Support Vector Machines

Antonio D'Ambrosio

Outline

Introduction I

About supervised learning, we will talk about

- Maximal margin classifier;
- Support Vector Classifier;
- Support Vector Machines.

People often loosely refer to the maximal margin classifier, the support vector classifier, and the support vector machine as *support vector machines*. To avoid confusion, we will carefully distinguish between these three notions. Bit let's start from the concept of hyperplane.

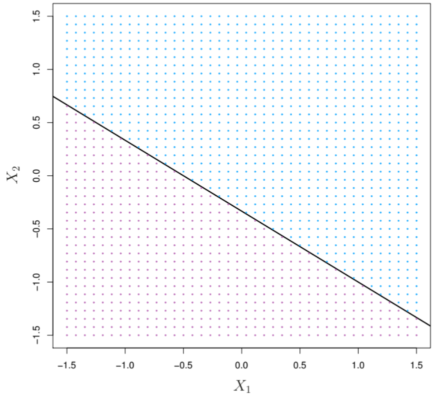
Hyperplane I

A hyperplane in p dimensions is a flat affine (it needs not pass through the origin) subspace of dimension $p - 1$.

$$f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p = 0$$

If $f(x_i) > 0$, then then x_i is located on one side of the hyperplane.
If $f(x_i) < 0$, it ias located on the other side.

Hyperplane II



Hyperplane $1 + 2X_1 + 3X_2 = 0$.

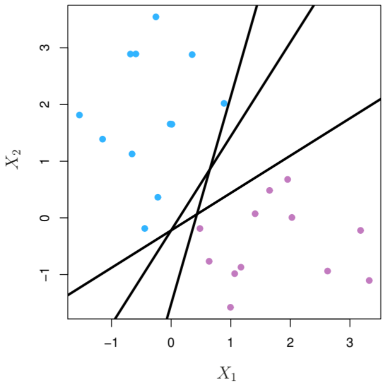
Separating hyperplane I

Suppose that we have a $n \times p$ data matrix X that consists of n training observations in p-dimensional space falling into two classes, $y_1, \ldots, y_n \in -1, 1$. Suppose also to have a test set, denoted by X^*

The goal is develop a classifier based on the training data that will correctly classify the test observations.

A separating hyperplane is a hyperplane that separates the training observations perfectly according to their class labels.

Separating hyperplane II



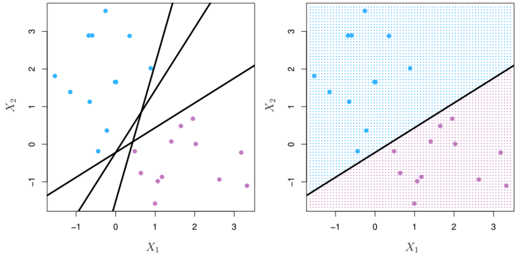
Three possible separating hyperplanes.

Maximal margin classifier I

In order to construct a classifier based upon a separating hyperplane, we must have a reasonable way to decide which of the infinite possible separating hyperplanes to use.

The maximal (or optimal) margin hyperplane is the separating hyperplane that is farthest from the training observations.

Maximal margin classifier II



In the figure (left) there are three separating hyperplanes, out of many possible, shown in black. On the right a separating hyperplane is shown, together with the decision rule (in blue and purple)

Constrained optimizazion problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{Maximize}}\ M$$

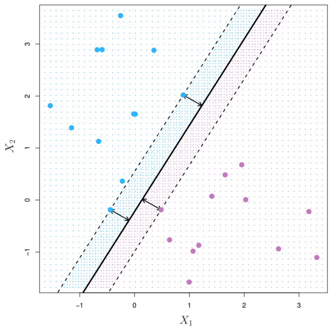
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$

Maximal margin classifier III

- The maximal margin hyperplane (or *maximal optimal separating hyperplane*), is the separating hyperplane that is farthest from the training observations.
- Once we compute the (perpendicular) distance from each training observation to a given separating hyperplane, the *margin* is defined as the minimal distance from the observations to the hyperplane.
- The *maximal margin hyperplane* is the separating hyperplane for which the margin is largest.
- Classifying a test observation based on which side of the maximal margin hyperplane it lies means that we're using the **maximal margin classifier**.

