Support Vector Machines

by

AINDRILA GARAI

MSC STATISTICS, IIT KANPUR

aindrilag22@iitk.ac.in

Maximal Margin Classifier:

Hyperplane:

In a p-dimensional space, a hyperplane is a flat affine subspace of hyperplane dimension $p-1$. The equation of a p-dimensional hyperplane is-

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

a point $X = \left(X_1, X_2, \ldots, X_p\right)^T$ in p-dimensional space satisfies the above equation. If the equation is greater than or less than 0, it means the point lies outside the hyperplane.

Classification Using a Separating Hyperplane:

Let x_{ij} is training observations and $y_i \in -1,1$'s are outputs, then a separating hyperplane has the property that

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0 \text{ if } y_i = 1$$

and

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} < 0 \text{ if } y_i = -1$$

The Maximal Margin Classifier:

- A natural choice is the **maximal margin hyperplane** among so many hyperplanes (also known as the maximal optimal separating hyperplane), which is the separating hyperplane that is **farthest** from the training observations.
- We can compute the perpendicular distance from each training observation to a given separating hyperplane; the smallest distance is known as the **margin**.'
- The maximal margin hyperplane is the separating hyperplane that has the farthest minimum distance to the training observations.
- We can then classify a test observation based on which side of the maximal margin hyperplane it lies. This is known as the **maximal margin classifier**.

- It can also lead to overfitting when p is large.
- The observations which "support" the maximal margin hyperplane in the sense that if these points were moved slightly then the maximal margin hyperplane would move as well is known as **support vectors**. The maximal margin hyperplane depends directly on the support vectors but not on the other observations.

Construction of the Maximal Margin Classifier:

To construct the maximal margin hyperplane based on a set of n training observations $x_1, \dots, x_n \in \mathbb{R}_p$ and associated class labels $y_1, \dots, y_n \in$ {\$−\$1, 1}. The maximal margin hyperplane is the solution to the optimization problem

$$\underset{\beta_0,\beta_1,\dots,\beta_p,M}{\text{maximize}} M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n$$

Support Vector Classifiers:

If we cannot exactly separate the two classes, a hyperplane that almost separates the classes is developed using a so-called **soft margin**. The generalization of the maximal margin classifier to the non-separable case is known as the **support vector classifier**.

- we allow some observations to be on the incorrect side of the margin or the hyperplane. (The margin is soft because it can be violated by some of the training observations.)
- To calculate the hyperplane, we solve the given optimization problem-

$$\underset{\beta_0,\beta_1,\dots,\beta_p,\epsilon_1,\dots,\epsilon_n,M}{\text{maximize}} M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

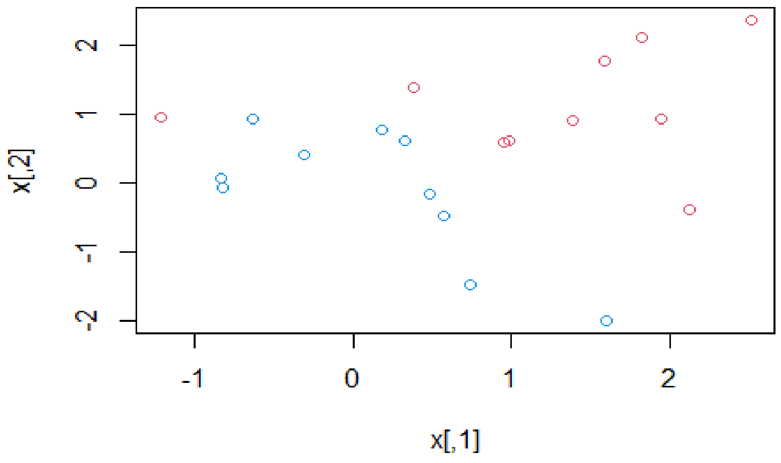
$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

where C is a nonnegative tuning parameter. M is the width of the margin; we seek to make this quantity as large as possible. $\epsilon_1, \dots, \epsilon_n$ are slack variables that allow individual observations to be on the wrong side of the margin or the hyperplane.

- If $\epsilon_i = 0$ then the ith observation is on the correct side of the margin. If $\epsilon_i > 0$ then the ith observation is on the wrong side of the margin.
- If $\epsilon_i > 0$ then it is on the wrong side of the hyperplane.

