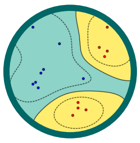
12

Classification with Support Vector Machines



In many situations, we want our machine learning algorithm to predict one of a number of (discrete) 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, i.e., there are only two possible outcomes. This machine learning task is called *binary classification*. This is in contrast to Chapter 9 where we considered a prediction problem with continuous-valued outputs.

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

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 by $\{+1, -1\}$. In other words, we consider predictors of the form

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

Recall from Chapter 8 that we represent each example (data point) $\boldsymbol{x}_n$ 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 12.6.

Input example $\boldsymbol{x}_n$ may also be referred to as inputs, data points, features or instances.

classes

For probabilistic models, it is mathematically convenient to use $\{0, 1\}$ as a binary representation, see remark after Example 6.12.

We present an approach known as the Support Vector Machine (SVM), which solves the binary classification task. As in regression, we have a supervised learning task, where we have a set of examples $\boldsymbol{x}_n \in \mathbb{R}^D$ along with their corresponding (binary) labels $y_n \in \{+1, -1\}$. Given a training data set consisting of example-label pairs $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, we would like to estimate parameters of the model that will give the smallest classification error. Similar to Chapter 9 we consider a linear model, and hide away the nonlinearity in a transformation ϕ of the examples (9.13). We will revisit ϕ in Section 12.4.

The SVM provides state-of-the-art results in many applications, with sound theoretical guarantees (Steinwart and Christmann, 2008). There are two main reasons why we chose to illustrate binary classification using

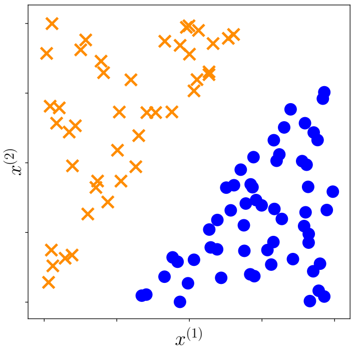


Figure 12.1 Example 2D data, illustrating the intuition of data where we can find a linear classifier that separates red crosses from blue dots.

SVMs. First, the SVM allows for a geometric way to think about supervised machine learning. While in Chapter 9 we considered the machine learning problem in terms of probabilistic models and 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. The second reason why we find SVMs instructive is that in contrast to Chapter 9, the optimization problem for SVM does not admit an analytic solution so that we need to resort to a variety of optimization tools introduced in Chapter 7.

The SVM view of machine learning is 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. We have seen something similar already in Chapter 10 where we derived PCA from geometric principles. In the SVM case, we start by designing an objective function that is to be minimized on training data, following the principles of empirical risk minimization 8.1. This can also be understood as designing a particular loss function.

Let us derive the optimization problem corresponding to training an SVM on example-label pairs. Intuitively, we imagine binary classification data, which can be separated by a hyperplane as illustrated in Figure 12.1. Here, every example $\boldsymbol{x}_n$ (a vector of dimension 2) is a two-dimensional location ($x_n^{(1)}$ and $x_n^{(2)}$), and the corresponding binary label y_n is one of two different symbols (red cross or blue disc). “Hyperplane” is a word that is commonly used in machine learning, and we encountered hyperplanes already in Section 2.8. A hyperplane is an affine subspace of dimension $D-1$ (if the corresponding vector space is of dimension D). The examples consist of two classes (there are two possible labels) that have features (the components of the vector representing the example) arranged in such a way as to allow us to separate/classify them by drawing a straight line.

In the following, we formalize the idea of finding a linear separator of the two classes. We introduce the idea of the margin and then extend linear separators to allow for examples to fall on the "wrong" side, incurring a classification error. We present 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 using Lagrange multipliers (Section 7.2). The dual SVM allows us to observe a third way of formalizing the SVM: in terms of the convex hulls of the examples of each class (Section 12.3.2). We conclude by briefly describing kernels and how to numerically solve the nonlinear kernel-SVM optimization problem.

12.1 Separating Hyperplanes

Given two examples 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 are closely related to the angle between two vectors. The value of the inner product between two vectors depends on the length (norm) of each vector. Furthermore, inner products allow us to rigorously define geometric 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, ideally in a way that examples with the same label are in the same partition (and no other examples). In the case of binary classification, the space would be divided into two parts corresponding to the positive and negative classes, respectively. We consider a particularly convenient partition, which is to (linearly) split the space into two halves using a hyperplane. Let example $\boldsymbol{x} \in \mathbb{R}^D$ be an element of the data space. Consider a function

$$f : \mathbb{R}^D \to \mathbb{R} \tag{12.2a}$$

$$\boldsymbol{x} \mapsto \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \,, \tag{12.2b}$$

parametrized by $\boldsymbol{w} \in \mathbb{R}^D$ and $b \in \mathbb{R}$. Recall from Section 2.8 that hyperplanes are affine subspaces. Therefore, we define the hyperplane that separates the two classes in our binary classification problem as

$$\left\{ \boldsymbol{x} \in \mathbb{R}^D : f(\boldsymbol{x}) = 0 \right\} . \tag{12.3}$$

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 a normal vector to the hyperplane in (12.3) by choosing any two examples $\boldsymbol{x}_a$ 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,

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

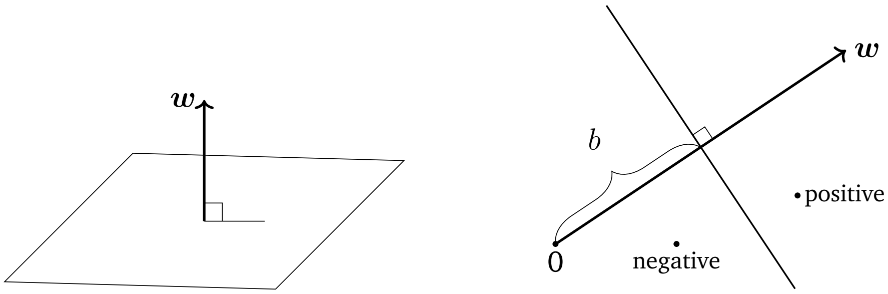


Figure 12.2 Equation of a separating hyperplane (12.3). (left) The standard way of representing the equation in 3D. (right) For ease of drawing, we look at the hyperplane edge on.

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.

$\boldsymbol{w}$ is orthogonal to any vector on the hyperplane.

