Support Vector Machines

Separable Hyperplanes

- Imagine a situation where you have a two class classification problem with two predictors X_1 and X_2.
- Suppose that the two classes are “linearly separable” i.e. one can draw a straight line in which all points on one side belong to the first class and points on the other side to the second class.
- Then a natural approach is to find the straight line that gives the biggest separation between the classes i.e. the points are as far from the line as possible
- This is the basic idea of a support vector classifier.

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

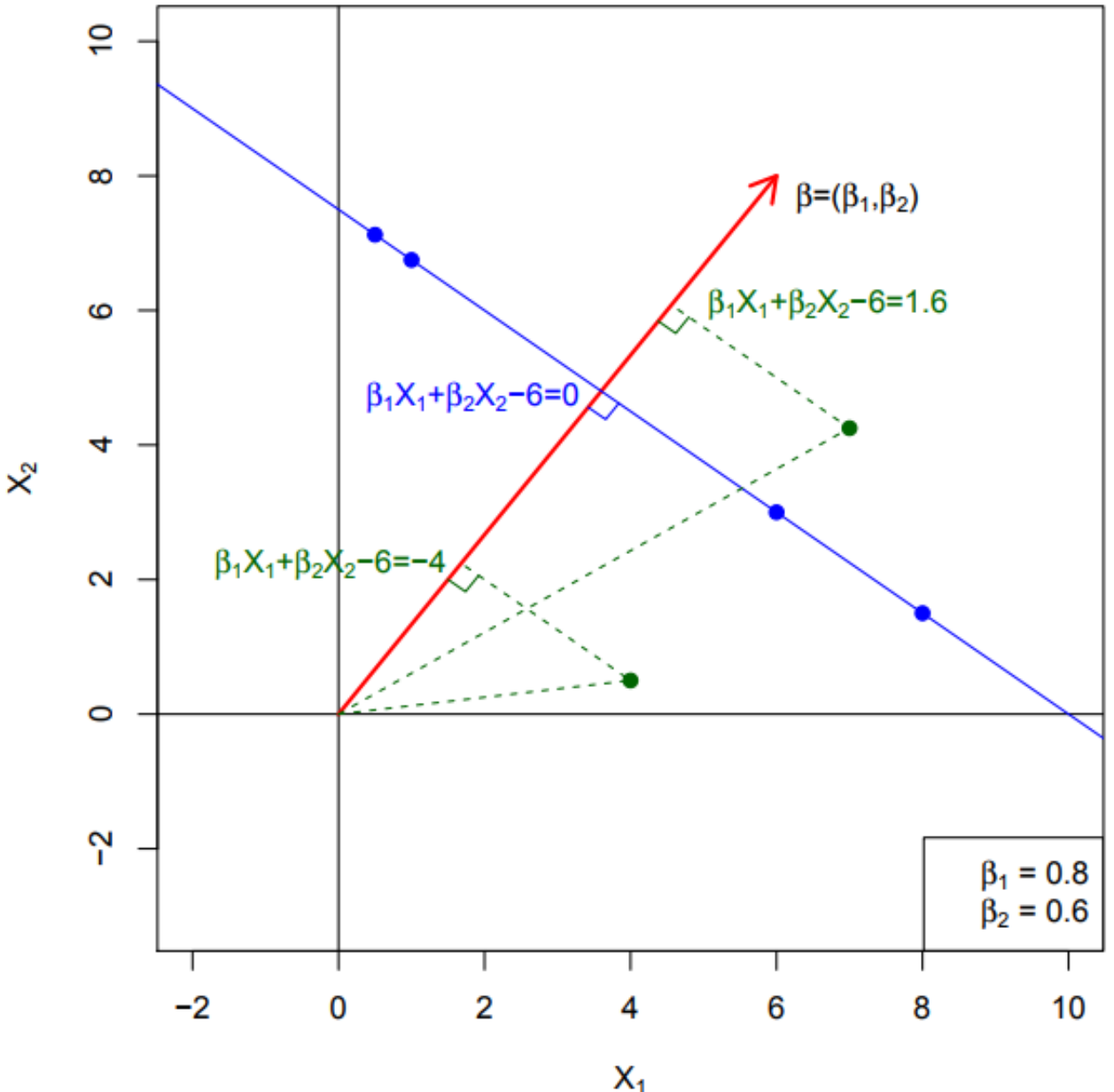
$f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$

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

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

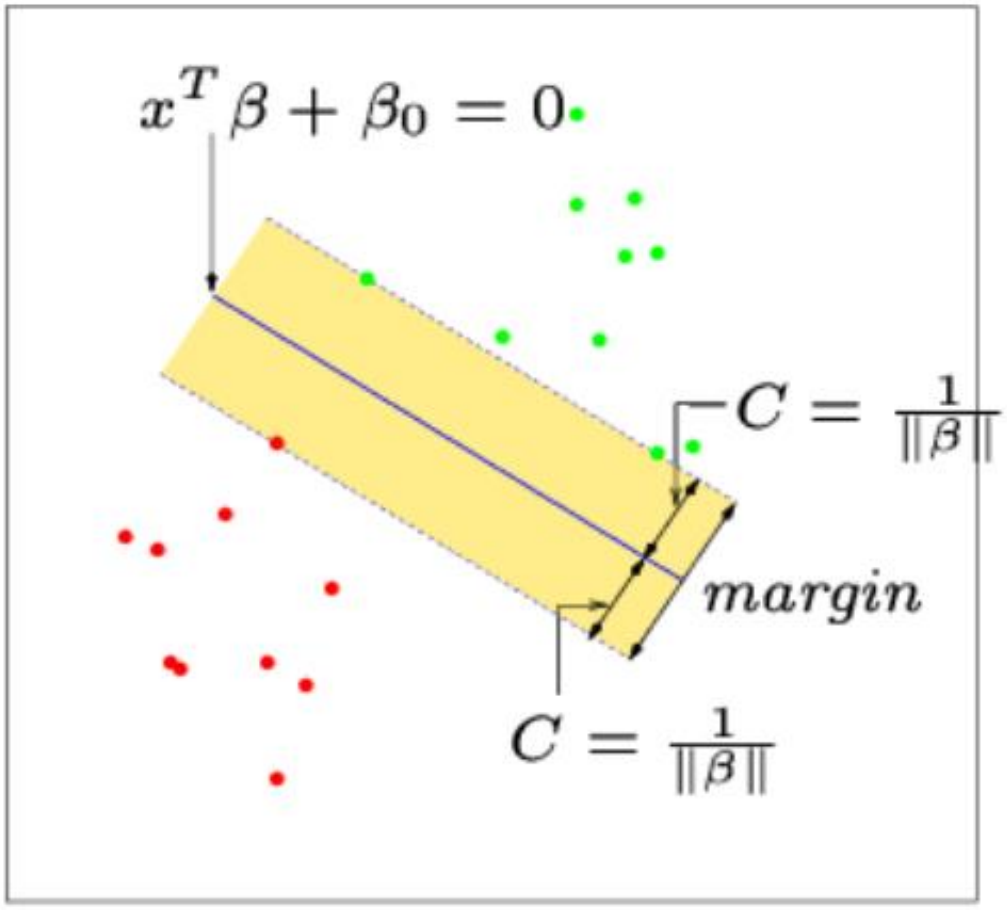
The vector $\beta = (\beta_1, \beta_2, \ldots \beta_p)$ is called the normal vector — it points in a direction orthogonal to the surface of a hyperplane.

