CAP 5625: Computational Foundations for Artificial Intelligence

Support vector machines

Consider a decision function based on sign

Consider a situation with 2 classes.

The decision boundary is a hyperplane, which has the form

$$f(X) = \beta_0 + X^T\beta = \beta_0 + \sum_{j=1}^{p} X_j\beta_j = 0$$

with

$$f(X) > 0$$

belonging to one class and with

$$f(X) < 0$$

belonging to the other class.

Consider a decision function based on sign

Here, we only care whether $f(X) = \beta_0 + X^T\beta$ is positive or negative (*i.e.*, what side of the hyperplane an observation X is on).

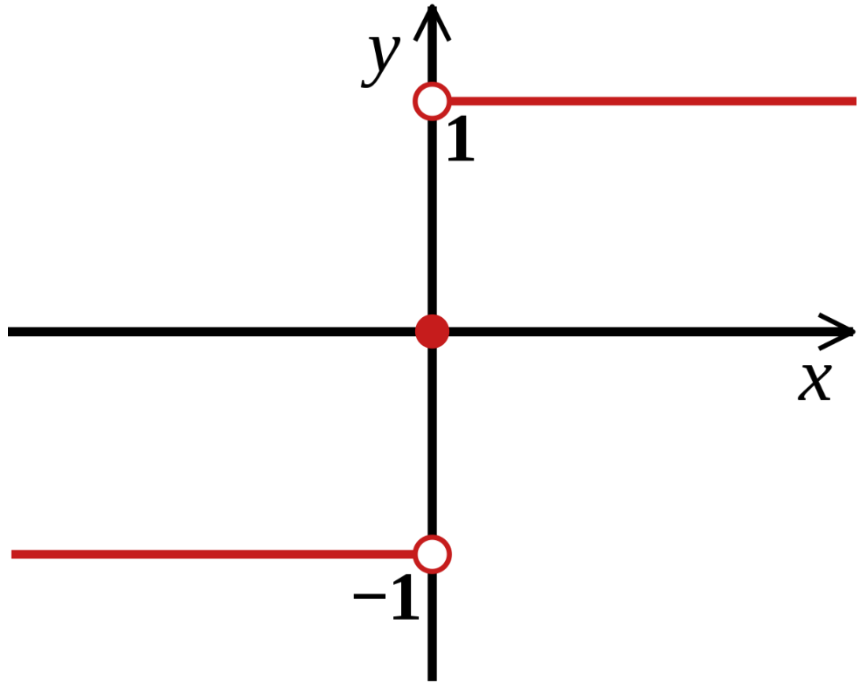
We can therefore model this hyperplane using the decision function

$$Y(X) = \text{sign}(f(X)) = \begin{cases} -1 & \text{if } f(X) < 0 \\ 0 & \text{if } f(X) = 0 \\ 1 & \text{if } f(X) > 0 \end{cases}$$

and where the response $y \in \{-1,1\}$ for training observations takes values of -1 or 1.

The hyperplane with sign function

The decision function $Y(X) = \text{sign}(f(X))$



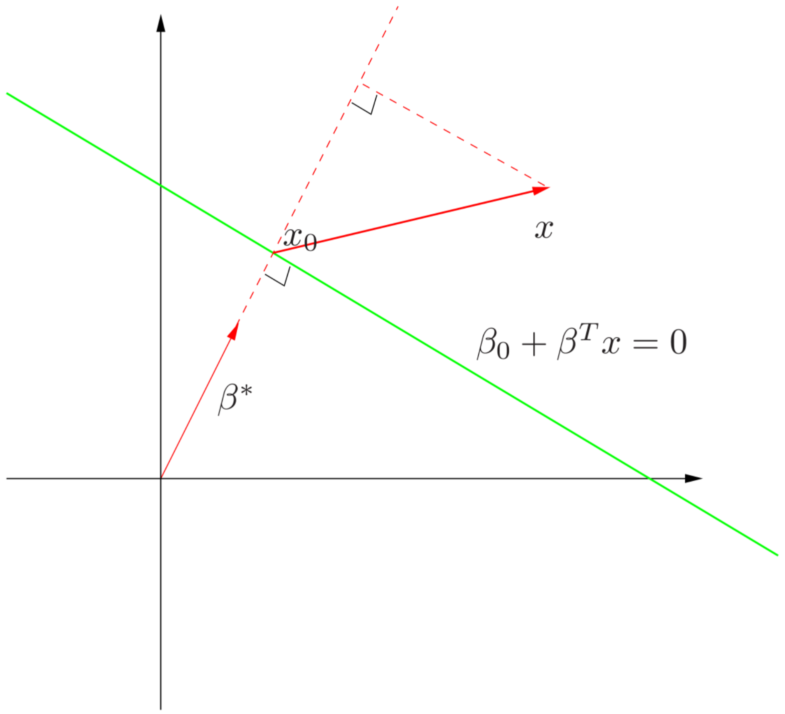
We will try to directly learn this hyperplane, by introducing the **perceptron learning algorithm**, which attempts to construct a classifier (known as a **perceptron**) with a decision boundary (hyperplane) of the form $L = \{x: \beta_0 + x^T\beta = 0\}$ that is a linear combination of input features, and that returns the sign.

https://en.wikipedia.org/wiki/Sign_function.

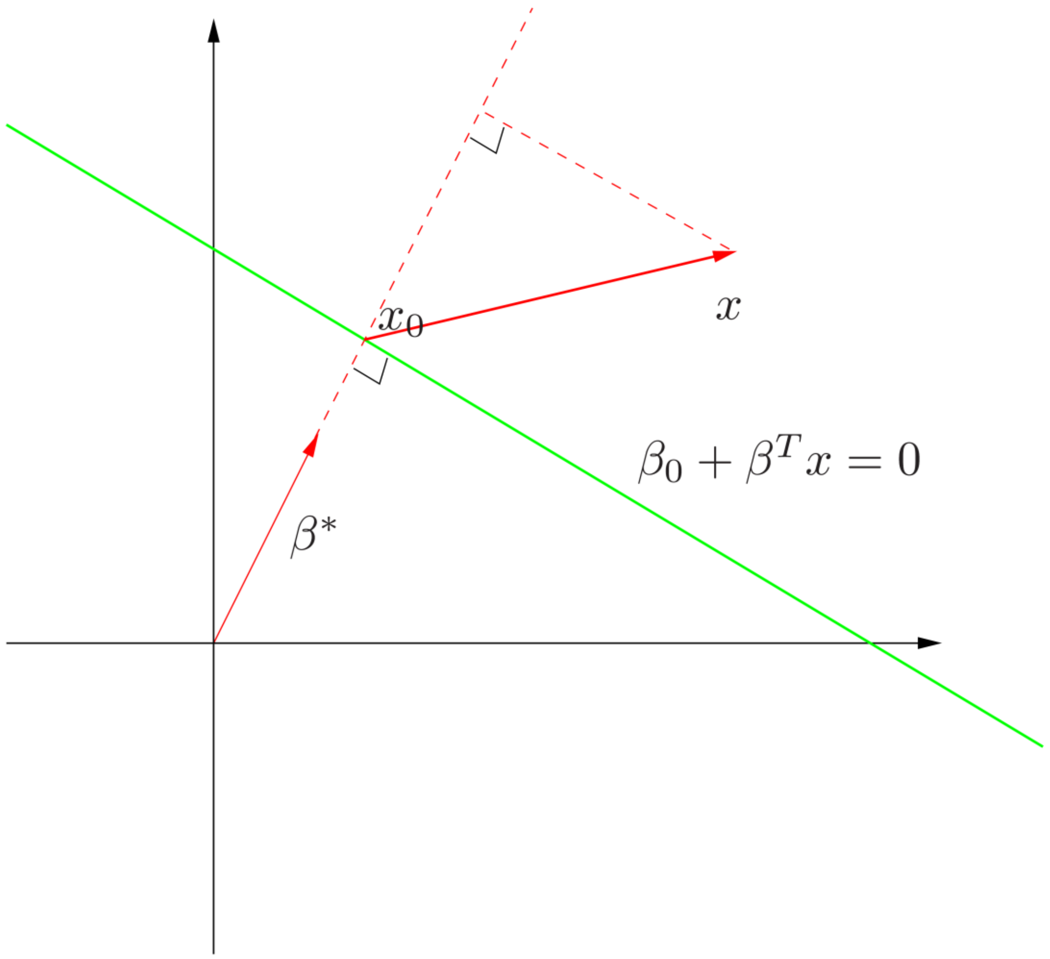
Some properties of hyperplanes

Suppose we have a hyperplane defined as $L = \{x: f(x) = 0\}$, where $f(x) = \beta_0 + x^T\beta$ for $\beta \in \mathbb{R}^p$.

Any two points x_1 and x_2 lying on the hyperplane have the property that $(x_1 - x_2)^T\beta = 0$, and thus $\beta^\star = \beta/||\beta||_2$ is the vector normal to the hyperplane, as it is orthogonal to vectors on the hyperplane.

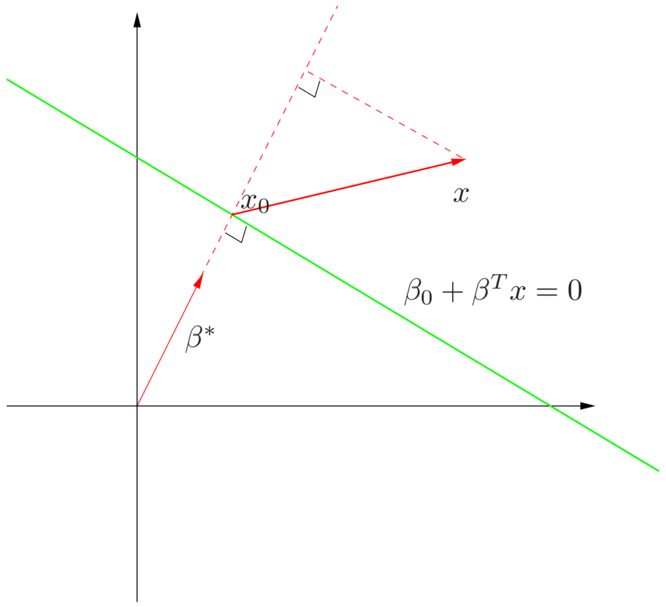


Some properties of hyperplanes



For any point x_0 in the hyperplane, $x_0^T\beta = -\beta_0$.

Some properties of hyperplanes



The signed distance of any point x to hyperplane $f(x) = 0$ is

$$
\begin{aligned}
(x - x_0)^T \beta^\star = (x^T - x_0^T)\beta^\star &= x^T \beta^\star - x_0^T \beta^\star \\
&= \frac{1}{\|\beta\|_2} x^T \beta - \frac{1}{\|\beta\|_2} x_0^T \beta = \frac{1}{\|\beta\|_2} x^T \beta + \frac{1}{\|\beta\|_2} \beta_0 \\
&= \frac{1}{\|\beta\|_2} (x^T \beta + \beta_0)
\end{aligned}
$$

Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

Rosenblatt’s perceptron learning algorithm

Tries to identify a separating hyperplane by minimizing the distance of misclassified points to the decision boundary.

If observation (x_i, y_i) with response $y_i = 1$ is misclassified, then $\beta_0 + x_i^T\beta < 0$, and if observation (x_i, y_i) with response $y_i = -1$ is misclassified, then $\beta_0 + x_i^T\beta > 0$.

We want to minimize

$$D(\beta_0, \beta) = -\sum_{i\in\mathcal{M}} y_i\left(\beta_0 + x_i^T\beta\right) = -\sum_{i\in\mathcal{M}} y_i\left(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\right)$$

where $\mathcal{M}$ be the set of misclassified observations and where for any $i \in \mathcal{M}$ we have $y_i(\beta_0 + x_i^T\beta) < 0$, leading to $D(\beta_0, \beta) > 0$.

Rosenblatt's perceptron learning algorithm

The quantity

$$D(\beta_0, \beta) = -\sum_{i \in \mathcal{M}} y_i\left(\beta_0 + x_i^T\beta\right)$$

is proportional to the distance of misclassified points to the decision boundary defined by $\beta_0 + x^T\beta = 0$, as $\beta_0 + x_i^T\beta$ is the signed distance, and multiplying by $-y_i$ ensures that this distance is positive.

Assuming that the set $\mathcal{M}$ of misclassified points is fixed, we compute the partial derivative of $D(\beta_0, \beta)$ with respect to β_k, $k = 0,1, \dots, p$, to compute the gradient.

The gradient of $D(\beta_0, \beta)$

$$D(\beta_0, \beta) = -\sum_{i \in \mathcal{M}} y_i \left(\beta_0 + \sum_{j=1}^{p} x_{ij} \beta_j \right)$$

Assuming $\mathcal{M}$ fixed

$$\frac{\partial}{\partial \beta_k} D(\beta_0, \beta) = \begin{cases} -\sum_{i \in \mathcal{M}} y_i & \text{if } k = 0 \\ -\sum_{i \in \mathcal{M}} y_i x_{ik} & \text{if } k = 1, 2, \ldots, p \end{cases}$$

which yields the gradient with respect to vector $\beta \in \mathbb{R}^p$ as

$$\frac{\partial}{\partial \beta} D(\beta_0, \beta) = -\sum_{i \in \mathcal{M}} y_i \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{bmatrix} = -\sum_{i \in \mathcal{M}} y_i x_i$$

Finding β_0 and β with stochastic gradient descent

Fix learning rate α

1. Randomly initialize β_0 and $\beta^T = [\beta_1, \beta_2, \dots, \beta_p]$.

2. Obtain the list $\mathcal{M}$ of misclassified training observations.

3. Randomly permute (shuffle) the order of the misclassified observations in $\mathcal{M}$ and update their indices.

4. For observation i in the permuted list $\mathcal{M}$ of misclassified training observations, update each of the $p + 1$ parameters as

$$\beta_k := \begin{cases} \beta_0 + \alpha y_i & \text{if } k = 0 \\ \beta_k + \alpha y_i x_{ik} & \text{if } k = 1,2,\dots,p \end{cases}$$

