Predictive Analytics (ISE529)

Support Vector Machines

Dr. Tao Ma
ma.tao@usc.edu

Tue/Thu, Aug 26 - Dec 6, 2024, Fall

USC Viterbi
School of Engineering
Daniel J. Epstein Department of Industrial and Systems Engineering

USC University of Southern California

Outline

- Maximal Margin Classifier
- Support Vector Classifiers
- Support Vector Machines

A New Approach for Classification

- Support vector machine (SVM), an approach for classification, was developed in the computer science community in the 1990s and has grown in popularity since then.

- The support vector machine is a generalization of two simple intuitive classifiers:
  - maximal margin classifier
  - support vector classifier

- People often loosely refer to the maximal margin classifier, the support vector classifier, and the support vector machine as "support vector machines".

MAXIMAL MARGIN CLASSIFIER

What is a Hyperplane?

In a p-dimensional space, a **hyperplane** is a flat *affine* subspace of dimension p-1. For instance,

- In two dimensions, a hyperplane is a flat one-dimensional subspace — a line.
- In three dimensions, a hyperplane is a flat two-dimensional subspace — that is, a plane.
- In $p > 3$ dimensions, it can be hard to visualize a hyperplane, but the notion of a (p-1)-dimensional flat subspace still applies.

The word *affine* indicates that the subspace need not pass through the origin.

What is a Hyperplane?

The mathematical definition of a *hyperplane* is quite simple. In two dimensions, a *hyperplane* is defined by the equation

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 \qquad (1)$$

where β_0, β_1, and β_2 are parameters. Any $X = (X_1, X_2)^{\mathrm{T}}$ for which equation (1) holds is a point on the hyperplane. Note that the equation (1) is simply the equation of a line.

If extended to the *p*-dimensional setting:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0 \qquad (2)$$

defines a *p*-dimensional *hyperplane*. Any point $X=(X_1, X_2, \ldots, X_p)^T$ in *p*-dimensional space (i.e., a vector of length *p*) satisfies equation (2), then *X* lies on the hyperplane.

A Separating Hyperplane

Now, suppose that X does not satisfy the equation, but rather,

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0.$$

Then this tells us that X lies to one side of the hyperplane. On the other hand, if

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0,$$

then X lies on the other side of the hyperplane. So, we can think of the hyperplane as dividing p-dimensional space into two halves.

One can easily determine on which side of the hyperplane a point lies by simply calculating the sign of the left-hand side of equation (2).

Classification with a Hyperplane

Suppose that we have an $n \times p$ data matrix X that consists of n training observations in p-dimensional space,

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \ldots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

and that these observations fall into two classes—that is, $y_1, \ldots, y_n \in \{-1, 1\}$ where -1 represents one class and 1 the other class. We also have a test observation, a p-vector of observed features $x^* = (x_1^* \ldots x_p^*)^T$.

Our goal is to develop a classifier with a *separating hyperplane* based on the training data that will correctly classify the test observation using its feature measurements.

Classification with a Hyperplane

Suppose that it is possible to construct a hyperplane that separates training observations perfectly according to their class labels.

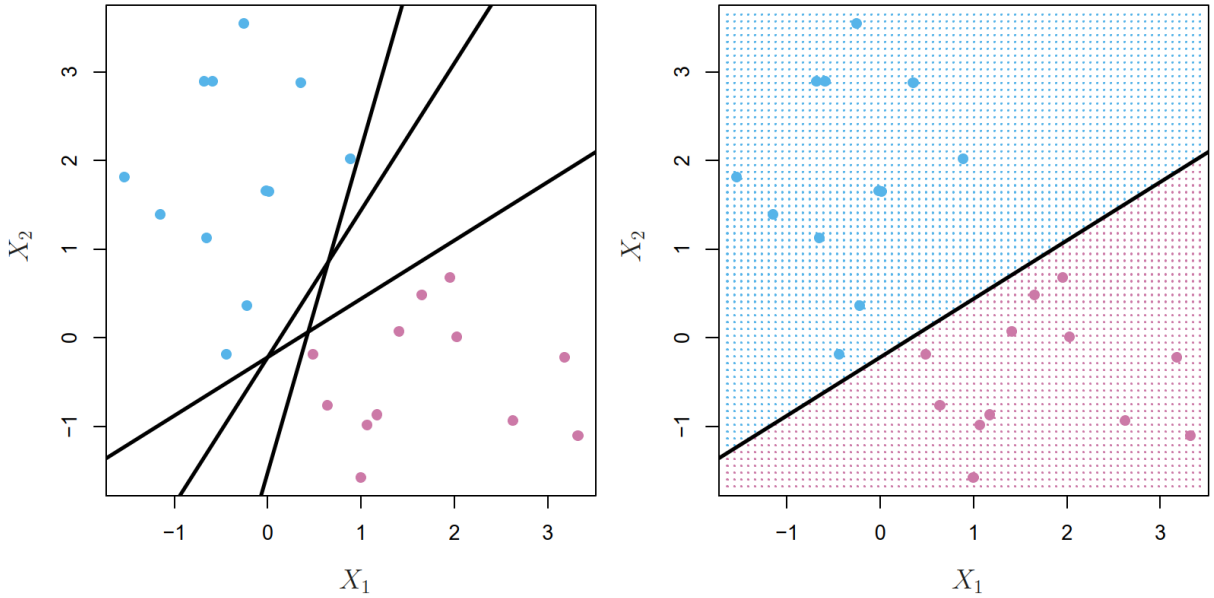


Figure 1

Label the observations from the blue class as $y_i = 1$ and those from the purple class as $y_i = -1$. Then a separating hyperplane has the property that

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0 \text{ if } y_i = 1,$$

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} < 0 \text{ if } y_i = -1.$$

Classification with a Hyperplane

Equivalently, a separating hyperplane has the property that

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0$$

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

If a separating hyperplane exists, i.e., $\beta_0, \beta_1, \ldots, \beta_p$ can be determined, then we can use it to construct a very natural classifier.

