12

Classification with Support Vector Machines

In many situations we want our machine learning predictor to predict one of a number of outcomes. For example an email client that sorts mail into personal mail and junk mail, which has two outcomes. Another example is a telescope that identifies whether an object in the night sky is a galaxy, star or planet. There are usually a small number of outcomes, and more importantly there is usually no additional structure on these outcomes. In this chapter, we consider predictors that output binary values, that is there are only two possible outcomes. This is in contrast to Chapter 9 where we considered a prediction problem with continuous-valued outputs. This machine learning task is called *binary classification*. For binary classification the set of possible values that the label/output can attain is binary, and for this chapter we denote them as $\{+1, -1\}$. In other words, we consider predictors of the form

An example of structure is if the outcomes were ordered, like in the case of small, medium and large t-shirts.

binary classification

$$f : \mathbb{R}^D \to \{+1, -1\} . \tag{12.1}$$

Recall from Chapter 8 that we represent each example $\boldsymbol{x}_n$ as a feature vector of D real numbers. The labels are often referred to as the positive and negative *classes*, respectively. One should be careful not to infer intuitive attributes of positiveness of the $+1$ class. For example, in a cancer detection task, a patient with cancer is often labelled $+1$. In principle, any two distinct values can be used, e.g., $\{\text{True}, \text{False}\}$, $\{0, 1\}$ or $\{\text{red}, \text{blue}\}$. The problem of binary classification is well studied, and we defer a survey of other approaches to Section 12.4.

classes

For probabilisitic models, it is mathematically convenient to use $\{0, 1\}$ as a binary representation.

We present an approach known as the Support Vector Machine (SVM), which solves the binary classification task. Similar to regression, we have a supervised learning task, where we have a set of inputs $\boldsymbol{x}_n \in \mathbb{R}^D$ along with their corresponding labels $y_n \in \{+1, -1\}$. Given the training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$, we would like to estimate parameters of the model that will give the best classification error. Similar to Chapter 9 we consider a linear model, and hide away the nonlinearity in a transformation ϕ of the input vectors (9.12). We will revisit ϕ later in this chapter in Section 12.3.4.

The first reason we choose to discuss the SVM is to illustrate a geometric way to think about machine learning. Whereas in Chapter 9 we considered the machine learning problem in terms of a noise model and

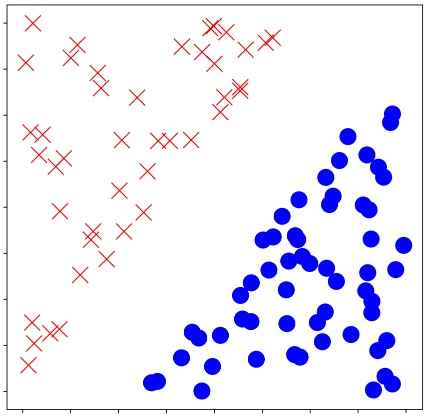

Figure 12.1 Example 2D data, illustrating the intuition of data where we can find a linear classifier that separates red crosses from blue dots.

attacked it using maximum likelihood estimation and Bayesian inference, here we will consider an alternative approach where we reason geometrically about the machine learning task. It relies heavily on concepts, such as inner products and projections, which we discussed in Chapter 3. In contrast to Chapter 9, the optimization problem for SVM does not admit an analytic solution. Hence, we resort to the tools introduced in Chapter 7. This is the second reason for introducing the SVM: as an illustration of what to do when we cannot analytically derive a solution.

The SVM view of machine learning is also subtly different from the maximum likelihood view of Chapter 9. The maximum likelihood view proposes a model based on a probabilistic view of the data distribution, from which an optimization problem is derived. In contrast the SVM view starts by designing a particular function that is to be optimized during training, based on geometric intuitions. In other words, it starts by designing an objective function that is to be minimized on training data. This can also be understood as designing a particular loss function.

Let us derive the optimization problem corresponding to training an SVM on data. Intuitively we imagine binary classification data which can be separated by a hyperplane as illustrated in Figure 12.1. Hyperplane is a word that is commonly used in machine learning, and we saw them in Section 2.8 introduced as an affine subspace, which is the phrase used in linear algebra. The data consists of two classes that have features arranged in such a way as to allow us to separate/classify them by drawing a straight line.

In the following, we start by formalizing this idea of finding a linear separator. We introduce the idea of the margin and then extend linear separators to allow for data points to fall on the wrong side. We present the two equivalent ways of formalizing the SVM: the geometric view (Section 12.2.4) and the loss function view (Section 12.2.5). We derive the dual version of the SVM in two different ways: using Lagrange multipliers (Section 7.2) and using the Legendre-Fenchel transform (Section 7.3.3). The dual SVM allows us to observe a third way of formalizing the SVM: in terms of the convex hulls of the feature vectors of each class (Section 12.3.3). We conclude by briefly describing kernels and how to numerically solve the nonlinear kernel-SVM optimization problem.

12.1 Separating Hyperplanes

Given two data points represented as vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, one way to compute the similarity between them is using a inner product $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$. Recall from Section 3.2 that inner products measure the angle between two vectors. The value of the inner product also depends on the length (norm) of each vector. Furthermore, inner products allow us to rigorously define geometrical concepts such as orthogonality and projections.

The main idea behind many classification algorithms is to represent data in $\mathbb{R}^D$ and then partition this space. In the case of binary classification, the space would be split into two parts corresponding to the positive and negative classes, respectively. We consider a particularly convenient partition, which is to split the space into two halves using a hyperplane. Let $\boldsymbol{x} \in \mathbb{R}^D$ be an element of the data space. Consider a function $f : \mathbb{R}^D \to \mathbb{R}$ parametrized by $\boldsymbol{w} \in \mathbb{R}^D$ and $b \in \mathbb{R}$ as follows

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b\,. \tag{12.2}$$

For fixed values of $\boldsymbol{w}$ and b, the set of all points where $\boldsymbol{w}$ and b is equal to a constant is a hyperplane. By convention we choose the hyperplane that separates the two classes in our binary classification problem to be

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0. \tag{12.3}$$

An illustration of the hyperplane is shown in Figure 12.2 where the vector $\boldsymbol{w}$ is a vector normal to the hyperplane and b the intercept. We can derive that $\boldsymbol{w}$ is normal vector to the hyperplane in Equation (12.3) by choosing any two points $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ on the hyperplane and showing that the vector between them is orthogonal to $\boldsymbol{w}$. In the form of an equation,

$$\begin{aligned} f(\boldsymbol{x}_a) - f(\boldsymbol{x}_b) &= \langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b - (\langle \boldsymbol{w}, \boldsymbol{x}_b \rangle + b) && (12.4)\\ &= \langle \boldsymbol{w}, \boldsymbol{x}_a - \boldsymbol{x}_b \rangle, && (12.5) \end{aligned}$$

where the second line is obtained by the linearity of the inner product (Section 3.2). Since we have chosen $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ to be on the hyperplane, this implies that $f(\boldsymbol{x}_a) = 0$ and $f(\boldsymbol{x}_b) = 0$ and hence $\langle \boldsymbol{w}, \boldsymbol{x}_a - \boldsymbol{x}_b \rangle = 0$. Recall that two vectors are orthogonal when their inner product is zero, therefore we obtain that $\boldsymbol{w}$ is orthogonal to any vector on the hyperplane.

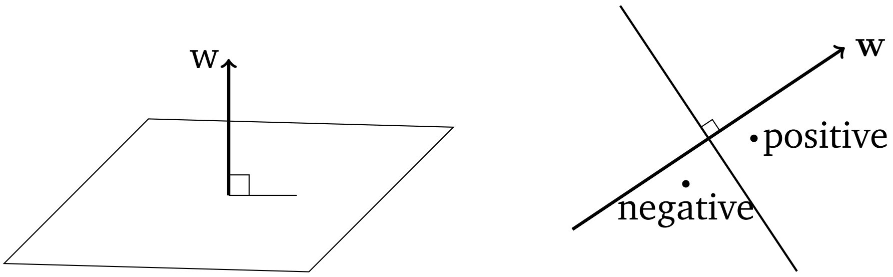

Figure 12.2 Equation of a separating hyperplane (12.3). (left) The standard way of representing the equation in 3D. (right) For ease of drawing, we look at the hyperplane edge on.

