ISLR 9. Support Vector Machines (SVM)

Topics : The Support Vector Classifier

: The Support Vector Machine Classifier

9.1 Support Vector Classifier

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
  All 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.

Its Easiest To See With A Picture

C is the minimum perpendicular distance

- between each point and the separating line.

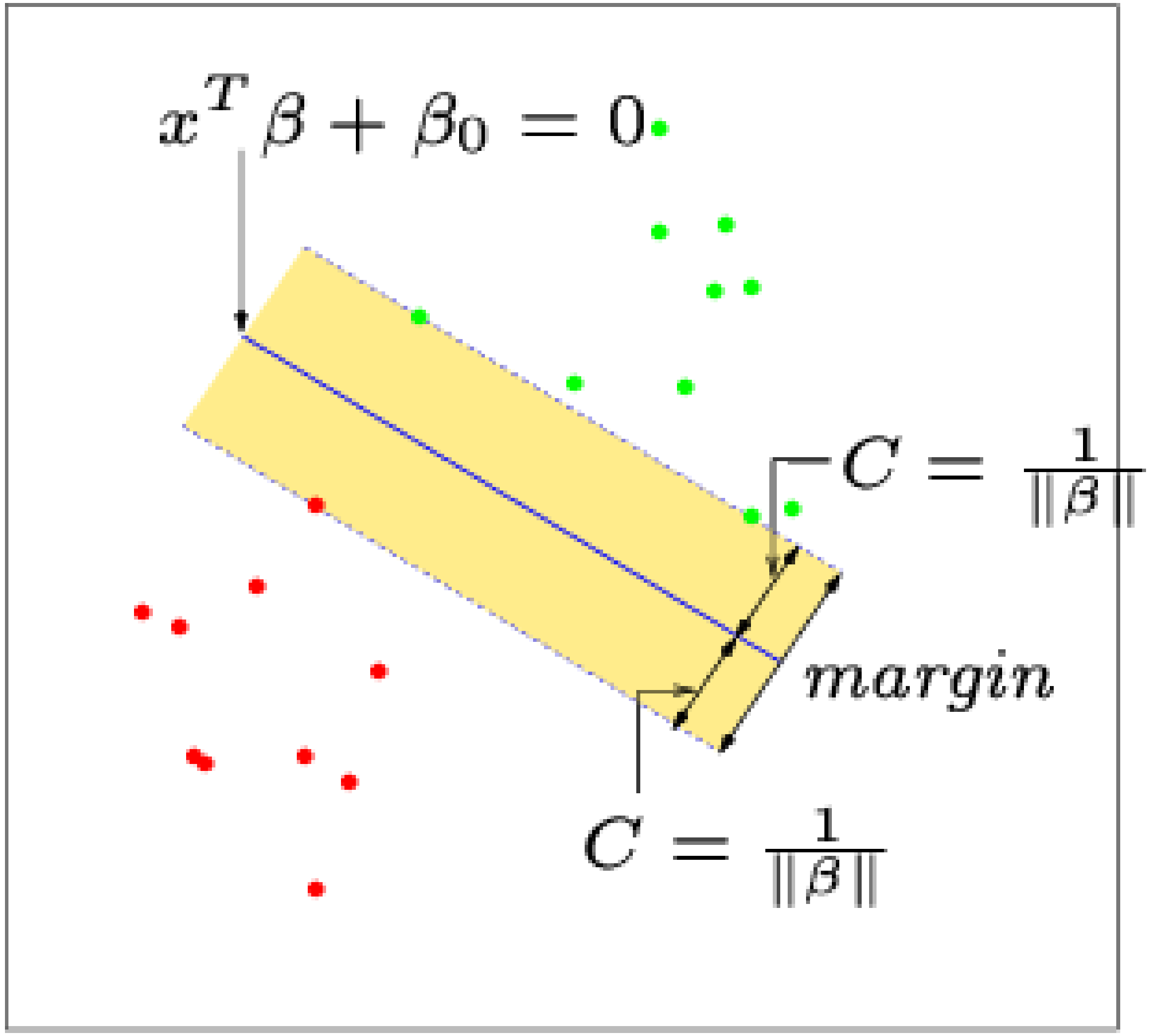
We find the line which maximizes C.

This line is called

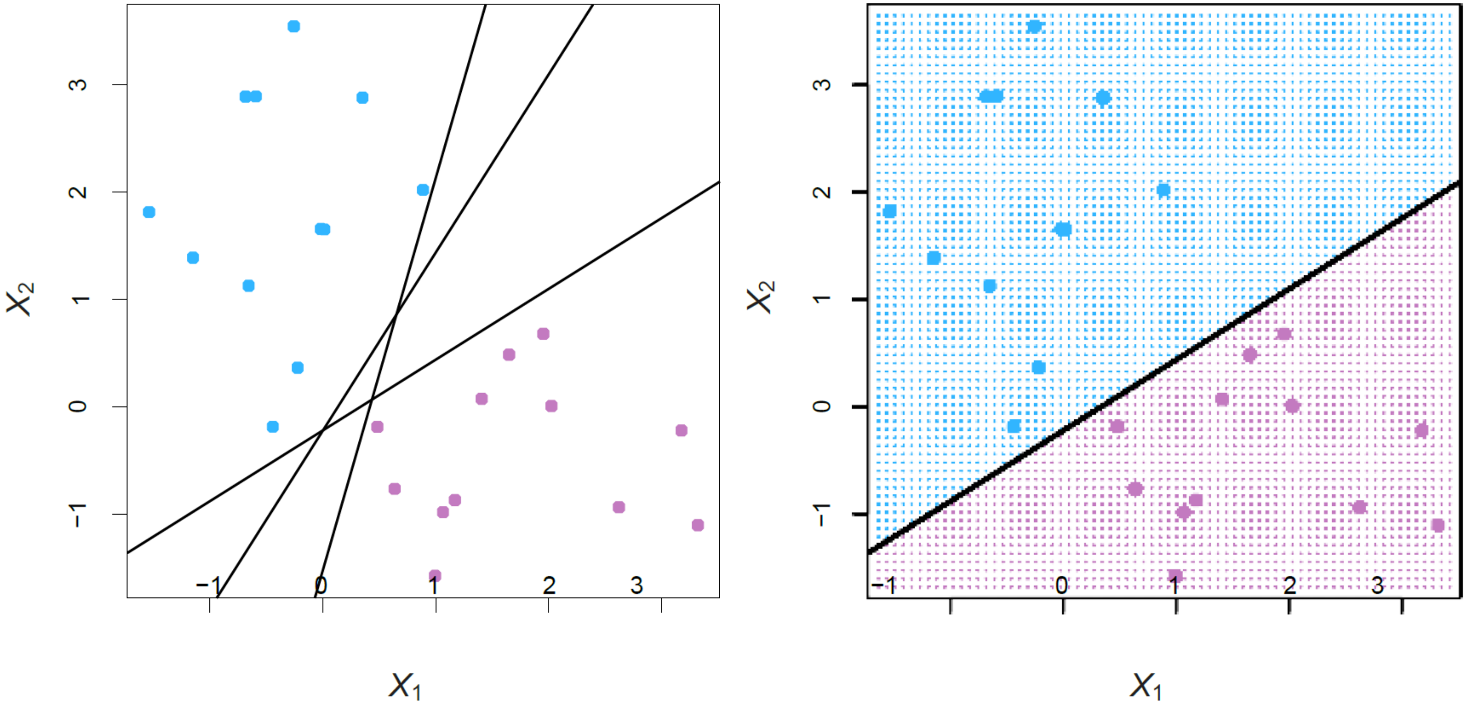
- the “optimal separating hyperplane”

