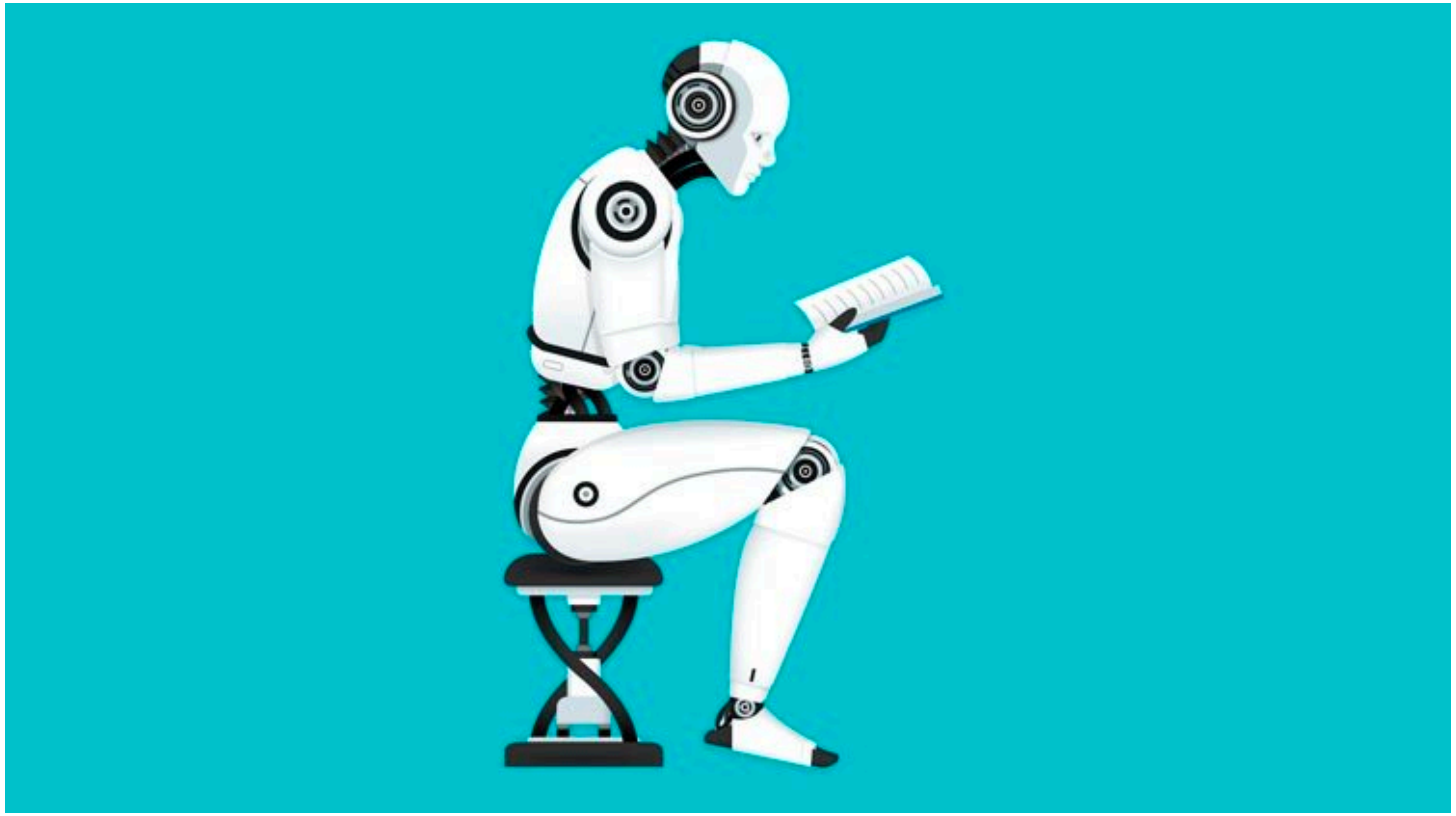
UNIVERSITAT DE BARCELONA



Support Vector Machines

Machine Learning | Enginyeria Informàtica

Santi Seguí | **2020-2021**

Support Vector Machines

- Here we approach the two-class classification problem in a direct way:
- **We try and find a plane that separates the classes in feature space.**
- If we cannot, we get creative in two ways
  - We soften what we mean by “separates”, and
  - We enrich and enlarge the feature space so that separation is possible.

What is a Hyperplane

- A hyperplane in p dimensions is a flat affine subspace of dimension $p - 1$.

- In general the equation for a hyperplane has the form

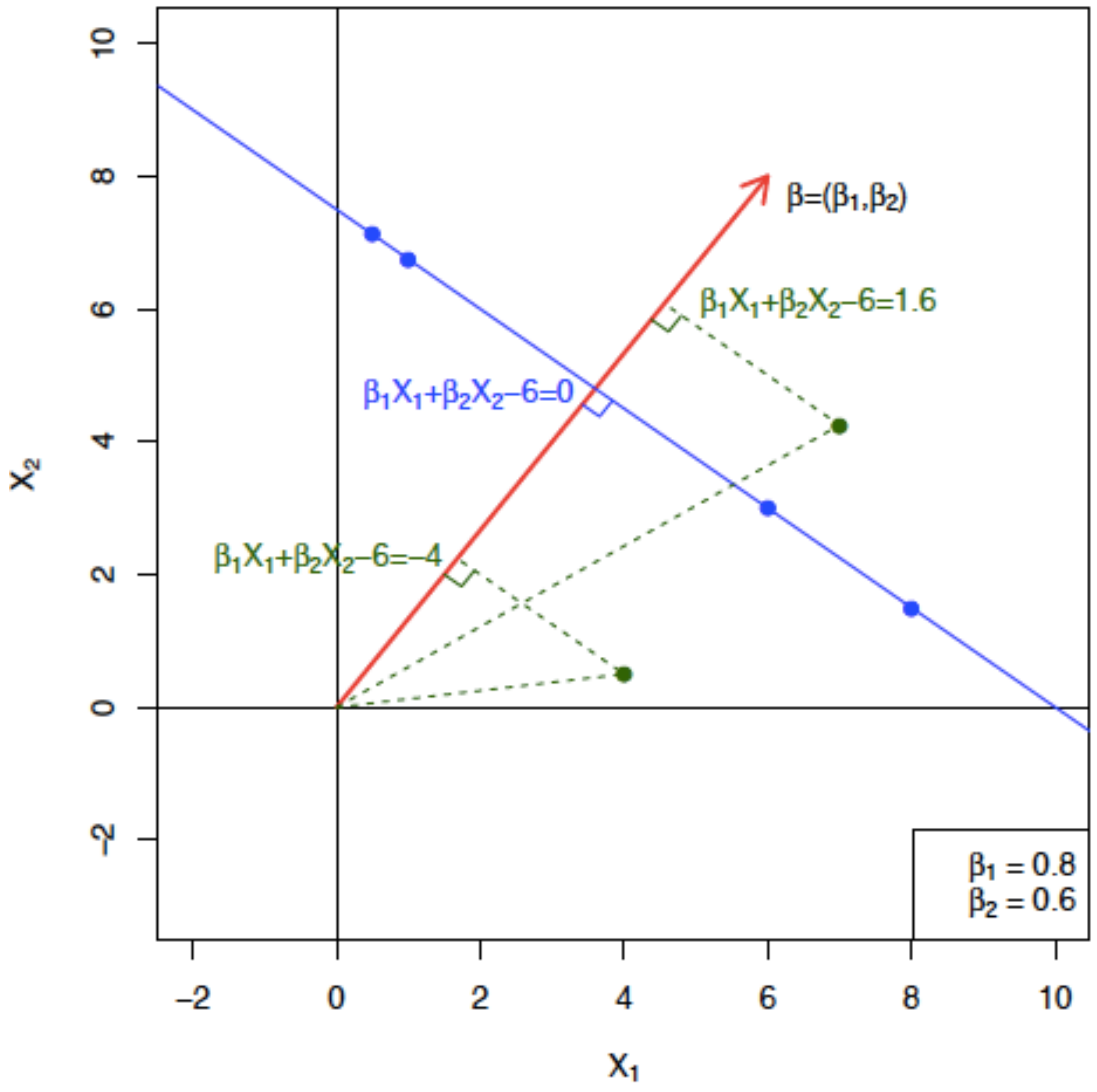
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$

- In $p = 2$ dimensions a hyperplane is a line.

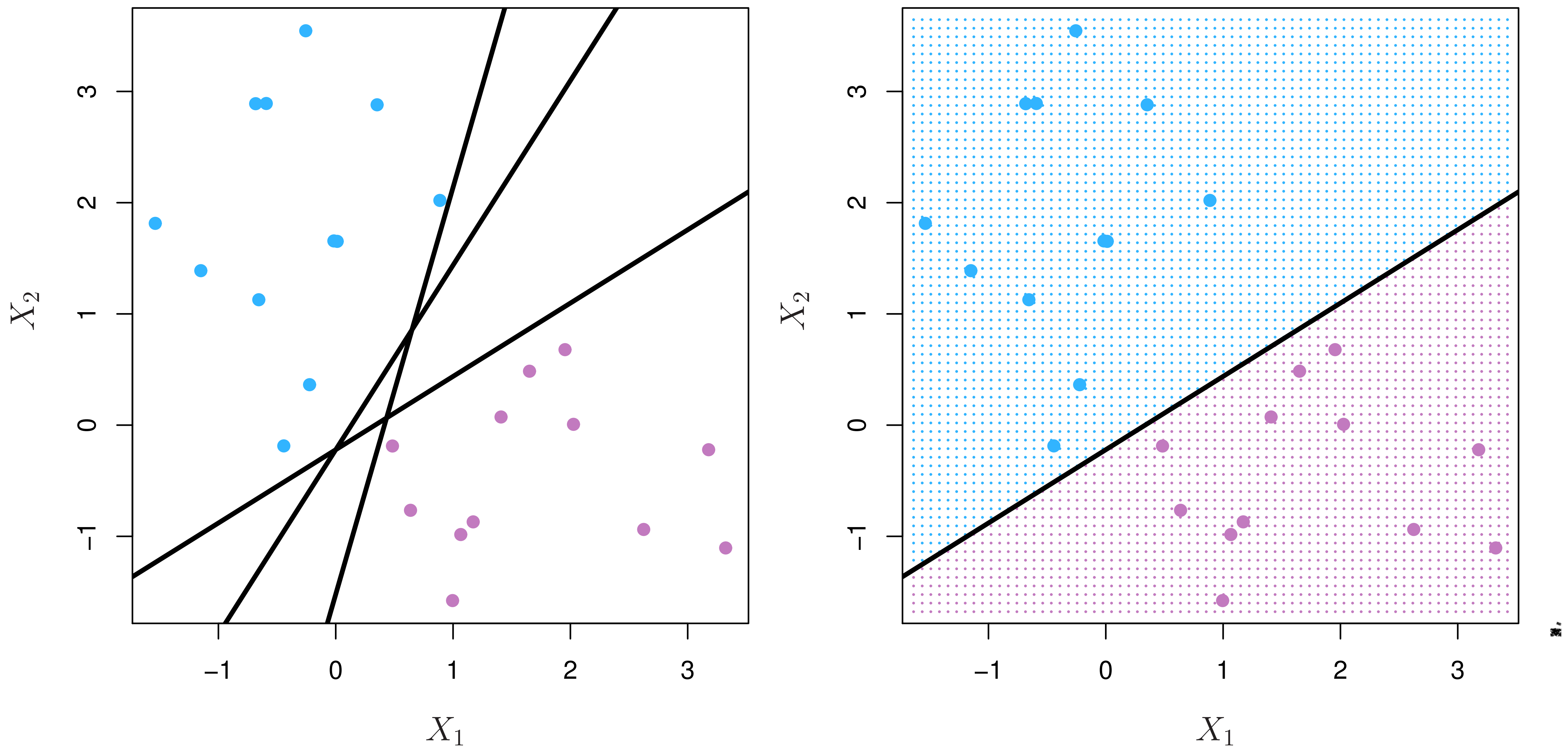
- If $\beta_0 = 0$, the hyperplane goes through the origin, otherwise not.

