6 Support Vector Machines

> This section is to a large extend taken from Chapter 9 in G. James et al., *An Introduction to Statistical Learning: with Applications in R*, (accessible online) and Chapter 17.3 of K. P. Murphy, *Probabilistic Machine Learning - An Introduction* (accessible online) with adaptions in phrasing and notation.
> It aims at explaining the fundamental concepts of support vector machines, but does not provide the full mathematical details on the solution of the optimisation problem. The video lecture on the mathematical foundations by Patrick Winston is highly recommended as supplemental material.

6.1 Hyperplanes - Concept and Notation

The following discussion makes use of the term dimension and space. Note that these dimensions (denoted $x_1, x_2, x_3, \ldots$) correspond to **features** in the dataset at hand, that is columns in the matrix $\mathbf{X}$ introduced in Section 1.3. So the scatterplots of data samples $\{(x_1^{(1)}, x_2^{(1)}), (x_1^{(2)}, x_2^{(2)}), \ldots, (x_1^{(M)}, x_2^{(M)})\}$ imply, that the respective datasets have two features, i.e. $N = 2$.

In N-dimensional space, a hyperplane is a flat affine[1] subspace of **dimensions** $N - 1$. For instance, in two dimensions, a hyperplane is a one-dimensional subspace - a line, as can be seen in Figure 6.1. In three dimensions, a hyperplane is a flat two-dimensional subspace — a plane. In $N > 3$ dimensions, it can be hard to visualize a hyperplane, but the notion of a $(N - 1)$-dimensional flat subspace still applies.
The mathematical definition of a hyperplane is quite simple. In two dimensions, a hyperplane is defined by the equation

$$b + w_1 x_1 + w_2 x_2 = 0 \tag{6.1}$$

for parameters b, w_1,and w_2. Any $\mathbf{x} = \begin{pmatrix} x_1 & x_2 \end{pmatrix}^T$ for which Equation 6.1 holds is a point on the hyperplane (in this case a line, see Figure 6.1).
The formulation can be easily extended to the N-dimensional setting:

$$b + w_1 x_1 + w_2 x_2 + \cdots + w_N x_N = 0 \tag{6.2}$$

defines a N-dimensional hyperplane, again in the sense that if a point $\mathbf{x} = \begin{pmatrix} x_1 & x_2 & \ldots & x_N \end{pmatrix}^T$ in N-dimensional space (i.e. a vector of length N) satisfies Equation 6.3, then $\mathbf{x}$ lies on the hyperplane. Also the w's can be expressed as a N-dimensional parameter vector and the scalar (dot) product $\langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}^T \mathbf{x} = \sum_{n=1}^{N} w_n x_n$ allows to express Equation 6.3 in compact form:

$$b + \mathbf{w}^T \mathbf{x} = 0 \tag{6.3}$$

This "w, b" notation allows us to explicitly treat the intercept term b separately from the other parameters w_n. Also the convention used in linear and logistic regression of letting $x_0 = 1$ be an extra coordinate in the input feature vector is not used here. Thus, b takes the role of what was previously θ_0, and $\mathbf{w}$ takes the role of $(\theta_1 \ldots \theta_n)^T$.

Now, suppose that $\mathbf{x}$ does not satisfy Equation 6.3; rather,

$$b + \mathbf{w}^T \mathbf{x} > 0.$$

Then this tells us that $\mathbf{x}$ lies to one side of the hyperplane. On the other hand, if

$$b + \mathbf{w}^T \mathbf{x} < 0,$$

then $\mathbf{x}$ lies on the other side of the hyperplane. So we can think of the hyperplane as dividing N-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 6.3.

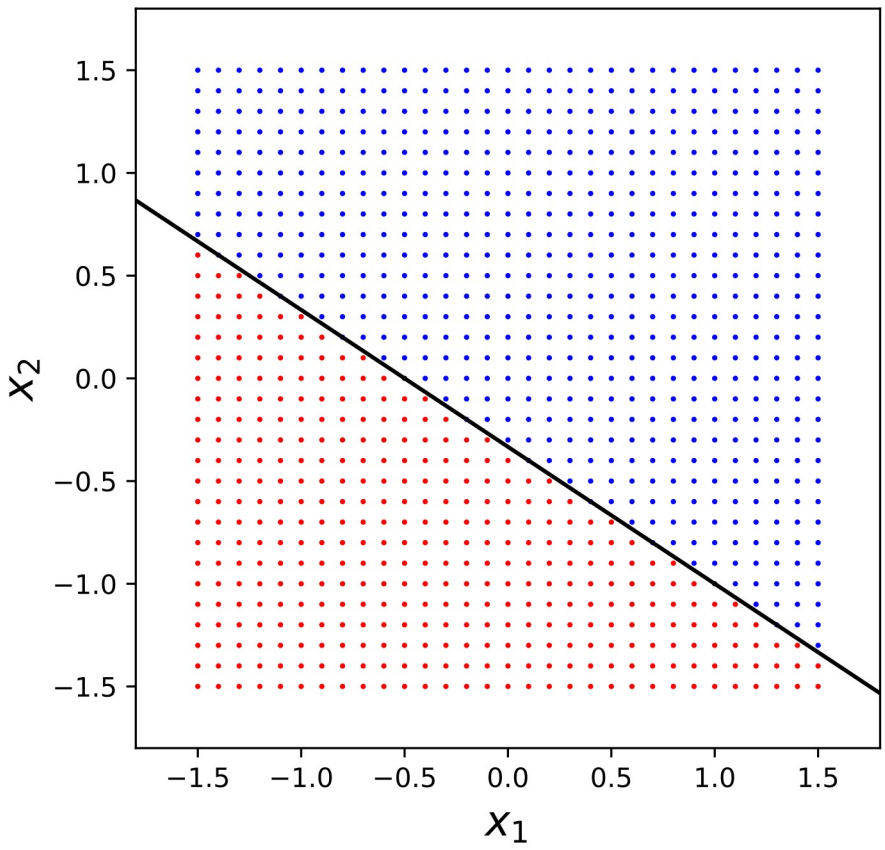


Figure 6.1: An example for a hyperplane in 2-dimensional feature space. That is, when our dataset comprises two features x_1 and x_2. The hyperplane $1 + 2x_1 + 3x_2 = 0$ is shown as a black solid line. The blue region is the set of points for which $1 + 2x_1 + 3x_2 > 0$ and the red region is the set of points for which $1 + 2x_1 + 3x_2 < 0$.

6.2 Classification Using a Separating Hyperplane

The binary classification setting involves M training samples in N-dimensional space,

$$\mathbf{x}^{(1)} = \begin{pmatrix} x_1^{(1)} \\ \vdots \\ x_N^{(1)} \end{pmatrix}, \ldots, \begin{pmatrix} x_1^{(M)} \\ \vdots \\ x_N^{(M)} \end{pmatrix},$$

with associated class labels $y^{(1)}, y^{(2)}, \ldots, y^{(M)} \in \{-1, +1\}$, where -1 represents one class and $+1$ the other class instead of $\{0, 1\}$, which makes some derivations more convenient.

A test sample is denoted as $\mathbf{x}^* = \begin{pmatrix} x_1^{(*)} & \ldots & x_N^{(*)} \end{pmatrix}^T$. The goal is to develop a classifier based on the training data that will correctly classify the test sample.

Suppose that it is possible to construct a hyperplane that separates the training samples perfectly according to their class labels. Examples of three such separating hyperplanes are shown in Figure 6.2 (a). The samples from the blue class can be labelled as $y^{(m)} = 1$ and those from the red class as $y^{(m)} = -1$. Then a separating hyperplane has the property that

$$b + \mathbf{w}^T \mathbf{x}^{(m)} > 0 \quad \text{if } y^{(m)} = 1, \tag{6.4}$$

and

$$b + \mathbf{w}^T \mathbf{x}^{(m)} < 0 \quad \text{if } y^{(m)} = -1 \tag{6.5}$$

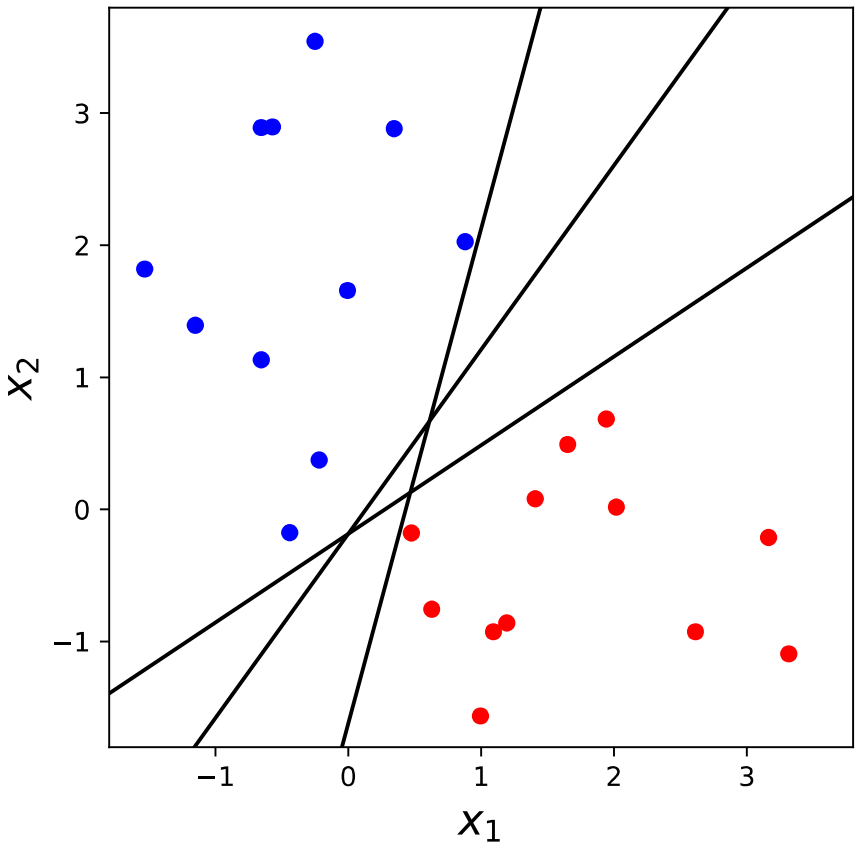
Equivalently, a separating hyperplane has the property that

$$y^{(m)} \left(b + \mathbf{w}^T \mathbf{x}^{(m)} \right) > 0 \tag{6.6}$$

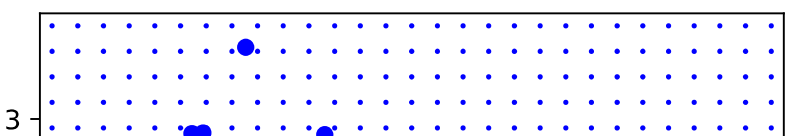
for all $m = 1, \ldots, M$.