A test observation is assigned a class depending on which side of the hyperplane it is located.

The right-hand panel of Figure 1 shows an example of such a classifier.

Classification with a Hyperplane

If a separating hyperplane, i.e., $\beta_0, \beta_1, \ldots, \beta_p$, can be determined, then we classify the test observation x^* based on the sign of

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$$

If $f(x^*)$ is positive, then we assign the test observation to class 1, and if $f(x^*)$ is negative, then we assign it to class -1.

We can also make use of the **magnitude** of $f(x^*)$. If $f(x^*)$ is far from zero, then this means that x^* lies far from the hyperplane, and so we can be **confident** about our class assignment for x^*.

On the other hand, if $f(x^*)$ is close to zero, then x^* is located near the hyperplane, and so we are less certain about the class assignment for x^*.

Maximal Margin Classifier

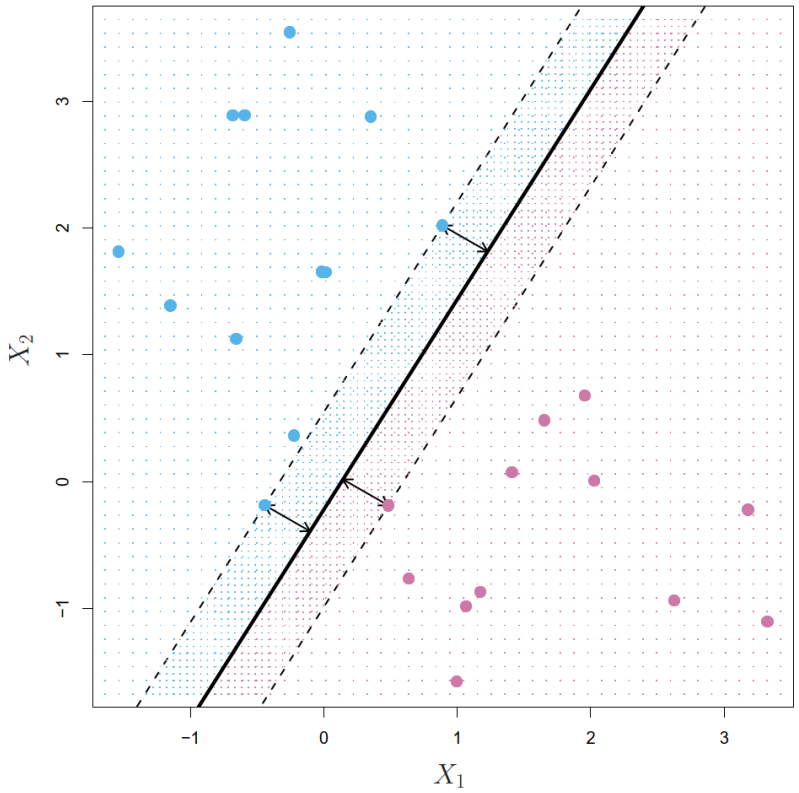
In general, if our data can be perfectly separated using a hyperplane, then there will in fact exist an infinite number of such hyperplanes. We must have a reasonable way to decide which of the infinite possible separating hyperplanes to use.

A natural choice is the ***maximal margin hyperplane*** (also known as optimal separating hyperplane), which is the separating hyperplane that is **farthest** from the training observations.

That is, we can compute the (perpendicular) distance from each training observation to a given separating hyperplane; the smallest such distance is the minimal distance from the observations to the hyperplane and is known as the **margin**.

Maximal Margin Classifier

The maximal margin hyperplane is the separating hyperplane for which the **margin is largest** — that is, it is the hyperplane that has the **farthest minimum distance** to the training observations.



The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the **support vectors**.

Maximal Margin Classifier

- The maximal margin hyperplane represents the **mid-line** of the **widest** "slab" that we can insert between the two classes. This is known as the *maximal margin classifier*.
- Three training observations are equidistant from the maximal margin hyperplane and lie along the dashed lines indicating the width of the margin. These three observations are known as **support vectors** in p-dimensional space.
- They "support" the maximal margin hyperplane in the sense that if these points were moved slightly then the maximal margin hyperplane would move as well.
- The maximal margin hyperplane depends **directly on the support vectors**, but **not on the other observations**: a movement to any of the other observations would not affect the separating hyperplane.
- The fact that the maximal margin hyperplane depends **directly on only a small subset** of the observations is an important property.

Maximal Margin Classifier

Construct the maximal margin hyperplane based on a set of n training observations $x_1, \ldots, x_n \in \mathrm{R}^p$ and associated class labels $y_1, \ldots, y_n \in \{-1, 1\}$.

Briefly, the maximal margin hyperplane is the solution to the optimization problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \; M \tag{9.9}$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1, \tag{9.10}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \;\; \forall\, i = 1, \ldots, n. \tag{9.11}$$

The constraint in (9.11)

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \;\; \forall\, i = 1, \ldots, n$$

guarantees that each observation will be on the correct side of the hyperplane, with some cushion, provided that M is positive.

