Machine Learning and Statistical Learning

Support Vector Machines

Ewen Gallic
ewen.gallic@gmail.com

MASTER in Economics - Track EBDS - 2nd Year

Support Vector Machines

This section presents another type of classifiers known as **support vector machines** (SVM). It corresponds to the 9th chapter of James et al. (2013).

Berk (2008) states that "*SVM can be seen as a worthy competitor to random forests and boosting*".

We will cover:

- The **maximal margin classifier**, which requires the classes of the response variable to be separable by a **linear boundary**.
- The **support vector classifier** which is a generalization of the maximal margin classifier and allows an boundary that does not perfectly separate the classes.
- The **support vector machines** which further allow for non-linear boundaries.

1. Maximal margin classifier

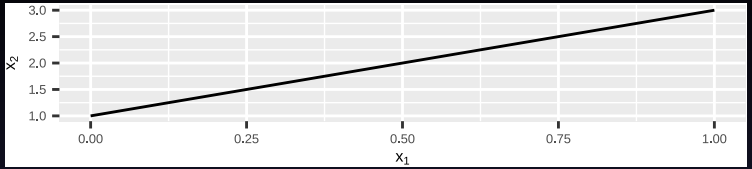
Hyperplane

We will use the notion of a separating hyperplane in what follows. Hence, a little detour on recalling what a hyperplane is seems fairly reasonable.

In a p-dimensional space, a **hyperplane** is a flat affine subspace of dimension $p-1$ (a subspace whose dimension is one less than that of its ambiant space).

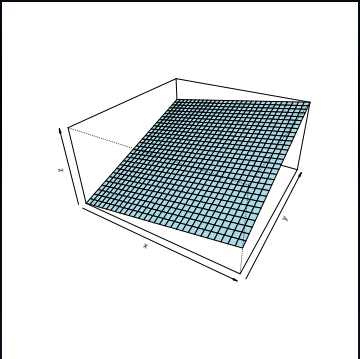
Let us consider an example.

In two dimensions ($p=2$), a hyperplane is a one-dimensional subspace: a line.



Hyperplane

In three dimensions ($p = 3$), a hyperplane is a flat two-dimensional subspace ($p = 2$): a plane.



Hyperplane

In two dimensions, for parameters β_0, β_1 and β_2, the following equation defines the hyperplane:

$$\beta_0 + \beta_1 \mathbf{x}_1 + \beta_2 \mathbf{x}_2 = 0 \tag{1}$$

Any point whose coordinates for which Eq. 1 holds is a point on the hyperplane.

In a p-dimension ambiant space, an hyperplace is defined for parameters β_0, β_1, ..., β_p by the following equation:

$$\beta_0 + \beta_1 \mathbf{x}_1 + ... + \beta_p \mathbf{x}_p = 0 \tag{2}$$

Any point whose coordinates are given in a vector of length p, *i.e.*, $\mathbf{X} = \begin{bmatrix} \mathbf{x_1} & ... & \mathbf{x}_p \end{bmatrix}^\top$ for which Eq. 2 holds is a point on the hyperplane.

Space divided in two halves

If a point does not satisfy Eq. 2, then it lies either in one side or another side of the hyperplane, *i.e.*,

$$\begin{aligned} \beta_0 + \beta_1 \mathbf{x}_1 + ... + \beta_p \mathbf{x}_p > 0 \text{or} \\ \beta_0 + \beta_1 \mathbf{x}_1 + ... + \beta_p \mathbf{x}_p < 0 \end{aligned} \tag{3}$$

Hence, the hyperplane can be viewed as a subspace that divides a p-dimensional space in two halves.

Example

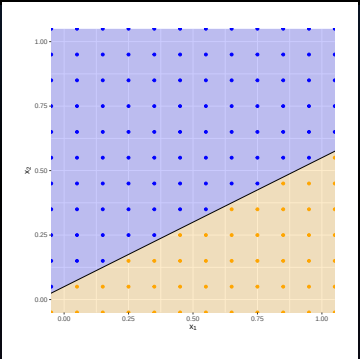


Figure 1: Hyperplane $x_1 - 2x_2 + 0.1$. The blue region corresponds to the set of points for which $x_1 - 2x_2 + 0.1 > 0$, the orange region corresponds to the set of points for which $x_1 - 2x_2 + 0.1 < 0$.

Classification and hyperplane

Let us consider a simplified situation to begin with the classification problem.

Suppose that we have a set of n observations $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, where the response variable can take two values $\{\text{class } 1, \text{class } 2\}$, depending on the relationship with the p predictors.

In a first simplified example, let us assume that it is possible to construct a **separating hyperplane** that separates **perfectly** all observations.