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*.

What is a hyperplane

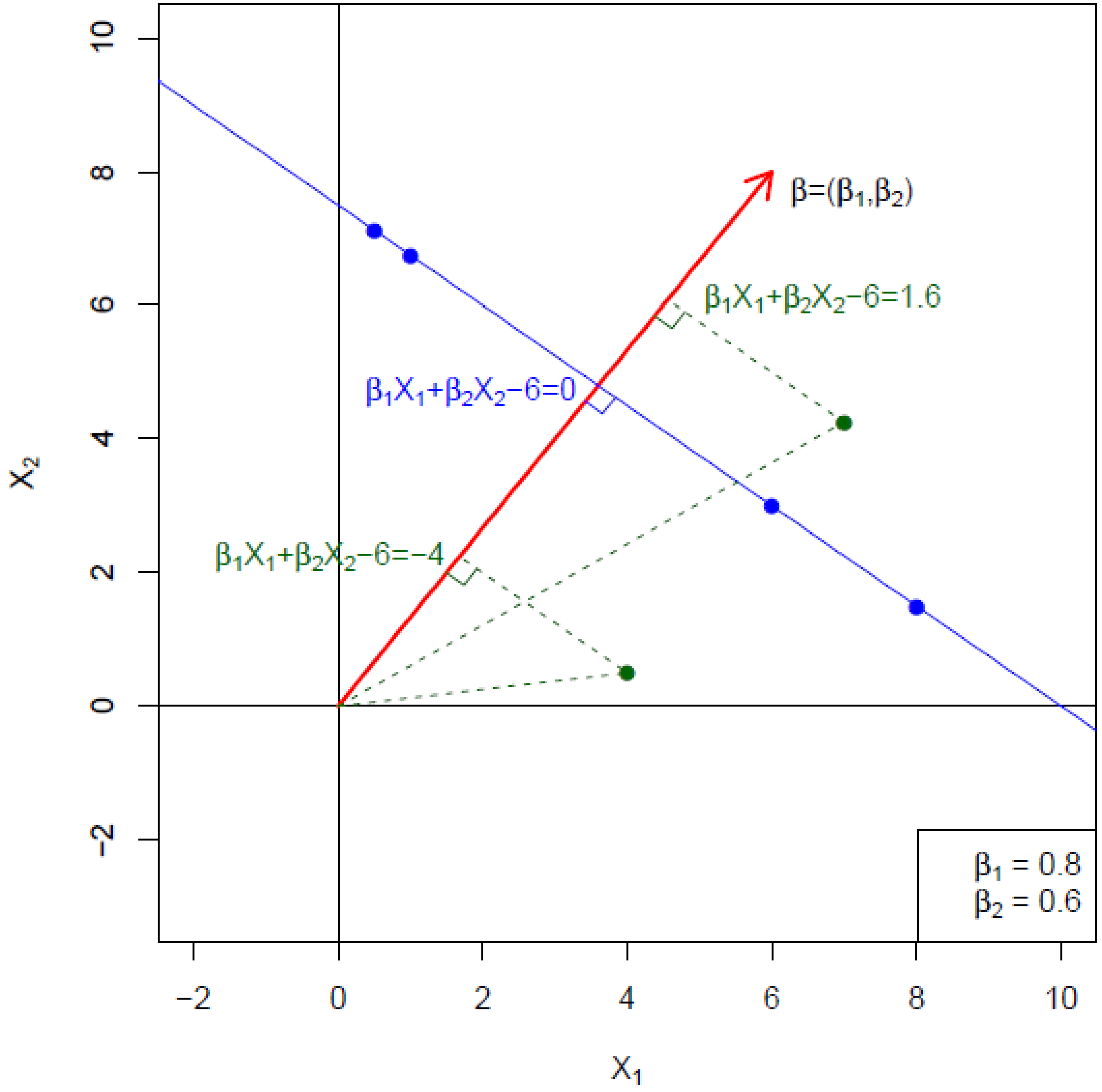
- A hyperplane in p dimensions is a flat affine subspace of dimension $p - 1$.
- In general the equation for a hyperplane has the form

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

- In $p = 2$ dimensions a hyperplane is a line.
- If $\beta_0 = 0$, the hyperplane goes through the origin, otherwise not.
- The vector $\beta = (\beta_1, \beta_2, \cdots, \beta_p)$ is called the normal vector

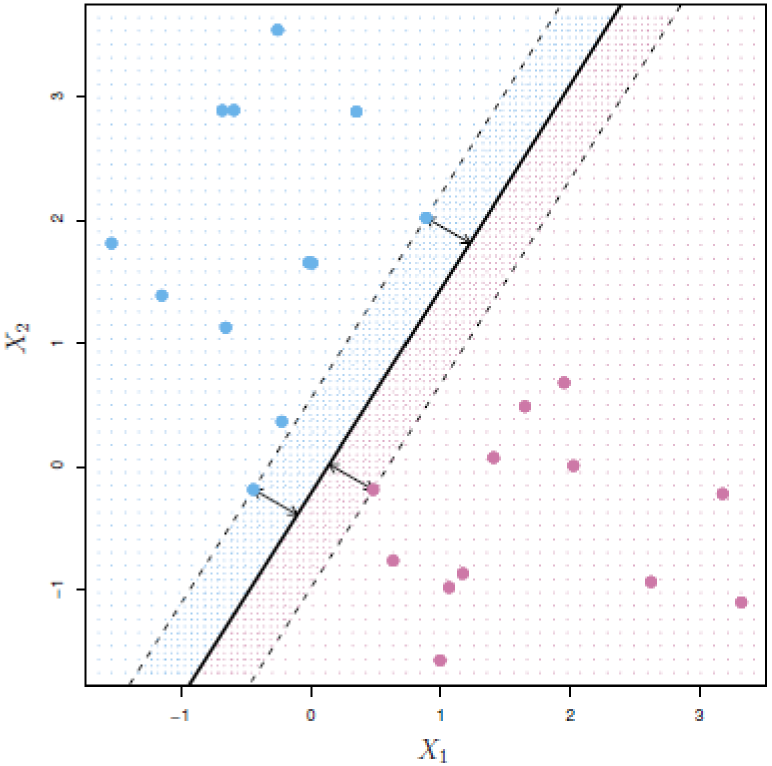
— it points in a direction orthogonal to the surface of a hyperplane.