Hyperplane in 2 Dimensions

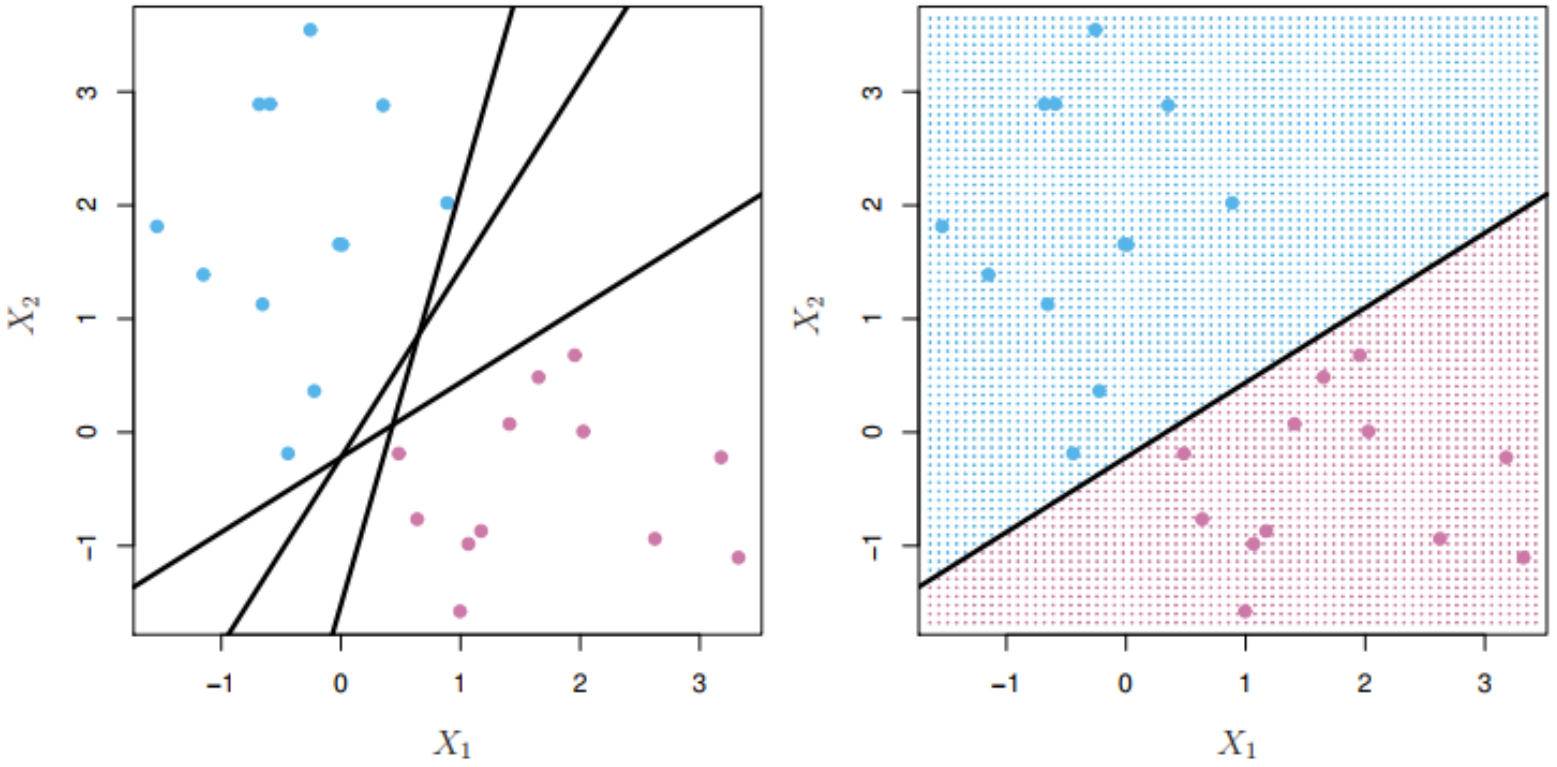


Another Picture

- C is the minimum perpendicular distance between each point and the separating line.
- We find the line which maximizes C.
- This line is called the "optimal separating hyperplane"
- The classification of a point depends on which side of the line it falls on.



Separating Hyperplanes

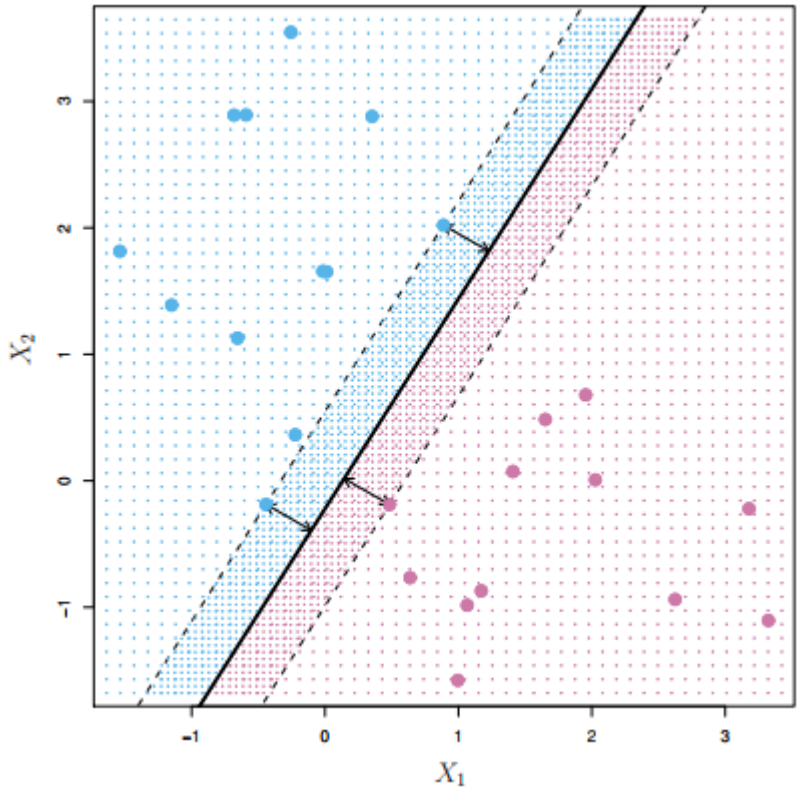


If $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ then $f(X) > 0$ for points on one side of the hyperplane, and $f(X) < 0$ for points on the other.

If we code the colored points as $Y_i = +1$ for blue, say, and $Y_i = -1$ for mauve, 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

$$\text{maximize} \quad M \quad \text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$\beta_0, \beta_1, \ldots, \beta_p$$

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

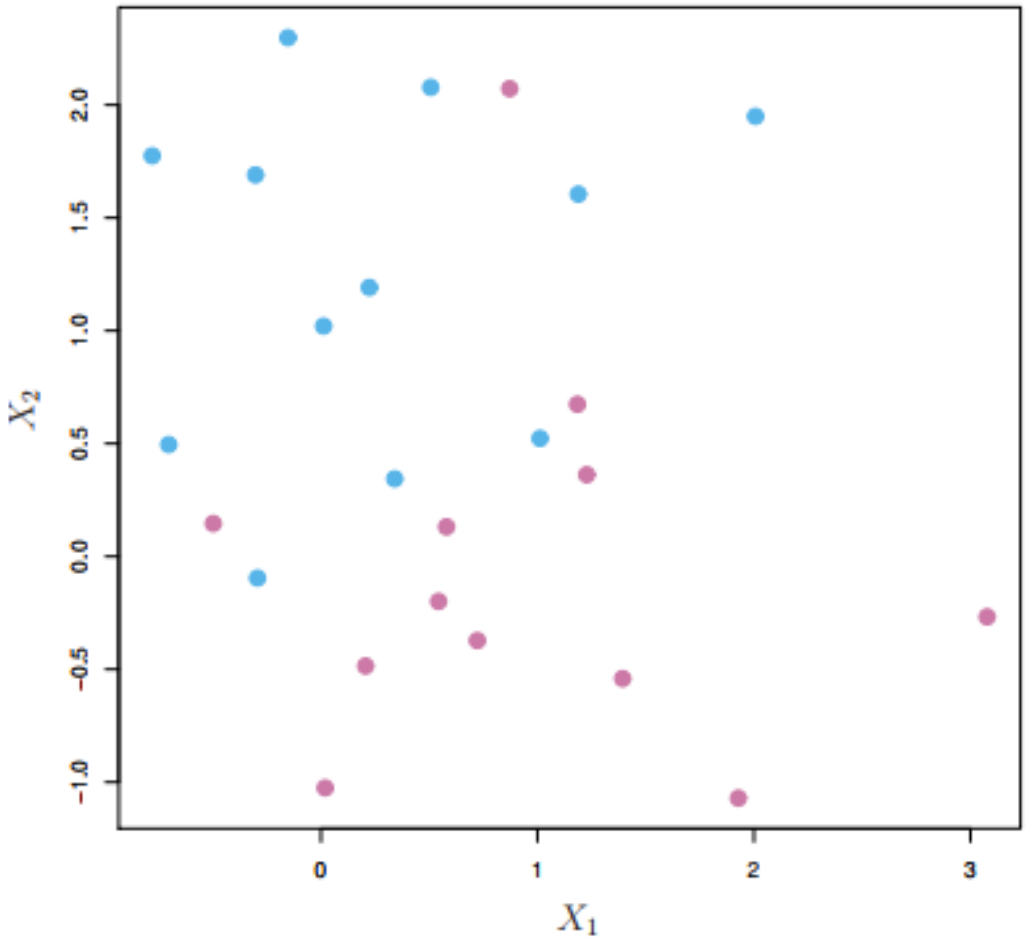
for all $i = 1, \ldots, N$.