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

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

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

$$\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b \geqslant 0 \quad \text{when} \quad y_n = +1 \tag{12.5}$$

and the examples with negative labels are on the negative side, i.e.,

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

Refer to Figure 12.2 for a geometric intuition of positive and negative examples. These two conditions are often presented in a single equation

$$y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 0\,. \tag{12.7}$$

Equation (12.7) is equivalent to (12.5) and (12.6) when we multiply both sides of (12.5) and (12.6) with $y_n = 1$ and $y_n = -1$, respectively.

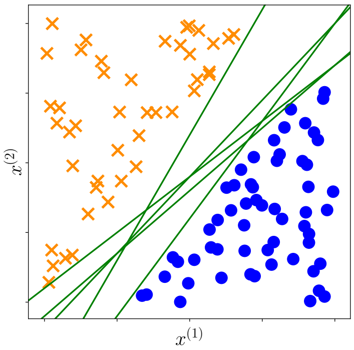


Figure 12.3 Possible separating hyperplanes. There are many linear classifiers (green lines) that separate red crosses from blue dots.

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 infinitely many candidate hyperplanes (refer to Figure 12.3), and therefore classifiers, that solve our classification problem without any (training) errors. To find a unique solution, one idea is to choose the separating hyperplane that maximizes the margin between the positive and negative examples. In other words, we want the positive and negative examples to be separated by a large margin (Section 12.2.1). In the following, we compute the distance between an example and a hyperplane to derive the margin. Recall that the closest point on the hyperplane to a given point (example $\boldsymbol{x}_n$) is obtained by the orthogonal projection (Section 3.8).

A classifier with large margin turns out to generalize well (Steinwart and Christmann, 2008).

12.2.1 Concept of the Margin

margin

There could be two or more closest examples to a hyperplane.

The concept of the *margin* is intuitively simple: It is the distance of the separating hyperplane to the closest examples in the dataset, assuming that the dataset is linearly separable. However, when trying to formalize this distance, there is a technical wrinkle that may be 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.

Consider a hyperplane $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$, and an example $\boldsymbol{x}_a$ as illustrated in Figure 12.4. Without loss of generality, we can consider the example $\boldsymbol{x}_a$ to be on the positive side of the hyperplane, i.e., $\langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b > 0$. We would like to compute the distance $r > 0$ of $\boldsymbol{x}_a$ from the hyperplane. We do so by considering the orthogonal projection (Section 3.8) of $\boldsymbol{x}_a$ onto the hyperplane, which we denote by $\boldsymbol{x}'_a$. Since $\boldsymbol{w}$ is orthogonal to the

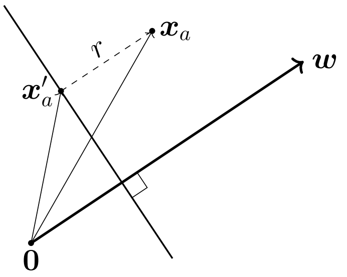


Figure 12.4 Vector addition to express distance to hyperplane: $\boldsymbol{x}_a = \boldsymbol{x}'_a + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$.

hyperplane, we know that the distance r is just a scaling of this vector $\boldsymbol{w}$. If the length of $\boldsymbol{w}$ is known, then we can use this scaling factor r factor to work out the absolute distance between $\boldsymbol{x}_a$ and $\boldsymbol{x}'_a$. For convenience we choose to use a vector of unit length (its norm is 1), and obtain this by dividing $\boldsymbol{w}$ by its norm, $\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$. Using vector addition (Section 2.4) we obtain

$$\boldsymbol{x}_a = \boldsymbol{x}'_a + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \,. \tag{12.8}$$

Another way of thinking about r is that it is the coordinate of $\boldsymbol{x}_a$ in the subspace spanned by $\boldsymbol{w}/\left\|\boldsymbol{w}\right\|$. We have now expressed the distance of $\boldsymbol{x}_a$ from the hyperplane as r, and if we choose $\boldsymbol{x}_a$ to be the point closest to the hyperplane, this distance r is the margin.

Recall that we would like the positive examples to be further than r from the hyperplane, and the negative examples to be further than distance r (in the negative direction) from the hyperplane. Analogously to the combination of (12.5) and (12.6) into (12.7), we formulate this objective as

$$y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant r \,. \tag{12.9}$$

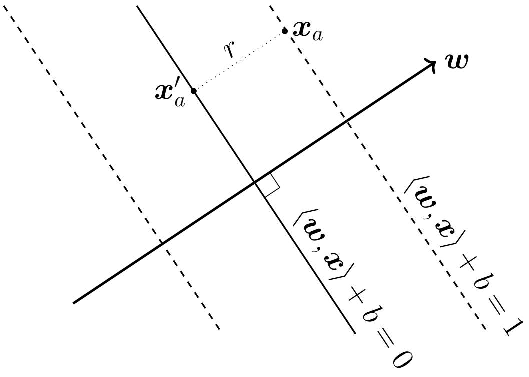
In other words, we combine the requirements that examples are at least r away from the hyperplane (in the positive and negative direction) into one single inequality.

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, i.e., $\|\boldsymbol{w}\| = 1$, where we use the Euclidean norm $\|\boldsymbol{w}\| = \sqrt{\boldsymbol{w}^\top \boldsymbol{w}}$ (Section 3.1). This assumption also allows a more intuitive interpretation of the distance r (12.8) since it is the scaling factor of a vector of length 1.

We will see other choices of inner products (Section 3.2) in Section 12.4.

Remark. A reader familiar with other presentations of the margin would notice that our definition of $\|\boldsymbol{w}\| = 1$ is different from the standard presentation if the SVM provided by Schölkopf and Smola (2002), for example. In Section 12.2.3, we will show the equivalence of both approaches. ♢

Collecting the three requirements into a single constrained optimization



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

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

problem, we obtain the objective

$$\begin{aligned} &\max_{\boldsymbol{w},b,r} \quad \underbrace{r}_{\text{margin}} \\ &\text{subject to} \quad \underbrace{y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant r}_{\text{data fitting}}, \ \underbrace{\|\boldsymbol{w}\| = 1}_{\text{normalization}}, \quad r > 0\,, \end{aligned} \tag{12.10}$$

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

Remark. The concept 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; Steinwart and Christmann, 2008). ♢

12.2.2 Traditional Derivation of the Margin

In the previous section, we derived (12.10) 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 example. Let us also denote the example in the dataset that is closest to the hyperplane by $\boldsymbol{x}_a$.