Support vectors



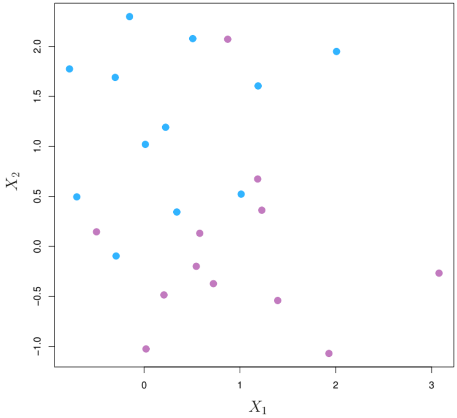
Two classes of observations (blue and purple). The **maximal margin hyperplane** is shown as a solid line. The **margin** is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the **support vectors**, and the distance from those points to the hyperplane is indicated by arrows.

The three training observations equidistant from the maximal margin hyperplane and lying along the dashed lines indicating the width of the margin are called *support vectors*, since they are vectors in p-dimensional space. They "support" the maximal margin hyperplane in the sense that if these points were moved slightly then the maximal margin hyperplane would move as well. The maximal margin hyperplane depends directly on the support vectors, not on the other observations.

Non-separable cases I

If cases are not perfectly separable, separating hyperplanes do not exist.

Non-separable cases II



Non-separable cases III

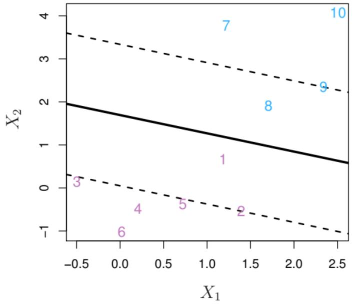
In this case, concept of a separating hyperplane can be extended to develop a hyperplane that *almost* separates the classes, using a so-called soft margin.

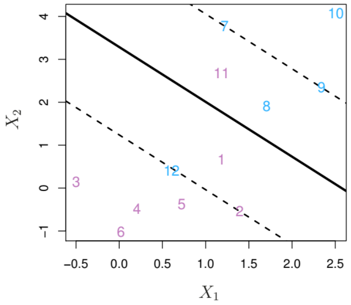
Support vector classifier I

We can consider a classifier based on a hyperplane that does not perfectly separate the two classes, in the interest of

- Greater robustness to individual observations;
- Better classification of most of the training observations.

Support vector classifier II





Most of the observations are on the correct side of the margin. However, a small subset of the observations are on the wrong side of the margin.

Observations on the wrong side of the hyperplane correspond to training observations that are misclassified by the support vector classifier

Formalization of maximal support classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_N,M}{Maximize} M$$

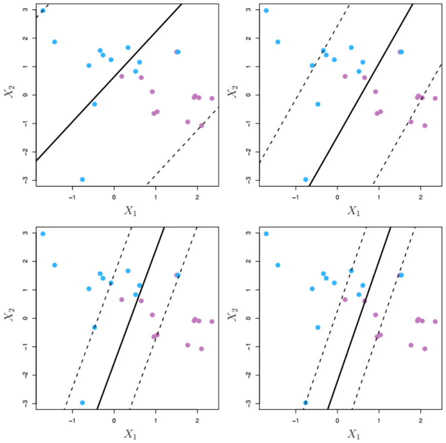
$$\text{subject to } \sum_{j=1}^{p}\beta_j^2 = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M(1-\epsilon_i)$$

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

C is a tuning parameter, M is the width of the margin, $\epsilon_1, \ldots, \epsilon_n$ are *slack* variables that allow individual observations to be on the wrong side of the margin or the hyperplane

Tuning parameter C



When C is large, then there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large. As C decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.