Hyperplane in 2 Dimensions



Maximal Margin Classier

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



Constrained optimization problem

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

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

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

$$\text{for all } i = 1, \ldots, N.$$

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

More Than Two Predictors

This idea works just as well

- with more than two predictors.

For example, with three predictors

- you want to find the plane
- that produces the largest separation between the classes.

With more than three dimensions

- it becomes hard to visualize a plane
  but it still exists.
- In general they are called hyper-planes.

Non-Separating Classes

Of course in practice it is not usually possible

- to find a hyper-plane that perfectly separates two classes.

In other words, for any straight line or plane that I draw

- there will always be at least some points on the wrong side of the line.

In this situation we try to find the plane

- that gives the best separation between the points
  that are correctly classified
- subject to the points on the wrong side of the line
  not being off by too much.

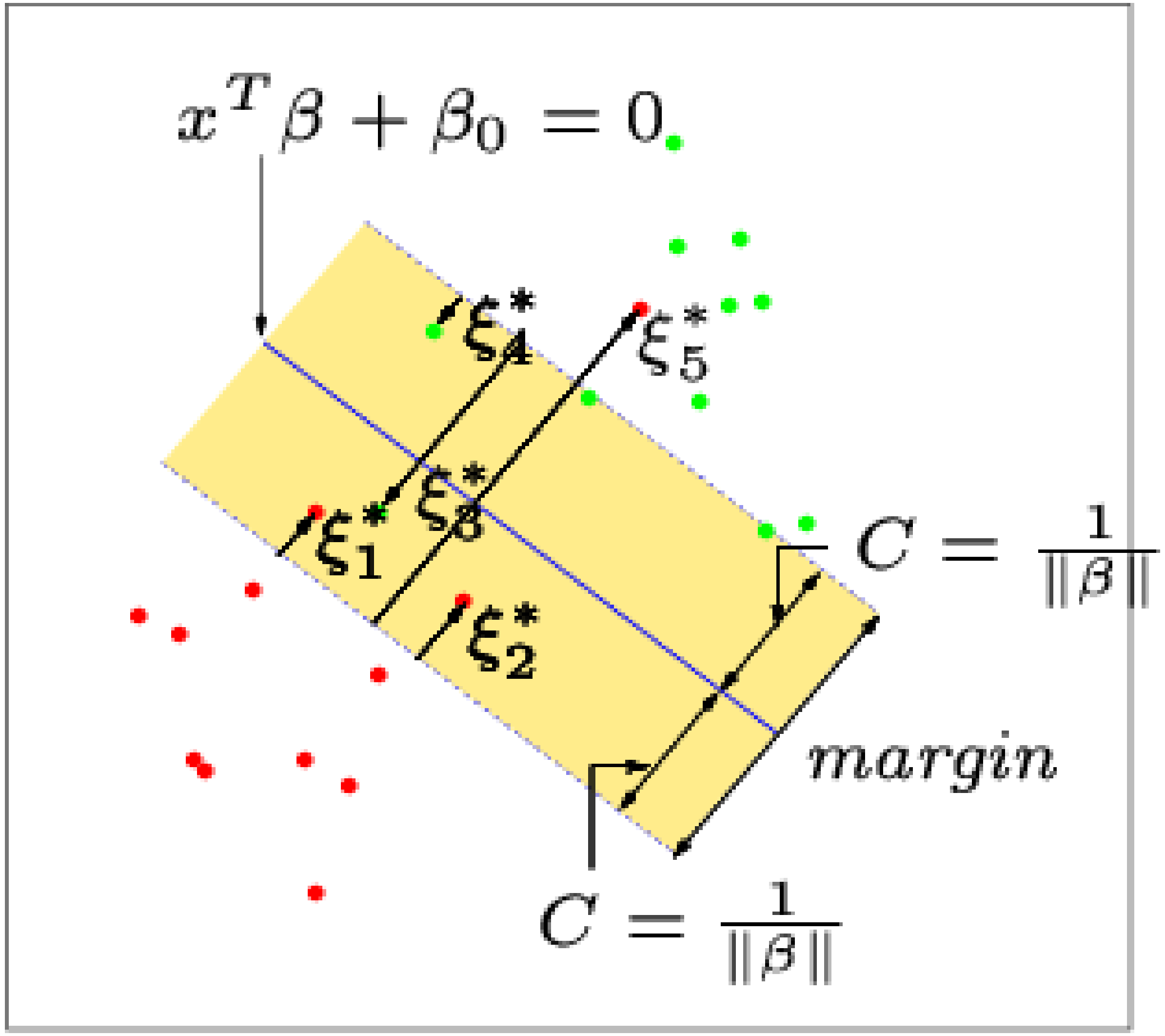
It is easier to see with a picture!

Non-Separating Example

Let ξ^*_i represent the amount
- that the ith point is
- on the wrong side
- of the margin (the dashed line).

Then we want to maximize C
- subject to $\frac{1}{C}\sum_{i=1}^{n}\xi_i^* \le \text{Constant}$

The constant C
- is a tuning parameter
- that we choose.