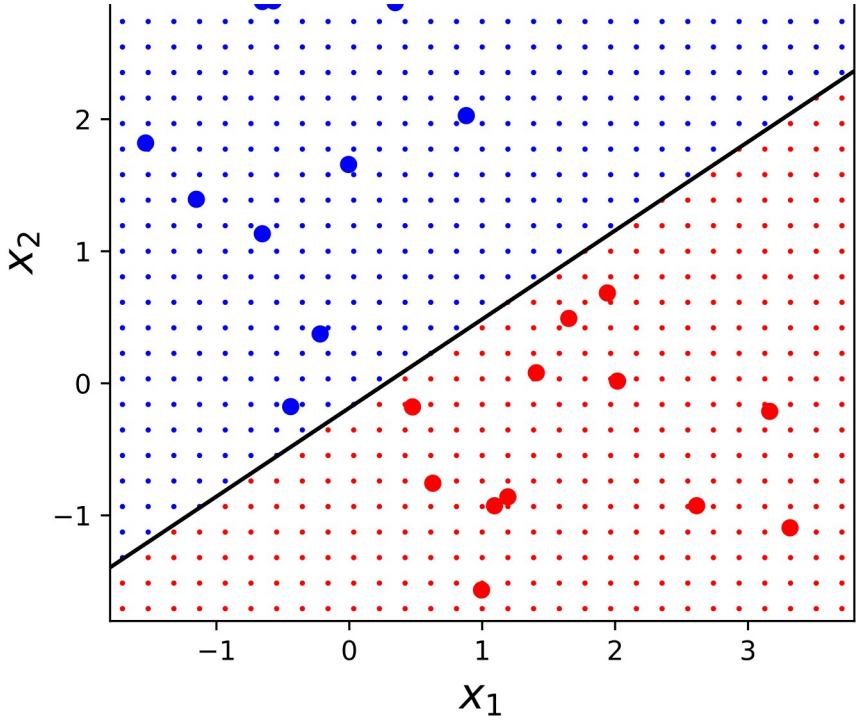
If a separating hyperplane exists, it can be used to construct a very natural classifier: a test sample is assigned a class depending on which side of the hyperplane it is located. Figure 6.2 (b) shows an example of such a classifier. That is, the test sample $\mathbf{x}^*$ is classified based on the sign of $f(\mathbf{x}^*) = b + \mathbf{w}^T \mathbf{x}^{(*)}$. If $f(\mathbf{x}^*)$ is positive, the test sample is assigned to class 1, and if $f(\mathbf{x}^*)$ is negative, then it is assigned to class -1. The magnitude of $f(\mathbf{x}^*)$ encodes the **confidence** with which the class label was assigned. If $f(\mathbf{x}^*)$ is far from zero, then this means that $\mathbf{x}^*$ lies far from the hyperplane, and so we can be confident about our class assignment for $\mathbf{x}^*$. On the other hand, if $f(\mathbf{x}^*)$ is close to zero, then $\mathbf{x}^*$ is located near the hyperplane, and so we are less certain about the class assignment for $\mathbf{x}^*$.

Naturally, a classifier that is based on a separating hyperplane leads to a linear **decision boundary** (Figure 6.2 (b)).



(a) Three separating hyperplanes, out of many possible, are shown in black.





(b) A separating hyperplane is shown in black. The blue and red grid indicates the decision rule made by a classifier based on this separating hyperplane: a test sample that falls in the blue portion of the grid will be assigned to the blue class, and a test sample that falls into the red portion of the grid will be assigned to the red class

Figure 6.2: A dataset consisting of two classes of samples, shown in blue and in red, and two variables (features). Hence a plot in 2 dimensions.

6.3 The Maximal Margin Classifier

In general, if the data can be perfectly separated using a hyperplane, then there will in fact exist an infinite number of such hyperplanes. This is because a given separating hyperplane can usually be shifted a tiny bit up or down, or rotated, without coming into contact with any of the samples. Three possible separating hyperplanes are shown in Figure 6.2 (a). In order to construct a classifier based upon a separating hyperplane we need a criterion to decide which of the infinite possible separating hyperplanes to use.
A natural choice is the maximal margin hyperplane (also known as the optimal separating hyperplane), which is the separating hyperplane that is farthest from the training samples. That is, we can compute the (perpendicular) distance from each training sample to a given separating hyperplane; the smallest such distance is the minimal distance from the samples to the hyperplane, and is known as the **margin**. The maximal margin hyperplane is the separating hyperplane for which the margin is largest—that is, it is the hyperplane that has the furthest minimum distance to the training samples. We can then classify a test sample based on which side of the maximal margin hyperplane it lies. This is known as the maximal margin classifier. We hope that a classifier that has a large margin on the training data will also have a large margin on the test data, and hence will classify the test samples correctly.

If $b, w_1, \ldots, w_N$ are the coefficients of the maximal margin hyperplane, then the maximal margin classifier classifies the test sample $\mathbf{x}^*$ based on the sign of $f(\mathbf{x}^*) = b + \mathbf{w}^T\mathbf{x}^{(*)}$.

Figure 6.3 shows the maximal margin hyperplane on the data set of Figure 6.2. Comparing Figure 6.2 (b) to Figure 6.3, we see that the maximal margin hyperplane shown in Figure 6.3 does indeed result in a greater

minimal distance between the samples and the separating hyperplane — that is, a larger margin. In a sense, the maximal margin hyperplane represents the mid-line of the widest street that can be drawn between the two classes. The two margin boundaries are therefore also called **gutters**.

Examining Figure 6.3, we see that three training samples are equidistant from the maximal margin hyperplane and lie along the dashed lines indicating the width of the margin - or the gutters of the street. These three samples are known as **support vectors**, since they are vectors in N-dimensional space (in Figure 6.3, $N = 2$) and 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. Interestingly, the maximal margin hyperplane depends directly on the support vectors, but not on the other samples: a movement to any of the other samples would not affect the separating hyperplane, provided that the sample's movement does not cause it to cross the boundary set by the margin. The fact that the maximal margin hyperplane depends directly on only a small subset of the samples is an important property that will arise later in the discussion around the support vector classifier (soft margin classifier) and support vector machine.

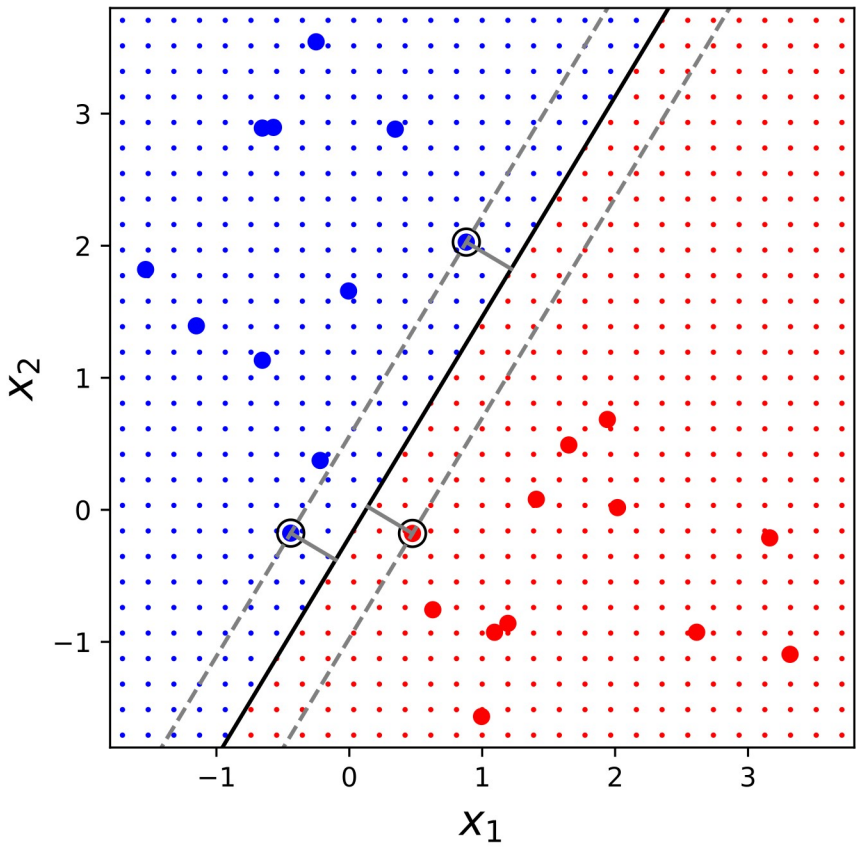
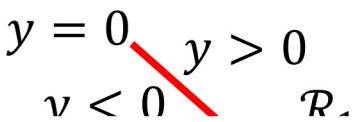


Figure 6.3: There are two classes of samples, shown in blue and in red. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the red point that lie on the dashed lines are the support vectors, and the distance from those points to the hyperplane is indicated by grey solid lines. The red and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.

6.4 Construction of the Maximal Margin Classifier



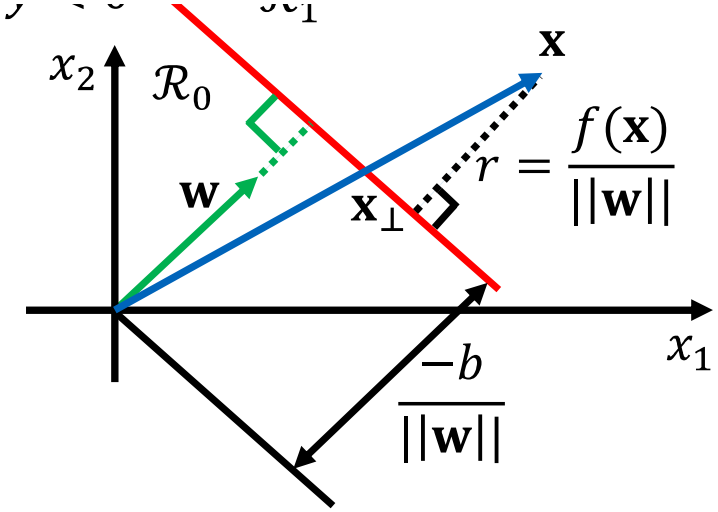


Figure 6.4: Illustration of the geometry of a linear decision boundary (red line) in 2 dimensions. A point $\mathbf{x}$ is classified belonging in decision region $\mathcal{R}_1$ if $f(\mathbf{x}) > 0$, otherwise it belongs in decision region $\mathcal{R}_0$; $\mathbf{w}$ is a vector which is perpendicular to the decision boundary. The term b controls the distance of the decision boundary from the origin. $\mathbf{x}_\perp$ is the orthogonal projection of $\mathbf{x}$ onto the boundary. The signed distance of $\mathbf{x}$ from the boundary is given by $r = \frac{f(\mathbf{x})}{||\mathbf{w}||}$.

We now consider the task of constructing the maximal margin hyperplane based on a set of M training samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)} \in \mathbb{R}^N$ and associated class labels $y^{(1)}, \ldots, y^{(M)} \in \{-1, 1\}$.

First we need to derive an expression for the distance of an arbitrary point $\mathbf{x}$ to the decision boundary. From Figure 6.4 we see that

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{||\mathbf{w}||}$$

where r is the distance of $\mathbf{x}$ from the decision boundary whose normal vector is $\mathbf{w}$, and $\mathbf{x}_\perp$ is the orthogonal projection of $\mathbf{x}$ onto this boundary.
We would like to maximise r, so we need to express it as a function of $\mathbf{w}$. First, note that

$$f(\mathbf{x}) = b + \mathbf{w}^T\mathbf{x} = (b + \mathbf{w}^T\mathbf{x}_\perp) + r\frac{\mathbf{w}^T\mathbf{w}}{||\mathbf{w}||} = (b + \mathbf{w}^T\mathbf{x}_\perp) + r||\mathbf{w}||$$

By construction, $\mathbf{x}_\perp$ lies on the decision boundary. Therefore, $(b + \mathbf{w}^T\mathbf{x}_\perp) = f(\mathbf{x}_\perp) = 0$. What remains is $f(\mathbf{x}) = r||\mathbf{w}||$ and hence

$$r = \frac{f(\mathbf{x})}{||\mathbf{w}||}. \tag{6.7}$$

We want to maximise r, but only for the training sample which is closest to the hyperplane (the margin), that is the one for which $f(\mathbf{x}^{(m)})$ evaluates to the minimal value. So the objective becomes

$$\max_{b,\mathbf{w}} \left\{ \frac{1}{||\mathbf{w}||} \min_{m=1}^{M} \left[y^{(m)} \left(b + \mathbf{w}^T\mathbf{x}^{(m)} \right) \right] \right\} \tag{6.8}$$

where we have taken the factor $\frac{1}{||\mathbf{w}||}$ outside the minimisation over m because $\mathbf{w}$ does not depend on m.
Furthermore, we want to ensure that each point is on the correct side of the boundary. We therefore add the constraint $y^{(m)} f(\mathbf{x}^{(m)}) > 0$ (as in Equation 6.4 - Equation 6.6).
Direct solution of this optimisation problem would be very complex, but it can be converted into an equivalent problem that is much easier to solve. To do this we note that by rescaling the parameters using

equivalent problem that is much easier to solve. To do this we note that by rescaling the parameters using $\mathbf{w} \rightarrow \kappa\mathbf{w}$ and $b \rightarrow \kappa b$, the distance from any point to the decision boundary, given by $y^{(m)}f(\mathbf{x}^{(m)})$, remains unchanged (the factor κ cancels out when we divide by $||\mathbf{w}||$). Therefore, we can define the scale factor κ such that $y^{(m)}f(\mathbf{x}^{(m)}) = 1$ for the point that is closest to the decision boundary, which means $\min_{m=1}^{M}\left[y^{(m)}\left(b + \mathbf{w}^T\mathbf{x}^{(m)}\right)\right] = 1$. The constraint becomes $y^{(m)}f(\mathbf{x}^{(m)}) \geq 1$ for all m.

