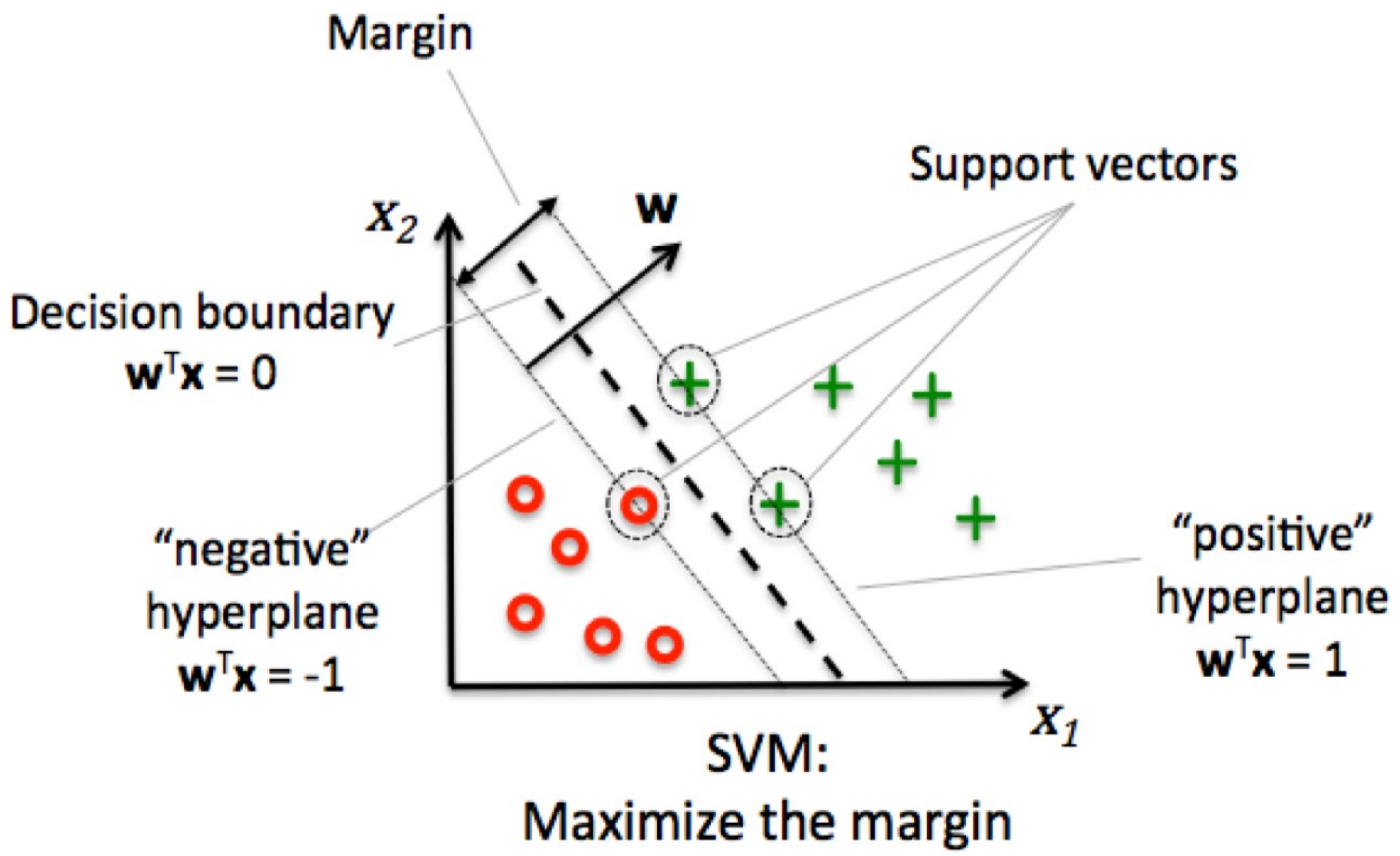
INF 552, Machine Learning for Data Science

University of Southern California

M. R. Rajati, PhD

Lesson 7
Support Vector Machines



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

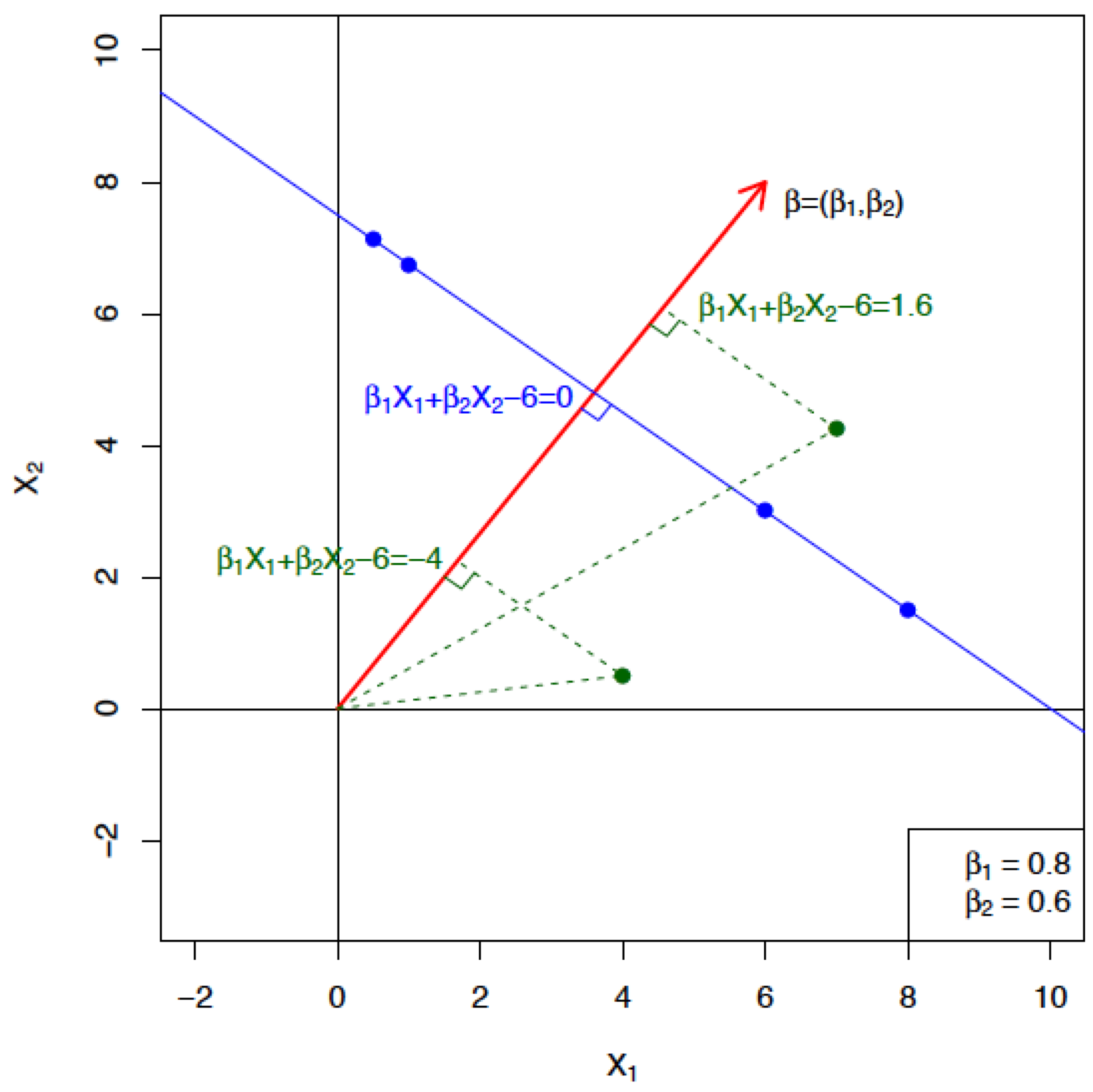
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$

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

What is a Hyperplane?

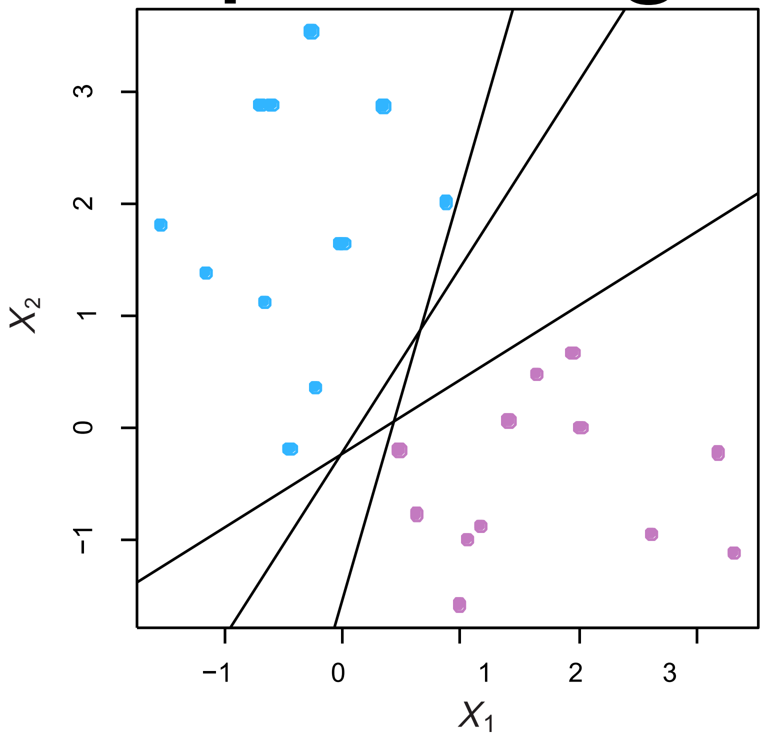
- If $\beta_0 = 0$, the hyperplane goes through the origin, otherwise not.
- The vector $\beta = (\beta_1, \beta_2, \cdots, \beta_p)$ is called the normal vector — it points in a direction orthogonal to the surface of a hyperplane.

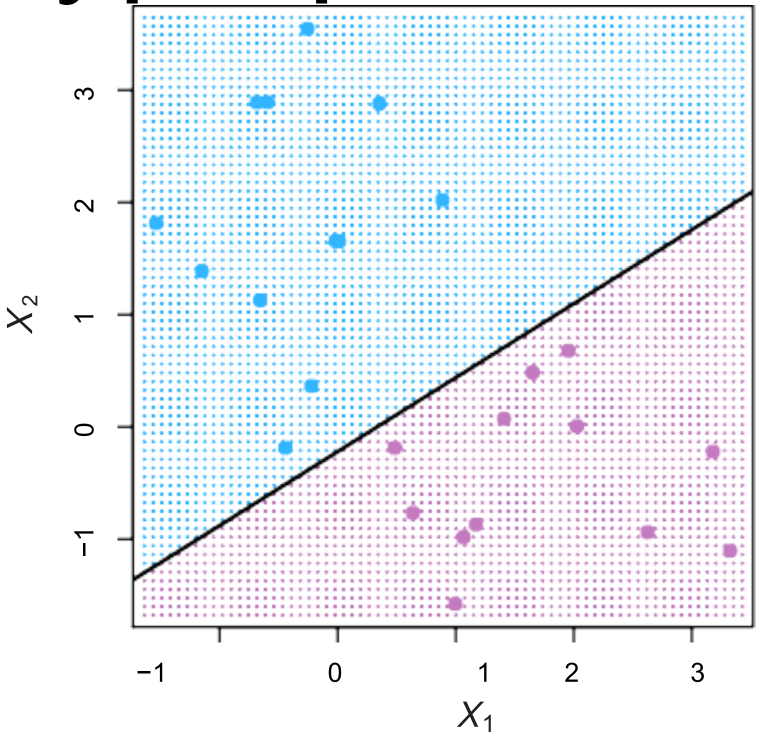
Hyperplane in 2 Dimensions



X_1

Separating Hyperplanes

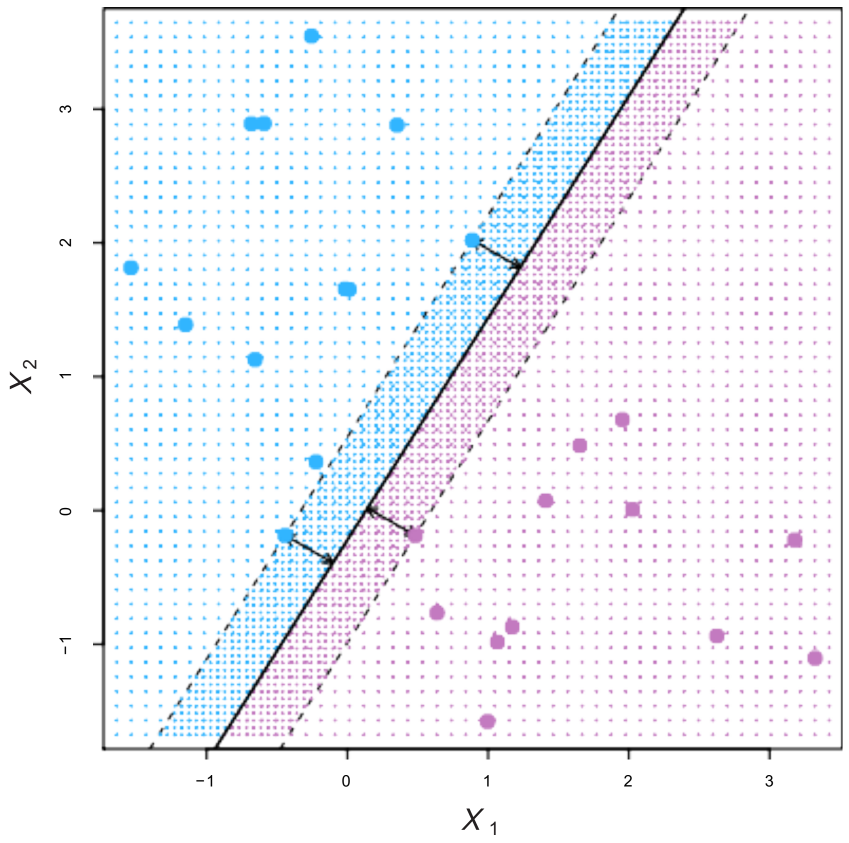




- If $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$, then $f(X) > 0$ for points on one side of the hyperplane, and $f(X) < 0$ for points on the other.
- If we code the colored points as $Y_i = +1$ for blue, say, and $Y_i = -1$ for mauve, then if $Y_i \cdot f(X_i) > 0$ for all i, $f(X) = 0$ defines 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

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

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

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

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

This can be rephrased as a convex quadratic program, and solved efficiently.