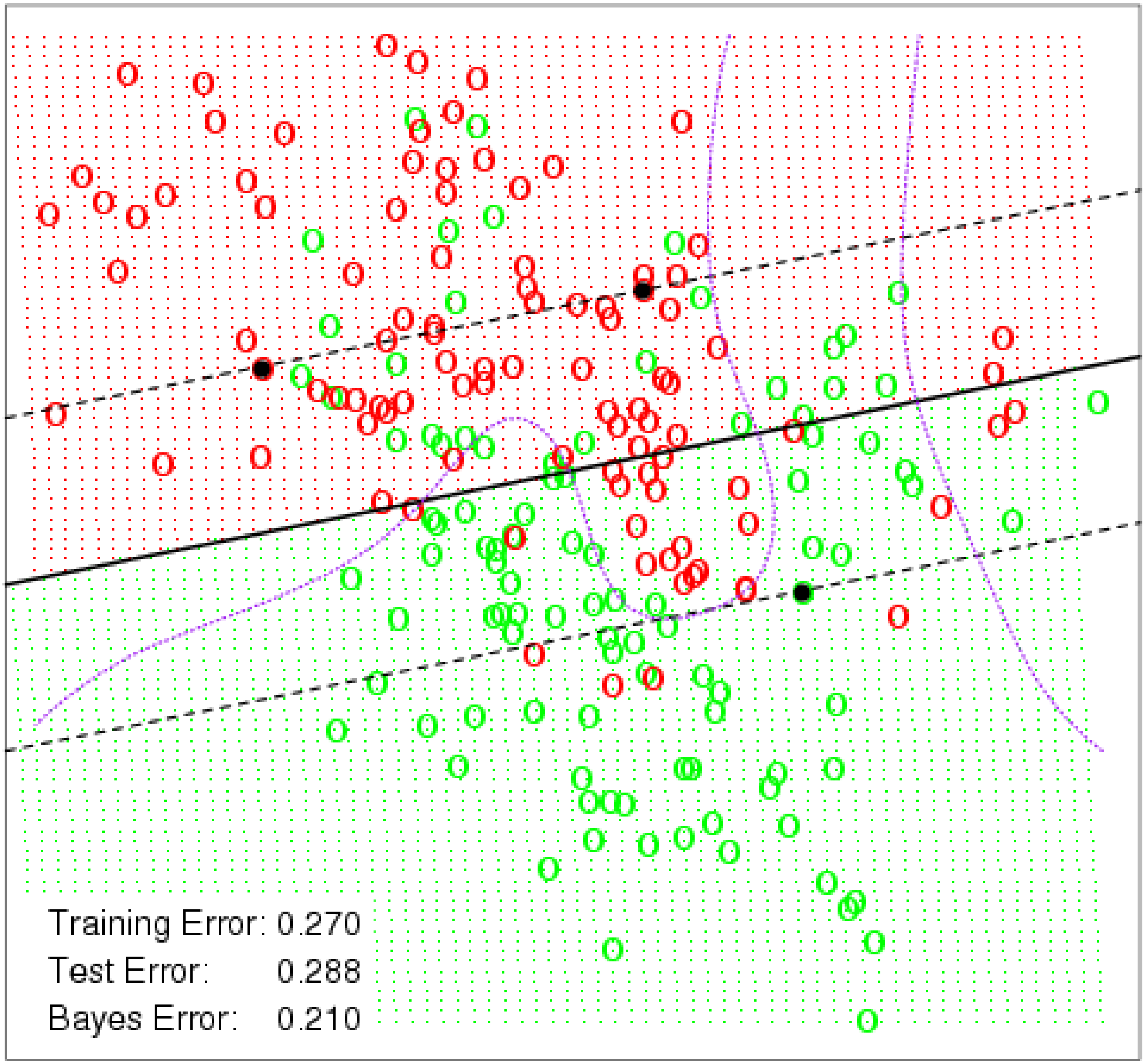


A Simulation Example With A Small Constant

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



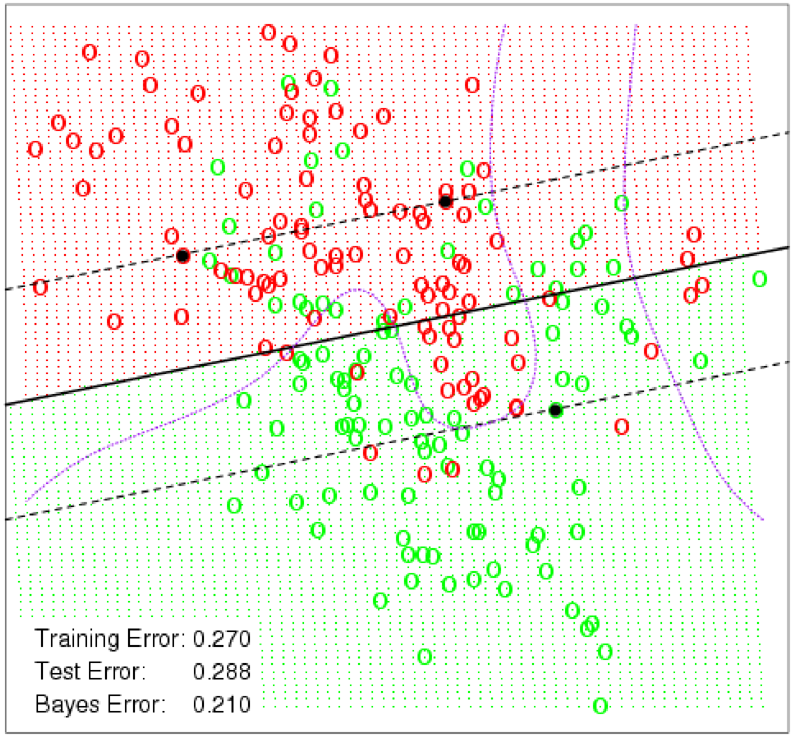
The Same Example With A Larger Constant

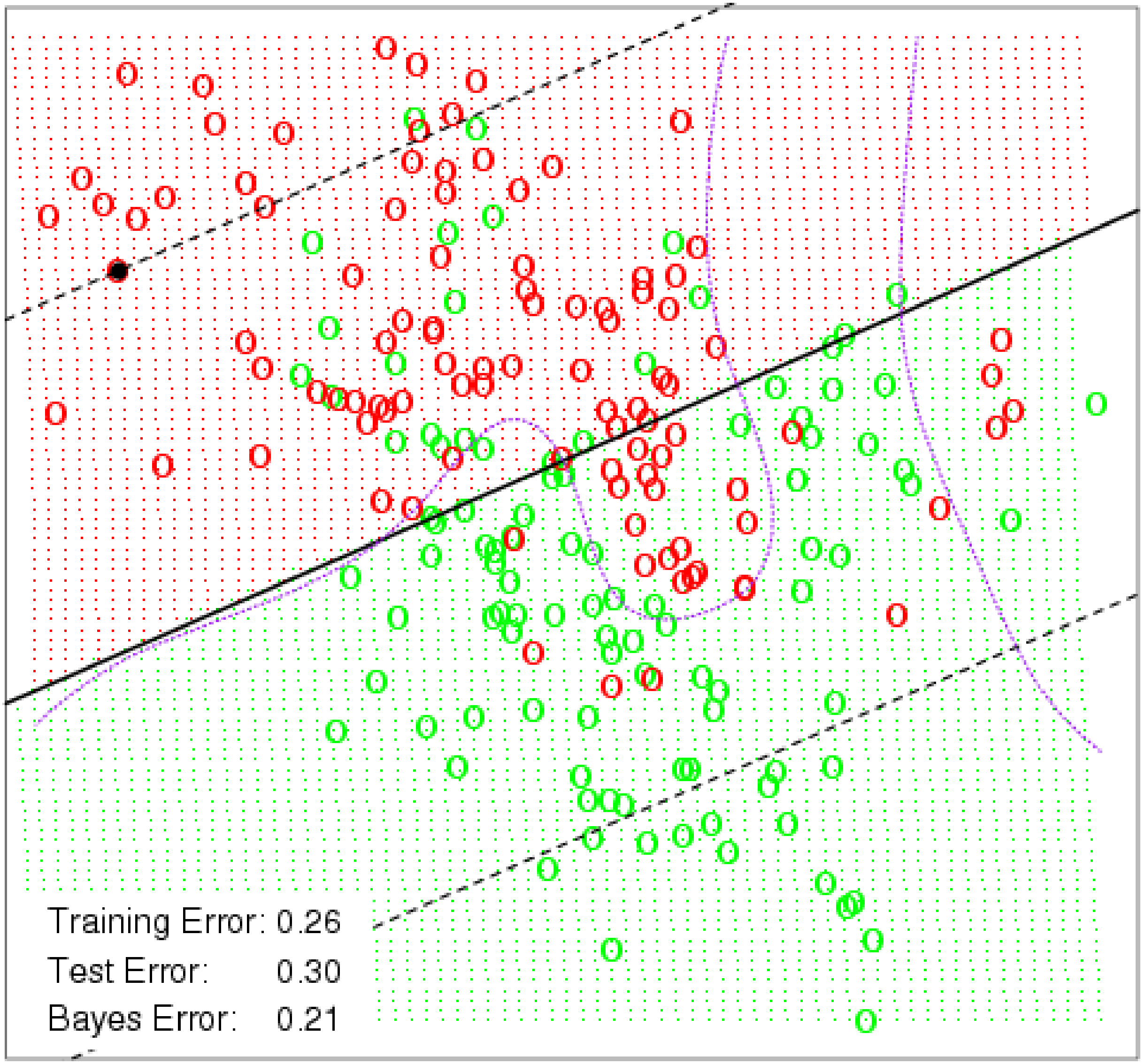
Using a larger constant C

- allows for a greater margin and
- creates a slightly different classifier.

Notice, however, that the decision boundary

- must always be linear.





9.2 Support Vector Machine Classifier

Non-Linear Classifier

The support vector classifier is fairly easy to think about.

- However, because it only allows for a linear decision boundary
- it may not be all that powerful.

Recall that in chapter 3 we extended linear regression

- to non-linear regression using a basis function i.e.

$$Y_i = \beta_0 + \beta_1 b_1(X_i) + \beta_2 b_2(X_i) + \cdots + \beta_p b_p(X_i) + \varepsilon_i$$

Feature Expansion

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

A Basis Approach