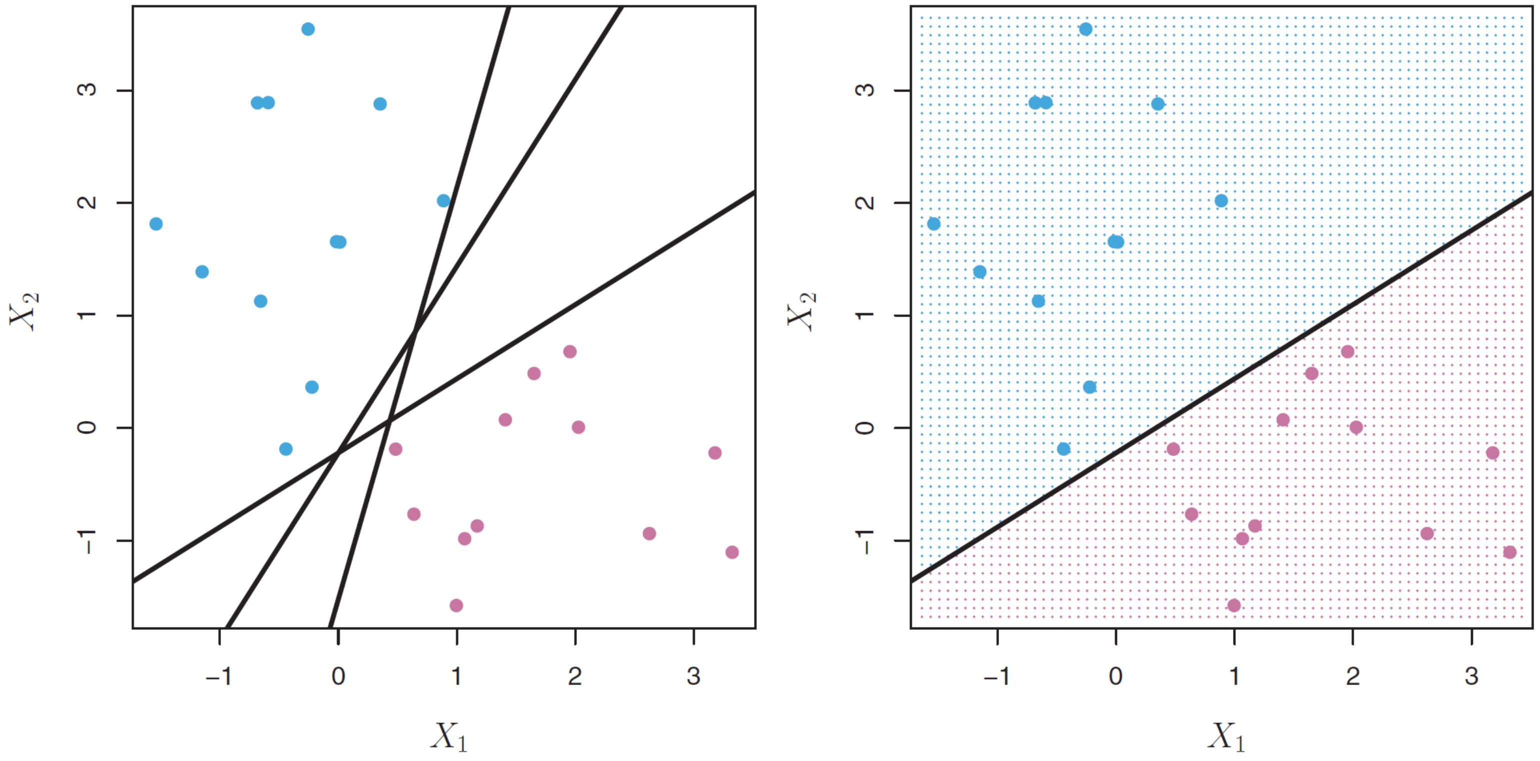
5. Repeat steps 2 through 4 until convergence.

Note: If the classes are linearly separable, then the algorithm converges to a separating hyperplane in a finite number of steps.

Issues with the perceptron learning algorithm

1. When data are separable, there are many solutions, and the one that is found depends on the starting values for β_0 and β.

   For example, 3 different hyperplanes depending on initial values (left), and decision rule for one of them (right).

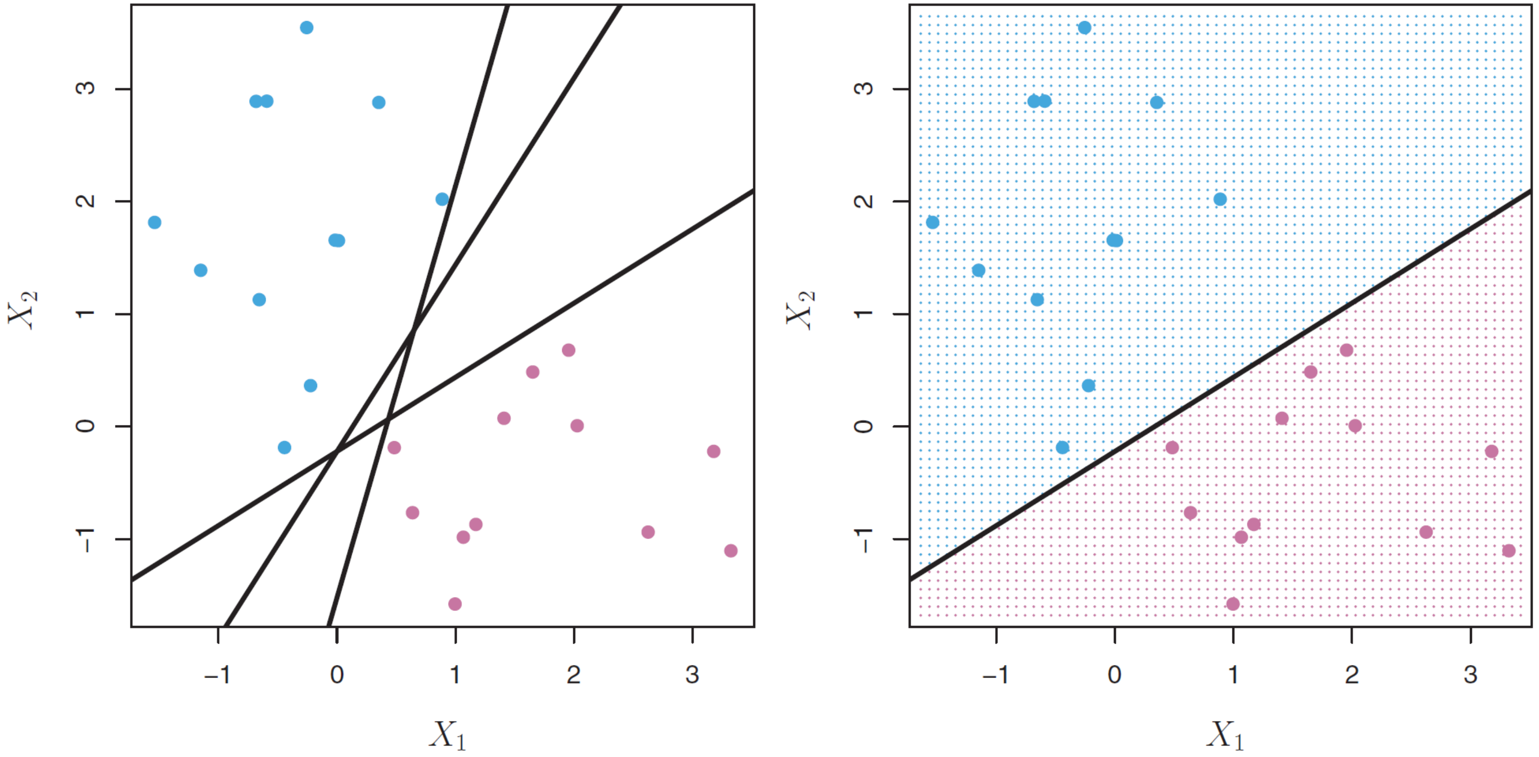


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Issues with the perceptron learning algorithm

2. Convergence in a finite number of steps can mean a very large (but still finite) number of steps.

The smaller the gap between classes, the longer time (larger number of steps) it will take to identify a separating hyperplane.

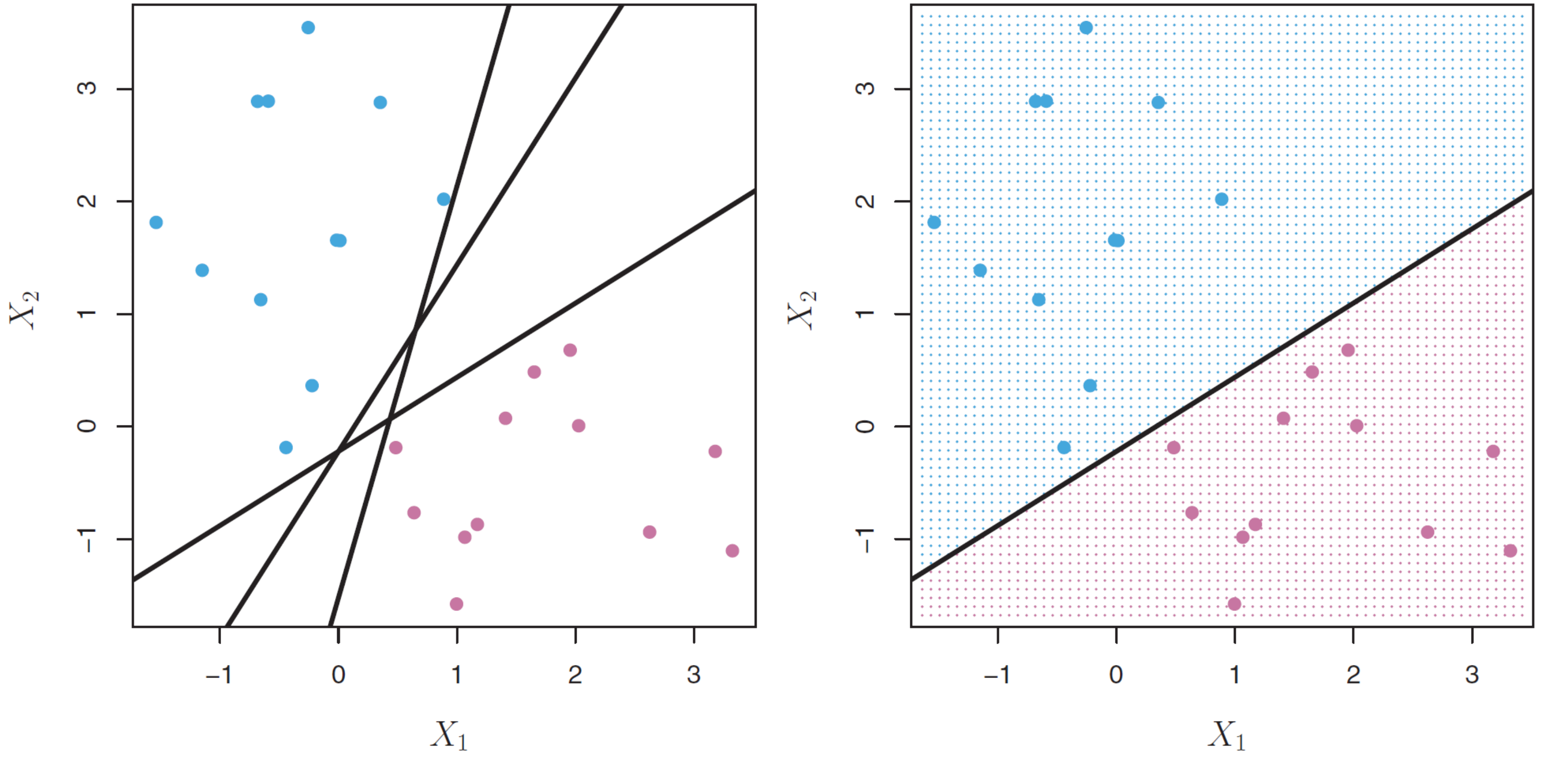


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Issues with the perceptron learning algorithm

3. When data are not separable, algorithm will not converge and cycles develop.

   Cycles can be long, and therefore difficult to detect.

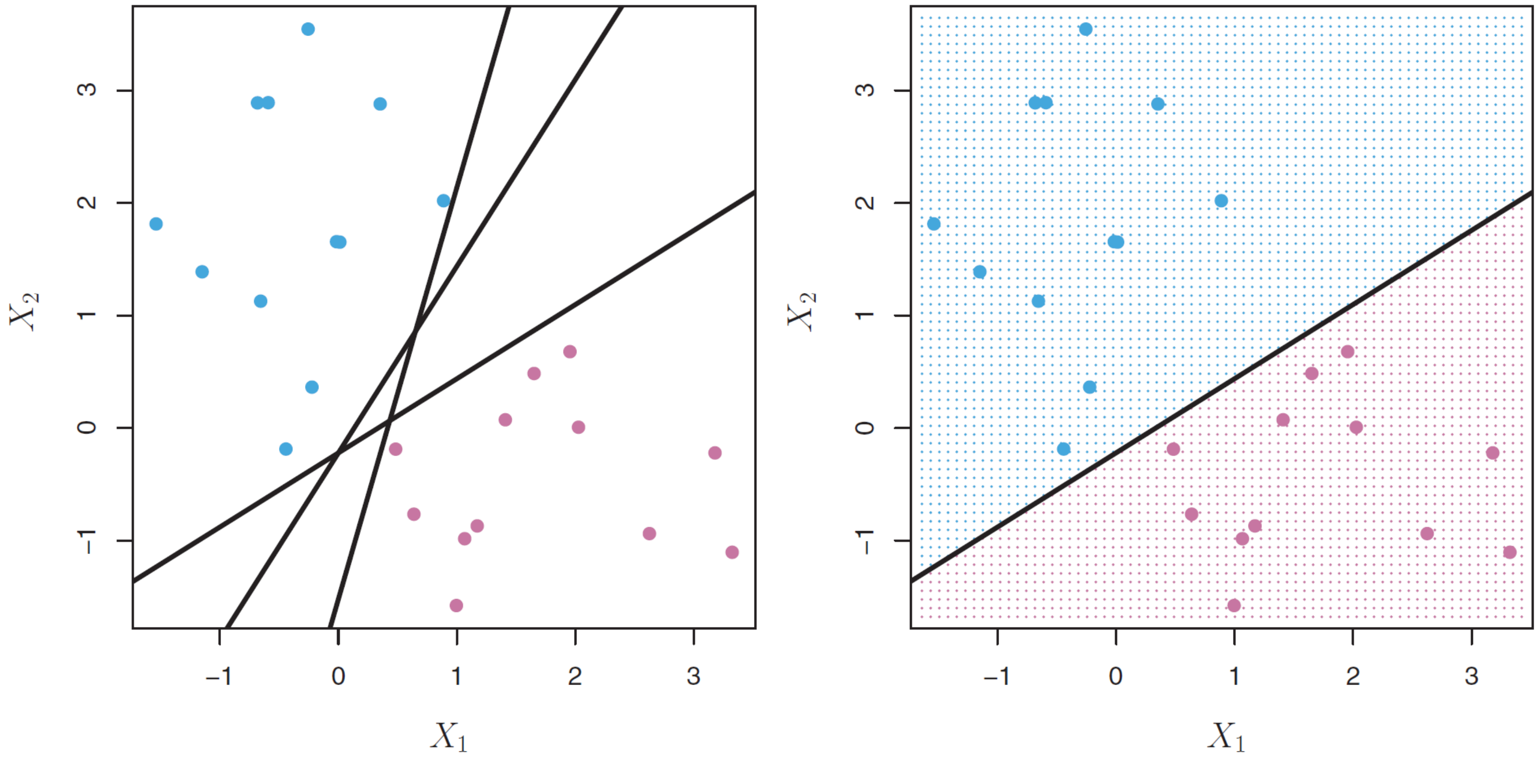


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Issues with the perceptron learning algorithm

Issue 1 related to there being more than one separating hyperplane can be addressed by adding constraints on the hyperplane.

