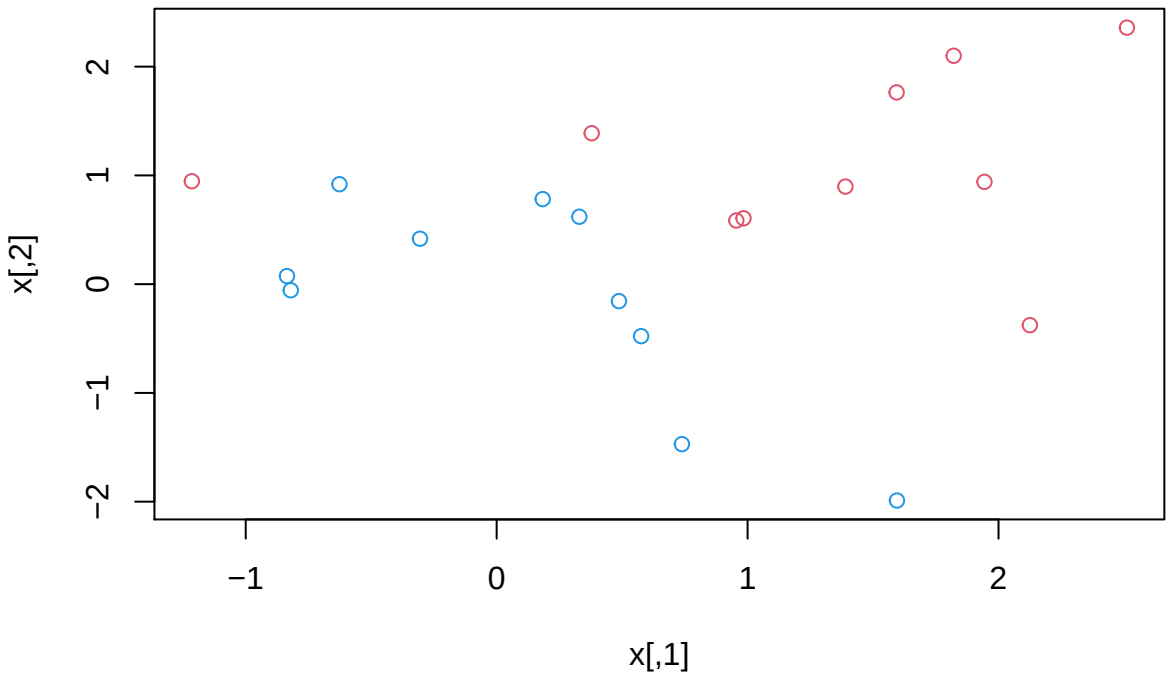
SVM example 3 (ISLR)

Abhirup Sen

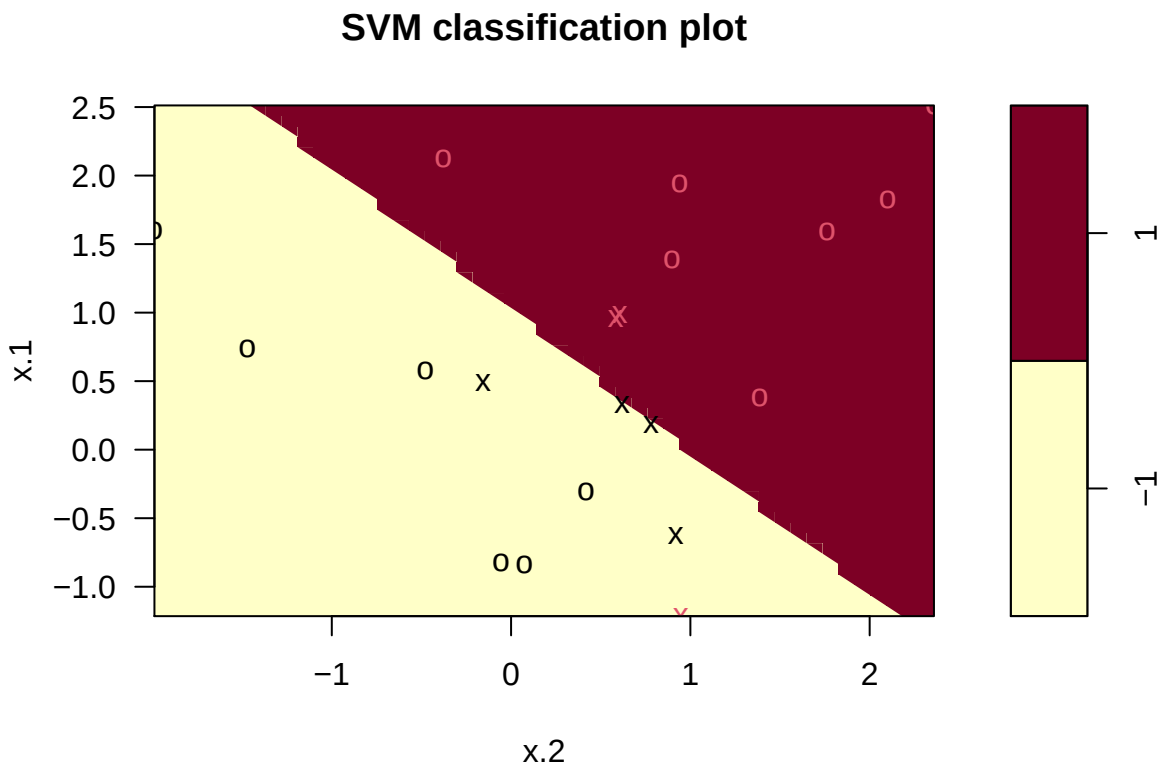
24/05/2021

```
set.seed(1)
x= matrix(rnorm(20*2),ncol=2)
y = c(rep(-1,10),rep(1,10))
x[y==1,]=x[y==1,]+1
plot(x,col=(3-y))
```



