Support Vector Machines

- Separating hyperplanes
- Support Vector Classifier (allowing for errors)
- Support Vector Machine (non-linear decision boundaries)

Separating hyperplane

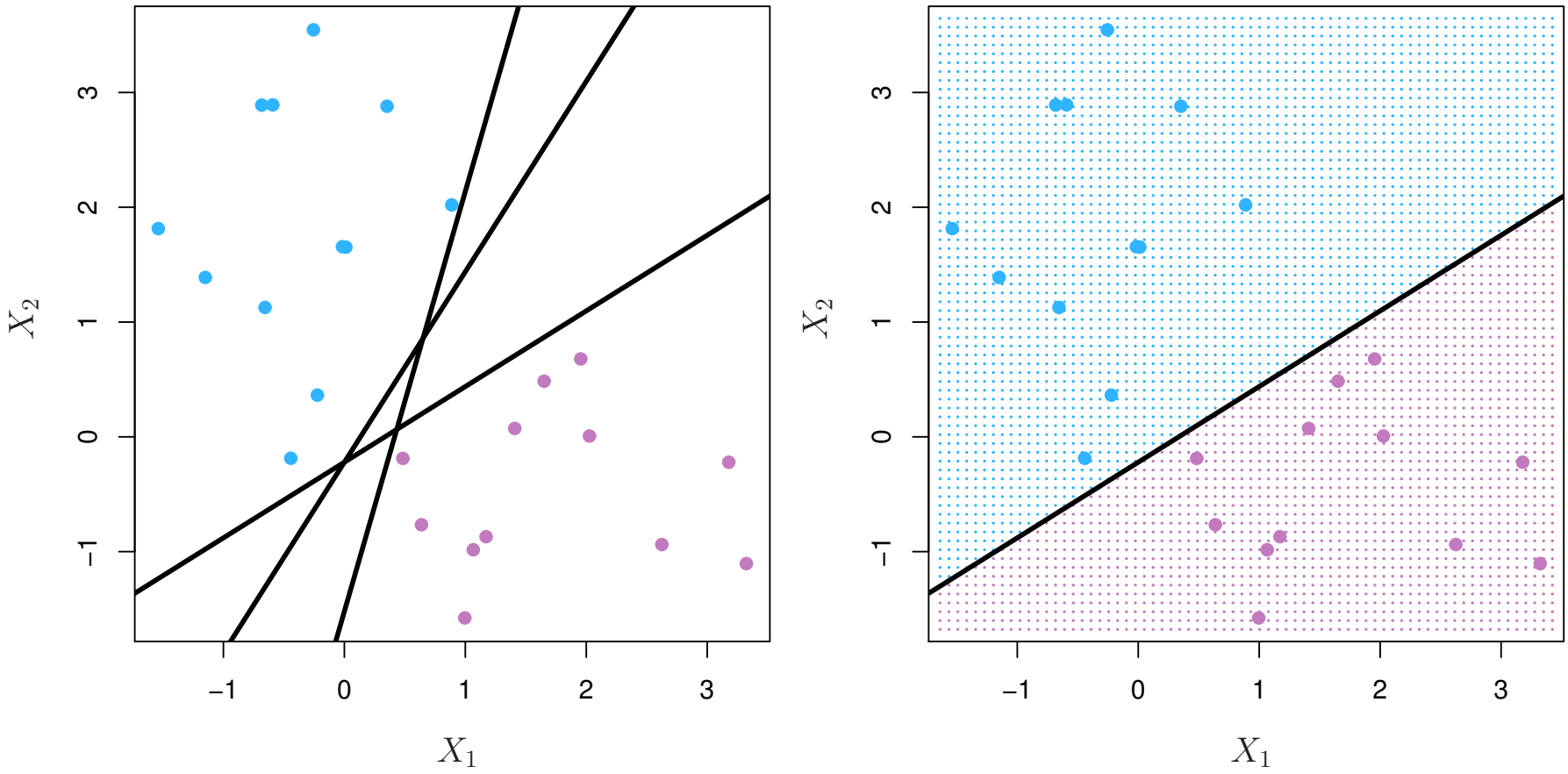
- A hyperplane can be seen as an equation of the features (predictor variables), e.g.:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

- If we code $y \in \{-1, 1\}$ then a perfect separating hyperplane has the following property

$$y_i \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}\right) > 0$$

Seperating hyperplane



Maximum Margin Classifier

- There may be several perfectly separating hyperplanes, then the one with the largest margin is chosen.