Conceptually, we can take a similar approach

- with the support vector classifier.

The support vector classifier

- finds the optimal hyper-plane in the space spanned by X_1, X_2,…, X_p.

Instead we can create transformations

- (or a basis) $b_1(x)$, $b_2(x)$, …, $b_M(x)$ and
- find the optimal hyper-plane in the space
  spanned by $b_1(X)$, $b_2(X)$, …, $b_M(X)$.

This approach produces a linear plane

- in the transformed space
- but a non-linear decision boundary
  in the original space.

This is called the Support Vector Machine Classifier.

In Reality

While conceptually the basis approach

- is how the support vector machine works,
- there is some complicated math (which I will spare you)

  which means that we don't actually choose $b_1(x)$, $b_2(x)$, …, $b_M(x)$.

Instead we choose something called a Kernel function

- which takes the place of the basis.

Common kernel functions include

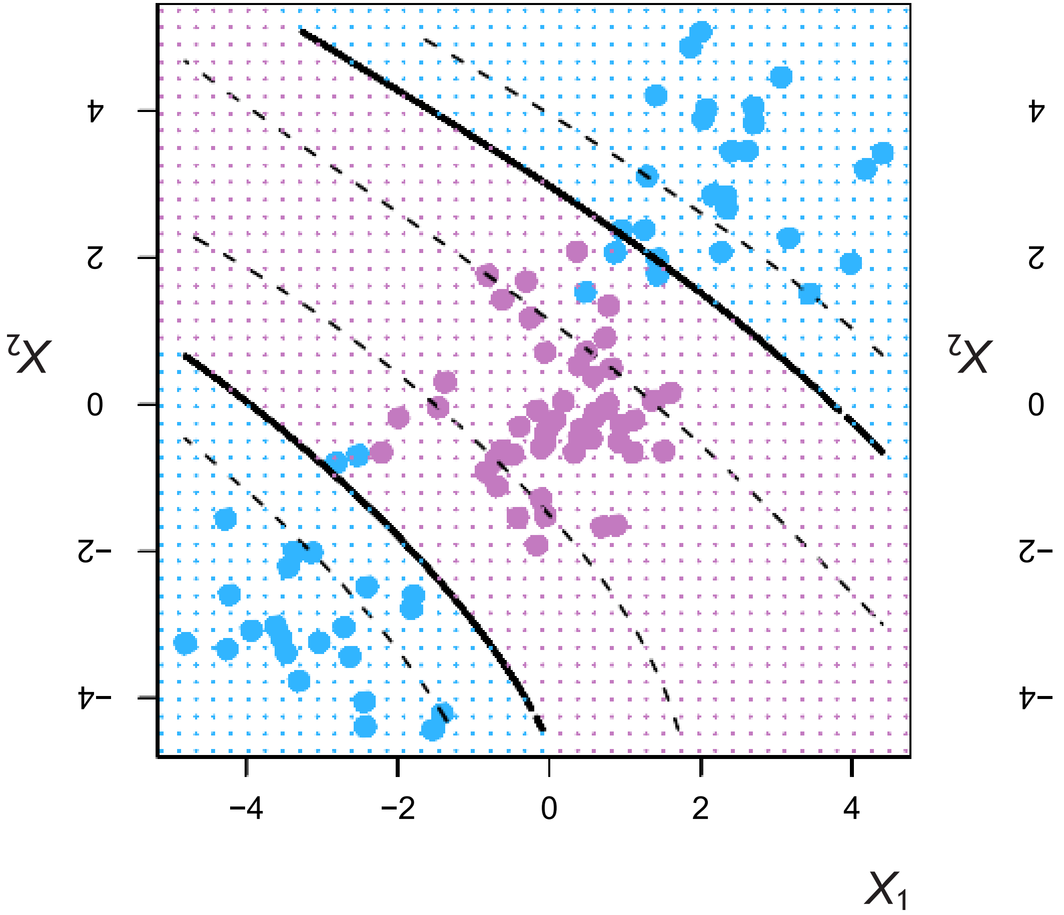
- Linear
- Polynomial
- Radial Basis
- Sigmoid