In such a case, the hyperplane is such that:

$$\begin{cases} \beta_0 + \beta_1 x_1 + ... + \beta_p x_p > 0 & \text{if } y_i = \text{class 1} \\ \beta_0 + \beta_1 x_1 + ... + \beta_p x_p < 0 & \text{if } y_i = \text{class 2} \end{cases} \tag{4}$$

Classification and hyperplane

For convenience, as it is often the case in classification problem with a binary outcome, the response variable y can be coded as 1 and -1 (for class 1 and class 2, respectively). In that case, the hyperplane has the property that, for all observations $i = 1, ..., n$:

$$y_i(\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}) > 0 \tag{5}$$

Classification and hyperplane

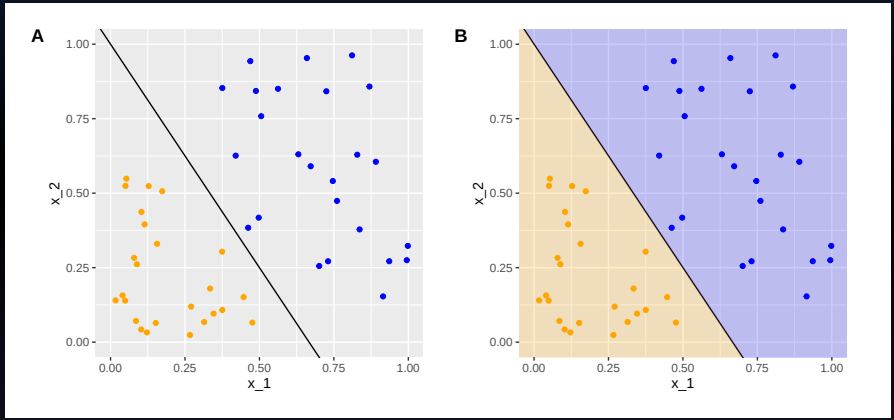


Figure 2: A perfectly separating linear hyperplane for a binary outcome.

Classification and hyperplane

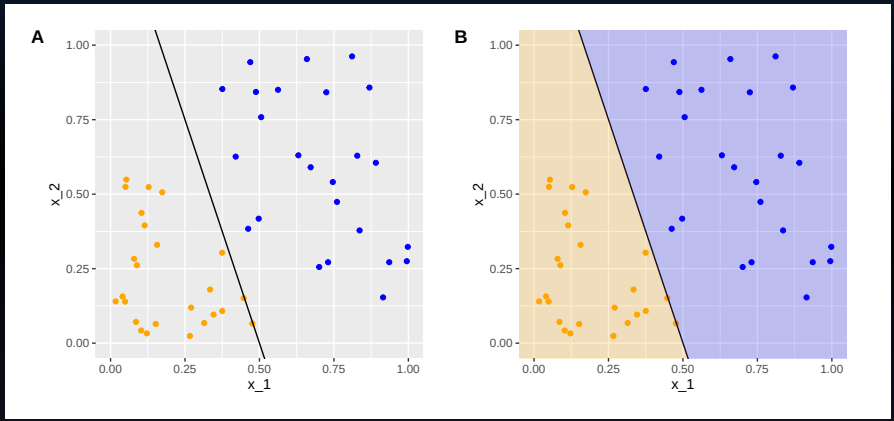


Figure 3: A perfectly separating linear hyperplane for a binary outcome.

Classification and hyperplane

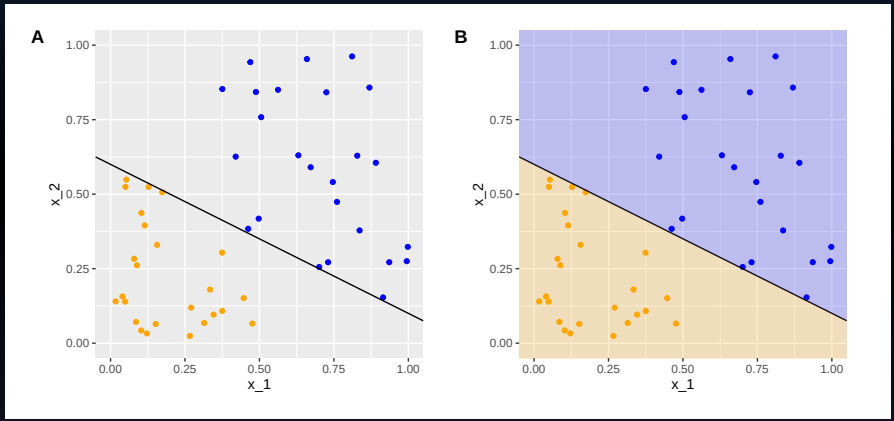


Figure 4: A perfectly separating linear hyperplane for a binary outcome.

Margin

In the previous example,there exists an infinity of perfectly separating hyperplanes.

As usual, we would like to decide among the possible set, what is the **optimal choice**, regarding some criterion.

A solution consists in computing the distance from each observation to a given separating hyperplane. The distance which is the smallest is called the **margin**. The objective is to select the separating hyperplane for which the margin is the farthest from the observations, *i.e.*, to select the maximal margin hyperplane.

This is known as **the maximal margin hyperplane**.