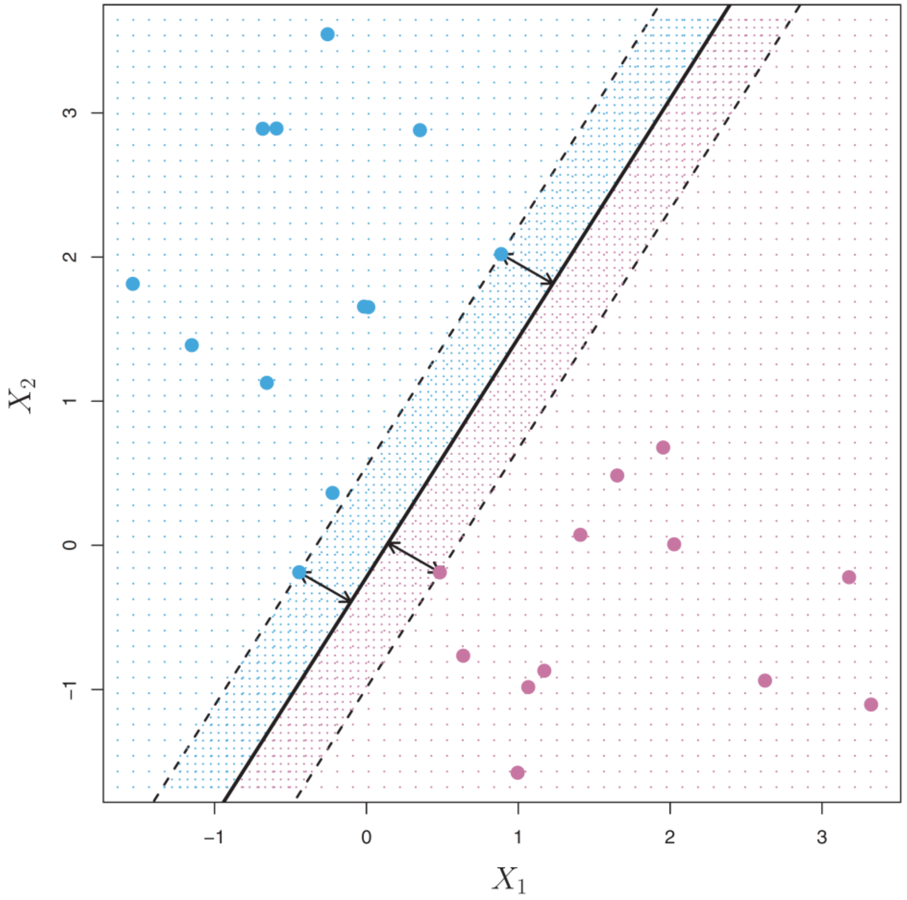
We address this here by next constructing optimal separating hyperplanes to construct a maximal margin classifier.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Optimal separating hyperplanes

We seek to choose a hyperplane that maximizes the margin (distance) between the hyperplane and the closest points for each class, with the distance (margin) from the hyperplane to the closest training observation in each class denoted by M.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Constructing an optimal separating hyperplane

We want to identify β_0 and unit length $\beta \in \mathbb{R}^p$ (*i.e.*, $\|\beta\|_2 = 1$) that maximize the margin M, subject to the constraint that

$$y_i\left(\beta_0 + x_i^T\beta\right) \geq M$$

for all observations $i = 1,2,\dots,N$.

That is, we want to solve the problem

$$\max_{\beta_0,\beta,\|\beta\|_2=1} M$$

$$\text{subject to } y_i\left(\beta_0 + x_i^T\beta\right) \geq M \text{ for } i = 1,2,\dots,N$$

Constructing an optimal separating hyperplane

Recall that when observation (x_i, y_i) is classified correctly, we have that $\beta_0 + x_i^T\beta < 0$ if $y_i = -1$ and $\beta_0 + x_i^T\beta > 0$ if $y_i = 1$.

Therefore, $y_i\left(\beta_0 + x_i^T\beta\right) > 0$ if observation (x_i, y_i) is classified correctly, then the constraint $y_i\left(\beta_0 + x_i^T\beta\right) \geq M$ enforces that it is at least a margin M from the hyperplane.

For data that are separable, we can always find a positive M.

Constructing an optimal separating hyperplane

We can include the unit length constraint ($\|\beta\|_2 = 1$) implicitly by restating the objective function and solve the problem

$$\max_{\beta_0,\beta} M$$

$$\text{subject to } \frac{1}{\|\beta\|_2} y_i\left(\beta_0 + x_i^T\beta\right) \geq M \text{ for } i = 1,2,\ldots,N$$

which also has the effect that it slightly redefines β_0 to be $\beta_0/\|\beta\|_2$.

Moreover, we can rewrite this problem as

$$\max_{\beta_0,\beta} M$$

$$\text{subject to } y_i\left(\beta_0 + x_i^T\beta\right) \geq M\|\beta\|_2 \text{for } i = 1,2,\ldots,N$$

Constructing an optimal separating hyperplane

Define $\|\beta\|_2 = 1/M$.

Instead of the objective as to identify β_0 and $\beta \in \mathbb{R}^p$ that <u>maximize</u> the margin M, subject to the constraint that

$$y_i\left(\beta_0 + x_i^T\beta\right) \geq M\|\beta\|_2$$

for all observations $i = 1,2,\dots,N$, we instead can have the objective as to identify β_0 and $\beta \in \mathbb{R}^p$ that <u>minimize</u> $\|\beta\|_2$, subject to the constraint that

$$y_i\left(\beta_0 + x_i^T\beta\right) \geq 1$$

These constraints define an empty slab (or margin) around the linear decision boundary of thickness $1/\|\beta\|_2$ and we choose β_0 and β to maximize this thickness.

Constructing an optimal separating hyperplane

Rewriting, we want to solve the problem

$$\min_{\beta_0,\beta} \frac{1}{2} ||\beta||_2^2$$

$$\text{subject to } y_i\left(\beta_0 + x_i^T\beta\right) \geq 1 \text{ for } i = 1,2,\ldots,N$$

which can be accomplished through convex optimization (specifically quadratic programming with linear constraints).

Before discussing the solution, we will start out by introducing a generalization to Lagrange multipliers for performing constrained optimization.

Lagrangian primal

Consider an optimization (Lagrangian primal) problem of the form

$$\min_x f(x)$$

$$\text{subject to } g_i(x) \leq 0 \text{ for } i = 1,2,\dots,k$$

The generalized Lagrange (primal) function is

$$\mathcal{L}(x,\alpha) = f(x) + \sum_{i=1}^{k} \alpha_i g_i(x)$$

where $\alpha^T = [\alpha_1, \alpha_2, \dots, \alpha_k]$ are Lagrange multipliers.

Lagrangian dual

The Lagrange dual function is defined as

$$\tilde{\mathcal{L}}(\alpha) = \min_{x \text{ constrained}} \mathcal{L}(x, \alpha)$$

The Lagrangian dual problem is

$$\max_{\alpha} \tilde{\mathcal{L}}(\alpha)$$

$$\text{subject to } \alpha_i \geq 0 \text{ for } i = 1,2, \ldots, k$$

and we have that

$$\tilde{\mathcal{L}}(\alpha) \leq f(x^\star) \text{ for all } \alpha$$

where $x^\star$ is the optimal vector.

We define a set of conditions for which $\tilde{\mathcal{L}}(\alpha^\star) = f(x^\star)$, where $\alpha^\star$ is the optimal α vector.

Karush-Kuhn-Tucker (KKT) conditions

If the constraints are affine, then the KKT conditions provide necessary and sufficient conditions for a point $x^\star$ to be an optimum.

$$\frac{\partial}{\partial x}\mathcal{L}(x,\alpha^\star)\bigg|_{x^\star} = 0 \quad \text{First}-\text{order derivative of optimality}$$

$$\alpha_i^\star g_i(x^\star) = 0 \qquad \text{Complementary slackness conditions}$$

$$\alpha_i^\star \geq 0 \qquad \text{Dual constraints}$$

$$g_i(x^\star) \leq 0 \qquad \text{Prime constraints}$$

The complementary slackness condition is essentially stating that if a dual variable $\alpha_i > 0$ (has slack), then the primal constraint $g_i(x) = 0$ (tight constraint, $i.e.$, no slack), and that if the primal constraint $g_i(x) < 0$ (has slack), then the dual variable $\alpha_i = 0$ (tight constraint, *i*.*e*., no slack).

Returning to the optimal separating hyperplane

Recall that we want to solve the

$$\min_{\beta_0,\beta} \frac{1}{2} \|\beta\|_2^2$$

$$\text{subject to } y_i\left(\beta_0 + x_i^T\beta\right) \geq 1 \text{ for } i = 1,2,\dots,N$$

We can obtain the Langrangian primal function for this constrained optimization problem as

$$\mathcal{L}(\beta_0,\beta,\alpha) = \frac{1}{2}\|\beta\|_2^2 + \sum_{i=1}^{N} \alpha_i\left[1 - y_i\left(\beta_0 + x_i^T\beta\right)\right]$$

$$= \frac{1}{2}\sum_{j=1}^{p} \beta_j^2 + \sum_{i=1}^{N} \alpha_i\left[1 - y_i\left(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\right)\right]$$

Partial derivative with respect to β_0

To obtain the Langrange dual $\tilde{\mathcal{L}}(\alpha)$, we first take the derivatives of

$$\mathcal{L}(\beta_0, \beta, \alpha) = \frac{1}{2}\sum_{j=1}^{p} \beta_j^2 + \sum_{i=1}^{N} \alpha_i \left[1 - y_i\left(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\right)\right]$$

with respect to $\beta_k, k = 0,1, \dots, p$ and set them to 0.

For $k = 0$, we have

$$\frac{\partial}{\partial \beta_0}\mathcal{L}(\beta_0, \beta, \alpha) = -\sum_{i=1}^{N} \alpha_i y_i$$

Setting to 0 gives

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

Partial derivative with respect to β_k, $k = 1,2,\ldots,p$

To obtain the Langrange dual $\tilde{\mathcal{L}}(\alpha)$, we first take the derivatives of

$$\mathcal{L}(\beta_0, \beta, \alpha) = \frac{1}{2}\sum_{j=1}^{p}\beta_j^2 + \sum_{i=1}^{N}\alpha_i\left[1 - y_i\left(\beta_0 + \sum_{j=1}^{p}x_{ij}\beta_j\right)\right]$$

with respect to $\beta_k, k = 0,1,\ldots,p$ and set them to 0.

For $k = 1,2,\ldots,p$, we have

$$\frac{\partial}{\partial\beta_k}\mathcal{L}(\beta_0, \beta, \alpha) = \beta_k - \sum_{i=1}^{N}\alpha_i y_i x_{ik}$$

Setting to 0 gives

$$\beta_k^\star = \sum_{i=1}^{N}\alpha_i y_i x_{ik}$$

Obtaining the parameter vector $\beta^\star$

We then have the parameter vector as

$$\beta^\star = \begin{bmatrix} \beta_1^\star \\ \beta_2^\star \\ \vdots \\ \beta_p^\star \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} \alpha_i y_i x_{i1} \\ \sum_{i=1}^{N} \alpha_i y_i x_{i2} \\ \vdots \\ \sum_{i=1}^{N} \alpha_i y_i x_{ip} \end{bmatrix} = \sum_{i=1}^{N} \alpha_i y_i \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{bmatrix} = \sum_{i=1}^{N} \alpha_i y_i x_i$$

Plugging back into the Lagrange primal function yields the Lagrange dual function.

The Lagrange dual function

The Langrange dual function is

$$\begin{aligned}
\tilde{\mathcal{L}}(\alpha) &= \frac{1}{2}\sum_{j=1}^{p} {\beta_j^\star}^2 + \sum_{i=1}^{N} \alpha_i \left[1 - y_i\left(\beta_0 + x_i^T \beta^\star\right)\right] \\
&= \frac{1}{2}\sum_{j=1}^{p} {\beta_j^\star}^2 + \sum_{i=1}^{N} \alpha_i - \beta_0 \sum_{i=1}^{N} \alpha_i y_i - \sum_{i=1}^{N} \alpha_i y_i x_i^T \beta^\star \\
&= \frac{1}{2}\sum_{j=1}^{p} \left(\sum_{i=1}^{N} \alpha_i y_i x_{ij}\right)^2 + \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N} \alpha_i y_i x_i^T \sum_{k=1}^{N} \alpha_k y_k x_k \\
&= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k
\end{aligned}$$

subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$.

The Lagrange dual function

The solution $\alpha^\star$ is obtained by maximizing $\tilde{\mathcal{L}}(\alpha)$ and requiring the solution to satisfy the KTT conditions

$\beta^\star = \sum_{i=1}^{N} \alpha_i^\star y_i x_i$	First−order derivative of optimality
$\sum_{i=1}^{N} \alpha_i^\star y_i = 0$	First−order derivative of optimality
$\alpha_i^\star \left[1 - y_i\left(\beta_0^\star + x_i^T \beta^\star\right)\right] = 0$	Complementary slackness conditions
$\alpha_i^\star \geq 0$	Dual constraints
$1 - y_i\left(\beta_0^\star + x_i^T \beta^\star\right) \leq 0$	Prime constraints

Defining the support points

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i \qquad \text{First-order derivative of optimality}$$

$$\alpha_i\left[1 - y_i\left(\beta_0 + x_i^T\beta\right)\right] = 0 \qquad \text{Complementary slackness conditions}$$

$$\alpha_i \geq 0 \qquad \text{Dual constraints}$$

If $\alpha_i > 0$, then $y_i(\beta_0 + x_i^T\beta) = 1$ due to complementary slackness, which indicates that x_i is on the boundary of the slab.

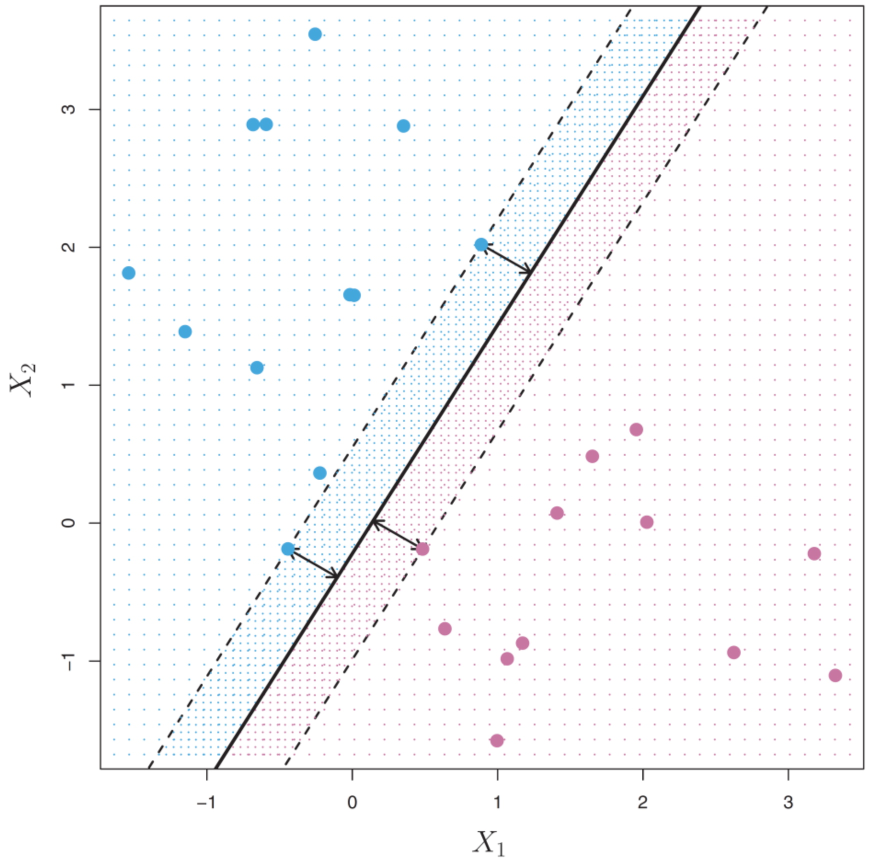
If $y_i\left(\beta_0 + x_i^T\beta\right) > 1$, then $\alpha_i = 0$ and x_i is not on boundary of the slab.