Recall that we currently consider linearly separable data.

Figure 12.5 is identical to Figure 12.4, except that now we rescaled the axes, such that the example $\boldsymbol{x}_a$ lies 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.,

$$\langle \boldsymbol{w}, \boldsymbol{x}'_a \rangle + b = 0\,. \tag{12.11}$$

By substituting (12.8) into (12.11) we obtain

$$\left\langle \boldsymbol{w}, \boldsymbol{x}_a - r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \right\rangle + b = 0\,. \tag{12.12}$$

Exploiting the bilinearity of the inner product (see Section 3.2), we get

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

Observe that the first term is 1 by our assumption of scale, i.e., $\langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b = 1$. From (3.16) in Section 3.1 we know that $\langle \boldsymbol{w}, \boldsymbol{w} \rangle = \|\boldsymbol{w}\|^2$. Hence, the second term reduces to $r\|\boldsymbol{w}\|$. Using these simplifications, we obtain

$$r = \frac{1}{\|\boldsymbol{w}\|}\,. \tag{12.14}$$

This means we derived the distance r in terms of the normal vector $\boldsymbol{w}$ of the hyperplane. 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. One way to think about it is to consider the distance r to be a temporary variable that we only use for this derivation. Therefore, for the rest of this section we will denote the distance to the hyperplane by $\frac{1}{\|\boldsymbol{w}\|}$. In Section 12.2.3, we will see that the choice that the margin equals 1 is equivalent to our previous assumption of $\|\boldsymbol{w}\| = 1$ in Section 12.2.1.

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 (12.9), we want the positive and negative examples to be at least 1 away from the hyperplane, which yields the condition

$$y_n(\langle \boldsymbol{w}, x_n \rangle + b) \geqslant 1\,. \tag{12.15}$$

Combining the margin maximization with the fact that examples need to be on the correct side of the hyperplane (based on their labels) gives us

$$\max_{\boldsymbol{w},b} \quad \frac{1}{\|\boldsymbol{w}\|} \tag{12.16}$$

$$\text{subject to } y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1 \quad \text{for all} \quad n = 1, \ldots, N. \tag{12.17}$$

Instead of maximizing the reciprocal of the norm as in (12.16), 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 compute the gradient. Then, our objective becomes

The squared norm results in a convex quadratic programming problem for the SVM (Section 12.5).

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{12.18}$$

$$\text{subject to } y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1 \quad \text{for all} \quad n = 1, \ldots, N\,. \tag{12.19}$$

Equation (12.18) 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 margin SVM

"hard" condition can be relaxed to accommodate violations if the data is not linearly separable.

12.2.3 Why we can set the Margin to 1

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

Theorem 12.1. *Maximizing the margin r, where we consider normalized weights as in* (12.10),

$$\begin{aligned} &\max_{\boldsymbol{w},b,r} \quad \underbrace{r}_{\text{margin}} \\ &\text{subject to} \quad \underbrace{y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant r}_{\text{data fitting}}, \underbrace{\|\boldsymbol{w}\| = 1}_{\text{normalization}}, \quad r > 0, \end{aligned} \tag{12.20}$$

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

$$\begin{aligned} &\min_{\boldsymbol{w},b} \quad \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{margin}} \\ &\text{subject to} \quad \underbrace{y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1}_{\text{data fitting}}. \end{aligned} \tag{12.21}$$

Proof Consider (12.20). Since the square is a strictly 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 $\frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|}$. We obtain

$$\begin{aligned} &\max_{\boldsymbol{w}',b,r} \quad r^2 \\ &\text{subject to} \quad y_n\left(\left\langle \frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|}, \boldsymbol{x}_n \right\rangle + b\right) \geqslant r, \quad r > 0. \end{aligned} \tag{12.22}$$

Equation (12.22) explicitly states that the distance r is positive. Therefore, we can divide the first constraint by r, which yields

Note that $r > 0$ because we assumed linear separability, and hence there is no issue to divide by r.

$$\begin{aligned} &\max_{\boldsymbol{w}',b,r} \quad r^2 \\ &\text{subject to} \quad y_n\left(\left\langle \underbrace{\frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|\, r}}_{\boldsymbol{w}''}, \boldsymbol{x}_n \right\rangle + \underbrace{\frac{b}{r}}_{b''}\right) \geqslant 1, \quad r > 0 \end{aligned} \tag{12.23}$$

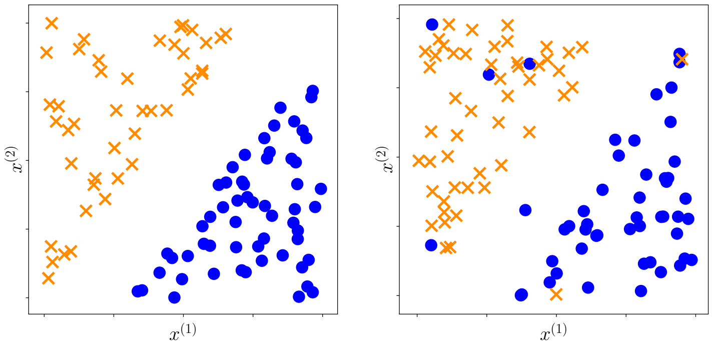


Figure 12.6 (left) linearly separable data, with a large margin. (right) non-separable data.

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

$$\|\boldsymbol{w}''\| = \left\| \frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\| \, r} \right\| = \frac{1}{r} \cdot \left\| \frac{\boldsymbol{w}'}{\|\boldsymbol{w}'\|} \right\| = \frac{1}{r} \,. \qquad (12.24)$$

By substituting this result into (12.23) we obtain

$$\max_{\boldsymbol{w}'', b''} \quad \frac{1}{\|\boldsymbol{w}''\|^2} \quad \text{subject to} \quad y_n \left(\langle \boldsymbol{w}'', \boldsymbol{x}_n \rangle + b'' \right) \geqslant 1 \,. \qquad (12.25)$$

The final step is to observe that maximizing $\frac{1}{\|\boldsymbol{w}''\|^2}$ yields the same solution as minimizing $\frac{1}{2} \|\boldsymbol{w}''\|^2$, which concludes the proof of Theorem 12.1. □

12.2.4 Soft Margin SVM: Geometric View

In the case where data is not linearly separable, we may wish to allow some examples to fall within the margin region, or even to be on the wrong side of the hyperplane as illustrated in Figure 12.6.