Now lets fit the Support vector classifier.

```
data = data.frame(x=x, y = as.factor(y))
library(e1071)
svmfit = svm(y ~., data = data, kernel ="linear", cost = 10, scale =FALSE)
plot(svmfit,data)
```




```
svmfit$index
```

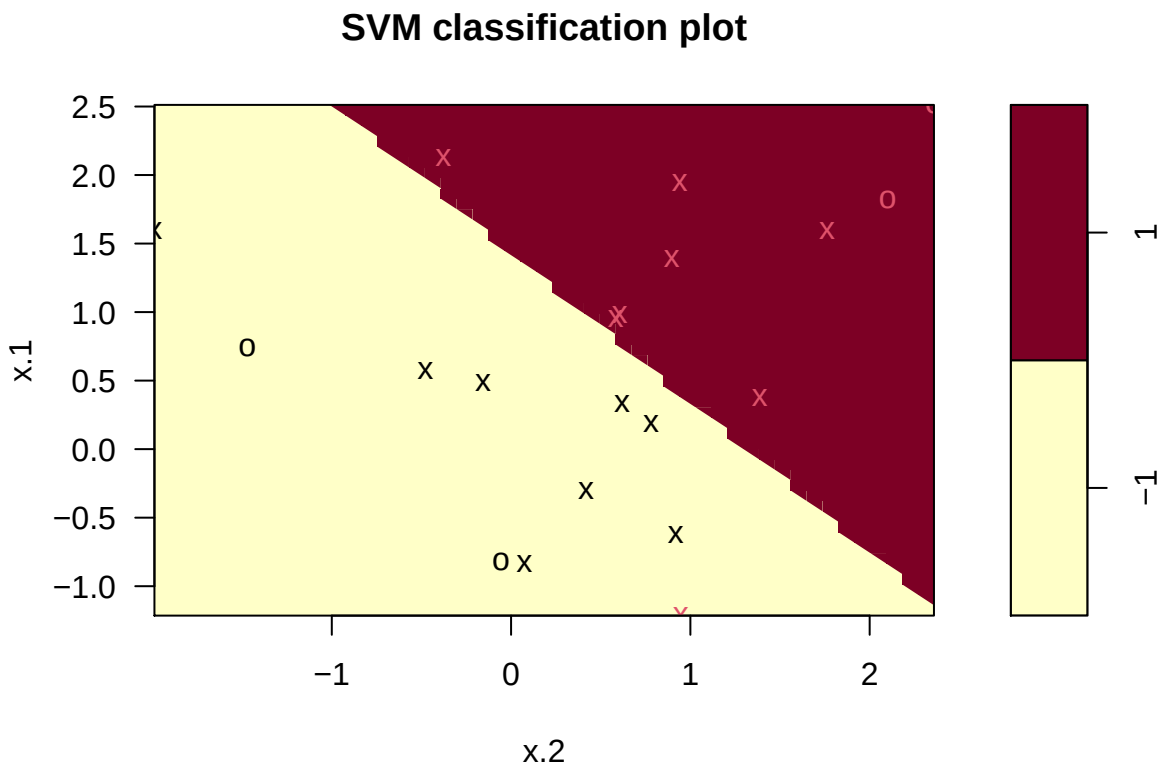
```
## [1]  1  2  5  7 14 16 17
```

```
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = data, kernel = "linear", cost = 10, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  7
##
##  ( 4 3 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

What if we use a smaller value of cost

```
svmfit = svm(y~., data = data, kernel ="linear",cost = 0.1, scale = FALSE)
plot(svmfit, data)
```




```
svmfit$index
```

```
##  [1]  1  2  3  4  5  7  9 10 12 13 14 15 16 17 18 20
```

The tune() function aspart of library e1071 ; performs crossValidation, 10 fold cross validation.