The solution $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$ is defined in terms of a linear combination of **support points** x_i that lie on the boundary of the slab, and result in $\alpha_i > 0$.

Maximal margin classifier

The optimal separating hyperplane has the form $\hat{f}(X) = \hat{\beta}_0 + X^T\hat{\beta} = 0$, and the classification under the **maximal margin classifier** is

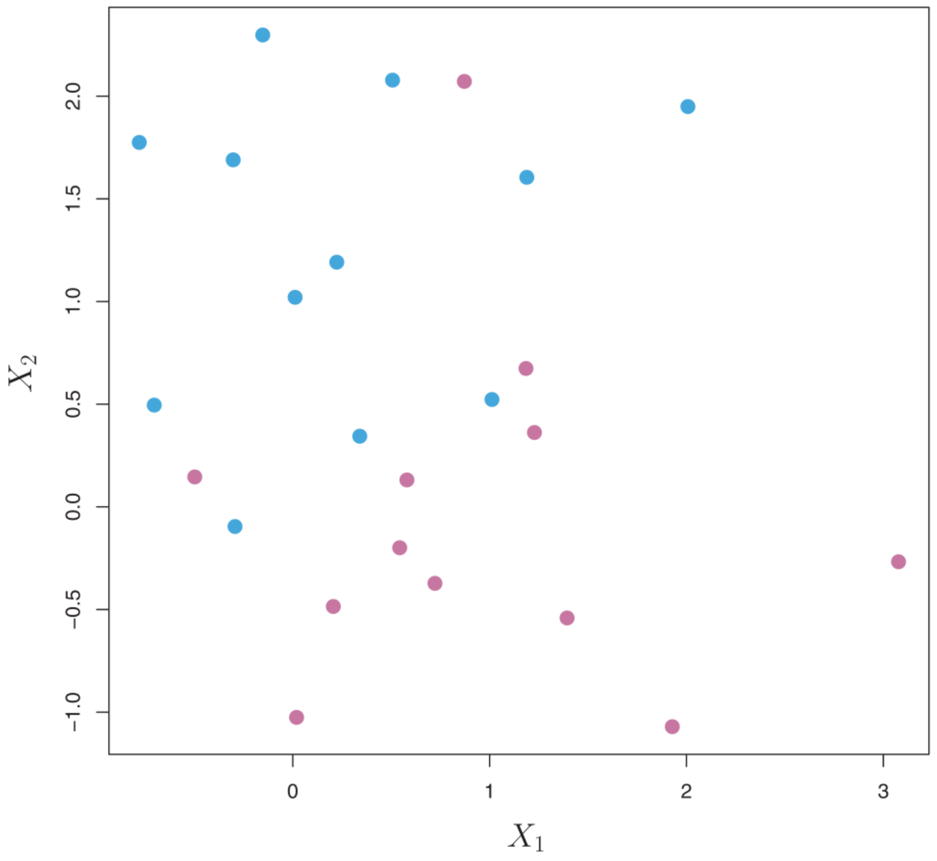
$$\hat{Y}(X) = \operatorname{sign}\left(\hat{f}(X)\right)$$



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

What about non-separable cases?

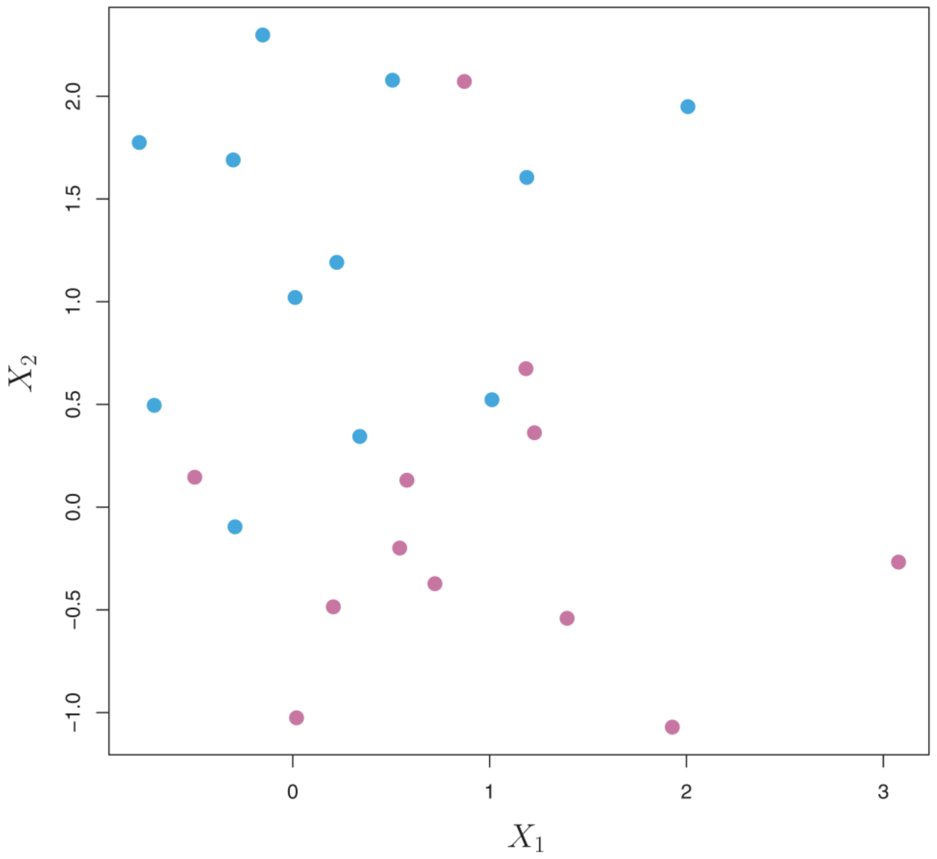
The maximal margin classifier assumed that we could identify a hyperplane that completely separates data points from different classes such that the margin between the closest points and the hyperplane was maximized.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

What about non-separable cases?

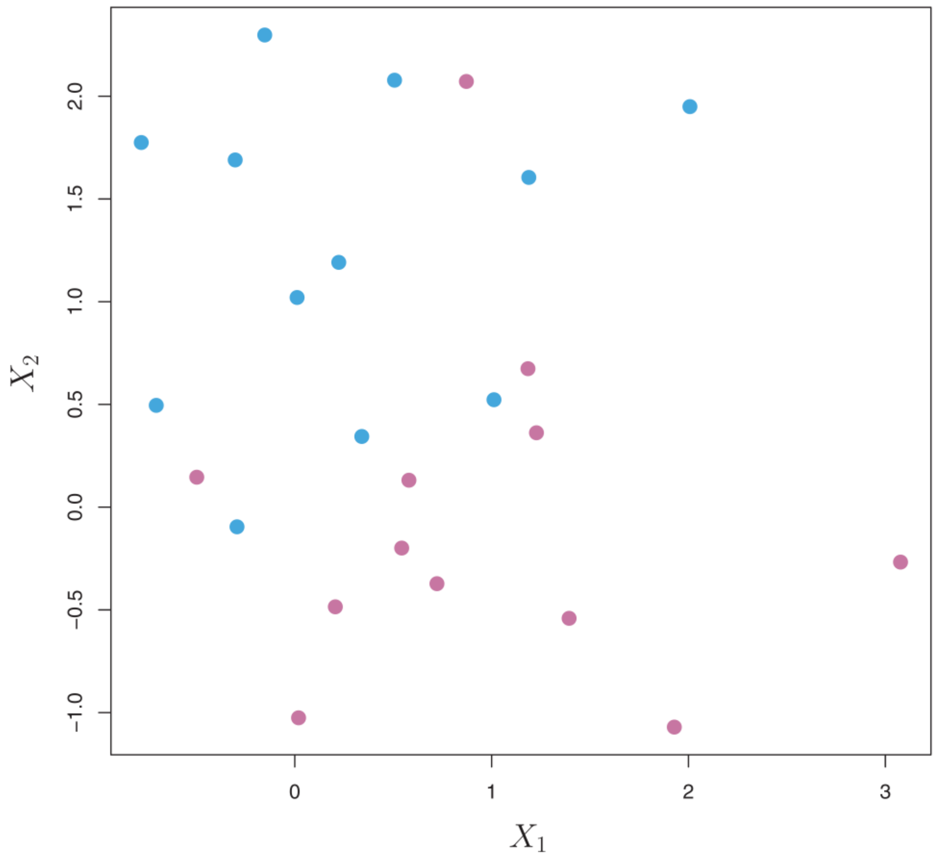
In most cases, the data will not be separable, such that classes overlap.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

What about non-separable cases?

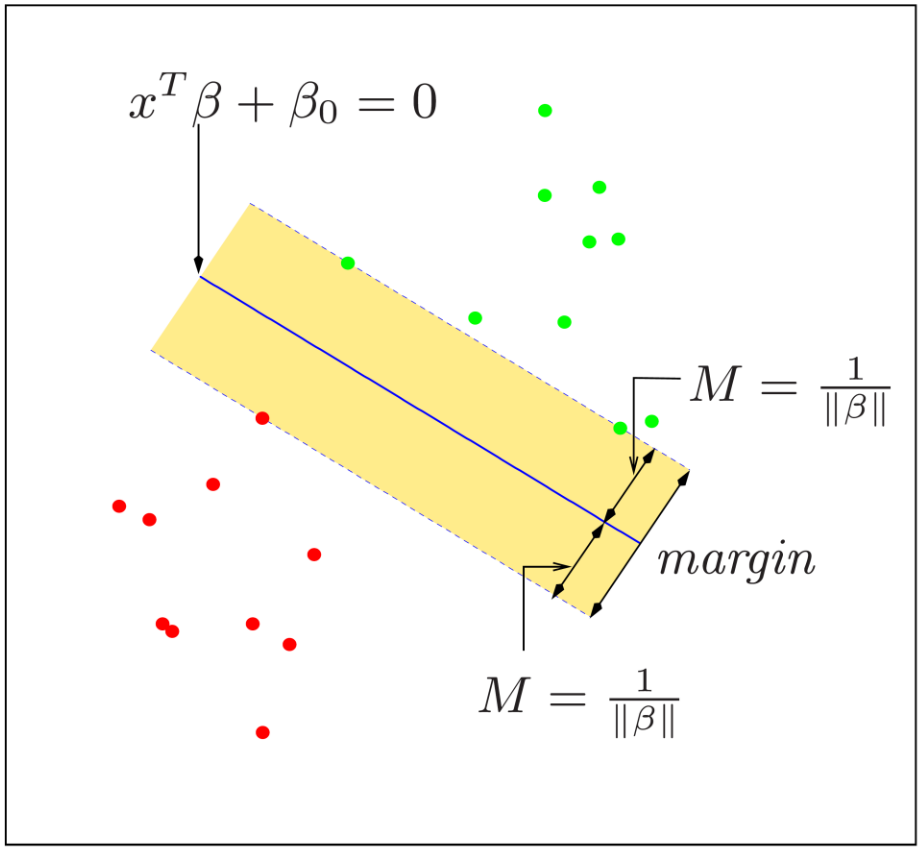
We can instead still try to find a hyperplane that maximizes the margin, while permitting points to be on the wrong side of the margin, and we will ultimately call a classifier based on this idea the **support vector classifier**.

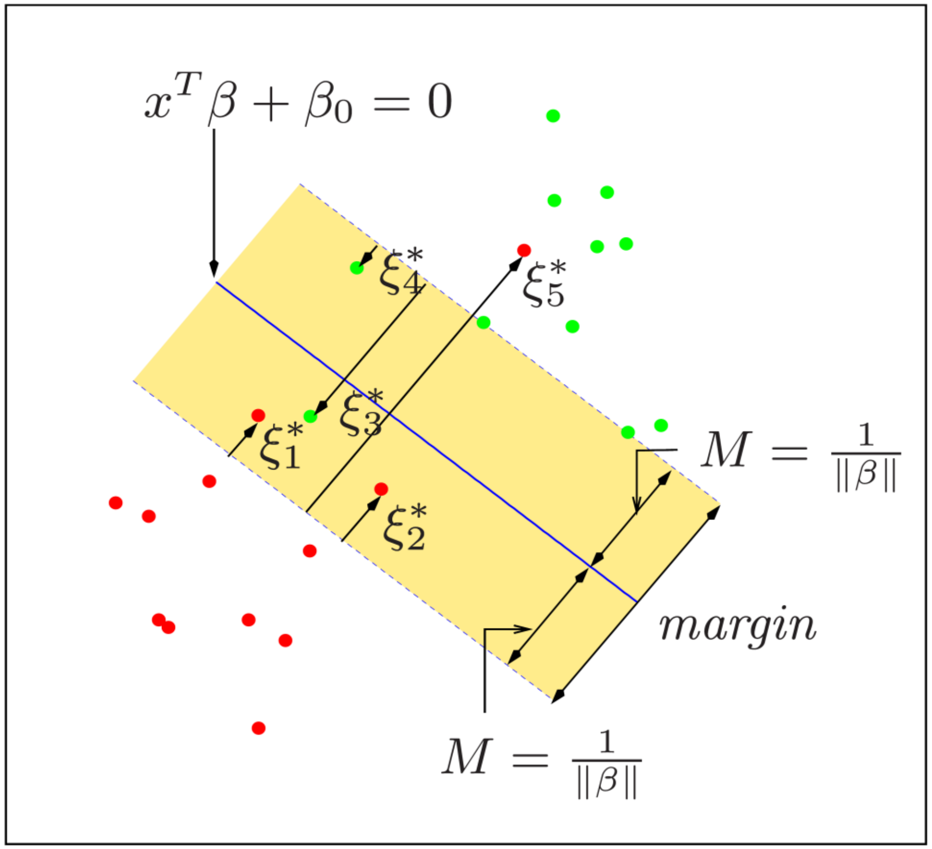


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Slack variables

Let $\xi = [\xi_1, \xi_2, \ldots, \xi_N]^T$ denote the vector of **slack variables**, in which $\xi_i \geq 0$ for $i = 1,2,\ldots,N$. The value of ξ_i is proportional to the amount by which the prediction $f(x_i) = \beta_0 + x_i^T\beta$ is on the wrong side of the margin (right).