Maximal Margin Classifier

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

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

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

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

The constraint $\sum_{j=1}^{p}\beta_j^2 = 1$ makes sure that a unique solution for β_i's exists.

Combined with

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

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

it guarantees that the minimum margin is *M* if *M* is positive.

Maximal Margin Classifier

For each observation to be on the correct side of the hyperplane we would simply need

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

so the constraint

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

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

in fact requires that each observation be on the correct side of the hyperplane, *with some cushion*, provided that *M* is positive.

Maximal Margin Classifier

Since if

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} = 0$$

Defines a hyperplane, then

$$k(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) = 0$$

also defines a hyperplane for any nonzero *k*

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

guarantees a unique set of β_i's exists,

Maximal Margin Classifier

Moreover, the constraint

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

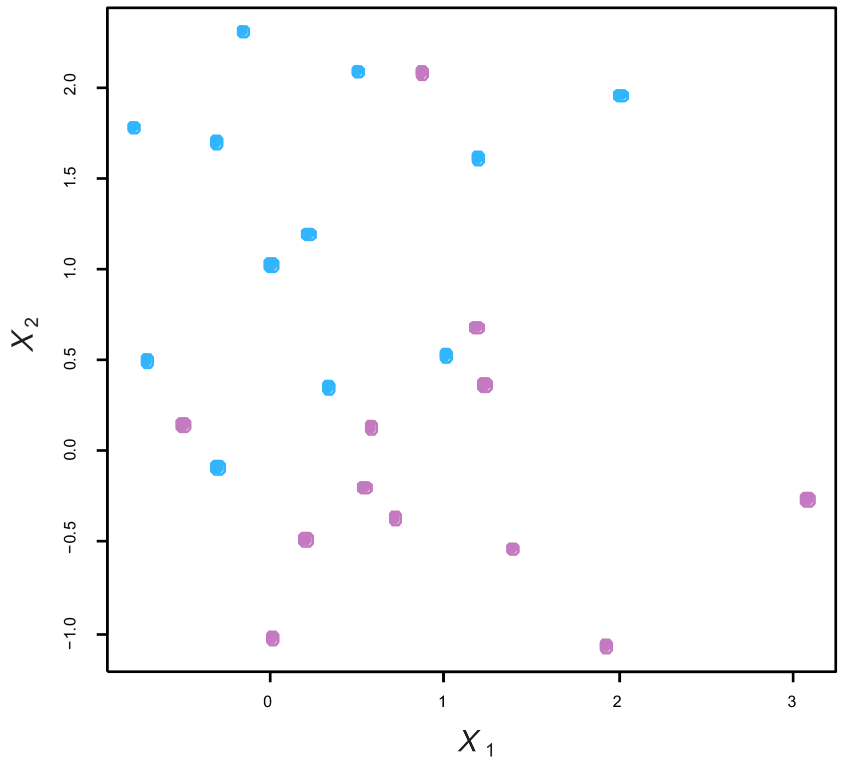
adds meaning to

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$
$$\text{for all } \; i = 1, \ldots, N.$$

one can show that with this constraint the perpendicular distance from the *i* th observation to the hyperplane is given by

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

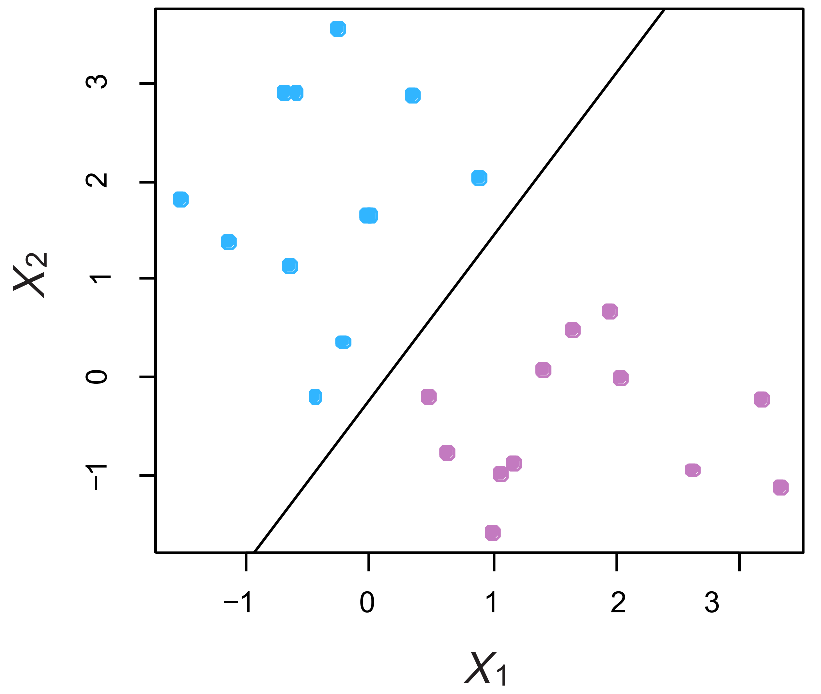
Non-separable Data

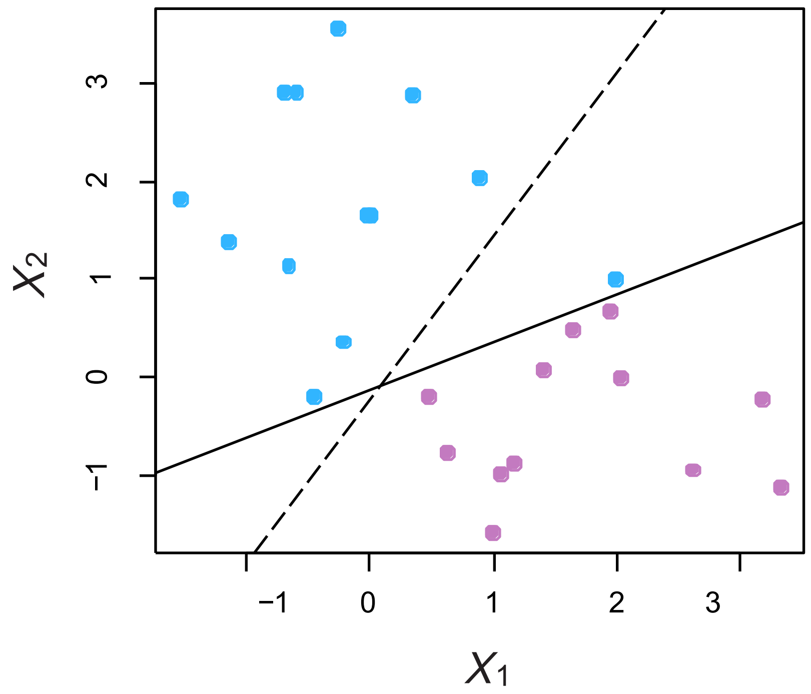


The data on the left are not separable by a linear boundary.

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

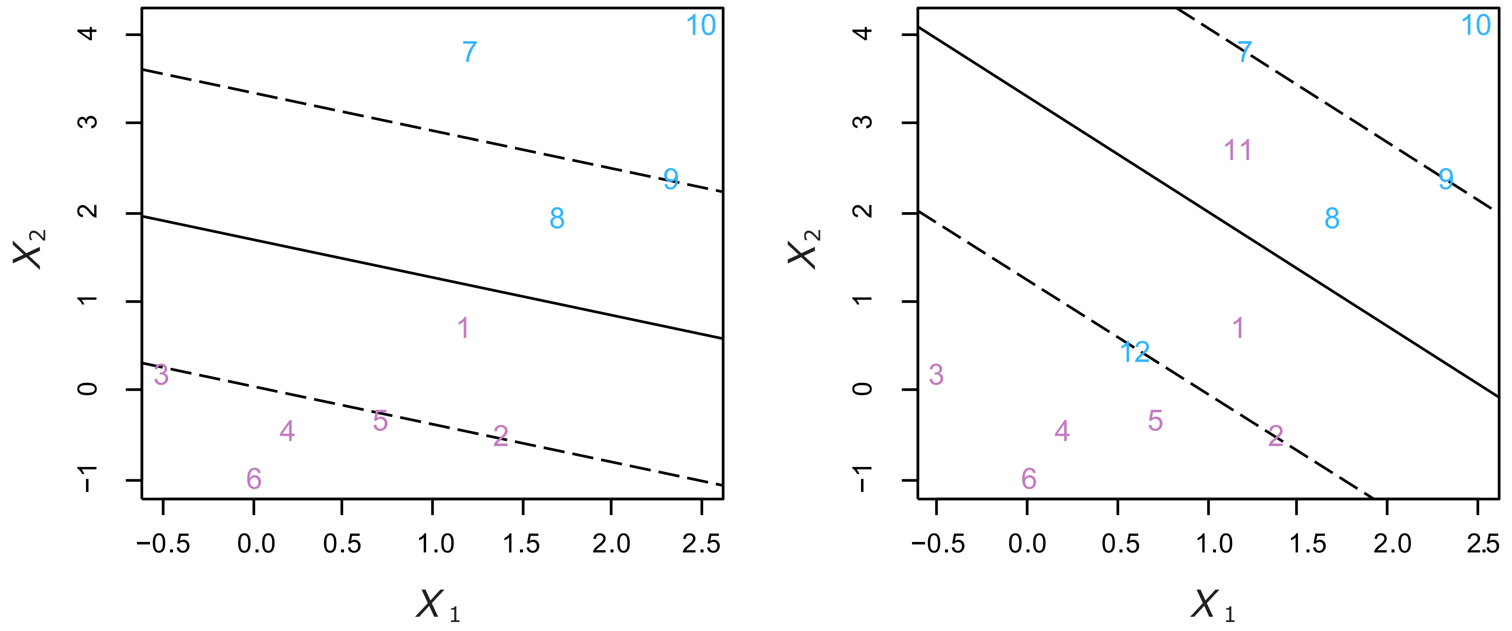
Noisy Data





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



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

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

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

Support Vector Classifier

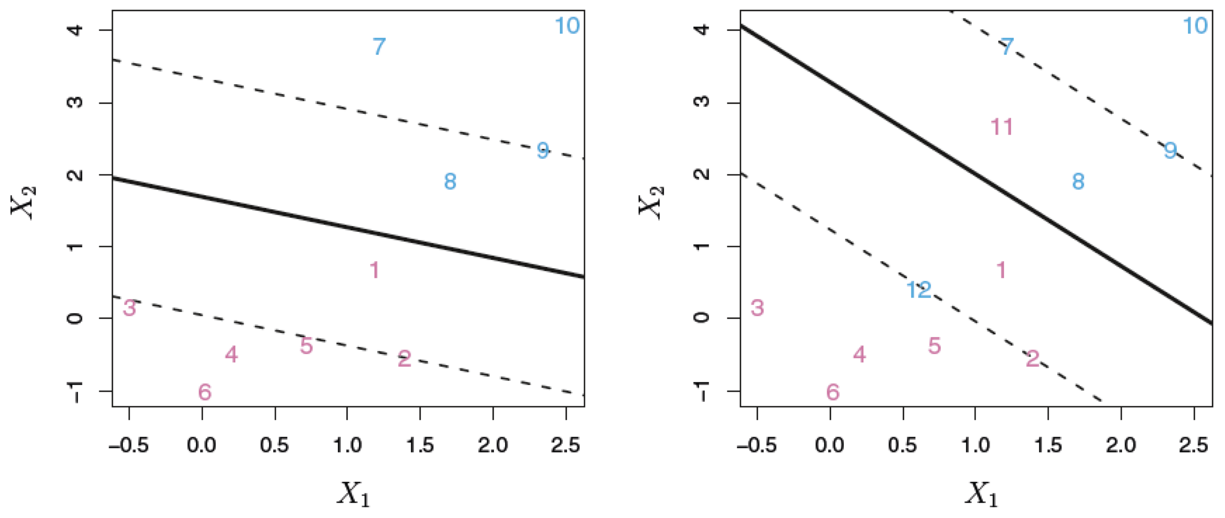


FIGURE 9.6. Left: *A support vector classifier was fit to a small data set. The hyperplane is shown as a solid line and the margins are shown as dashed lines.* Purple observations: *Observations* 3, 4, 5, *and* 6 *are on the correct side of the margin, observation* 2 *is on the margin, and observation 1 is on the wrong side of the margin.* Blue observations: *Observations* 7 *and* 10 *are on the correct side of the margin, observation* 9 *is on the margin, and observation* 8 *is on the wrong side of the margin. No observations are on the wrong side of the hyperplane.* Right: *Same as left panel with two additional points,* 11 *and* 12*. These two observations are on the wrong side of the hyperplane and the wrong side of the margin.*

Details of Optimization Problem

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

- If $\varepsilon_i = 0$ then the ith observation is on the correct side of the margin

Details of Optimization Problem

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

- If $\varepsilon_i > 0$ then the *i*th observation is on the wrong side of the margin, and we say that the *i*th observation has *violated* the margin.