- This amounts to:

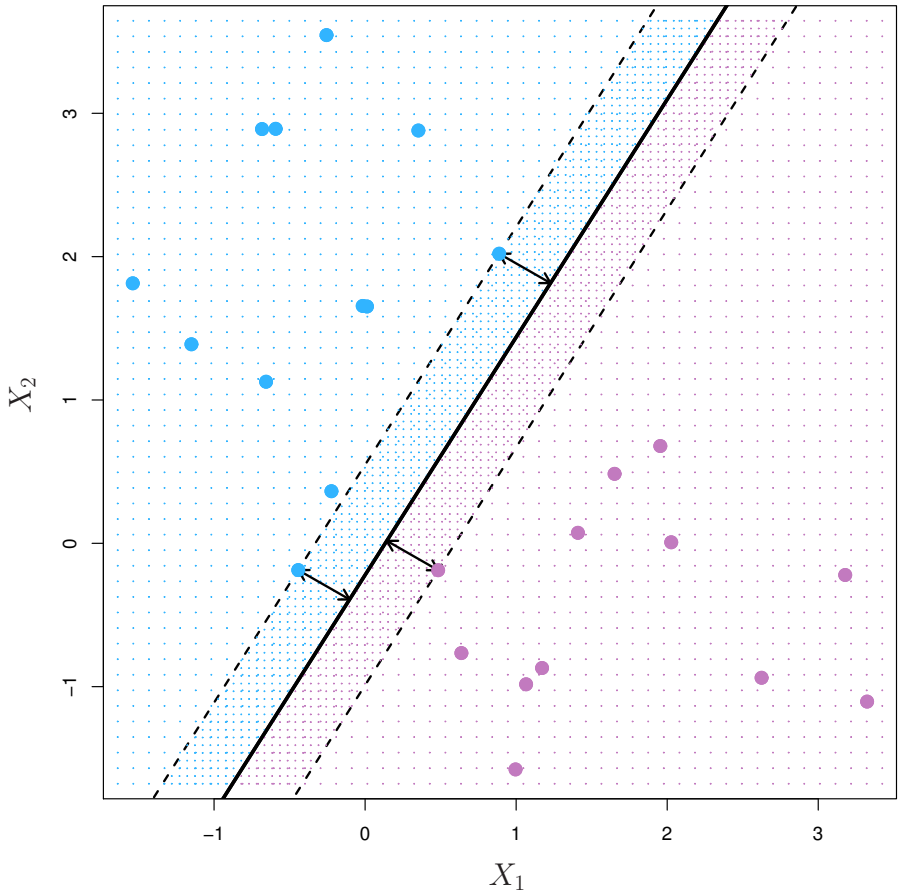
$$\text{maximize } M(\beta_0, \ldots, \beta_p)$$

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

$$y_i \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}\right) \geq M$$

  where $i = 1, \ldots, n$

Maximum Margin Classifier



Support Vector Classifier

- Often no perfectly separating hyperplane exists.
- Generalize the maximum margin classifier idea using a *soft margin*:
- Allow some observations to be on the wrong side of the margin or even of the hyperplane.
- We introduce *slack variables*: $\epsilon_1, \ldots, \epsilon_n$ and a tuning parameter C:

$$\text{maximize } M(\beta_0, \ldots, \beta_p, \epsilon_1, \ldots, \epsilon_n)$$

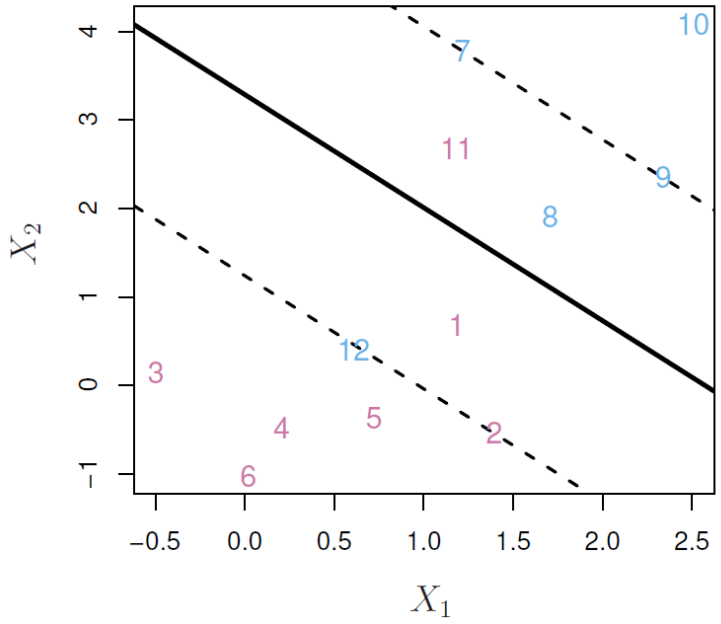
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i \left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}\right) \geq M(1 - \epsilon_i)$$

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

Support Vector Classifier

- The *slack variable*: $\epsilon_1, \ldots, \epsilon_n$ tells us where the i-th observation is located, relative to margin and hyperplane:
  - If $\epsilon_i = 0$, observation i is on the correct side of the margin.
  - If $\epsilon_i > 0$, observation i is on the incorrect side of the margin.
  - If $\epsilon_i > 1$, observation i s on the incorrect side of the hyperplane.

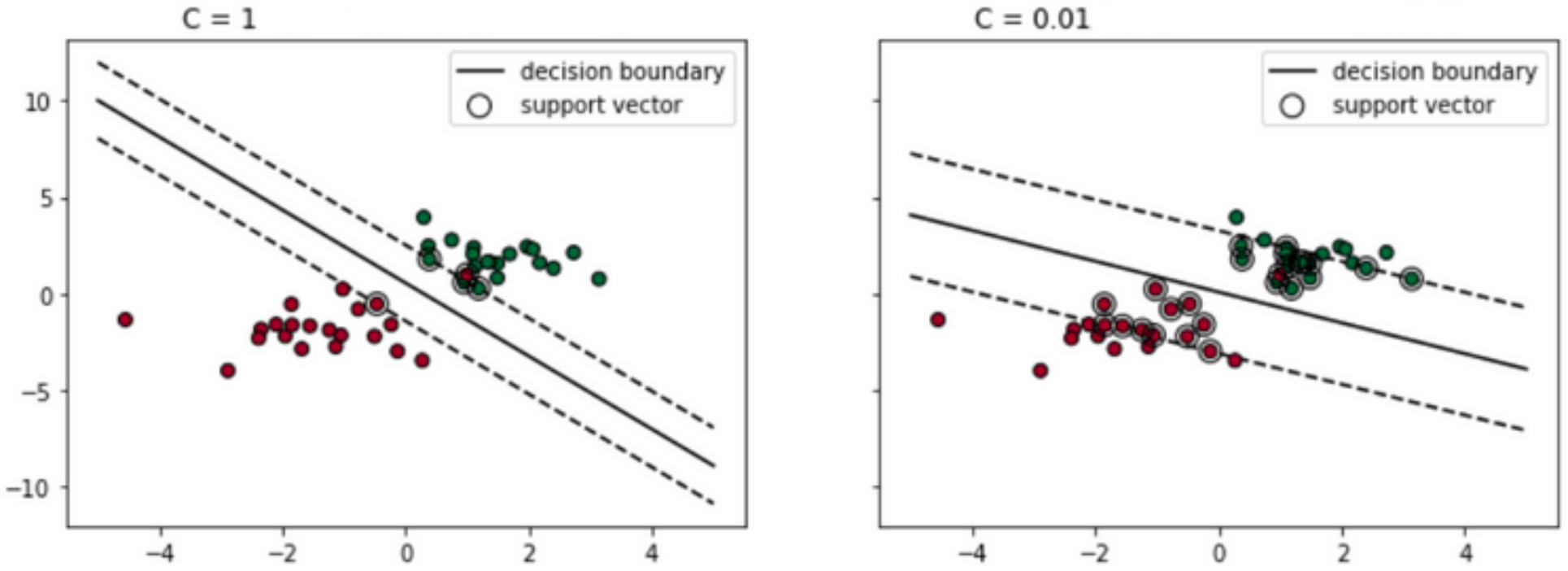


Support Vector Classifier

- In the book's definition, tuning parameter C limits the sum of ϵ_i values, functioning like a budget:
  - $C = 0$: no budget for violations
  - $C > 0$: no more than C observations can be on wrong side of the hyperplane
  - The larger C, the more tolerant the classifier becomes of errors (allows more *slack*), margin becomes wider)