- The vector $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ is called the normal vector- it points in a direction orthogonal to the surface of a hyperplane.

Hyperplane in 2 Dimensions



Separating Hyperplanes

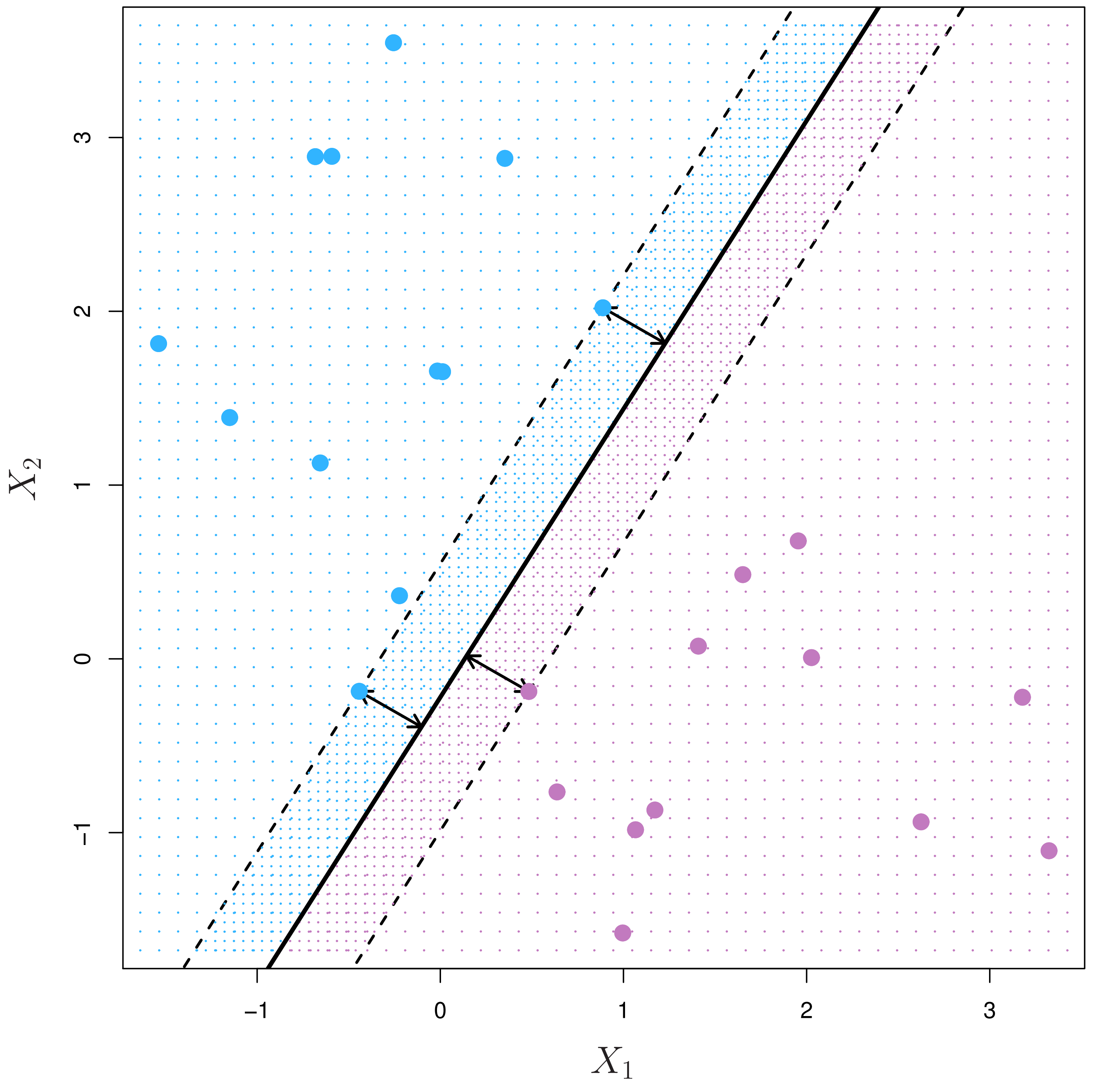


- If $f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$, then $f(X) > 0$ for points on one side of the hyperplane, and $f(X) < 0$ for points on the other.

- If we code the colored points as $Y_i = +1$ for blue, say, and $Y_i = -1$ for magenta, then if $Y_i \cdot f(X_i) > 0$ for all i, $f(X) = 0$ defines a separating hyperplane.

Maximal Margin Classifier

- Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.



Constrained optimization problem

$$\text{maximize}_{\beta_0,\ldots,\beta_p} M$$
$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$
$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$
$$\text{for all} \quad i = 1, \ldots, N$$

Maximal Margin Classifier

- The previous formulation is equivalent to:

$$minimize_{\beta_0,\ldots,\beta_p} ||\beta||^2$$

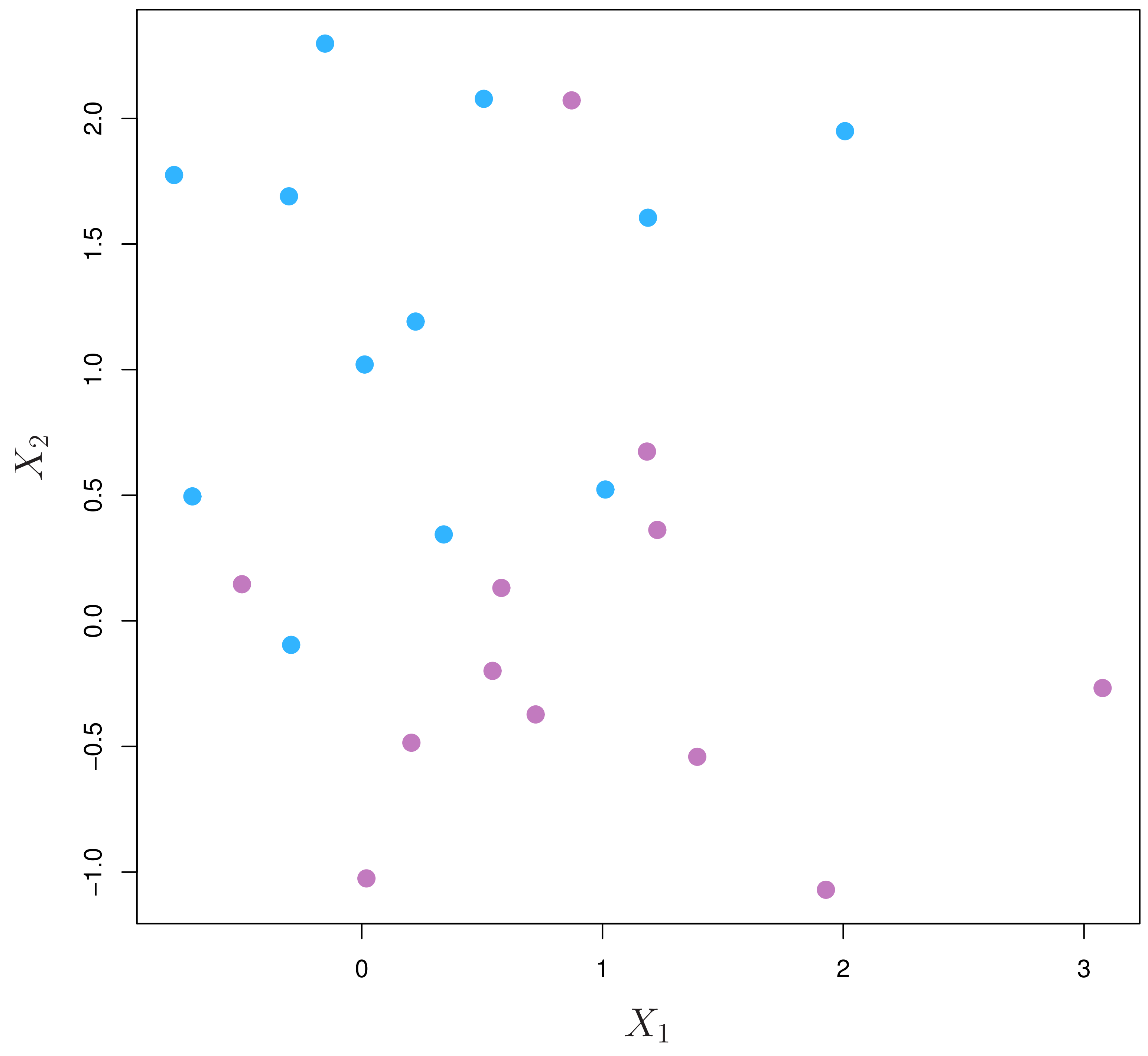
$$subject\ to\ y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq 1$$

$$for\ all\quad i = 1,\ldots,N$$

- This is an easy problem (the objective function is differentiable and convex) that can be solved using standard software.