Details of Optimization Problem

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

- If $\varepsilon_i > 1$ then it is on the wrong side of the hyperplane.

Details of Optimization Problem

- *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.

Details of Optimization Problem

- We can think of C as a budget for the amount that the margin can be violated by the N observations.
- If $C = 0$ then there is no budget for violations to the margin, so all ε_i's = 0, which leads to the maximal margin hyperplane optimization problem

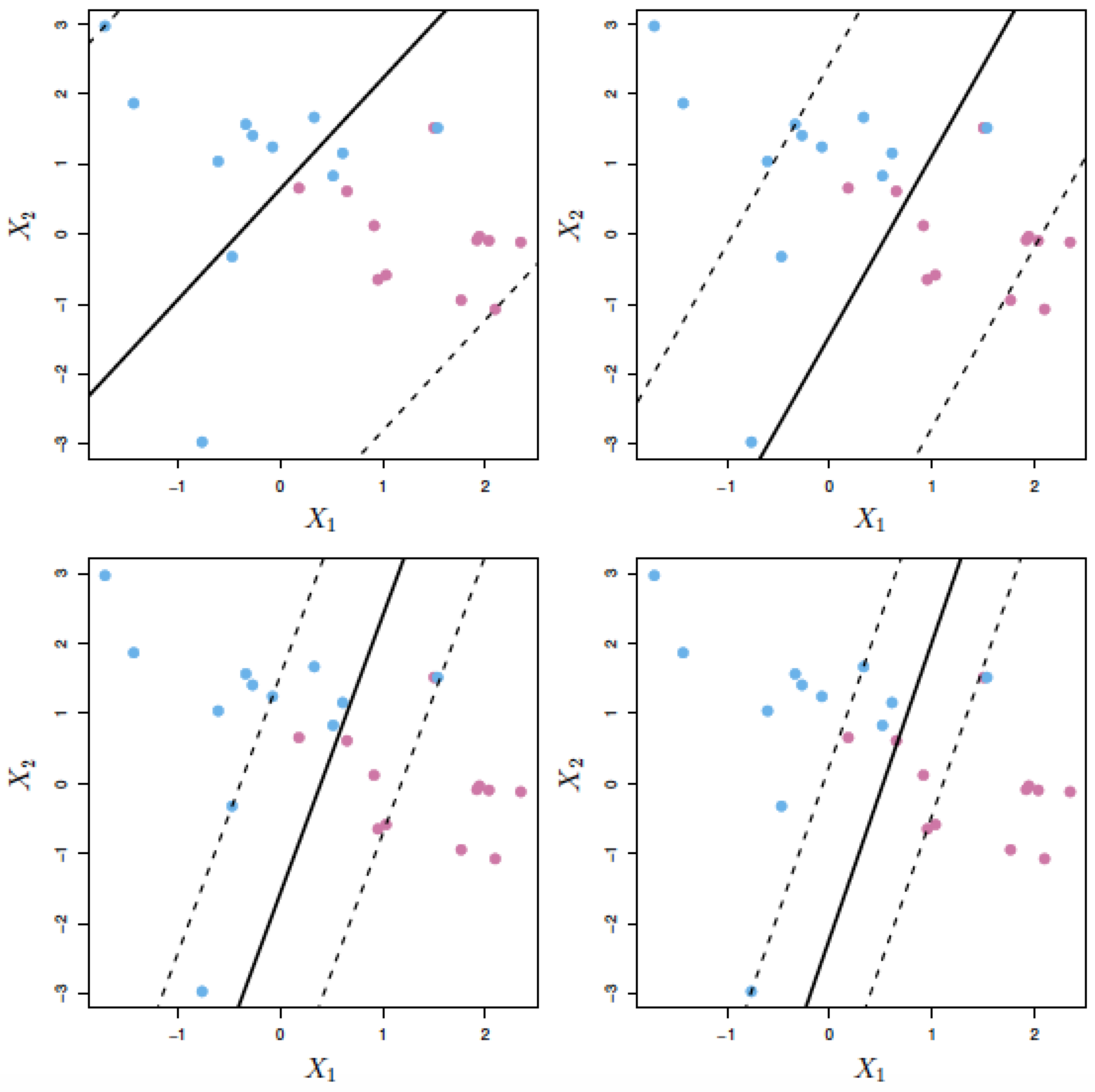
Details of Optimization Problem

For $C > 0$ no more than C observations can be on the wrong side of the hyperplane, because if an observation is on the wrong side of the hyperplane then $\varepsilon_i > 1$, and the optimization problem requires that the sum of ε_i's be less than C.

Details of Optimization Problem

- As the budget *C* increases, we become more tolerant of violations to the margin, and so the margin will widen.
- Conversely, as *C* decreases, we become less tolerant of violations to the margin and so the margin narrows.
    - An example is shown in the next slide.

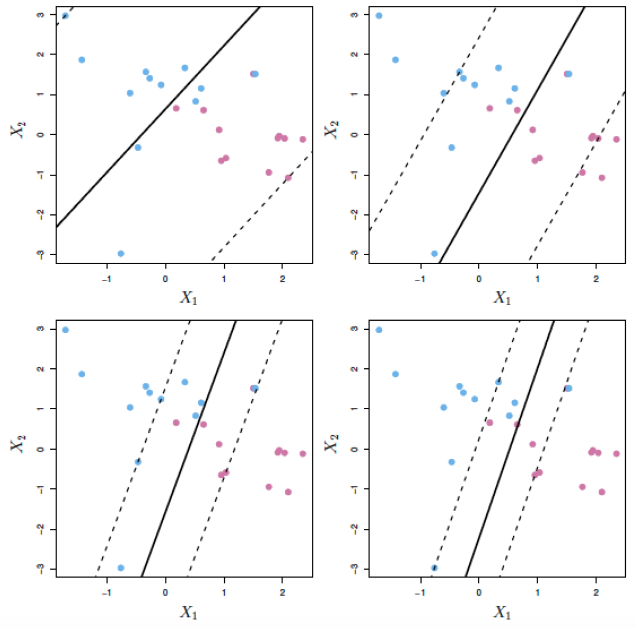
C is a regularization parameter



C is a regularization parameter

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

- The largest value of *C* was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels.



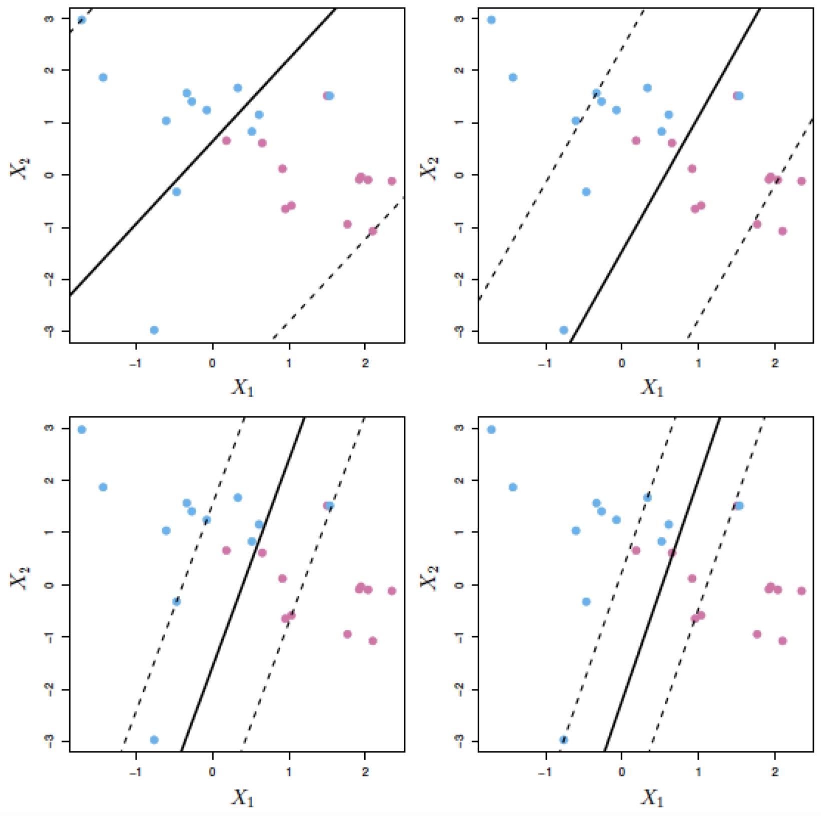
C is a regularization parameter

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

- When *C* is large, there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large.
- As *C* decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.

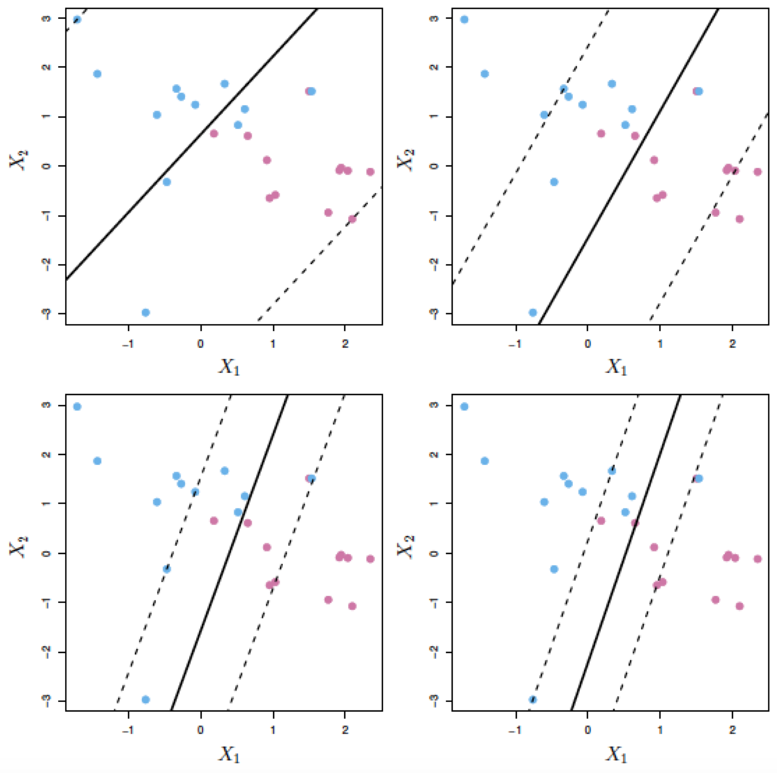
C is a regularization parameter

- *C* is treated as a tuning parameter generally chosen via cross-validation.
- *C* controls the bias-variance trade-off of the statistical learning technique.



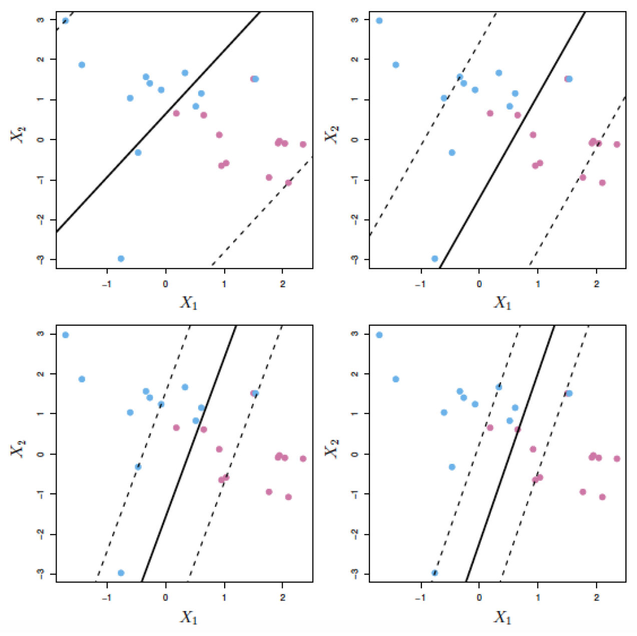
C is a regularization parameter

- 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.



C is a regularization parameter

- On the other hand, 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.



Support Vectors

- Only observations that either lie on the margin or that violate the margin will affect the hyperplane, and hence the classifier obtained.
- In other words, an observation that lies strictly on the correct side of the margin does not affect the support vector classifier!

Support Vectors

- Changing the position of that observation would not change the classifier at all, provided that its position remains on the correct side of the margin.