Cubic Polynomials

Here we use a basis expansion of cubic poly- nomials

From 2 variables to 9

The support-vector clas- sifier in the enlarged space solves the problem in the lower-dimensional space



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

Polynomial Kernel On Sim Data

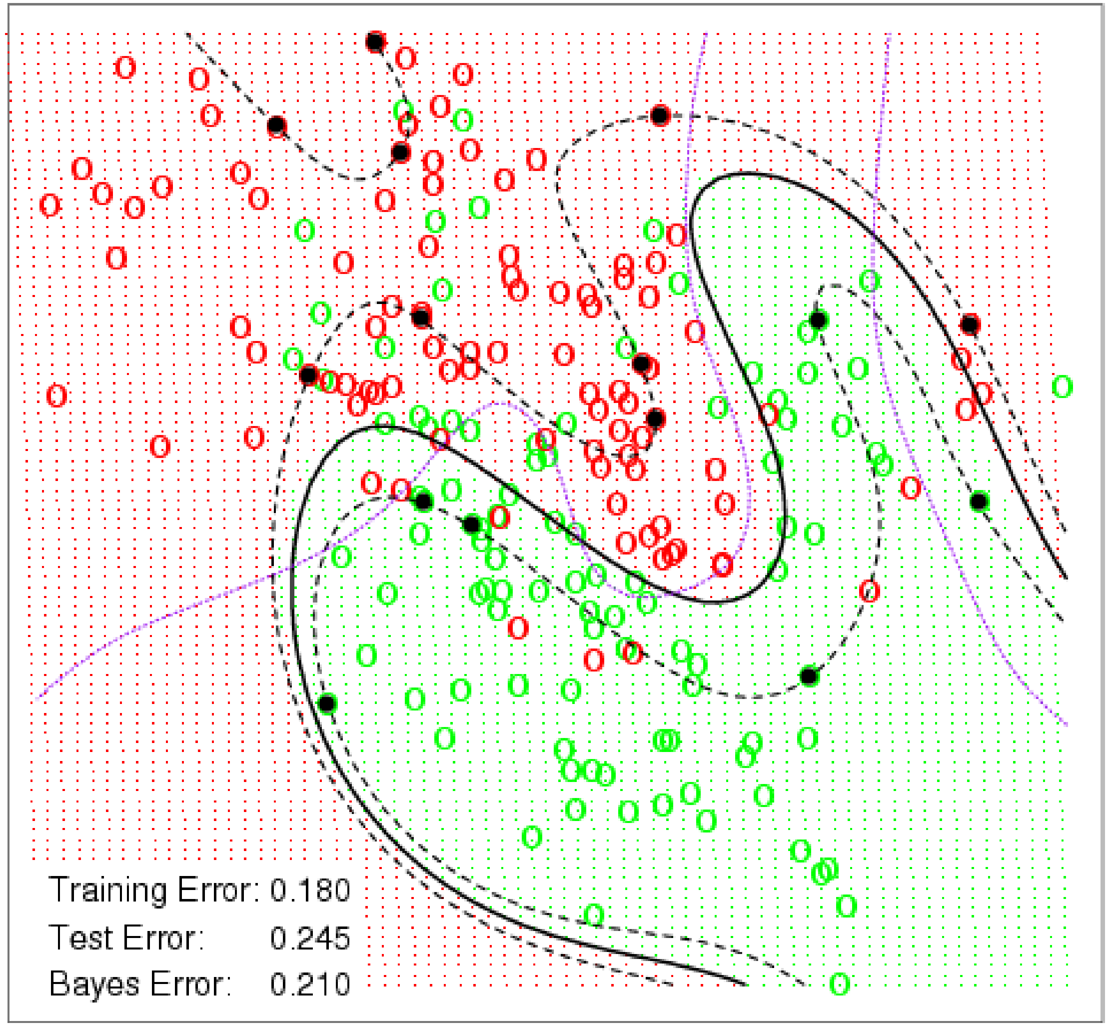
Using a polynomial kernel

- we now allow SVM to
- produce a non-linear decision boundary.

Notice that the test error rate

- is a lot lower.