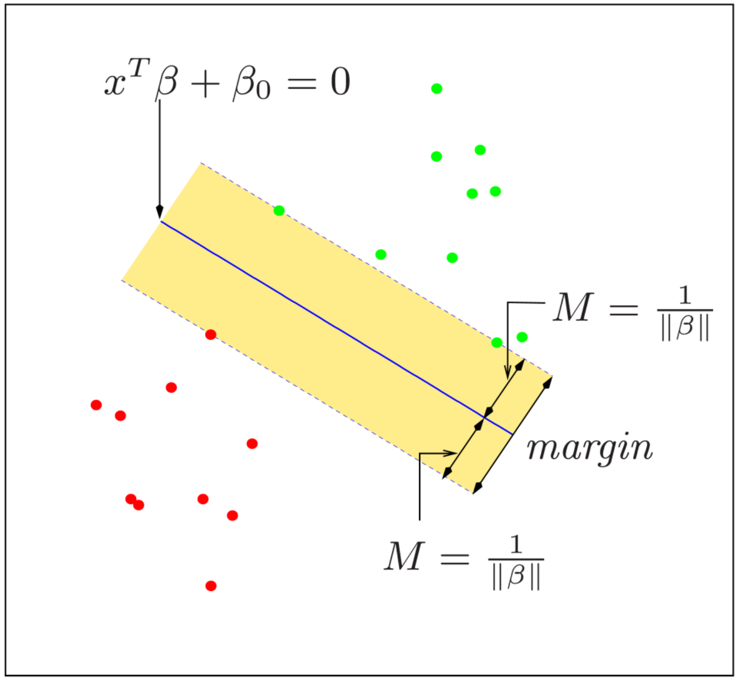
Hastie *et al* (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd Ed.

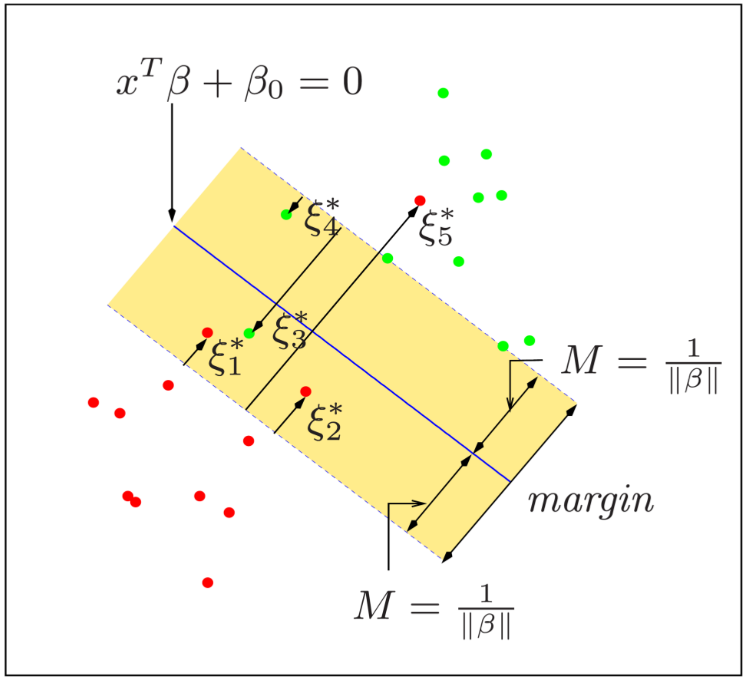
Slack variables

Misclassifications occur when $\xi_i > 1$, and so we will introduce a constant K, $K \geq 0$, such that

$$\sum_{i=1}^{N} \xi_i \leq K$$

where K bounds the total number of training misclassifications at K.





Formulating the optimization problem

Recall that when identifying the optimal separating hyperplane, we wanted to solve the

$$\min_{\beta_0,\beta} \frac{1}{2} \|\beta\|_2^2$$

$$\text{subject to } y_i\left(\beta_0 + x_i^T \beta\right) \geq 1 \text{ for } i = 1,2, \dots, N$$

We can incorporate slack variables into this optimization problem as

$$\min_{\beta_0,\beta} \frac{1}{2} \|\beta\|_2^2$$

$$\text{subject to } y_i\left(\beta_0 + x_i^T \beta\right) \geq 1 - \xi_i \text{ for } i = 1,2, \dots, N$$

$$\xi_i \geq 0$$

$$\sum_{i=1}^{N} \xi_i \leq K$$

Formulating the optimization problem

We can reformulate the problem as

$$\min_{\beta_0,\beta} \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to } y_i\left(\beta_0 + x_i^T\beta\right) \geq 1 - \xi_i \text{ for } i = 1,2,\dots,N$$

$$\xi_i \geq 0$$

Denoting $\mu^T = [\mu_1, \mu_2, \dots, \mu_N]$, the Lagrangian primal function is

$\mathcal{L}(\beta_0, \beta, \xi, \alpha, \mu)$

$$= \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i\left[1 - \xi_i - y_i\left(\beta_0 + x_i^T\beta\right)\right] - \sum_{i=1}^{N}\mu_i\xi_i$$

$$= \frac{1}{2}\sum_{j=1}^{p}\beta_j^2 + \sum_{i=1}^{N}\alpha_i\left[1 - y_i\left(\beta_0 + \sum_{j=1}^{p}x_{ij}\beta_j\right)\right] + \sum_{i=1}^{N}(C - \alpha_i - \mu_i)\xi_i$$

Partial derivative with respect to β_0

To obtain the Langrange dual $\tilde{\mathcal{L}}(\alpha, \mu)$, we first take the derivatives of $\mathcal{L}(\beta_0, \beta, \xi, \alpha, \mu)$

$$= \frac{1}{2}\sum_{j=1}^{p} \beta_j^2 + \sum_{i=1}^{N} \alpha_i \left[1 - y_i \left(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\right)\right] + \sum_{i=1}^{N} (C - \alpha_i - \mu_i)\xi_i$$

with respect to $\beta_k, k = 0,1, \dots, p$ and ξ_k, $k = 1,2, \dots, N$ and set to 0.

For $k = 0$, we have

$$\frac{\partial}{\partial \beta_0} \mathcal{L}(\beta_0, \beta, \xi, \alpha, \mu) = -\sum_{i=1}^{N} \alpha_i y_i$$

Setting to 0 gives

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

Partial derivative with respect to β_k, $k = 1,2,\ldots,p$

To obtain the Langrange dual $\tilde{\mathcal{L}}(\alpha,\mu)$, we first take the derivatives of $\mathcal{L}(\beta_0,\beta,\xi,\alpha,\mu)$

$$= \frac{1}{2}\sum_{j=1}^{p}\beta_j^2 + \sum_{i=1}^{N}\alpha_i\left[1 - y_i\left(\beta_0 + \sum_{j=1}^{p}x_{ij}\beta_j\right)\right] + \sum_{i=1}^{N}(C-\alpha_i-\mu_i)\xi_i$$

with respect to $\beta_k, k = 0,1,\ldots,p$ and ξ_k, $k = 1,2,\ldots,N$ and set to 0.

For $k = 1,2,\ldots,p$, we have

$$\frac{\partial}{\partial\beta_k}\mathcal{L}(\beta_0,\beta,\xi,\alpha,\mu) = \beta_k - \sum_{i=1}^{N}\alpha_i y_i x_{ik}$$

Setting to 0 gives

$$\beta_k^\star = \sum_{i=1}^{N}\alpha_i y_i x_{ik}$$

Partial derivative with respect to ξ_k, $k = 1,2, \dots, N$

To obtain the Langrange dual $\tilde{\mathcal{L}}(\alpha, \mu)$, we first take the derivatives of $\mathcal{L}(\beta_0, \beta, \xi, \alpha, \mu)$

$$= \frac{1}{2}\sum_{j=1}^{p} \beta_j^2 + \sum_{i=1}^{N} \alpha_i \left[1 - y_i \left(\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\right)\right] + \sum_{i=1}^{N} (C - \alpha_i - \mu_i)\xi_i$$

with respect to $\beta_k, k = 0,1, \dots, p$ and ξ_k, $k = 1,2, \dots, N$ and set to 0.

For $k = 1,2, \dots, N$, we have

$$\frac{\partial}{\partial \xi_k} \mathcal{L}(\beta_0, \beta, \xi, \alpha, \mu) = C - \alpha_k - \mu_k$$

Setting to 0 gives

$$\alpha_k^\star = C - \mu_k$$

Obtaining the parameter vector $\beta^\star$

We then have the parameter vector as

$$\beta^\star = \begin{bmatrix} \beta_1^\star \\ \beta_2^\star \\ \vdots \\ \beta_p^\star \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} \alpha_i y_i x_{i1} \\ \sum_{i=1}^{N} \alpha_i y_i x_{i2} \\ \vdots \\ \sum_{i=1}^{N} \alpha_i y_i x_{ip} \end{bmatrix} = \sum_{i=1}^{N} \alpha_i y_i \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{bmatrix} = \sum_{i=1}^{N} \alpha_i y_i x_i$$

Plugging $\beta^\star$ and $\alpha_i = C - \mu_i$ back into the Lagrange primal function yields the Lagrange dual function.

The Lagrange dual function

The Langrange dual function is

$$\tilde{\mathcal{L}}(\alpha,\mu)=\frac{1}{2}\sum_{j=1}^{p}\beta_j^{\star 2}+\sum_{i=1}^{N}\alpha_i\left[1-y_i\left(\beta_0+x_i^T\beta^\star\right)\right]+\sum_{i=1}^{N}(C-\alpha_i-\mu_i)\xi_i$$

$$=\frac{1}{2}\sum_{j=1}^{p}\beta_j^{\star 2}+\sum_{i=1}^{N}\alpha_i-\beta_0\sum_{i=1}^{N}\alpha_i y_i-\sum_{i=1}^{N}\alpha_i y_i x_i^T\beta^\star+\sum_{i=1}^{N}(C-\alpha_i-\mu_i)\xi_i$$

$$=\frac{1}{2}\sum_{j=1}^{p}\left(\sum_{i=1}^{N}\alpha_i y_i x_{ij}\right)^2+\sum_{i=1}^{N}\alpha_i-\sum_{i=1}^{N}\alpha_i y_i x_i^T\sum_{k=1}^{N}\alpha_k y_k x_k$$

$$=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N}\alpha_i\alpha_k y_i y_k x_i^T x_k$$

subject to $0\le\alpha_i\le C$, $\xi_i\ge 0$, $\mu_i\ge 0$, and $\sum_{i=1}^{N}\alpha_i y_i=0$.

The Lagrange dual function

Solution $\alpha^\star$ is obtained by maximizing $\tilde{\mathcal{L}}(\alpha, \mu)$ under KKT conditions

$\beta^\star = \sum_{i=1}^{N} \alpha_i^\star y_i x_i$	First−order derivative of optimality
$\sum_{i=1}^{N} \alpha_i^\star y_i = 0$	First−order derivative of optimality
$\alpha_i^\star = C - \mu_i^\star$	First−order derivative of optimality
$\alpha_i^\star\left[1 - \xi^\star - y_i\left(\beta_0^\star + x_i^T \beta^\star\right)\right] = 0$	Complementary slackness conditions
$\mu_i^\star \xi_i^\star = 0$	Complementary slackness conditions
$0 \leq \alpha_i^\star \leq C$	Dual constraints
$\mu_i^\star \geq 0$	Dual constraints
$\alpha_i^\star \geq 0$	Dual constraints
$1 - \xi_i^\star - y_i\left(\beta_0^\star + x_i^T \beta^\star\right) \leq 0$	Prime constraints

Defining the support vectors

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i \qquad \text{First-order derivative of optimality}$$

$$\alpha_i\left[1 - \xi_i - y_i\left(\beta_0 + x_i^T\beta\right)\right] = 0 \quad \text{Complementary slackness conditions}$$

$$\mu_i \xi_i = 0 \qquad \text{Complementary slackness conditions}$$

$$0 \le \alpha_i \le C \qquad \text{Dual constraints}$$

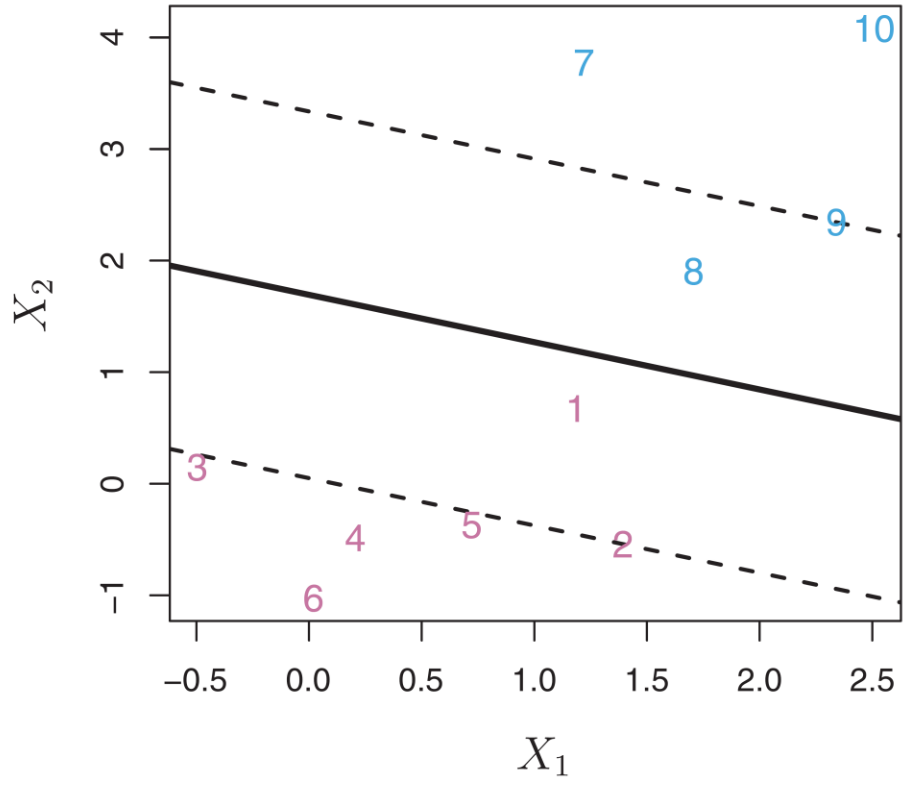
If $\alpha_i > 0$, then $y_i\left(\beta_0 + x_i^T\beta\right) = 1 - \xi_i$. Some points will lie on the boundary of the slab ($\xi_i = 0$ with $0 < \alpha_i < C$), with the remainder on the wrong side of margin ($\xi_i > 0$ with $\alpha_i = C$).