Remark. Recall from Chapter 2 that we can think of vectors in different ways. In this chapter, we think of the vector $\boldsymbol{w}$ as an arrow indicating a direction. That is we consider $\boldsymbol{w}$ to be a geometric vector. In contrast we think of the vector $\boldsymbol{x}$ as a point (as indicated by its coordinates). That is we consider $\boldsymbol{x}$ to be the coordinates of a vector with respect to the standard basis. ◇

When presented with a test point, we classify the point as positive or negative by deciding on which side of the hyperplane it occurs. Note that (12.3) not only defines a hyperplane, it actually defines a direction. In other words it defines the positive and negative side of the hyperplane. Therefore, to classify a test point $\boldsymbol{x}_{\text{test}}$, we calculate the value of the function $f(\boldsymbol{x}_{\text{test}})$ and classify the point as $+1$ if $f(\boldsymbol{x}_{\text{test}}) \geqslant 0$ and -1 otherwise. Thinking geometrically, the positive points lie "above" the hyperplane and the negative points "below" the hyperplane.

When training the classifier, we want to ensure that the points with positive labels are on the positive side of the hyperplane, i.e.,

$$\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b \geqslant 0 \quad \text{when} \quad y_n = +1 \tag{12.6}$$

and the points with the negative labels are on the negative side,

$$\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b < 0 \quad \text{when} \quad y_n = -1 \,. \tag{12.7}$$

Refer to Figure 12.2 for a geometric intuition of positive and negative points. These two conditions are often presented in a single equation, which may be puzzling at first glance:

$$y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 0 \,. \tag{12.8}$$

The equation (12.8) above is equivalent to (12.6) and (12.7) when we multiply both sides of (12.6) and (12.7) with $y_n = 1$ and $y_n = -1$, respectively.

12.2 Primal Support Vector Machine

Based on the concept of distances from points to a hyperplane, we now are in a position to discuss the support vector machine. For a dataset $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)$ that is linearly separable, we have many candidate

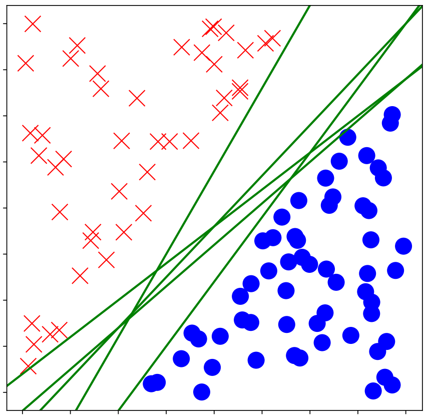

Figure 12.3 Possible separating hyperplanes. There are many linear classifiers (green lines) that separate red crosses from blue dots.

hyperplanes (refer to Figure 12.3) that solve our classification problem without any (training) errors. In other words, for a given training set we have many possible classifiers. One idea is to choose the separating hyperplane that maximizes the margin between the positive and negative data points. In the following, we use the concept of a hyperplane, see also Section 2.8, and derive the distance between a point and a hyperplane. Recall that the closest point on the hyperplane to a given point is obtained by the orthogonal projection (Section 3.6). We will see in the next section how to use the orthogonal projection to derive the margin.

12.2.1 Concept of the Margin

margin

The concept of the *margin* is intuitively simple: It is the distance of the separating hyperplane to the closest point in the dataset, assuming that the dataset is linearly separable. However, when trying to formalize this distance, there is a technical wrinkle that is confusing. The technical wrinkle is that we need to define a scale at which to measure the distance. A potential scale is to consider the scale of the data, i.e., the raw values of $\boldsymbol{x}_n$. There are problems with this, as we could change the units of measurement of $\boldsymbol{x}_n$ and change the values in $\boldsymbol{x}_n$, and, hence, change the distance to the hyperplane. As we will see shortly, we define the scale based on the equation of the hyperplane (12.3) itself. But first let us recall vector addition (Section 2.4) and apply it to derive the margin.

Consider a hyperplane $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$, and two points $\boldsymbol{x}_a$ and $\boldsymbol{x}'_a$ as illustrated in Figure 12.4. Without loss of generality, we can consider the point $\boldsymbol{x}_a$ to be on the positive side of the hyperplane, i.e., $\langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b > 0$. We would like to derive the distance $r > 0$ of $\boldsymbol{x}_a$ from the hyperplane. We

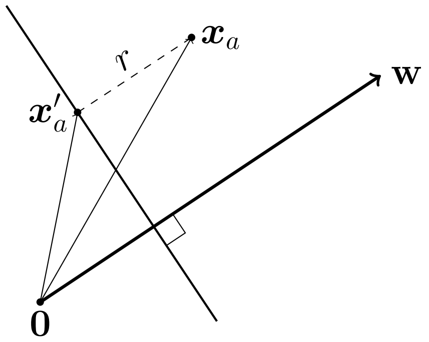

Figure 12.4 Vector addition to express distance to hyperplane: $\boldsymbol{x}_a = \boldsymbol{x}'_a + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$

do so by considering the orthogonal projection (Section 3.6) of $\boldsymbol{x}_a$ onto the hyperplane, which we denote by $\boldsymbol{x}'_a$. Since $\boldsymbol{w}$ is orthogonal to the hyperplane, we know that the distance r is just a scaling of this vector $\boldsymbol{w}$. However, we need to use a vector of unit length (its norm must be 1), and obtain this by dividing $\boldsymbol{w}$ by its norm, $\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$. Using vector addition we obtain

$$\boldsymbol{x}_a = \boldsymbol{x}'_a + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\,. \tag{12.9}$$

Another way of thinking about r is that it is the coordinate of $\boldsymbol{x}_a$ in the subspace spanned by $\boldsymbol{w}$. We have now computed the distance of $\boldsymbol{x}_a$ from the hyperplane, and will see that this distance is the margin.

Recall that we would like the positive points to be further than r from the hyperplane, and the negative points to be further than distance r (in the negative direction) from the hyperplane. Analogously to the combination of (12.6) and (12.7) into (12.8), we have

$$y_n(\langle \boldsymbol{w}, x_n\rangle + b) \geqslant r\,. \tag{12.10}$$

In other words we can combine the requirements that points are further than r from the hyperplane (in the positive and negative direction) into one single inequality.

Let us consider the parameter vector $\boldsymbol{w}$ again, and observe that we actually use it to indicate the direction of the normal of the hyperplane. Since we are interested only in the direction, we add an assumption to our model that the parameter vector $\boldsymbol{w}$ is of unit length, that is $\|\boldsymbol{w}\| = 1$. Collecting the three requirements into one constrained optimization problem, we obtain the following

A reader familiar with other presentations of the margin would notice that our definition of $\|\boldsymbol{w}\| = 1$ is different from the presentation in for example Schölkopf and Smola (2002). We will show that the two approaches are equivalent in Section 12.2.3.

$$\max_{\boldsymbol{w},b,r} \underbrace{r}_{\text{margin}} \quad \text{subject to} \quad \underbrace{y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b) \geqslant r}_{\text{data fitting}}, \quad \underbrace{\|\boldsymbol{w}\| = 1}_{\text{normalization}}, \; r > 0 \tag{12.11}$$

which says that we want to maximize the margin r, while ensuring that the data lies on the correct side of the hyperplane.