Margin

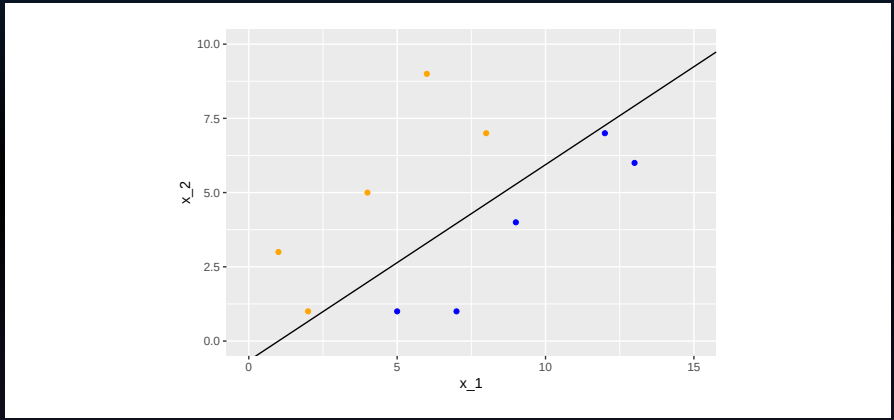


Figure 5: Margin given a specific separating hyperplane (1/4).

Margin

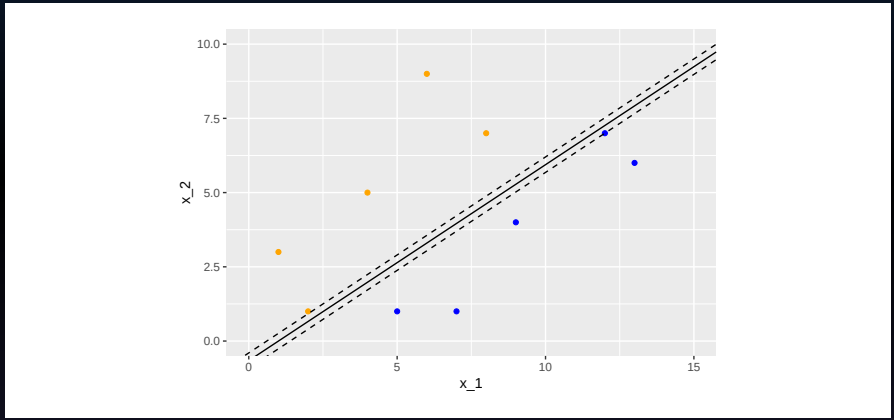


Figure 6: Margin given a specific separating hyperplane (1/4).

Margin

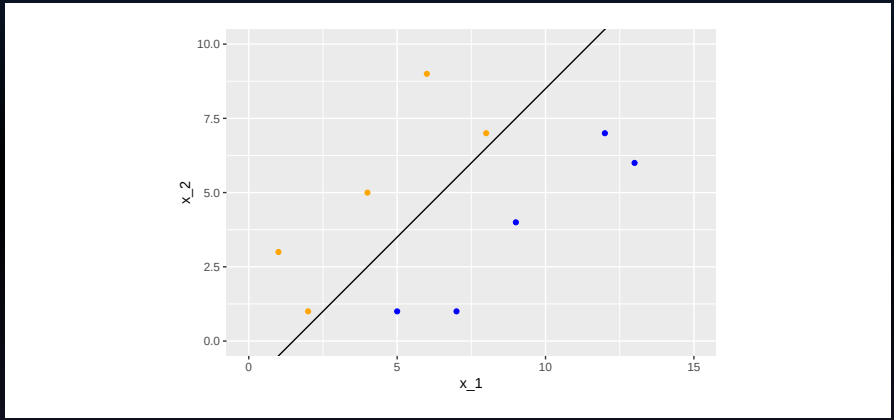


Figure 7: Margin given a specific separating hyperplane (2/4).

Margin

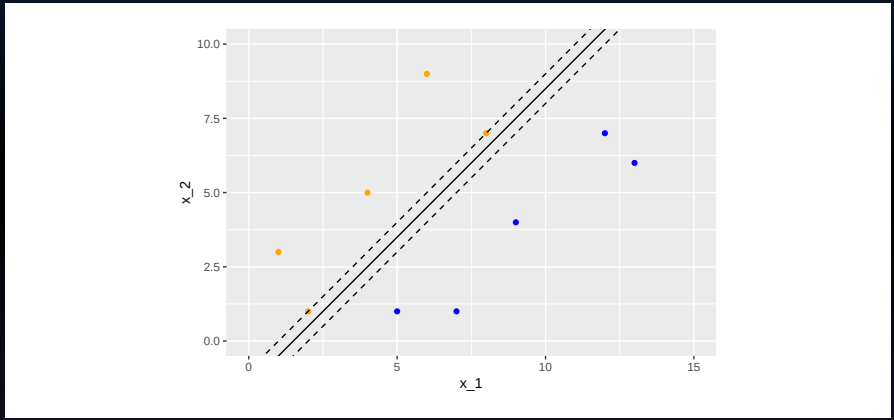


Figure 8: Margin given a specific separating hyperplane (2/4).

