Support Vector Machines

Olena Smotrova

13/03/2018

Contents

Perceptron

p-dimensional hyperplane in a p-dimensional space is defined by

$$\beta_0 + \beta_1 x_1 + ... + \beta_p x_p = 0 \tag{1}$$

Suppose, we have n training observations in p-dimensional space that fall into two classes. $y_1, ..., y_n \in -1, 1$, where -1 represents one class and 1 the other class.

Our goal is to develop a classifier based on the training data that will correctly classify test observations. Separating hyperplane has the property that

$$y_i(\beta_0 + \beta_1 x_1 + ... + \beta_p x_p) > 0, \forall i = 1, ..., n \tag{2}$$

If a separating hyperplane exist, a test observations are assigned a class depending on which side of hyperplane it is located. A simple linear algorithm (Perceptron algorithm) is following

- Update β, β_0 based on just one data point (x,y) at a time
- Initial guess $\beta_0, \beta = 0$
- If $\beta \cdot x > 0$ no update
- If $\beta \cdot x \leq 0$ (point is missclassified):
  $\beta = \beta + yx$, $\beta_0 = \beta_0 + y$

Maximal margin classifier

In general, if our data can be perfectly separated using a hyperplane, then there will in fact exist an infinite number of such hyperplanes. In order to construct a classifier based on a separating hyperplane, we need to choose one. A natural choice is the *maximal margin hyperplane*, which is the separating htperplane that is farthest from the training observations. We compute the distance from each training observation to a given separating hyperplane. The minimal distance from observations to hyperplane is the *margin*. The maximal margin hyperplane is a separating hyperplane for which the margin is largest. In Fig.2 (right) the maximal margin hyperplane and the margin are shown. Two red points and one blue point on the dashed lines are the *support vectors*. The "support" the maximal margin hyperplane in the sense that if these points were moved slightly then the maximal margin hypeplane would move as well. The maximal margin hyperplane depends directly on only small subset of the observations (support vectors) is an important property. The margin lines are defined by $\beta_0 + \beta \cdot x = \pm 1$, and the margin equals $1/||\beta||$.

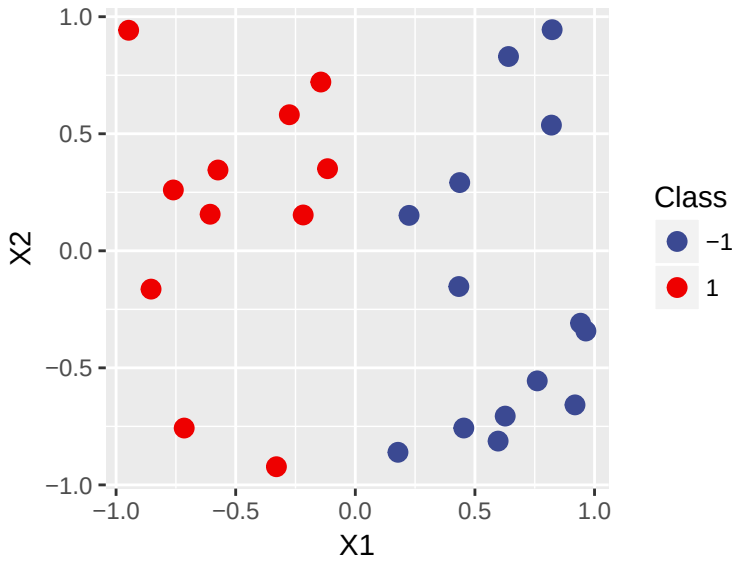

Figure 1: There are two classes of observations, shown in red and blue.