- C determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate.
- When C is small, we seek narrow margins that are rarely violated; this amounts to a classifier that is highly fit to the data, which may have low bias but high variance.
- When C is larger, the margin is wider and we allow more violations to it; this amounts to fitting the data less hard and obtaining a classifier that is potentially more biased but may have lower variance.
- Same as we discussed earlier, observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors. These observations do affect the support vector classifier.

```
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))
```

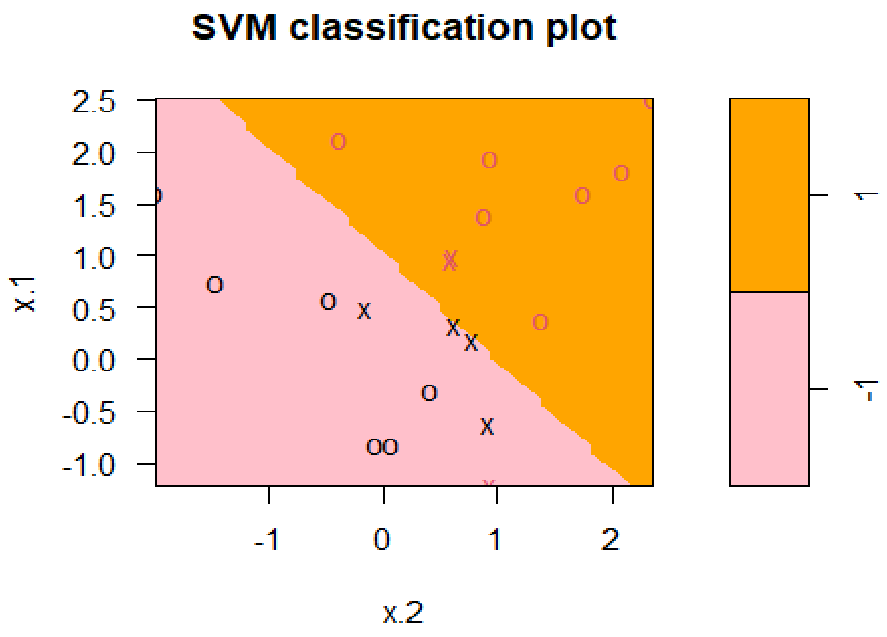


```
dat <- data.frame (x = x, y = as.factor(y))
library (e1071)
svmfit <- svm ( y ~ ., data = dat , kernel = "linear", cost = 5, scale = FALS
E) # to fit the support vector classifier
summary (svmfit)

##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "linear", cost = 5, scale = FALS
E)
##
##
```

```
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  5
##
## Number of Support Vectors:  7
##
##  ( 4 3 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1

plot (svmfit , dat , col= c("pink" , "orange"))
```



```
set.seed (1)
tune.out <- tune (svm , y ~ ., data = dat , kernel = "linear", ranges = list
(cost = c (0.001, 0.01, 0.1, 1, 5, 10, 100))) # performs ten-fold cross-valid
ation
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 1e-03  0.55  0.4377975
## 2 1e-02  0.55  0.4377975
## 3 1e-01  0.05  0.1581139
## 4 1e+00  0.15  0.2415229
## 5 5e+00  0.15  0.2415229
## 6 1e+01  0.15  0.2415229
## 7 1e+02  0.15  0.2415229
```

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

```
##
## Call:
## best.tune(METHOD = svm, train.x = y ~ ., data = dat, 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
testdat <- data.frame (x = xtest , y = as.factor (ytest))
ypred <- predict (bestmod , testdat)
table (predict = ypred , truth = testdat$y)
```

```
##        truth
## predict -1 1
##      -1  9 1
##      1   2 8
```

Support Vector Machines:

For non-linear decision boundaries we fit a support vector classifier using $2p$ features