The model that allows for some classification errors 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 an equivalent 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 examples (Section 12.3.2).

soft margin SVM

The key geometric idea is to introduce a *slack variable* ξ_n corresponding to each example-label pair $(\boldsymbol{x}_n, y_n)$ 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 non-negative. To encourage correct classification of the samples

slack variable

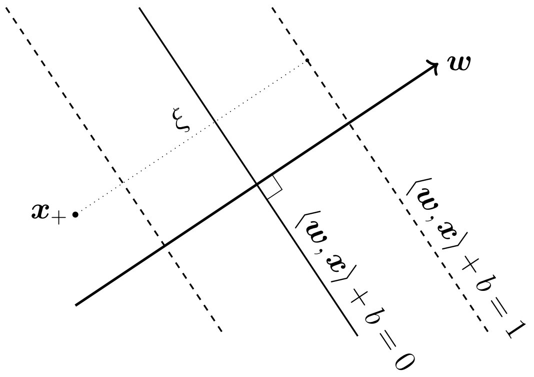

$\boldsymbol{w}$
ξ
$\boldsymbol{x}_+$
$\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$
$\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 1$

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

we add ξ_n to the objective

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N} \xi_n \tag{12.26a}$$

$$\text{subject to } y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1 - \xi_n \tag{12.26b}$$

$$\xi_n \geqslant 0 \tag{12.26c}$$

soft margin SVM

regularization parameter

regularizer

for $n = 1, \ldots, N$. In contrast to the optimization problem (12.18) for the hard margin SVM, this one is called the *soft margin SVM*. The parameter $C > 0$ 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 (12.26a) 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 8.1.3). This is in contrast to our formulation in this section. Here a large value of C implies low regularization, as we give the slack variables larger weight, hence giving more priority to examples which do not lie on the correct side of the margin.

There are alternative parametrizations of this regularization, which is why (12.26a) is also often referred to as the C-SVM.

Remark. In the formulation of the soft margin SVM (12.26a) $\boldsymbol{w}$ is regularized, but b is not regularized. We can see this by observing that the regularization term does not contain b. The unregularized term b complicates theoretical analysis (Steinwart and Christmann, 2008, Chapter 1) and decreases computational efficiency (Fan et al., 2008). ♢

12.2.5 Soft Margin SVM: Loss Function View

Let us consider a different approach for deriving the SVM, following the principle of empirical risk minimization (Section 8.1). For the SVM we choose hyperplanes as the hypothesis class, that is

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b. \tag{12.27}$$

We will see in this section that the margin corresponds to the regularization term. The remaining question is: what is the *loss function*? 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/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 (9.10b) is not suitable for binary classification.

loss function

Remark. The ideal loss function between binary labels is to count the number of mismatches between the prediction and the label. This means that for a predictor f 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 combinatorial optimization problem for finding the best parameters $\boldsymbol{w}, b$. Combinatorial optimization problems (in contrast to continuous optimization problems discussed in Chapter 7) are in general more challenging to solve. ♢

zero-one loss

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 describes the error that is made on the training data. An equivalent way to derive (12.26a) is to use the *hinge loss*

hinge loss

$$\ell(t) = \max\{0, 1-t\} \quad \text{where} \quad t = yf(\boldsymbol{x}) = y(\langle \boldsymbol{w}, \boldsymbol{x}\rangle + b)\,. \tag{12.28}$$

If $f(\boldsymbol{x})$ is on the correct side (based on the corresponding label 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 too close to the hyperplane ($0 < t < 1$) the example $\boldsymbol{x}$ is within the margin, and the hinge loss returns a positive value. When the example 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 too close 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

$$\ell(t) = \begin{cases} 0 & \text{if} \quad t \geqslant 1 \\ 1-t & \text{if} \quad t < 1 \end{cases}, \tag{12.29}$$

as illustrated in Figure 12.8. The loss corresponding to the hard margin SVM 12.18 is defined as

$$\ell(t) = \begin{cases} 0 & \text{if} \quad t \geqslant 1 \\ \infty & \text{if} \quad t < 1 \end{cases}. \tag{12.30}$$

This loss can be interpreted as never allowing any examples inside the margin.

For a given training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ we seek to minimize

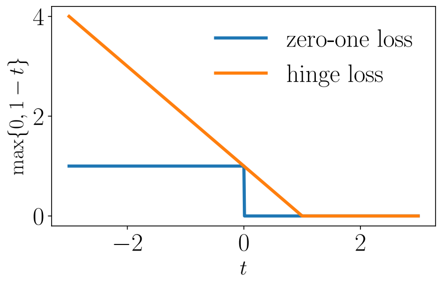


Figure 12.8 The hinge loss is a convex upper bound of zero-one loss.

the total loss, while regularizing the objective with ℓ_2-regularization (see Section 8.1.3). Using the hinge loss (12.28) gives us the unconstrained optimization problem

$$\min_{\boldsymbol{w},b} \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{regularizer}} + \underbrace{C\sum_{n=1}^{N}\max\{0, 1-y_n(\langle\boldsymbol{w},\boldsymbol{x}_n\rangle + b)\}}_{\text{error term}} . \tag{12.31}$$

regularizer

loss term

error term

regularization

The first term in (12.31) 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 $\frac{1}{2}\|\boldsymbol{w}\|^2$ arises directly from the margin. In other words, margin maximization can be interpreted as *regularization*.

In principle, the unconstrained optimization problem in (12.31) can be directly solved with (sub-)gradient descent methods as described in Section 7.1. To see that (12.31) and (12.26a) are equivalent, observe that the hinge loss (12.28) essentially consists of two linear parts, as expressed in (12.29). Consider the hinge loss on for a single example-label pair (12.28). We can equivalently replace minimization of the hinge loss over t with a minimization of a slack variable ξ with two constraints. In equation form,

$$\min_t \max\{0, 1-t\} \tag{12.32}$$

is equivalent to

$$\begin{aligned} &\min_{\xi,t} \quad \xi \\ &\text{subject to} \quad \xi \geqslant 0\,, \quad \xi \geqslant 1-t\,. \end{aligned} \tag{12.33}$$

By substituting this expression into (12.31) and rearranging one of the constraints, we obtain exactly the soft margin SVM (12.26a).