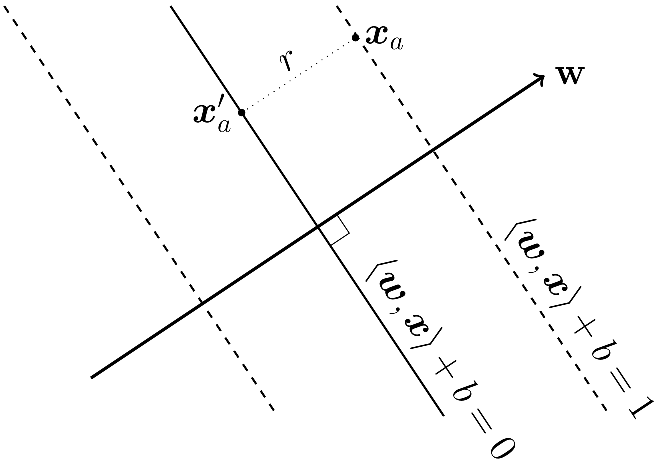
$\boldsymbol{x}_a$
r
$\mathbf{w}$
$\boldsymbol{x}'_a$
$\langle \boldsymbol{w}, \boldsymbol{x}\rangle + b = 1$
$\langle \boldsymbol{w}, \boldsymbol{x}\rangle + b = 0$

Figure 12.5 Derivation of the margin: $r = \frac{1}{\|\boldsymbol{w}\|}$

Remark. The idea of the margin turns out to be highly pervasive in machine learning. It was used by Vladimir Vapnik and Alexey Chervonenkis to show that when the margin is large, the "complexity" of the function class is low, and, hence, learning is possible (Vapnik, 2000). It turns out that the concept is useful for various different approaches for theoretically analyzing generalization error (Shalev-Shwartz and Ben-David, 2014). ♢

12.2.2 Traditional Derivation of the Margin

In the previous section, we derived Equation (12.11) by making the observation that we are only interested in the direction of $\boldsymbol{w}$ and not its length, leading to the assumption that $\|\boldsymbol{w}\| = 1$. In this section, we derive the margin maximization problem by making a different assumption. Instead of choosing that the parameter vector is normalised, we choose a scale for the data. We choose this scale such that the value of the predictor $\langle \boldsymbol{w}, \boldsymbol{x}\rangle + b$ is 1 at the closest point. Recall that we consider linearly separable data. Let us also consider $\boldsymbol{x}_a$ to be the example in the dataset that is closest to the hyperplane.

Figure 12.5 is the same as Figure 12.4, except that now we have rescaled the axes, such that we have the point $\boldsymbol{x}_a$ exactly on the margin, i.e., $\langle \boldsymbol{w}, \boldsymbol{x}_a\rangle + b = 1$. Since $\boldsymbol{x}'_a$ is the orthogonal projection of $\boldsymbol{x}_a$ onto the hyperplane, it must by definition lie on the hyperplane, i.e.,

$$\langle \boldsymbol{w}, \boldsymbol{x}'_a\rangle + b = 0\,. \tag{12.12}$$

By substituting (12.9) into (12.12) we obtain

$$\langle \boldsymbol{w}, \boldsymbol{x}_a - r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\rangle + b = 0\,. \tag{12.13}$$

Multiplying out the inner product, we get

$$\langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b - r\frac{\langle \boldsymbol{w}, \boldsymbol{w} \rangle}{\|\boldsymbol{w}\|} = 0\,, \tag{12.14}$$

where we exploited the linearity of the inner product (see Section 3.2). Observe that the first term is unity by our assumption of scale, that is $\langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b = 1$. From (3.18) in Section 3.1 we recall that $\langle \boldsymbol{w}, \boldsymbol{w} \rangle = \|\boldsymbol{w}\|^2$, and hence the second term reduces to $r\|\boldsymbol{w}\|$. Using these simplifications, we obtain

$$r = \frac{1}{\|\boldsymbol{w}\|}\,, \tag{12.15}$$

where we have derived the distance r in terms of the hyperplane $\boldsymbol{w}$. At first glance this equation is counterintuitive as we seem to have derived the distance from the hyperplane in terms of the length of the vector $\boldsymbol{w}$, but we do not yet know this vector. We will revisit the choice that the margin is 1 in Section 12.2.3. One way to think about it is to consider the distance r to be a temporary variable that we only use for this derivation. In fact, for the rest of this section we will refer to the distance to the hyperplane by $\frac{1}{\|\boldsymbol{w}\|}$.

We can also think of the distance as the projection error that incurs when projecting $\boldsymbol{x}_a$ onto the hyperplane.

Similar to the argument to obtain Equation (12.10), we want the positive points to be further than 1 from the hyperplane, and the negative points to be further than distance 1 (in the negative direction) from the hyperplane

$$y_n(\langle \boldsymbol{w}, x_n \rangle + b) \geqslant 1\,. \tag{12.16}$$

Combining the margin maximization with the fact that data needs to be on the correct side of the hyperplane gives us

$$\max_{w,b} \frac{1}{\|\boldsymbol{w}\|} \tag{12.17}$$

$$\text{subject to } y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1 \quad \text{for all} \quad n = 1, \ldots, N. \tag{12.18}$$

Instead of maximizing the reciprocal of the norm as in (12.17), we often minimize the squared norm. We also often include a constant $\frac{1}{2}$ that does not affect the optimal $\boldsymbol{w}, b$ but yields a tidier form when we take the derivative. Then, our objective becomes

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{12.19}$$

$$\text{subject to } y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1 \quad \text{for all} \quad n = 1, \ldots, N\,. \tag{12.20}$$

Equation (12.19) is known as the *hard margin SVM*. The reason for the expression "hard" is because the above formulation does not allow for any violations of the margin condition. We will see in Section 12.2.4 that this "hard" condition can be relaxed to accommodate violations.

hard margin SVM

12.2.3 Why we can set the Margin to 1

In Section 12.2.1 we argue that we would like to maximize some value r, which represents the distance of the closest point to the hyperplane. In Section 12.2.2 we scaled the data such that the closest point is of distance 1 to the hyperplane. Here we relate the two derivations, and show that the they are actually equivalent.

Theorem 12.1. *Maximizing the margin r where we consider normalized weights as in Equation* (12.11)*,*

$$\max_{\boldsymbol{w},b,r} \underbrace{r}_{\textit{margin}} \quad \text{subject to} \quad \underbrace{y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b) \geqslant r}_{\textit{data fitting}}, \quad \underbrace{\|\boldsymbol{w}\| = 1}_{\textit{normalization}} \quad, r > 0 \tag{12.21}$$

is equivalent to scaling the data such that the margin is unity

$$\min_{\boldsymbol{w},b} \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\textit{margin}} \quad \text{subject to} \quad \underbrace{y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b) \geqslant 1}_{\textit{data fitting}}. \tag{12.22}$$

Proof Consider (12.11), and note that because the square is a monotonic transformation for non-negative arguments, the maximum stays the same if we consider r^2 in the objective. Since $\|\boldsymbol{w}\| = 1$ we can reparameterize the equation with a new weight vector $\boldsymbol{w}'$ that is not normalized by explicitly using $\frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|}$,

$$\max_{\boldsymbol{w}',b,r} \quad r^2 \quad \text{subject to} \quad y_n\left(\langle \frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|}, \boldsymbol{x}_n\rangle + b\right) \geqslant r, \quad r > 0. \tag{12.23}$$

In (12.23) we have explicitly written that distances are non-negative. We can divide the first constraint by r,

$$\max_{\boldsymbol{w}',b,r} \quad r^2 \quad \text{subject to} \quad y_n\left(\langle \underbrace{\frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|\, r}}_{\boldsymbol{w}''}, \boldsymbol{x}_n\rangle + \underbrace{\frac{b}{r}}_{b''}\right) \geqslant 1, \quad r > 0 \tag{12.24}$$

renaming the parameters to $\boldsymbol{w}''$ and b''. Since $\boldsymbol{w}'' = \frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|r}$, rearranging for r gives

$$\|\boldsymbol{w}''\| = \left\|\frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|\, r}\right\| = \left|\frac{1}{r}\right| \cdot \left\|\frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|}\right\| = \frac{1}{r}. \tag{12.25}$$

Substituting into (12.24), we obtain

$$\max_{\boldsymbol{w}'',b''} \quad \frac{1}{\|\boldsymbol{w}''\|^2} \quad \text{subject to} \quad y_n\left(\langle \boldsymbol{w}'', \boldsymbol{x}_n\rangle + b''\right) \geqslant 1. \tag{12.26}$$

The final step is to observe that maximizing $\frac{1}{\|\boldsymbol{w}\|^2}$ yields the same solution as minimizing $\frac{1}{2}\|\boldsymbol{w}\|^2$. □

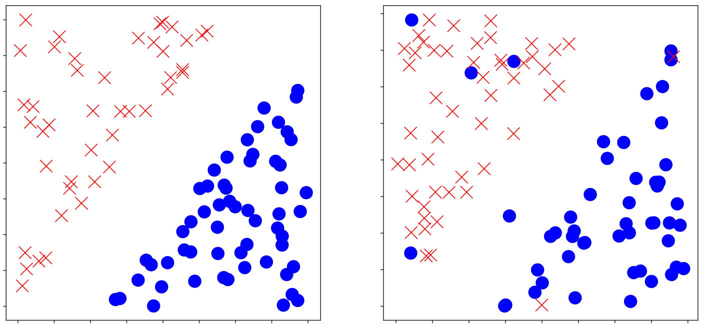

Figure 12.6 (left) linearly separable data, with a large margin. (right) non-separable data