Support Vectors

- Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors . These observations do affect the support vector classifier.

Support Vectors

The fact that the support vector classifier's decision rule is based only on a potentially small subset of the training observations (the support vectors) means that it is quite robust to the behavior of observations that are far away from the hyperplane.

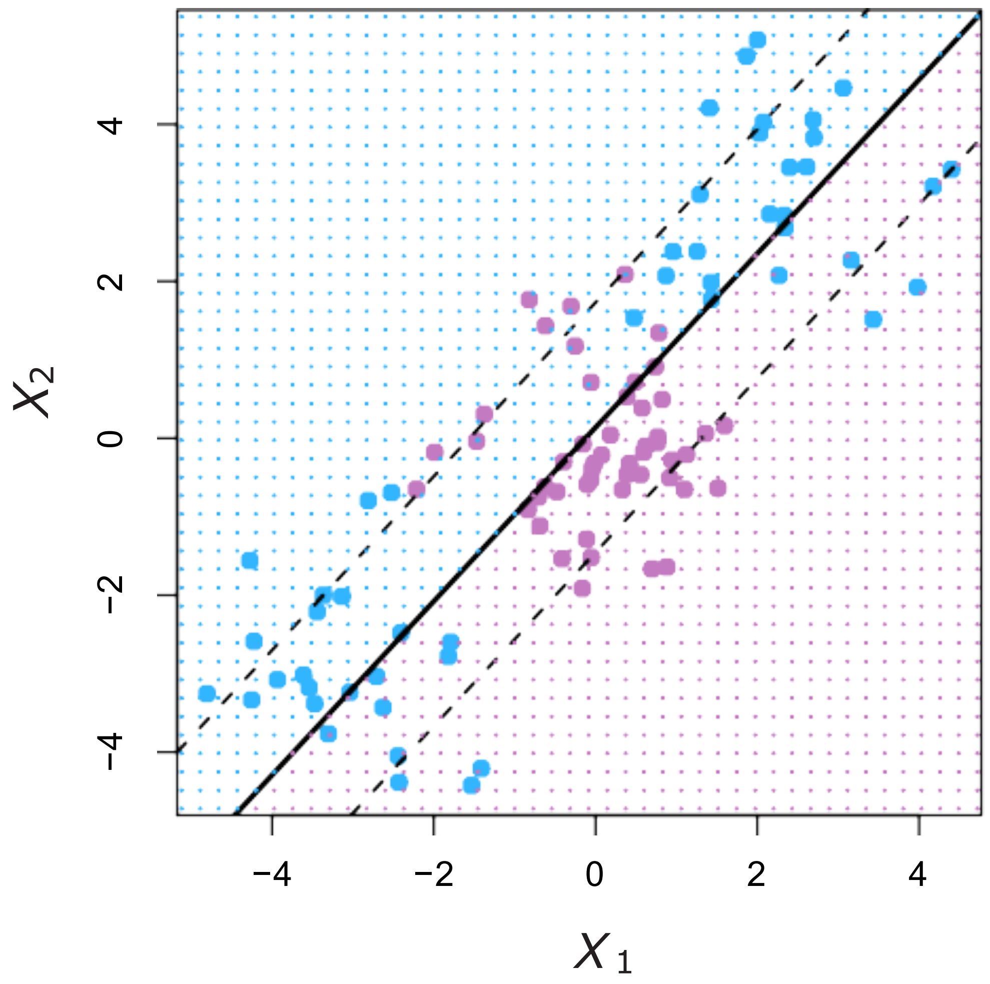
Comparison with LDA

- This property is distinct from some of the other classification methods that we have seen, such as linear discriminant analysis.
- The LDA classification rule depends on the mean of all of the observations within each class, as well as the within-class covariance matrix computed using all of the observations.

Comparison with LR

In contrast, logistic regression, unlike LDA, has very low sensitivity to observations far from the decision boundary. In fact we will see that the support vector classifier and logistic regression are closely related.

Linear boundary can fail



Sometimes 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, $X_1X_2^2$,. . .. 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

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

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

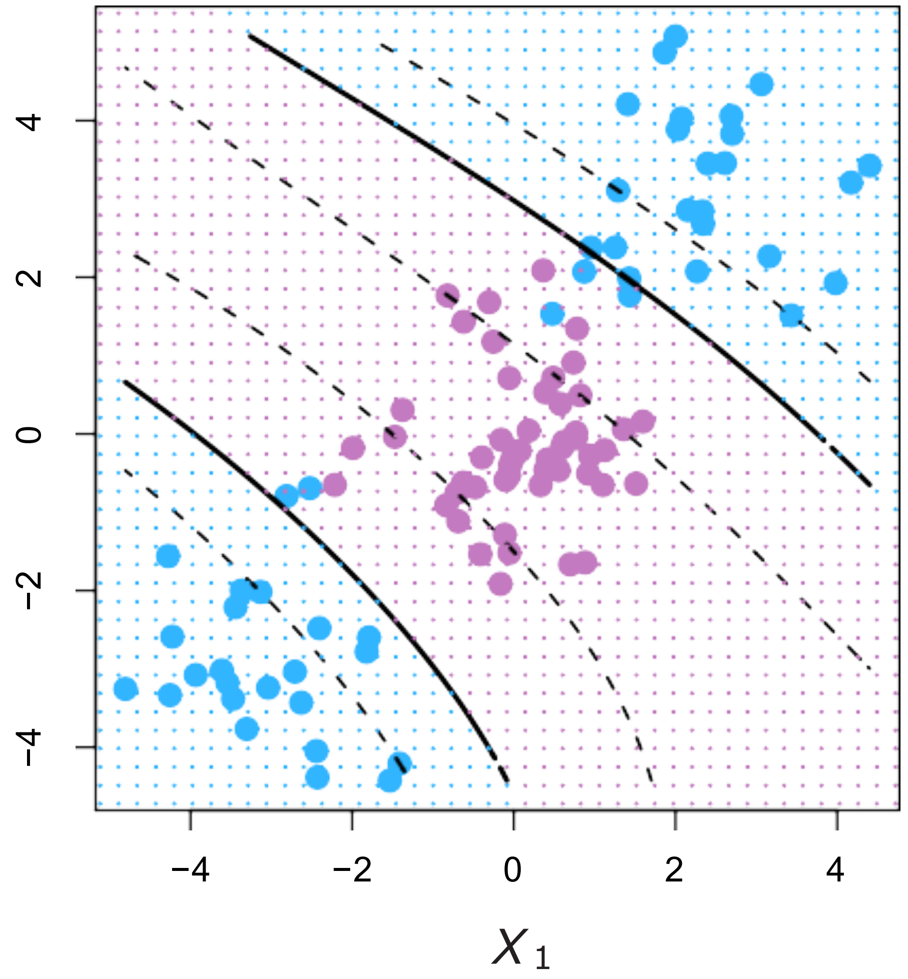
This leads to nonlinear decision boundaries in the original space (quadratic conic sections).

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



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

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

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

with n parameters

Inner products and support vectors

- The linear support vector classifier can be represented as

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

with n parameters

in order to evaluate the function *f*(*x*), we need to compute the inner product between the new point *x* and each of the training points x_i .

Inner products and support vectors

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.

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.

Inner products and support vectors

In other words

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

where $\mathcal{S}$ is the support set of indices *i* for support vectors.

Inner products and support vectors

To expand the feature space, one can use a transformed set of features *u* = *φ*(*x*). Note that *u* does not need to be of the same dimension as *x*.

The classifier in the new feature space can be represented as

$$f_1(x) = f[\varphi(x)] = \sum_{i \in S} \hat{\alpha}_i < \varphi(x), \varphi(x_i) >$$

Inner products and support vectors

It can be shown that a class of functions called *kernels* can be represented as inner products $< \varphi(x), \varphi(x_i) >$

The interesting point is that we do not need to explicitly know the function $\varphi(x)$ to use the kernel that correspond to it!

They are generalizations of the inner product.

Kernels and Support Vector Machines

Now suppose that every time the inner product appears in our equations, we replace it with

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

where K is a *kernel* .

A kernel is a function that quantifies the similarity of two observations.

For SVC, the Kernel is the usual inner product, which is called a *linear kernel*, because $\varphi(x) = x$.

Kernels and Support Vector Machines

For example

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

computes the inner-products needed for *d* dimensional polynomials-basis functions!

Kernels and Support Vector Machines

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

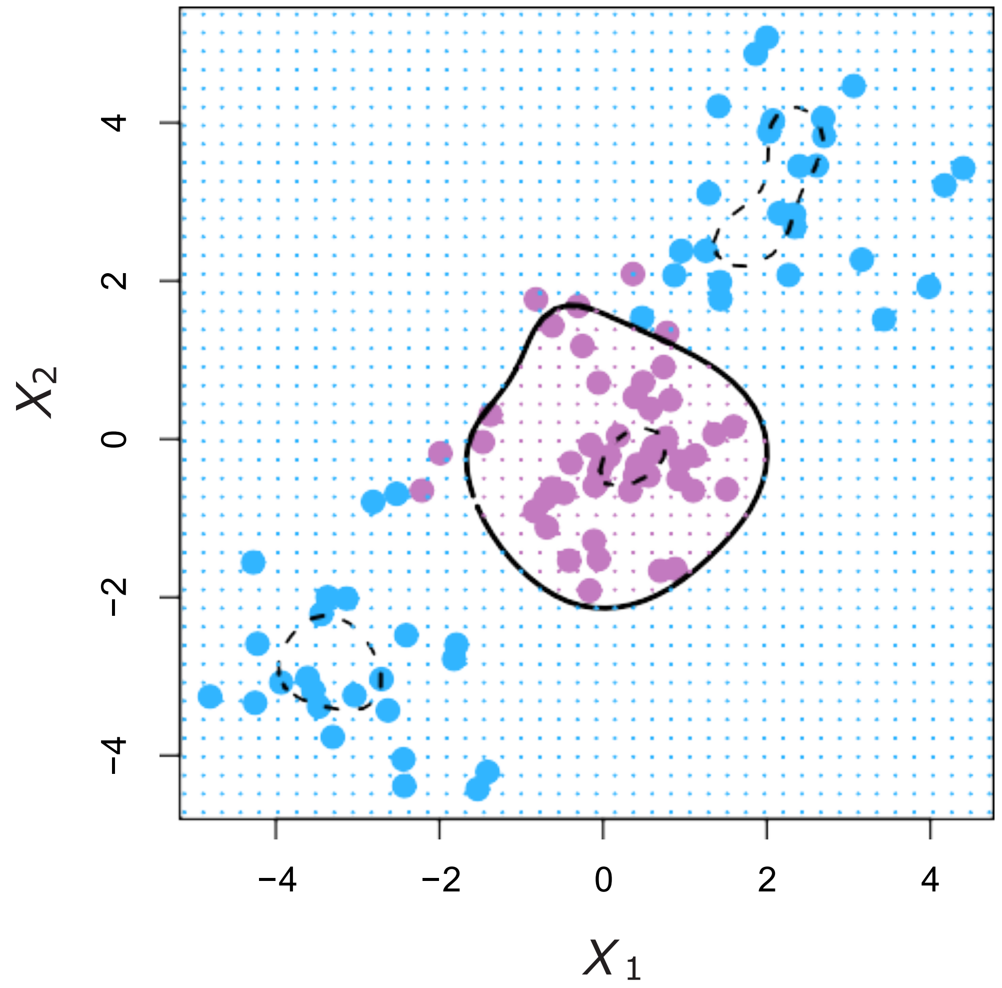
Try it for p = 2 and d = 2.

- The solution has the form

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

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.

RBF Kernel

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

- 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 the exponent will be very negative, and so $K(x^*, x_i)$ will be very tiny.

RBF Kernel

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

- Therefore, x_i will play virtually no role in $f(x^*)$.
- Recall that the predicted class label for the test observation x^* is based on the sign of $f(x^*)$.
- The radial kernel has very local behavior: only nearby training observations have an effect on the class label of a test observation. (Similar to?)