Remark. Let us contrast our choice of the loss function in this section to the loss function for linear regression in Chapter 9. Recall from Section 9.2.1 that for finding maximum likelihood estimators, we usually minimize 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 The squared error function is the loss function that is minimized when looking for the maximum likelihood solution. ♢

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 consider inputs $\boldsymbol{x} \in \mathbb{R}^D$ 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. Instead the number of parameters increases with the number of examples in the training set. We saw a similar idea appear in Chapter 10, where we expressed 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 the number of examples in the training dataset. 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, which we discussed in Section 7.2.

12.3.1 Convex Duality via Lagrange Multipliers

Recall the primal soft margin SVM (12.26a). We call the variables $\boldsymbol{w}$, b and ξ corresponding to the primal SVM the primal variables. We use $\alpha_n \geqslant 0$ as the Lagrange multiplier corresponding to the constraint (12.26b) that the examples are classified correctly and $\gamma_n \geqslant 0$ as the Lagrange multiplier corresponding to the non-negativity constraint of the slack variable, see (12.26c). The Lagrangian is then given by

In Chapter 7 we used λ as Lagrange multipliers. In this section we follow the notation commonly chosen in SVM literature, and use α and γ.

$$\begin{aligned}\mathfrak{L}(\boldsymbol{w}, b, \xi, \alpha, \gamma) &= \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n \qquad (12.34)\\ &\quad \underbrace{-\sum_{n=1}^{N}\alpha_n(y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b) - 1 + \xi_n)}_{\text{constraint (12.26b)}} \underbrace{-\sum_{n=1}^{N}\gamma_n\xi_n}_{\text{constraint (12.26c)}} \;.\end{aligned}$$

By differentiating the Lagrangian (12.34) with respect to the three primal variables $\boldsymbol{w}$, b and ξ respectively, we obtain

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{w}} = \boldsymbol{w}^\top - \sum_{n=1}^{N}\alpha_n y_n \boldsymbol{x}_n{}^\top , \qquad (12.35)$$

$$\frac{\partial \mathfrak{L}}{\partial b} = \sum_{n=1}^{N} \alpha_n y_n \,, \tag{12.36}$$

$$\frac{\partial \mathfrak{L}}{\partial \xi_n} = C - \alpha_n - \gamma_i \,. \tag{12.37}$$

We now find the maximum of the Lagrangian by setting each of these partial derivatives to zero. By setting (12.35) to zero we find

$$\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n \,, \tag{12.38}$$

representer theorem

which is a particular instance of the *representer theorem* (Kimeldorf and Wahba, 1970). Equation (12.38) states that the optimal weight vector in the primal is a linear combination of the examples $\boldsymbol{x}_n$. Recall from Section 2.6.1 that this means that the solution of the optimization problem lies in the span of training data. Additionally the constraint obtained by setting (12.36) to zero implies that the optimal weight vector is an affine combination of the examples. The representer theorem turns out to hold for very general settings of regularized empirical risk minimization (Hofmann et al., 2008; Argyriou and Dinuzzo, 2014). The theorem has more general versions (Schölkopf et al., 2001), and necessary and sufficient conditions on its existence can be found in (Yu et al., 2013).

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 examples.

Remark. The representer theorem (12.38) also provides an explaination of the name Support Vector Machine. The examples $\boldsymbol{x}_n$, for which the corresponding parameters $\alpha_n = 0$, do not contribute to the solution $\boldsymbol{w}$ at all. The other examples, where $\alpha_n > 0$, are called *support vectors* since they "support" the hyperplane. ◇

support vectors

By substituting the expression for $\boldsymbol{w}$ into the Lagrangian (12.34), we obtain the dual

$$\begin{aligned}\mathfrak{D}(\xi, \alpha, \gamma) = &\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - \sum_{i=1}^{N} y_i \alpha_i \left\langle \sum_{j=1}^{N} y_j \alpha_j \boldsymbol{x}_j, \boldsymbol{x}_i \right\rangle \\ &+ C\sum_{i=1}^{N} \xi_i - b\sum_{i=1}^{N} y_i \alpha_i + \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N} \alpha_i \xi_i - \sum_{i=1}^{N} \gamma_i \xi_i \,.\end{aligned} \tag{12.39}$$

Note that there are no longer any terms involving the primal variable $\boldsymbol{w}$. By setting (12.36) 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 bilinear (see Section 3.2). Therefore, the first two terms in (12.39) are over the same objects. These terms (colored blue) can be simplified, and we obtain the Lagrangian

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

The last term in this equation is a collection of all terms that contain slack variables ξ_i. By setting (12.37) to zero, we see that the last term in (12.39) is also zero. Furthermore, by using the same equation and recalling that the Lagrange multipliers γ_i are 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 multipliers α_i. Recall from Lagrangian duality (Definition 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

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - \sum_{i=1}^{N} \alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & 0 \leqslant \alpha_i \leqslant C \quad \text{for all} \quad i = 1, \ldots, N \,. \end{aligned} \tag{12.41}$$

The equality constraint in (12.41) is obtained from setting (12.36) to zero. The inequality constraint $\alpha_i \geqslant 0$ is the condition imposed on Lagrange multipliers of inequality constraints (Section 7.2). The inequality constraint $\alpha_i \leqslant C$ is discussed in the previous paragraph.

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).

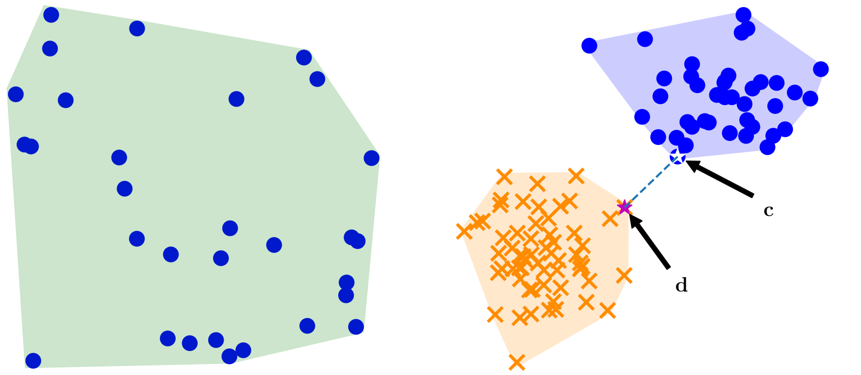
It turns out examples that lie exactly on the margin are examples whose dual parameters lie strictly inside the box constraints, $0 < \alpha_i < C$. This is derived using the Karush Kuhn Tucker conditions, for example in Schölkopf and Smola (2002).