Margin

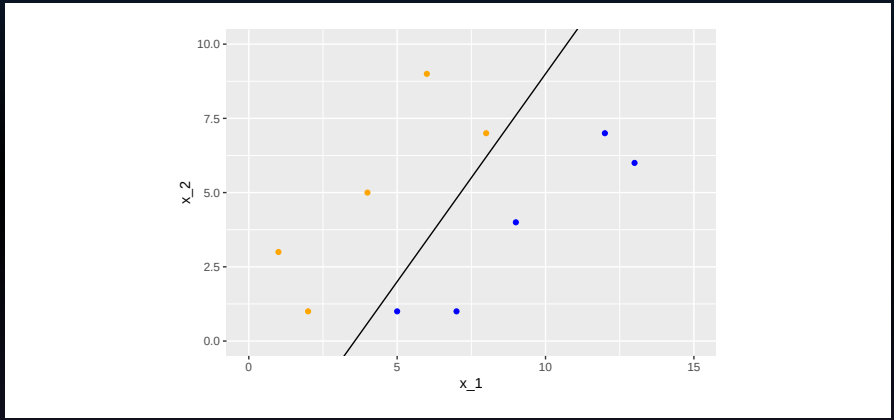


Figure 9: Margin given a specific separating hyperplane (3/4).

Margin

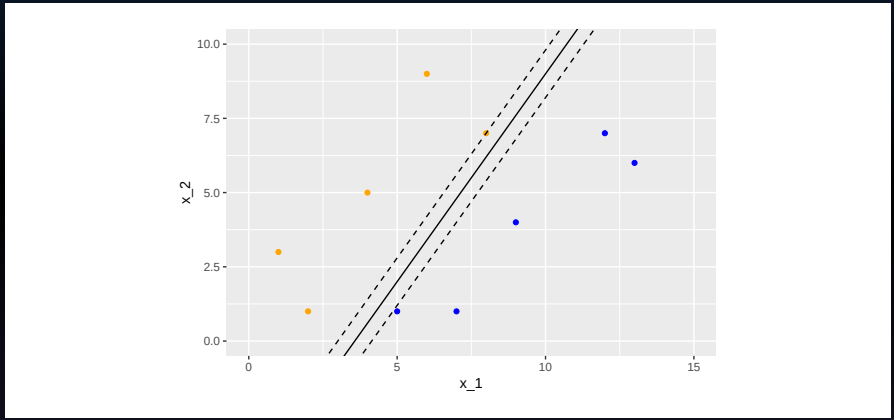


Figure 10: Margin given a specific separating hyperplane (3/4).

Margin

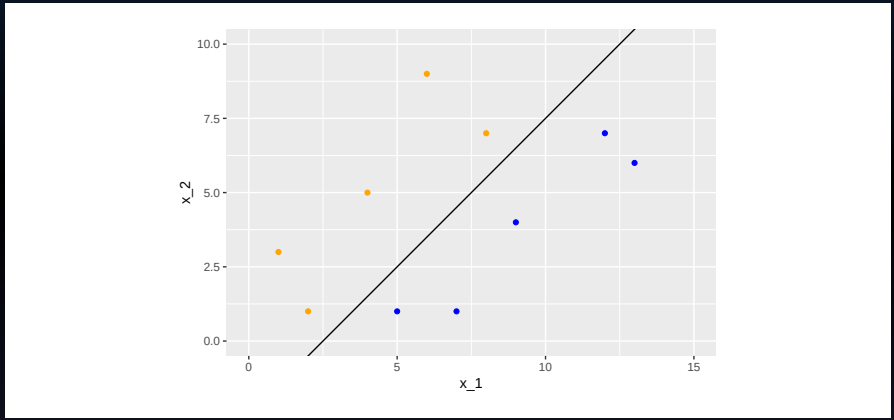


Figure 11: Margin given a specific separating hyperplane (4/4).

Margin

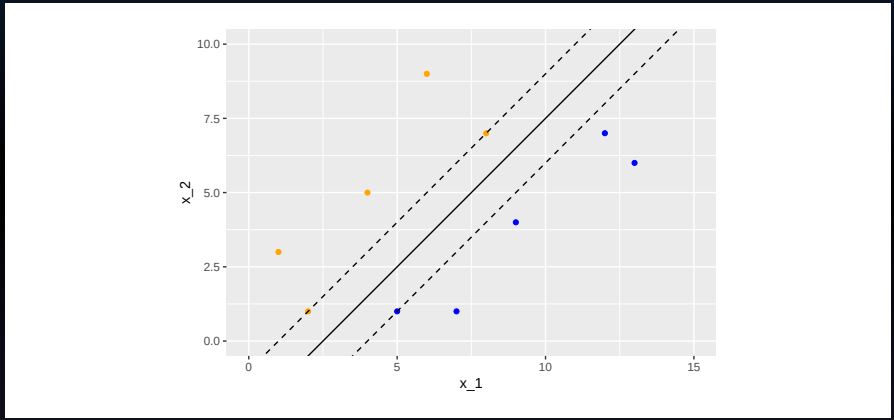


Figure 12: Margin given a specific separating hyperplane (4/4).

Maximum Margin

How do we find the maximum margin? It is an **optimization problem**:

$$\max_{\beta_0,\beta_1,...,\beta_p,M} M \tag{6}$$

$$\text{s.t. } \sum_{j=1}^{p} \beta_j^2 = 1 \tag{7}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}) \geq M, \forall i = 1, ..., n \tag{8}$$

Constraint 8 ensures that each obs. is on the correct side of the hyperplane.