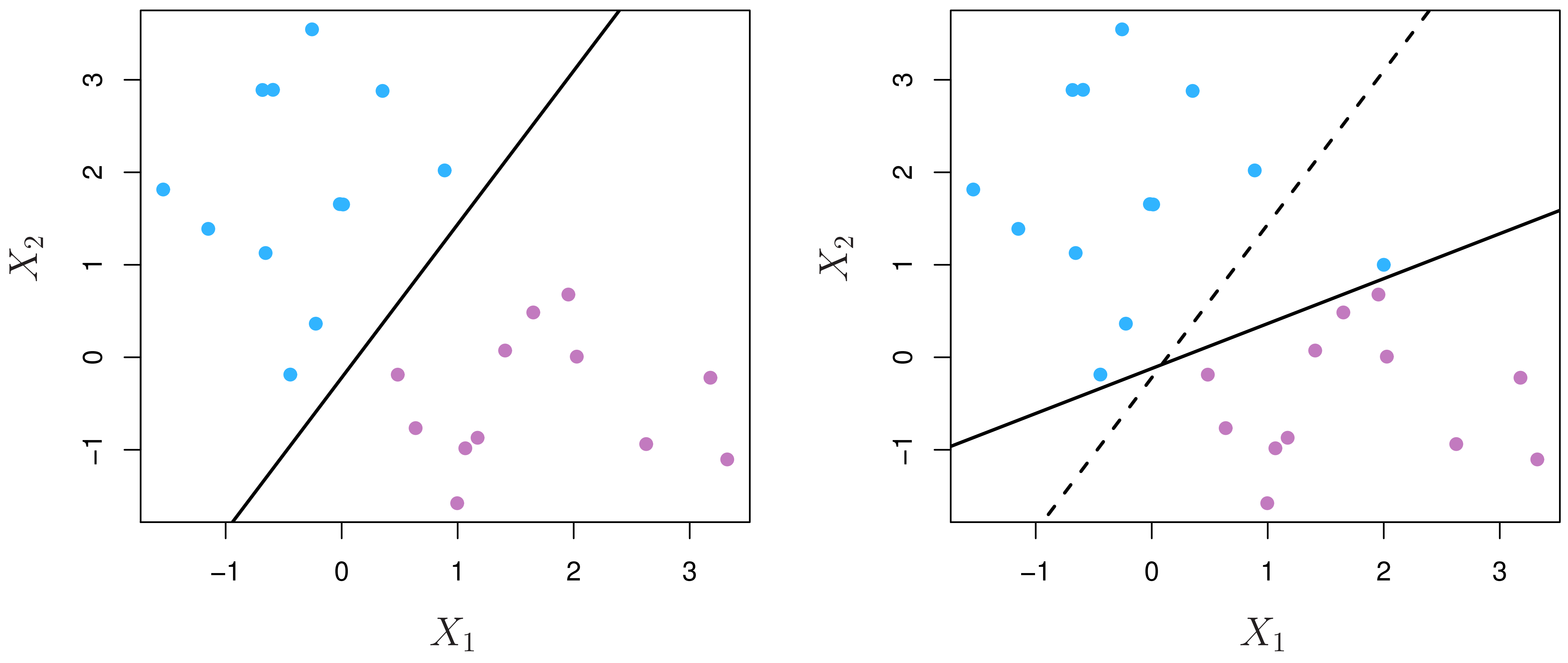
Non-separable Data

- Sometimes the data are not separable by a linea boundary



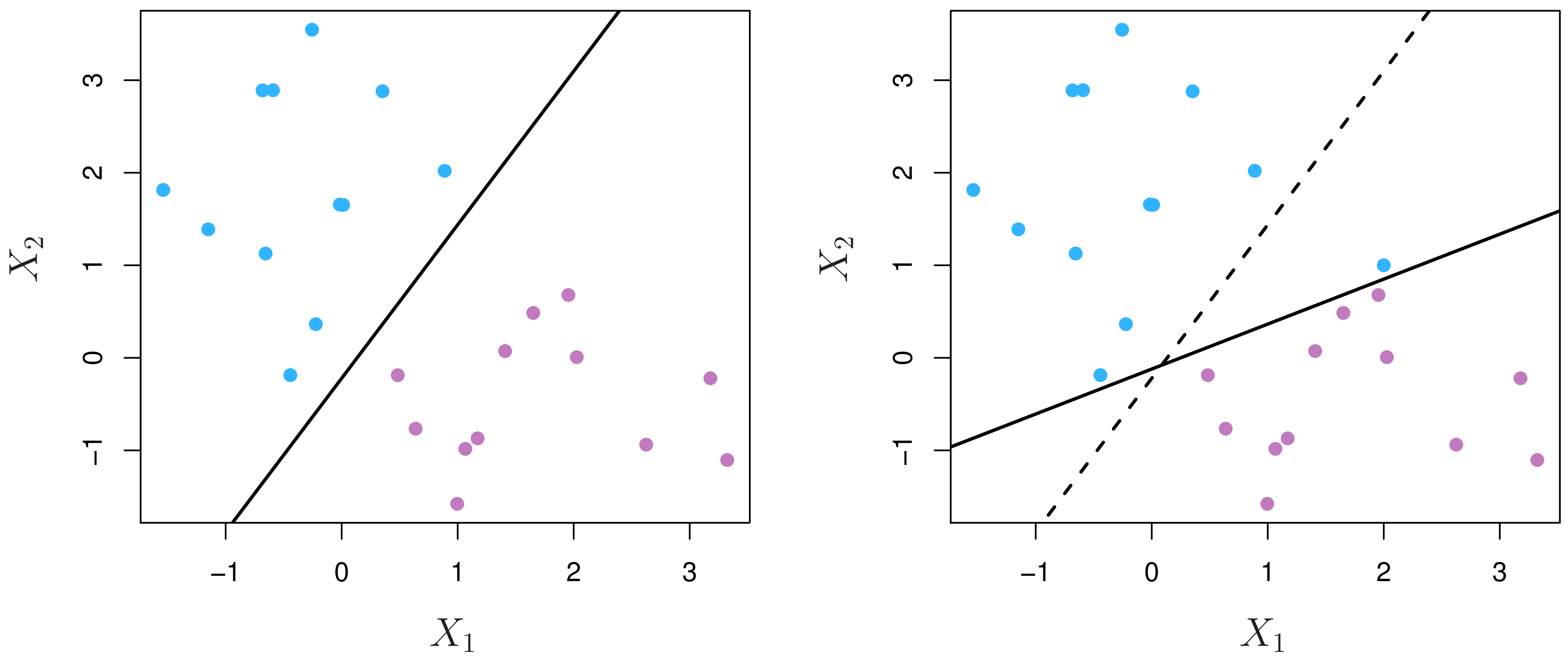
- This is often the case, unless $N < p$

Noisy Data



- Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.

Noisy Data



- Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier. The **Support Vector Classifier** maximizes a **soft margin**.

The case of not separable data

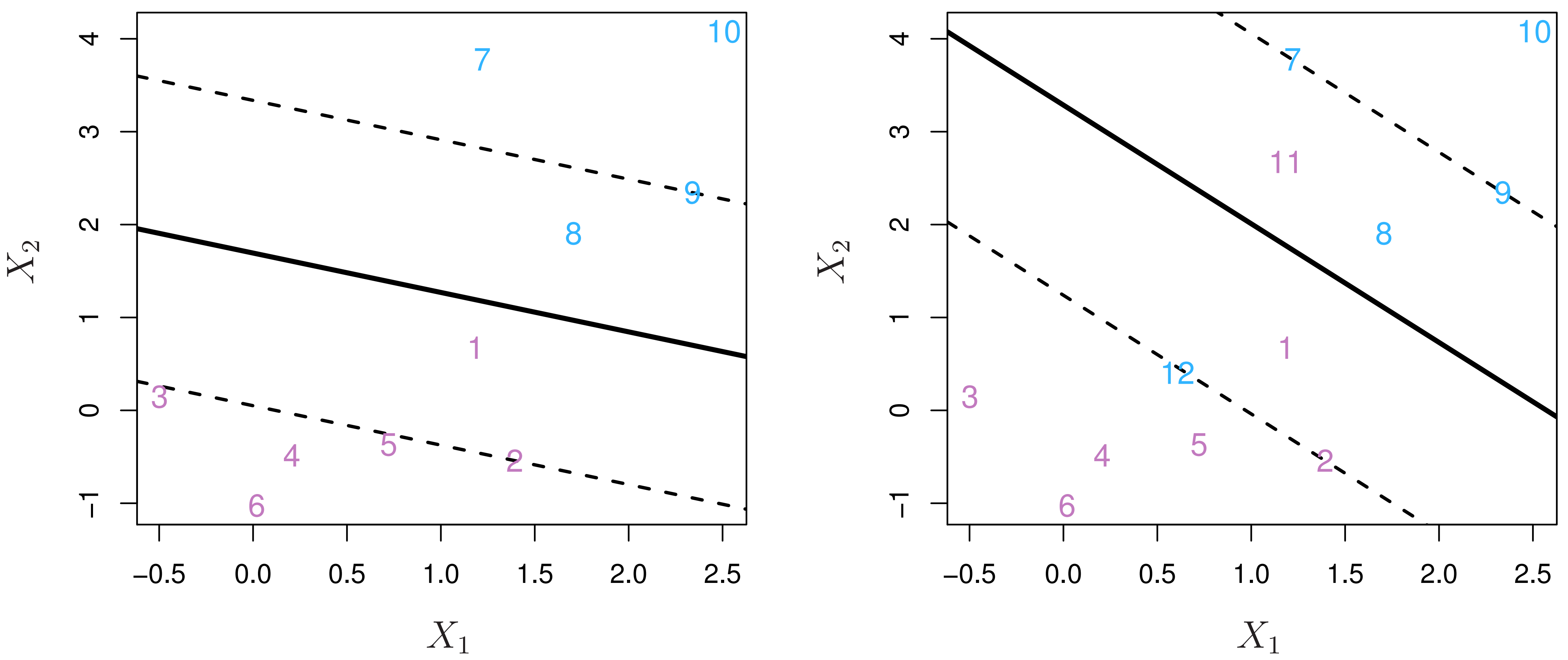
- In general, if data is non-separable this means that for at least one data point, say x_i , the term $y_i(\mathrm{w} \cdot \mathrm{x}_i)$ will fail to exceed the threshold value 1 no matter what w is set to.

