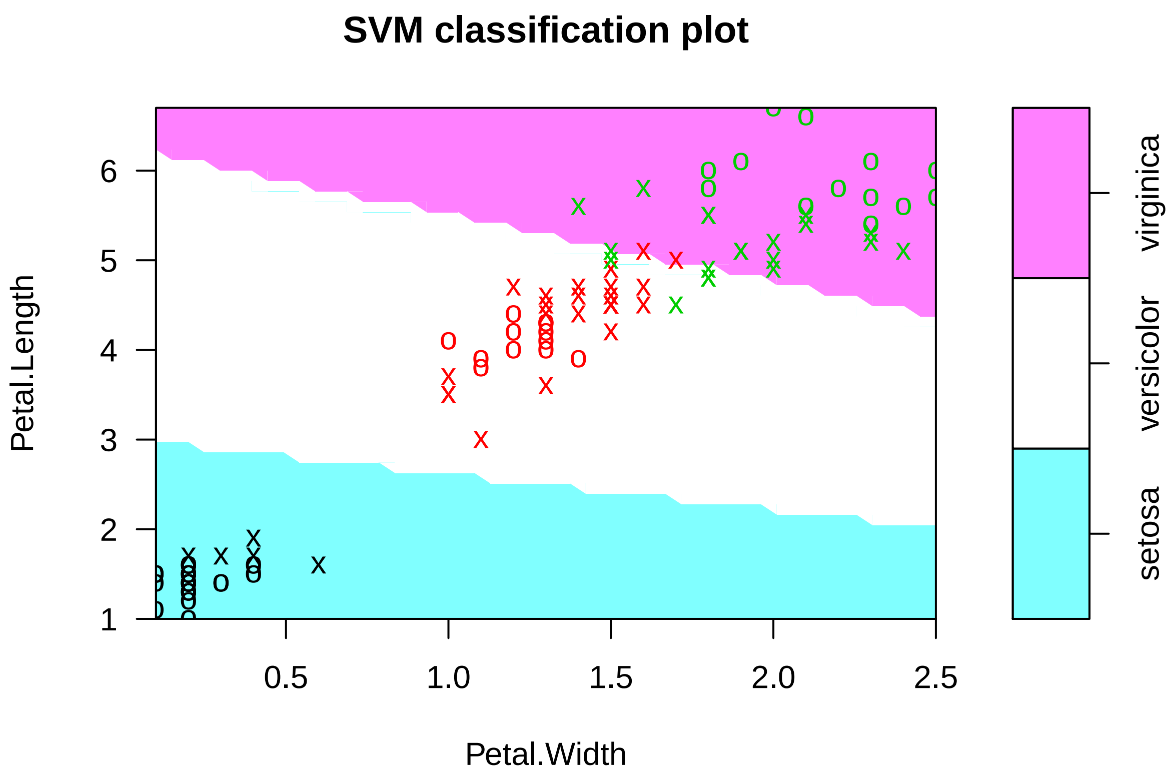
```
########################################################
###           SVM using iris data set in R           ###
###             Code written by Melvin L             ###
########################################################

#unclassified, colored by species
plot(iris$Petal.Width, iris$Petal.Length, col = iris$Species)
```



```
#split data into train and test sets
s           <- sample(150,100)
col         <- c("Petal.Length", "Petal.Width", "Species")
iris_train  <- iris[s, col]
iris_test   <- iris[-s, col]

#fit svm
svmfit      <- svm(Species ~ ., data = iris_train, kernel = "linear", cost = 0.1, scale = F)

#see svm output
print(svmfit)
```

```
##
## Call:
## svm(formula = Species ~ ., data = iris_train, kernel = "linear",
##     cost = 0.1, scale = F)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.5
##
## Number of Support Vectors:  50
```

```r
#visualize svm results
plot(svmfit, iris_train[,col])
```



```r
#identify optimal cost parameter using cross validation
tuned       <- tune(svm, Species ~ ., data = iris_train, kernel = "linear", ranges = list(cost = c(0.00
summary(tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.06
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.73 0.11595018
## 2 1e-02  0.43 0.17029386
## 3 1e-01  0.07 0.06749486
## 4 1e+00  0.06 0.06992059
## 5 1e+01  0.06 0.06992059
## 6 1e+02  0.06 0.06992059
```

```r
#predictions for the test data
p           <- predict(svmfit, iris_test[,col], type = "class")

#test model performance
table(p, iris_test[,3])
```

```
##
## p             setosa versicolor virginica
##   setosa          20          0         0
##   versicolor       0         13         0
##   virginica        0          0        17
```

```r
mean(p == iris_test[,3])
```

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

In multiple runs of the code, we see that 45-50 support vectors are used, and the test model predicts correctly between 94% and 100% of the time.