Maximal Margin Classifier

Briefly, the maximal margin hyperplane is the solution to the optimization problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{\text{maximize}} \; M \tag{9.9}$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1, \tag{9.10}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \;\; \forall\, i = 1, \ldots, n. \tag{9.11}$$

The constraint in (9.10)

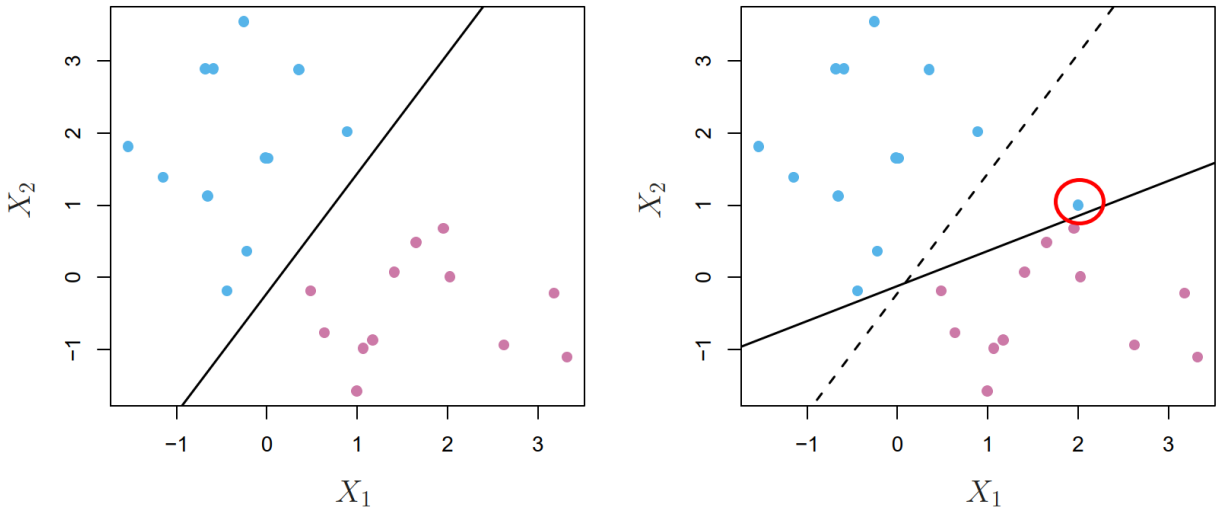
$$\sum_{j=1}^{p} \beta_j^2 = 1,$$

guarantees that there is unique solution, as well, the perpendicular distance from the *i*th observation to the hyperplane is given by

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}).$$

Maximal Margin Classifier

- Therefore, the constraints (9.10) and (9.11) ensure that each observation is on the correct side of the hyperplane and at least a distance M from the hyperplane.
- Hence, M represents the margin of our hyperplane, and the optimization problem chooses $\beta_0, \beta_1, \ldots, \beta_p$ to maximize M.
- Although the maximal margin classifier is often successful, it can also lead to **overfitting** when p is large due to sensitivity to individual observations



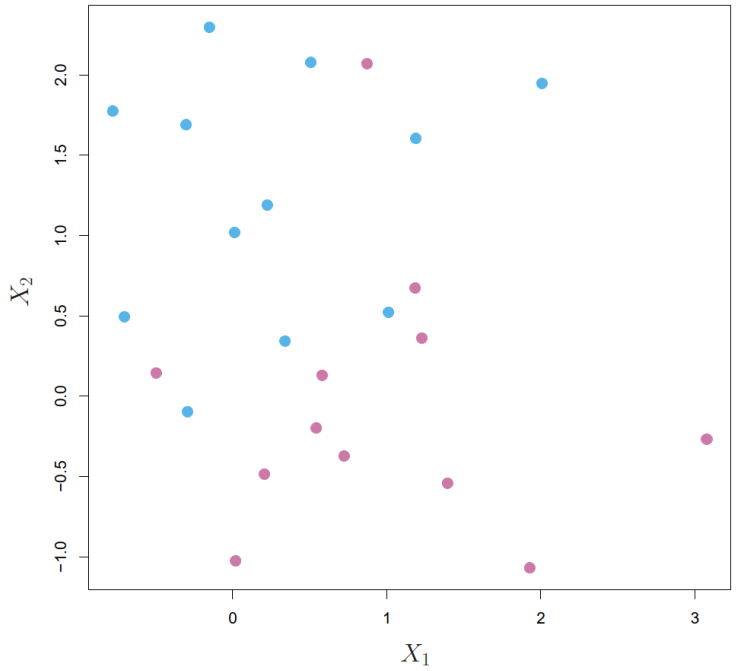
Right: An additional blue observation has been added, leading to a dramatic shift in the maximal margin hyperplane shown as a solid line.

SUPPORT VECTOR CLASSIFIERS

Issue with Non-separable Case

Maximal margin classifier unfortunately cannot be applied to most data sets, since it requires that the classes be separable by a linear boundary.

In many cases no separating hyperplane exists, and so there is no maximal margin classifier. In this case, the optimization problem (9.9)–(9.11) has no solution with $M > 0$.



For example, there are two classes of observations, shown in blue and in purple. In this case, the two classes are **not separable** by a hyperplane, and so the maximal margin classifier cannot be used.

Support Vector Classifier

In this case, we might be willing to consider a classifier based on a hyperplane that *does not perfectly* separate the two classes, in the interest of

- greater robustness to individual observations, and
- better classification of most of the training observations.