- Software implementations of SVMs generally have a `cost` argument, which has the opposite effect: It specifies the cost of misclassifications; if cost increases, classifier becomes *less* tolerant of errors:

Support Vector Classifier



Larger cost yields more narrow margin; lower bias and higher variance.

Support Vector Classifier

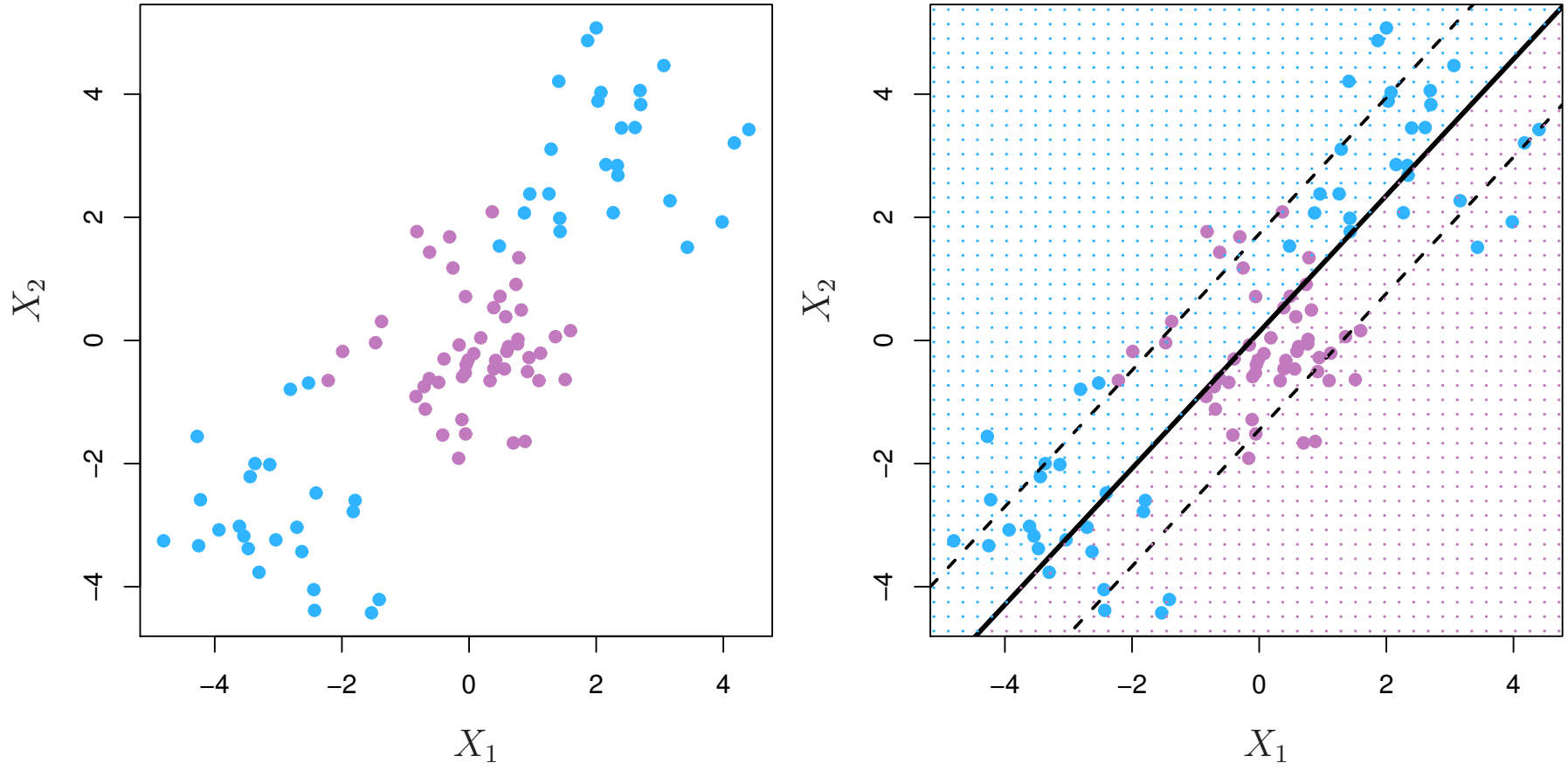
- Observations that are within margins or on the wrong side of the hyperplane are called *support vectors*