This can be rephrased as a convex quadratic program, and solved efficiently. The function svm() in package e1071 solves this problem efficiently

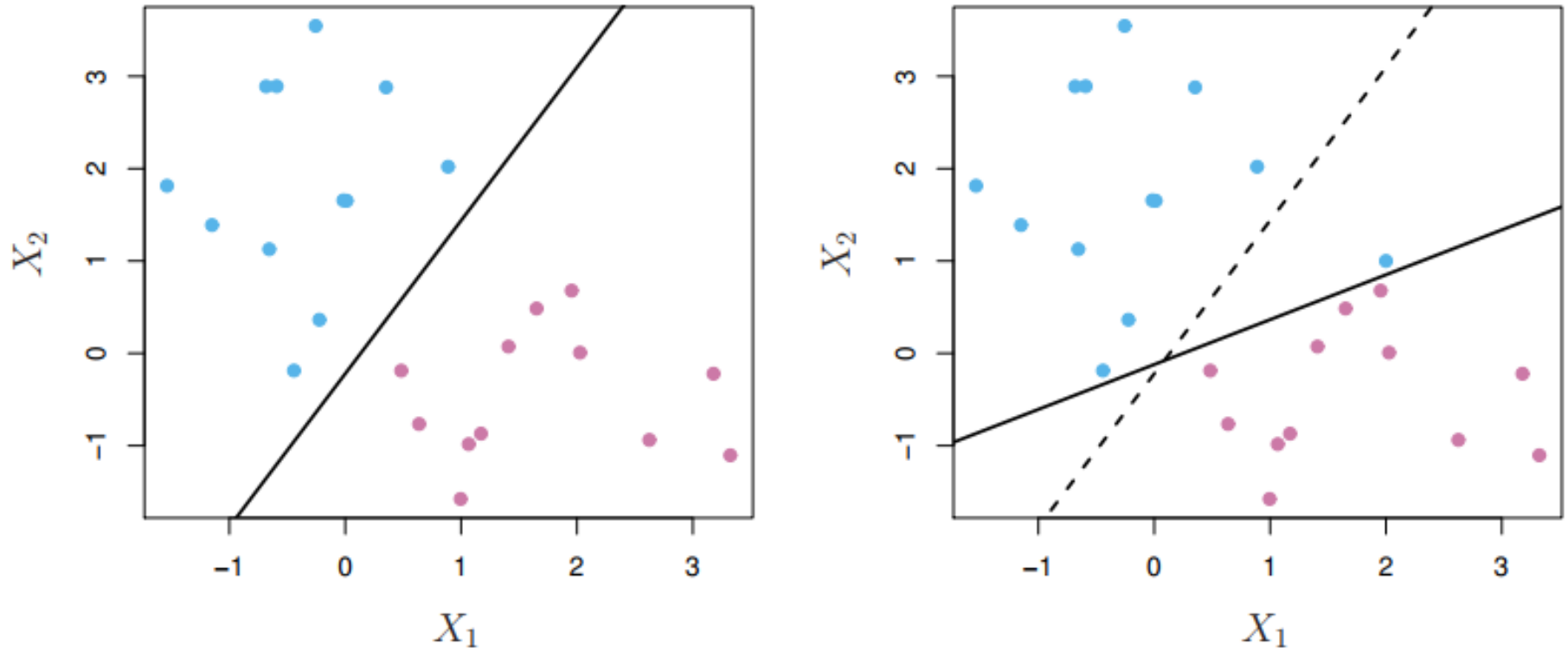
Non-separable Data

- The data on the right are not separable by a linear boundary.

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



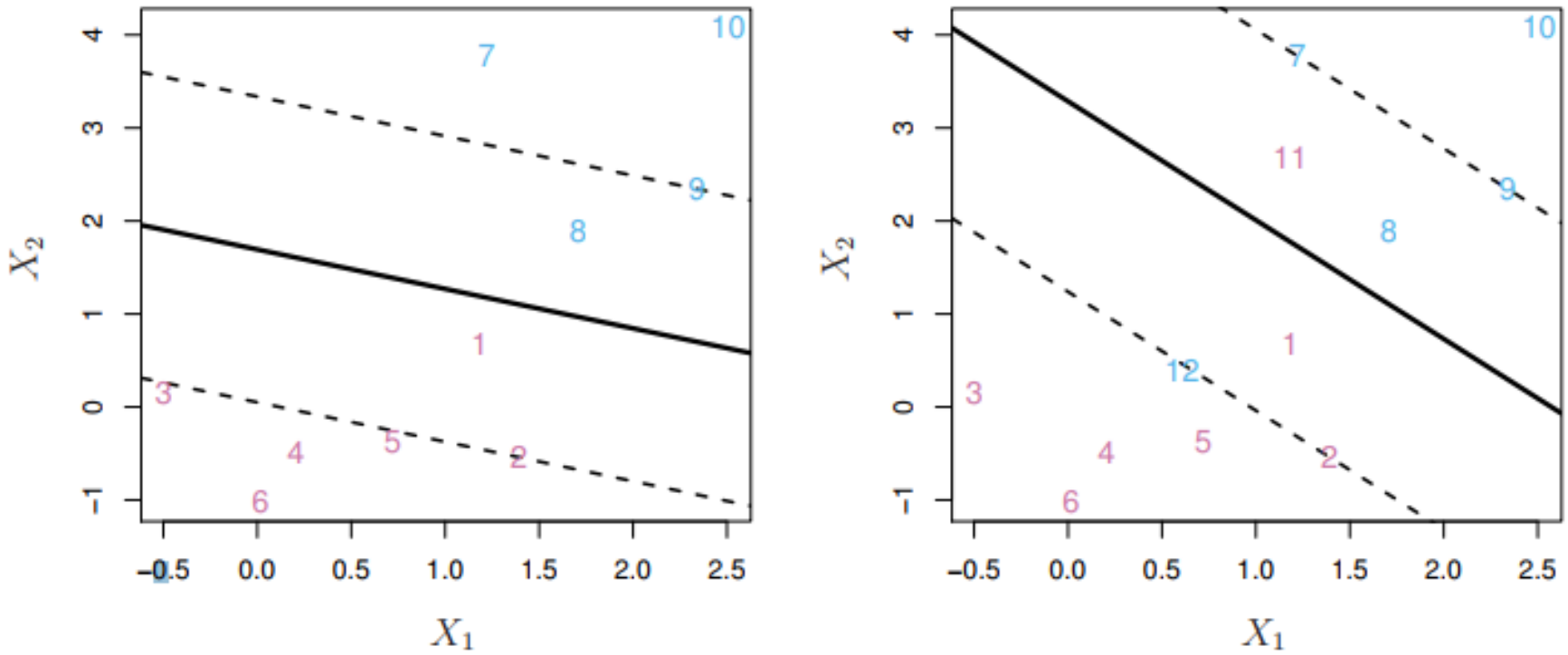
Noisy Data



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

The support vector classifier maximizes a soft margin.

Support Vector Classifier

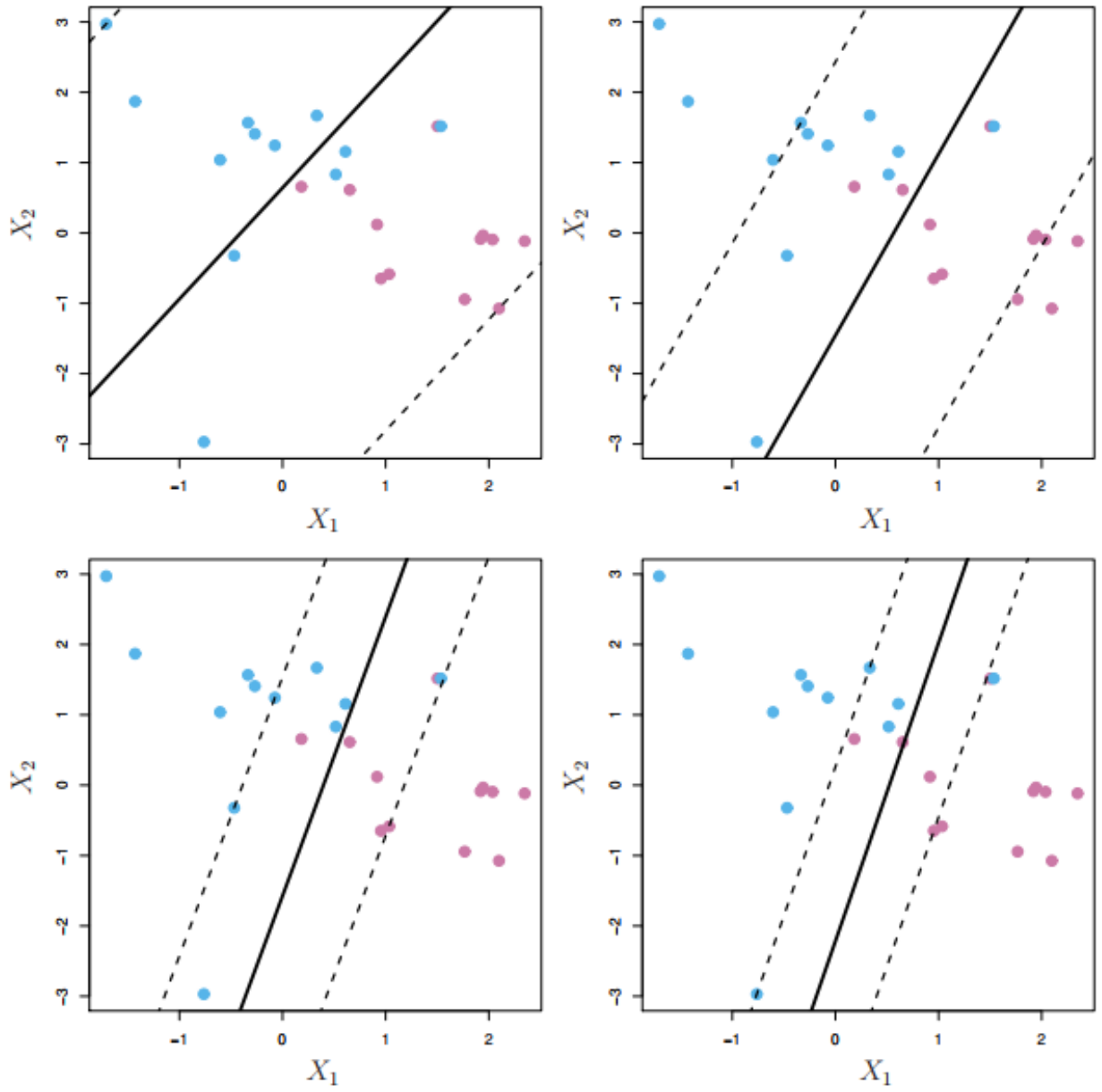


$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \; M \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