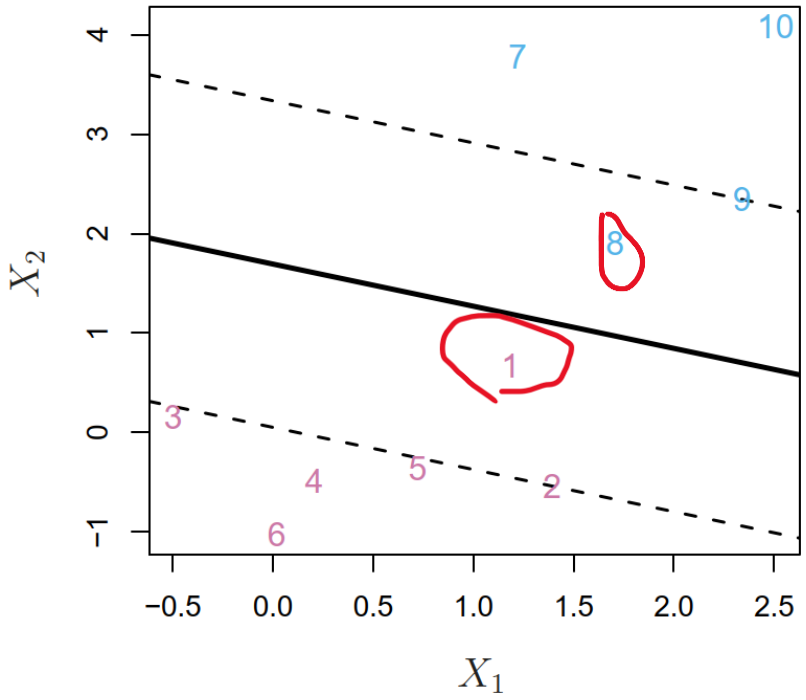
That is, it could be worthwhile to misclassify a few training observations to do a better job in classifying the remaining observations.

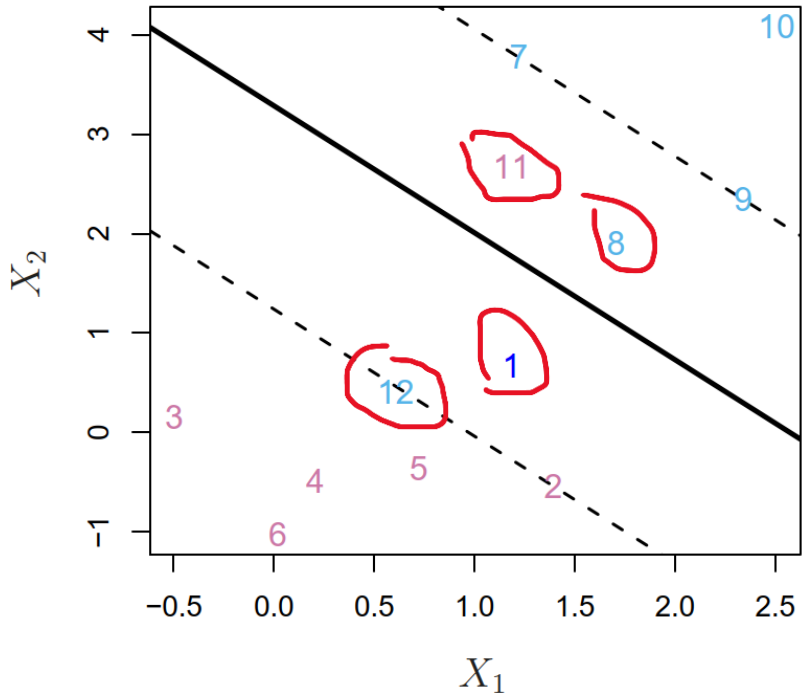
Developing a hyperplane that ***almost*** separates the classes is to use a so-called **soft margin**.

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

Support Vector Classifier

The *support vector classifier*, sometimes called a *soft margin classifier*, allows some observations to be on the incorrect side of the margin, or even the incorrect side of the hyperplane. (The margin is ***soft*** because it can be violated by some of the training observations.)





Support Vector Classifier

The hyperplane is chosen to correctly separate **most** of the training observations into the two classes, but may misclassify a few observations. It is the solution to the optimization problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,\,M}{\text{maximize}} \quad M \tag{9.12}$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \tag{9.13}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \tag{9.14}$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C, \tag{9.15}$$

where C is a nonnegative tuning parameter. M is the width of the margin. In (9.14), $\epsilon_1, \ldots, \epsilon_n$ are *slack variables* indicating the distance from the margin that individual observations are on the wrong side of the margin or the hyperplane.

Parameters

The slack variable ϵ_i tells us where the ith observation is located, relative to the hyperplane and relative to the margin.

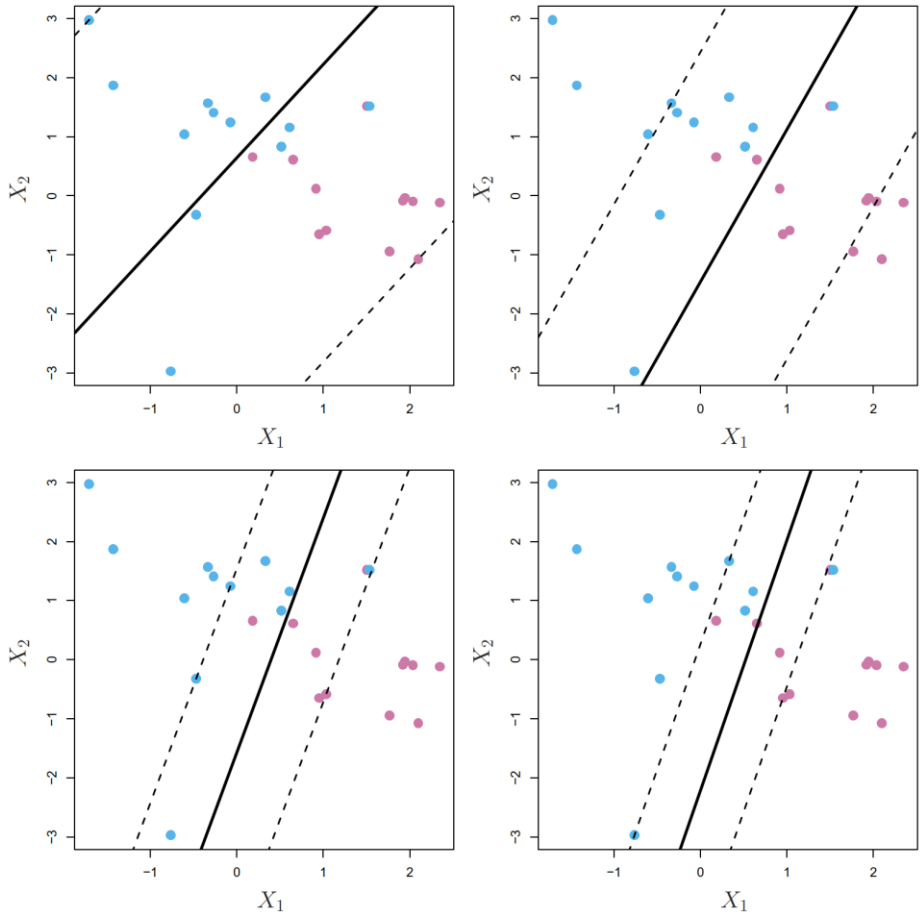
- If $\epsilon_i = 0$ then the ith observation is on the correct side of the margin.
- If $\epsilon_i > 0$ then the ith observation is on the wrong side of the margin.
- If $\epsilon_i > 1$ then it is on the wrong side of the hyperplane.