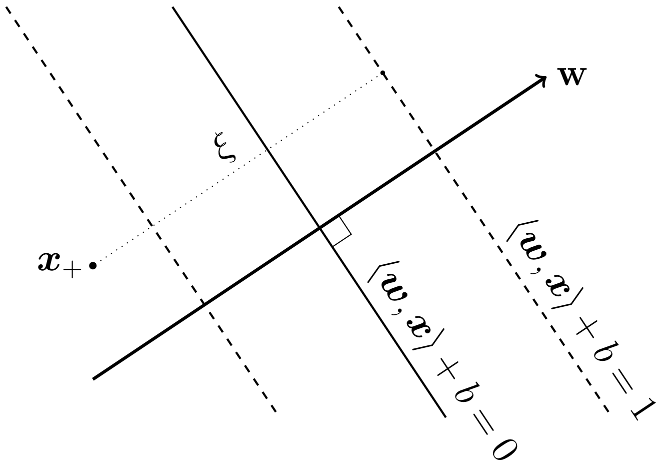
$\mathbf{w}$
ξ
$\boldsymbol{x}_+$
$\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$
$\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 1$

Figure 12.7 Soft Margin SVM allows points to be within the margin or on the wrong side of the hyperplane. The slack variable ξ measures the distance of a positive point $\boldsymbol{x}_+$ to the positive margin hyperplane $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 1$ when $\boldsymbol{x}_+$ is on the wrong side.

12.2.4 Soft Margin SVM: Geometric View

We may wish to allow some points to fall within the margin region, or even to be on the wrong side of the hyperplane (as illustrated in Figure 12.6). This also naturally provides us with an approach that works when we do not have linearly separable data.

soft margin SVM

The resulting model is called the *soft margin SVM*. In this section, we derive the resulting optimization problem using geometric arguments. In Section 12.2.5, we will derive the same optimization problem using the idea of a loss function. Using Lagrange multipliers (Section 7.2), we will derive the dual optimization problem of the SVM in Section 12.3. This dual optimization problem allows us to observe a third interpretation of the SVM, as a hyperplane that bisects the line between convex hulls corresponding to the positive and negative data points (Section 12.3.3).

slack variable

The key geometric idea is to introduce a *slack variable* ξ_n corresponding to each example $(\boldsymbol{x}_n, y_n)$ that allows a particular example to be within the margin or even on the wrong side of the hyperplane (refer to Figure 12.7). We subtract the value of ξ_n from the margin, constraining ξ_n to be positive. To avoid all points from being assigned incorrectly, we add ξ_n to the objective

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n \tag{12.27}$$

$$\text{subject to } y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b) \geqslant 1 - \xi_n \quad \text{for all} \quad n = 1, \ldots, N \tag{12.28}$$

$$\xi_n \geqslant 0 \quad \text{for all} \quad n = 1, \ldots, N. \tag{12.29}$$

Support Vector Machine

soft margin SVM

regularization parameter

regularizer

C-SVM

The resulting optimization problem (12.27) is called the *Support Vector Machine* (SVM). In contrast to the optimization problem (12.19) from the previous section (the hard margin SVM), this is one called the *soft margin SVM*. The parameter C trades off the size of the margin and the total amount of slack that we have. This parameter is called the *regularization parameter* since, as we will see in the following section, the margin term in the objective function (12.27) is a regularization term. The margin term $\|\boldsymbol{w}\|^2$ is called the *regularizer,* and in many books on numerical optimization, the regularization parameter multiplied with this term (Section 8.3.3). This is in contrast to our formulation in this section. Some care needs to be taken when interpreting the regularizer, as a large value of C implies low regularization, as we give the slack variables larger weight. There are alternative parametrizations of this regularization, which is why (12.27) is also often referred to as the *C-SVM*.

Remark. One detail to note is that in the formulation of the SVM (Equation (12.27)) $\boldsymbol{w}$ is regularized by b is not regularized. We can see this by observing that the regularization term does not contain b. ♢

12.2.5 Soft Margin SVM: Loss Function View

loss function

Recall from Section 9.2.1 that when performing maximum likelihood estimation we usually consider the negative log likelihood. Furthermore since the likelihood term for linear regression with Gaussian noise is Gaussian, the negative log likelihood for each example is a squared error function (Equation (9.8)). The squared error function is the term that is minimized when looking for the maximum likelihood solution. Let us consider the error function point of view, which is also known as the *loss function* point of view. Note that in contrast to Chapter 9 where we consider regression problems (the output of the predictor is a real number), in this chapter we consider binary classification problems (the output of the predictor is one of two labels $\{+1, -1\}$). Therefore the error function or the loss function for each single (example, label) pair needs to be appropriate for binary classification. For example, the squared loss that is used for regression (Equation (9.9b)) is not suitable for binary classification.

Remark. The ideal loss function between binary labels is to count the number of mismatches between the prediction and the label. That is for a predictor $f(\cdot)$ applied to an example $\boldsymbol{x}_n$, we compare the output $f(\boldsymbol{x}_n)$ with the label y_n. We define the loss to be zero if they match, and one if they do not match. This is denoted by $\mathbf{1}(f(\boldsymbol{x}_n) \neq y_n)$ and is called the zero-one loss. Unfortunately the zero-one loss results in a difficult optimization problem for finding the best parameters $\boldsymbol{w}, b$. ♢

What is the loss function corresponding to the SVM? Consider the error between the output of a predictor $f(\boldsymbol{x}_n)$ and the label y_n. The loss should capture how much we care about the error that is made on the training data. An equivalent way to derive (12.27) is to use the *hinge loss*

hinge loss

$$\ell(t) = \max\{0, 1 - t\} \quad \text{where} \quad t = yf(\boldsymbol{x}) = y(\langle \boldsymbol{w}, \boldsymbol{x}\rangle + b)\,. \tag{12.30}$$

If $f(\boldsymbol{x})$ is on the correct side (based on y) of the hyperplane, and further than distance 1, this means that $t \geqslant 1$ and the hinge loss returns a value of zero. If $f(\boldsymbol{x})$ is on the correct side but close to the hyperplane, that is $0 < t < 1$, then the point $\boldsymbol{x}$ is within the margin and the hinge loss returns a positive value. When the point is on the wrong side of the hyperplane ($t < 0$) the hinge loss returns an even larger value, which increases linearly. In other words we pay a penalty once we are closer than the margin, even if the prediction is correct, and the penalty increases linearly. An alternative way to express the hinge loss is by considering it as two linear pieces

$$\ell(t) = \begin{cases} 0 & \textbf{if} \quad t \geqslant 1 \\ 1 - t & \textbf{if} \quad t < 1 \end{cases}, \tag{12.31}$$

as illustrated in Figure 12.8.

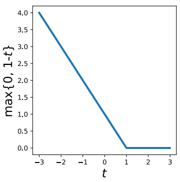

Figure 12.8 Hinge loss

For a given training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ we would like to minimize the total loss, while regularizing the objective with ℓ_2 regularization (see Section 8.3.3). This gives us the unconstrained optimization problem

$$\min_{\boldsymbol{w},b} \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{regularizer}} + \underbrace{C\sum_{n=1}^{N} \max\{0, 1 - y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b)\}}_{\text{error term}}\,. \tag{12.32}$$

The first term in (12.32) is called the regularization term or the *regularizer* (see Section 9.2.3), and the second term is called the *loss term* or the *error term*. Recall from Section 12.2.4 that the term $\frac{1}{2}\|\boldsymbol{w}\|^2$ is actually the term arising from the margin. In other words margin maximization can be interpreted as a regularizer.

regularizer

loss term

error term

Margin maximization can be interpreted as a regularizer.

In principle, the unconstrained optimization problem in (12.32) can be directly solved with (sub-)gradient descent methods as described in Section 7.1. To see that (12.32) and (12.27) are equivalent, observe that the hinge loss (12.30) essentially consists of two linear parts, as expressed in (12.31). Therefore, we can equivalently replace the hinge loss with two constraints, i.e.,

$$\min_{t} \max\{0, 1-t\} \tag{12.33}$$

is equivalent to

$$\begin{aligned} \min_{\xi, t} \quad & \xi \\ \text{subject to} \quad & \xi \geqslant 0 \\ & \xi \geqslant 1 - t\,. \end{aligned} \tag{12.34}$$

By substituting this into (12.32) and rearranging one of the constraints, we obtain exactly the soft margin SVM (12.27).