- The decision rule (hyperplane and margins) are based on a small fraction of training observations, which makes it robust to observations far from the hyperplane

- In contrast, Linear Discriminant Analysis depends on the mean of all observations to construct the discriminant function and is therefore sensitive to points far from the decision line.

Support Vector Machine

In practice we might need a non-linear hyperplane. E.g.:



Support Vector Machine

- We can enlarge the feature space to allow for non-linear boundaries
- Instead of fitting a support vector classifier on

$$X_1, X_2, \ldots, X_p$$

  we can fit it on

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

  or on

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2, X_1X_2, X_1X_3, \ldots$$

- In the enlarged space the decision boundary is linear, in the original space the decision boundary is non-linear.

Support Vector Machine

- The linear support vector classifier can be represented as:

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

- where $< x, x_i >$ is the *inner product* defined by $\langle x, x_i \rangle = \sum_{j=1}^{p} x_{ij} x_j$
- The inner product is the correlation or more general a measure of similarity.
- It is possible to define the matrix K with all inner products:

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

Support Vector Machine

	mdd	aconscie	neurot
1	0	-1.10	-0.22
2	0	-0.09	-1.42
3	0	-1.27	-0.33
4	1	-0.93	0.55
5	0	-0.59	-0.43

- Need to standardize variables to have equal a-priori importance for all features
- $\langle x_i, x_{i'} \rangle$ is the *inner product* $\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$
- For obs. 1 and 2: $\langle x_1, x_2 \rangle = -1.10 \times -0.09 + -0.22 \times -1.42 = 0.41$
- It is the 'covariance' between the observations for person 1 and 2
- The inner product is the correlation, or more general a measure of similarity

Support Vector Machine: Kernel trick

- K is called the kernel, it is a (symmetric) $n \times n$ matrix of similarities.
- It is possible to generalize the kernel, e.g. a polynomial kernel of degree d:

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

- This is still an $n \times n$ matrix but it gives the similarities in the higher-dimensional space composed of the polynomials of degree d.
- Using this kernel amounts to fitting a linear support vector classifier in the higher-dimensional space. E.g.:

Support Vector Machine: Kernel trick

- For subjects i and i':

$$\begin{aligned} K(x_i, x_{i'}) &= (1 + \langle x_i, x_{i'} \rangle)^d \\ &= (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \end{aligned}$$

- For $p = 2$ and $d = 2$

$$\begin{aligned} K(x_i, x_{i'}) &= (1 + x_{i1}x_{i'1} + x_{i2}x_{i'2})^2 \\ &= 1 + (x_{i1}x_{i'1})^2 + (x_{i2}x_{i'2})^2 \\ &\quad +2x_{i1}x_{i'1} + 2x_{i2}2x_{i'2} + 2x_{i1}x_{i'2}x_{i1}x_{i'2} \end{aligned}$$