Finally, note that maximising $\frac{1}{||\mathbf{w}||}$ is equivalent to minimising $||\mathbf{w}||^2$. Thus we get the new objective

$$\min_{b,\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 \tag{6.9}$$

$$\text{subject to } y^{(m)}\left(b + \mathbf{w}^T\mathbf{x}^{(m)}\right) \geq 1 \; \forall \; m = 1, \ldots, M \tag{6.10}$$

The factor $\frac{1}{2}$ is added for mathematical convenience and does not affect the optimal parameters. The constraint in Equation 6.10 says that we want all points to be on the correct side of the decision boundary with a margin of at least 1.

Equation 6.9, Equation 6.10 is the so called **primal representation** of the maximal margin problem. It is a quadratic programming problem which can be solved efficiently using Lagrange multipliers: The corresponding **dual representation** of the maximum margin problem

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{m=1}^{M}\alpha_m - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\alpha_i\alpha_j y^{(i)}y^{(j)}\mathbf{x}^{(i)^T}\mathbf{x}^{(j)} \tag{6.11}$$

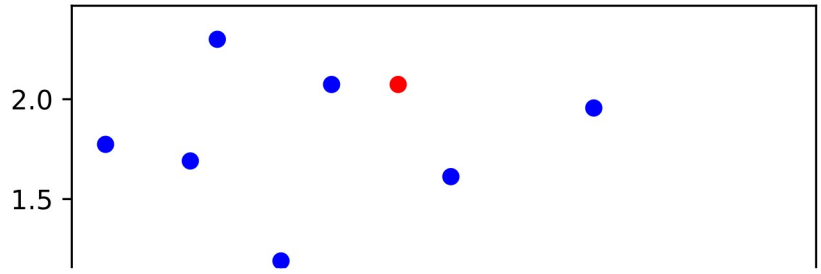
does not contain the parameters $\mathbf{w}$ and b anymore, but instead is maximised with respect to the Lagrange multipliers $\boldsymbol{\alpha} = (\alpha_1 \quad \alpha_2 \quad \ldots \quad \alpha_M)$, subject to the constraints that $\sum_{m=1}^{M}\alpha_m y^{(m)} = 0$ and $\alpha_m \geq 0 \; \forall \; m = 1, \ldots, M$. From the optimised $\hat{\boldsymbol{\alpha}}$ the optimal $\hat{\mathbf{w}}$ and $\hat{b}$ can be computed, which in turn describe the resulting maximal margin hyperplane.
Predictions can then be made using

$$f(\mathbf{x}) = \hat{b} + \hat{\mathbf{w}}^T\mathbf{x} = \hat{b} + \sum_{m\in\mathcal{S}}\hat{\alpha}_m y^{(m)}\mathbf{x}^{(m)^T}\mathbf{x} \tag{6.12}$$

which depends on the scalar products of only a subset of the training samples: The support vectors, denoted by $\mathcal{S}$.

Because the maximal margin classifier does not allow any violation of the separating hyperplane (each sample has to be on the correct side), it is also called **hard margin classifier** - as opposed to the **soft margin classifier**, which is discussed in the following section.

6.5 Support Vector Classifier (Soft Margin Classifier)



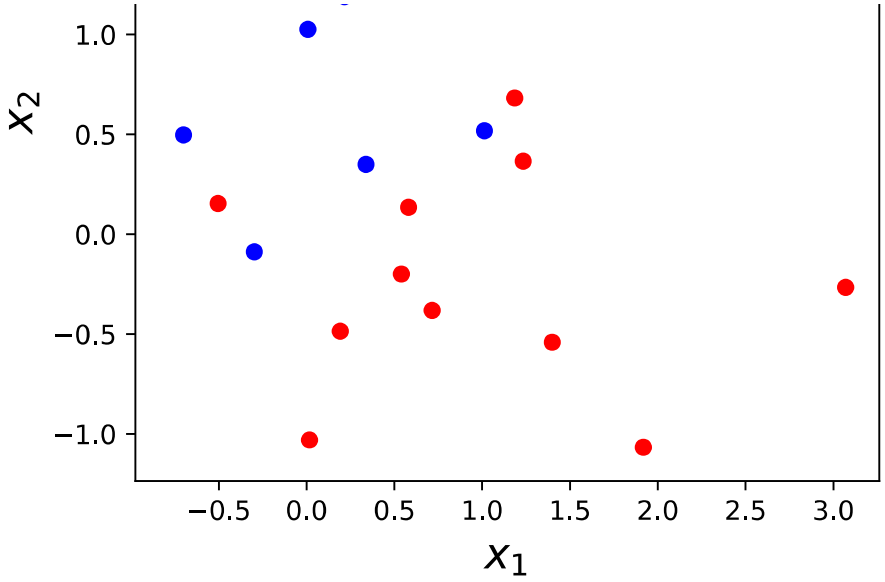
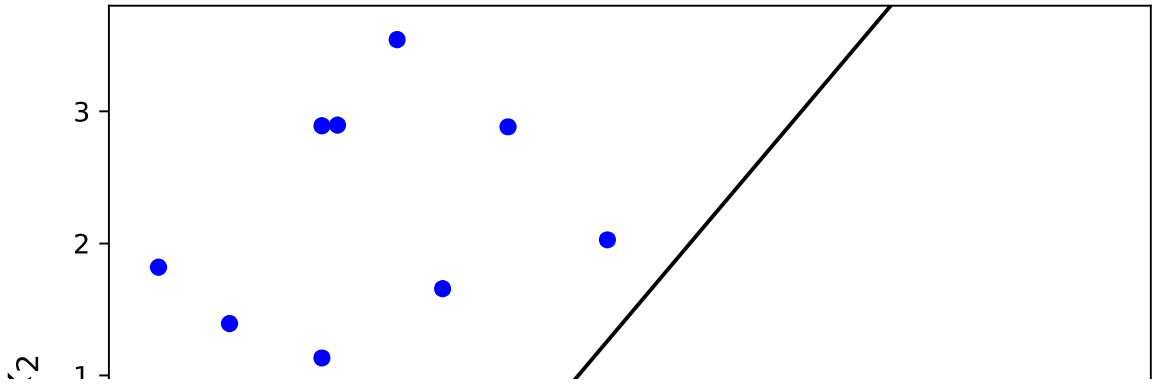


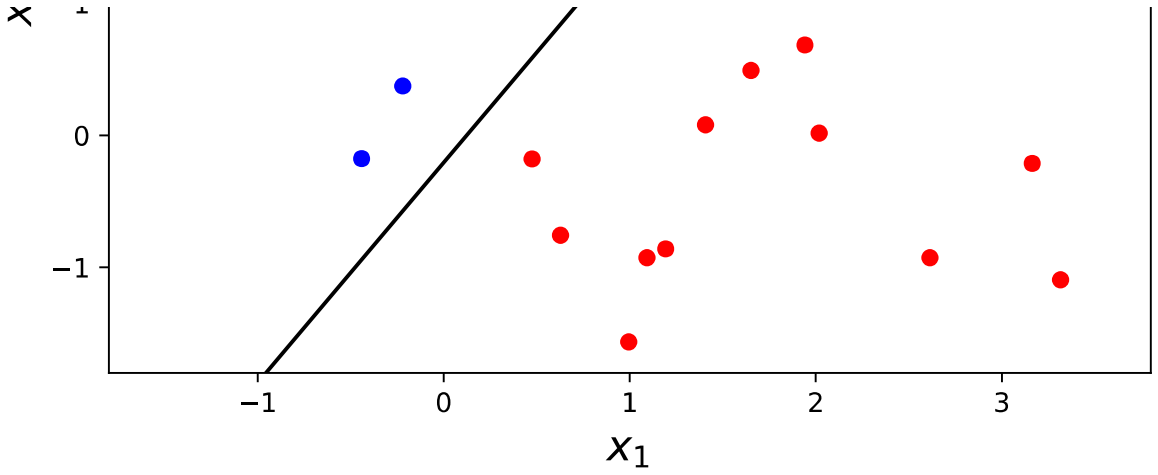
Figure 6.5: There are two classes of samples, shown in blue and in red. In this case, the two classes are not separable by a hyperplane, and so the maximal margin classifier cannot be used.

In many cases no separating hyperplane exists, that is, the optimization problem Equation 6.9 has no solution with a margin > 0 - and so there is no maximal margin classifier. An example is shown in Figure 6.5. In this case, the two classes cannot be separated exactly.
In fact, even if a separating hyperplane does exist, a hard margin classifier might not always be desirable. A classifier based on a separating hyperplane will necessarily perfectly classify all of the training samples; this can lead to sensitivity to individual samples. An example is shown in Figure 6.6. The addition of a single sample in Figure 6.6 (b) leads to a dramatic change in the maximal margin hyperplane. The resulting maximal margin hyperplane is not satisfactory - for one thing, it has only a tiny margin. This is problematic because the distance of an sample from the hyperplane can be seen as a measure of the confidence that the sample was correctly classified. Moreover, the fact that the maximal margin hyperplane is extremely sensitive to a change in a single sample suggests that it may have **overfit** the training data.
In this case, a classifier based on a hyperplane that does not perfectly separate the two classes might be a better choice in the interest of

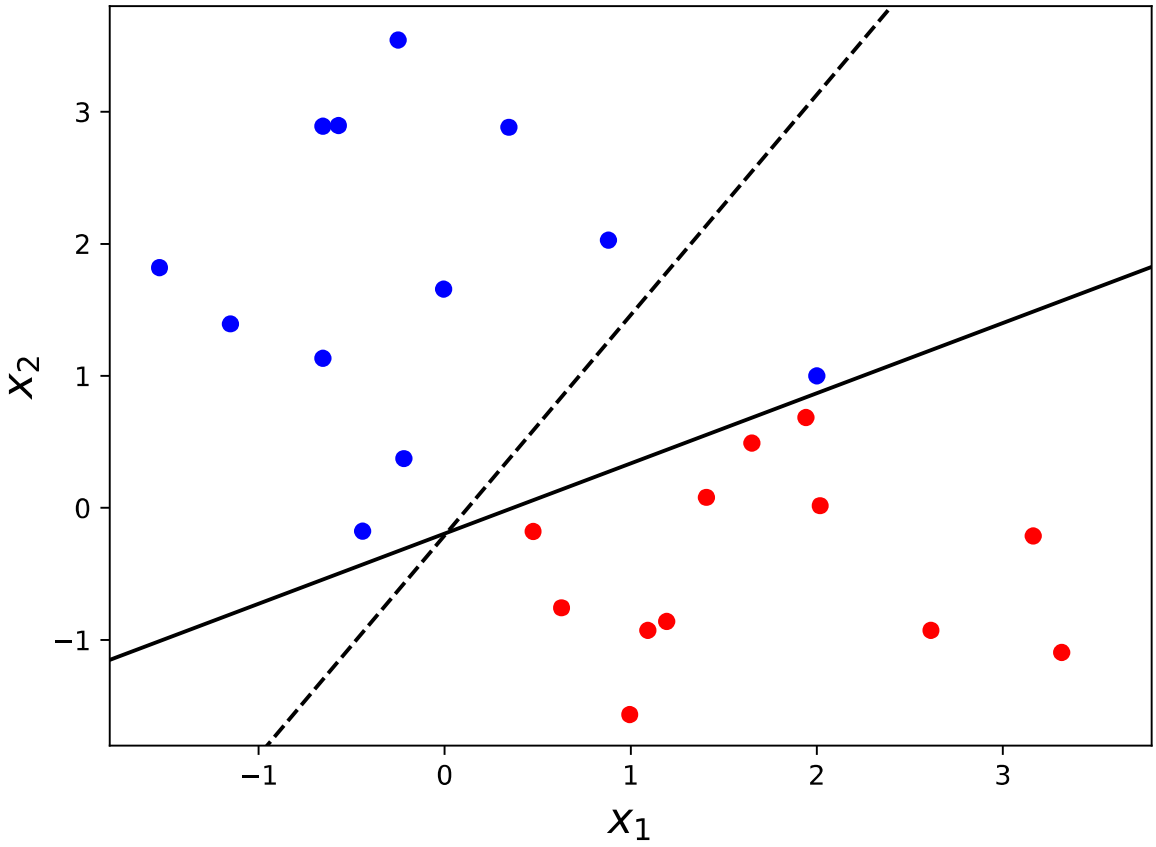
- greater robustness to individual samples, and
- better generalisability.