Remark. Observe that the hinge loss has three equivalent representations, as shown by (12.30) and (12.31), as well as the constrained optimization problem in (12.34). ♢

12.3 Dual Support Vector Machine

The description of the SVM in the previous sections, in terms of the variables $\boldsymbol{w}$ and b, is known as the primal SVM. Recall that we are considering input vectors $\boldsymbol{x}$, which have dimension D, i.e., we are looking at input examples with D features. Since $\boldsymbol{w}$ is of the same dimension as $\boldsymbol{x}$, this means that the number of parameters (the dimension of $\boldsymbol{w}$) of the optimization problem grows linearly with the number of features.

In the following, we consider an equivalent optimization problem (the so-called dual view) which is independent of the number of features. We will see a similar idea appear in Chapter 10 where we express the learning problem in a way that does not scale with the number of features. This is useful for problems where we have more features than number of data points. Instead the number of parameters increases with the number of data points in the training set. The dual SVM also has the additional advantage that it easily allows kernels to be applied, as we shall see at the end of this chapter. The word "dual" appears often in mathematical literature, and in this particular case it refers to convex duality. The following subsections are essentially an application of convex duality as discussed in Section 7.2.

12.3.1 Convex Duality via Lagrange Multipliers

Recall the primal soft margin SVM (12.27). We call the variables $\boldsymbol{w}$, b and ξ corresponding to the primal SVM the primal variables. We use $\alpha \geqslant 0$ as the Lagrange multiplier corresponding to the constraint (12.28) that the points are classified correctly and $\gamma \geqslant 0$ as the Lagrange multiplier corresponding to the non-negativity constraint of the slack variable, see (12.29). The Lagrangian is then given by

$$\mathcal{L}(\boldsymbol{w}, b, \xi, \alpha, \gamma) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n \\ \underbrace{-\sum_{n=1}^{N}\alpha_n(y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b) - 1 + \xi_n)}_{\text{constraint (12.28)}} \underbrace{-\sum_{n=1}^{N}\gamma_i\xi_n}_{\text{constraint (12.29)}} . \tag{12.35}$$

Differentiating the Lagrangian (12.35) with respect to the three primal variables $\boldsymbol{w}$, b and ξ respectively, we obtain

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}} = \boldsymbol{w} - \sum_{n=1}^{N}\alpha_n y_n \boldsymbol{x}_n , \tag{12.36}$$

$$\frac{\partial\mathcal{L}}{\partial b} = \sum_{n=1}^{N}\alpha_n y_n , \tag{12.37}$$

$$\frac{\partial\mathcal{L}}{\partial\xi_n} = C - \alpha_n - \gamma_i . \tag{12.38}$$

We now find the maximum of the Lagrangian by setting each of these partial derivatives to zero. By setting (12.36) to zero we find

$$\boldsymbol{w} = \sum_{n=1}^{N}\alpha_n y_n \boldsymbol{x}_n , \tag{12.39}$$

which is a particular instance of the representer theorem (Kimeldorf and Wahba, 1970). Equation (12.39) says that the optimal weight vector in the primal is a convex combination of the data points. Recall from Section 2.6.1 that this means that the solution of the optimization problem lies in the span of training data. The representer theorem turns out to hold for very general settings of regularized empirical risk minimization (Hofmann et al., 2008; Argyriou and Dinuzzo, 2014). By substituting the expression for $\boldsymbol{w}$ into the Lagrangian (12.35), we obtain the dual

The representer theorem is actually a collection of theorems saying that the solution of minimizing empirical risk lies in the subspace (Section 2.4.3) defined by the data points.

$$\mathcal{D}(\xi, \alpha, \gamma) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle - \sum_{i=1}^{N} y_i \alpha_i \langle \sum_{j=1}^{N} y_j \alpha_j \boldsymbol{x}_j, \boldsymbol{x}_i\rangle \\ + C\sum_{i=1}^{N}\xi_i - b\sum_{i=1}^{N} y_i\alpha_i + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i\xi_i - \sum_{i=1}^{N}\gamma_i\xi_i . \tag{12.40}$$

Note that the terms involving $\boldsymbol{w}$ have cancelled out. By setting (12.37) to zero, we obtain $\sum_{n=1}^{N} y_n\alpha_n = 0$. Therefore, the term involving b also vanishes. Recall that inner products are symmetric and linear (see Section 3.2). Therefore, the first two terms in (12.40) are over the same objects. These terms (coloured blue) can be simplified, and we obtain the Lagrangian

$$\mathcal{L}(\boldsymbol{w}, b, \xi, \alpha, \gamma) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_{i=1}^{N} \alpha_i + \sum_{i=1}^{N} (C - \alpha_i - \gamma_i)\xi_i \,. \tag{12.41}$$

The last term in this equation is a collection of all terms that contain the slack variable ξ. By setting (12.38) to zero, we see that the last term in (12.40) is also zero. Furthermore, by using the same equation and recalling that the Lagrange multipler γ is non-negative, we conclude that $\alpha_i \leqslant C$. We now obtain the dual optimization problem of the SVM, which is expressed exclusively in terms of the Lagrange multiplier α. Recall from Lagrangian duality (Theorem 7.1) that we maximize the dual problem. This is equivalent to minimizing the negative dual problem, such that we end up with the *dual SVM*

dual SVM

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - \sum_{i=1}^{N} \alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & 0 \leqslant \alpha_i \leqslant C \quad \text{for all} \quad i = 1, \ldots, n \,. \end{aligned} \tag{12.42}$$

The set of inequality constraints in the SVM are called "box constraints" because they limit the vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^\top \in \mathbb{R}^N$ of Lagrange multipliers to be inside the box defined by 0 and C on each axis. These axis-aligned boxes are particularly efficient to implement in numerical solvers (Dostál, 2009, Chapter 5).

12.3.2 Convex Duality via the Convex Conjugate

In this section, we use the idea of the Legendre-Fenchel transform, also known as the convex conjugate (Hiriart-Urruty and Lemaréchal, 2001, Chapter 5), to derive the dual SVM. Convex conjugates were discussed in Section 7.3.3. Recall the unconstrained version of the SVM given by

$$\min_{\boldsymbol{w}} \underbrace{\frac{\lambda}{2}\|\boldsymbol{w}\|^2}_{\text{regularizer}} + \sum_{n=1}^{N} \underbrace{\max\{0, 1 - y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle)\}}_{\text{hinge loss}} \,. \tag{12.43}$$

For simplicity, we removed the bias term b from the predictor. It turns out that for high-dimensional feature spaces, such as when we use a nonlinear kernel (Section 12.3.4) the bias or offset term b does not have a large effect (Steinwart and Christmann, 2008). This simplifies matters because now both terms in the objective are functions of the same parameter

We retain b in the presentation in the other sections to be in line with other presentations of SVM.

$\boldsymbol{w}$. By looking at the derivation of the Lagrangian duality (12.37), we observe that the equality constraint is due to the bias term b. Since we do not model the bias term here, this term does not appear. We have also changed the regularization parameter from C multiplying the loss term to λ multiplying the regularization term. This turns out to simplify algebra later in this section.

The presentation with C is common in the SVM literature, but the presentation with λ is common in the regularization methods literature

Following the convention by Rifkin and Lippert (2007), the dual of the sum of two functions

$$\min_{\boldsymbol{y}\in\mathbb{R}^N} F(\boldsymbol{y}) + G(\boldsymbol{y}) \tag{12.44}$$

is given by

$$\min_{\boldsymbol{z}\in\mathbb{R}^N} F^*(\boldsymbol{z}) + G^*(-\boldsymbol{z}) \tag{12.45}$$

where $F^*(\cdot)$ and $G^*(\cdot)$ are the convex conjugates of $F(\cdot)$ and $G(\cdot)$ respectively (Section 7.3.3).