In (9.15), C bounds the sum of the ϵ_i's, and so it determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate.

- If $C = 0$ then there is no budget for violations to the margin, and it must be the case that $\epsilon_1 = \cdots = \epsilon_n = 0$, in which case (9.12)–(9.15) simply amounts to the maximal margin hyperplane (**only if** the two classes are separable).
- For $C > 0$ no more than C observations can be on the wrong side of the hyperplane ($\epsilon_i > 1$).
- As C increases, the margin will widen; as C decreases, the margin narrows.

Support Vectors

Only observations that either lie on the margin or that violate the margin will affect the hyperplane. Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as **support vectors**.



A support vector classifier was fit using **four** different values of the tuning parameter *C*.

When *C* is large, more support vectors, the margin will be large. (top-left)

As *C* decreases, less support vectors, the margin narrows. (bottom-right)

Bias-Variance Trade-off

In practice, *C* is treated as a tuning parameter that is generally chosen via *k*-fold cross-validation.

C controls the bias-variance trade-off.

When *C* is small, we seek narrow margins that are rarely violated; this amounts to a classifier that is highly fit to the data, which may have **low bias but high variance**.

When *C* is larger, the margin is wider, and we allow more violations to it; this amounts to fitting the data less hard and obtaining a classifier that is potentially **more biased but** may have **lower variance**.

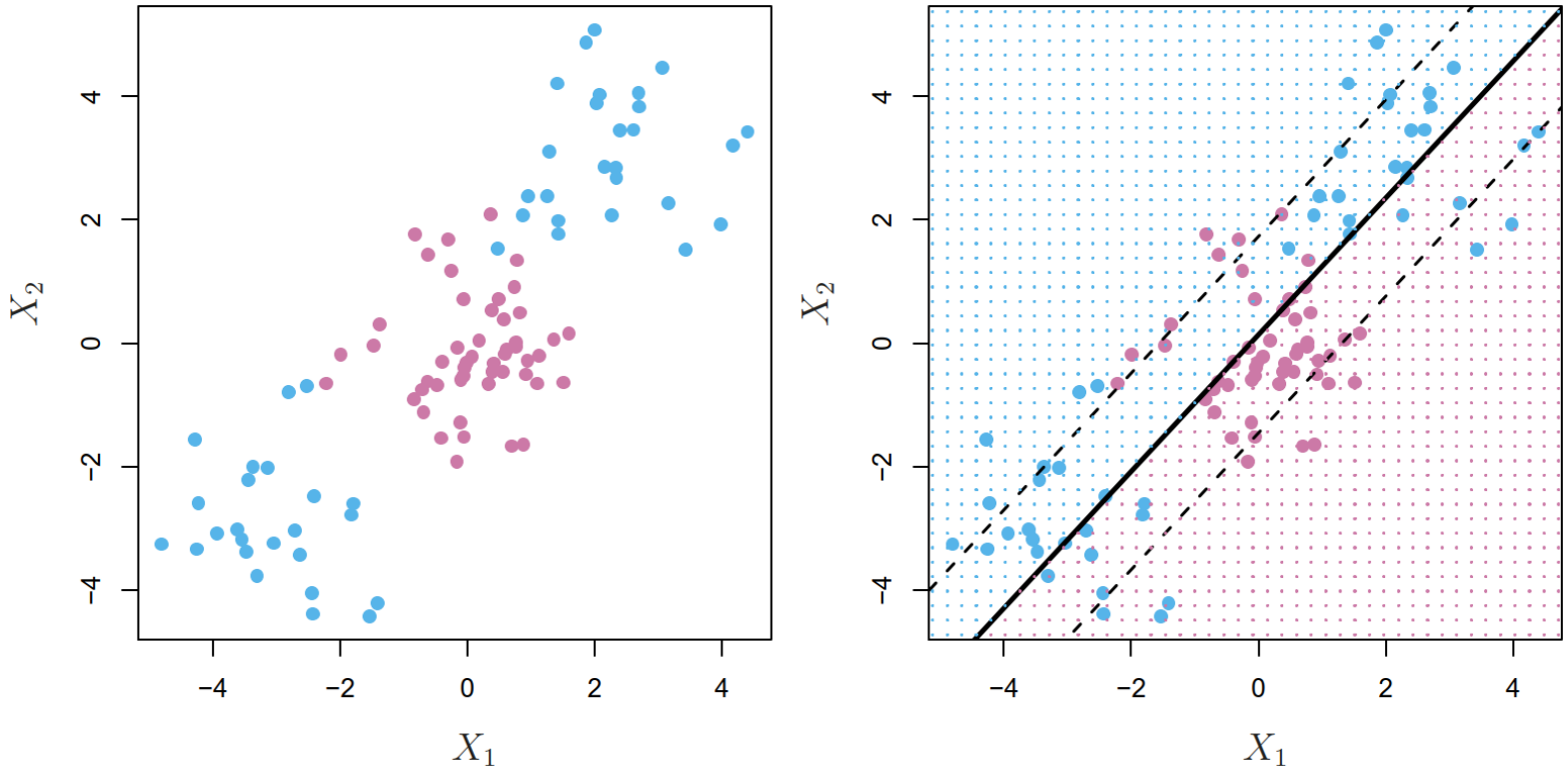
Once we have solved (9.12)–(9.15), we classify the test observation based on the sign of

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$$

SUPPORT VECTOR MACHINES

Non-Linear Decision Boundaries

The limitation of maximal margin classifier and support vector classifier is that they can only classify the data with ***linear*** class boundary.