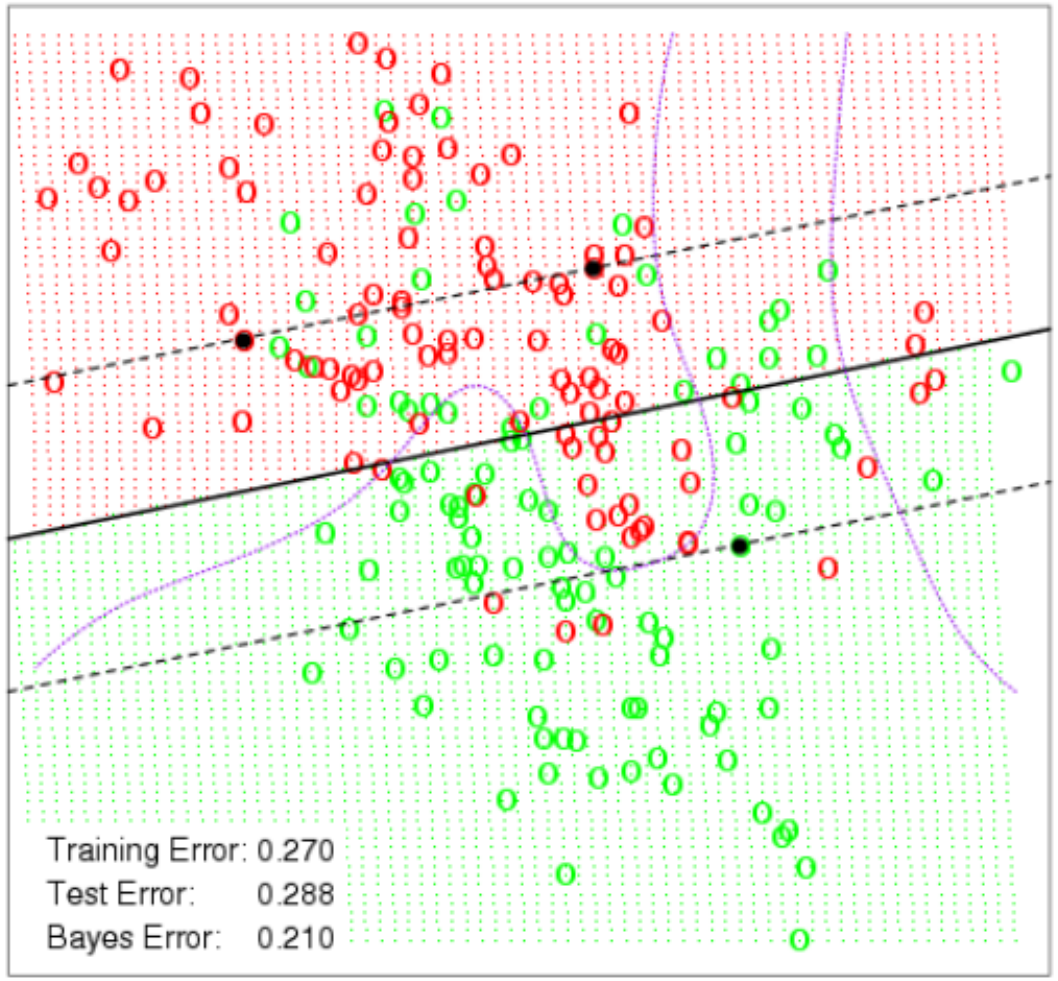
$$\epsilon_i \geq 0, \; \sum_{i=1}^{n} \epsilon_i \leq C,$$

C is a regularization parameter



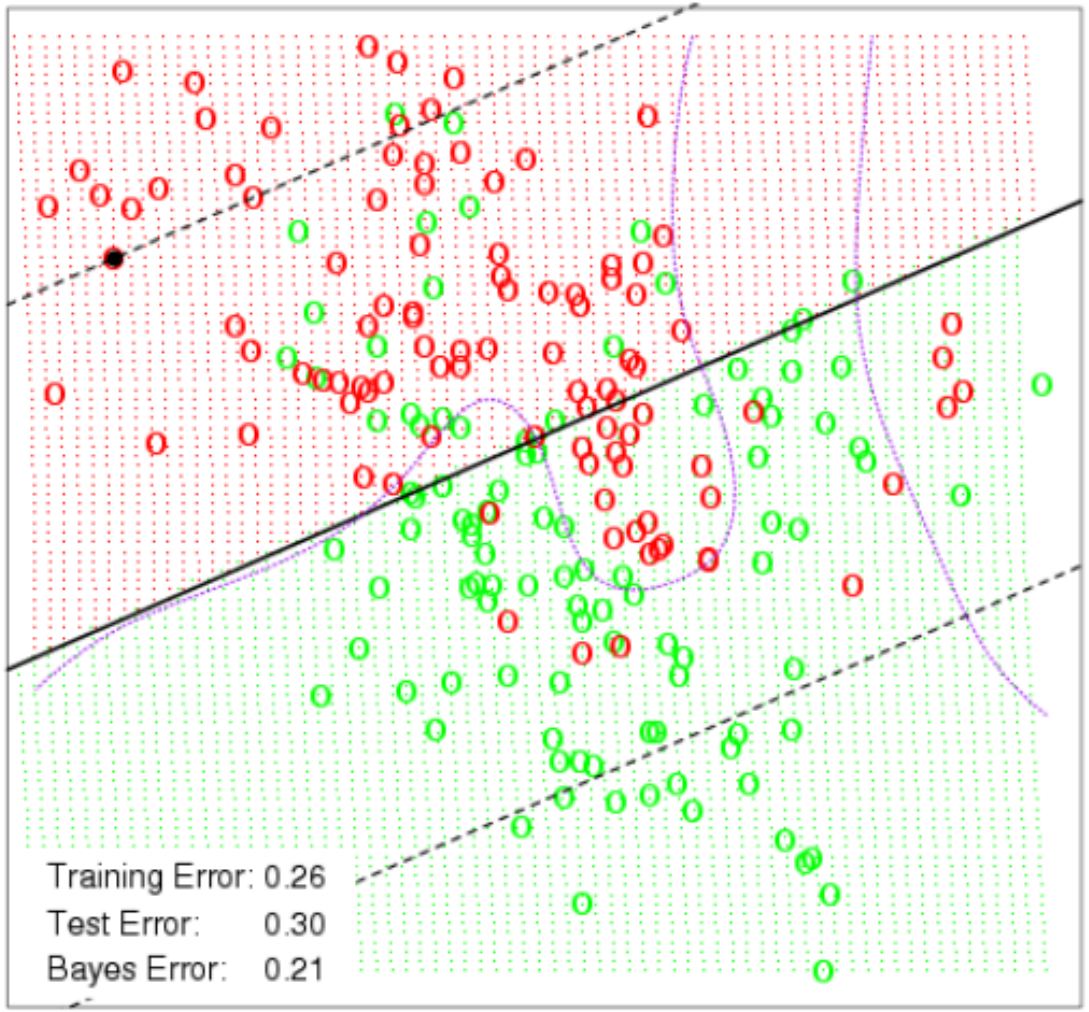
A Simulation Example With A Small Constant