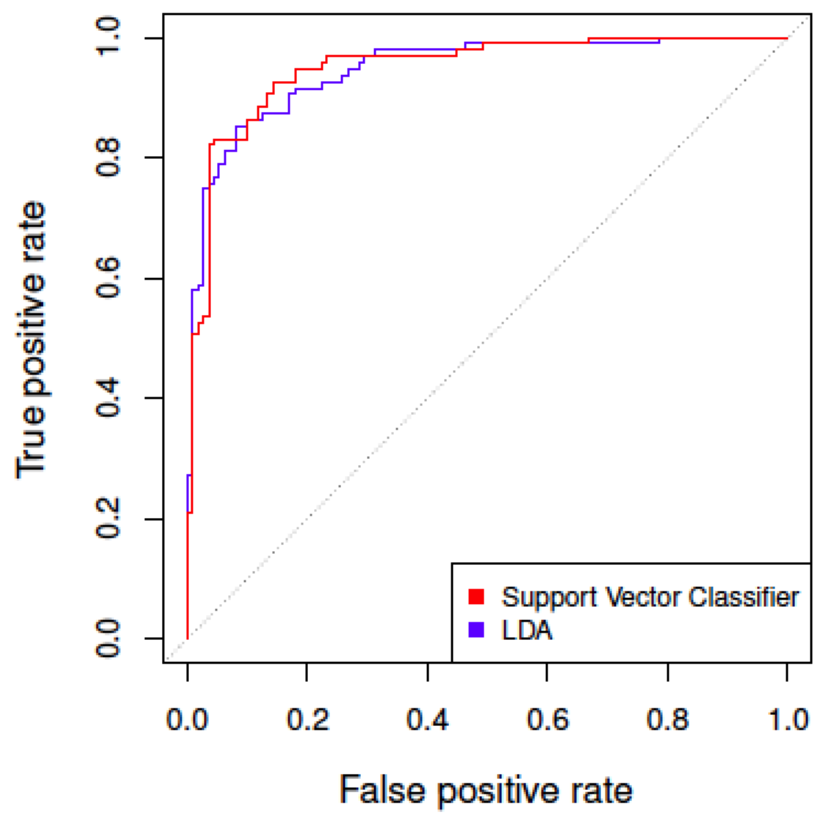
Computational Advantage

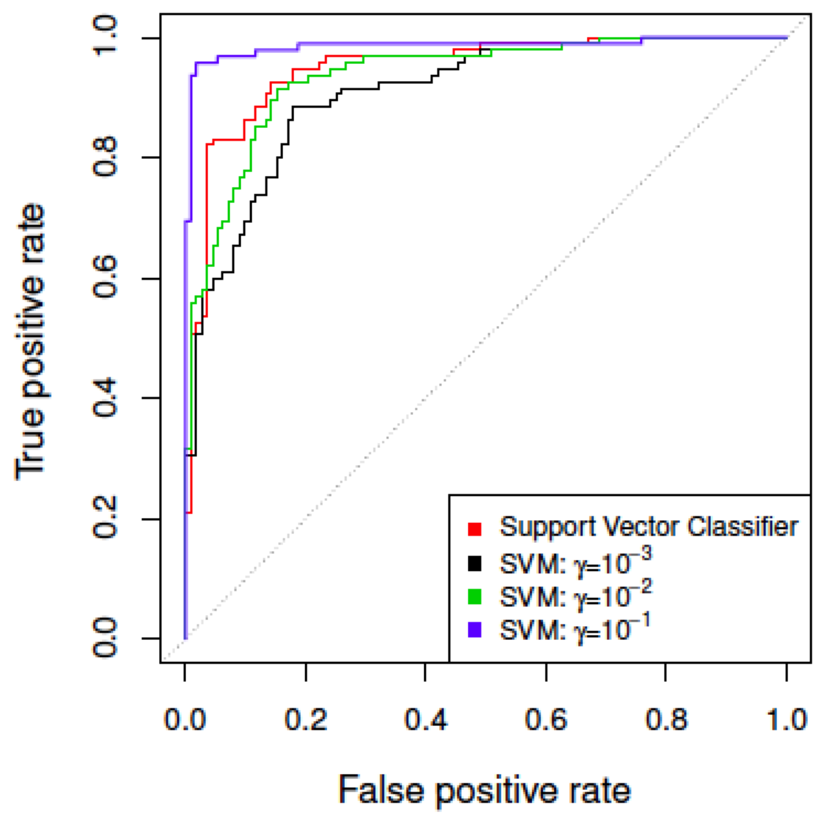
- What is the advantage of using a kernel rather than simply enlarging the feature space using functions of the original features?
- One advantage is computing without explicitly working in the enlarged feature space.

Computational Advantage

- This is important because in many applications of SVMs, the enlarged feature space is so large that computations are intractable.
- For some kernels, such as the radial, the feature space is implicit and infinite-dimensional, so we could never do the computations there anyway!

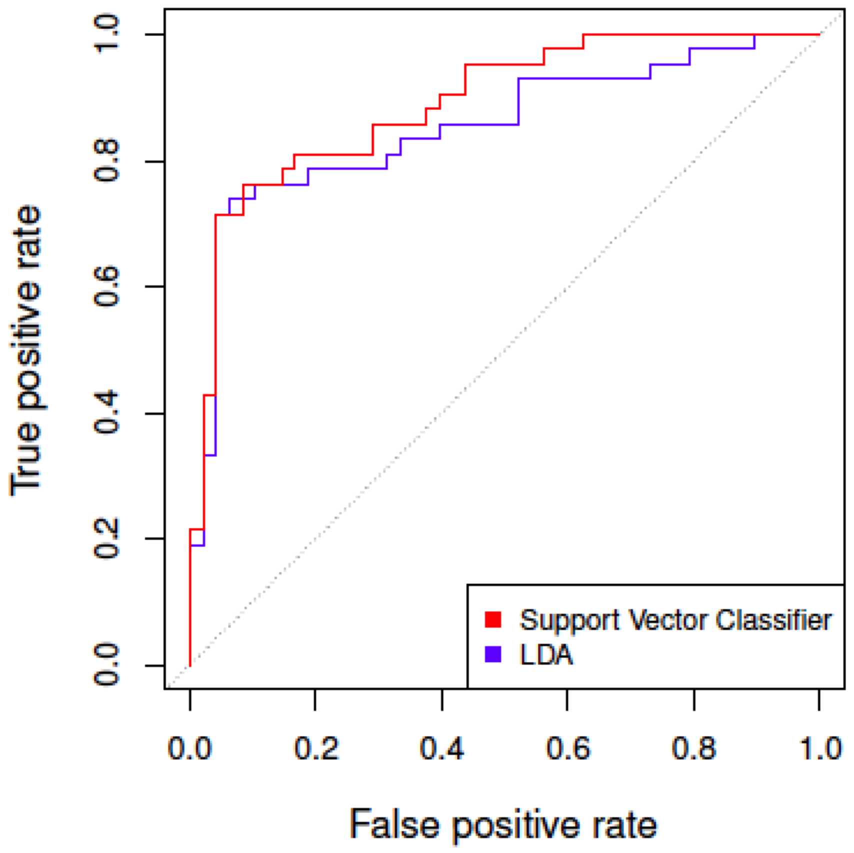
Example: Heart Data

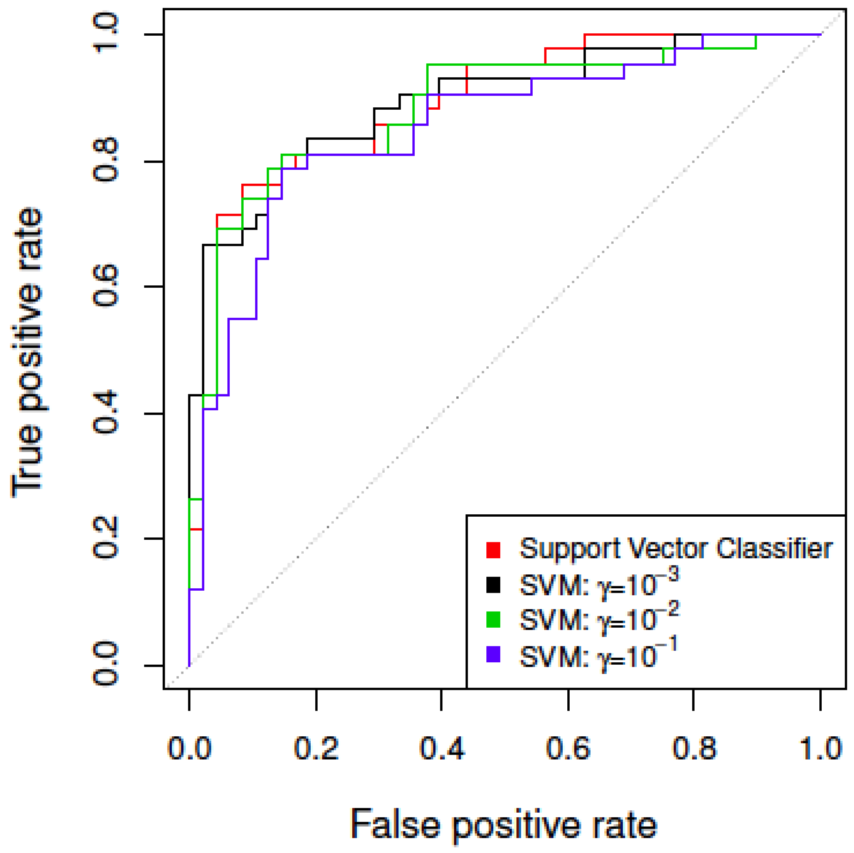




ROC curve is obtained by changing the threshold 0 to threshold t in $\hat{f}(X) > t$, and recording *false positive* and *true positive* rates as t varies. Here we see ROC curves on training data.

Example continued: Heart Test Data





Multi-Class and Multi-Label Classification Revisited

Multiclass classification means a classification task with more than two classes; e.g., classify a set of images of animals which may be horses, birds, or fish.

Multiclass classification makes the assumption that each sample is assigned to one and only one label: an animal can be either a horse or a bird but not both at the same time.

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 (The rest). 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.

SVMs: more than 2 classes?

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

- 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.

Multi-Class and Multi-Label Problems

Multilabel classification assigns to each sample a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document.

A text might be about any of religion, politics, finance or education at the same time or none of these.

Multi-Class and Multi-Label Classification Revisited

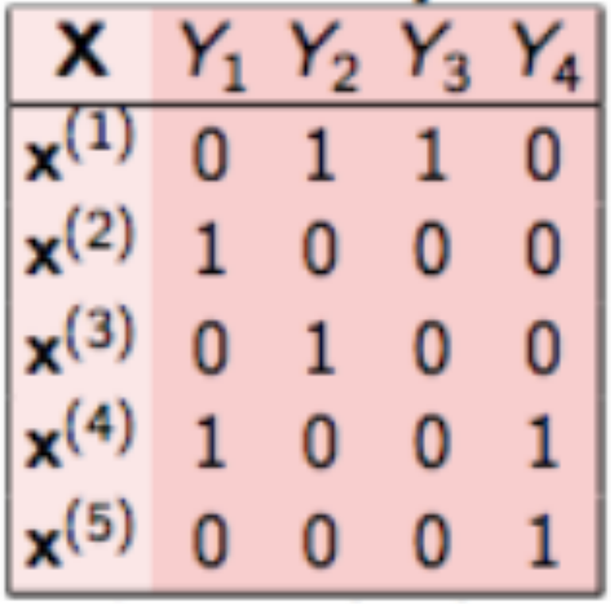
- Three methods to solve a multi-label classification problem:
  - Problem Transformation
    - Transform multi-label problem into single-label problem(s)
  - Adapted Algorithms
    - adapting conventional algorithms to directly perform multi-label classification
  - Ensemble approaches
    - Combining multiple classifiers

Multi-Class and Multi-Label Classification Revisited

- Problem Transformation
  - Binary Relevance
  - Classifier Chains
  - Label Powerset

Multi-Class and Multi-Label Classification Revisited

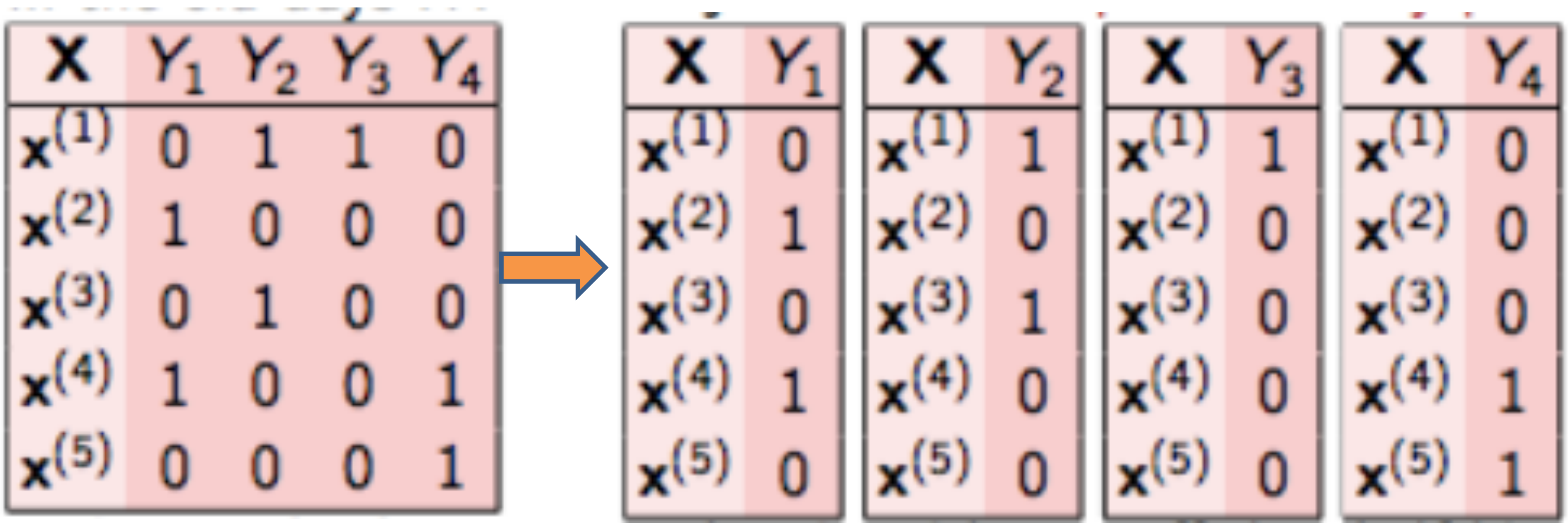
- Binary Relevance
- The simplest technique, which treats each label as a separate binary or multi-class classification problem.
- Example: consider a case as shown below. X is the independent feature and Y's are the target variable.