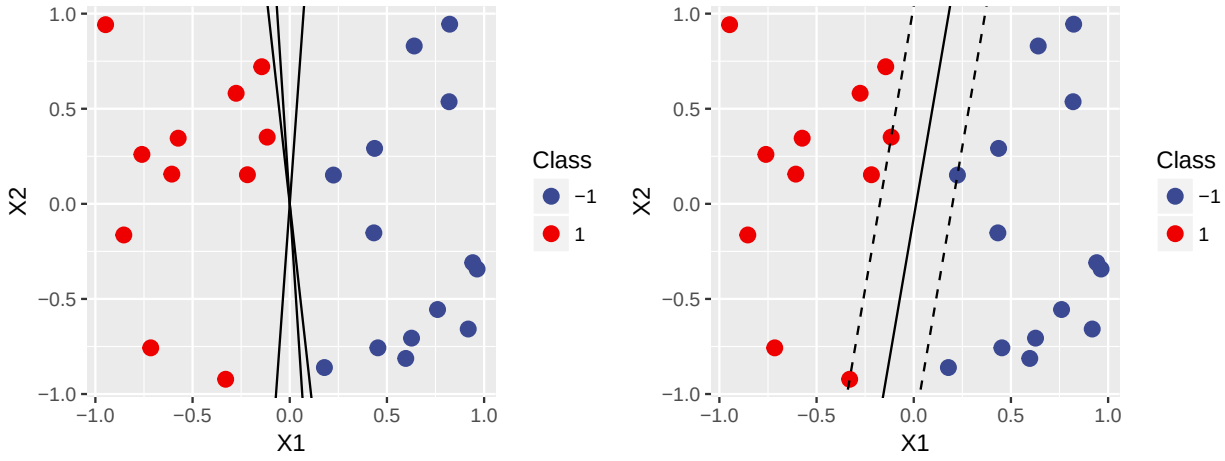

Figure 2: Left: Black lines are three (of many) separating hyperplane given by perceptron algorithm. Right: Maximal margin hyperplane is shown in black solid line. The margin is the distance from solid line to either of the dashed lines.

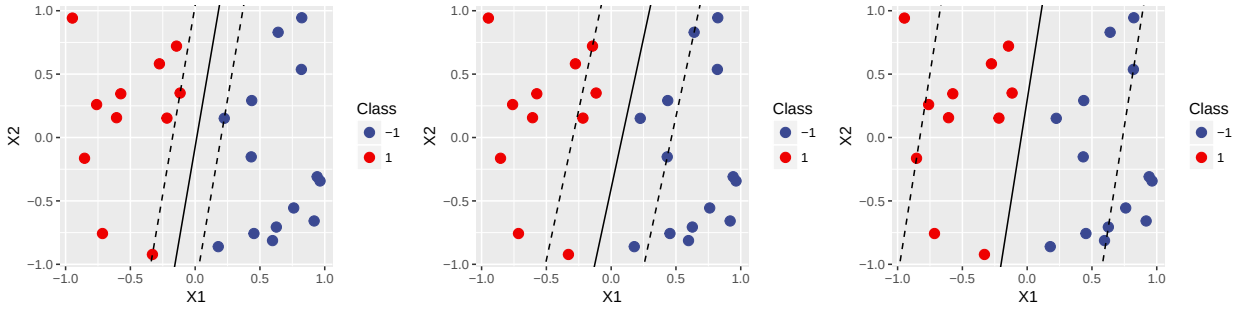

Figure 3: Black lines are separating hyperplanes. Dashed lines are the margin lines. Left: 3 support vectors. Middle: 9 support vectors Right: 20 support vectors.

Support vector classifier

The maximal margin classifier is a very natural way to perform classification, if a separating hyperplane exists. The generalization of the maximal margin classifier to the non-separable case is known as the *support vector classifier*.Even if a separating hyperplane exists, a classifier based on a separating hyperplane will necessarily perfectly classify all of the training observations but can caused sensitivity of individual observations. In this case we can consider a classifier based on a hyperplane that does not perfectly separate two classes. This lead to greater robustness to individual observations and better classifications of most of the training classification. The support vector classifier is also called *soft margin classifier*. The hyperplane is chosen to correctly separate most of training observations into two classes, but may misclassify a few observations. The observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as *support vectors*.

The support vector classifier is demonstrated using `svm()` function from `e1071` library.

```
svm(y ~ ., data, kernel='linear', cost, scale = FALSE)
```

A very large value of `cost` corresponds to the case that no observations are misclassified (maximal margin classifier). The small value of cost enables one to build the `soft` margin classifier.