That is, it could be worthwhile to misclassify a few training samples in order to better classify the remaining samples. The **support vector classifier**, also called **soft margin classifier**, as an extension to the maximal margin classifier does exactly this. The margin is called soft because it is allowed to be violated by some of the training samples.





(a) A dataset consisting of two classes of samples, shown in blue and in red, and two variables (features), along with the maximal margin hyperplane as a solid line

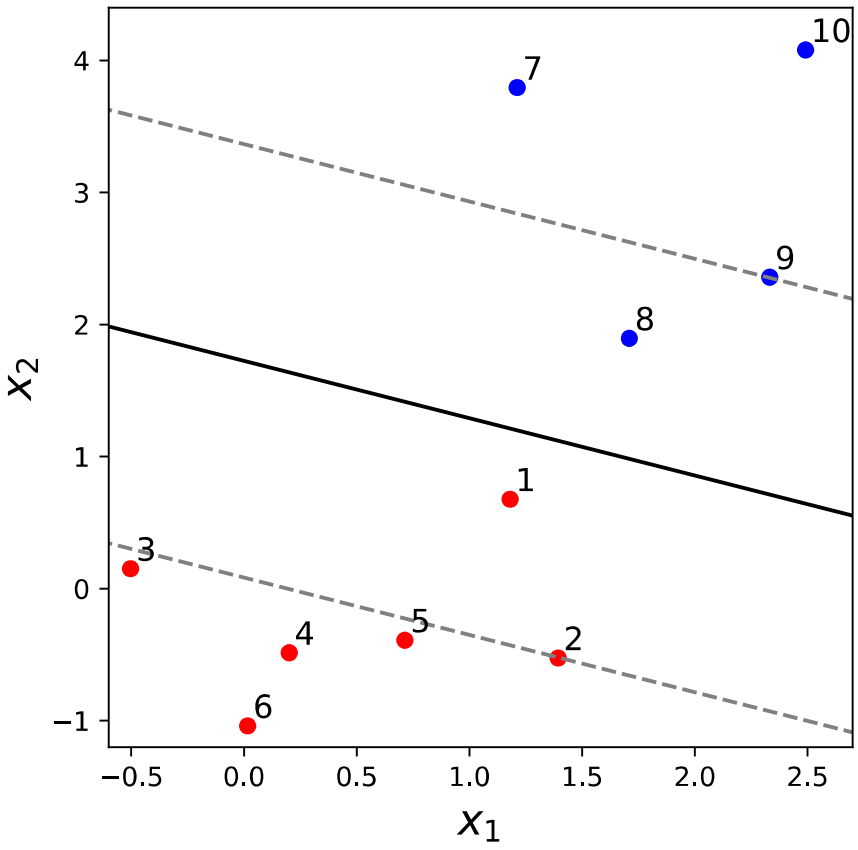


(b) An additional blue sample has been added, leading to a dramatic shift in the maximal margin hyperplane shown as a solid line. The dashed line indicates the maximal margin hyperplane that was obtained in the absence of this additional point.

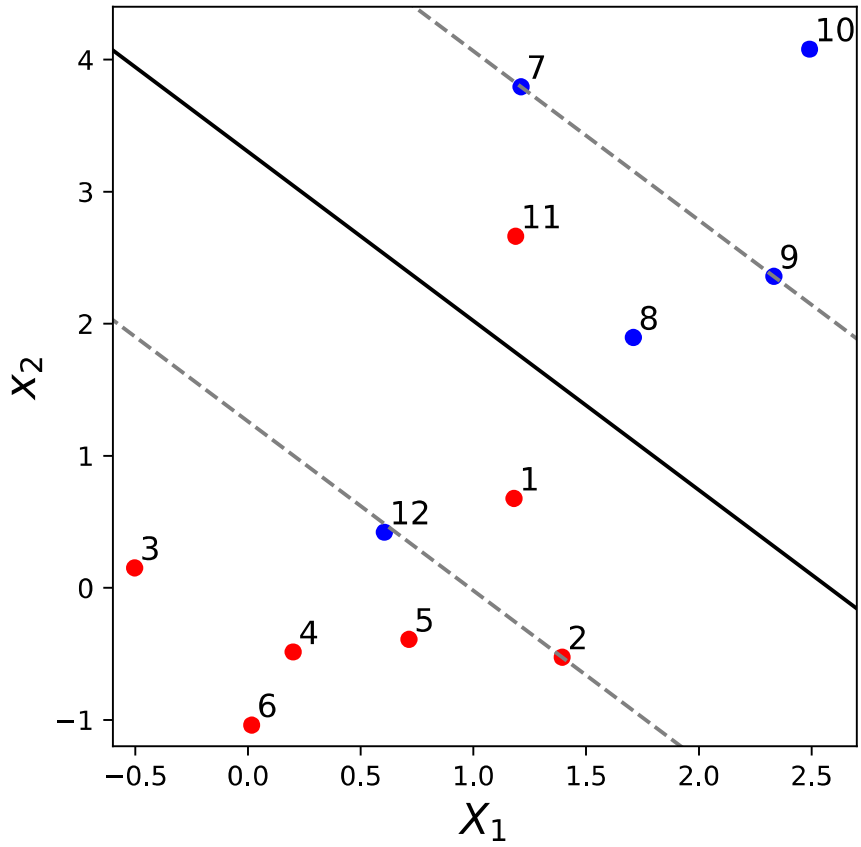
Figure 6.6: Illustration of the sensitivity of the maximal margin to individual samples.

The soft margin classifier does not aim for the largest possible margin so that every sample is not only on the correct side of the hyperplane but also on the correct side of the margin, but allows some samples to be on the incorrect side of the margin, or even on the incorrect side of the hyperplane. An example is shown in Figure 6.7 (a). Most of the samples are on the correct side of the margin. However, a small subset of the samples are on the wrong side of the margin. An sample can be not only on the wrong side of the margin, but also on the wrong side of the hyperplane. In fact, when there is no separating hyperplane, such a

situation is inevitable. Samples on the wrong side of the hyperplane correspond to training samples that are misclassified by the support vector classifier. Figure 6.7 (b) illustrates such a scenario.



(a) Red samples 3, 4, 5 and 6 are on the correct side of the margin, sample 2 is on the margin, and sample 1 is on the wrong side of the margin. Blue samples 7 and 10 are on the correct side of the margin, sample 9 is on the margin and sample 8 is on the wrong side of the margin. No samples are on the wrong side of the hyperplane.



(b) Two additional points, 11 and 12, have been added. These two samples are on the wrong side of the hyperplane and the wrong side of the margin.

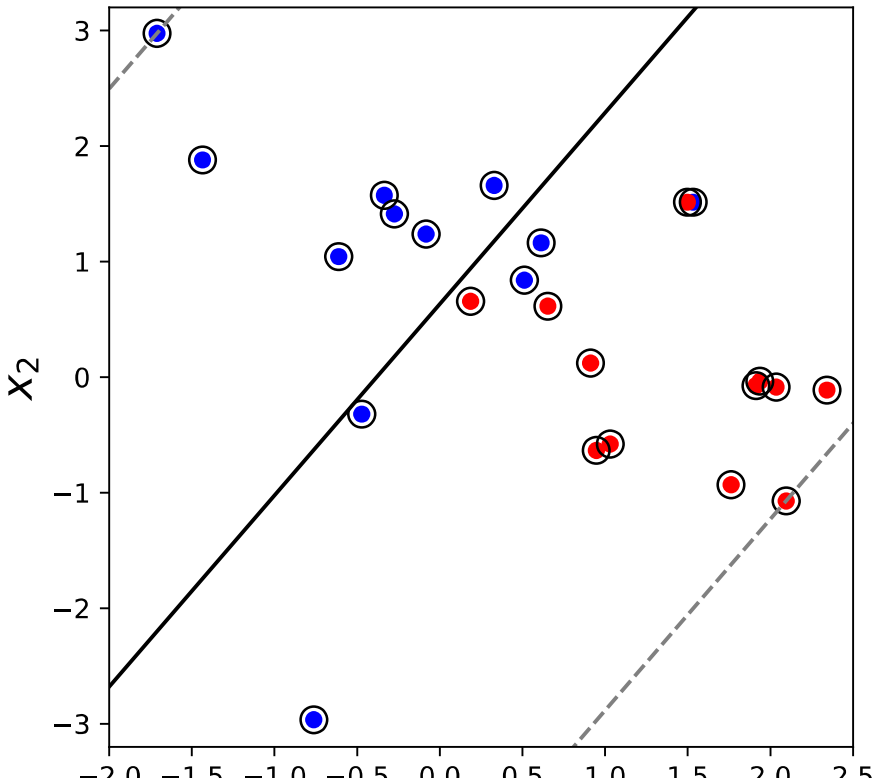
Figure 6.7: A soft margin support vector machine with a linear kernel was fit to a small dataset of two classes red and blue. The hyperplane is shown as a solid line and the margins are shown as dashed lines.

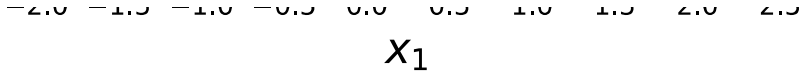
For the support vector classifier we introduce **slack variables** $\epsilon = \begin{pmatrix}\epsilon_1 & \epsilon_2 & \dots & \epsilon_M\end{pmatrix}^T, \epsilon_m \geq 0$ that allow a few individual samples to be on the wrong side of the margin or the hyperplane. The hard constraint that $y^{(m)} f(\mathbf{x}^{(m)}) \geq 0$ is replaced by the soft margin constraint that $y^{(m)} f(\mathbf{x}^{(m)}) \geq 1 - \epsilon_m$. The new objective becomes

$$\min_{b,\mathbf{w},\epsilon} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{m=1}^{M}\epsilon_m \tag{6.13}$$