Constraint 7 ensures that the perpendicular distance from the ith observation to the hyperplane is given by:

$$y_i(\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip})$$

Constraints 7 and 8: each observation is on the correct side of the hyperplane and at least a distance M from the hyperplane.

Maximum Margin

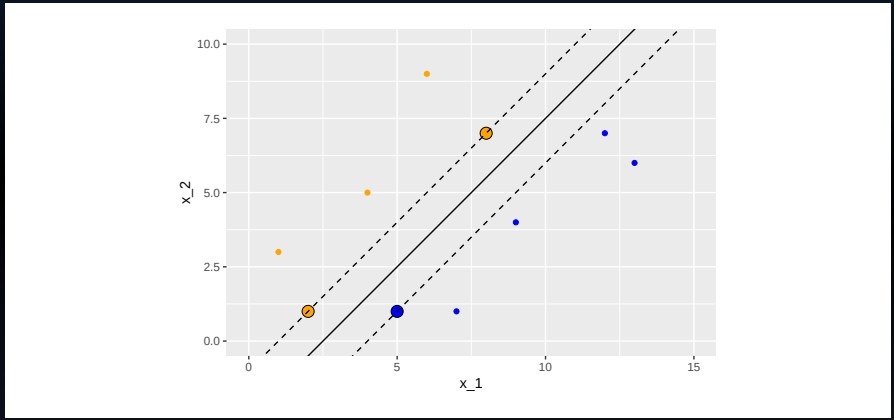


Figure 13: Maximum margin classifier for a perfectly separable binary outcome variable.

Maximum Margin

In the example shown previously, there are 3 observations from the training set that are equidistant from the maximal margin hyperplane.

These points are known as the **support vectors**:

- they are vectors in p-dimensional space
- they "support" the maximal margin hyperplane (if they move, the maximal margin hyperplane also moves)

For any other points, if they move but stay outside the boundary set by the margin, this does not affect the separating hyperplane.

So the observations that fall in top of the fences are called **support vector** because they directly determine where the fences will be located.

In our example, the maximal margin hyperplane only depends on three points, but this is not a general result. The number of support vectors can vary according to the data.

Maximum margin and classification

Once the maximum margin is known, **classification** follows directly:

- cases that fall on one side of the maximal margin hyperplane are labeled as one class
- cases that fall on the other side of the maximal margin hyperplane are labeled as the other class

The classification rule that follows from the decision boundary is known as **hard thresholding**.

Leaving the perfectly separable margin

So far, we were in a simplified situation in which it is possible to find a perfectly separable hyperplane.

In reality, data are not always that cooperative, in that:

- there is no maximal margin classifier (the set of values are no longer linearly separable)
- the optimization problem gives no solution with $M > 0$

In such cases, we can allow some number of observations to violate the rules so that they can lie on the wrong side of the margin boundaries. We can develop a hyperplane that *almost* separates the classes.

The generalization of the maximal margin classifier to the non-separable case is known as the **support vector classifier**.

2. Support vector classifiers

Support vector classifiers

In this section, we consider the case in which finding a maximal margin classifier is either **not possible or not desirable**.

A maximal margin classifier may not be desired as:

- its margins can be too narrow and therefore lead to relatively higher generalization errors
- the maximal margin hyperplane may be too sensitive to a change in a single observation

As it is usually the case in statistical methods, a trade-off thus arises: here it consists in trading some accuracy in the classification for more robustness in the results.

Support vector classifiers

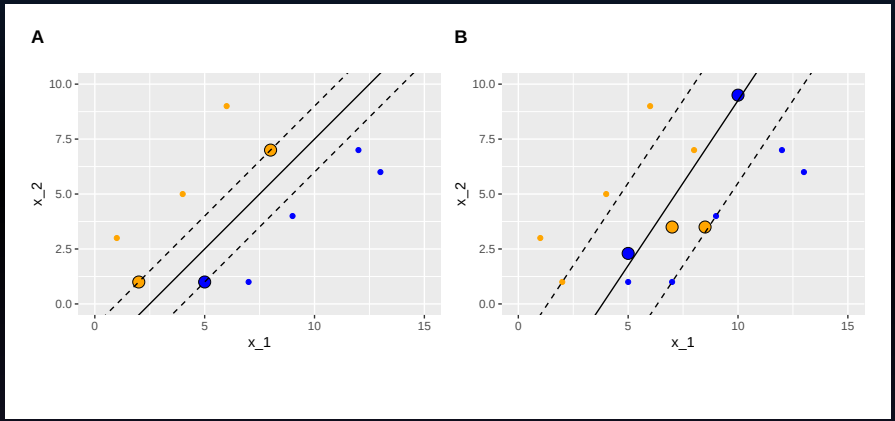


Figure 14: Maximal margin classifier for the initial dataset with linearly separable observations (panel A) and support vector classifier for the same dataset where four points were added (Panel B).