```
set.seed(1)
tune.out = tune(svm, y~., data = data, kernel ="linear",
                ranges = list(cost = c(0.001, 0.01, 0.1, 1.5, 10, 100)))
```

```
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
```

```
##   0.1
##
## - best performance: 0.05
##
## - Detailed performance results:
##      cost error dispersion
## 1   0.001  0.55  0.4377975
## 2   0.010  0.55  0.4377975
## 3   0.100  0.05  0.1581139
## 4   1.500  0.15  0.2415229
## 5  10.000  0.15  0.2415229
## 6 100.000  0.15  0.2415229
```

```
bestmod = tune.out$best.model
summary(bestmod)
```

```
##
## 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:  0.1
##
## Number of Support Vectors:  16
##
##  ( 8 8 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

```
xtest = matrix(rnorm(20*2), ncol=2)
ytest = sample(c(-1,1),20,rep = TRUE)
xtest[ytest==1,]==xtest[ytest==1,]+1
```

```
##        [,1]  [,2]
##  [1,] FALSE FALSE
##  [2,] FALSE FALSE
##  [3,] FALSE FALSE
##  [4,] FALSE FALSE
##  [5,] FALSE FALSE
##  [6,] FALSE FALSE
##  [7,] FALSE FALSE
##  [8,] FALSE FALSE
##  [9,] FALSE FALSE
```

```
testdata = data.frame(x=xtest,y=as.factor(ytest))
```

```
ypred = predict(bestmod,testdata)
table(predict = ypred, truth = testdata$y)
```

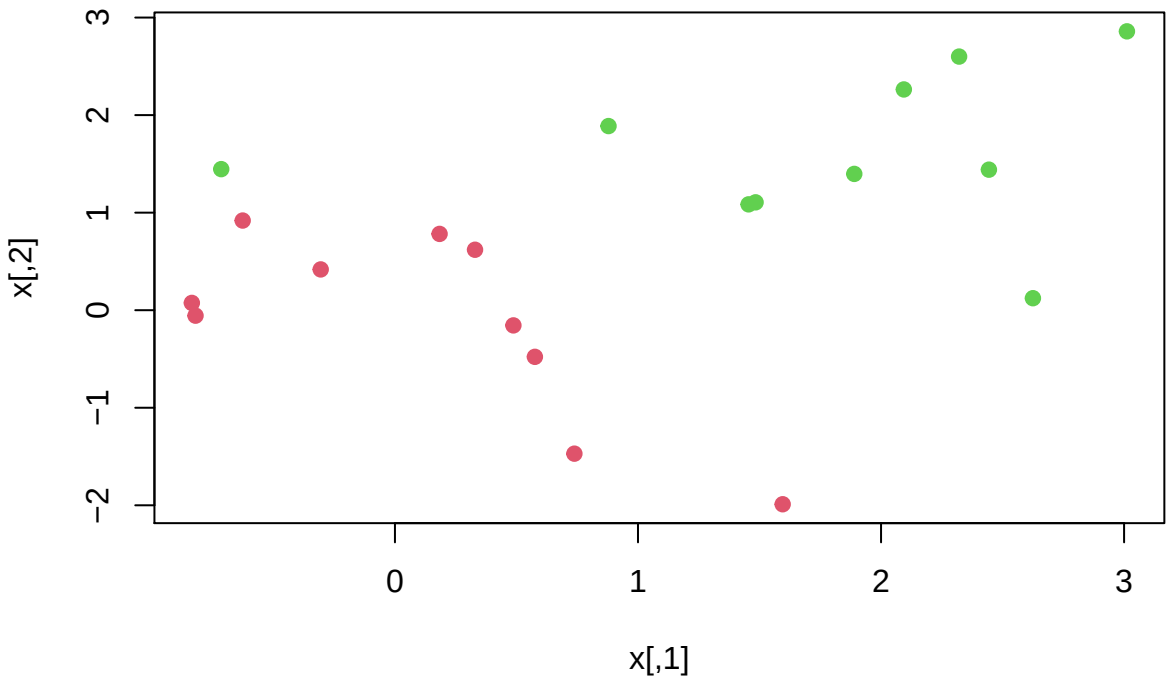
```
##         truth
## predict -1 1
##      -1  9 9
##      1   2 0
```

```
svmfit = svm(y~., data = data, kernel ="linear", cost = 0.01, scale =FALSE)
ypred = predict(svmfit, testdata)
table(predict = ypred,truth = testdata$y)
```

```
##         truth
## predict -1  1
##      -1 11  9
##      1   0  0
```

Now if the 2 classes are linearly seperable

```
x[y==1,]=x[y==1,]+0.5
plot(x,col = (y+5)/2, pch = 19)
```



```r
data = data.frame(x=x,y=as.factor(y))
svmfit = svm(y~., data = data, kernel = "linear", cost = 1e+05)
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = data, kernel = "linear", cost = 1e+05)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1e+05
##
## Number of Support Vectors:  3
##
##  ( 1 2 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```