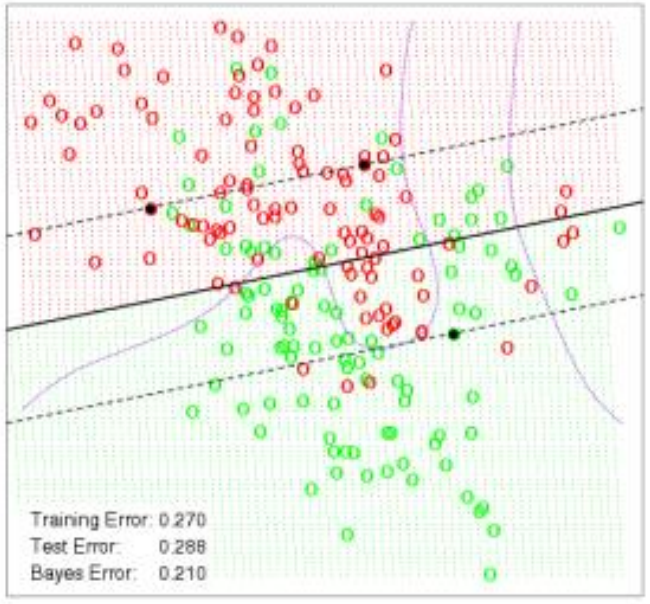
- This is the simulation example from chapter 1.
- The distance between the dashed lines represents the margin or $2C$.
- The purple lines represent the Bayes decision boundaries



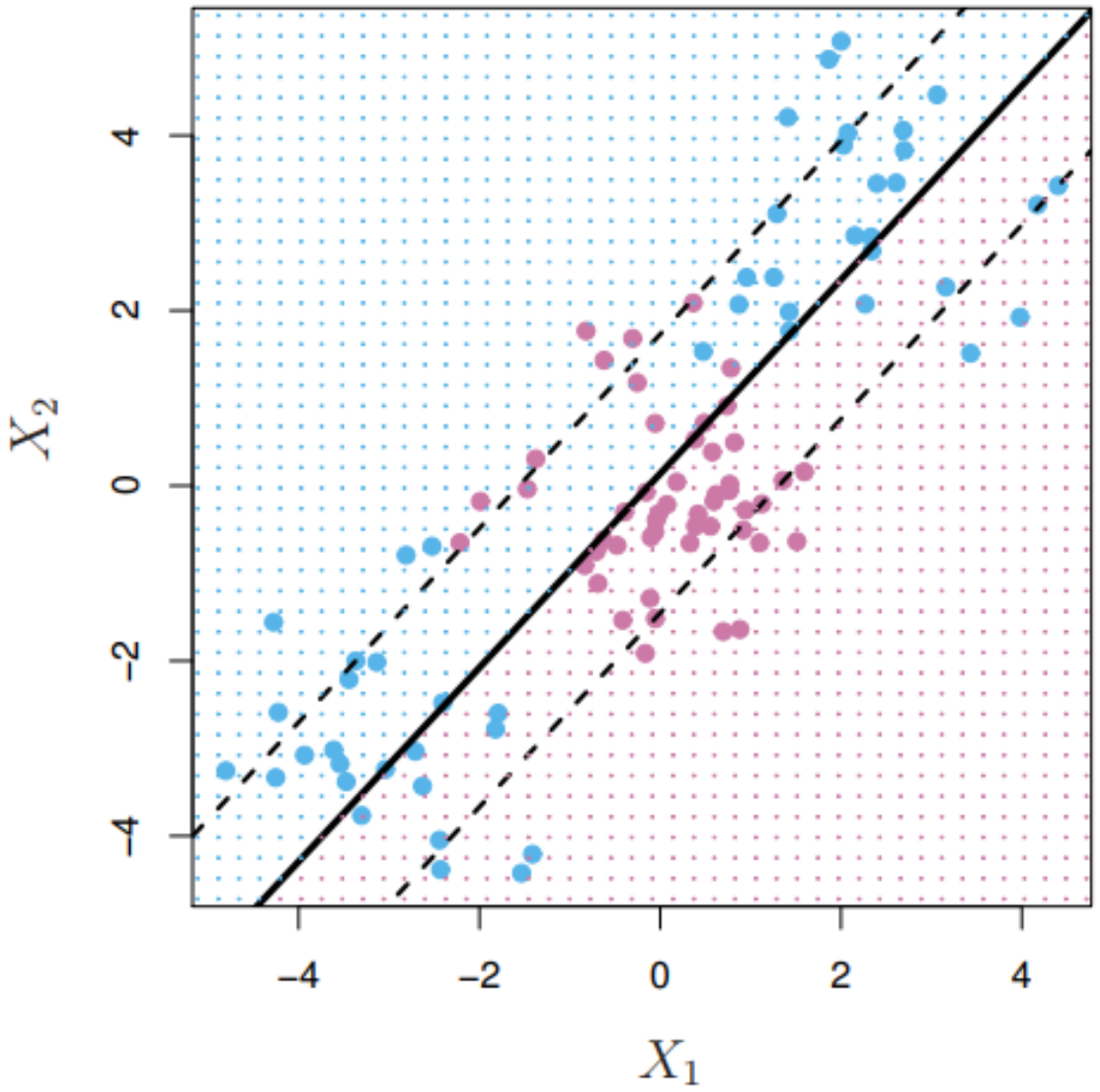
The Same Example With A Larger Constant

- Using a larger constant allows for a greater margin and creates a slightly different classifier.
- Notice, however, that the decision boundary must always be linear.





Linear boundary can fail



- Sometime a linear boundary simply won't work, no matter what value of C.
- The example on the left is such a case. What to do?

Feature Expansion

- Enlarge the space of features by including transformations; e.g.
- $X_1^2, X_1^3, X_1X_2, X_1X_2^2, \ldots$ 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.

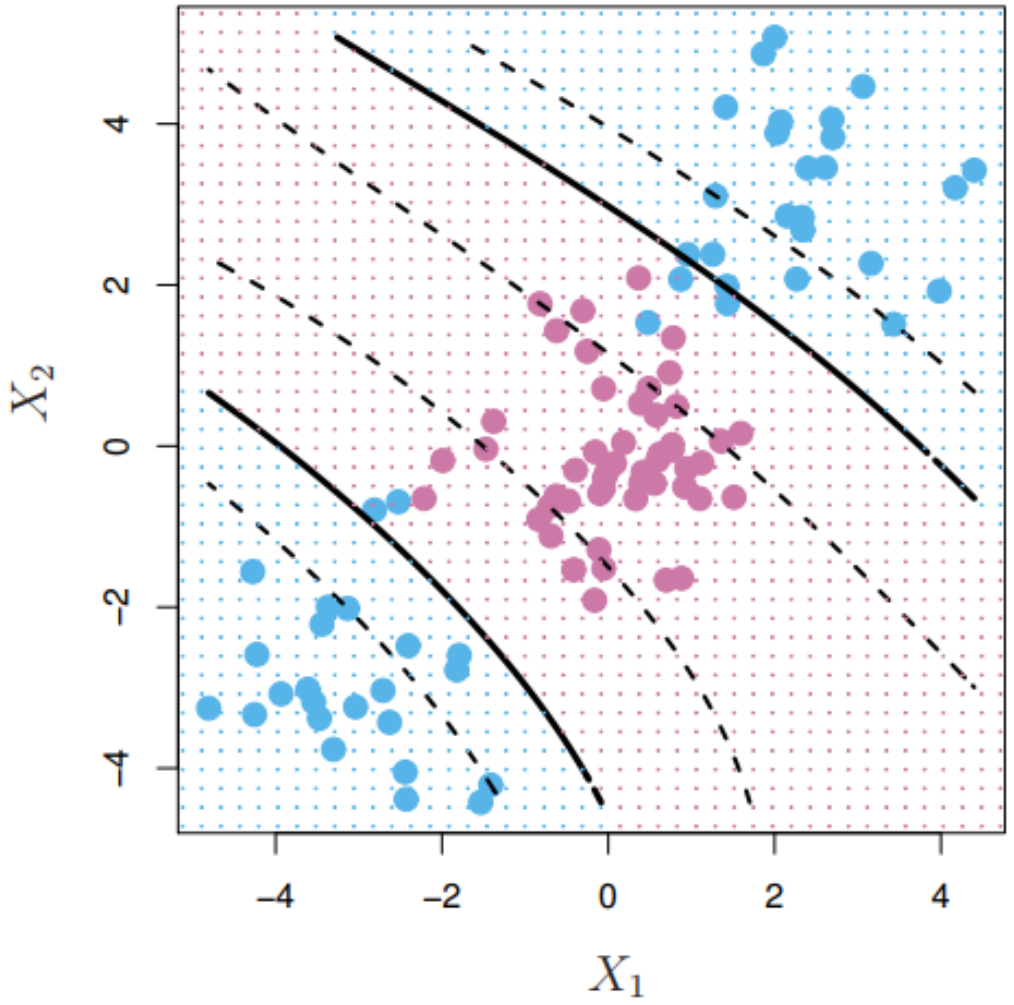
Example: Suppose we use $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$ instead of just (X_1, X_2). Then the decision boundary would be of the form

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

- This leads to nonlinear decision boundaries in the original space (quadratic conic sections).