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$$

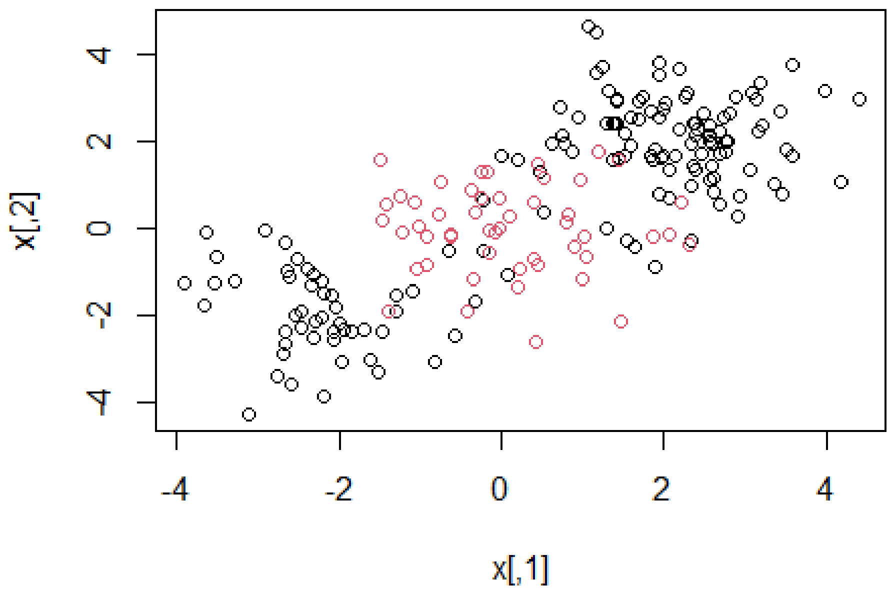
and solve the optimization problem:

$$\underset{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} M$$

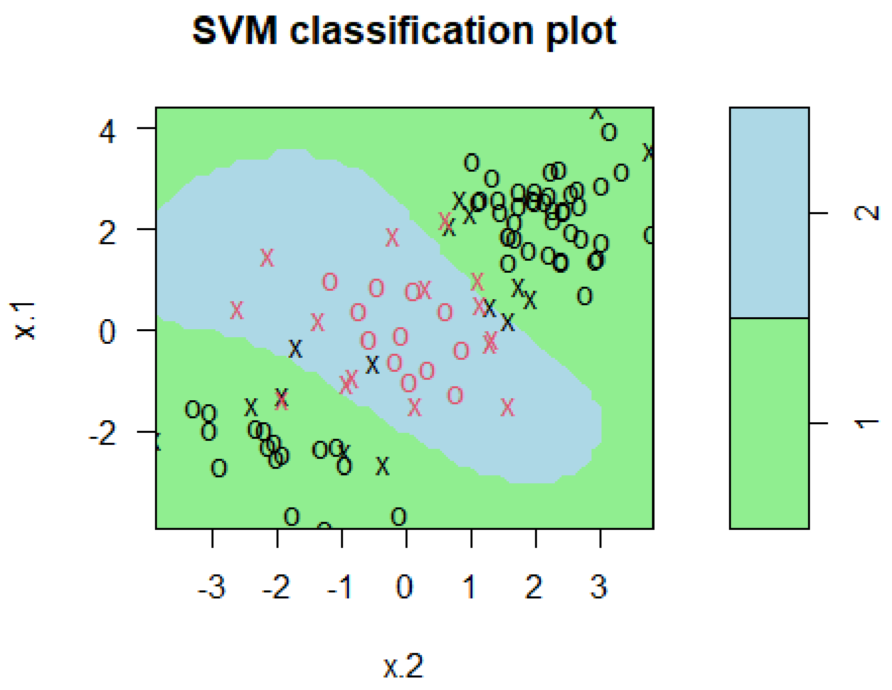
$$\text{subject to } y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_{j1}\,x_{ij} + \sum_{j=1}^{p}\beta_{j2}\,x_{ij}^2\right) \geq M(1-\epsilon_i),$$