To accommodate non-linear class boundaries, we discuss a general mechanism for converting a linear classifier into one that produces non-linear decision boundaries.

Non-Linear Decision Boundaries

- With an analogous situation, how do we make linear regression to fit a nonlinear relationship between the predictors and the outcome?

- We enlarge the feature space using functions of the predictors, such as quadratic and cubic terms, polynomial terms, to address this non-linearity.

- In the case of the support vector classifier, we could address the problem of possibly non-linear boundaries between classes **in a similar way**, by enlarging the feature space using quadratic, cubic, and even higher-order polynomial functions of the predictors.

- For instance, we could fit a support vector classifier using $2p$ features

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

Non-Linear Decision Boundaries

Then (9.12)–(9.15) would become

$$\underset{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n,\,M}{\text{maximize}} \quad M \tag{9.16}$$

$$\text{subject to } y_i\left(\beta_0+\sum_{j=1}^{p}\beta_{j1}x_{ij}+\sum_{j=1}^{p}\beta_{j2}x_{ij}^2\right)\geq M(1-\epsilon_i),$$

$$\sum_{i=1}^{n}\epsilon_i\leq C,\ \ \epsilon_i\geq 0,\ \ \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2=1.$$

In the enlarged feature space, the decision boundary that results from (9.16) is in fact linear. But in the **original feature space**, the decision boundary is of the form of a quadratic polynomial, and its solutions are generally non-linear.

Support Vector Machine

The support vector machine (SVM) is an extension of the support vector classifier that results from enlarging the feature space in a specific way, **using kernels**.

Kernel approach is simply an efficient computational approach for enlarging our feature space to accommodate a non-linear boundary between the classes.

Skipping the technical details of how the support vector classifier is computed, briefly, the **solution** to the support vector classifier problem (9.12)–(9.15) involves **only the inner products** of the observations.

For instance, the inner product of two observations x_i, $x_{i'}$ is given by

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

With Inner Product

- 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 there are n parameters α_i, $i = 1, \ldots, n$, one per training observation.

- To estimate the parameters $\alpha_1, \ldots, \alpha_n$ and β_0, all we need are the $\binom{n}{2} = n(n-1)/2$ inner products $\langle x_i, x_{i'} \rangle$ between all pairs of training observations.

- However, it turns out that α_i is nonzero only for the **support vectors** in the solution — that is, if a training observation is not a support vector, then its α_i equals zero.

With Kernel Function

A generalization of the inner product of the form

$$K(x_i, x_{i'})$$

where *K* is a function that is referred to as a kernel. Then support vector classifier can be represented as

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

When the kernel function takes the form

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

f (x) is a linear support vector classifier.

Support Vector Machine

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

When the kernel function takes the form

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

known as a polynomial kernel of degree *d*, where *d* is a positive integer and $d > 1$ for non-linear. Then the resulting classifier $f(x)$ is known as a **support vector machine**. Another popular choice is the radial kernel, which takes the form

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

where γ is a positive constant.

Radial Kernel Behavior

How does the radial kernel actually work?

If a given test observation $x^* = (x_1^*, \ldots, x_p^*)^T$ is far from a training observation x_i in terms of Euclidean distance, then $\sum_{j=1}^{p}(x_j^* - x_{ij})^2$ will be large, and so $K(x^*, x_i) = \exp(-\gamma \sum_{j=1}^{p}(x_j^* - x_{ij})^2)$ will be tiny. This means that x_i will play virtually no role in $f(x^*)$.

In other words, training observations that are far from x^* will play essentially no role in the predicted class label for x^*. This means that the radial kernel has very ***local*** behavior, in the sense that **only nearby** training observations have an effect on the class label of a test observation.