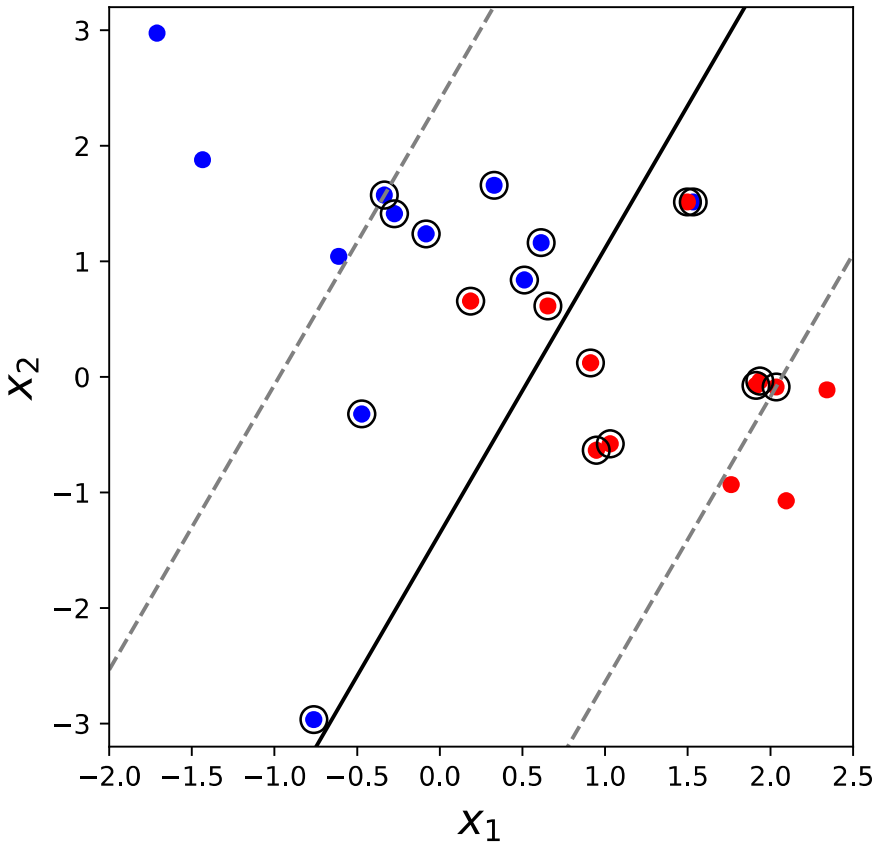
$$\text{subject to } \epsilon_m \geq 0,\ y^{(m)}\left(b + \mathbf{w}^T\mathbf{x}^{(m)}\right) \geq 1 - \epsilon_m \tag{6.14}$$

where C is a nonnegative tuning parameter (regularisation) controlling how many points are allowed to violate the margin constraint. A small C allows more violations, while a $C = \infty$ reduces to the unregularised, hard-margin classifier -of course, a maximal margin hyperplane exists only if the two classes are separable. An example is shown in Figure 6.8.
As in the case of the hard margin classifier, Equation 6.13, Equation 6.14 can be recast in a corresponding dual form using the Lagrangian and solved via quadratic optimisation techniques. Once the optimised parameters $\mathbf{w}$ and b have been found, predictions can be made, as before, by $f(\mathbf{x}; \hat{b}, \hat{\mathbf{w}}) = \hat{b} + \hat{\mathbf{w}}^T\mathbf{x}$.
The slack variable ϵ_m informs where the m-th sample is located, relative to the hyperplane and relative to the margin: If $\epsilon_m = 0$ then the m-th sample is on the correct side of the margin. If $\epsilon_m > 0$ then the mth sample is on the wrong side of the margin (the m-th sample has violated the margin). If $\epsilon_m > 1$ then it is on the wrong side of the hyperplane.

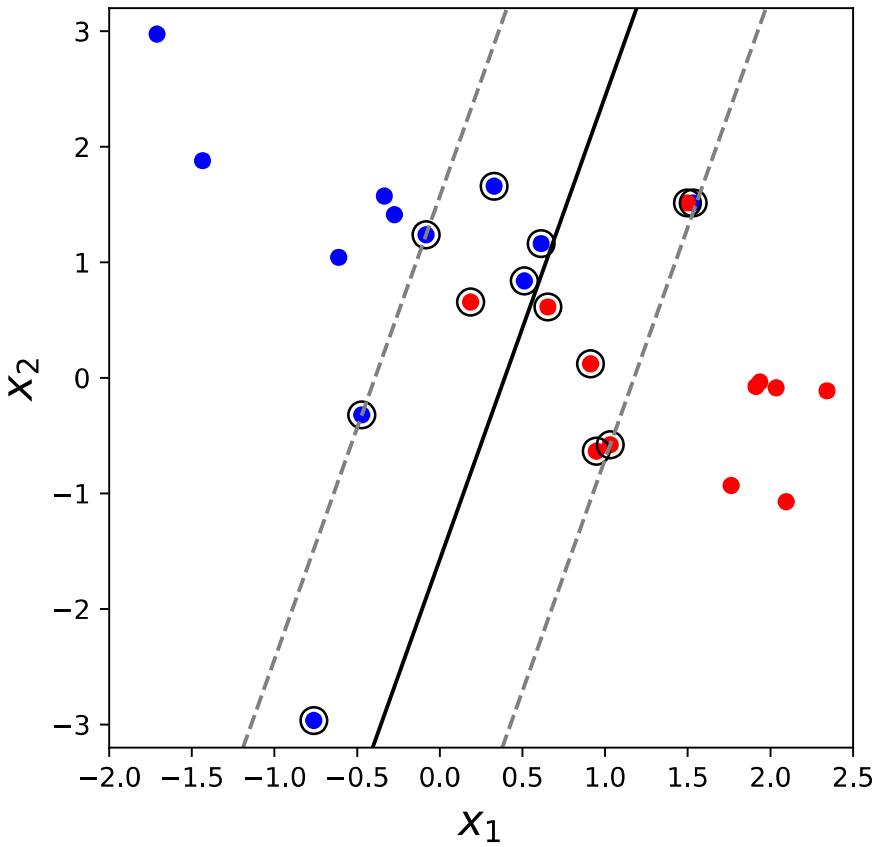



x_1

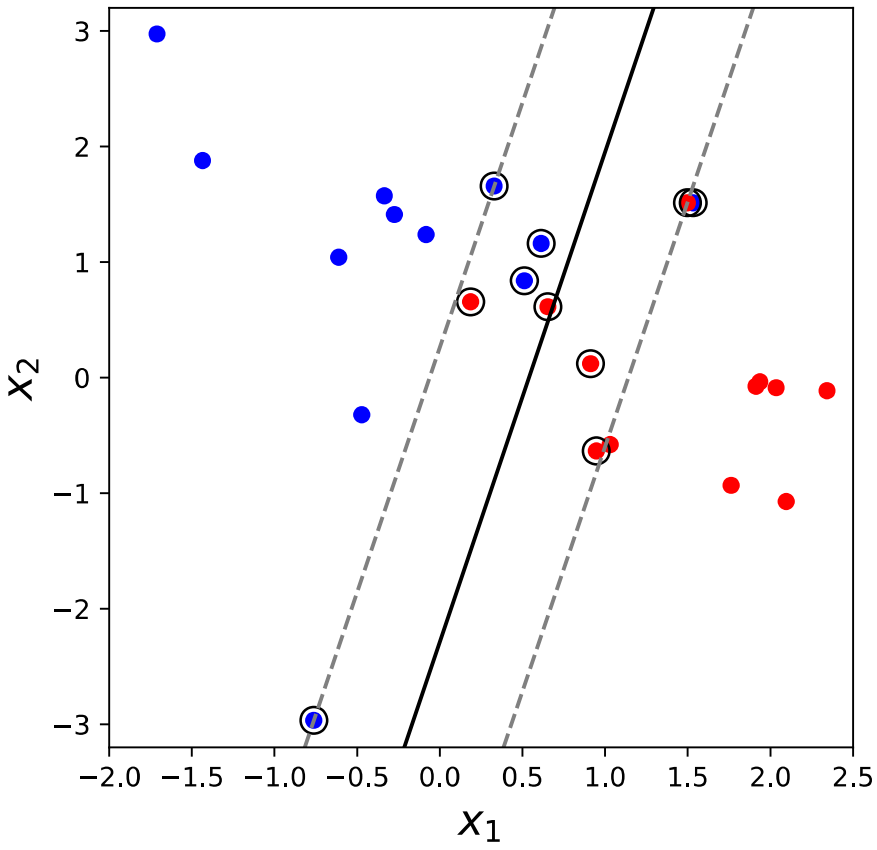
(a)


3
2
1
0
−1
−2
−3
x_2
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5
x_1

(b)


3
2
1
0
−1
−2
−3
x_2
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5
x_1

(c)



(d)

Figure 6.8: Illustration of the sensitivity of the hyperplane and margin to the hyperparameter C in Equation 6.14. The smallest value was used in *(a)*, and larger values were used in *(b)*, *(c)* and *(d)*. When C is small, then there is a high tolerance for samples being on the wrong side of the margin, and so the margin will be large. As C increases, the tolerance for samples being on the wrong side of the margin decreases, and the margin narrows. Support vectors are indicated with a black outer circle.

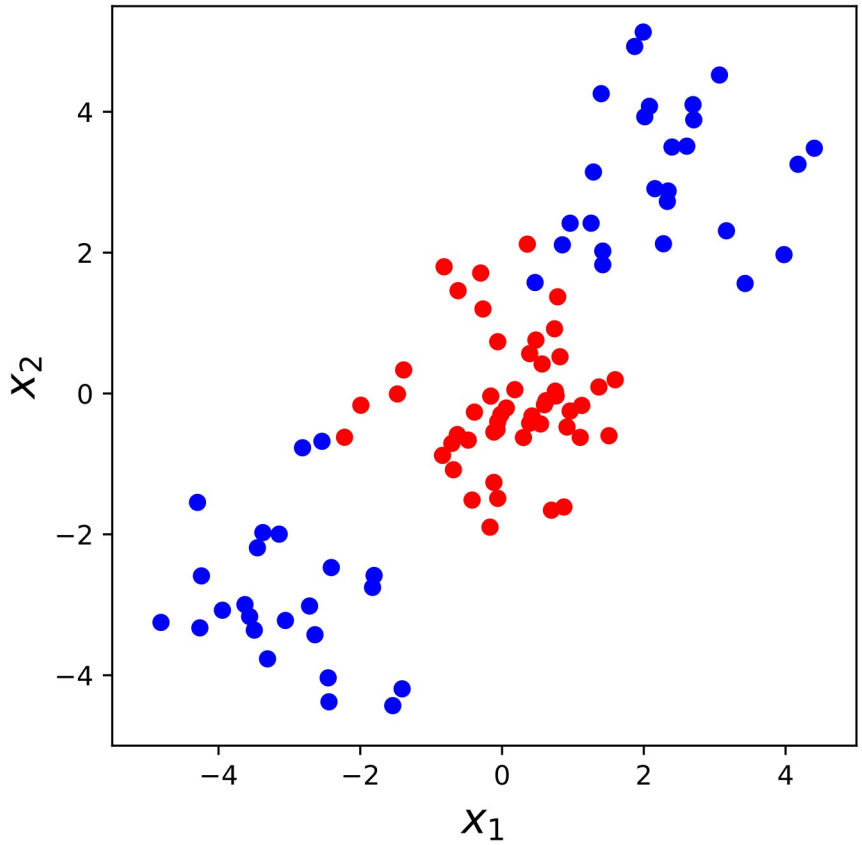
In practice, C is treated as a tunable hyperparameter that is generally chosen via cross-validation. As other hyperparameters, C controls the **bias-variance trade-off** of the model. When C is large, narrow margins are prefered that are rarely violated; this amounts to a classifier that is highly fit to the data, which may have low bias but high variance. On the other hand, when C is small, the margin is wider and more violations are allowed; this amounts to fitting the data less hard and obtaining a classifier that is potentially more biased but may have lower variance.

The optimization problem Equation 6.13, Equation 6.14 has a very interesting property: it turns out that only samples that either lie on the margin or that violate the margin will affect the hyperplane, and hence the classifier obtained. In other words, an sample that lies strictly on the correct side of the margin does not affect the support vector classifier! Changing the position of that sample would not change the classifier at all, provided that its position remains on the correct side of the margin. Samples that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors. These samples do affect the support vector classifier.