Support vector classifiers

When we allow some points to violate the buffer zone, the optimization problem becomes:

$$\max_{\beta_0,\beta_1,...,\beta_p,\varepsilon_1,...,\varepsilon_n,M} M \tag{9}$$

$$\text{s.t. } \sum_{j=1}^{p} \beta_j^2 = 1, \tag{10}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}) \geq M(1 - \varepsilon_t), \forall i = 1, ..., \tag{11}$$

$$n) \tag{12}$$

$$\varepsilon_i \geq 0, \sum_{i=1}^{n} \varepsilon_i \leq C, \tag{13}$$

where C is a nonnegative **tuning** parameter, M is the width of the margin.

$\varepsilon_1, ..., \varepsilon_n$ allow individual observations to lie on the wrong side of the margin or the hyperplane, they are called **slack variables**.

Classification

Once the optimization problem is solved, the **classification** follows instantly by looking at **which side of the hyperplane** the observation lies:

- hence, for a new observation x_0, the classification is based on the sign of $\beta_0 + \beta_1 x_0 + ... + \beta_p x_0$.

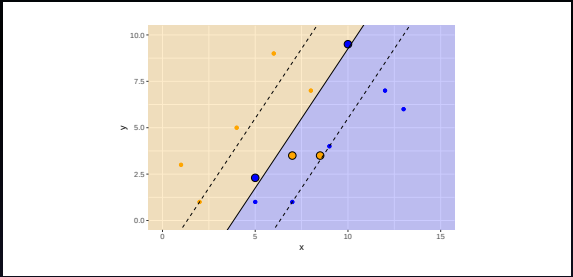


Figure 15: Support vector classifier for the binary outcome variable.

More details on the slack variables

The slack variable ε_i indicates where the ith observation is located relative to both the hyperplane and the margin:

- $\varepsilon_i = 0$: the ith observation is on the correct side of the margin
- $\varepsilon_i > 0$: the ith observation is on the wrong side of the margin
- $\varepsilon_i > 1$: the ith observation is on the wrong side of the hyperplane

More details on the tuning parameter

The tuning parameter C from Eq. 13 reflects a measure of **how permissive** we were when the margin was maximized, as it bounds the sum of ε_i.

- Setting a value of C to 0 implies that we do not allow any observations from the training sample to lie in the wrong side of the hyperplane. The optimization problem then boils down to that of the maximum margin (if the two classes are perfectly separable).
- If the value of C is greater than 0, then no more than C observations can be on the wrong side of the hyperplane:
  - $\varepsilon_i > 0$ for each observation that lies on the wrong side of the hyperplane
  - $\sum_{i=1}^{n} \varepsilon_i \leq C$

More details on the tuning parameter

So, the idea of the support vector classifier can be viewed as maximizing the width of the buffer zone conditional on the slack variables. But the distance of some slack variables to the boundary can vary from one observation to another. The sum of these distances can then be viewed as a measure of how permissive we were when the margin was maximized:

- the **more permissive**, the larger the sum, the **higher the number of support vectors**, and the easier to locate a separating hyperplane within the margin
- but on the other hand, being more permissive can lead to a **higher bias** as we introduce **more misclassifications**.

More details on the tuning parameter

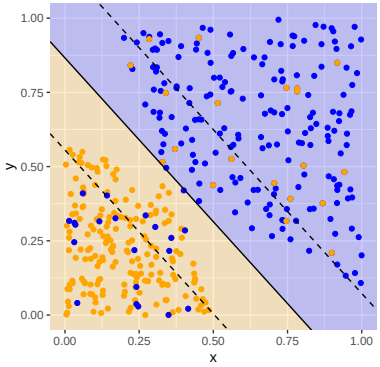


Figure 16: Support vector classifier fitted using different values of C (increasing values) (1/4).

More details on the tuning parameter

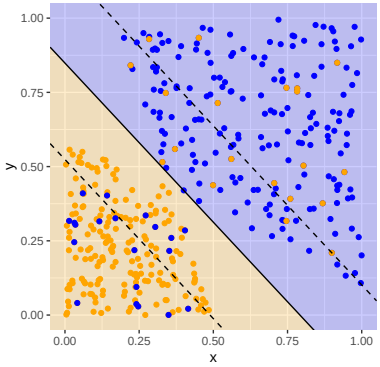


Figure 17: Support vector classifier fitted using different values of C (increasing values) (2/4).

More details on the tuning parameter

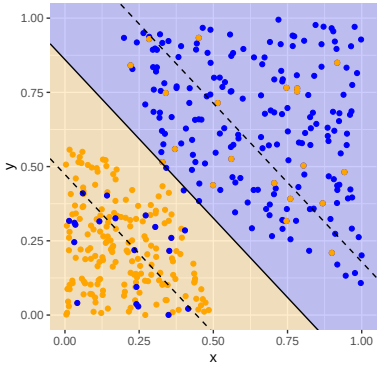


Figure 18: Support vector classifier fitted using different values of C (increasing values) (3/4).

More details on the tuning parameter

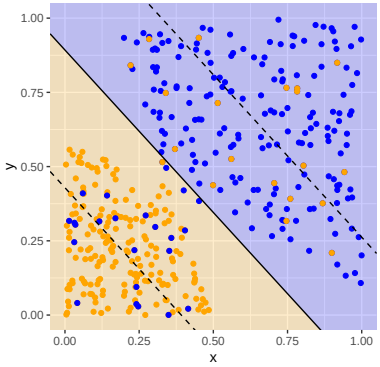


Figure 19: Support vector classifier fitted using different values of C (increasing values) (4/4).