Cubic Polynomials

- 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



$$\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_1^2+\beta_4X_2^2+\beta_5X_1X_2+\beta_6X_1^3+\beta_7X_2^3+\beta_8X_1X_2^2+\beta_9X_1^2X_2=0$$

Nonlinearities 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

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$$ — inner product between vectors

The linear support vector classifier can be represented as

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

To estimate the parameters $\alpha_1, \ldots, \alpha_n$ and β_0, all we need are the $\binom{n}{2}$ inner products $\langle x_i, x_{i'} \rangle$ between all pairs of training observations.
It turns out that most of the $\hat{\alpha}_i$ can be zero:

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

S is the support set of indices i such that $\hat{\alpha}_i > 0$.

Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SV classifier. Can be quite abstract!
- Some special kernel functions can do this for us. E.g.

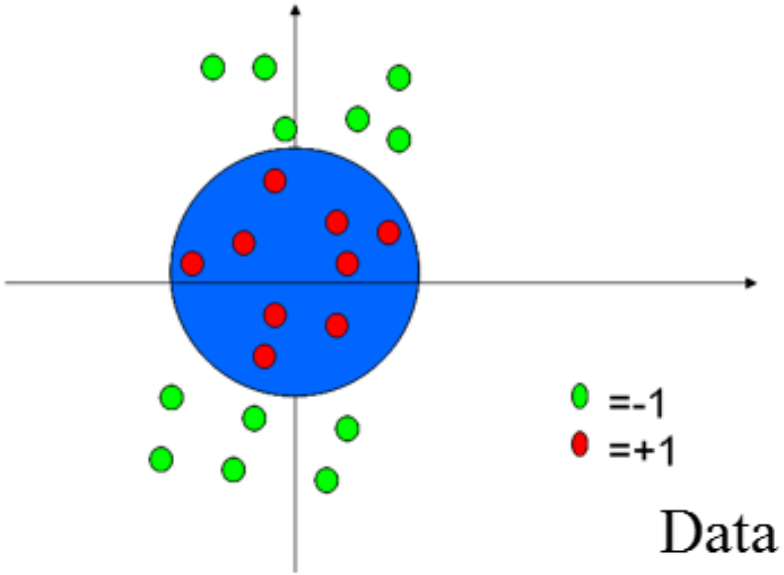
$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

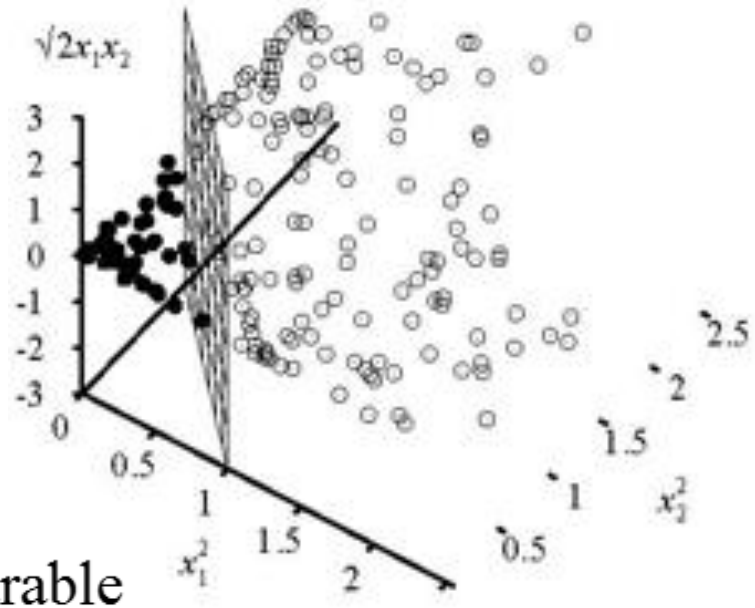
computes the inner-products needed for d dimensional polynomials $\binom{p+d}{d}$ basis functions! Try it for p = 2 and d = 2.

- The solution has the form

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

Kernel Trick





Data points are linearly separable

in the space $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$

We want to maximize $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \left\langle F(\mathbf{x}_i) \cdot F(\mathbf{x}_j) \right\rangle$

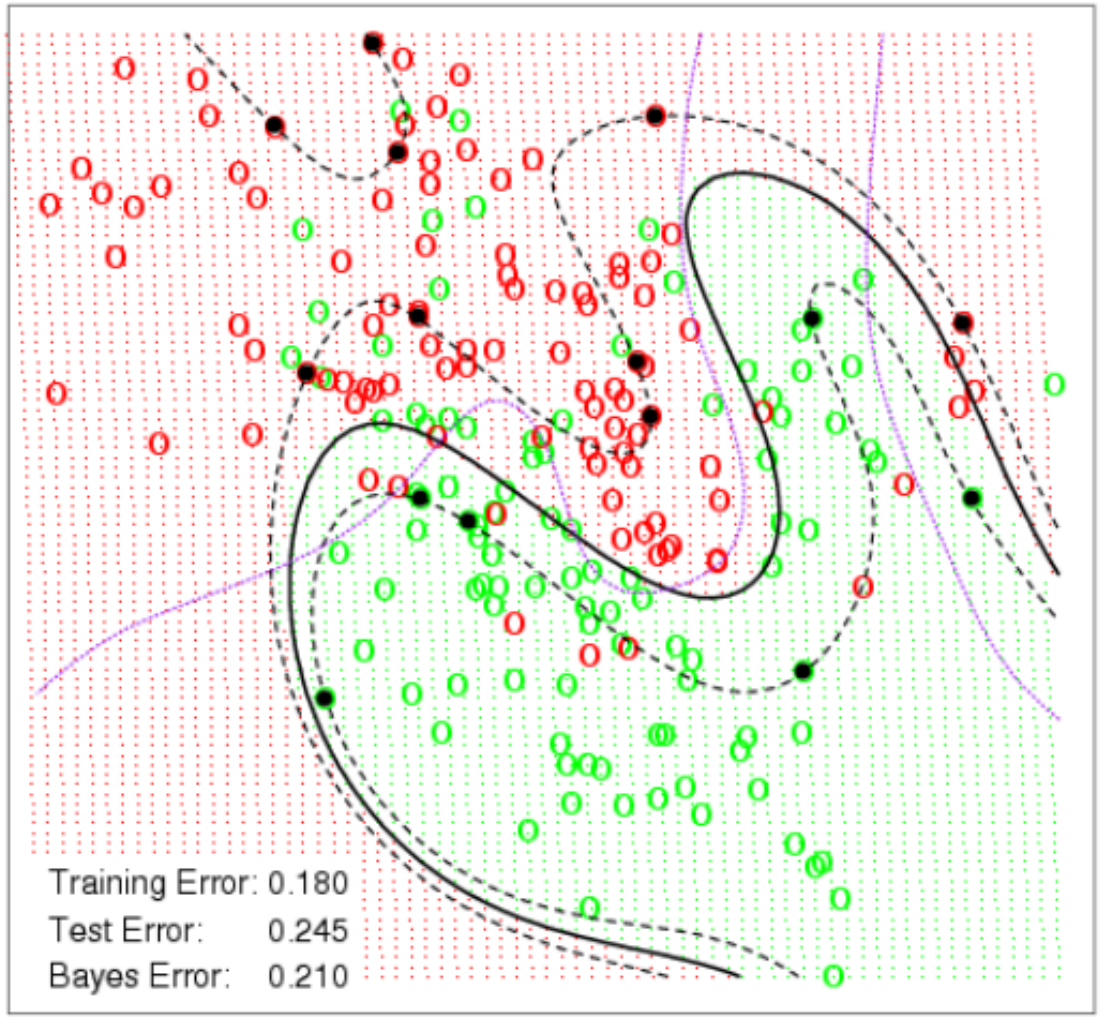
Define $K(\mathbf{x}_i, \mathbf{x}_j) = \left\langle F(\mathbf{x}_i) \cdot F(\mathbf{x}_j) \right\rangle$

Cool thing : K is often easy to compute directly! Here,

$K(\mathbf{x}_i, \mathbf{x}_j) = \left\langle \mathbf{x}_i \cdot \mathbf{x}_j \right\rangle^2$