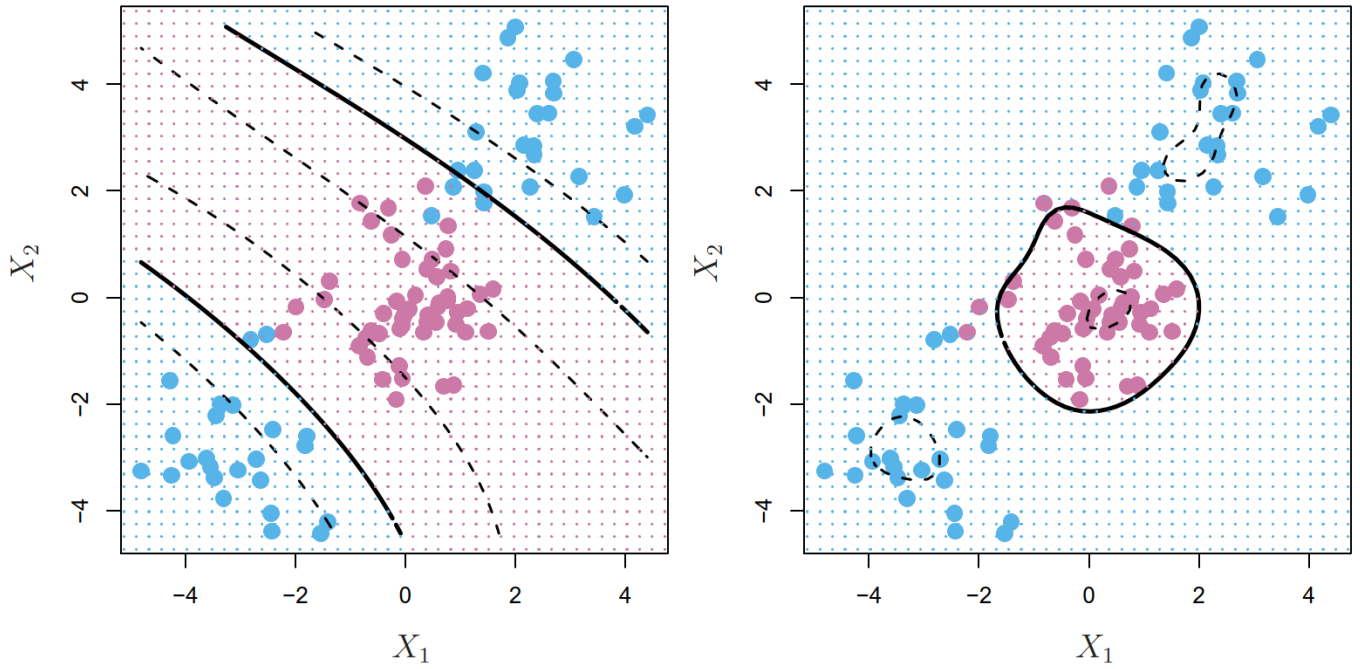
Advantage of Kernel

What is the advantage of using a **kernel** rather than simply enlarging the feature space using functions of the original features, as in (9.16)?

The greatest advantage is computational, and it amounts to the fact that using kernels, one needs only compute $K(x_i, x_{i'})$ for all $\binom{n}{2} = n(n-1)/2$ distinct pairs i, i'.

This can be done without explicitly working in the enlarged feature space. This is important because in many applications of SVMs, the enlarged feature space is so large that computations are intractable.

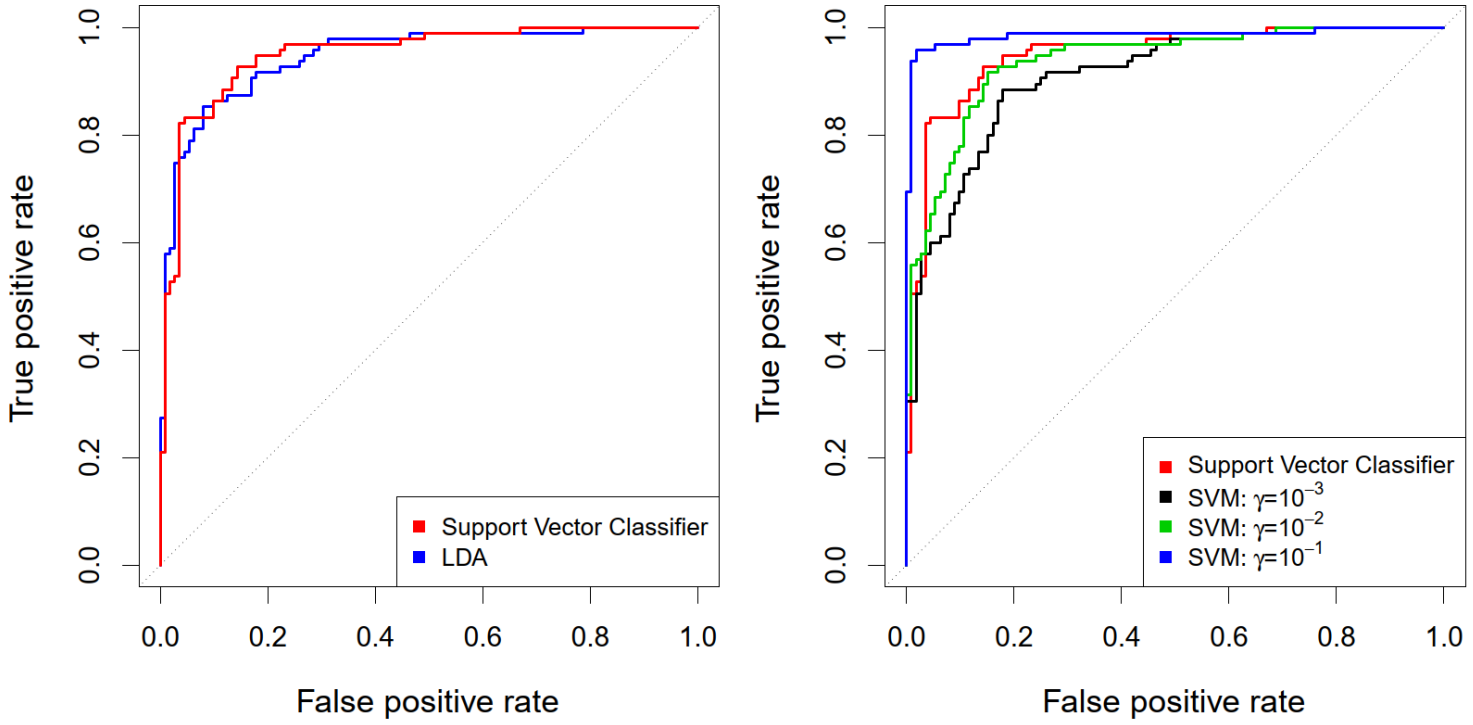
Example



Left: An SVM with a polynomial kernel of degree 3 is applied to the non-linear data.
Right: An SVM with a radial kernel is applied.