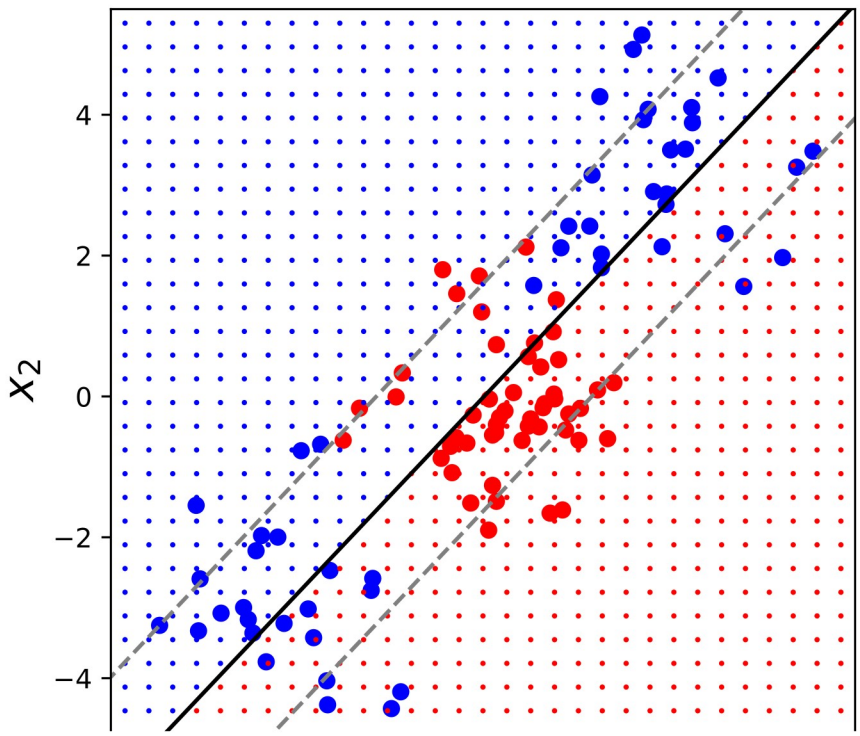
The fact that only support vectors affect the classifier is in line with our previous assertion that C controls the bias-variance trade-off of the support vector classifier. When the tuning parameter C is small, then the margin is wide, many samples may violate the margin, and so there are many support vectors. In this case,

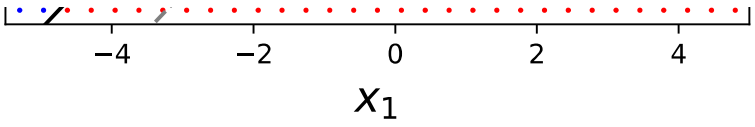
many samples are involved in determining the hyperplane. The Figure 6.8 (a) illustrates this setting: this classifier has low variance (essentially all samples are support vectors) but potentially high bias. In contrast, if C is large, then there will be fewer support vectors and hence the resulting classifier will have low bias but high variance. Figure 6.8 (d) illustrates this setting, with only 10 support vectors.

6.6 Support Vector Machines and the Kernel-Trick



(a) A classification dataset consisting of blue and red samples with a non-linear boundary between the two classes.





(b) The soft margin classifier seeks a linear boundary, and consequently performs very poorly.

Figure 6.9: The linear decision boundary learned from a soft margin support vector machine on a non-linearly separable dataset

Many classification problems are not linearly separable, such as the data in Figure 6.9 (a). It is clear that a support vector classifier or any linear classifier will perform poorly here. Indeed, the support vector classifier shown in Figure 6.9 (b) useless here.
The problem could be addressed by enlarging the feature space using quadratic, cubic, and even higher-order polynomial functions of the predictors (x_1 and x_2). For instance, rather than fitting a support vector classifier using N features

$$x_1, x_2, \ldots, x_N$$

a support vector classifier could be fit using $2N$ features

$$x_1, x_1^2, x_2, x_2^2, \ldots, x_N, x_N^2$$

Then we would need two parameter vectors $\mathbf{w}_1 = \begin{pmatrix} w_{11} & w_{21} & \cdots & w_{N1} \end{pmatrix}^T$, $\mathbf{w}_2 = \begin{pmatrix} w_{12} & w_{22} & \cdots & w_{N2} \end{pmatrix}^T$ and Equation 6.13,Equation 6.14 would become

$$\min_{b, \mathbf{w}_1, \mathbf{w}_2, \epsilon} \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{m=1}^{M} \epsilon_m \tag{6.15}$$

$$\text{subject to } \epsilon_m \geq 0,\ y^{(m)} \left(b + \sum_{n=1}^{N} w_{n1}^T x_n^{(m)} \sum_{n=1}^{N} w_{n2}^T x_n^{(m)} \right) \geq 1 - \epsilon_m$$

Why does this lead to a non-linear decision boundary? In the enlarged feature space, the decision boundary that results from Equation 6.15 is in fact linear. But in the original feature space, the decision boundary is of the form $q(x) = 0$, where q is a quadratic polynomial, and its solutions are generally non-linear. One might additionally want to enlarge the feature space with higher-order polynomial terms, or with interaction terms of the form $x_n x_{n'}$ for $n \neq n'$. Alternatively, other functions of the predictors could be considered rather than polynomials. It is not hard to see that there are many possible ways to enlarge the feature space, and that unless we do not have knowledge about which specific features are useful on the given dataset , we could end up with a huge number of features and computations would become unmanageable. The support vector machine, allows to enlarge the feature space used by the support vector classifier in a way that leads to efficient computations.

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, that allow for efficient computation.
We have not discussed exactly how the support vector classifier is computed. However, it turns out that the solution to the support vector classifier problem Equation 6.13, Equation 6.14 involves only the inner products of the samples (as opposed to the samples themselves)

$$\langle \mathbf{x}^{(m)}, \mathbf{x}^{(m')} \rangle = \sum^{N} x_n^{(m)} x_n^{(m')} \tag{6.16}$$

$$\overbrace{\quad\quad\quad\quad\quad\quad\quad\quad}^{n=1}$$

It can be shown that

- the linear support vector classifier can be represented as

$$f(\mathbf{x}) = b + \sum_{m=1}^{M} \alpha_m y^{(m)} \langle \mathbf{x}, \mathbf{x}^{(m)} \rangle \tag{6.17}$$

  where there are m parameters $\alpha_m, m = 1, \ldots, m$, one per training sample.
- to estimate the parameters $\alpha_1, \ldots, \alpha_m$ and b, all we need are the $\binom{m}{2}$ inner products $\langle \mathbf{x}^{(m)}, \mathbf{x}^{(m')} \rangle$ between all unique pairs of training samples. (The notation $\binom{m}{2}$ means $m(m-1)/2$, and gives the number of pairs among a set of m items.)

Notice that in Equation 6.17, in order to evaluate the function $f(\mathbf{x})$, we need to compute the inner product between the new point $\mathbf{x}$ and each of the training points $\mathbf{x}^{(m)}$. However, it turns out that α_m is nonzero only for the support vectors in the solution - that is, if a training sample is not a support vector, then its α_m equals zero. So if $\mathcal{S}$ is the collection of indices of these support points, we can rewrite any solution function of the form Equation 6.17 as

$$f(\mathbf{x}) = b + \sum_{m \in \mathcal{S}} \alpha_m y^{(m)} \langle \mathbf{x}, \mathbf{x}^{(m)} \rangle \tag{6.18}$$

which typically involves far fewer terms than in Equation 6.17.[2]
To summarize, in representing the linear classifier $f(\mathbf{x})$, and in computing its coefficients, all we need are inner products. Now suppose that every time the inner product Equation 6.16 appears in the representation Equation 6.17, or in a calculation of the solution for the support vector classifier, we replace it with a generalization of the inner product of the form

$$\mathcal{K}(\mathbf{x}^{(m)}, \mathbf{x}^{(m')}), \tag{6.19}$$

where $\mathcal{K}$ is a function called a **kernel**, which quantifies the similarity of two samples. For instance

$$\mathcal{K}(\mathbf{x}^{(m)}, \mathbf{x}^{(m')}) = \sum_{n=1}^{N} x_n^{(m)} x_n^{(m')} \tag{6.20}$$

would just give us back the support vector classifier. Equation Equation 6.20 is known as a linear kernel because the support vector classifier is linear in the features. But one could instead choose another form for Equation 6.19. For example, every instance of $\sum_{n=1}^{N} x_n^{(m)} x_n^{(m')}$ could be replaced with the quantity

$$\mathcal{K}(\mathbf{x}^{(m)}, \mathbf{x}^{(m')}) = \left(1 + \sum_{n=1}^{N} x_n^{(m)} x_n^{(m')}\right)^d. \tag{6.21}$$

This is known as a **polynomial kernel** of degree d,where d is a positive integer. Using such a kernel with $d > 1$, instead of the standard linear kernel Equation 6.20, in the support vector classifier algorithm leads to a much more flexible decision boundary. It essentially amounts to fitting a support vector classifier in a higher-dimensional space involving polynomials of degree d, rather than in the original feature space.
When the support vector classifier is combined with a non-linear kernel such as Equation 6.21, the resulting classifier is known as a support vector machine. Note that in this case the (non-linear) function has the form

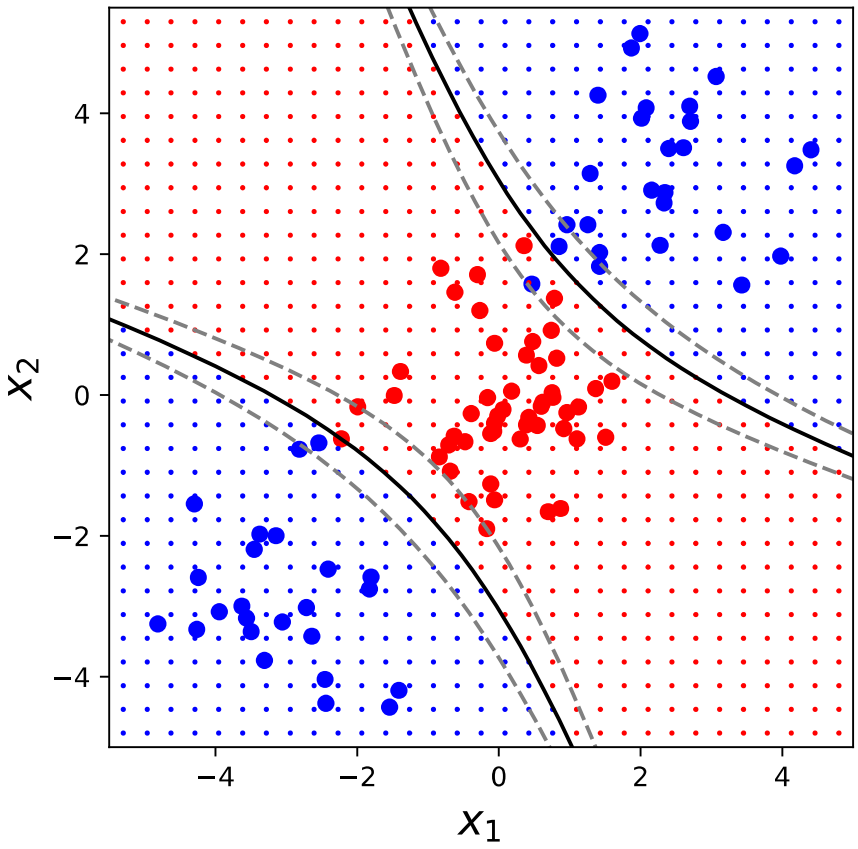
$$f(\quad) = \quad + \sum \quad (m) \mathcal{K}(\quad, \quad (m)) \tag{6.22}$$

$$f(\mathbf{x}) = b + \sum_{m \in \mathcal{S}} \alpha_m y^{(m)} \mathcal{K}(\mathbf{x}, \mathbf{x}^{(m)}) \tag{6.22}$$