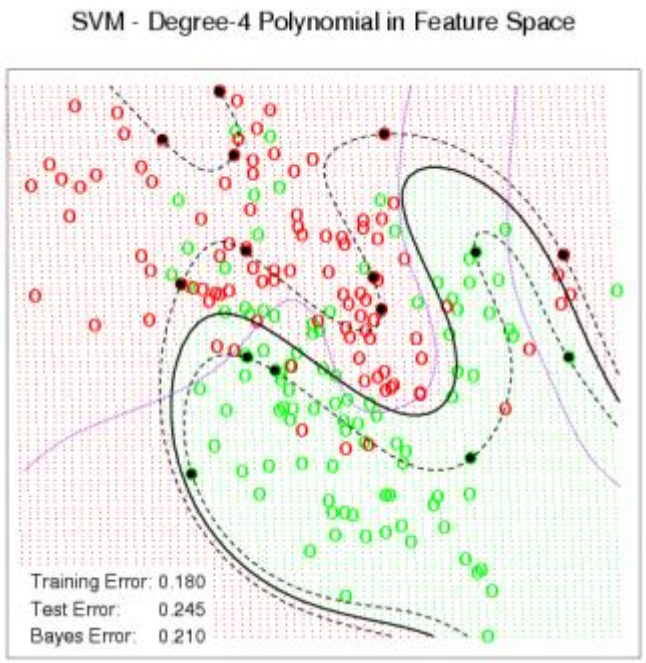
Polynomial Kernel On Sim Data

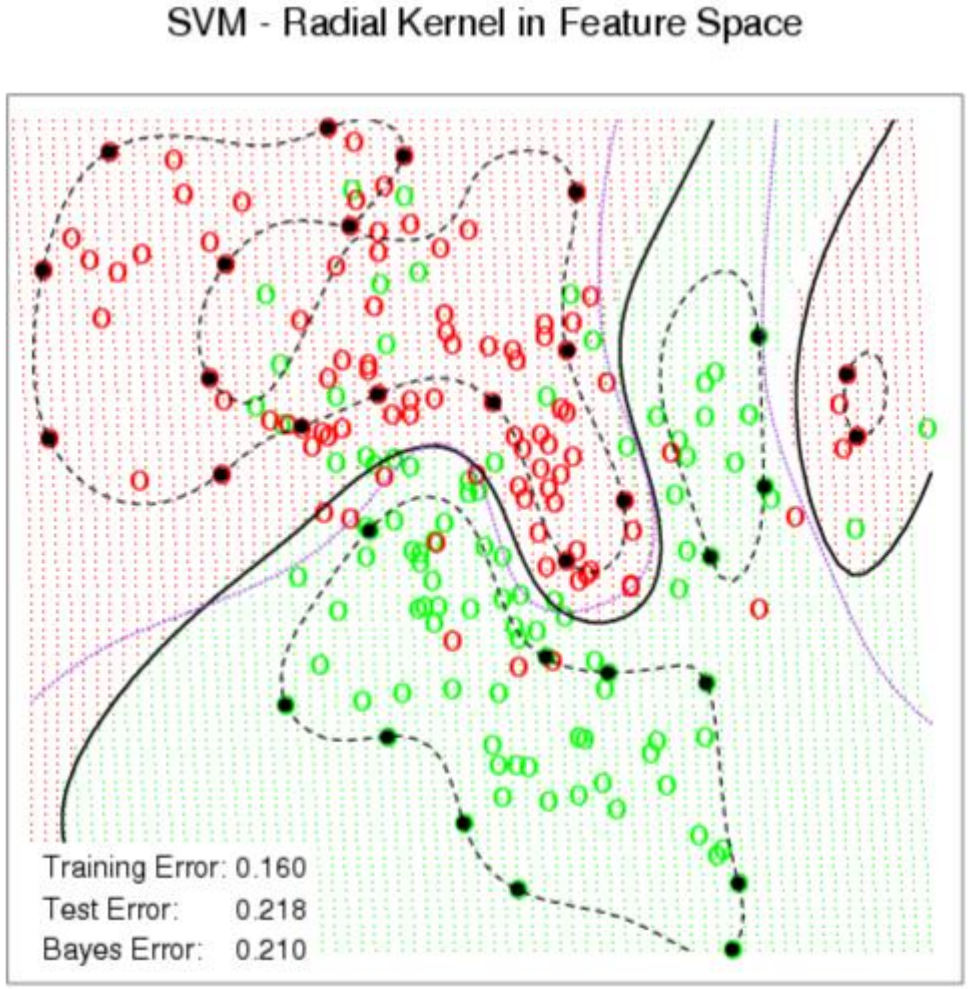
- Using a polynomial kernel we now allow SVM to produce a non-linear decision boundary.
- Notice that the test error rate is a lot lower.



Radial Basis Kernel

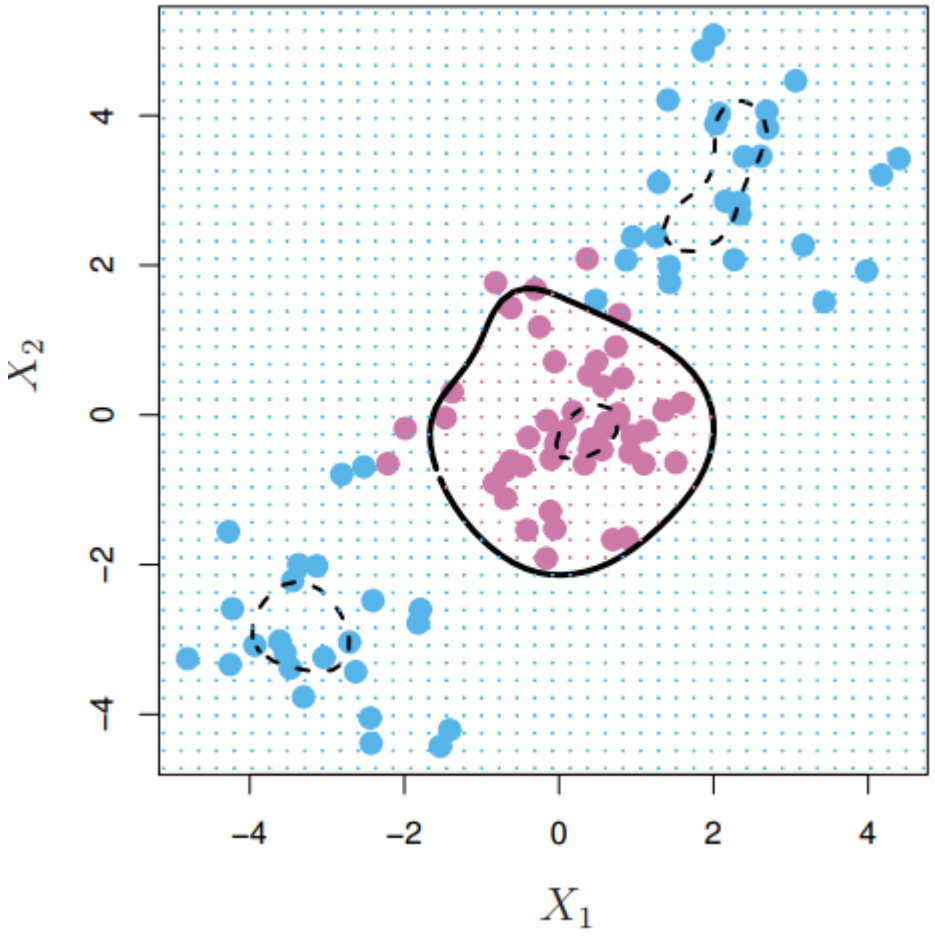
- Using a Radial Basis Kernel you get an even lower error rate.





Radial Kernel

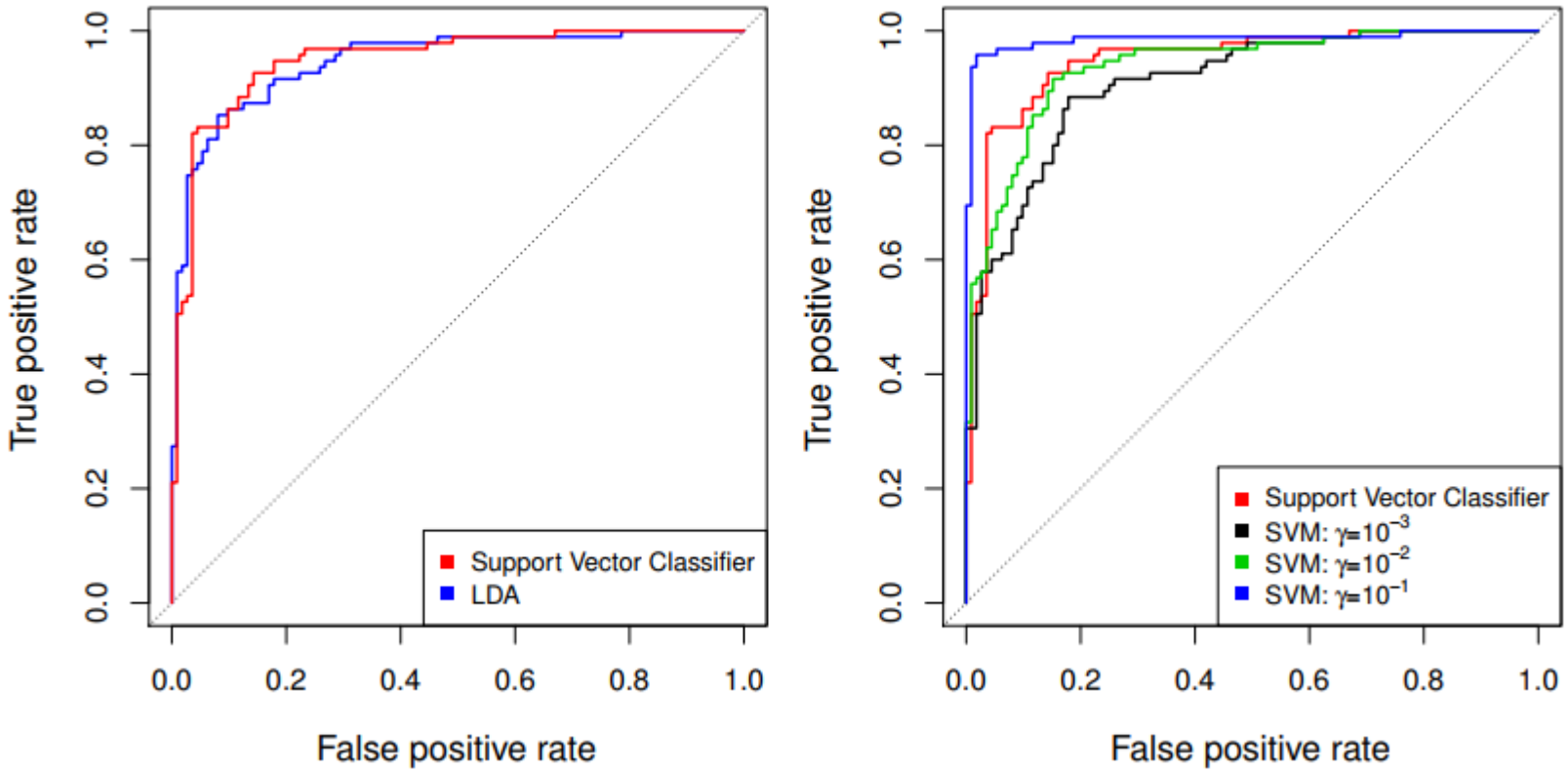
$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$