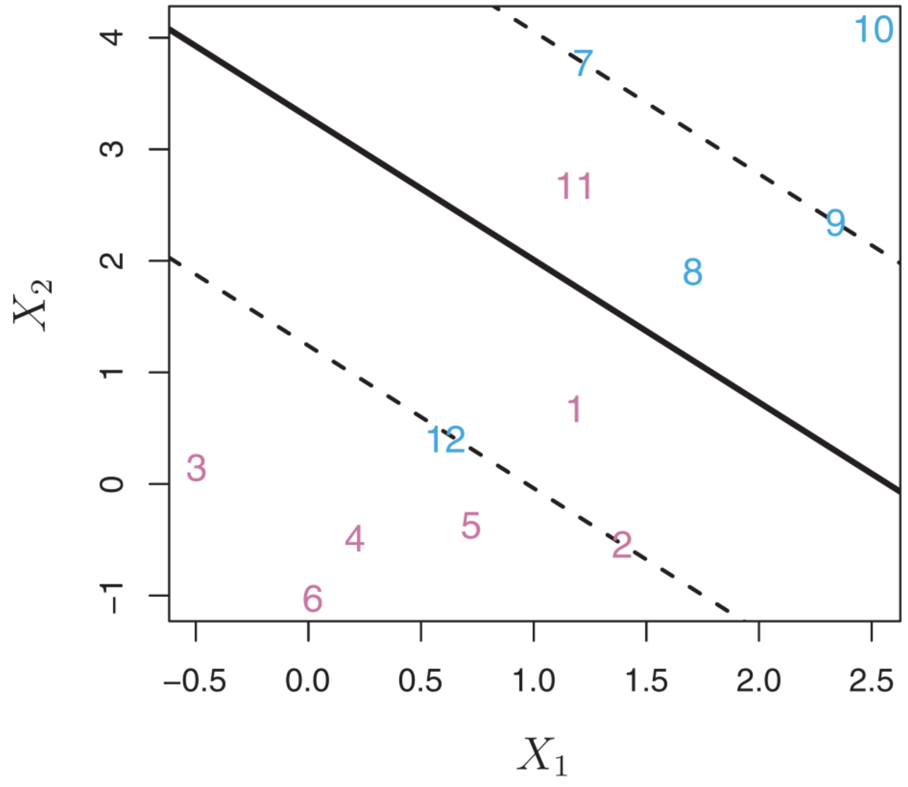
The solution $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$ is defined in terms of a linear combination of **support vectors** x_i that lie either on the boundary or the incorrect side of the margin, and result in $\alpha_i > 0$.

Support vector classifier

The optimal separating hyperplane has the form $\hat{f}(X) = \hat{\beta}_0 + X^T\hat{\beta}$, and the classification under the **support vector classifier** is

$$\hat{Y}(X) = \text{sign}\left(\hat{f}(X)\right)$$

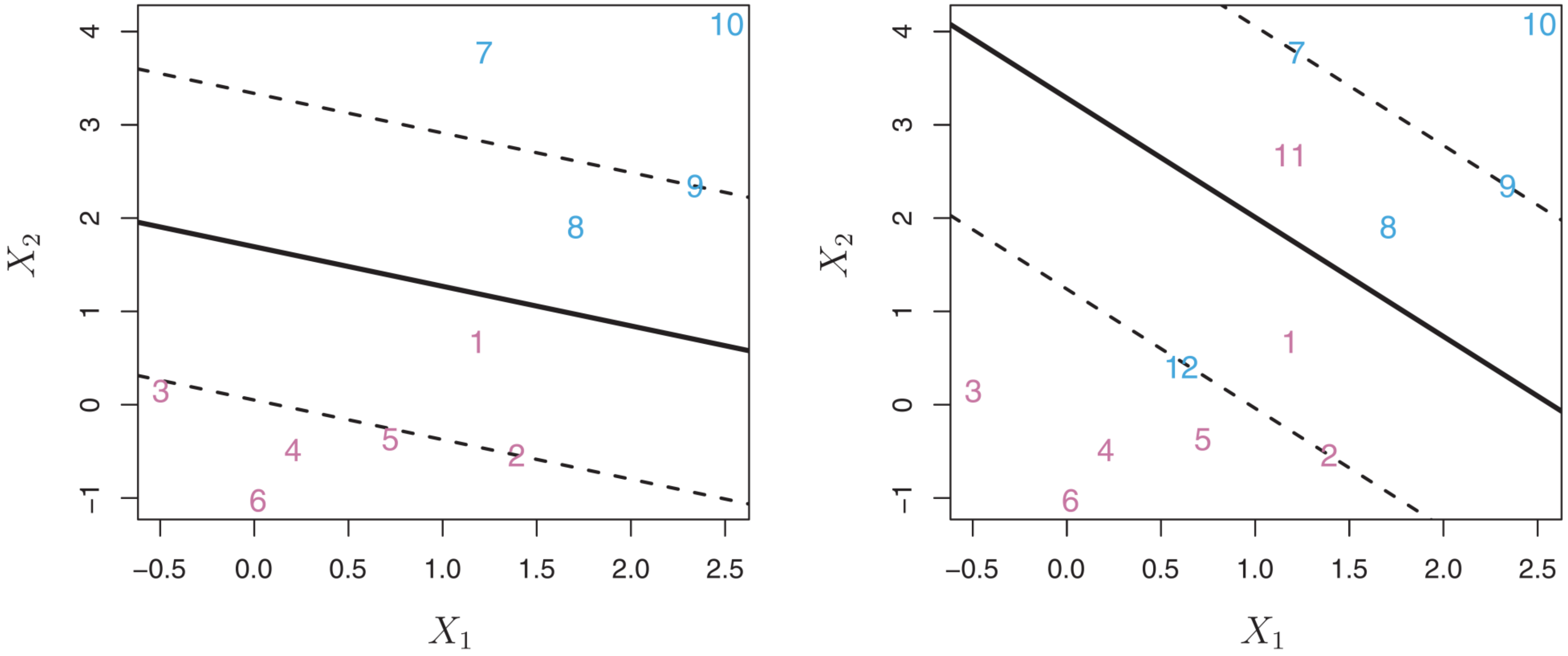




James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Support vector classifier

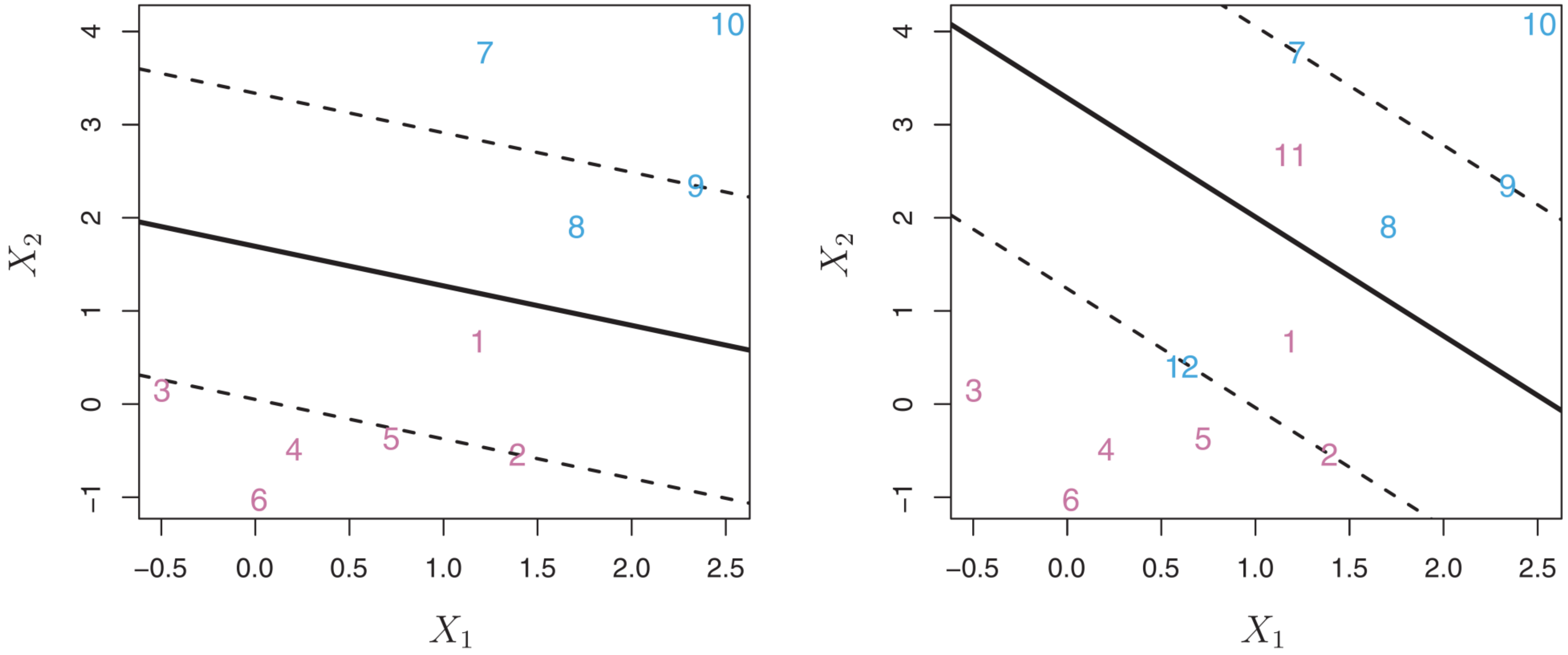
If the classes are separable, then some observations can still have $\xi_i > 0$, as they are on the incorrect side of the margin, but on the correct side of the hyperplane (left).



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Support vector classifier

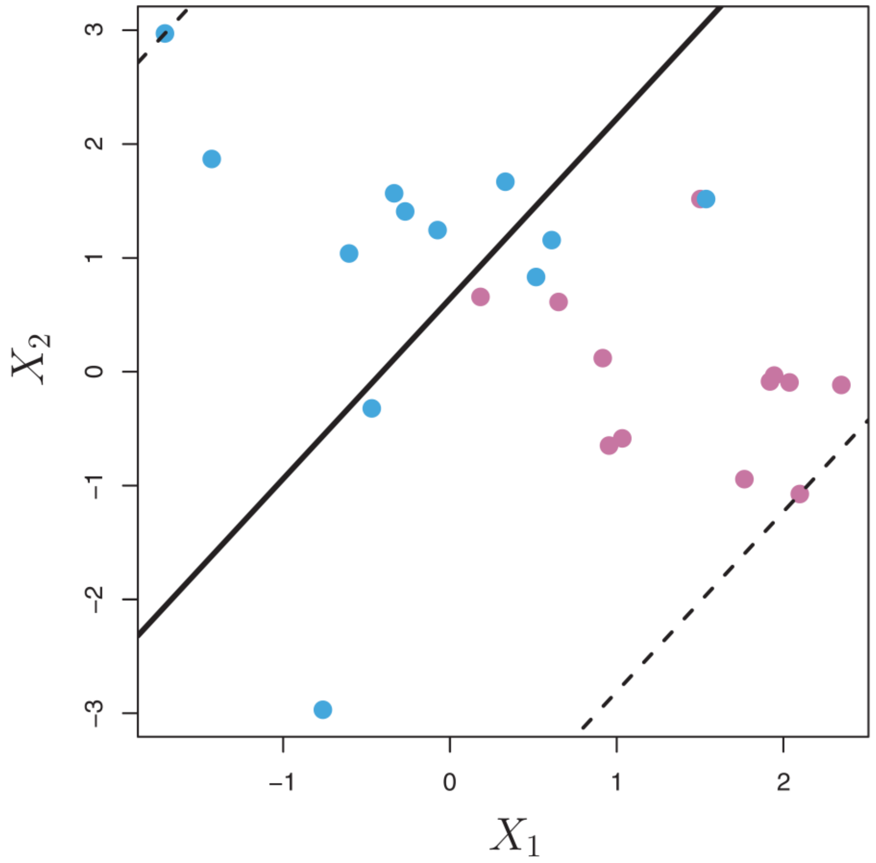
If the classes are non-separable, then some observations will have $\xi_i > 1$, as they are on the incorrect side of the hyperplane (right).



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Effect of tuning parameter C

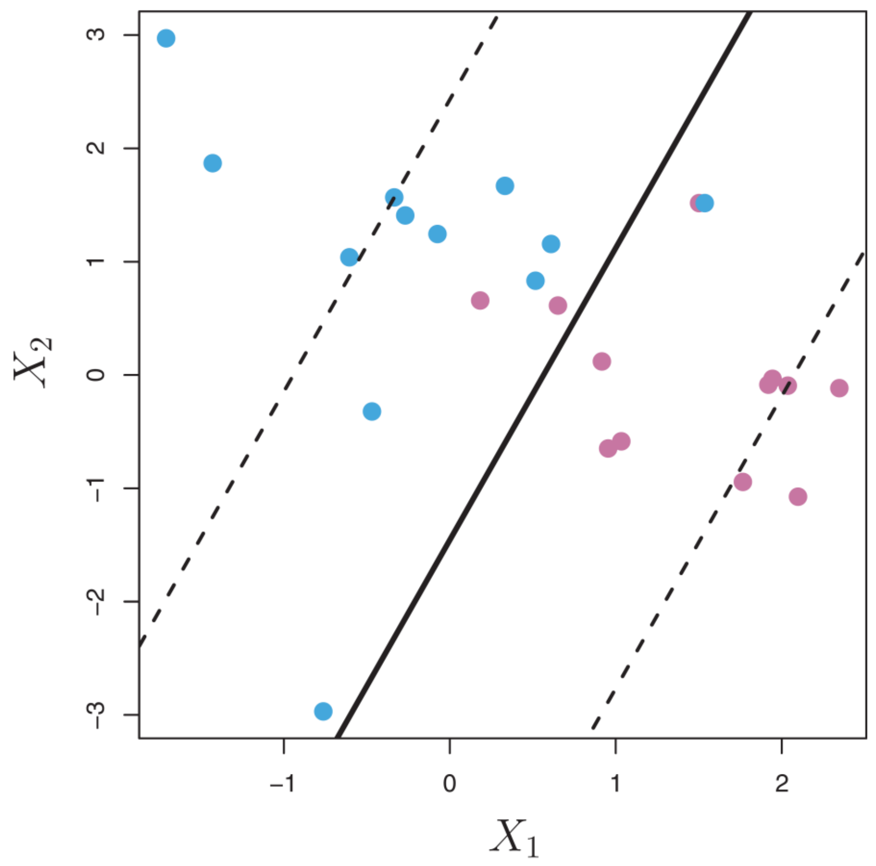
Small values for the tuning parameter C lead to many observations with positive slack $\xi_i > 0$, some of which are simply on the wrong side of the margin, and others that are on the wrong side of the hyperplane.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Effect of tuning parameter C