Support Vector Machine

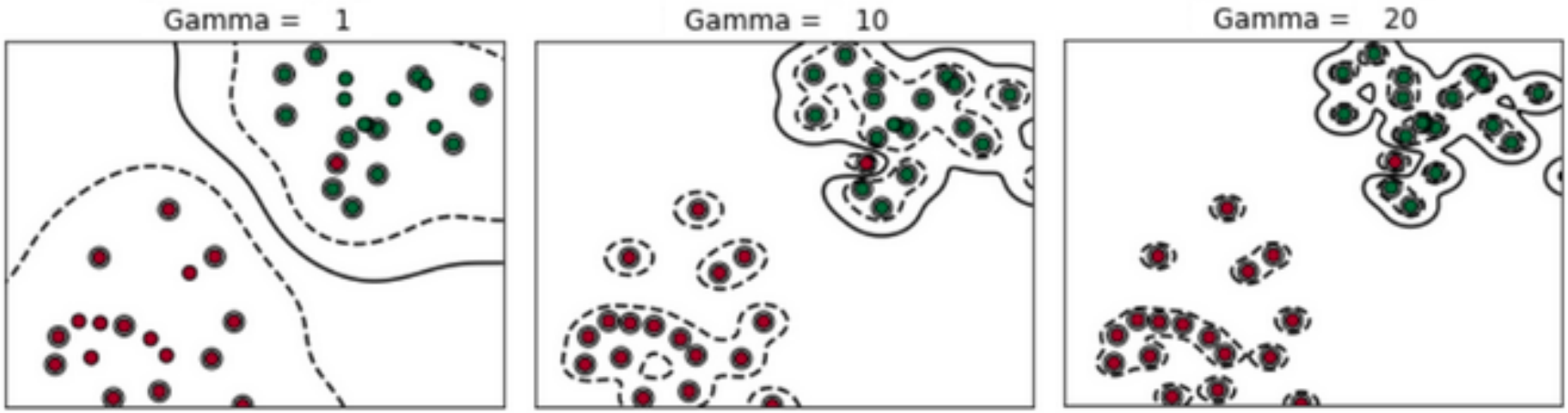
- Another choice of kernel is the *radial basis kernel*:

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

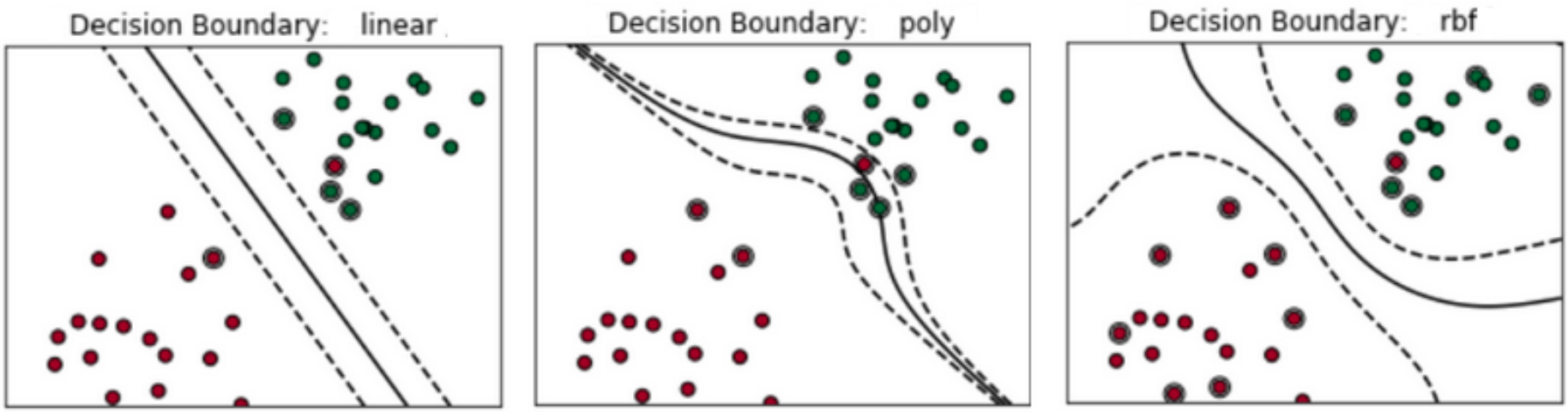
  For this kernel the *implicit* feature space is infinite dimensional.

- γ is a tuning parameter, which defines how far the influence of a single observation reaches:
  - Lower γ values yield a further reach, with very small values yielding a decision boundary close to linear.
  - Higher γ values yield more localized influence, higher flexibility; i.e., lower bias, higher variance

Support Vector Machine



Support Vector Machine



Support Vector Machine

- In order to use kernels properly it is important to scale the variables first (different scalings give different kernels give different solutions).

- Kernels avoid the need to actually transform to an enlarged space; only need to compute the kernel and apply the support vector classifier on this kernel.

- This is known as the kernel trick.