Once we obtain the dual parameters $\boldsymbol{\alpha}$ we can recover the primal parameters $\boldsymbol{w}$ by using the representer theorem (12.38). Let us call the optimal primal parameter $\boldsymbol{w}^*$. However, there remains the question on how to obtain the parameter b^*. Consider an example $\boldsymbol{x}_n$ that lies exactly on the margin's boundary, i.e., $\langle \boldsymbol{w}^*, \boldsymbol{x}_n \rangle + b = y_n$. Recall that y_n is either $+1$ or -1. Therefore, the only unknown is b, which can be computed by

$$b^* = y_n - \langle \boldsymbol{w}^*, \boldsymbol{x}_n \rangle \,. \tag{12.42}$$

Remark. In principle, there may be no examples that lie exactly on the margin. In this case, we should compute $|y_n - \langle \boldsymbol{w}^*, \boldsymbol{x}_n \rangle|$ for all support vectors and take the median value of this absolute value difference to be the value of b^*. A derivation of this can be found in `http://fouryears.eu/2012/06/07/the-svm-bias-term-conspiracy/`. ◇

Figure 12.9 Convex hulls. (a) Convex hull of points, some of which lie within the boundary; (b) Convex hulls around positive and negative examples.



(a) Convex hull.

(b) Convex hulls around positive (blue) and negative (red) examples. The distance between the two convex sets is the length of the difference vector $\boldsymbol{c} - \boldsymbol{d}$.

12.3.2 Dual SVM: Convex Hull View

Another approach to obtain the dual SVM is to consider an alternative geometric argument. Consider the set of examples $\boldsymbol{x}_n$ with the same label. We would like to build a convex set that contains all the examples such that it is the smallest possible set. This is called the convex hull and is illustrated in Figure 12.9.

Let us first build some intuition about a convex combination of points. Consider two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and corresponding non-negative weights $\alpha_1, \alpha_2 \geqslant 0$ such that $\alpha_1 + \alpha_2 = 1$. The equation $\alpha_1\boldsymbol{x}_1 + \alpha_2\boldsymbol{x}_2$ describes each point on a line between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Consider what happens when we add a third point $\boldsymbol{x}_3$ along with a weight $\alpha_3 \geqslant 0$ such that $\sum_{n=1}^{3} \alpha_n = 1$. The convex combination of these three points $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ span a two-dimensional area. The *convex hull* of this area is the triangle formed by the edges corresponding to each pair of of points. As we add more points, and the number of points become greater than the number of dimensions, some of the points will be inside the convex hull, as we can see in Figure 12.9(a).

convex hull

In general, building a convex convex hull can be done by introducing non-negative weights $\alpha_n \geqslant 0$ corresponding to each example $\boldsymbol{x}_n$. Then the convex hull can be described as the set

$$\operatorname{conv}(\boldsymbol{X}) = \left\{ \sum_{n=1}^{N} \alpha_n \boldsymbol{x}_n \right\} \quad \text{with} \quad \sum_{n=1}^{N} \alpha_n = 1 \quad \text{and} \quad \alpha_n \geqslant 0, \tag{12.43}$$

for all $n = 1, \ldots, N$. If the two clouds of points corresponding to the positive and negative classes are separated, then the convex hulls do not overlap. 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 examples, and is closest to the negative class distribution. Similarly we pick a point $\boldsymbol{d}$ in the convex hull of the set of negative examples, and is closest to the positive class distribution, see Figure 12.9(b). We define a difference vector between $\boldsymbol{d}$ and $\boldsymbol{c}$ as

$$\boldsymbol{w} := \boldsymbol{c} - \boldsymbol{d}\,. \tag{12.44}$$

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 $\boldsymbol{w}$, so that we end up with the corresponding optimization problem

$$\arg\min_{\boldsymbol{w}} \|\boldsymbol{w}\| = \arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2\,. \tag{12.45}$$

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

$$\boldsymbol{c} = \sum_{n:y_n=+1} \alpha_n^+ \boldsymbol{x}_n\,. \tag{12.46}$$

In (12.46) we use the notation $n : y_n = +1$ to indicate the set of indices n for which $y_n = +1$. Similarly, for the examples with negative labels we obtain

$$\boldsymbol{d} = \sum_{n:y_n=-1} \alpha_n^- \boldsymbol{x}_n\,. \tag{12.47}$$

By substituting (12.44), (12.46) and (12.47) into (12.45), we obtain the objective

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\left\| \sum_{n:y_n=+1} \alpha_n^+ \boldsymbol{x}_n - \sum_{n:y_n=-1} \alpha_n^- \boldsymbol{x}_n \right\|^2\,. \tag{12.48}$$

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 their coefficients sum to one,

$$\sum_{n:y_n=+1} \alpha_n^+ = 1 \quad \text{and} \quad \sum_{n:y_n=-1} \alpha_n^- = 1\,. \tag{12.49}$$

This implies the constraint

$$\sum_{n=1}^{N} y_n \alpha_n = 0. \tag{12.50}$$

This result can be seen by multiplying out the individual classes

$$\sum_{n=1}^{N} y_n \alpha_n = \sum_{n:y_n=+1} (+1)\alpha_n^+ + \sum_{n:y_n=-1} (-1)\alpha_n^- \tag{12.51a}$$

$$= \sum_{n:y_n=+1} \alpha_n^+ - \sum_{n:y_n=-1} \alpha_n^- = 1 - 1 = 0\,. \tag{12.51b}$$

The objective function (12.48) and the constraint (12.50), along with the assumption that $\boldsymbol{\alpha} \geqslant \mathbf{0}$, give us a constrained (convex) optimization problem. This optimization problem can be shown to be the same as that of the dual hard margin SVM (Bennett and Bredensteiner, 2000a).

reduced hull

Remark. 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 (Bennett and Bredensteiner, 2000b). ♢

12.4 Kernels

Consider the formulation of the dual SVM (12.41). 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. In this section we discuss the representation $\boldsymbol{\phi}(\boldsymbol{x})$ and briefly introduce the idea of kernels, but do not go into the technical details.