3. Support vector machines

Support vector machines

In this section, we will look at a solution to classification problems when the classes are **not linearly separable**.

The basic idea is to **convert a linear classifier** into a classifier that produced **non-linear decision boundaries**.

Support vector classifier with non-linear boundaries

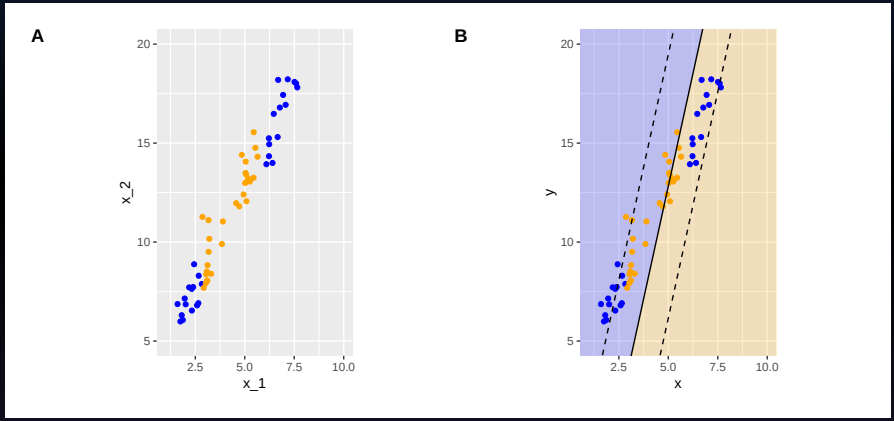


Figure 20: Binary outcome variables (panel A), support vector classifier boundaries (panel B).

Support vector classifier with non-linear boundaries

To account for **non-linear boundaries**, it is possible to add more dimensions to the observation space:

- by adding polynomial functions of the predictors
- by adding interaction terms between the predictors.

However, as the number of predictors is enlarged, the computations become harder...

The **support vector machine** allows to enlarge the number of predictors while keeping efficient computations.

Support vector machine

The idea of the **support vector machine** is to fit a separating hyperplane in a space with a higher dimension than the predictor space.

Instead of using the set of predictors, the idea is to use a kernel.

The solution of the optimization problem given by Eq. 9 to Eq. 13 involves only the **inner products** of the observations.

The inner product of two observations x_1 and x_2 is given by $\langle x_1, x_2 \rangle = \sum_{j=1}^{p} x_{1j} x_{2j}$.

Support vector machine

The **linear support vector classifier** can be represented as:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle, \tag{14}$$

where the n parameters α_i need to be estimated, as well as the parameter β_0.

This requires to compute all the $\binom{n}{2}$ inner products $\langle x_i, x_{i'} \rangle$ between all pairs of training observations.

We can see in Eq. 14 that if we want to evaluate the function f for a new point x_0, we need to compute the inner product between x_0 and each of the points x_i from the training sample.

Support vector machine

If a point x_i from the training sample **is not from the set $\mathcal{S}$ of the support vectors**, then it can be shown that α_i is equal to zero.

Hence, Eq. 14 eventually writes:

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle, \tag{15}$$

thus reducing the computational effort to perform when evaluating f.

Using a Kernel

Now, rather than using the actual inner product $\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$ when it needs to be computed, let us assume that we replace it with a **generalization** of the inner product, following some functional form K known as a **kernel**: $K(x_i, x_{i'})$.

A kernel will compute the similarities of two observations.

For example, if we pick the following functional form:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}, \tag{16}$$

it leads back to the **support vector classifier** (or the **linear kernel**).

Using a non-linear kernel

We can use a non-linear kernel, for example a **polynomial kernel** of degree d:

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d, \tag{17}$$

If we do so, the decision boundary will be more flexible. The functional form of the classifier becomes:

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i) \tag{18}$$

an is such a case, when the support vector classifier is combined with a non-linear kernel, the resulting classifier is known as a **support vector machine**.

Using a polynomial kernel

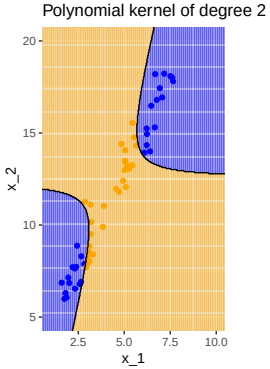


Figure 21: Support vector machine with a polynomial kernel (1/4).

Using a polynomial kernel

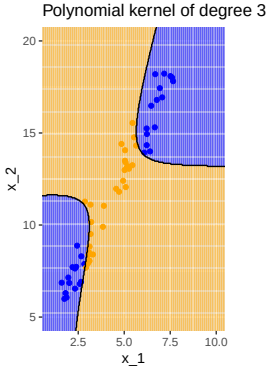


Figure 22: Support vector machine with a polynomial kernel (2/4).