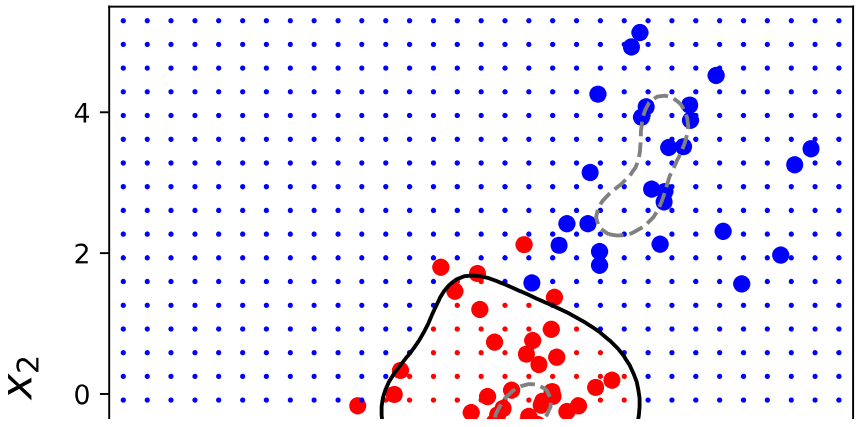
Figure 6.10 (a) shows an example of an SVM with a polynomial kernel applied to the non-linear data from Figure 6.9 (a). The fit is a substantial improvement over the linear support vector classifier. When $d = 1$, then the SVM reduces to the support vector classifier. The polynomial kernel shown in Equation 6.21 is one example of a possible non-linear kernel. Another popular choice is the radial or Gaussian kernel (rbf, short for Radial Basis Function):

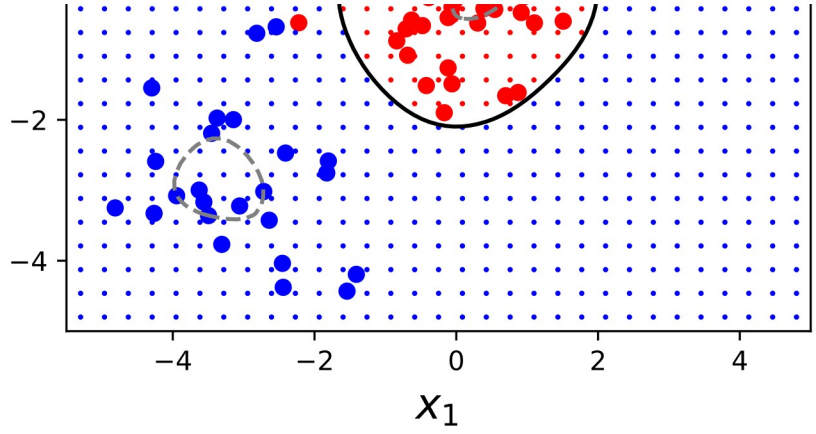
$$\mathcal{K}\left(\mathbf{x}^{(m)}, \mathbf{x}^{(m')}\right) = \exp\left(-\gamma ||\mathbf{x}^{(m)} - \mathbf{x}^{(m')}||^2\right). \tag{6.23}$$

In Equation 6.23, γ is a positive constant inversely proportional to the variance of the Gaussian $\gamma = \frac{1}{\sigma^2}$. That is, with smaller value for γ (larger variance σ^2) the Kernel has a wider spread and "reaches further away". Figure 6.10 (b) shows an SVM with a rbf-kernel applied to the non-linearly separable data from Figure 6.9 (a).



(a) SVM with a polynomial kernel of degree 2.





(b) SVM with a rbf-kernel.

Figure 6.10: Illustration how non-linear kernels in a SVM can capture the non-linear decision boundary in the dataset from Figure 6.9 (a).

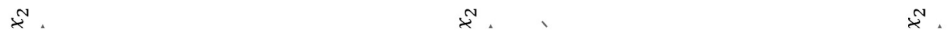
How does the radial kernel Equation 6.23 actually work? If a given test sample $\mathbf{x}^* = (x_1^* \quad x_2^* \quad \dots \quad x_N^*)^T$ is far from a training sample $\mathbf{x}^{(m)}$ in terms of Euclidean distance, then $||\mathbf{x}^{(*)} - \mathbf{x}^{(m)}||^2$ will be large, and so $\mathcal{K}\left(\mathbf{x}^{(m)}, \mathbf{x}^{(m')}\right) = \exp\left(-\gamma||\mathbf{x}^{(m)} - \mathbf{x}^{(m')}||^2\right)$ will be very tiny. This means that in Equation 6.22, $\mathbf{x}^{(m)}$ will play virtually no role in $f(\mathbf{x}^{(*)})$. Recall that the predicted class label for the test sample $\mathbf{x}^{(*)}$ is based on the sign of $f(\mathbf{x}^{(*)})$. In other words, training samples that are far from $\mathbf{x}^{(*)}$ will play essentially no role in the predicted class label for $\mathbf{x}^{(*)}$. This means that the radial kernel has very local behavior, in the sense that only nearby training samples have an effect on the class label of a test sample.

What is the advantage of using a kernel rather than simply enlarging the feature space using functions of the original features, as in Equation 6.15? One advantage is computational, and it amounts to the fact that using kernels, one needs only compute $\mathcal{K}\left(\mathbf{x}^{(m)}, \mathbf{x}^{(m')}\right)$ for all $\binom{M}{2}$ distinct pairs m, m'.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. For some kernels, such as the radial kernel Equation 6.23, the feature space is implicit and infinite-dimensional, so the computations could never be done there anyway.

6.7 SVMs With More Than Two Classes

The SVM is formulated per se as a binary classification problem. It turns out that the concept of separating hyperplanes can not readily be extended to more than two classes $K > 2$. However, the one-versus-all and one-vs-one approaches can be employed.

The **one-vs-all** (or one-vs-rest) approach is described in Section 5.7.1. K SVMs are fitted, each one classifying one of the K classes to the remaining $K - 1$ classes. Let $b_k, w_{1k}, \dots, w_{Nk}$ denote the parameters that result from fitting an SVM comparing the kth class (coded as $+1$) to the others (coded as -1). Let $\mathbf{x}^{(*)}$ denote a test sample (yellow in Figure 6.11). We assign the sample to the class for which $f(\mathbf{x}^*) = b_k + \mathbf{w}_k^T\mathbf{x}^{(*)}$ is largest, as this amounts to a high level of confidence that the test sample belongs to the kth class rather than to any of the other classes.



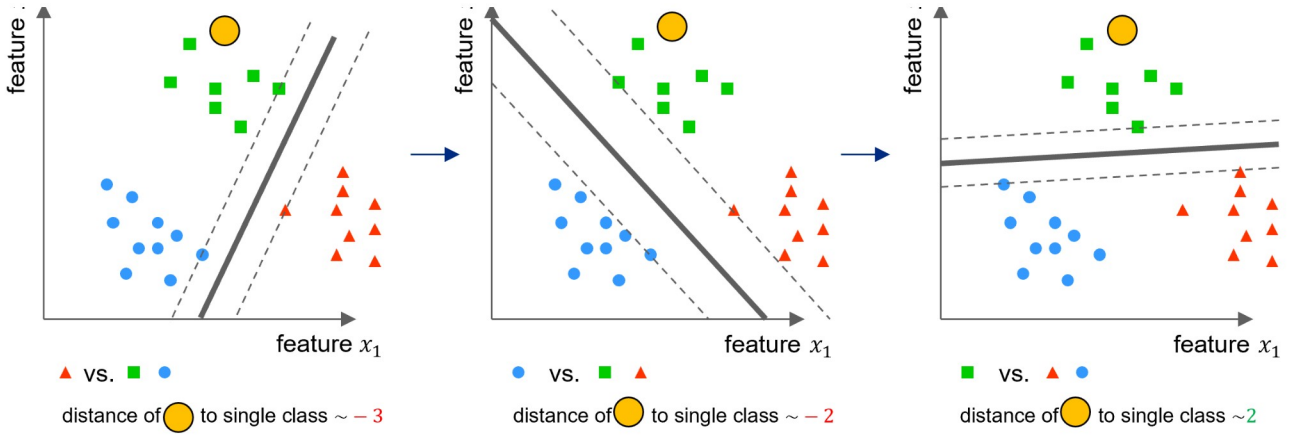


Figure 6.11: Illustration of the one-vs-all scheme applied to 3 classes in colors red, blue and green. With the unknown test sample in yellow

In the **one-versus-one** approach a separate SVM is constructed for each of the $\binom{K}{2}$ distinct pairs of classes Figure 6.12. A test sample is classified using each of the classifiers and the number of times that the test sample is assigned to each of the K classes is counted. The test sample is the assigned to the class to which it was most frequently assigned in those pairwise classifications (majority vote).

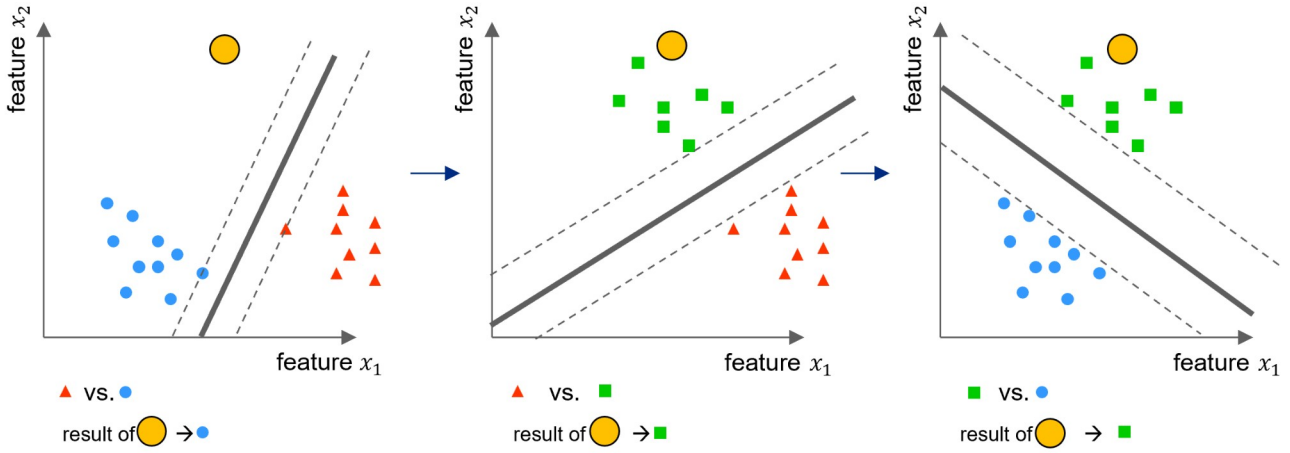


Figure 6.12: Illustration of the one-vs-one scheme applied to 3 classes in colors red, blue and green. The unknown test instance (yellow) is assigned by majority vote

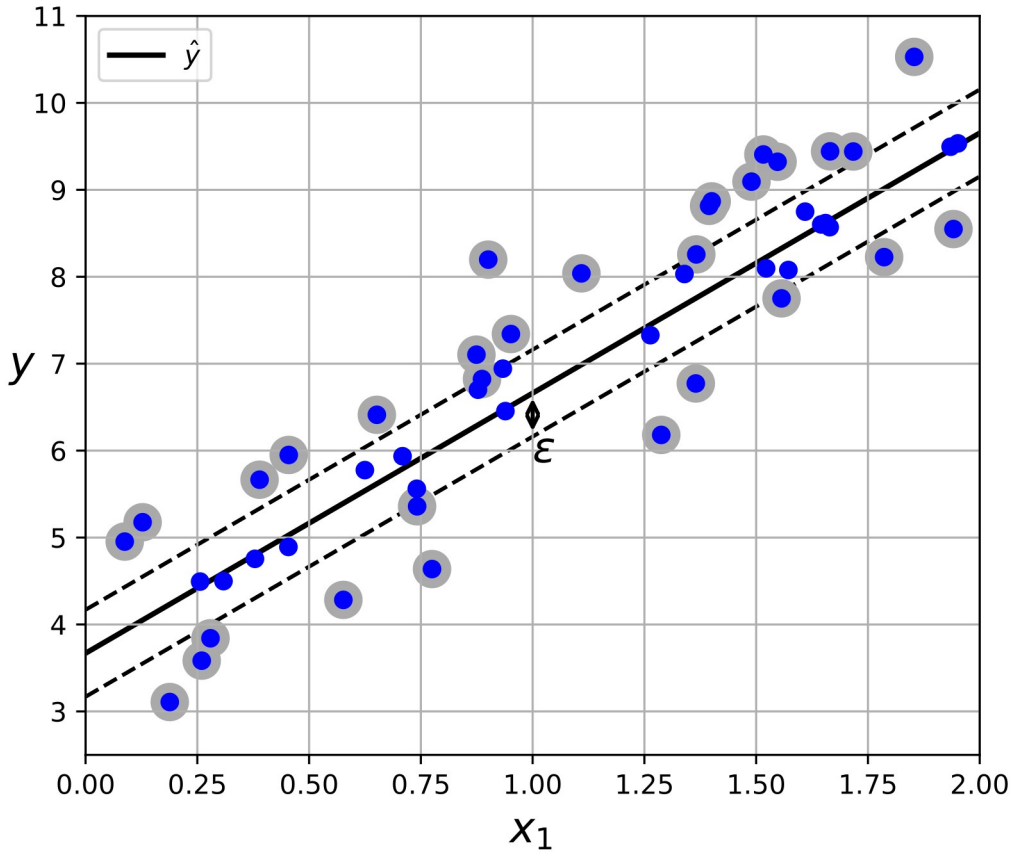
6.8 SVMs for Regression

To use SVMs for regression instead of classification, the trick is to tweak the objective: instead of trying to fit the largest possible street between two classes while limiting margin violations, SVM regression tries to fit as many instances as possible on the street while limiting margin violations (i.e., instances off the street). The width of the street is controlled by a hyperparameter, ϵ. Figure 6.13 shows two linear SVM regression models trained on some linear data, Figure 6.13 (a) with a small margin and Figure 6.13 (b) with a larger margin.

