5.5 Support Vector Machine (SVM)

A classification technique that has received considerable attention is support vector machine (SVM). This technique has its roots in statistical learning theory and has shown promising empirical results in many practical applications, from handwritten digit recognition to text categorization. SVM also works very well with high-dimensional data and avoids the curse of dimensionality problem. Another unique aspect of this approach is that it represents the decision boundary using a subset of the training examples, known as the **support vectors**.

To illustrate the basic idea behind SVM, we first introduce the concept of a **maximal margin hyperplane** and explain the rationale of choosing such a hyperplane. We then describe how a linear SVM can be trained to explicitly look for this type of hyperplane in linearly separable data. We conclude by showing how the SVM methodology can be extended to non-linearly separable data.

5.5.1 Maximum Margin Hyperplanes

Figure 5.21 shows a plot of a data set containing examples that belong to two different classes, represented as squares and circles. The data set is also linearly separable; i.e., we can find a hyperplane such that all the squares reside on one side of the hyperplane and all the circles reside on the other

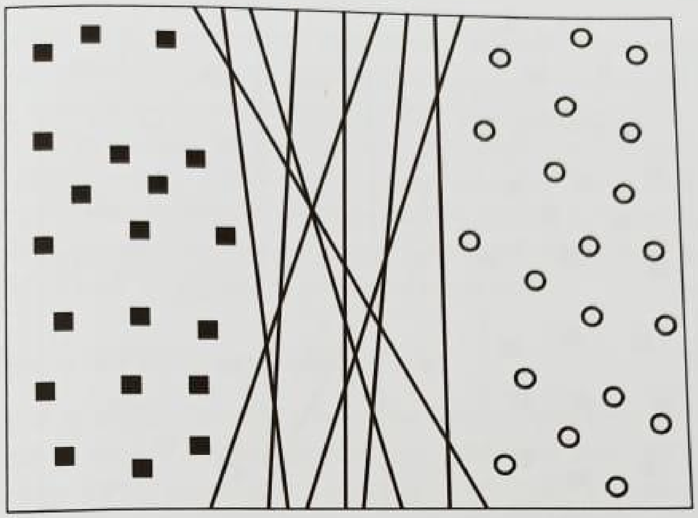


Figure 5.21. Possible decision boundaries for a linearly separable data set.

side. However, as shown in Figure 5.21, there are infinitely many such hyperplanes possible. Although their training errors are zero, there is no guarantee that the hyperplanes will perform equally well on previously unseen examples. The classifier must choose one of these hyperplanes to represent its decision boundary, based on how well they are expected to perform on test examples.

To get a clearer picture of how the different choices of hyperplanes affect the generalization errors, consider the two decision boundaries, B_1 and B_2, shown in Figure 5.22. Both decision boundaries can separate the training examples into their respective classes without committing any misclassification errors. Each decision boundary B_i is associated with a pair of hyperplanes, denoted as b_{i1} and b_{i2}, respectively. b_{i1} is obtained by moving a parallel hyperplane away from the decision boundary until it touches the closest square(s), whereas b_{i2} is obtained by moving the hyperplane until it touches the closest circle(s). The distance between these two hyperplanes is known as the margin of the classifier. From the diagram shown in Figure 5.22, notice that the margin for B_1 is considerably larger than that for B_2. In this example, B_1 turns out to be the maximum margin hyperplane of the training instances.

Rationale for Maximum Margin

Decision boundaries with large margins tend to have better generalization errors than those with small margins. Intuitively, if the margin is small, then

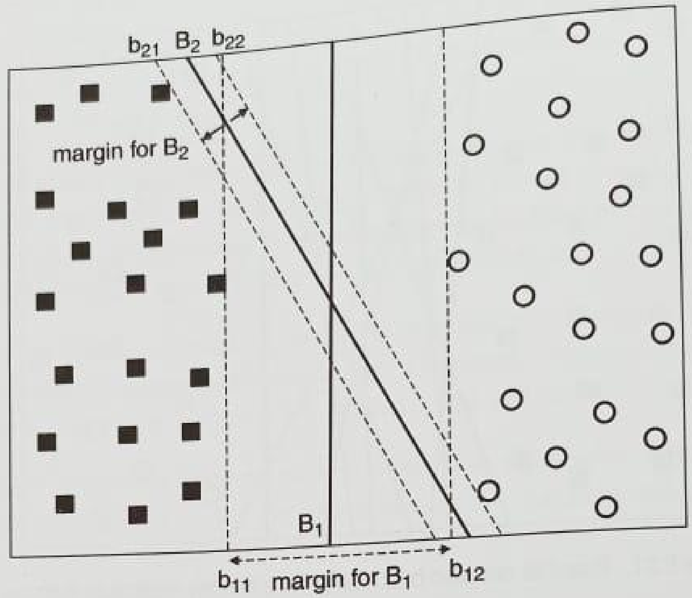


Figure 5.22. Margin of a decision boundary.

any slight perturbations to the decision boundary can have quite a significant impact on its classification. Classifiers that produce decision boundaries with small margins are therefore more susceptible to model overfitting and tend to generalize poorly on previously unseen examples.