Slack variables and Tuning parameter

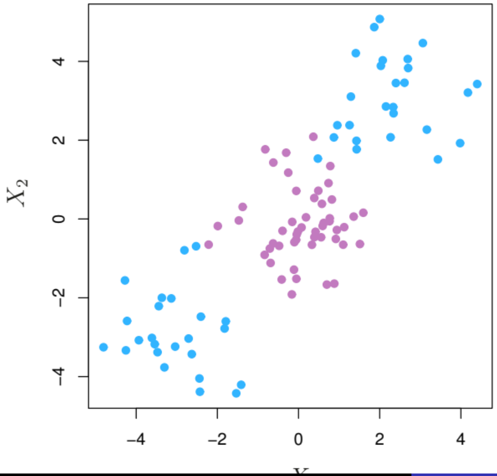
If $\epsilon_i = 0$ the ith observation is on the correct side of the margin. If $\epsilon_i > 0$ the ith observation is on the wrong side of the margin (violated the margin). If $\epsilon_i > 1$ the ith observation is on the wrong side of the hyperplane.

C can be seen as a budget for the amount that the margin can be violated by the n observations. If $C = 0$, then there is no budget (hence, support vector support classifier coincides with maximal margin classifier).
For $C > 0$ no more than C observations can be on the wrong side of the hyperplane.
C is generally chosen via cross-validation.

From Support vector classifiers to SVM I

The support vector machine (SVM) is an extension of the support vector support vector classifier resulting from enlarging the feature space in a specific way, using kernels.

From Support vector classifiers to SVM II



When there is a non-linear boundary between classes, the support vector classifier seeks a linear boundary, and consequently performs very poorly

From Support vector classifiers to SVM III

- Enlarge the space of features by including transformations; e.g. X_1^2, X_1^3, X_1X_2, $X_1X_2^2$, ... Hence go from a p-dimensional space to a $M > p$ dimensional space.
- Fit a support-vector classifier in the enlarged space.
- This results in non-linear decision boundaries in the original space.

Inner products I

The solution to the support vector classifier problem involves only the inner products of the observations $\langle x_i, x_{i'} \rangle$ (as opposed to the observations themselves).

Inner products II

- The linear support vector classifier can be represented as $f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$, where there are n parameters α_i, one per training observation.
- To estimate the parameters we need the $n(n-1)/2$ inner products $\langle x_i, x_{i'} \rangle$ between all pairs of training observations.
- It turns out that α_i is nonzero only for the support vectors in the solution, so we obtain $f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$, with $\mathcal{S}$ equal to the set of support vectors.

Kernels I

In representing the linear classifier $f(x)$, and in computing its coefficients, all we need are inner products.

We replace any inner product with a generalization of the inner product of the form $K(x_i, x_{i'})$, where K is a function called kernel, which is a function that quantifies the similarity of two observations.

Kernels II

- Linear kernel:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

The linear kernel essentially quantifies the similarity of a pair of observations using Pearson correlation

- Polynomial kernel:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} (1 + x_{ij} x_{i'j})^d$$

where d is the degree of the polynomial.

Kernels III

- Radial kernel:

$$K(x_i, x_{i'}) = exp\left(-\gamma\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2\right)$$

with γ being a positive constant.

Example 1 I

```
> #load the package performing SVM
> require(e1071)
> #load source for making nice plot
> source("funzione_plot_svm.R")
> #generate a data set
> set.seed(1) #for reproducibility
> x <- matrix(rnorm(20 * 2), ncol = 2)
> y <- c(rep(-1, 10), rep(1, 10))
> x[y == 1, ] <- x[y == 1, ] + 1
> #arrange data as a data frame
> dat <- data.frame(x = x, y = as.factor(y))
```

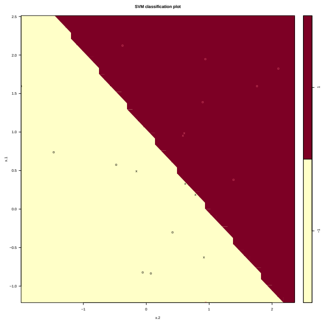
Example 1 II

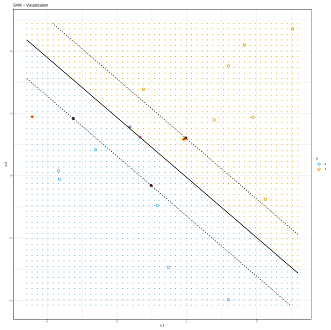
```
> #SVM with linear kernel, with
> #cost of constraints violation = 10
> svmfit <- svm(y ~ ., data = dat,
+               kernel = "linear",cost = 10,
+               scale = FALSE)


> #let's make the plot
> plot(svmfit, dat)
```