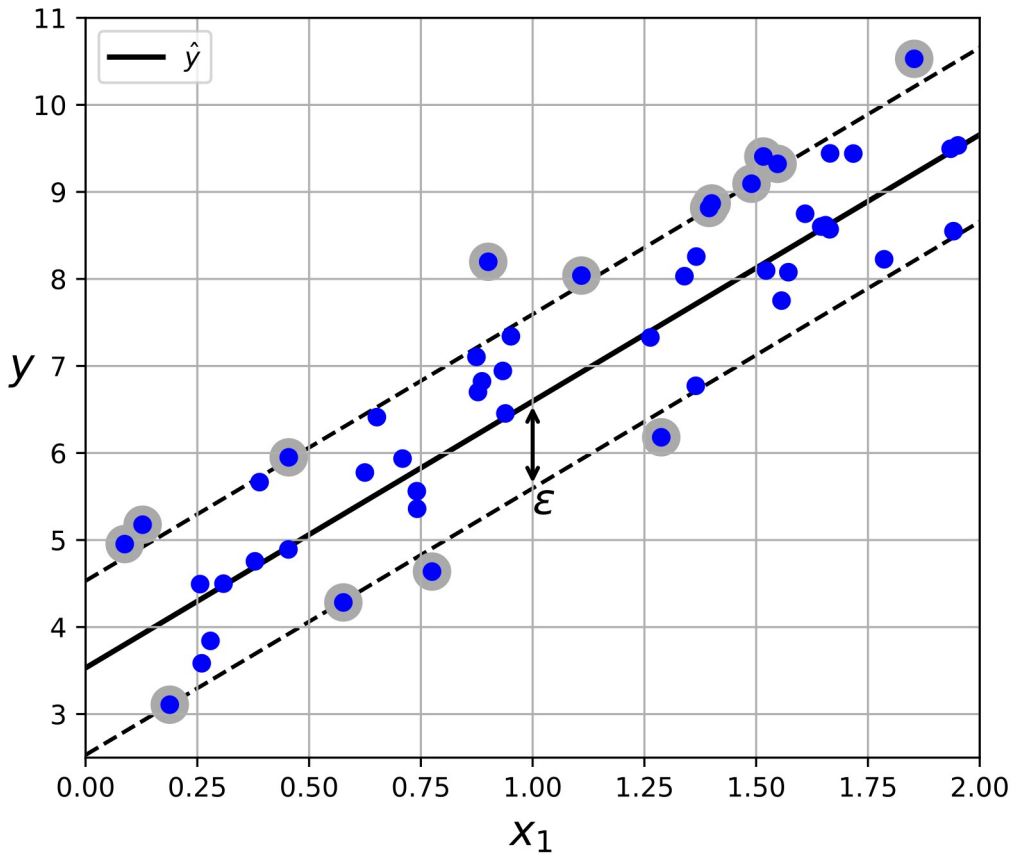
```
/Users/doem/anaconda3/envs/mldm/lib/python3.10/site-packages/sklearn/svm/_classes.py:32:
FutureWarning: The default value of `dual` will change from `True` to ``'auto'`` in 1.5. Set
the value of `dual` explicitly to suppress the warning.
  warnings.warn(

/Users/doem/anaconda3/envs/mldm/lib/python3.10/site-packages/sklearn/svm/_classes.py:32:
```

```
FutureWarning: The default value of `dual` will change from `True` to ``'auto'`` in 1.5. Set
the value of `dual` explicitly to suppress the warning.
  warnings.warn(
```



(a) Parameter epsilon=0.5

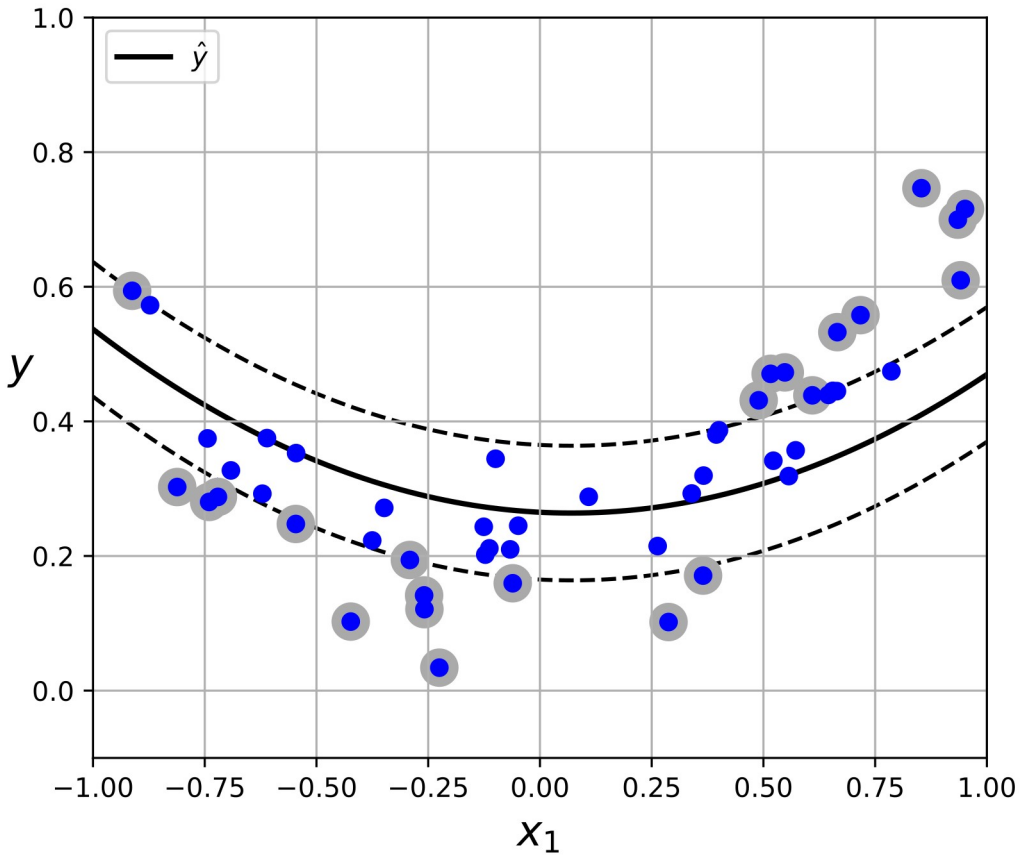


(b) Parameter epsilon=1.0 leads to a wider margin

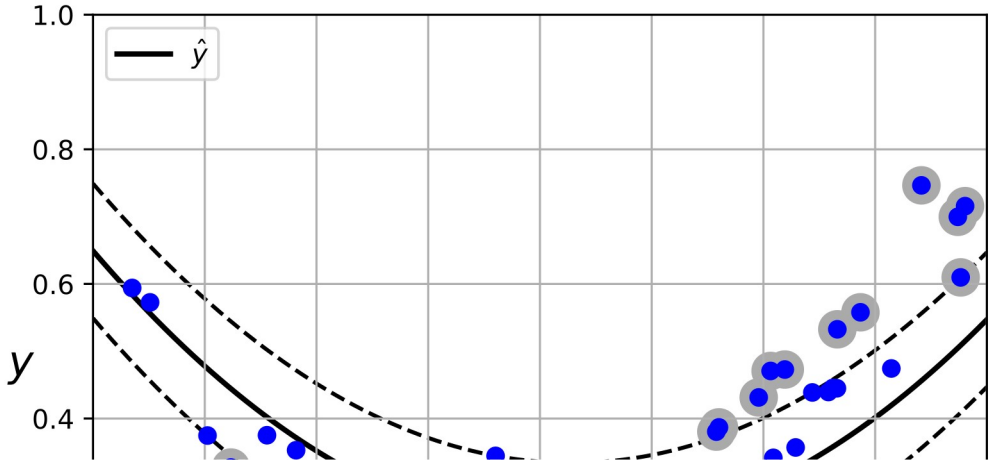
Figure 6.13: Linear SVM regression with different parameter settings controlling the width of the margin. Training data in blue, support vectors marked by grey circles. The solid line represents the predictions from the model, while the dashed lines indicate the margins.

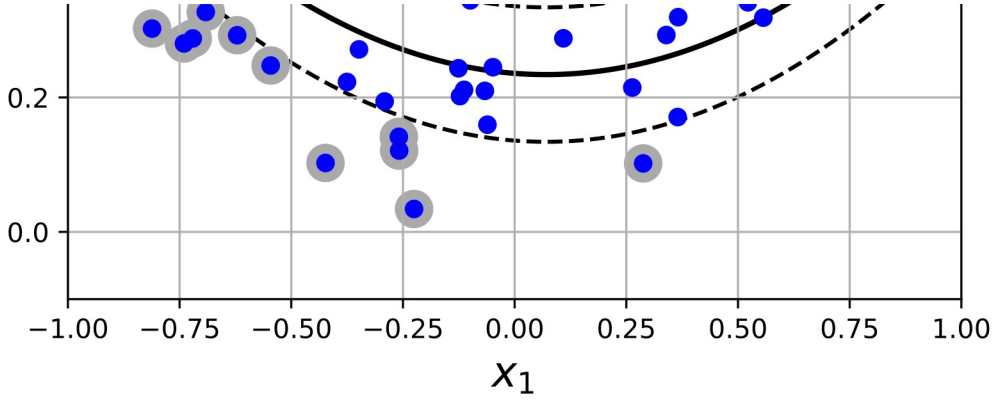
Reducing ϵ increases the number of support vectors, which regularizes the model. Moreover, if more training instances within the margin are added, it will not affect the model's predictions; thus, the model is said to be ϵ-insensitive.

To tackle nonlinear regression tasks, kernelized SVM models can be employed. Figure 6.14 shows SVM regression on a random quadratic training set, using a second-degree polynomial kernel. There is some regularization in Figure 6.14 (a) (i.e., a small C value), and much less in Figure 6.14 (b) (i.e., a large C value).



(a) C=0.01





(b) C=1000

Figure 6.14: SVM regression using a second-degree polynomial kernel with different parameters C. Training data in blue, support vectors marked by grey circles. The solid line represents the predictions from the model, while the dashed lines indicate the margins.

1. The subspace does not need to pass through the origin.↩

2. By expanding each of the inner products in Equation 6.18, it is easy to see that $f(x)$ is a linear function of the coordinates of $\mathbf{x}$. Doing so also establishes the correspondence between the α_m and the original parameters w_m.↩