Using a polynomial kernel

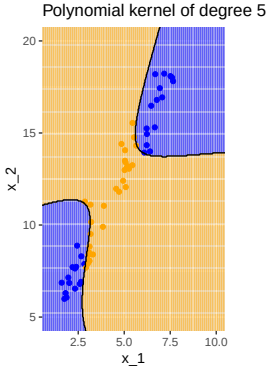


Figure 23: Support vector machine with a polynomial kernel (3/4).

Using a polynomial kernel

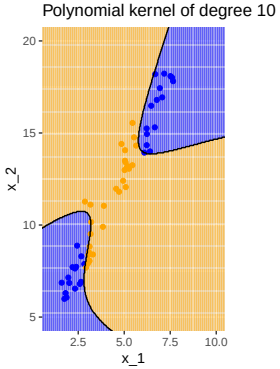


Figure 24: Support vector machine with a polynomial kernel (4/4).

Using a radial kernel

Other kernels are possible, such as the **radial kernel**:

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right), \tag{19}$$

where γ is a positive constant that accounts for the smoothness of the decision boundary (and also controls the variance of the model):

- very large values lead to fluctuating decision boundaries that accounts for high variance (and may lead to overfitting)
- small values lead to smoother boundaries and low variance.

Using a radial kernel

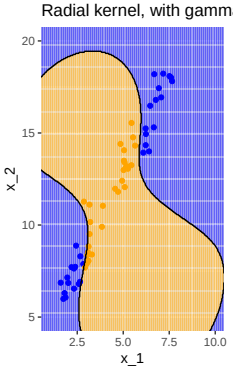


Figure 25: Support vector machine with a radial kernel (1/5).

Using a radial kernel

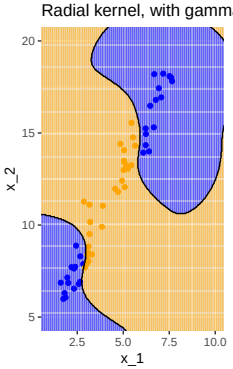


Figure 26: Support vector machine with a radial kernel (2/5).

Using a radial kernel

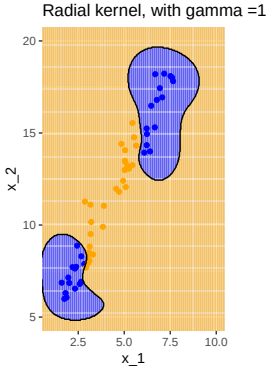


Figure 27: Support vector machine with a radial kernel (3/5).

Using a radial kernel

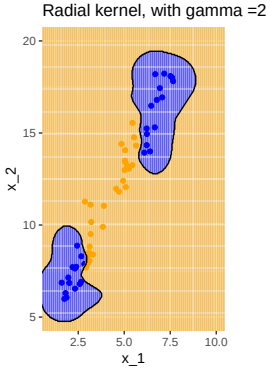


Figure 28: Support vector machine with a radial kernel (4/5).

Using a radial kernel

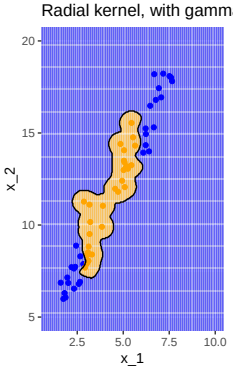


Figure 29: Support vector machine with a radial kernel (5/5).

Using a radial kernel

Recall the form of the radial kernel:

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

If a test observation x_0 is far (considering the Euclidean distance) from a training observation x_i:

- $\sum_{j=1}^{p}(x_{0j} - x_{ij})^2$ will be large
- hence $K(x_0, x_i) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{0j} - x_{ij})^2\right)$ will be really small
- hence x_i will play no role in $f(x_0)$

So, observations far from x_0 will play no role in its predicted class: the radial kernel therefore has a *local* behavior.

4. Support vector machines with more than two classes

Support vector machines with more than two classes

The classification of binary response variables using SVM can be extended to multi-classes response variables.

We will briefly look at two popular solutions:

1. the one-versus-one approach
2. the one-versus-all approach.

One-versus-one classification

If we face a response variable with $K > 2$ different levels, an approach to perform classification using SVM, known as **one-versus-one classification**, consists in constructing $\binom{K}{2}$ SVM:

- each SVM compares a pair of classes

For example, one of these $\binom{K}{2}$ SVM may compare the k-th class coded as $+1$ to another class k' coded as -1.

To assign a final classification to a test observation x_0:

- the class to which it was most frequently assigned in the $\binom{K}{2}$ pairwise classifications is selected.

One-versus-all classification

If we face a response variable with $K > 2$ different levels, another approach to perform classification using SVM, known as **one-versus-all classification**, consists in constructing K SVM:

- each SVM compares a one class to the other classes.

For example, one of these K SVM may compare the kth class coded $+1$ to the remaining classes coded as -1.

To assign a final classification to a test observation x_0:

- the class for which $\beta_{0k} + \beta_{1k}x_{01} + ... + \beta_{pk}x_{0p}$ is the largest is selected
  - it amounts to a high level of confidence that the test observation belongs to the kth class rather than to any of the other classes.

Berk, Richard A. 2008. *Statistical Learning from a Regression Perspective*. Vol. 14. Springer.
James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Vol. 112. Springer.