- Although we might perhaps like to minimize the number of misclassified points, this is an NP-hard problem. Instead, to handle the case of points on the wrong side of the fence, SVMs introduce slack variables.

- These variables allow all the constraints to be satisfied.

- A corresponding slack variable penalty term is added to the objective, such that the more the slack variables are relied upon, the worse the value of the objective.

Support Vector Classifier



$$maximize_{\beta_0,\ldots,\beta_p,e_1,\ldots,e_n} M \quad subject\ to \sum_{j=1}^{p} \beta_j^2 = 1,$$

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

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

The case of non separable data

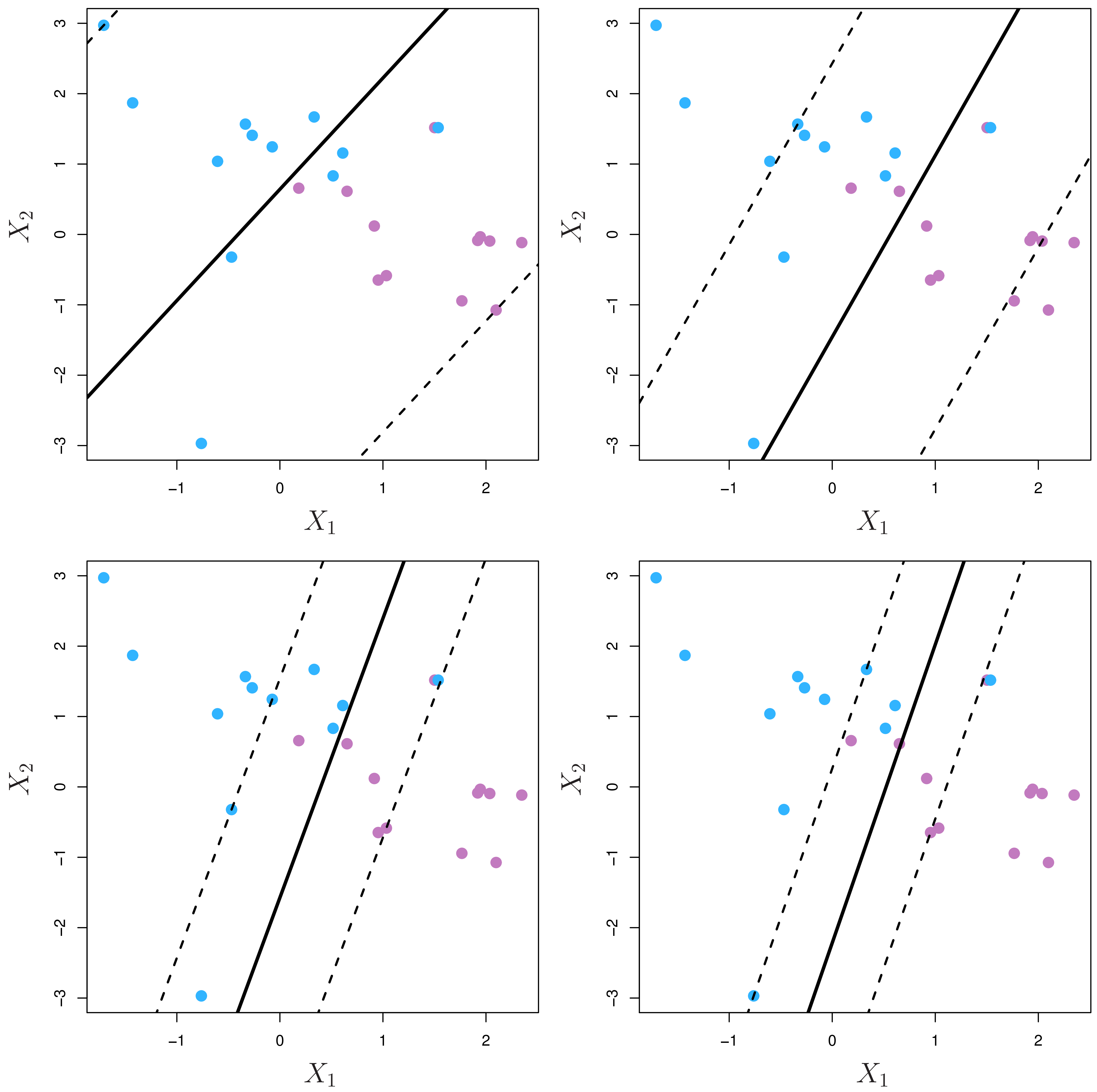
- This SVM variant takes a parameter C that determines how hard points violating the constraint should be penalized. This parameter appears in the objective function of the problem, which now is formulated as:

$$min_{\mathrm{w},\xi} \frac{1}{2}\|\mathrm{w}\| + \frac{C}{N}\sum_{i} \xi$$

$$s.t.\ y_i(\mathrm{w} \cdot \mathrm{x}_i) \geq 1 - \xi_i, \xi \geq 0, \forall i \in \{1,\ldots,N\}$$

- Large C means high penalty, and in the limit $C \rightarrow \infty$ we obtain the separable case.