SVM - Degree-4 Polynomial in Feature Space

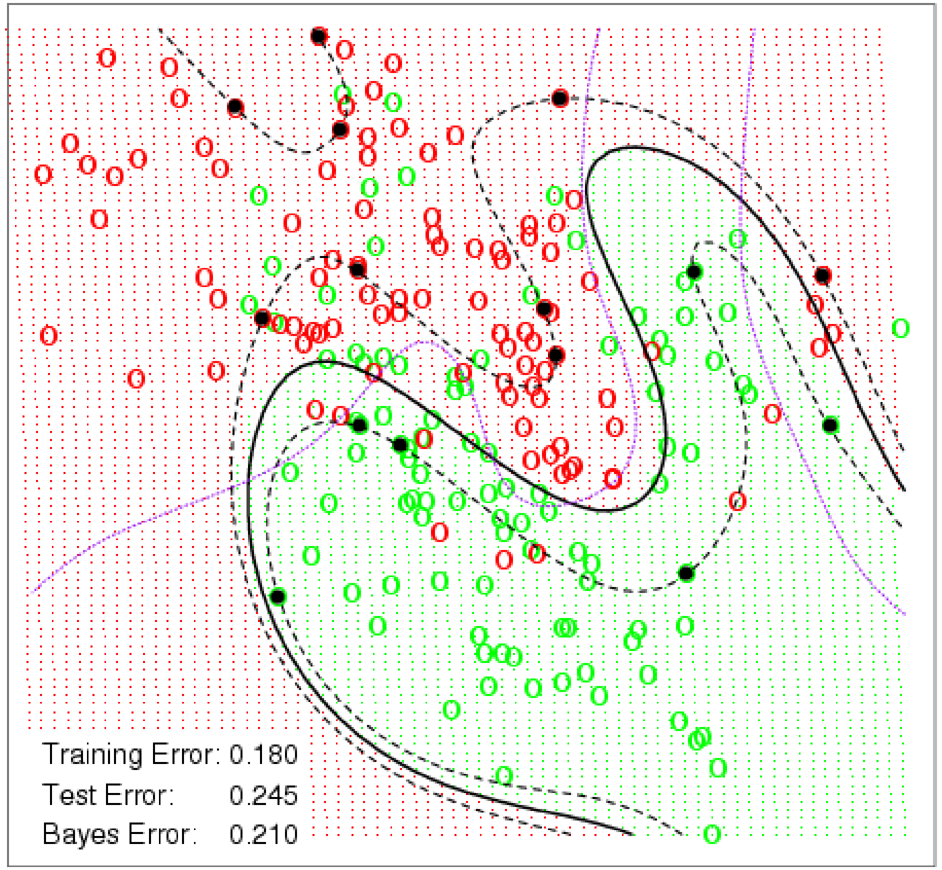


Radial Basis Kernel

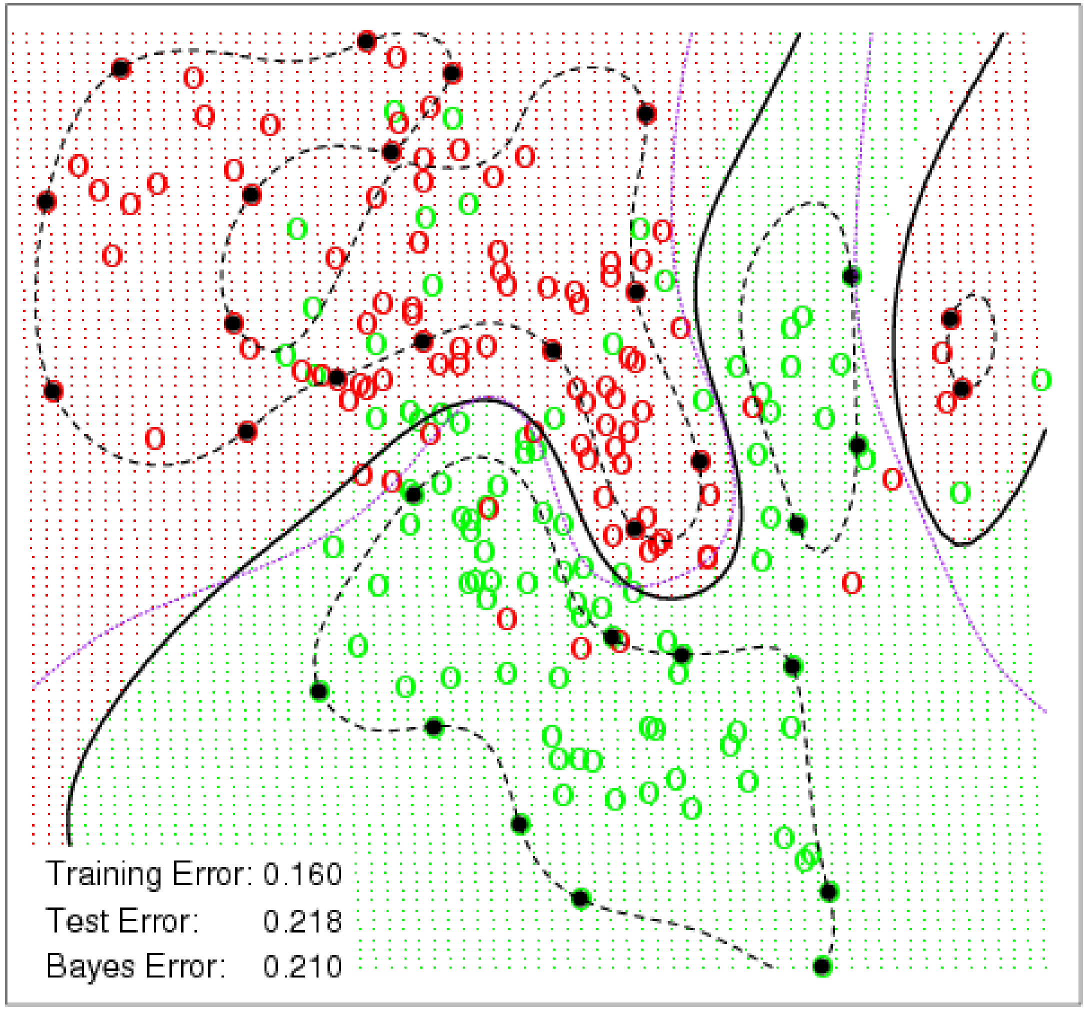
Using a Radial Basis Kernel

- you get an even lower error rate.

SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space



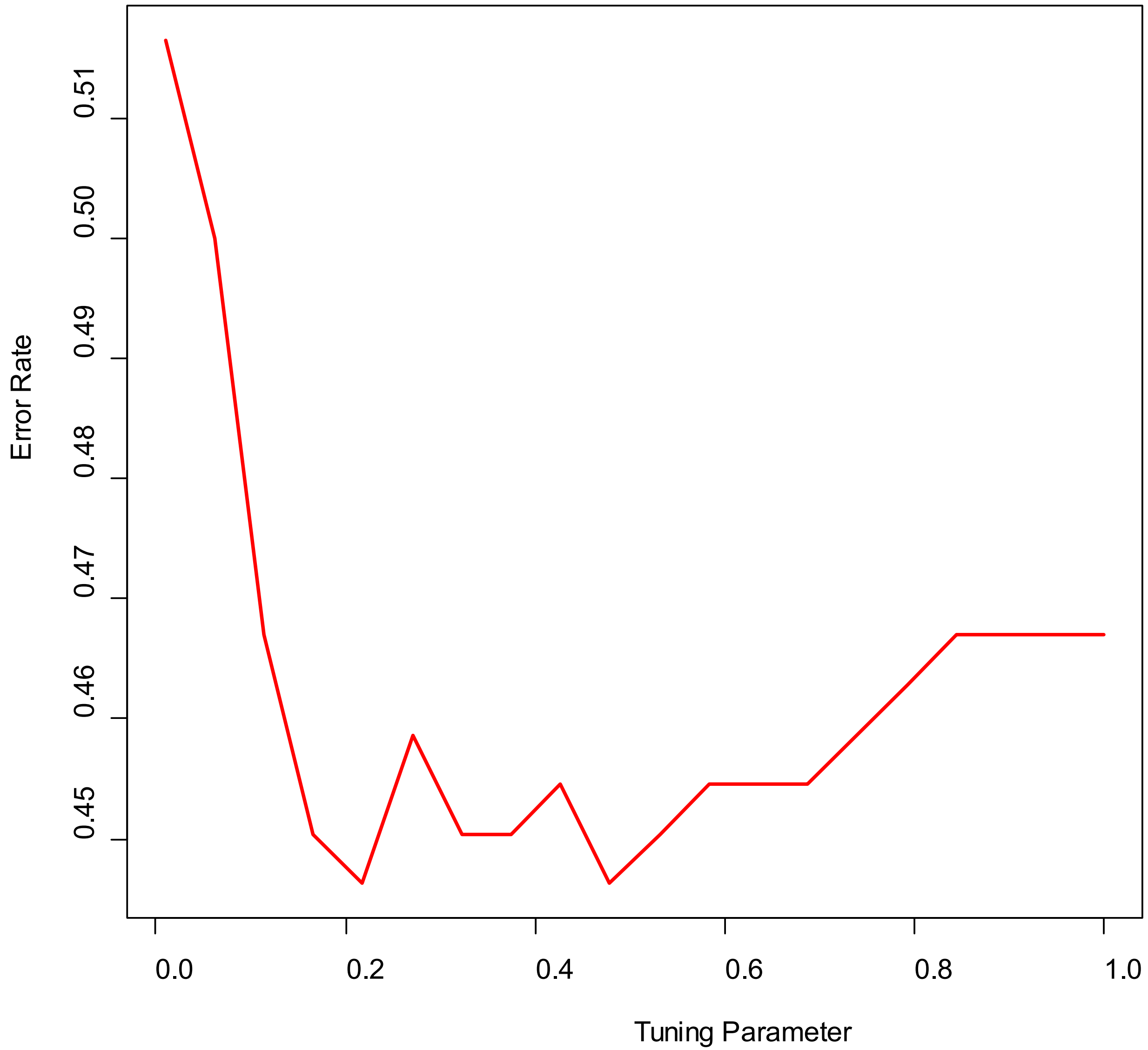
Error Rates on S&P Data

Here a Radial Basis Kernel is used and

- calculate the error rate
- for different values of the tuning parameter.

The results on this data were

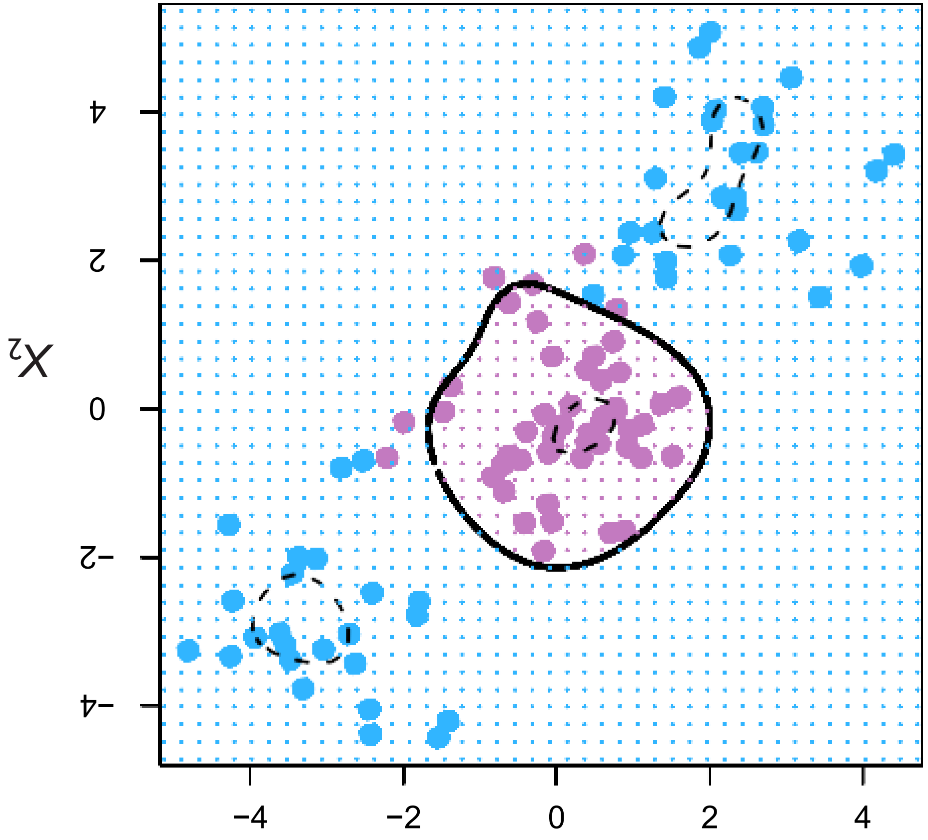
- similar to GAM
- but not as good as Boosting.



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

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.

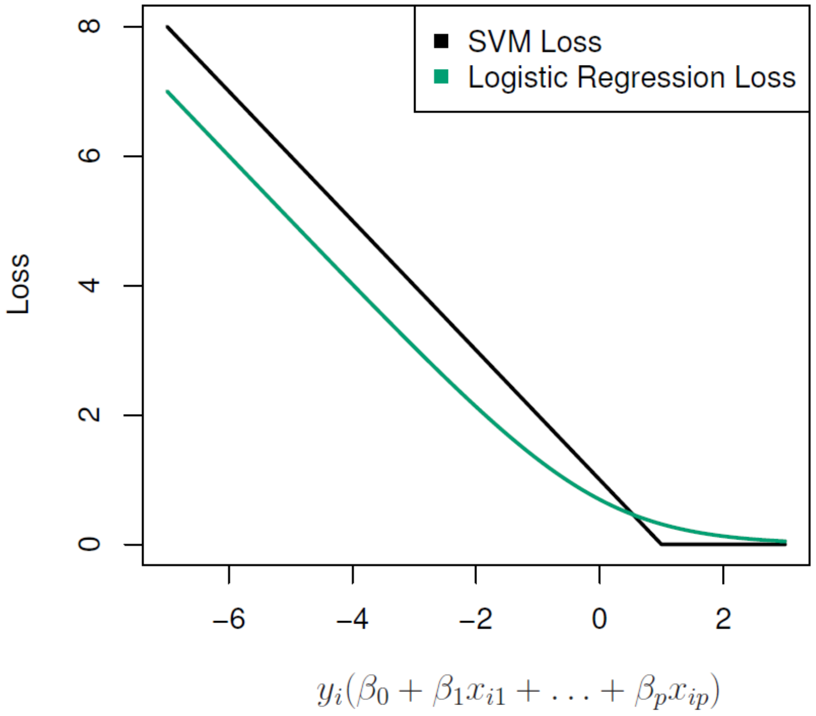
OVO One versus One. Fit all $\binom{K}{2}$ pairwise classifiers $\hat{f}_{k\ell}(x)$. Classify x^* to the class that wins the most pairwise competitions.

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

Support Vector versus Logistic Regression (LR)?

With $f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$ 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 Linear Discriminant Analysis (LDA).

When they aren't separable

- LR (with ridge penalty)
- and SVM very similar.

If you wish to <u>estimate probabilities</u>,

- 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.