Example 1 III

```
> #let's make a better plot
> plsvm <- plot_svm_jk(dat,svmfit)
> #type plsvm to see the plot.
> #it's better highlights the support vectors
> #by exploiting the output of svm()
> plsvm+geom_point(data=as.data.frame(
+   svmfit$SV),colour="red")
```

Example 1 IV





The plot on the left inverts the axis (x_2 is on the horizontal axis, x_1 on the vertical one.) There are 7 support vectors (type `summary(svmfit)`)

More than two classes

The SVM as defined works for $k = 2$ classes. What do we do if we have $k > 2$ classes?

- *OVA* One versus All. Fit k different 2-class SVM classifiers, each class versus the rest. Classify x^* to the class for which $\hat{f}_k(x^*)$ is largest.
- *OVO* One versus One. Fit all $k(k-1)/2$ pairwise classifiers. Classify x^* to the class that wins the most pairwise competitions.

Which to choose? If K is not too large, use OVO.

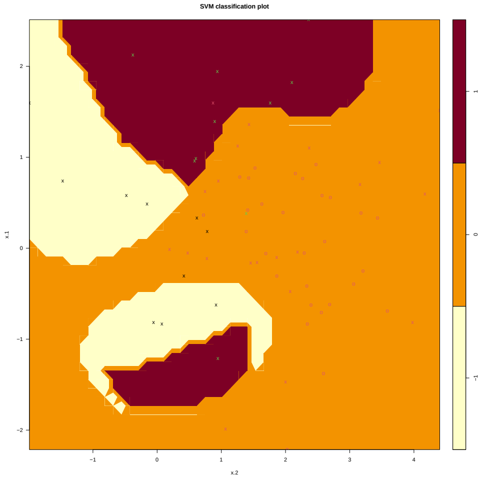
Example 2 I

```
> #Three classes example
> #generate a data set
> set.seed(1) #for reproducibility
> x <- rbind(x, matrix(rnorm(50 * 2), ncol = 2))
> y <- c(y, rep(0, 50))
> x[y == 0, 2] <- x[y == 0, 2] + 2
> #arrange data as a data frame
> dat <- data.frame(x = x, y = as.factor(y))
> svmfit <- svm(y ~ ., data = dat,
+              kernel = "radial",cost = 10,
+              gamma = 1)
> #from the help of svm: "For multiclass
```

Example 2 II

```
> #classification, libsvm uses the ‘one-against-
> #one’ approach

> plot(svmfit, dat)
```

Example 2 III

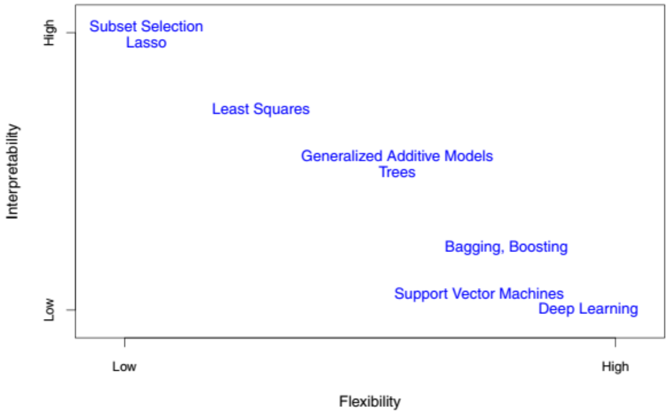


Which classifier we should use I

We studied several supervised learning methods: Linear models, Logit regression, Naive Bayes, Linear (and Quadratic) Discriminant Analysis, GLMs' family, KNN, tree-based methods, Ensemble Methods, SVM....

Prediction accuracy or Model interpretation?

Which classifier we should use II



Tradeoff between flexibility and interpretability

Which classifier we should use III

- If we are mainly interested in inference, then restrictive models are much more interpretable
- Fully non-linear methods such as bagging, boosting, support vector machines are highly flexible approaches that are harder to interpret.
- If we are only interested in prediction, and the interpretability of the predictive model is simply not of interest, it will be best to use the most flexible model available
- Not always it is the case: potential overfitting in highly flexible methods is highly probable.