C is a regularization parameter

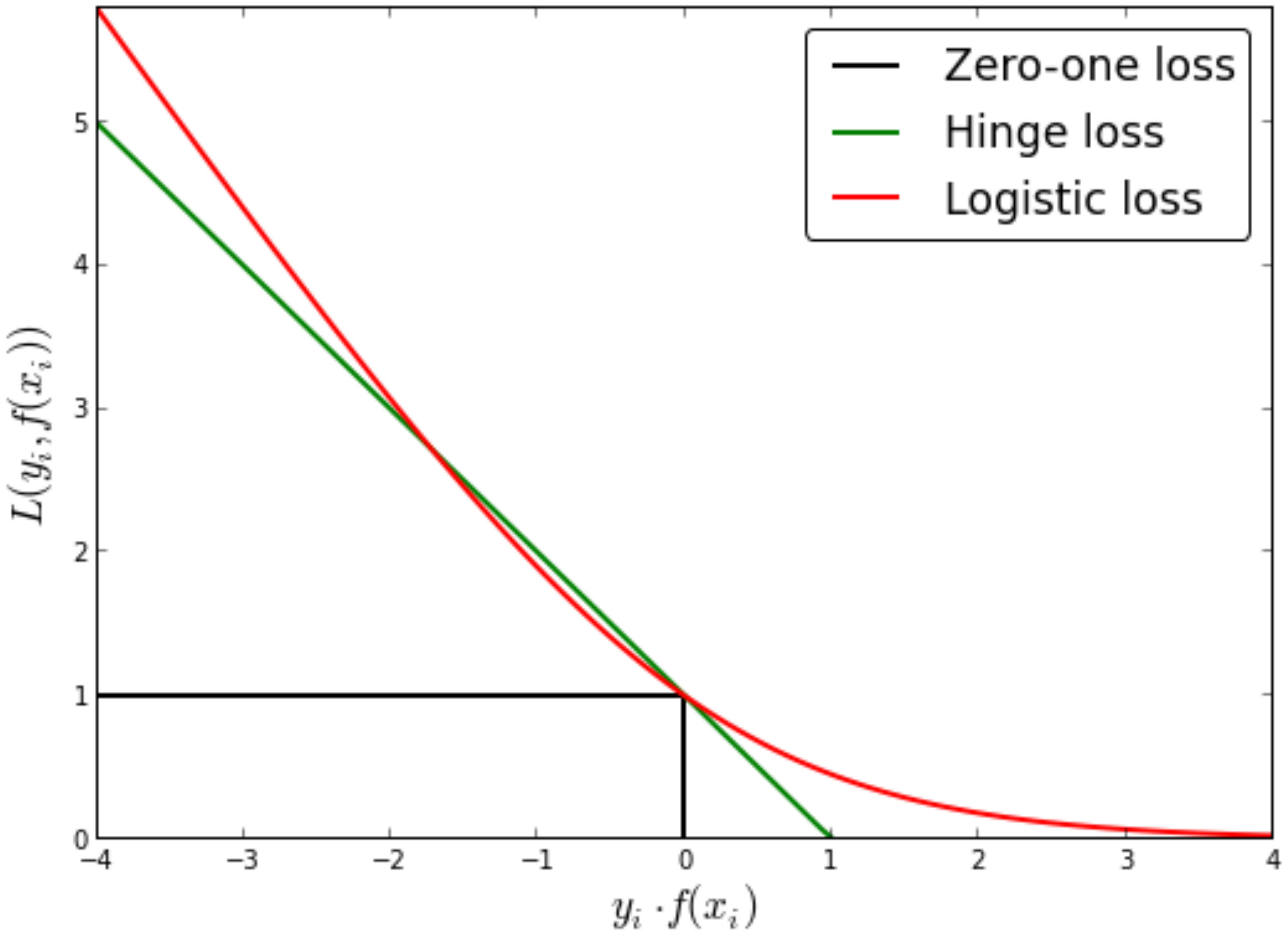


The case of non separable data

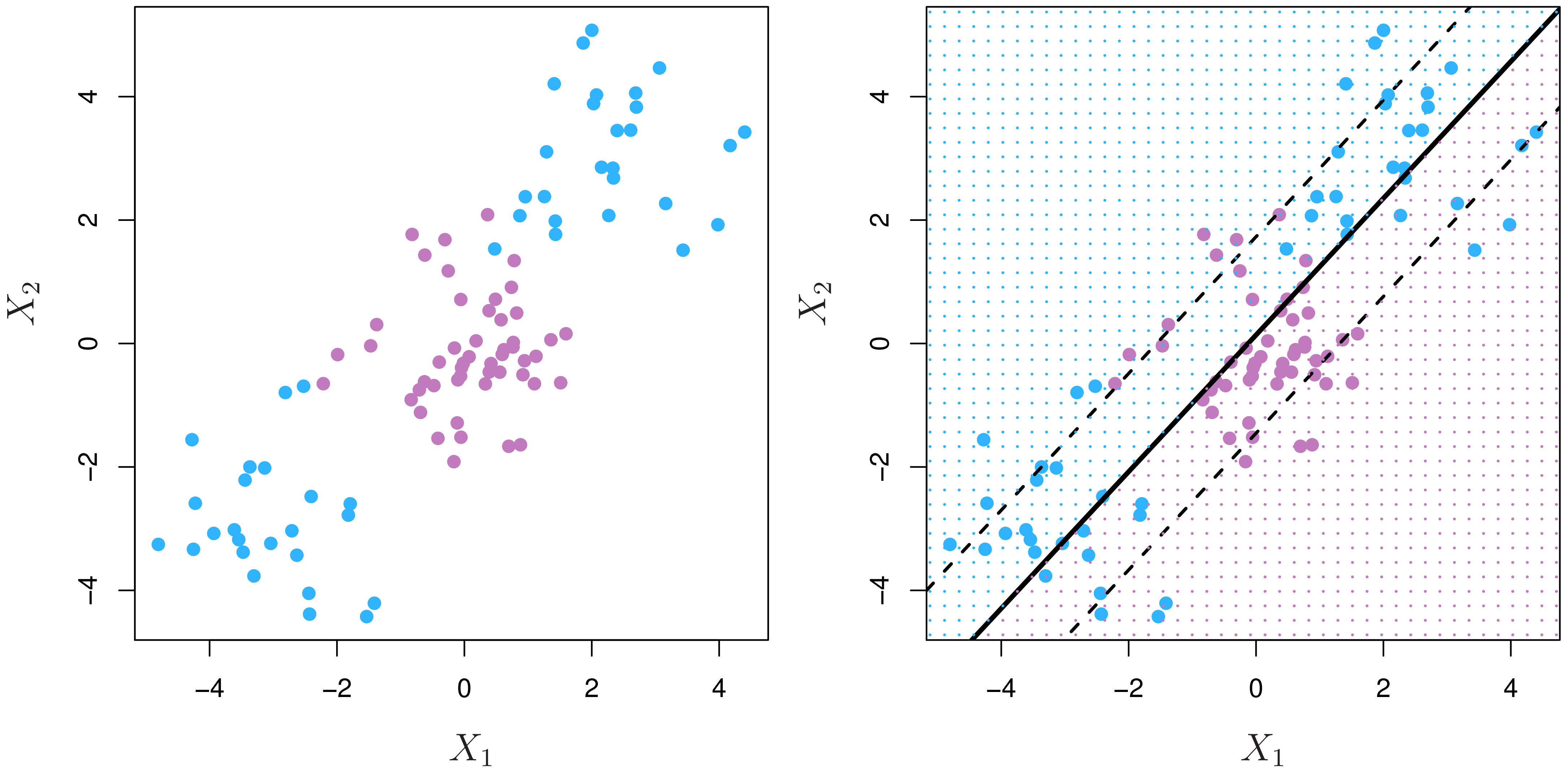
- Observe that if $y_j(\mathrm{w} \cdot \mathrm{x}_j) \geq 1$, then $\xi_j = 0$ and in this case there is no contribution to the penalty term, but if margin$y_j(\mathrm{w} \cdot \mathrm{x}_j) < 1$, $\xi_j > 0$ and the penalty terms increases $\frac{C}{N}\xi_j$:

$$\xi_j = max(0,1 - y_i(\mathrm{w} \cdot \mathrm{x}_j))$$

Loss Functions



Linear boundary can fail



What to do?

Feature Expansion

- Enlarge the space of features by including transformations; e.g. $X_1, X_1^2, X_1X_2, X_1X_2^2, \ldots$ 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.

Feature Expansion

- Enlarge the space of features by including transformations; e.g. $X_1, X_1^2, X_1X_2, X_1X_2^2, \ldots$ 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.

- Example: Suppose we use $(X_1; X_2; X_1^2, X_2^2, X_1X_2)$ instead of just $(X_1; X_2)$. Then the decision boundary would be of the form

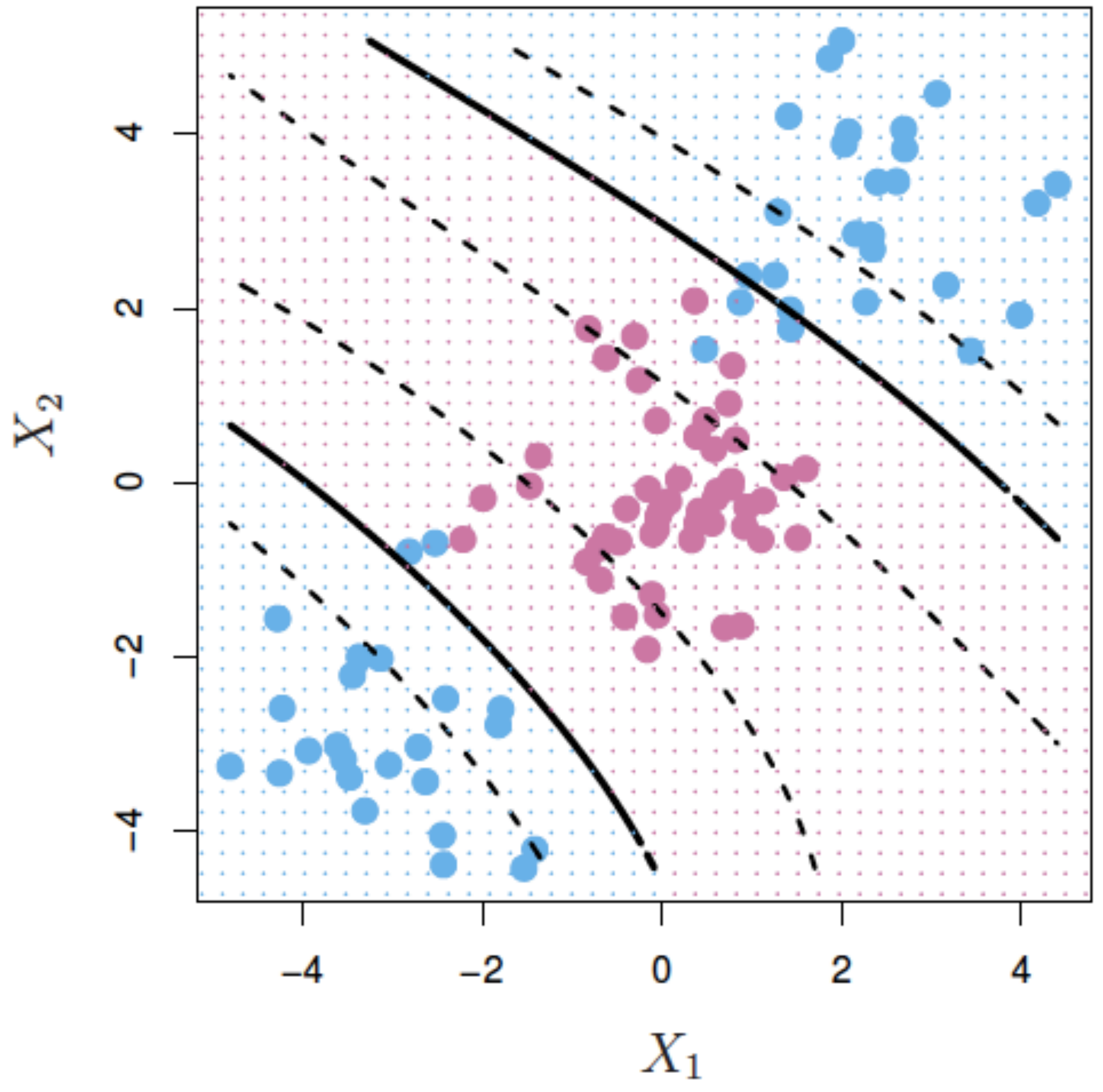
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$$

- This leads to nolinear decision boundares in the original space.

Feature Expansion

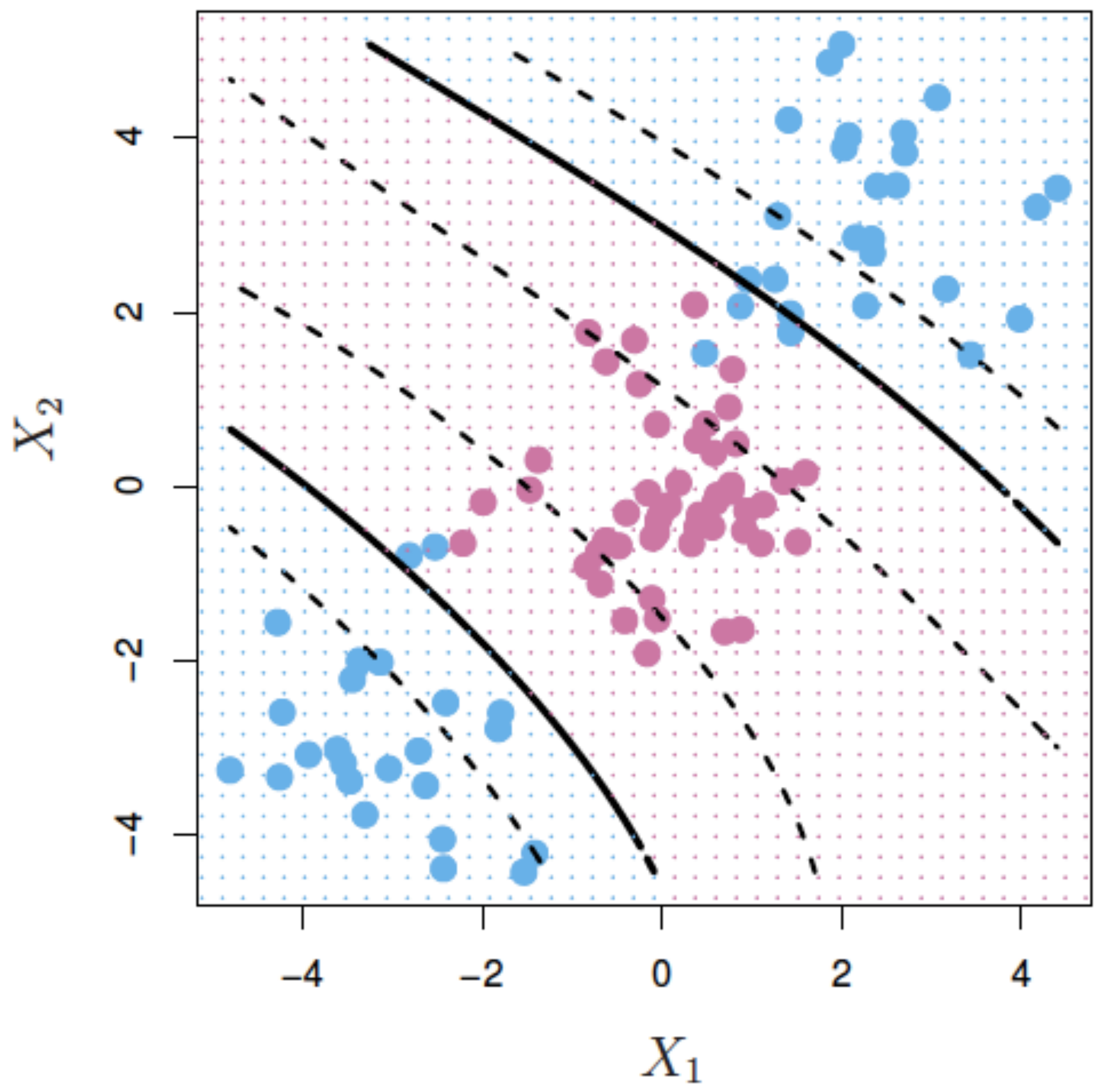
- Here we use a basis expansion of cubic polynomials. From 2 variables to 9.