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

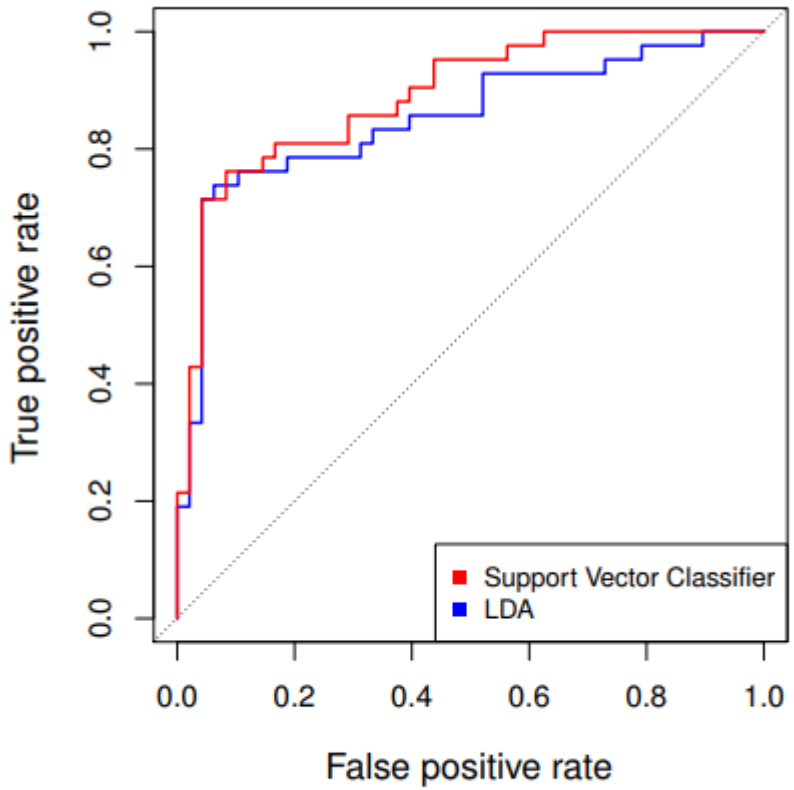
- Implicit feature space; very high dimensional.
- Controls variance by squashing down most dimensions severely

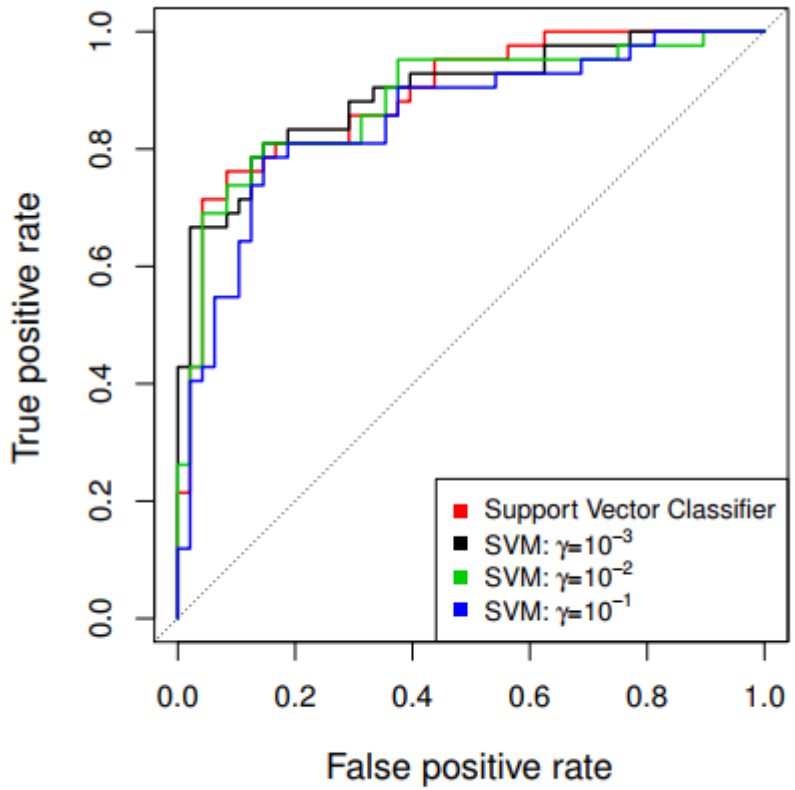
Example: Heart Data



- 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?

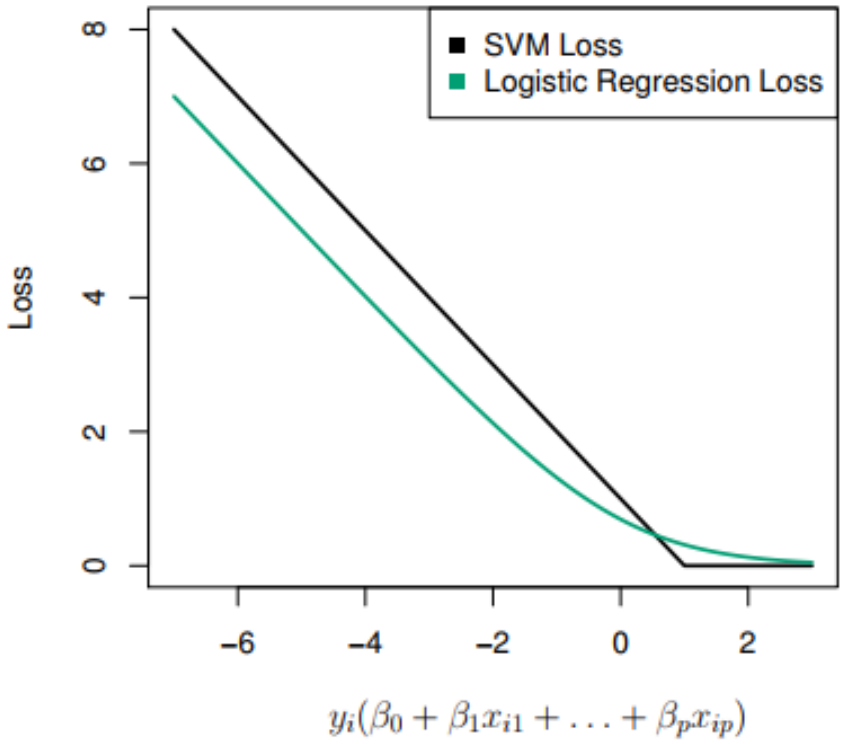
- The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?
  - OVA One versus All. Fit K different 2-class SVM classifiers $\hat{f}_k(x),\ k = 1, \ldots, K$; each class versus the rest. Classify x^* to the class for which $\hat{f}_{k\ell}(x)$ is largest.
  - OVO One versus One. Fit all $\binom{K}{2}$ pairwise classifiers $\hat{f}_k(x^*)$. 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 $f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$ we can rephrase support-vector classifier optimization as

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



- 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).

Which to use: SVM or Logistic Regression

- When classes are (nearly) separable, SVM does better than LR. So does LDA.
- When not, LR (with ridge penalty) and SVM very similar.
- If you wish to estimate probabilities, LR is the choice.
- For nonlinear boundaries, kernel SVMs are popular. Can use kernels with LR and LDA as well, but computations are more expensive.