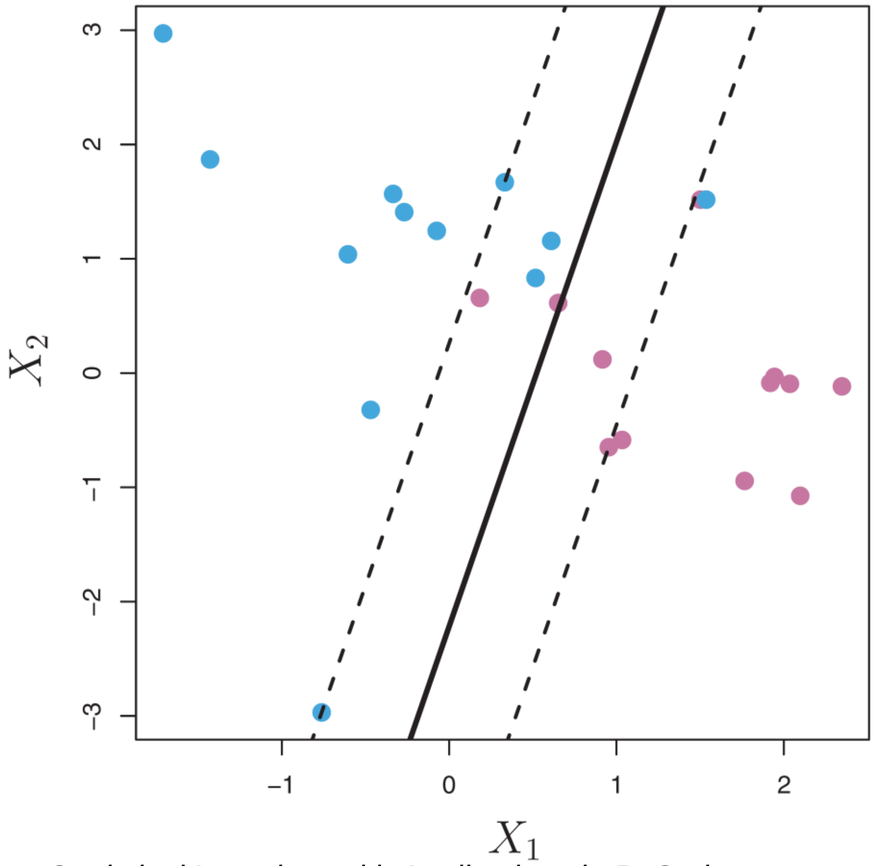
Moderate values for the tuning parameter C lead to some observations with positive slack $\xi_i > 0$, only a few of which may be on the wrong side of the hyperplane.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Effect of tuning parameter C

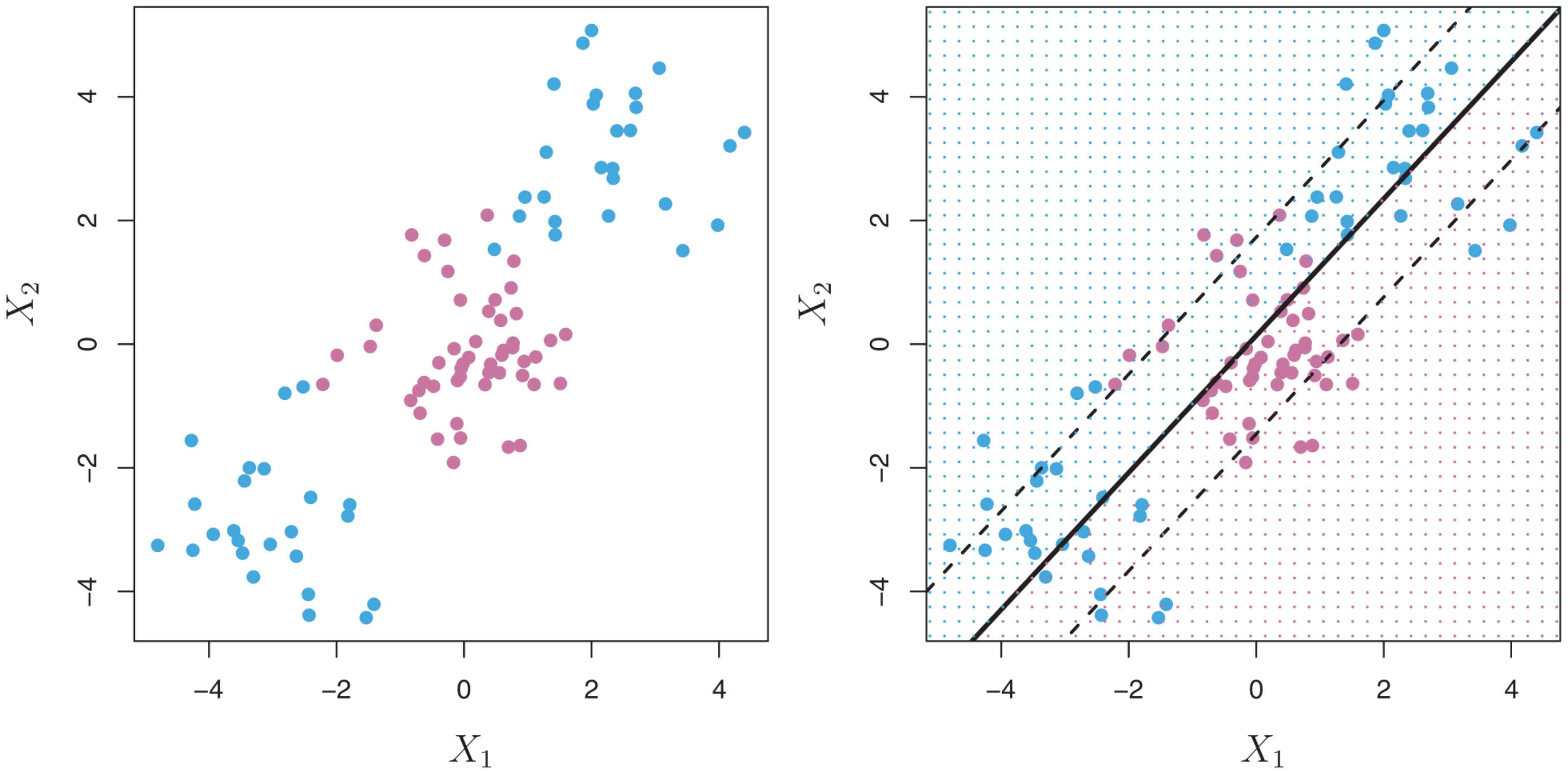
Large values for the tuning parameter C lead to few observations with positive slack $\xi_i > 0$, with most points on the correct side of the hyperplane.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Application on non-linear decision boundaries

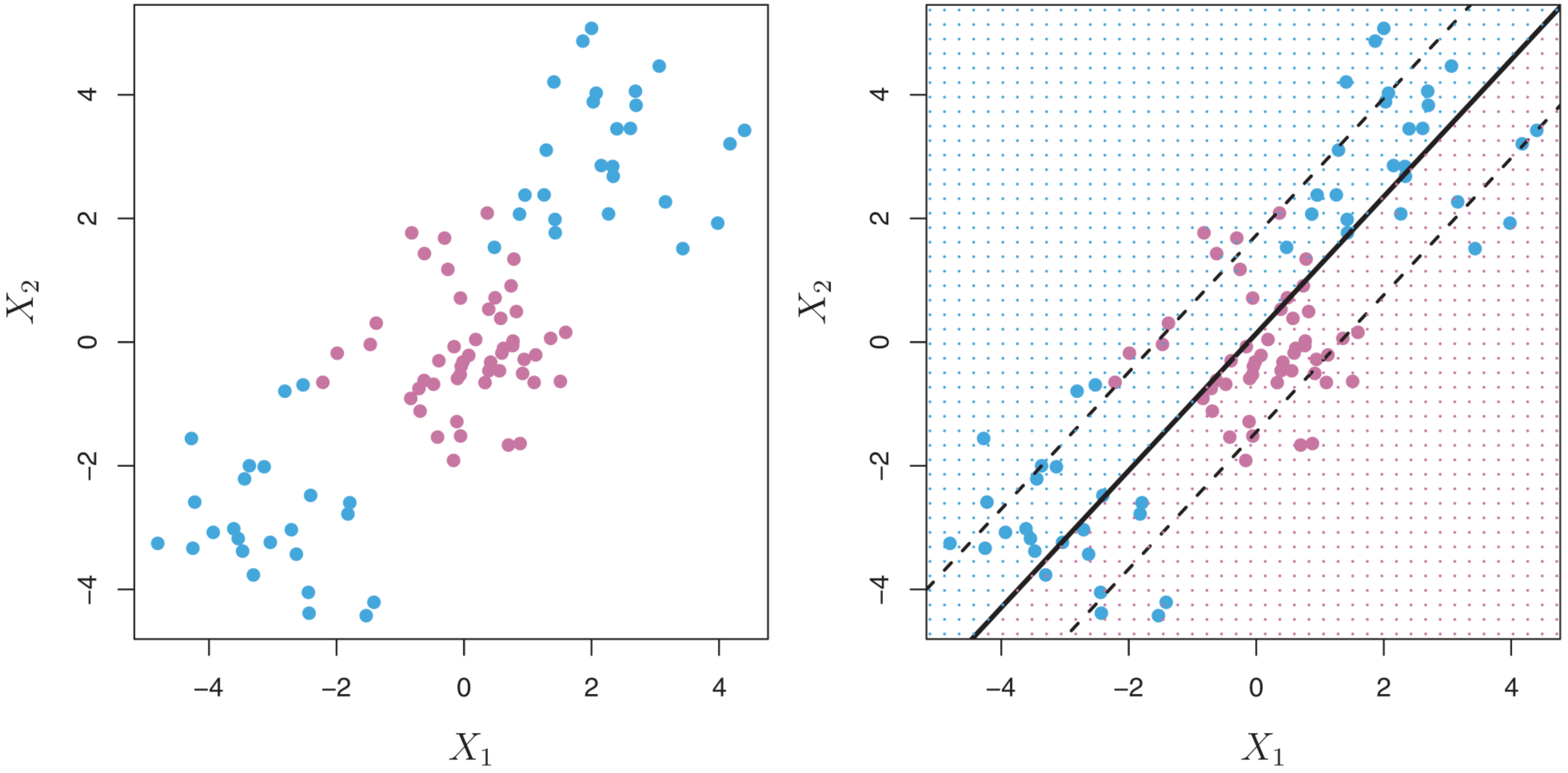
The support vector classifier finds a linear decision boundary in input feature space.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Application on non-linear decision boundaries

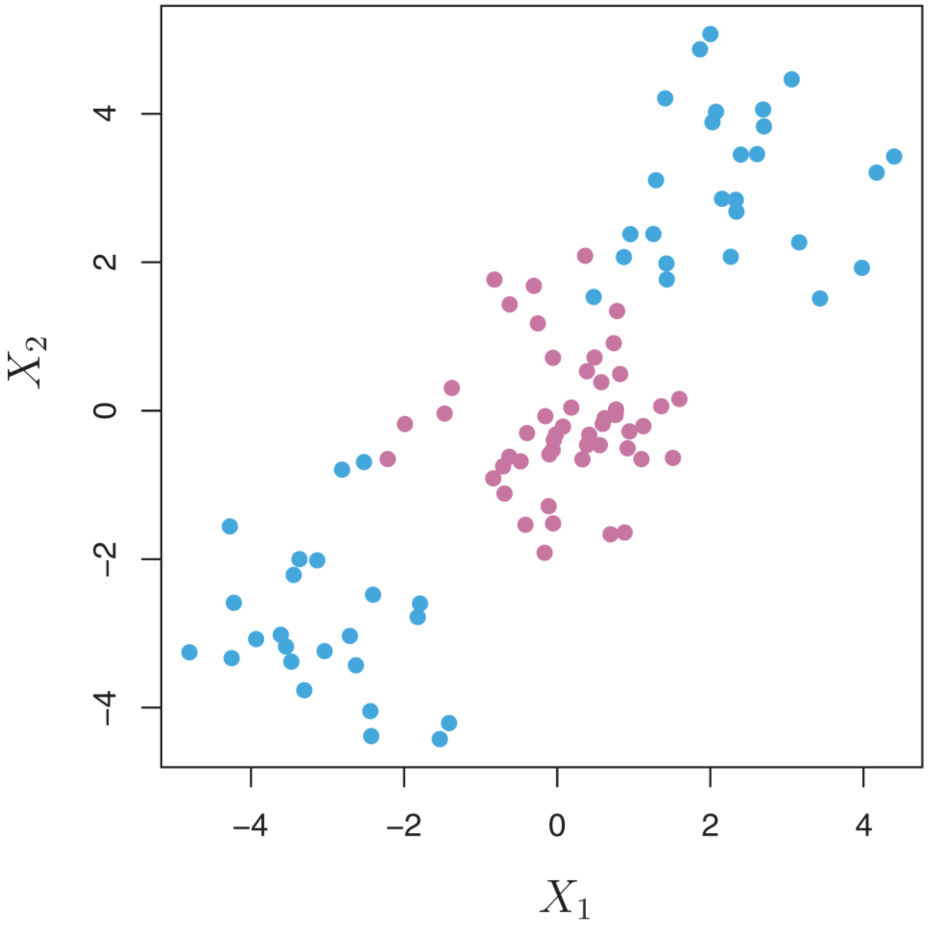
Consider a set of points in two-dimensional feature space X_1 and X_2 (left) for which no linear decision function can be used for classification.

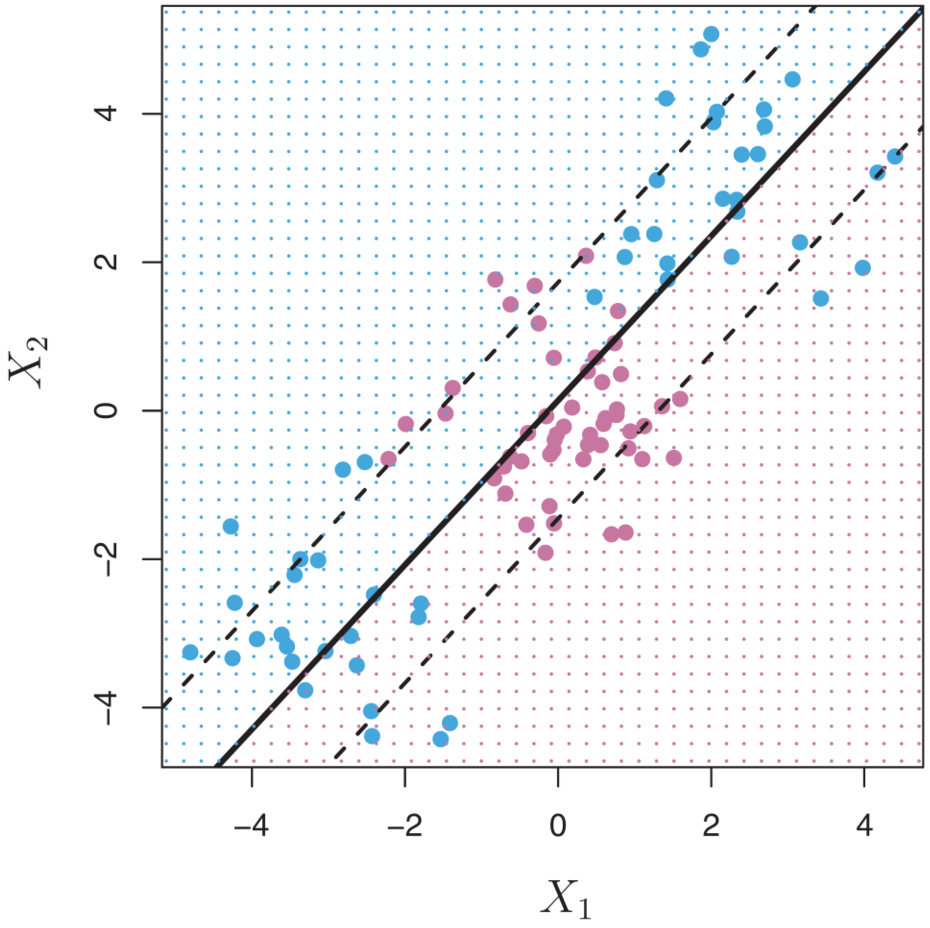


James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Application on non-linear decision boundaries

Applying the support vector classifier to these points (right) leads to a linear decision boundary that clearly cannot distinguish between the two classes.