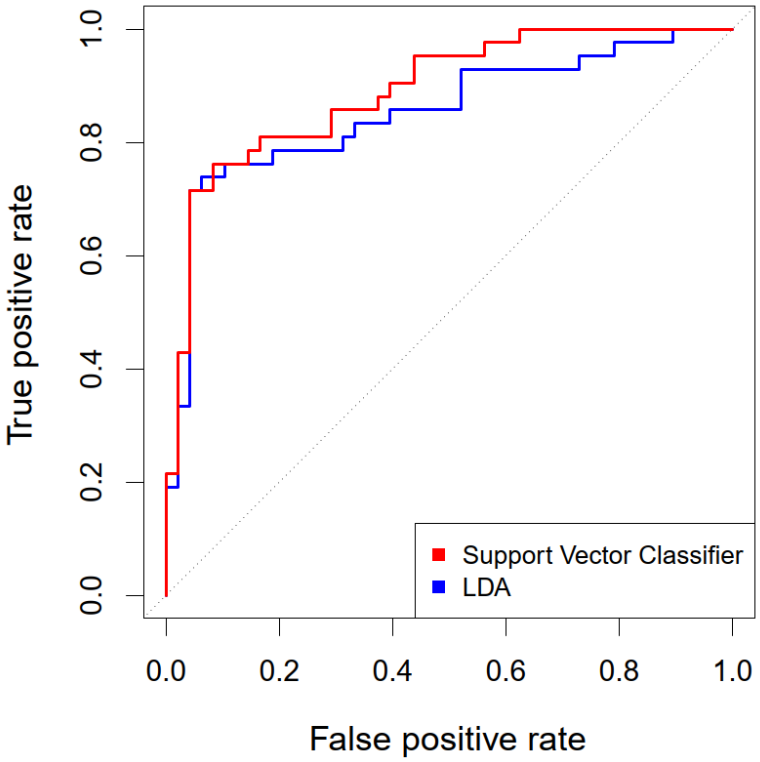
In this example, either kernel is capable of capturing the decision boundary.

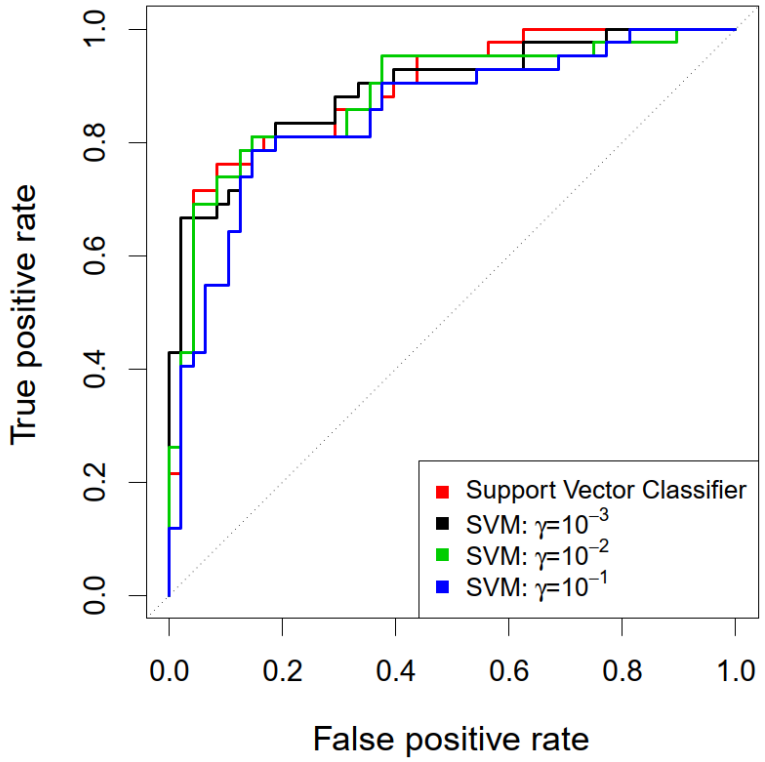
Application to the Heart Disease Data



ROC curves for the Heart data **training set**.
Left: The support vector classifier and LDA are compared.
Right: The support vector classifier is compared to an SVM using a radial basis kernel. Using $\gamma = 10^{-1}$ appears to give an almost perfect ROC curve.

Application to the Heart Disease Data





ROC curves for the **test set** of the Heart data.
Left: The support vector classifier and LDA are compared.
Right: The support vector classifier is compared to an SVM using a radial basis kernel. Using $\gamma = 10^{-1}$ produces the worst estimates on the test data.

SVMs with More than Two Classes

SVM is for classification in the two-class setting.

Though a number of proposals for extending SVMs to the *K*-class case have been made, the two most popular are the

- one-versus-one
- one-versus-all approaches

One-Versus-One Classification

Suppose that we would like to perform classification using SVMs, and there are $K > 2$ classes. A *one-versus-one* or *all-pairs* approach constructs $\binom{K}{2}$ SVMs, each of which compares a pair of classes.

For example, one such SVM might compare the kth class, coded as +1, to the k'th class, coded as −1.

We classify a test observation using each of the $\binom{K}{2}$ classifiers, and we tally the number of times that the test observation is assigned to each of the K classes.

The final classification is performed by assigning the test observation to the class to which it was most frequently assigned in these pairwise classifications.

One-Versus-All Classification

The one-versus-all approach (also referred to as one-versus-rest) is an alternative procedure for applying SVMs in the case of $K > 2$ classes. We fit K SVMs, each time comparing one of the K classes to the remaining $K - 1$ classes.

Let $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$ denote the parameters that result from fitting an SVM comparing the kth class (coded as +1) to the others (coded as −1).

Let x^* denote a test observation.

We assign the observation to the class for which

$$\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \cdots + \beta_{pk}x_p^*$$

is **largest**, as this amounts to a high level of confidence that the test observation belongs to the kth class rather than to any of the other classes.