X	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

Multi-Class and Multi-Label Classification Revisited

- Binary Relevance
- Solution: The problem is broken into 4 different single class classification problems.



$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

$\mathbf{X}$	Y_1
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	1
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	0

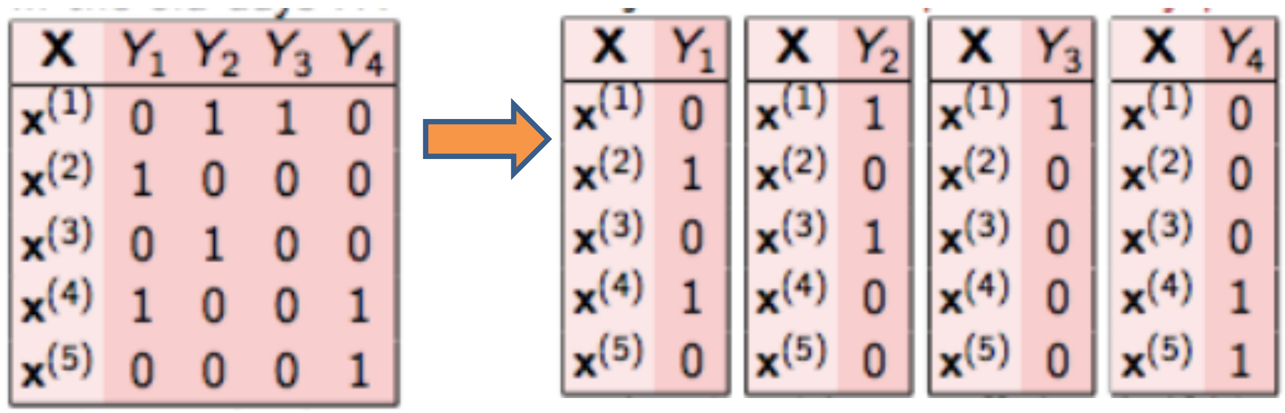
$\mathbf{X}$	Y_2
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	1
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

$\mathbf{X}$	Y_3
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

$\mathbf{X}$	Y_4
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	1

Multi-Class and Multi-Label Classification Revisited

- Binary Relevance
- Solution: For a new data point **x***, each of the labels $Y_1,\ldots, Y_4$ is predicted separately.
- Note1: Any classifier (e.g. Naïve Baye's, Logistic Regression, and SVM) can be used for predicting each label
- Note 2: Each label may give rise to a binary or multi-class classification problem.



$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

$\mathbf{X}$	Y_1
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	1
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	0

$\mathbf{X}$	Y_2
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	1
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

$\mathbf{X}$	Y_3
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

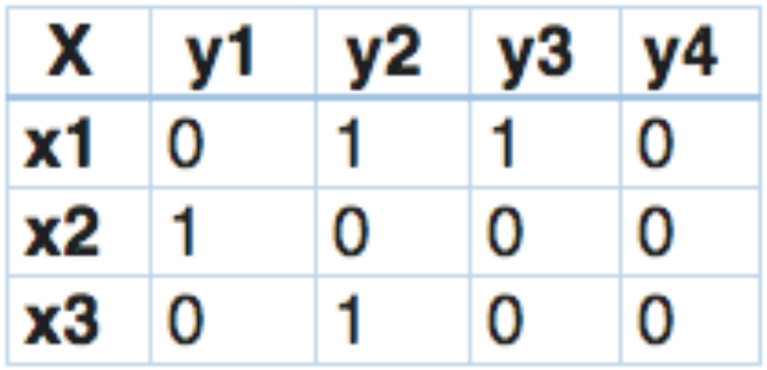
$\mathbf{X}$	Y_4
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	1

Multi-Class and Multi-Label Classification Revisited

- Classifier Chains
- In this approach, the first classifier is trained just on the input data and then each next classifier is trained on the input space and all the previous classifiers in the chain.

Multi-Class and Multi-Label Classification Revisited

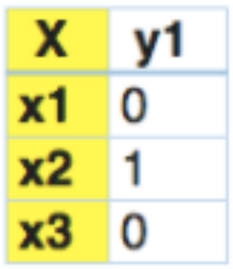
- Example: X as the input space and *Y*'s as the labels.



X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

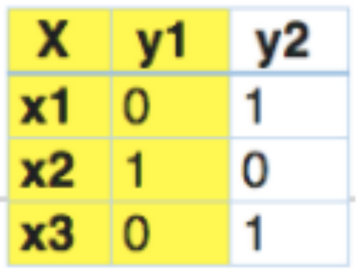
Multi-Class and Multi-Label Classification Revisited

- Solution: In classifier chains, this problem would be transformed into 4 different single label problems, just like shown below. Here yellow colored is the input space and the white part represent the target variable.



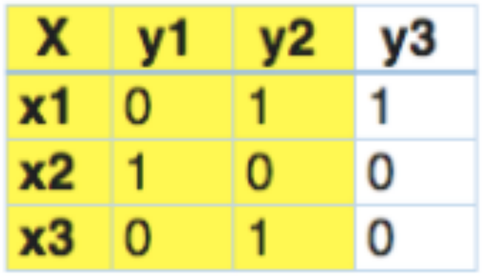
X	y1
x1	0
x2	1
x3	0

Classifier 1



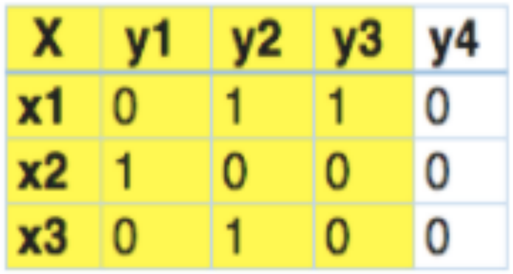
X	y1	y2
x1	0	1
x2	1	0
x3	0	1

Classifier 2



X	y1	y2	y3
x1	0	1	1
x2	1	0	0
x3	0	1	0

Classifier 3

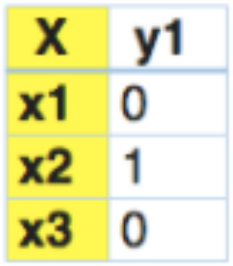


X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

Classifier 4

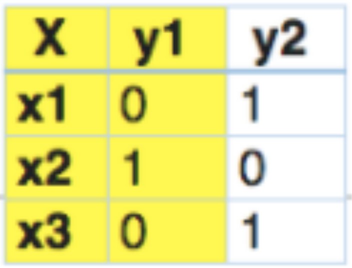
Multi-Class and Multi-Label Classification Revisited

- Note: This is quite similar to binary relevance, the only difference being it forms chains in order to preserve label correlation.



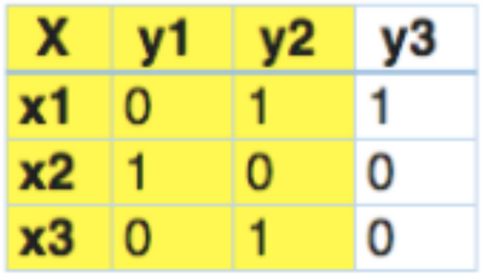
X	y1
x1	0
x2	1
x3	0

Classifier 1



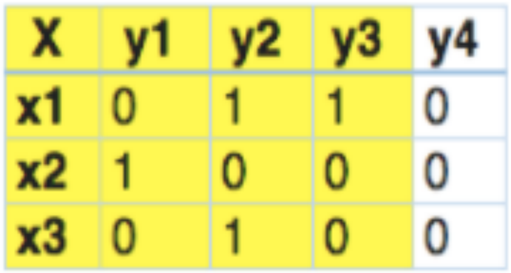
X	y1	y2
x1	0	1
x2	1	0
x3	0	1

Classifier 2



X	y1	y2	y3
x1	0	1	1
x2	1	0	0
x3	0	1	0

Classifier 3



X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

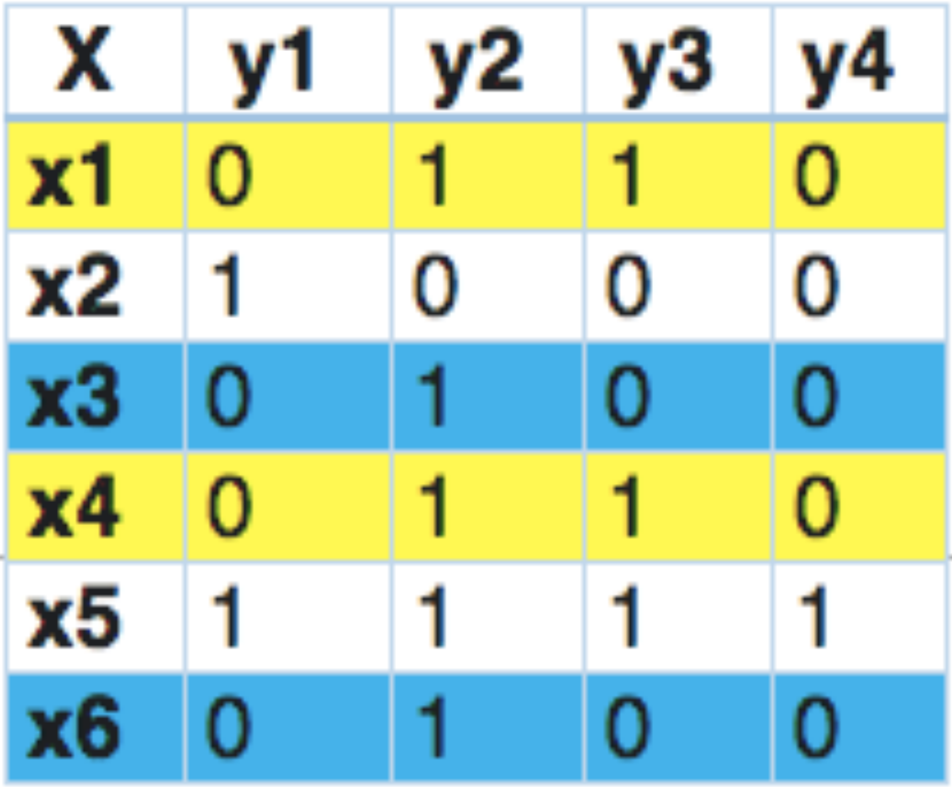
Classifier 4

Multi-Class and Multi-Label Classification Revisited

- Label Powerset
- This method transforms the problem into a multi-class problem with one multi-class classifier is trained on all unique label combinations found in the training data.

Multi-Class and Multi-Label Classification Revisited

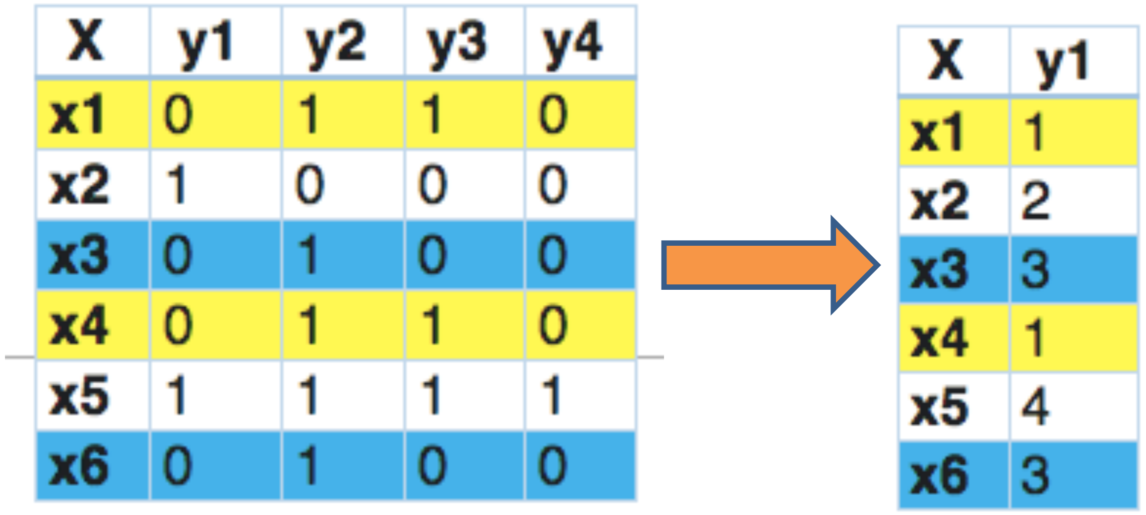
- Example: X are features, $Y_1,\ldots Y_4$ are labels



X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0
x4	0	1	1	0
x5	1	1	1	1
x6	0	1	0	0

Multi-Class and Multi-Label Classification Revisited

- Solution: $\mathbf{x}_1$ and $\mathbf{x}_4$ have the same labels, similarly, $\mathbf{x}_3$ and $\mathbf{x}_6$ have the same set of labels. So, label powerset transforms this problem into a single multi-class problem as shown below.