- The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space.



Feature Expansion

- Here we use a basis expansion of cubic polynomials. From 2 variables to 9.
- The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space.



$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 + \beta_6 X_1^3 + \beta_7 X_2^3 + \beta_8 X_1^2 X_2 + \beta_9 X_1 X_2^2 = 0$$

Non Linearities and Kernels

- Polynomials (especially high-dimensional ones) get wild rather fast.
- There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers - through the use of **kernels**.
- Before we discuss these, we must understand the role of inner products in support-vector classifiers.

Inner products and support vectors

$$\langle x_i, x_i' \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

$$\langle x_i, x_i' \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

Inner products and support vectors

$$\langle x_i, x_i' \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

The linear support vector classifier can be represented as:

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

i.e. it has n <u>**parameteres**</u>

To estimate the parameters $\alpha_1, \ldots, \alpha_n$ and β_0 , all we need are the $\binom{n}{k}$ inner products $= \langle x_i, x_i' \rangle$ between all the pairs of training observations.

Inner products and support vectors

- I It turns out that most of the $\hat{\alpha}_i$ can be zero:
  - $f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i \langle x, x_i \rangle$
  - where S is the support set of indexes i such that $\hat{\alpha}_i > 0$

Primal vs Dual Formulation

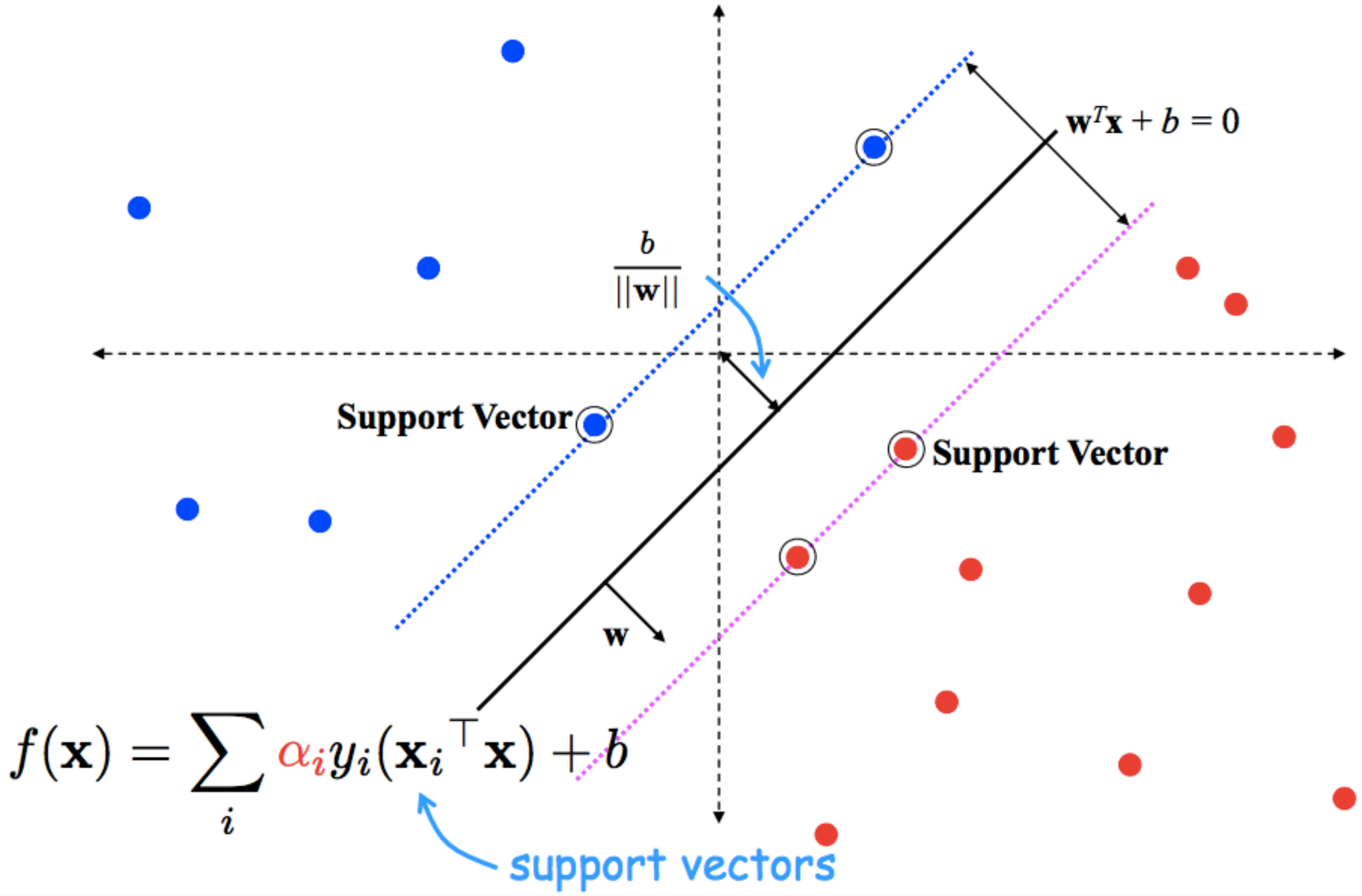
- Primal version of the classifier:

$$f(x) = w^T x + b$$

- Dual version of the classifier:

$$f(x) = \sum_{i}^{N} \alpha_i y_i (x_i^T x) + b$$

Dual Version of SVM



Dual Formulation

- We can write the dual form of the problem:

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

under the constraints.

- This problem can also be easily solved using standard optimization software.

- β can be computed from α terms: $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$.

The Kernel trick

- The original SVM algorithm, proposed by Vladimir Vapnik in 1963, was a linear classifier, but there is a way (**the kernel trick**) to design non-linear classifiers.

- The trick is based on the observation that all data terms are exclusively used to compute dot products:

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

- This fact suggest a way to compute SVM solutions in higher dimensional spaces: to find a function $k(x_i, x_j)$, called the kernel, that corresponds to the dot product of x_i and x_j in such a space.

The Kernel trick

- In this case, the problem can be written as:

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i^T x_j)$$

and the optimization problem does not change.

Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SVM classifier.

- Some special kernel functions can do this for us. E.g

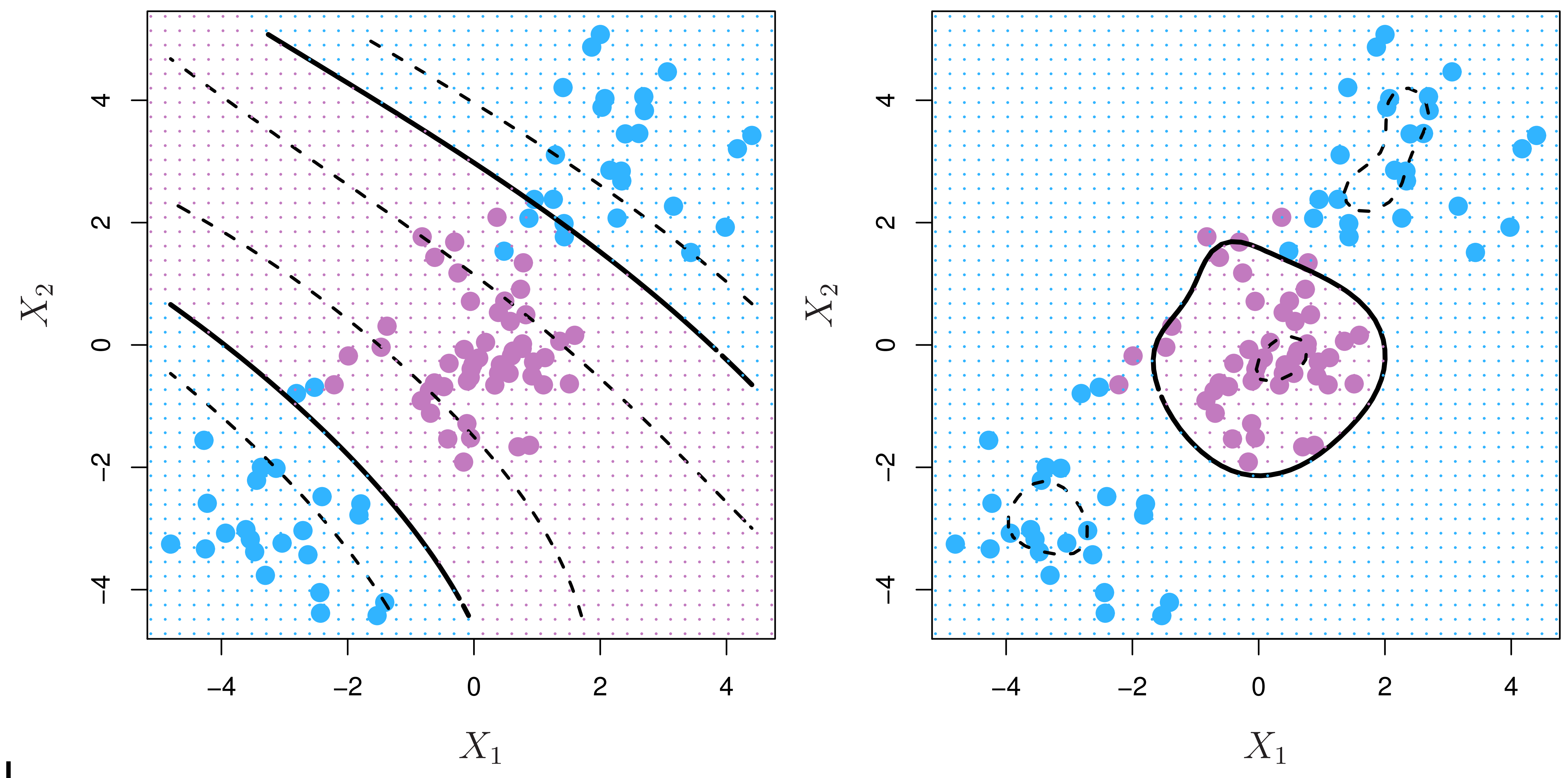
$$K(x_i, x_i') = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

- computes the inner-products needed for d dimensional polynomials $\binom{p+d}{d}$ basis functions!