Since $\boldsymbol{\phi}(\boldsymbol{x})$ could be a non-linear function, we can use the SVM (which assumes a linear classifier) to construct classifiers that are nonlinear in the examples $\boldsymbol{x}_n$. 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. Instead of *explicitly* defining a non-linear feature map $\boldsymbol{\phi}(\cdot)$ and computing the resulting inner product between examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we define a similarity function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. For a certain class of similarity functions, called *kernels*, the similarity function *implicitly* defines a non-linear feature map $\boldsymbol{\phi}(\cdot)$. Kernels are by definition functions $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for which there exists a Hilbert space $\mathcal{H}$ and $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{H}$ a feature map such that

kernels

The inputs $\mathcal{X}$ of the kernel function can be very general, and is not necessarily restricted to $\mathbb{R}^D$.

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle_{\mathcal{H}}\,. \tag{12.52}$$

There is a unique reproducing kernel Hilbert space associated with every kernel k (Aronszajn, 1950; Berlinet and Thomas-Agnan, 2004). In this unique association $\boldsymbol{\phi}(\boldsymbol{x}) = k(\cdot, \boldsymbol{x})$ is called the *canonical feature map*. The generalization from an inner product to a kernel function (12.52) is

canonical feature map

known as the *kernel trick* (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004), as it hides away the explicit non-linear feature map.

kernel trick

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 functions so that every kernel matrix $\boldsymbol{K}$ is symmetric and positive semi-definite (Section 3.2.3):

Gram matrix

kernel matrix

$$\forall \boldsymbol{z} \in \mathbb{R}^N \qquad \boldsymbol{z}^\top \boldsymbol{K}\boldsymbol{z} \geqslant 0\,. \tag{12.53}$$

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 (Schölkopf and Smola, 2002; Rasmussen and Williams, 2006). Figure 12.10 illustrates the effect of different kernels on separating hyperplanes on an example dataset. Note that we are still solving for hyperplanes, that is the hypothesis class of functions are still linear. The non-linear surfaces are due to the kernel function.

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 another word for the null space. The third common use of the word kernel in machine learning is the smoothing kernel in kernel density estimation (Section 11.5). ♢

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 (Schölkopf and Smola, 2002), 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 example is the Gaussian radial basis function kernel (Schölkopf and Smola, 2002; Rasmussen and Williams, 2006) where the corresponding feature space is infinite dimensional. In this case, we cannot explicitly represent the feature space but can still compute similarities between a pair of examples using the kernel.

The choice of kernel, as well as the parameters of the kernel are often chosen using nested cross validation (Section 8.5.2).

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, graphs and distributions (Ben-Hur et al., 2008; Gärtner, 2008; Shi et al., 2009; Vishwanathan et al., 2010; Sriperumbudur et al., 2010).

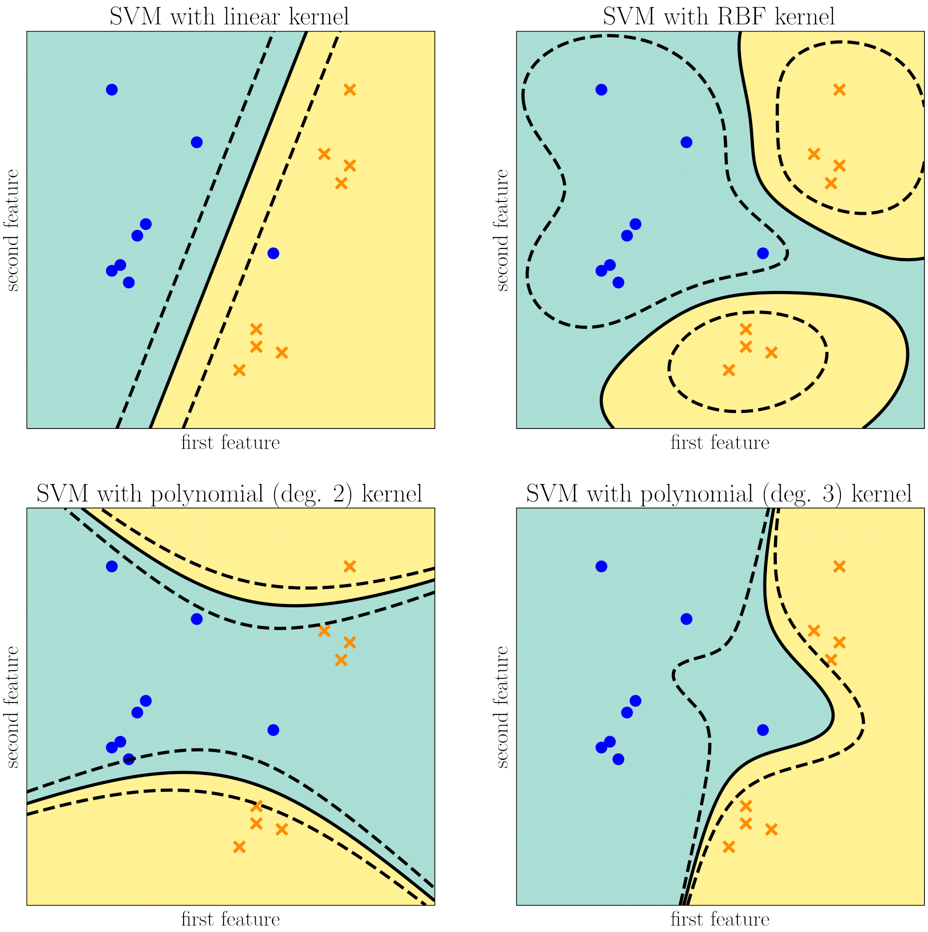


Figure 12.10 SVM with different kernels. Note that while the decision boundary is nonlinear, the underlying problem being solved is for a linear separating hyperplane (albeit with a nonlinear kernel).

12.5 Numerical Solution

We conclude our discussion of SVMs by looking at how to express the problems derived in this chapter in terms of the concepts presented in Chapter 7. We consider two different approaches for finding the optimal solution for the SVM. First we consider the loss view of SVM 8.1.2 and express this as an unconstrained optimization problem. Then we express the constrained versions of the primal and dual SVMs as quadratic programs in standard form 7.3.2.