Support vector machines (non-linear decision boundaries)

The *support vector machine* is an extension of the support vector classifier using kernels. This enables one to deal with non-linear class boundaries. The support vector machine can be represent by a function in a form

$$f(x) = \beta_0 + \sum_{s \in S} \alpha_i K(x, x_i) \tag{3}$$

where K is some function that we refer to as a *kernel*. S is a set of indices of the support vectors. We could take different kernels

- linear

$$K(x, x_i) = x \cdot x_i = \sum_{j=1}^{p} x_j x_{ij} \tag{4}$$

- polynomial of degree d

$$K(x, x_i) = (1 + x \cdot x_i)^d = (1 + \sum_{j=1}^{p} x_j x_{ij})^d \tag{5}$$

- radial, γ is a positive constant

$$K(x, x_i) = exp(-\gamma||x - x_i||^2) = exp(-\gamma \sum_{j=1}^{p} (x_j - x_{ij})^2) \tag{6}$$

By substituting (4) in (3) and expanding each of the inner product, we could establish the correspondence between α_i (3) and parameters β_i in (1). For any type of kernel we can use `svm()` function giving appropriate value of the parameter `kernel`.

In Fig.4 examples for some data with non-linear boundary are presented.

```
##
## Call:
## svm(formula = as.factor(label) ~ ., data = D, kernel = "linear",
##     cost = 1e+05, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1e+05
##       gamma:  0.5
##
## Number of Support Vectors:  29
##
##  ( 13 16 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

```
##
## Call:
## svm(formula = as.factor(label) ~ ., data = D, kernel = "polynomial",
##     degree = 2, cost = 1e+05, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1e+05
##      degree:  2
##       gamma:  0.5
##      coef.0:  0
##
## Number of Support Vectors:  7
##
##  ( 3 4 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

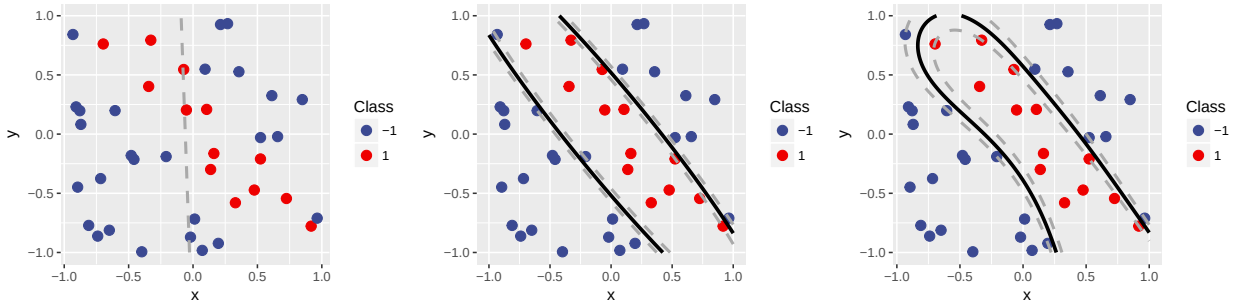

Figure 4: Black lines are class-separating boundaries. Grey lines are the margin lines. Cost is large. Left: SVM with a linear kernel. Support vector classifier peforms bad in this case. Data set is not linearly separable. Middle: SVM with a polynomial kernel of degree 2. Right: SVM with a radial kernel and parameter gamma is 0.5.

```
##
## Call:
## svm(formula = as.factor(label) ~ ., data = D, kernel = "radial",
##     cost = 1e+05, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1e+05
##       gamma:  0.5
##
## Number of Support Vectors:  9
##
##  ( 4 5 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

To find the best values for parameters `cost`, `gamma` and `degree` we use cross-validation with `tune()` function, which is a part of package `e1071`.

```
set.seed(2018)
svm.cv = tune(svm, as.factor(label) ~ ., data = D, kernel='radial',
              ranges = list(cost = c(1, 10, 100, 1000),
                            gamma = c(0.1, 0.5, 1, 2, 10)) )
svm.cv$best.parameters
```

```
##   cost gamma
## 6   10   0.5
```

```
svmfit.cv = svm(formula = as.factor(label) ~ ., data = D, kernel='radial',
                cost = 10,
                gamma = 0.5, scale = FALSE)
summary(svmfit.cv)
```

```
##
```

```
## Call:
## svm(formula = as.factor(label) ~ ., data = D, kernel = "radial",
##     cost = 10, gamma = 0.5, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.5
##
## Number of Support Vectors:  22
##
##  ( 11 11 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```