Considering the regularization term, recall from Chapter 9 that the closed-form solution of the parameters is obtained in (9.11c). Recall that the closed form solution is

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}\,, \tag{12.46}$$

where $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N\times D}$ and $\boldsymbol{y} = [y_1, \dots, y_N]^\top \in \mathbb{R}^N$ are the collections of training inputs and targets, respectively. Plugging $\hat{\boldsymbol{w}}$ into the expression of the regularizer we obtain

$$\begin{aligned}\frac{\lambda}{2}\|\boldsymbol{w}\|^2 &= \frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w} &(12.47)\\ &= \frac{\lambda}{2}\left[(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}\right]^\top (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} &(12.48)\\ &= \frac{\lambda}{2}\boldsymbol{y}^\top\boldsymbol{K}^{-1}\boldsymbol{y} &(12.49)\end{aligned}$$

where the last line was obtained by defining $\boldsymbol{K} := \boldsymbol{X}\boldsymbol{X}^\top$ and using the matrix identity in Equation (12.50).

Remark. This matrix identity allows us to commute the required matrix multiplications to prove Equation (12.49).

$$(\boldsymbol{X}\boldsymbol{X}^\top)^{-1} = \boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top \tag{12.50}$$

Observe that the number of terms containing $\boldsymbol{X}$ coincide with the needed number for $\boldsymbol{K}^{-1}$, but we cannot easily swap terms containing $\boldsymbol{X}$ because matrix multiplication is not commutative. We derive the above identity in two steps. First directly multiply the left hand side (without the inverse) with the right hand side.

$$\begin{aligned}&(\boldsymbol{X}\boldsymbol{X}^\top)\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top &(12.51)\\ &\quad= \boldsymbol{X}[(\boldsymbol{X}^\top\boldsymbol{X})(\boldsymbol{X}^\top\boldsymbol{X})^{-1}](\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top &(12.52)\end{aligned}$$

$$= \boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top. \tag{12.53}$$

where we have used the definition of the identity matrix in the middle equation for the terms in the square brackets.

Second observe that for identity matrices $\boldsymbol{I}$ of appropriate size, we can pre and post multiply $\boldsymbol{X}$. Then by using the definition of identity $(\boldsymbol{X}^\top\boldsymbol{X})^{-1}(\boldsymbol{X}^\top\boldsymbol{X}) = \boldsymbol{I}$, we get

$$\boldsymbol{X}\boldsymbol{I} = \boldsymbol{I}\boldsymbol{X} \tag{12.54}$$

$$\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}(\boldsymbol{X}^\top\boldsymbol{X}) = \boldsymbol{I}\boldsymbol{X} \tag{12.55}$$

$$[\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top]\boldsymbol{X} = \boldsymbol{I}\boldsymbol{X}. \tag{12.56}$$

The last line above allows us to observe that the term in the square brackets results in the identity matrix $\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top = \boldsymbol{I}$. Since the multiplication results in an indentity matrix, we conclude that

$$(\boldsymbol{X}\boldsymbol{X}^\top)^{-1} = \boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top, \tag{12.57}$$

as is needed to show Equation (12.49). ♢

Remark. Note that the expression in (12.49) is derived for the closed-form solution of maximum likelihood regression. In turns out that a more general argument can be made akin to the representer theorem (Rifkin and Lippert, 2007; Rasmussen and Williams, 2006). In general, the regularization term can be expressed as in (12.49). ♢

Recall also the facts we derived in Section 7.3.3, where we found the following convex conjugate pairs:

$$\mathcal{L}(\boldsymbol{t}) = \sum_{n=1}^{N} \ell_n(t_n) \quad \text{has conjugate} \quad \mathcal{L}^*(\boldsymbol{z}) = \sum_{n=1}^{N} \ell_n^*(z_n) \tag{12.58}$$

and

$$f(\boldsymbol{y}) = \frac{\lambda}{2}\boldsymbol{y}^\top\boldsymbol{K}^{-1}\boldsymbol{y} \quad \text{has conjugate} \quad f^*(\boldsymbol{\alpha}) = \frac{1}{2\lambda}\boldsymbol{\alpha}^\top\boldsymbol{K}\boldsymbol{\alpha}. \tag{12.59}$$

Using the conjugate of the regularizer (12.59), we obtain the quadratic term in the dual SVM (Equation (12.42)).

The definition of convex conjugate in Section 7.3.3 and the hinge loss in (12.30) yield

$$\ell^*(u) = \sup_{t\in\mathbb{R}} tu - \max(0, 1-t) = \begin{cases} u & \text{if} -1 \leqslant u \leqslant 0 \\ \infty & \text{otherwise} \end{cases}. \tag{12.60}$$

The best way to see that (12.60) is correct is to look at the hinge loss in Figure 12.9 and imagine linear functions that are tangent to it. Consider the value of the slope of these lines that just touch the loss function and are below it, and observe that the slopes can only be between -1 and 0. All other values are not possible.

Using the conjugate of the hinge loss (12.60) and the property on a sum

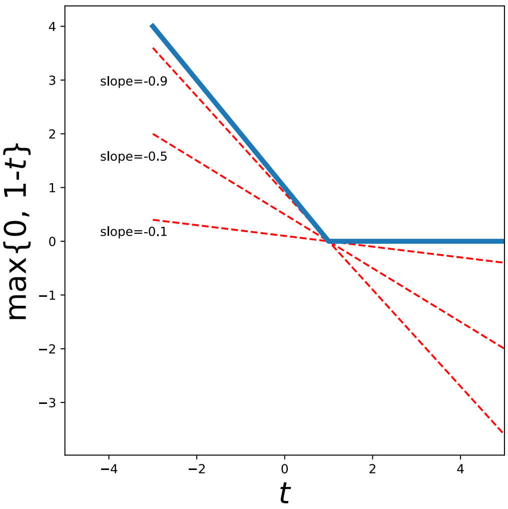

Figure 12.9 Hinge loss with tangents to it

over functions (12.58), we obtain the linear term in the objective of the dual SVM (12.42) and the box constraints. Note that the box constraints were previously scaled by C but we now scale the regularizer by λ. Note that is also a change in sign due to the convention used in (12.45).

In summary, we have derived the dual SVM (12.42) using the Legendre-Fenchel transform. The two derivations, based on Lagrange multipliers in Section 12.3.1 and on convex conjugates in this section, result in the same dual SVM. In general, it turns out that the two concepts of duality are the same for linear constraints. This was discussed more generally in the example at the end of Section 7.3.3.

12.3.3 Soft Margin SVM: Convex Hull View

Another approach to obtain the SVM is to consider an alternative geometrical argument. Consider the set of points $\boldsymbol{x}_n$ with the same label. We would like to build a convex boundary around this set of points that is the smallest possible. This is called the convex hull and is illustrated in Figure 12.10.

Building a convex boundary of points (called the *convex hull*) can be done by introducing non-negative weights $\alpha_i \geqslant 0$ corresponding to each data point $\boldsymbol{x}_n$. Then the convex hull can be described as the set

convex hull

$$H(\boldsymbol{X}) = \left\{ \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n \right\} \tag{12.61}$$

$$\text{with} \quad \sum_{n=1}^{N} \alpha_n = 1$$

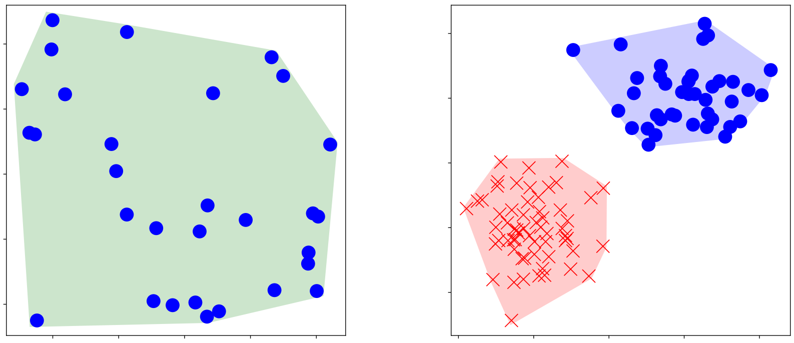

Figure 12.10 (left) Convex hull of points (right) A convex hull around positive and negative examples

$$\text{and} \quad \alpha_n \geqslant 0 \quad \text{for all} \quad n = 1, \ldots, N\,.$$

If the two clouds of points corresponding to the positive and negative classes are well separated, then we expect that the convex hulls do not overlap. We consider the linearly separable case here. The non-separable case can analogously be derived with a bit more care by reducing the size of the convex hull (Bennett and Bredensteiner, 2000). Given the training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ we form two convex hulls, corresponding to the positive and negative classes respectively.