Radial kernel

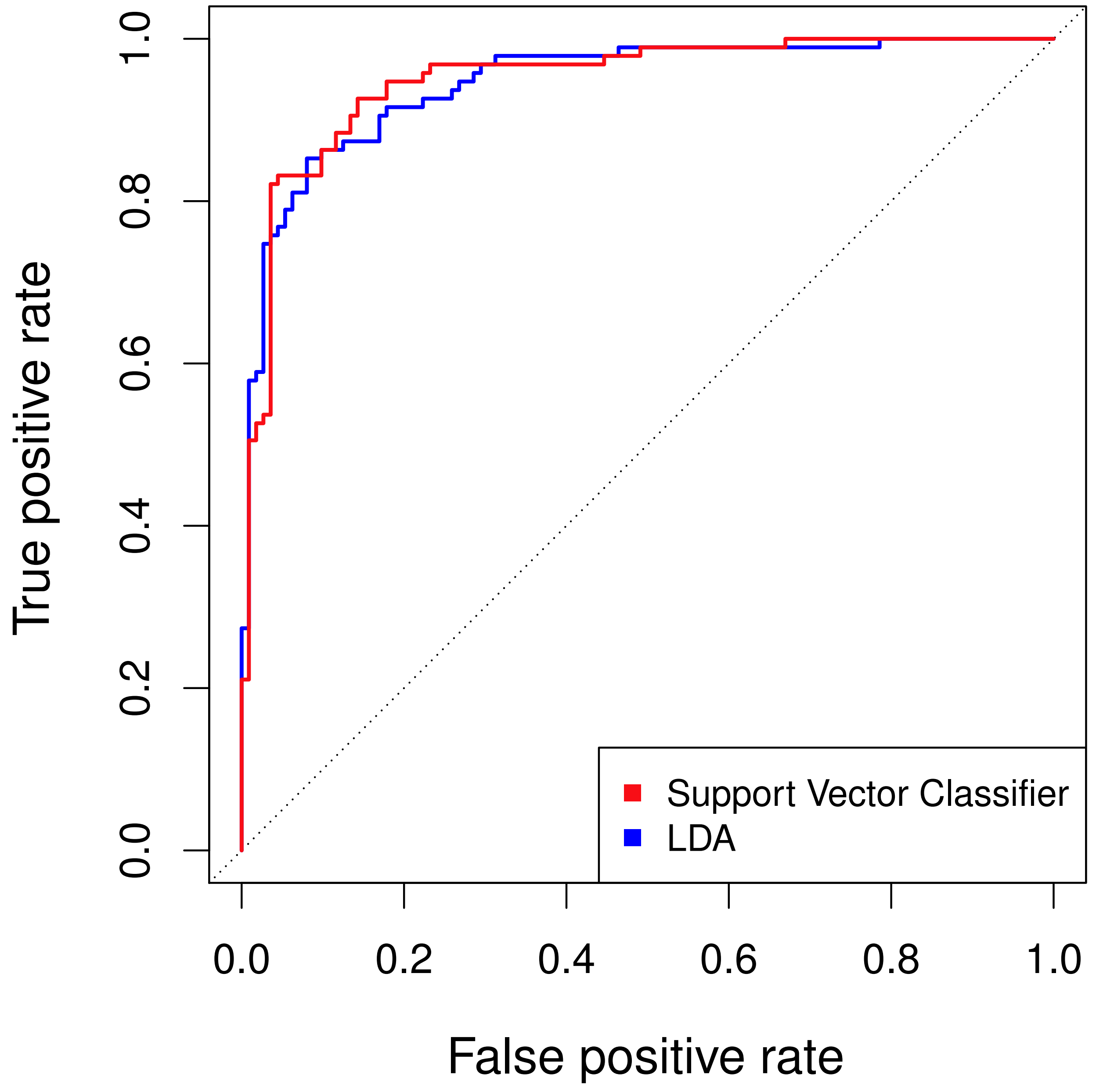
$$K(x_i, x_i') = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

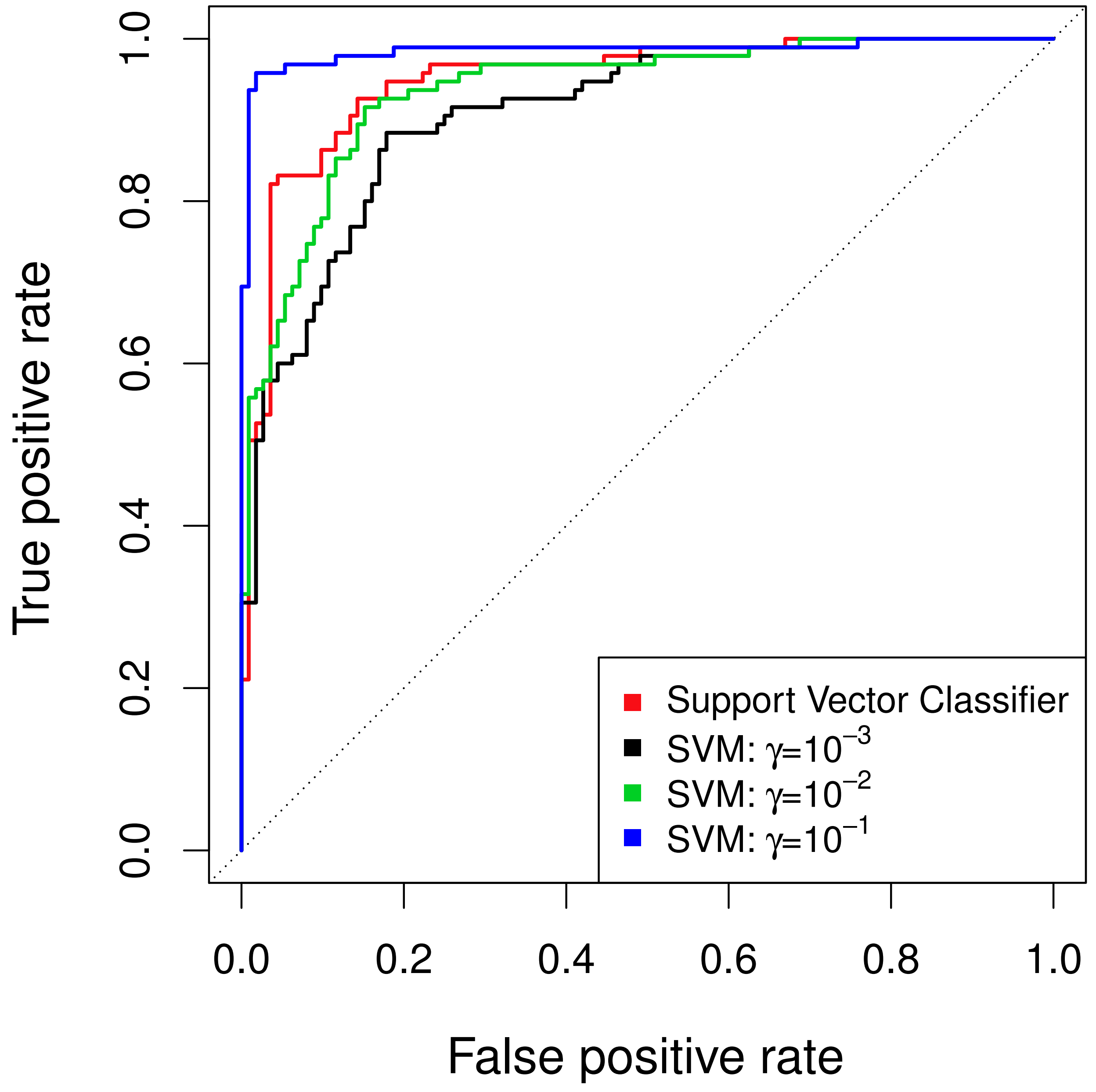
$$f(x) = \beta_0 + \sum_{i \in S} \hat{\alpha}_i K(x, x_i)$$



- Implicit feature space; very high dimensional.
- Controls variance by squashing down most dimensions severely