Consider the loss function view of the SVM (12.31). This is a convex unconstrained optimization problem, but the hinge loss (12.28) is not differentiable. Therefore, we apply a subgradient approach for solving it. However, the hinge loss 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

$$g(t) = \begin{cases} -1 & t < 1 \\ [-1, 0] & t = 1 \\ 0 & t > 1 \end{cases} . \tag{12.54}$$

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 (12.26a) has optimization variables that have the size of the dimension D of the input examples. The dual SVM in (12.41) has optimization variables that have the size of the number N of examples.

To express the primal SVM in the standard form (7.45) for quadratic programming, let us assume that we use the dot product (3.5) as the inner product. We rearrange the equation for the primal SVM (12.26a), such that the optimization variables are all on the right and the inequality of the constraint matches the standard form. This yields the optimization

Recall from Section 3.2 that in this book, we use the phrase dot product to mean the inner product on Euclidean vector space.

$$\begin{aligned} \min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N} \xi_n \\ \text{subject to} \quad & -y_n \boldsymbol{x}_n^\top \boldsymbol{w} - y_n b - \xi_n \leqslant -1 \\ & -\xi_n \leqslant 0 \end{aligned} \tag{12.55}$$

$n = 1, \ldots, N$. By concatenating the variables $\boldsymbol{w}, b, \boldsymbol{x}_n$ into a single vector, and carefully collecting the terms, we obtain the following matrix form of the soft margin SVM.

$$\begin{aligned} \min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\begin{bmatrix} \boldsymbol{w} \\ b \\ \boldsymbol{\xi} \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{I}_D & \boldsymbol{0}_{D,N+1} \\ \boldsymbol{0}_{N+1,D} & \boldsymbol{0}_{N+1,N+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ b \\ \boldsymbol{\xi} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0}_{D+1,1} & C\boldsymbol{1}_{N,1} \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{w} \\ b \\ \boldsymbol{\xi} \end{bmatrix} \\ \text{subject to} \quad & \begin{bmatrix} -\boldsymbol{Y}\boldsymbol{X} & -\boldsymbol{y} & -\boldsymbol{I}_N \\ \boldsymbol{0}_{N,D+1} & & -\boldsymbol{I}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ b \\ \boldsymbol{\xi} \end{bmatrix} \leqslant \begin{bmatrix} -\boldsymbol{1}_{N,1} \\ \boldsymbol{0}_{N,1} \end{bmatrix} . \end{aligned} \tag{12.56}$$

In the above optimization problem, the minimization is over $[\boldsymbol{w}^\top, b, \boldsymbol{\xi}^\top]^\top \in \mathbb{R}^{D+1+N}$, and we use the notation: $\boldsymbol{I}_m$ to represent the identity matrix of size $m \times m$, $\boldsymbol{0}_{m,n}$ to represent the matrix of zeros of size $m \times n$, and $\boldsymbol{1}_{m,n}$ to represent the matrix of ones of size $m \times n$. In addition $\boldsymbol{y}$ is the vector of labels $[y_1, \ldots, y_N]^\top$, $\boldsymbol{Y} = \operatorname{diag}(\boldsymbol{y})$ 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 (12.41). 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

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha} - \boldsymbol{1}_{N,1}^\top \boldsymbol{\alpha} \tag{12.57}$$

$$\text{subject to} \begin{bmatrix} \boldsymbol{y}^\top \\ -\boldsymbol{y}^\top \\ -\boldsymbol{I}_N \\ \boldsymbol{I}_N \end{bmatrix} \boldsymbol{\alpha} \leqslant \begin{bmatrix} \boldsymbol{0}_{N+2,1} \\ C\boldsymbol{1}_{N,1} \end{bmatrix}. \tag{12.58}$$

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.,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad \text{is replaced by} \quad \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b} \quad \text{and} \quad \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b} \tag{12.59}$$

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).

12.6 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 development of SVMs is closely linked to empirical risk minimization discussed in Section 8.1. Hence, the SVM has strong theoretical properties (Vapnik, 2000; Steinwart and Christmann, 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.

An alternative derivation of the dual SVM can be obtained using the idea of the Legendre-Fenchel transform (Section 7.3.3). The derivation considers each term of the unconstrained formulation of the SVM (12.31) separately and calculates their convex conjugates (Rifkin and Lippert, 2007). 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). The idea of kernels have been generalized to Banach spaces (Zhang et al., 2009) and Kreĭn spaces (Ong et al., 2004; Loosli et al., 2016).

Observe that the hinge loss has three equivalent representations, as shown in (12.28) and (12.29), as well as the constrained optimization problem in (12.33). The formulation (12.28) is often used when comparing the SVM loss function with other loss functions (Steinwart, 2007). The two-piece formulation (12.29) is convenient for computing subgradients, as each piece is linear. The third formulation (12.33), as seen in Section 12.5, enables the use of convex quadratic programming (Section 7.3.2) tools.

Since binary classification is a well-studied task in machine learning, other words are also sometimes used, such as discrimination, separation or decision. Furthermore, there are three quantities that can be the output of a binary classifier. First is the output of the linear function itself (often called the score), which can take any real value. 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, for example in the interval $[0, 1]$. 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 $\{+1, -1\}$, which is the one most commonly assumed to be the output of the classifier.

The SVM is a binary classifier that does not naturally lend itself to a probabilistic interpretation. There are several approaches for converting the raw output of the linear function (the score) into a calibrated class probability estimate ($P(Y = 1|X = \boldsymbol{x})$) which involve an additional calibration step (Platt, 2000; Lin et al., 2007; Zadrozny and Elkan, 2001). From the training perspective, there are many related probabilistic approches. We mentioned at the end of Section 12.2.5 that there is a relationship between loss function and the likelihood (also compare Section 8.1 and Section 8.2). The maximum likelihood approach corresponding to a well calibrated transformation during training is called logistic regression, which comes from a class of methods called generalized linear models. Details of logistic regression from this point of view can be found in Agresti (2002, Chapter 5) and McCullagh and Nelder (1989, Chapter

4). Naturally, one could take a more Bayesian view of the classifier output by estimation a posterior distribution using Bayesian logistic regression. The Bayesian view also includes the specification of the prior, which includes design choices such as conjugacy (Section 6.6.1) with the likelihood. Additionally, one could consider latent functions as priors, which results in Gaussian process classification (Rasmussen and Williams, 2006, Chapter 3).