We pick a point $\boldsymbol{c}$, which is in the convex hull of the set of positive points, and is closest to the negative class distribution. Similarly we pick a point $\boldsymbol{d}$ in the convex hull of the set of negative points, and is closest to the positive class. We draw a vector from $\boldsymbol{d}$ to $\boldsymbol{c}$

$$\boldsymbol{w} = \boldsymbol{c} - \boldsymbol{d}\,. \tag{12.62}$$

Picking the points $\boldsymbol{c}$ and $\boldsymbol{d}$ as above, and requiring them to be closest to each other is the same as saying that we want to minimize the length/ norm of $\boldsymbol{w}$, such that we end up with the corresponding optimization problem

$$\min_{\boldsymbol{w}} \|\boldsymbol{w}\| = \min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2\,. \tag{12.63}$$

Since $\boldsymbol{c}$ must be in the positive convex hull, it can be expressed as a convex combination of the positive points, i.e., for non-negative coefficients α_n^+

$$\boldsymbol{c} = \sum_{y_n=+1} \alpha_n^+ \boldsymbol{x}_n\,. \tag{12.64}$$

Recall that α_n are non-negative by definition. Similarly, for the examples with negative labels we obtain

$$\boldsymbol{d} = \sum_{y_n=-1} \alpha_n^- \boldsymbol{x}_n\,. \tag{12.65}$$

Let $\boldsymbol{\alpha}$ be the set of all coefficients, i.e., the concatenation of $\boldsymbol{\alpha}^+$ and $\boldsymbol{\alpha}^-$. Recall that we require that for each convex hull that

$$\sum_{y_n=+1} \alpha_n^+ = 1 \quad \text{and} \quad \sum_{y_n=-1} \alpha_n^- = 1\,. \tag{12.66}$$

The summations in the equation above are over the set of data points corresponding to the positive and negative class respectively. This can be reduced to the constraint

$$\sum_{n=1}^{N} y_n \alpha_n = 0 \tag{12.67}$$

by multiplying out the individual classes

$$\begin{aligned}\sum_{n=1}^{N} y_n \alpha_n &= \sum_{y_n=+1} (+1)\alpha_n^+ + \sum_{y_n=-1} (-1)\alpha_n^- \\ &= \sum_{y_n=+1} \alpha_n^+ - \sum_{y_n=-1} \alpha_n^- = 1 - 1 = 0\,.\end{aligned} \tag{12.68}$$

This optimization problem is the same as that of the dual hard margin SVM. To obtain the soft margin dual, we consider the reduced hull. The *reduced hull* is similar to the convex hull but has an upper bound to the size of the coefficients $\boldsymbol{\alpha}$. The maximum possible value of the elements of $\boldsymbol{\alpha}$ restricts the size that the convex hull can take. In other words, the bound on $\boldsymbol{\alpha}$ shrinks the convex hull to a smaller volume, and the reduced hull is given by

reduced hull

$$RH(X) = \left\{ \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n \right\} \tag{12.69}$$

$$\text{with} \sum_{n=1}^{N} \alpha_n = 1 \quad \text{and} \qquad 0 \leqslant \alpha_n \leqslant C \quad \text{for all} \quad n = 1, \ldots, N\,.$$

By performing the same reasoning as for the previous case, we obtain the dual SVM.

12.3.4 Kernels

Consider the formulation of the dual SVM (12.42). Notice that the inner product in the objective occurs only between examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. There are no inner products between the examples and the parameters. Therefore if we consider a set of features $\boldsymbol{\phi}(\boldsymbol{x}_i)$ to represent $\boldsymbol{x}_i$, the only change in the dual SVM will be to replace the inner product. This modularity, where the choice of the classification method (the SVM) and the choice of the feature representation $\boldsymbol{\phi}(\boldsymbol{x})$ can be considered separately, provides flexibility for us to explore the two problems independently.

Since $\boldsymbol{\phi}(\boldsymbol{x})$ could be a non-linear function, we can use the SVM (which assumes a linear classifier) to construct nonlinear classifiers. This provides a second avenue, in addition to the soft margin, for users to deal with a dataset that is not linearly separable. It turns out that there are many algorithms and statistical methods, which have this property that we observed in the dual SVM: the only inner products are those that occur between examples. This is known as the *kernel trick* (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004), by defining a kernel function

kernel trick

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle \,. \tag{12.70}$$

There is a one-to-one mapping between the kernel function $k(\cdot,\cdot)$ and the feature map $\boldsymbol{\phi}(\cdot)$, therefore, designing one implies a choice in the other. The matrix $\boldsymbol{K} \in \mathbb{R}^{N\times N}$, resulting from the inner products or the application of $k(\cdot,\cdot)$ to a dataset, is called the *Gram matrix*, and is often just referred to as the *kernel matrix*. Kernels must be symmetric and positive semi-definite, i.e., every kernel matrix $\boldsymbol{K}$ must be symmetric and positive semi-definite (Section 3.2.3):

Gram matrix

kernel matrix

$$\forall \boldsymbol{z} \in \mathbb{R}^N \qquad \boldsymbol{z}^\top \boldsymbol{K} \boldsymbol{z} \geqslant 0 \,. \tag{12.71}$$

Some popular examples of kernels for multivariate real-valued data $\boldsymbol{x}_i \in \mathbb{R}^D$ are the polynomial kernel, the Gaussian radial basis function kernel, and the rational quadratic kernel. Figure 12.11 illustrates the effect of different kernels on separating hyperplanes on an example dataset.

Remark. Unfortunately for the fledgling machine learner, there are multiple meanings of the word kernel. In this chapter, the word kernel comes from the idea of the Reproducing Kernel Hilbert Space (RKHS) (Aronszajn, 1950; Saitoh, 1988). We have discussed the idea of the kernel in linear algebra (Section 2.7.3), where the kernel is the same as the nullspace. The third common use of the word kernel in machine learning is in kernel density estimation. ♢

Since the explicit representation $\boldsymbol{\phi}(\boldsymbol{x})$ is mathematically equivalent to the kernel representation $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ a practitioner will often design the kernel function, such that it can be computed more efficiently than the inner product between explicit feature maps. For example, consider the polynomial kernel, where the number of terms in the explicit expansion grows very quickly (even for polynomials of low degree) when the input dimension is large. The kernel function only requires one multiplication per input dimension, which can provide significant computational savings.

Another useful aspect of the kernel trick is that there is no need for the original data to be already represented as multivariate real-valued data. Note that the inner product is defined on the output of the function $\boldsymbol{\phi}(\cdot)$, but does not restrict the input to real numbers. Hence, the function $\boldsymbol{\phi}(\cdot)$ and the kernel function $k(\cdot,\cdot)$ can be defined on any object, e.g., sets, sequences, strings and graphs (Ben-Hur et al., 2008; Gärtner, 2008; Shi et al., 2009; Vishwanathan et al., 2010).