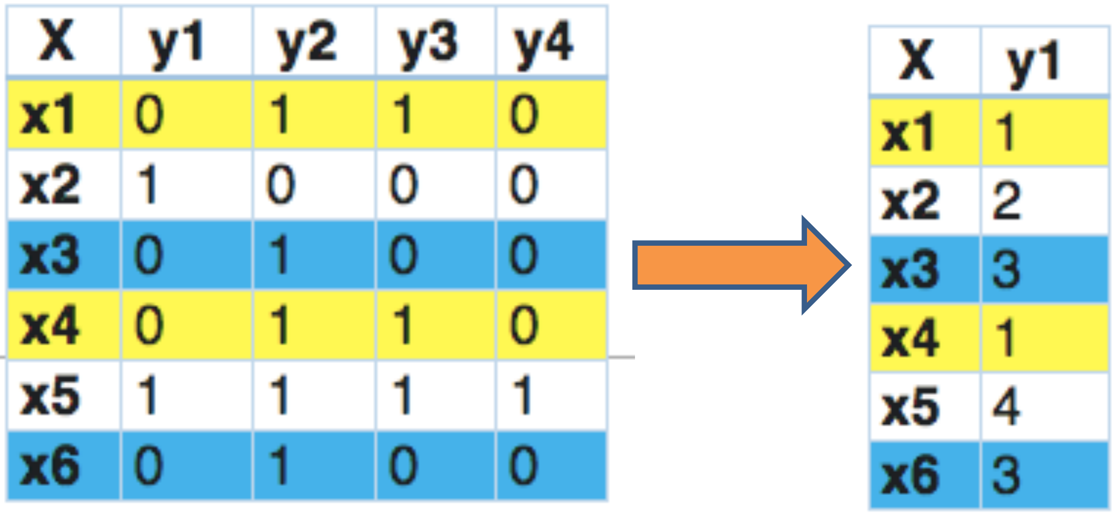


X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0
x4	0	1	1	0
x5	1	1	1	1
x6	0	1	0	0

X	y1
x1	1
x2	2
x3	3
x4	1
x5	4
x6	3

Multi-Class and Multi-Label Classification Revisited

- Solution: The label powerset method has given a unique class to every possible label combination that is present in the training set.



X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0
x4	0	1	1	0
x5	1	1	1	1
x6	0	1	0	0

X	y1
x1	1
x2	2
x3	3
x4	1
x5	4
x6	3

Multi-Class and Multi-Label Classification Revisited

- Adapted Algorithms
  - The most famous adapted algorithm is Multilabel kNN (MLkNN)

Multi-Class and Multi-Label Classification Revisited

- ML-kNN uses the kNN algorithm independently for each label ℓ.
- It finds the k nearest examples to the test instance and considers those that are labeled at least with ℓ as positive and the rest as negative.

Multi-Class and Multi-Label Classification Revisited

- What mainly differentiates this method from other binary relevance (BR) methods is the use of prior probabilities. ML-kNN can also rank labels.

Multi-Class and Multi-Label Classification Revisited

- Adapted Algorithms
  - Decision Trees can be modified to perform multi label classification.
  - The main modification is in calculating purities for each region.

Multi-Class and Multi-Label Classification Revisited

- Ensemble Algorithms
  - AdaBoost.MH and AdaBoost.MR

Evaluation Metrics

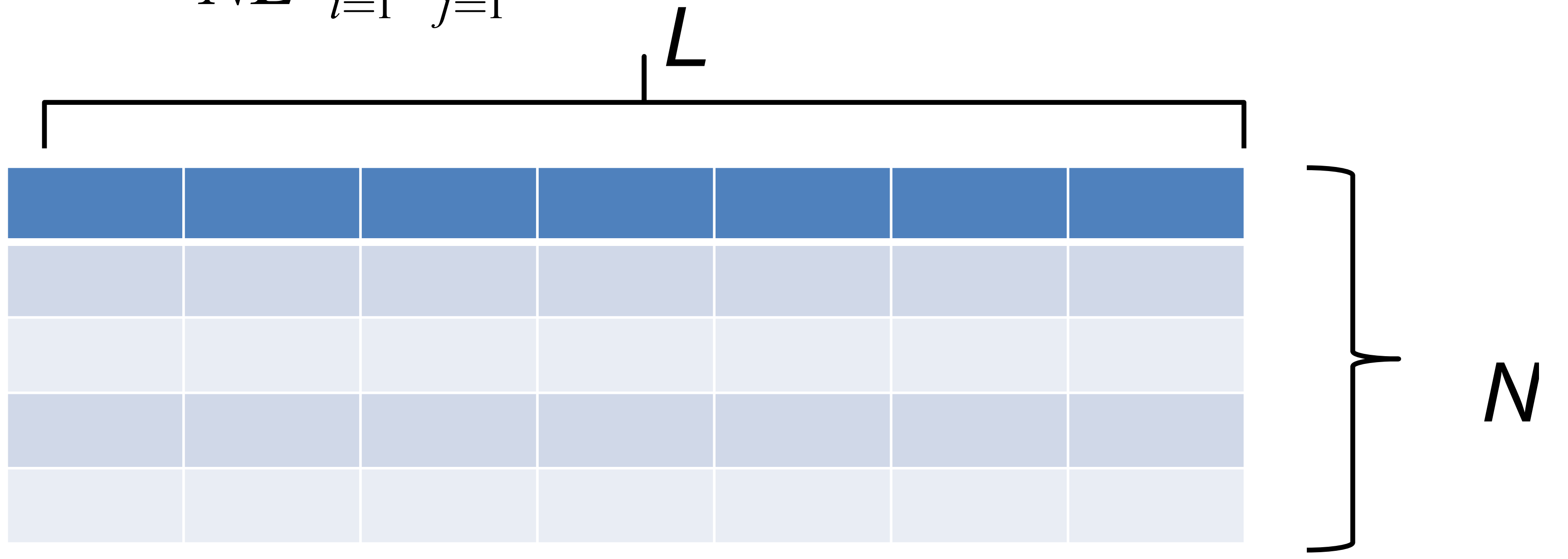
- One cannot simply use our normal metrics to calculate the accuracy of the predictions.
- **Accuracy score/ Exact match** metric: This function calculates subset accuracy meaning the predicted set of labels should exactly match with the true set of labels.

Evaluation Metrics

- **Hamming Loss**: The fraction of the wrong labels to the total number of labels, which for binary labels becomes:

$$\frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L}\mathrm{xor}(\hat{y}_{ij}, y_{ij})$$

-



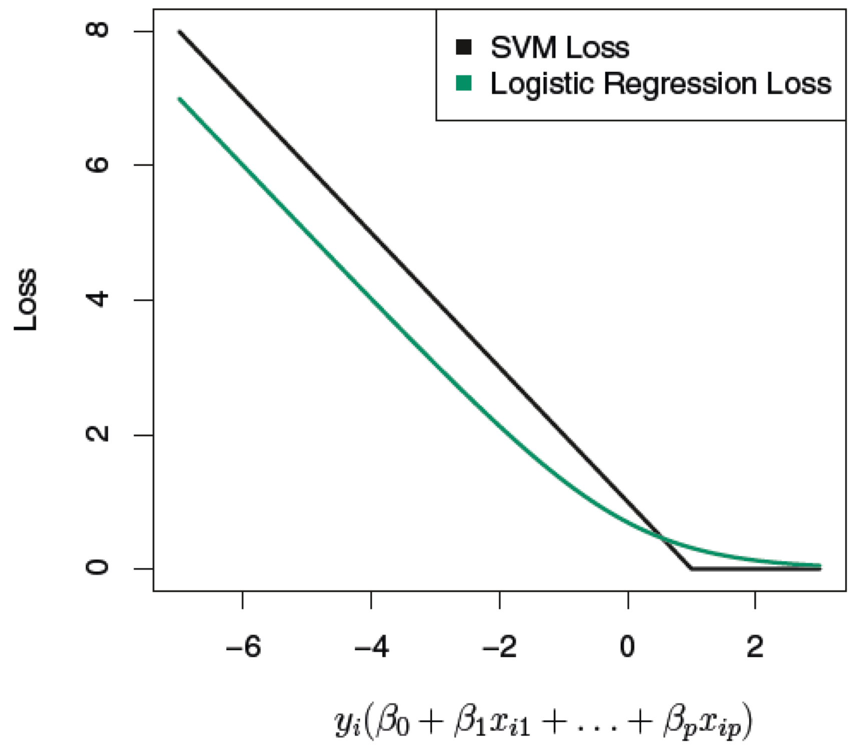
Multi-Class and Multi-Label Classification Revisited

- For an interesting overview, see:
- https://arxiv.org/pdf/1703.08991.pdf

SVC versus Logistic Regression?

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

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

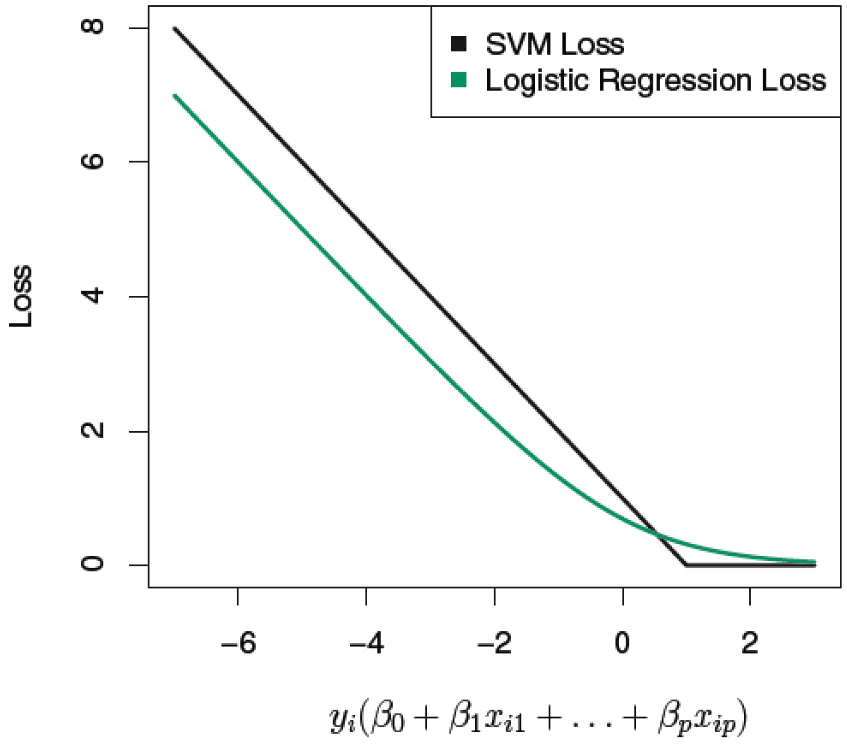


This has the form *loss plus penalty*.
The loss is known as the *hinge loss*.
Very similar to "loss" in logistic regression (negative log-likelihood).

SVC versus Logistic Regression?

Similarly, one can rewrite logistic regression when Y is -1 or 1 (instead of 0,1) as

$$P(Y = y_i | X = x_i) = \frac{1}{1 + \exp(-y_i f(x_i))}$$



The negative log likelihood loss function is:

$$-\log[P(Y = y_i | X = x_i)]$$
$$= \log[(1 + \exp(-y_i f(x_i))]$$

Loss Plus Penalty: L1 Regularization

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

The above optimization problem has the form of *loss plus penalty*.

Note that the original penalty in SVC is a ridge penalty.

One can change the penalty with L1 penalty and perform variable selection along with Support Vector Classification.