James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Non-linear decision boundaries with basis functions

As with other approaches we have considered, we can make the support vector classifier more flexible by expanding the feature space.

Linear decision boundaries in the enlarged space often achieve better training class separation, and translate to non-linear boundaries in the input space.

We can expand the space using what are called **basis functions**.

Basis functions

A set of M basis functions, $h_m(x)$, $m = 1,2,\ldots,M$, can be selected prior to performing a machine learning procedure to alter the input feature space x into an expanded feature space

$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_M(x) \end{bmatrix}$$

We have already seen basis functions in action during prior lectures.

As an example, we have expanded the dimension of univariate inputs to polynomials of degree M with basis functions $h_m(x) = x^m$, such that univariate feature x is transformed to the M-dimensional space

$$h(x)^T = [x, x^2, \ldots, x^M]$$

Support vector machines (SVMs)

Within the support vector classifier framework, we can expand the dimension of each training input x_i using a basis expansion $h(x_i)$ to produce a (non-linear in feature space) decision function of the form

$$\hat{f}(X) = \hat{\beta}_0 + h(X)^T \hat{\beta}$$

and generate the classifier

$$\hat{Y}(X) = \mathrm{sign}\big[\hat{f}(X)\big] = \mathrm{sign}\big[\hat{\beta}_0 + h(X)^T \hat{\beta}\big]$$

The **support vector machine** (**SVM**) classifier is an extension of this idea, where dimension of feature space is allowed to get very large.

It would appear that these computations would be prohibitive, and that with sufficient basis functions, the data would be separable, and overfitting would occur, but these issues can be overcome.

Recall the problem of the support vector classifier

Recall the optimization problems of the support vector classifier

$$\mathcal{L} = \frac{1}{2}\|\beta\|_2^2 + \sum_{i=1}^{N} \alpha_i\left[1 - y_i\left(\beta_0 + x_i^T\beta\right)\right] - \sum_{i=1}^{N} (C - \alpha_i - \mu_i)\xi_i$$

$$\tilde{\mathcal{L}} = \sum_{i=1}^{N} \alpha_i - \frac{1}{2\lambda}\sum_{i=1}^{N}\sum_{k=1}^{N} \alpha_i\alpha_k y_i y_k x_i^T x_k$$

as well as the decision function

$$f(x) = \beta_0 + x^T\beta = \beta_0 + \sum_{i=1}^{N} \alpha_i y_i x^T x_i$$

In all these scenarios, we see inner products $\langle x_i, x_k\rangle = x_i^T x_k$ for feature vectors being computed.

Replace with inner products of basis functions

Replacing the inner products of features with inner products of basis expansions of features, we have

$$\tilde{\mathcal{L}} = \sum_{i=1}^{N} \alpha_i - \frac{1}{2\lambda} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \cdot \langle h(x_i), h(x_k) \rangle$$

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

where the computations to learn the decision function $f(x)$ are the same, except we have replaced the input feature vector x with an basis expansion $h(x)$.

We need not specify transformations $h(x)$ at all, but instead require knowledge only of the kernel function

$$K(x, x') = \langle h(x), h(x') \rangle$$

Replace with kernels

Replacing the inner products with kernels, we have

$$\tilde{\mathcal{L}} = \sum_{i=1}^{N} \alpha_i - \frac{1}{2\lambda} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \cdot K(x_i, x_k)$$

$$f(x) = \beta_0 + \sum_{i=1}^{N} \alpha_i y_i \cdot K(x, x_i)$$

where the computations to learn the decision function $f(x)$ are again the same, except we have replaced the inner products of basis expansions with the kernels of two input feature sets.

If we choose an appropriate kernel, then we can expand the input space in an automatic fashion, and the use of kernels instead of inner products in support vector classifiers turns them into what is known as **support vector machines** (**SVM**s) in the literature.

Example kernel functions

The **polynomial kernel of degree $\boldsymbol{d}$** has the form

$$\begin{aligned} K(x, x') &= (1 + \langle x, x' \rangle)^d \\ &= (1 + x^T x')^d \\ &= \left(1 + \sum_{j=1}^{p} x_j x_j'\right)^d \end{aligned}$$

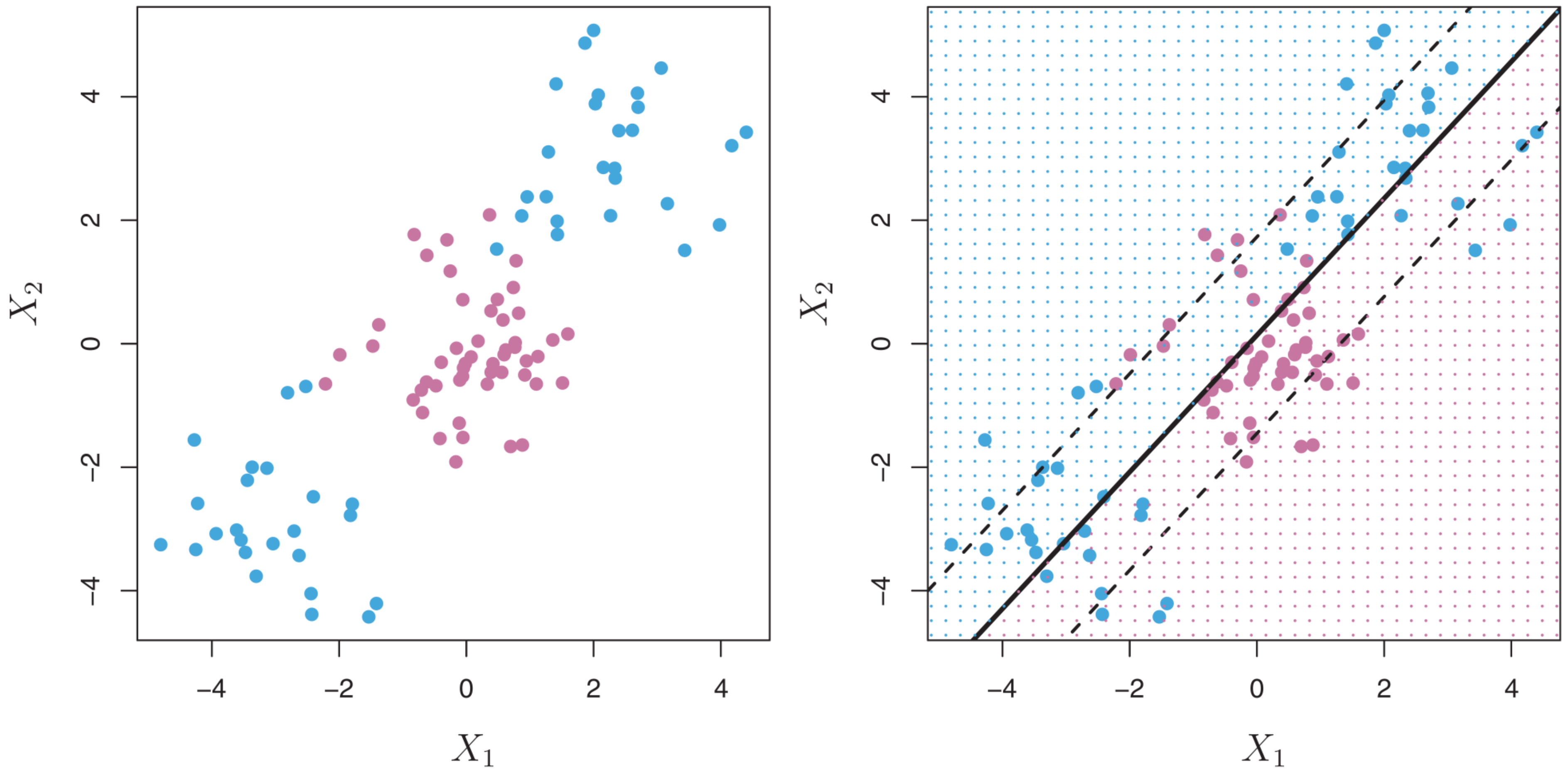
The **radial basis (Gaussian) kernel** has the form

$$\begin{aligned} K(x, x') &= \exp(-\gamma \|x - x'\|_2^2) \\ &= \exp\left(-\gamma \sum_{j=1}^{p} (x_j - x_j')^2\right) \end{aligned}$$

where the parameter γ controls the width of the peak.

Expanding dimension with kernels can help

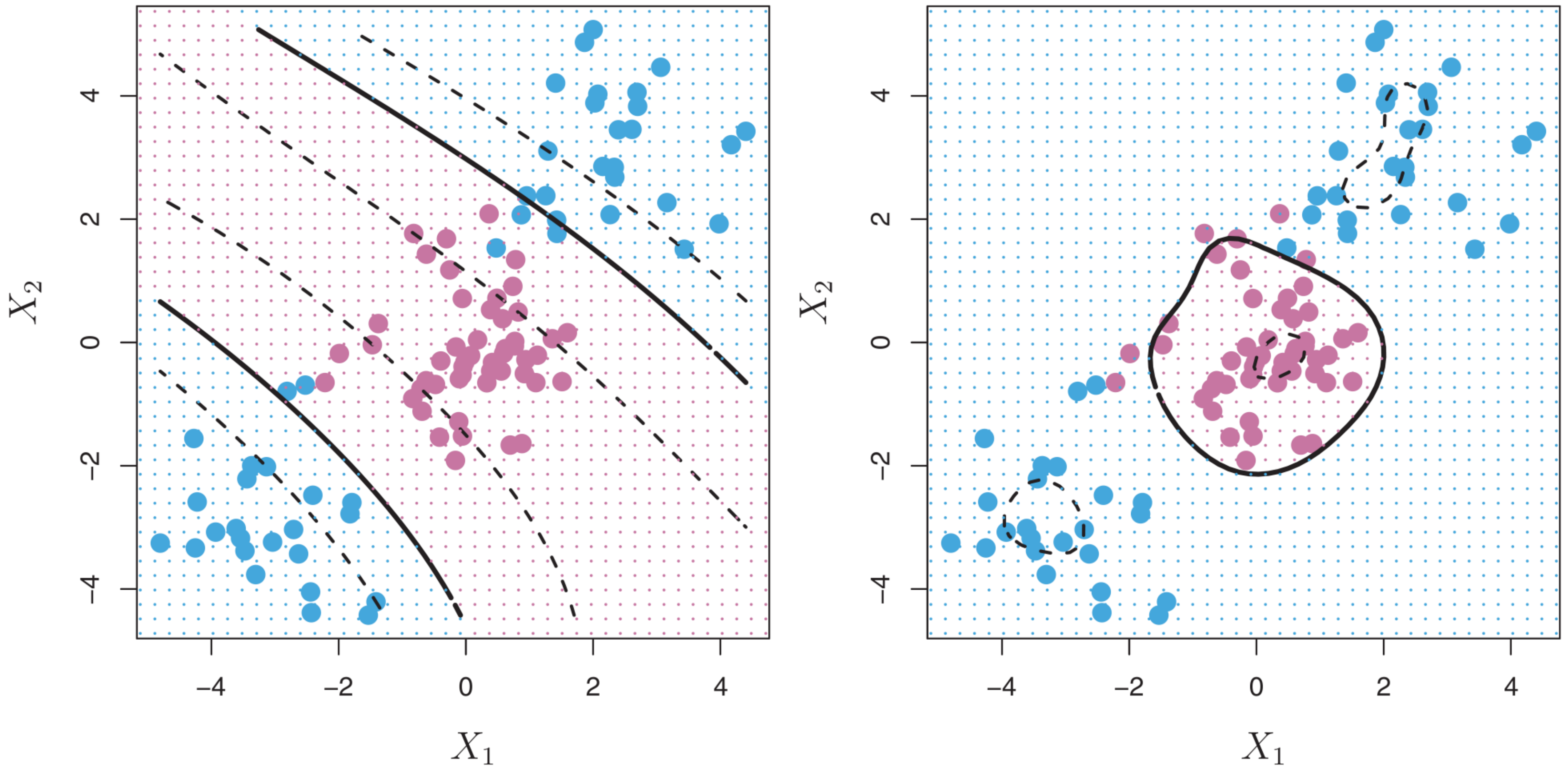
Consider again the following example for which the decision boundary is clearly non-linear.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Expanding dimension with kernels can help

Application of a polynomial kernel of degree 3 (left) and the radial basis kernel (right) demonstrate that correct classification can be achieved by expanding input dimension using kernels.



James *et al* (2017) *An Introduction to Statistical Learning: with Applications in R*. Springer.

Example relationship of kernel to basis expansion

Consider a feature space with $p = 2$ inputs X_1 and X_2 and a polynomial kernel of degree $d = 2$.

For 2 points X and X', the kernel is expanded as

$$
\begin{aligned}
K(X, X') &= (1 + \langle X, X' \rangle)^2 \\
&= (1 + X_1X_1' + X_2X_2')^2 \\
&= 1 + 2X_1X_1' + 2X_2X_2' + (X_1X_1')^2 + (X_2X_2')^2 + 2X_1X_1'X_2X_2'
\end{aligned}
$$

Note that this kernel has symmetry, between the two inputs, and that there is a limit to its flexibility (*e.g.*, some components are always multiplied by 2).

Example relationship of kernel to basis expansion

The kernel

$$K(X, X') = 1 + 2X_1X_1' + 2X_2X_2' + (X_1X_1')^2 + (X_2X_2')^2 + 2X_1X_1'X_2X_2'$$

can be written as

$$K(X, X') = \langle h(X), h(X') \rangle$$

where

$$h(X)^T = [h_1(X), h_2(X), h_3(X), h_4(X), h_5(X), h_6(X)]$$

and with $h_m(X)$, $m = 1,2, \ldots, 6$, defined as

$$\begin{aligned} h_1(X) &= 1 & h_4(X) &= X_1^2 \\ h_2(X) &= \sqrt{2}X_1 & h_5(X) &= X_2^2 \\ h_3(X) &= \sqrt{2}X_2 & h_6(X) &= \sqrt{2}X_1X_2 \end{aligned}$$