$$\sum_{i=1}^{n}\epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1.$$

```
set.seed (1)
x <- matrix ( rnorm (200 * 2), ncol = 2)
x[1:100, ] <- x[1:100, ] + 2
x[101:150, ] <- x[101:150, ] - 2
y <- c( rep (1, 150), rep (2, 50))
dat <- data.frame (x = x, y = as.factor (y))
plot (x, col = y)
```



```
train <- sample (200, 100)
svmfit <- svm ( y ~ ., data = dat[train , ], kernel = "radial", gamma = 1, co
st = 1)
plot (svmfit , dat[train , ], col= c("lightgreen","lightblue"))
```



```
summary (svmfit)

##
## Call:
## svm(formula = y ~ ., data = dat[train, ], kernel = "radial", gamma = 1,
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  31
##
##  ( 16 15 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

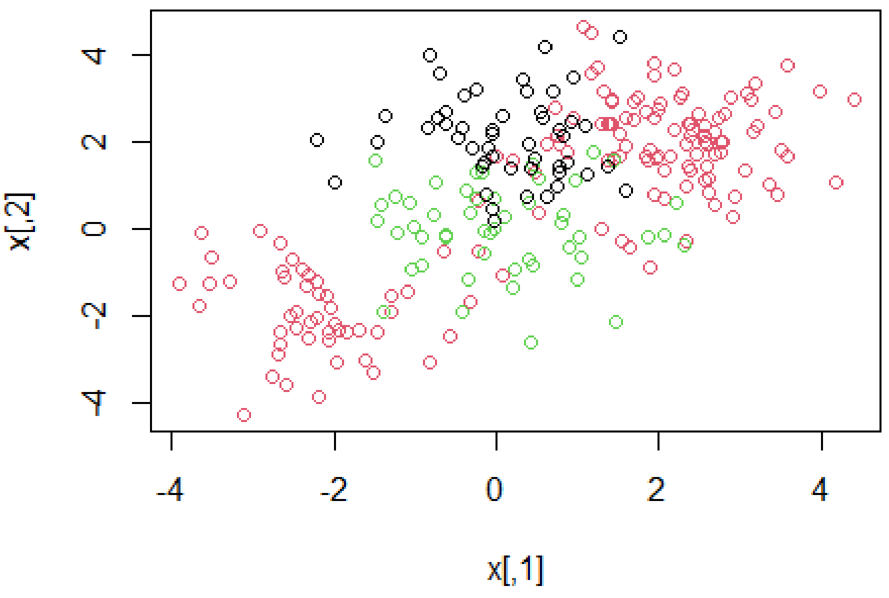
One-Versus-One Classification:

A one-versus-one or all-pairs approach constructs $\binom{K}{2}$ one-versusSVMs, each of which compares a pair of classes.

One-Versus-All Classification:

The one-versus-all approach is an alternative procedure for applying SVMs in the case of $K > 2$ classes. We fit K SVMs, each time comparing one of the K classes to the remaining $K - 1$ classes.

```
set.seed (1)
x <- rbind (x, matrix ( rnorm (50 * 2), ncol = 2))
y <- c(y, rep (0, 50))
x[y == 0, 2] <- x[y == 0, 2] + 2
dat <- data.frame (x = x, y = as.factor (y))
par (mfrow = c(1, 1))
plot (x, col = (y + 1))
```

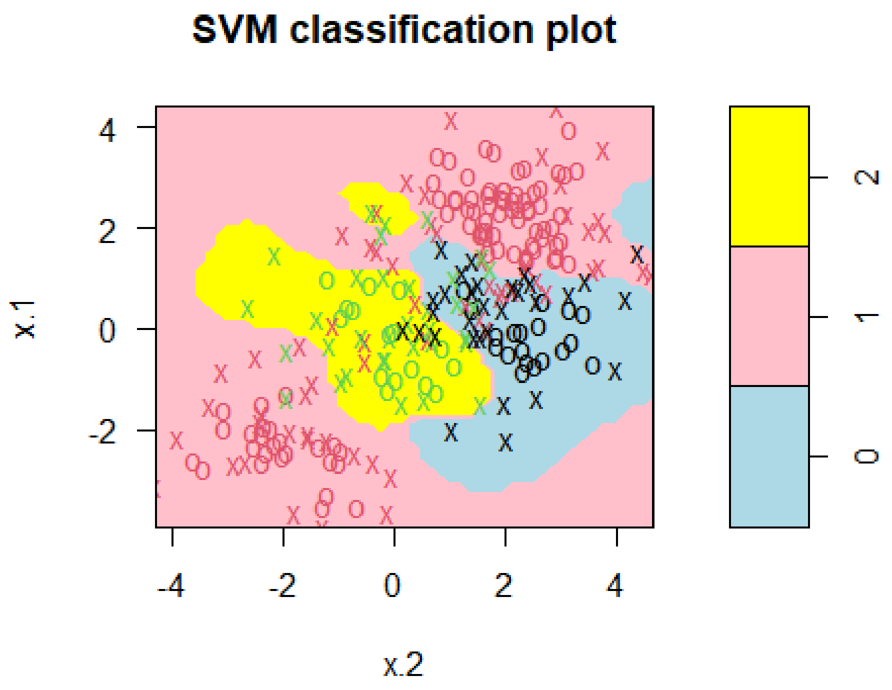


```
svmfit <- svm ( y ~ ., data = dat , kernel = "radial", cost = 10 , gamma = 3)
svmfit

##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "radial", cost = 10, gamma = 3)
##
##
```

```
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  126

plot (svmfit , dat , col = c("lightblue" , "pink" , "yellow"))
```


SVM classification plot
x.1
x.2