Loss Plus Penalty: L1 Regularization

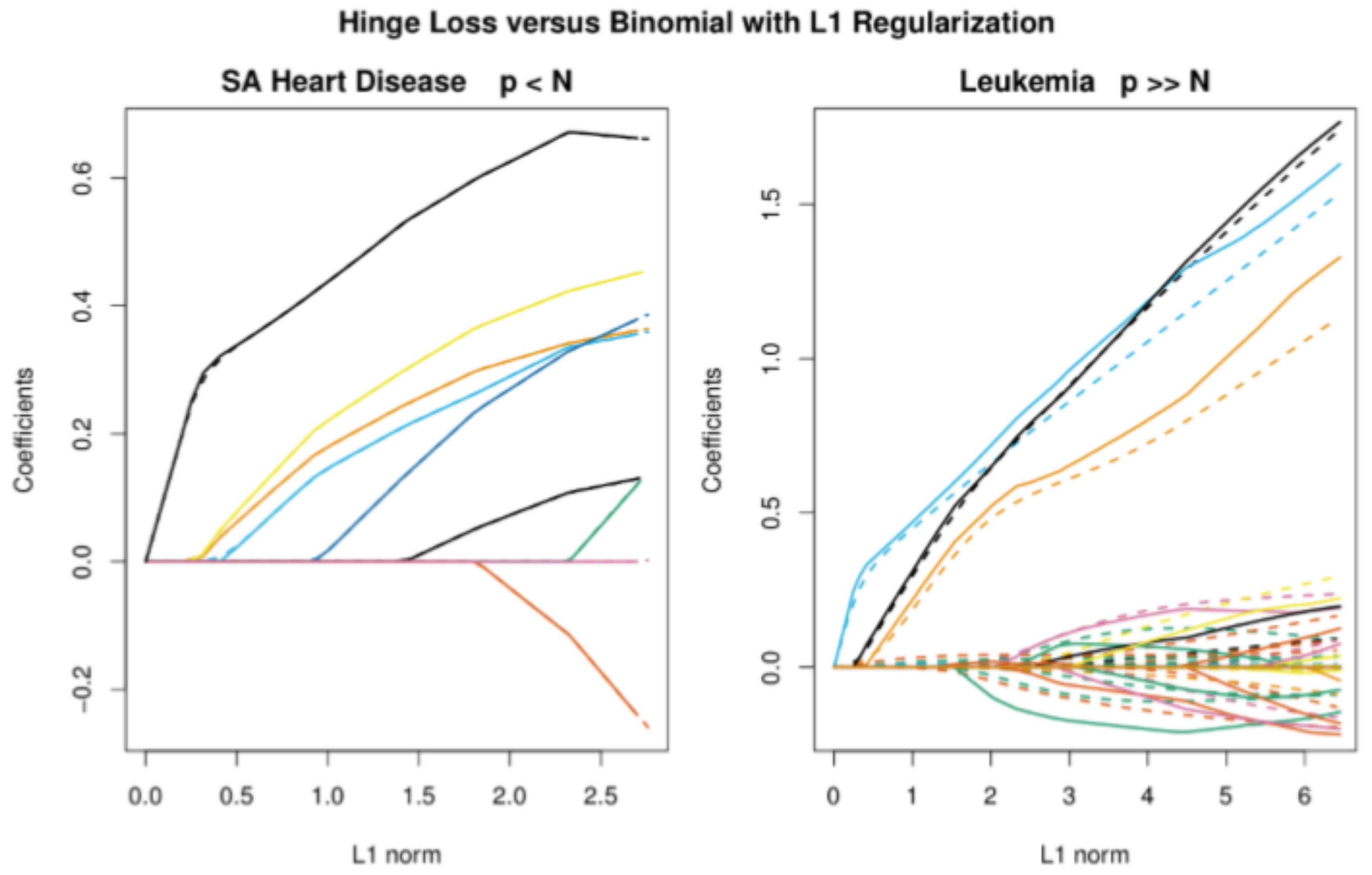
$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

Because the hinge loss function is not differentiable, the solution paths for different values of λ may have jumps.

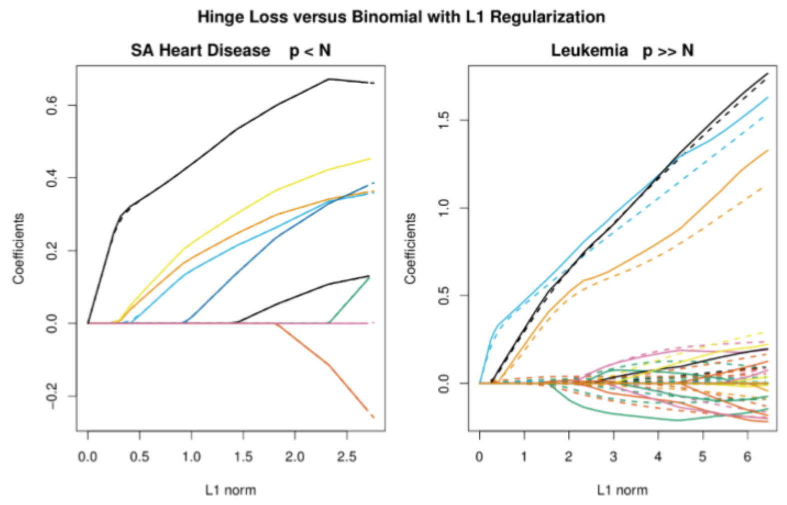
Therefore, some authors replace the hinge loss with a differentiable squared hinge loss.

Different combinations of hinge loss and squared hinge loss with L1 and L2 penalty result in various versions of SVM.

Comparison of L1 Regularized SVM and Logistic Regression



Details of Previous Figure



A comparison of the coefficient paths for the L1- SVM versus L1 logistic regression on two examples.
In the left we have the South African heart disease data (N = 462 and p = 9), and on the right the Leukemia data (N = 38 and p = 6087). The dashed lines are the SVM coefficients, the solid lines logistic regression. The similarity is striking in the left example, and strong in the right.

Which to use: SVM or Logistic Regression

- When classes are (nearly) separable, SVM does better than LR. So does LDA.

Which to use: SVM or Logistic Regression

- SVMs are popular in high-dimensional classification problems with p<<n, since the computations are $O(pn^2)$ for both linear and nonlinear kernels. (Some references say between $O(n^2)$ and $O(n^3)$).
- Additional efficiencies can be realized for linear SVMs (Shalev-Shwartz, Singer and Srebro 2007).

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.

Which one to use: SVM or Logistic Regression

- If you wish to estimate probabilities, LR is the choice.
- However, one can estimate probabilities from distances of data points from the SVC hyperplane:

Which one to use: SVM or Logistic Regression

- For nonlinear boundaries, kernel SVMs are popular, as we will see. Can use kernels with LR and Bayesian LDA as well, but computations are more expensive.
- Also, Kernels try to find a feature space in which decision boundaries are linear. That's not commensurate with characteristics of LR.

The Vapnik-Chervonenkis Dimension

- The **VC dimension** is a measure of the **capacity** (complexity, expressive power, richness, or flexibility) of a space of functions that can be learned by a classification algorithm.

The Vapnik-Chervonenkis Dimension

- A classification model *f* with some parameter vector **β** is said to *shatter* a set of data points ($\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n$) if for all assignments of labels to those points, there exists a **β** such that the model *f* makes no errors when evaluating that set of data points.

The Vapnik-Chervonenkis Dimension

- The **VC dimension** of a model *f* is the maximum number of points that can be arranged so that *f* shatters them.

The Vapnik-Chervonenkis Dimension

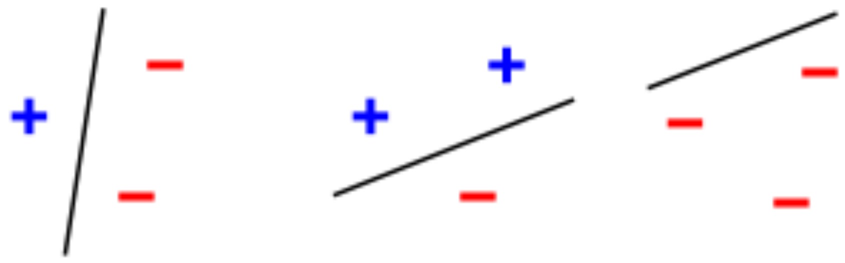
- Example: *f* is a straight line as a classification model on points in a two-dimensional plane (this is the model used by a perceptron).
- The line should separate positive data points from negative data points.

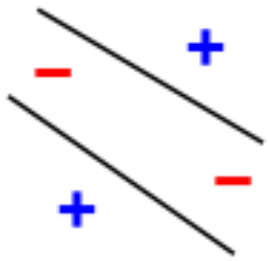
The Vapnik-Chervonenkis Dimension

- There exist sets of 3 points that can indeed be shattered using this model (any 3 points that are not collinear can be shattered). However, no set of 4 points can be shattered. Thus, the VC dimension of this particular classifier is 3.

The Vapnik-Chervonenkis Dimension

- Note, only 3 of the $2^3 = 8$ possible label assignments are shown for the three points.

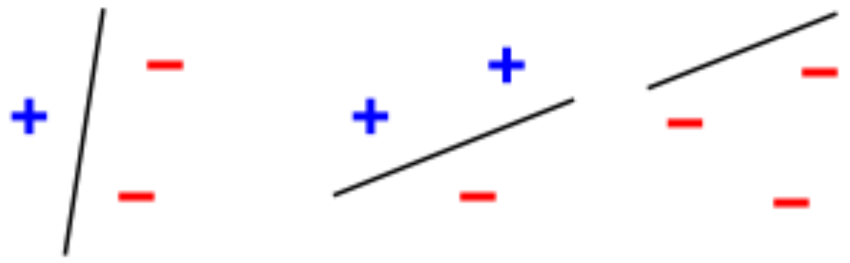


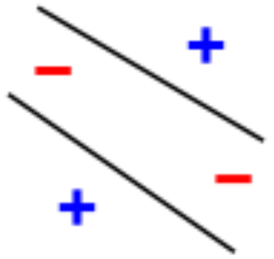


3 points shattered **4 points impossible**

The Vapnik-Chervonenkis Dimension

- Note, only 3 of the $2^3 = 8$ possible label assignments are shown for the three points.





3 points shattered **4 points impossible**

The Vapnik-Chervonenkis Dimension

- The VC Dimension of affine classifiers of the form f($\mathbf{x}$)= $\boldsymbol{\beta}^T\mathbf{x}+ \beta_0$, $\boldsymbol{\beta} \in \mathbb{R}^p$ is p+1.
- This includes SVC, a linear SVM.
- The VC Dimension of an SVM equipped with an RBF kernel is infinite.

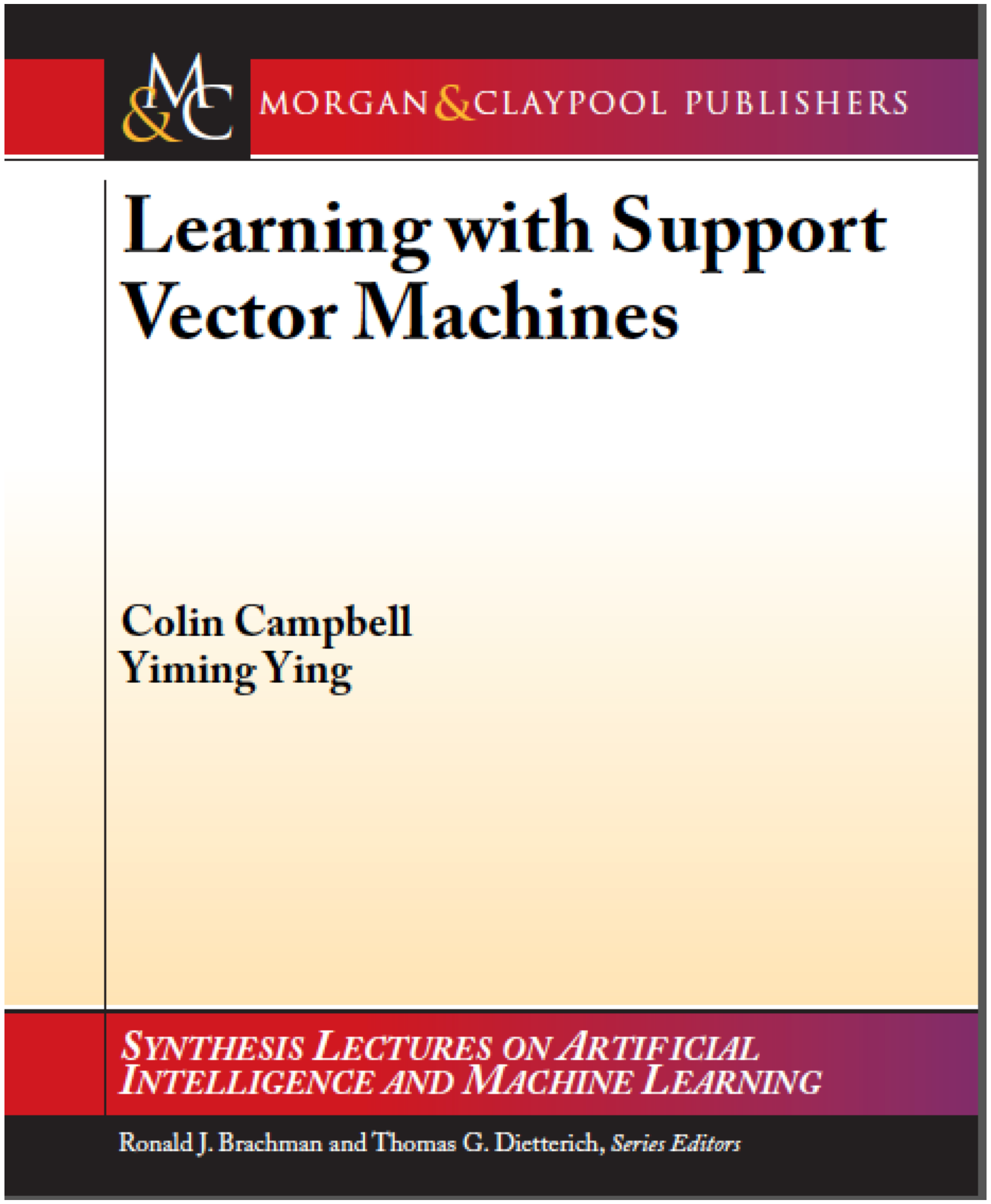
Support Vector Regression

- There is an extension of the SVM for regression, called support vector regression (SVR) .
- We saw that least squares regression seeks coefficients β_0, $\beta_1, \ldots, \beta_p$ such that the sum of squared residuals is as small as possible.

Support Vector Regression

- Support vector regression instead seeks coefficients that minimize a different type of loss, where only residuals larger in absolute value than some positive constant contribute to the loss function.
- This is an extension of the margin used in support vector classifiers to the regression setting.

SVM Learning



MORGAN & CLAYPOOL PUBLISHERS

Learning with Support Vector Machines

Colin Campbell
Yiming Ying

SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Ronald J. Brachman and Thomas G. Dietterich, *Series Editors*