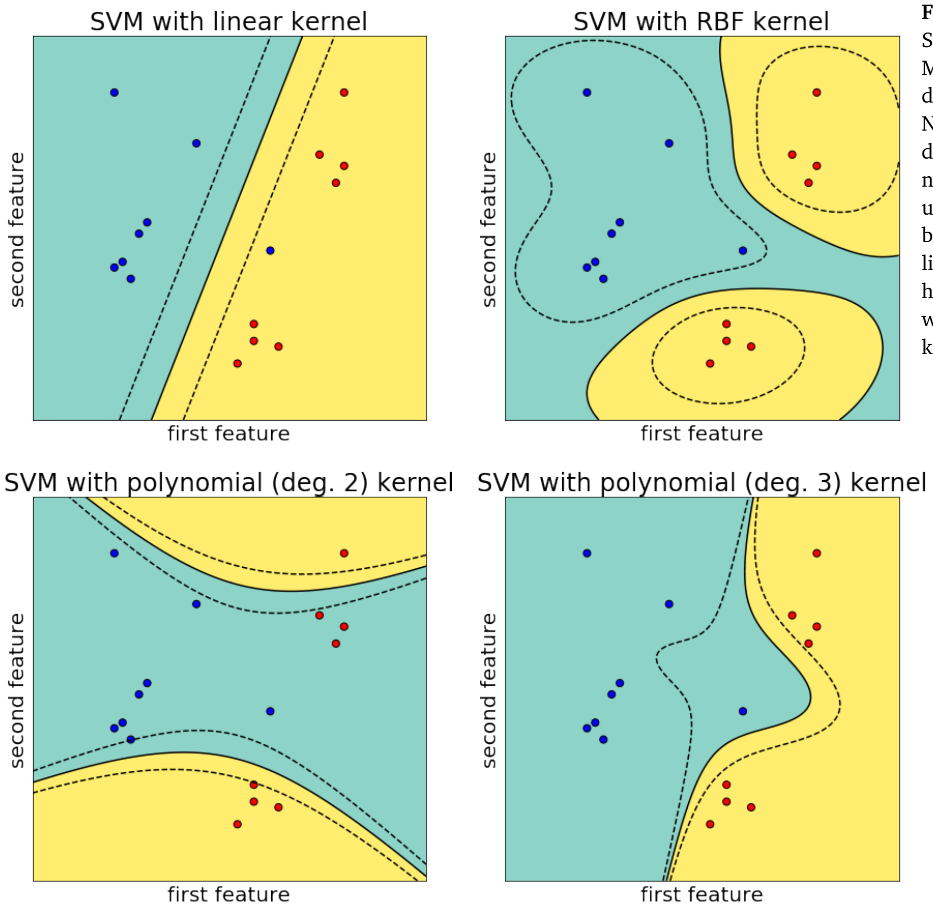

Figure 12.11 Support Vector Machine with different kernels. Note that while the decision boundary is nonlinear, the underlying problem being solved is for a linear separating hyperplane (albeit with a nonlinear kernel).

12.3.5 Numerical Solution

We consider two different approaches for finding the optimal solution for the SVM: constrained and unconstrained optimization.

Consider the loss function view of the SVM (12.32). This is a convex unconstrained optimization problem, but the hinge loss is not differentiable at one single point. Therefore, we apply a subgradient approach for solving it. Consider the hinge loss (12.30), which is the only non differentiable part of the SVM. However, it is differentiable almost everywhere, except for one single point at the hinge $t = 1$. At this point, the gradient is a set of possible values that lie between 0 and -1. Therefore, the subgradient g of the hinge loss is given by

$$g(t) = \begin{cases} -1 & t < 1 \\ [-1, 0] & t = 1 \\ 0 & t > 1 \end{cases} . \tag{12.72}$$

Using this subgradient above, we can apply the optimization methods presented in Section 7.1.

Both the primal and the dual SVM result in a convex quadratic programming problem (constrained optimization). Note that the primal SVM in (12.27) has optimization variables that have the size of the dimension D of the input examples. The dual SVM in (12.42) has optimization variables that have the size of the number N of data points.

To express the primal SVM in the standard form (7.35) for quadratic programming, let us assume that we use the dot product (3.6) as the inner product. We rearrange the equation for the primal SVM (12.27), such that the optimization variables are all on the right and the inequality of the constraint matches the standard form. This yields the optimization

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n \tag{12.73}$$

$$\text{subject to} \quad \begin{aligned} &-y_n\boldsymbol{x}_n^\top\boldsymbol{w} - y_n b - \xi_n \leqslant -1 \\ &-\xi_n \leqslant 0 \end{aligned} \tag{12.74}$$

for all $n = 1, \dots, N$. By concatenating the variables $\boldsymbol{w}, b, \boldsymbol{x}_n$ into one single vector, and carefully collecting the terms, we obtain the following matrix form of the soft margin SVM, where the minimization is over $[\boldsymbol{w}^\top, b, \boldsymbol{\xi}^\top]^\top \in \mathbb{R}^{D+1+N}$:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\begin{bmatrix}\boldsymbol{w}\\ b\\ \boldsymbol{\xi}\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{I}_D & \boldsymbol{0}_{D,N+1}\\ \boldsymbol{0}_{N+1,D} & \boldsymbol{0}_{N+1,N+1}\end{bmatrix}\begin{bmatrix}\boldsymbol{w}\\ b\\ \boldsymbol{\xi}\end{bmatrix} + \begin{bmatrix}\boldsymbol{0}_{D+1,1} & C\boldsymbol{1}_{N,1}\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{w}\\ b\\ \boldsymbol{\xi}\end{bmatrix} \tag{12.75}$$

$$\text{subject to} \quad \begin{bmatrix}\boldsymbol{Y}\boldsymbol{X} & \boldsymbol{y} & -\boldsymbol{I}_N\\ \boldsymbol{0}_{N,D+1} & & -\boldsymbol{I}_N\end{bmatrix}\begin{bmatrix}\boldsymbol{w}\\ b\\ \boldsymbol{\xi}\end{bmatrix} \leqslant \begin{bmatrix}-\boldsymbol{1}_{N,1}\\ \boldsymbol{0}_{N,1}\end{bmatrix}, \tag{12.76}$$

where $\boldsymbol{y}$ is the vector of labels $[y_1, \dots, y_N]^\top$, $\boldsymbol{Y} = \operatorname{diag}(\boldsymbol{y})$ is an N by N matrix where the elements of the diagonal are from $\boldsymbol{y}$, and $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ is the matrix obtained by concatenating all the examples.

We can similarly perform a collection of terms for the dual version of the SVM (12.42). To express the dual SVM in standard form, we first have to express the kernel matrix $\boldsymbol{K}$ such that each entry is $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Or if we are using an explicit feature representation $K_{ij} = \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle$. For convenience of notation we introduce a matrix with zeros everywhere except on the diagonal, where we store the labels, that is $\boldsymbol{Y} = \operatorname{diag}(\boldsymbol{y})$. The dual SVM can be written as

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha} + \boldsymbol{1}_{N,1}^\top\boldsymbol{\alpha} \tag{12.77}$$

$$\text{subject to} \begin{bmatrix} \boldsymbol{y}^\top \\ -\boldsymbol{y}^\top \\ -\boldsymbol{I}_N \\ \boldsymbol{I}_N \end{bmatrix} \leqslant \begin{bmatrix} \mathbf{0}_{N+2,1} \\ \mathbf{1}_{N,1} \end{bmatrix} . \tag{12.78}$$

Remark. In Section 7.3.1 and 7.3.2 we introduced the standard forms of the constraints to be inequality constraints. We will express the dual SVM's equality constraint as two inequality constraints, i.e.,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad \text{is replaced by} \quad \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b} \quad \text{and} \quad \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b} \tag{12.79}$$

Particular software implementations of convex optimization methods may provide the ability to express equality constraints. ♢

Since there are many different possible views of the SVM, there are many approaches for solving the resulting optimization problem. The approach presented here, expressing the SVM problem in standard convex optimization form, is not often used in practice. The two main implementations of SVM solvers are (Chang and Lin, 2011) (which is open source) and (Joachims, 1999). Since SVMs have a clear and well defined optimization problem, many approaches based on numerical optimization techniques (Nocedal and Wright, 2006) can be applied (Shawe-Taylor and Sun, 2011).

12.4 Further Reading

The SVM is one of many approaches for studying binary classification. Other approaches include the perceptron, logistic regression, Fisher discriminant, nearest neighbor, naive Bayes, and random forest (Bishop, 2006; Murphy, 2012). A short tutorial on SVMs and kernels on discrete sequences can be found in Ben-Hur et al. (2008). The book about kernel methods (Schölkopf and Smola, 2002) includes many details of support vector machines and how to optimize them. A broader book about kernel methods (Shawe-Taylor and Cristianini, 2004) also includes many linear algebra approaches for different machine learning problems. Readers interested in the functional analysis view (also the regularization methods view) of SVMs are referred to the work by Wahba (1990). Theoretical exposition of kernels (Manton and Amblard, 2015; Aronszajn, 1950; Schwartz, 1964; Saitoh, 1988) require a basic grounding of linear operators (Akhiezer and Glazman, 1993).

Since binary classification is a well studied task in machine learning, other words are also sometimes used, such as discrimination, separation or decision. To further add to the confusion, there are three quantities that can be the output of a binary classifier. First is the output of the linear function itself. This output can be used for ranking the examples, and binary classification can be thought of as picking a threshold on the ranked examples (Shawe-Taylor and Cristianini, 2004). The second quantity that is often considered the output of a binary classifier is after the output is passed through a non-linear function to constrain its value to a bounded range. A common non-linear function is the sigmoid function (Bishop, 2006). When the non-linearity results in well calibrated probabilities (Gneiting and Raftery, 2007; Reid and Williamson, 2011), this is called class probability estimation. The third output of a binary classifier is the final binary decision, which is the one most commonly assumed to be the output of the classifier.