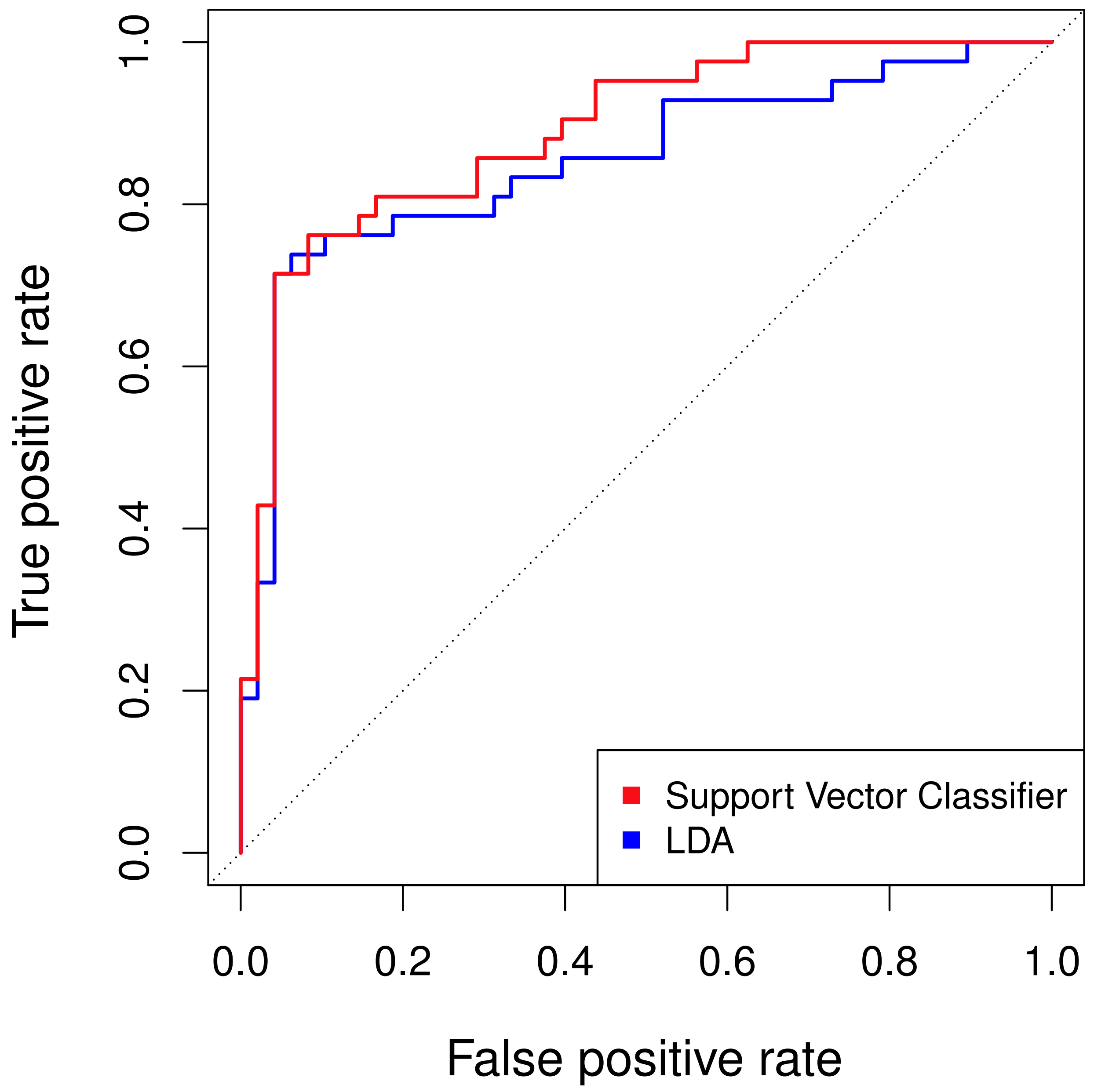
Example: Heart data

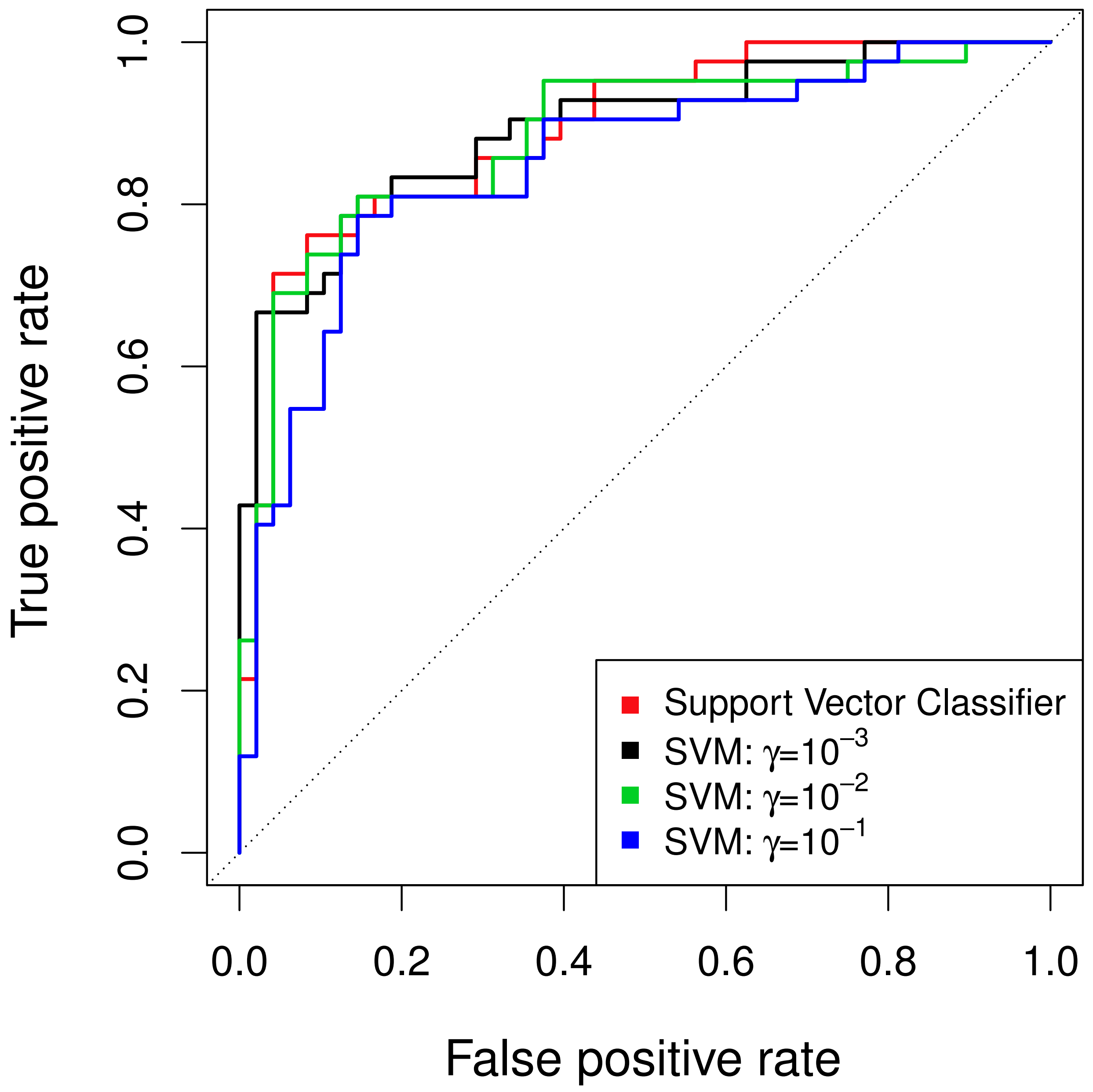




- ROC curve is obtained by changing the threshold 0 to threshold t in $\hat{f}(X) > t$, and recording false positive and true positive rates as t varies. Here we see ROC curves on training data.

Example continued: Heart Test Data





SVMs: more than 2 classes?

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

SVMs: more than 2 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 $\hat{f}_k(x), k = 1,\ldots,K$; each class versus the rest. Classify x^* to the class for which $\hat{f}_k(x^*)$ is largest.

  - **OVO**: One versus One. Fit all $\binom{K}{2}$ pairwise classifiers $\hat{f}_{kl}(x)$. Classify x^* to the class that wins the most pairwise competitions.

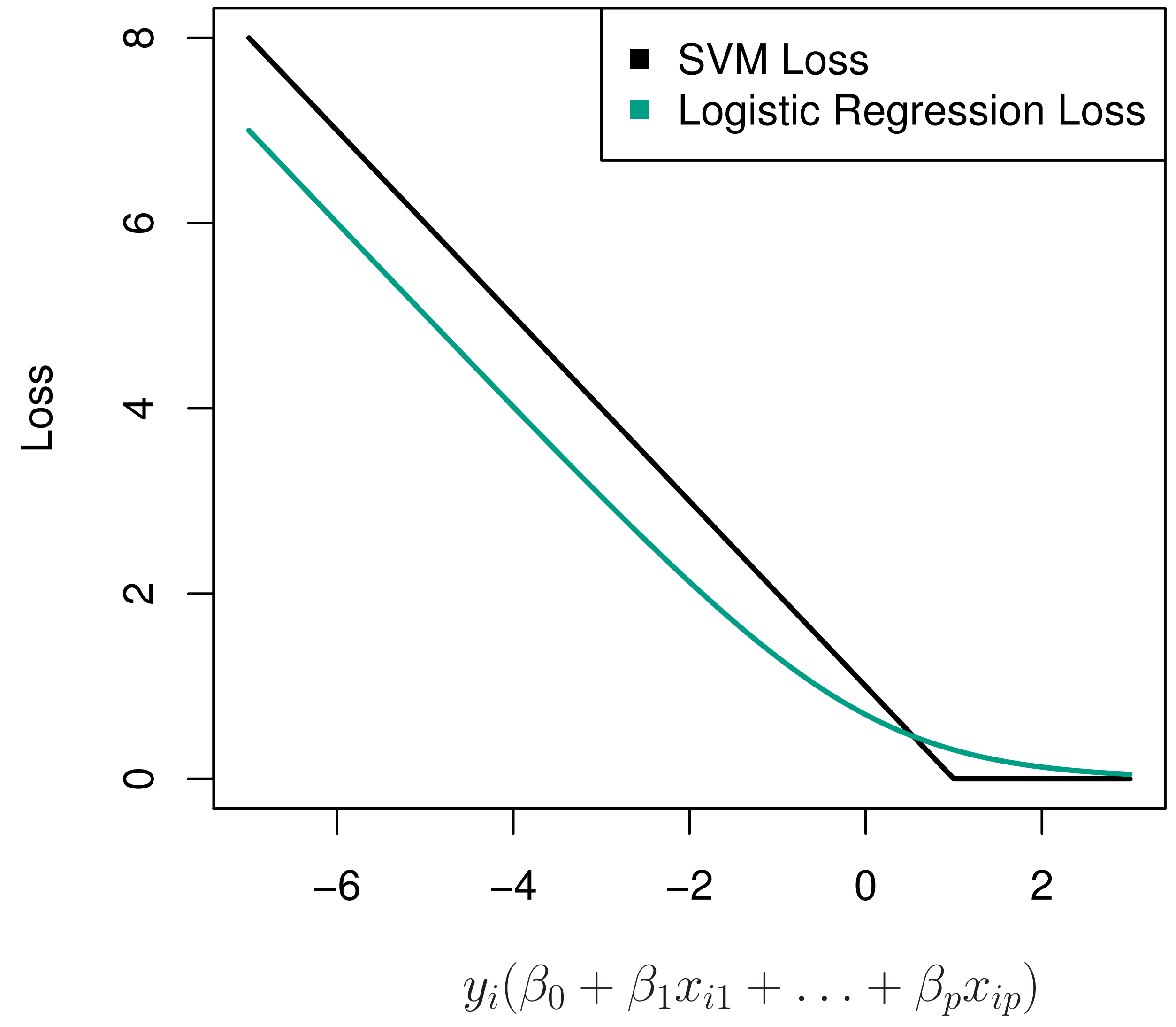
- Which to choose? If K is not too large, use OVO.

Support Vector versus Logistic Regression?

- With $f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$ can rephrase support-vector classifier optimization as

$$minimize_{\beta_0, \beta_1, \ldots, \beta_p} \left\{ \sum_{i=1}^{n} max[0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

Support Vector versus Logistic Regression?



- This has the form loss plus penalty.
- The loss is known as the hinge loss. Very similar to “loss” in logistic regression (negative log-likelihood).