A more formal explanation relating the margin of a linear classifier to its generalization error is given by a statistical learning principle known as **structural risk minimization** (SRM). This principle provides an upper bound to the generalization error of a classifier (R) in terms of its training error (R_e), the number of training examples (N), and the model complexity, otherwise known as its **capacity** (h). More specifically, with a probability of $1 - \eta$, the generalization error of the classifier can be at worst

$$R \leq R_e + \varphi\left(\frac{h}{N}, \frac{\log(\eta)}{N}\right), \tag{5.27}$$

where φ is a monotone increasing function of the capacity h. The preceding inequality may seem quite familiar to the readers because it resembles the equation given in Section 4.4.4 (on page 179) for the minimum description length (MDL) principle. In this regard, SRM is another way to express generalization error as a tradeoff between training error and model complexity.

The capacity of a linear model is inversely related to its margin. Models with small margins have higher capacities because they are more flexible and can fit many training sets, unlike models with large margins. However, according to the SRM principle, as the capacity increases, the generalization error bound will also increase. Therefore, it is desirable to design linear classifiers that maximize the margins of their decision boundaries in order to ensure that their worst-case generalization errors are minimized. One such classifier is the linear **SVM**, which is explained in the next section.

5.5.2 Linear SVM: Separable Case

A linear SVM is a classifier that searches for a hyperplane with the largest margin, which is why it is often known as a **maximal margin classifier**. To understand how SVM learns such a boundary, we begin with some preliminary discussion about the decision boundary and margin of a linear classifier.

Linear Decision Boundary

Consider a binary classification problem consisting of N training examples. Each example is denoted by a tuple $(\mathbf{x_i}, y_i)$ $(i = 1, 2, \ldots, N)$, where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})^T$ corresponds to the attribute set for the i^{th} example. By convention, let $y_i \in \{-1, 1\}$ denote its class label. The decision boundary of a linear classifier can be written in the following form:

$$\mathbf{w} \cdot \mathbf{x} + b = 0, \tag{5.28}$$

where $\mathbf{w}$ and b are parameters of the model.

Figure 5.23 shows a two-dimensional training set consisting of squares and circles. A decision boundary that bisects the training examples into their respective classes is illustrated with a solid line. Any example located along the decision boundary must satisfy Equation 5.28. For example, if $\mathbf{x}_a$ and $\mathbf{x}_b$ are two points located on the decision boundary, then

$$\mathbf{w} \cdot \mathbf{x}_a + b = 0,$$
$$\mathbf{w} \cdot \mathbf{x}_b + b = 0.$$

Subtracting the two equations will yield the following:

$$\mathbf{w} \cdot (\mathbf{x}_b - \mathbf{x}_a) = 0,$$

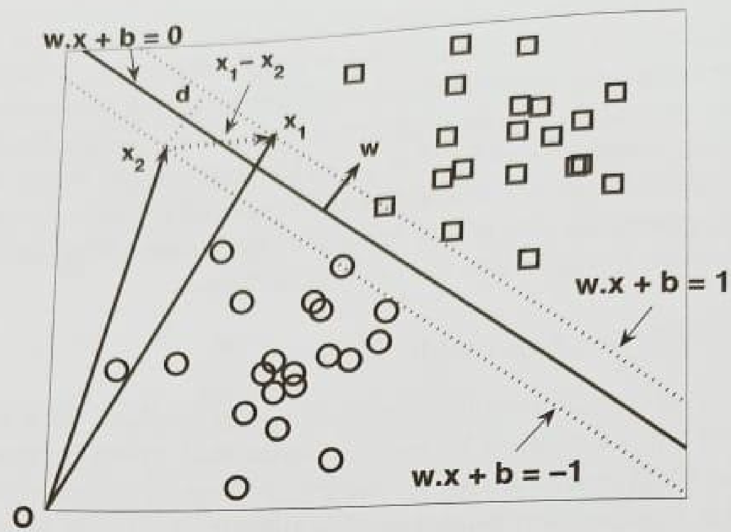


Figure 5.23. Decision boundary and margin of SVM.

where $\mathbf{x}_b - \mathbf{x}_a$ is a vector parallel to the decision boundary and is directed from $\mathbf{x}_a$ to $\mathbf{x}_b$. Since the dot product is zero, the direction for $\mathbf{w}$ must be perpendicular to the decision boundary, as shown in Figure 5.23.

For any square $\mathbf{x}_s$ located above the decision boundary, we can show that

$$\mathbf{w} \cdot \mathbf{x}_s + b = k, \tag{5.29}$$

where $k > 0$. Similarly, for any circle $\mathbf{x}_c$ located below the decision boundary, we can show that

$$\mathbf{w} \cdot \mathbf{x}_c + b = k', \tag{5.30}$$

where $k' < 0$. If we label all the squares as class $+1$ and all the circles as class -1, then we can predict the class label y for any test example $\mathbf{z}$ in the following way:

$$y = \begin{cases} 1, & \text{if } \mathbf{w} \cdot \mathbf{z} + b > 0; \\ -1, & \text{if } \mathbf{w} \cdot \mathbf{z} + b < 0. \end{cases} \tag{5.31}$$

Margin of a Linear Classifier

Consider the square and the circle that are closest to the decision boundary. Since the square is located above the decision boundary, it must satisfy Equation 5.29 for some positive value k, whereas the circle must satisfy Equation

5.30 for some negative value k'. We can rescale the parameters $\mathbf{w}$ and b of the decision boundary so that the two parallel hyperplanes b_{i1} and b_{i2} can be expressed as follows:

$$b_{i1}: \ \mathbf{w} \cdot \mathbf{x} + b = 1, \tag{5.32}$$

$$b_{i2}: \ \mathbf{w} \cdot \mathbf{x} + b = -1. \tag{5.33}$$

The margin of the decision boundary is given by the distance between these two hyperplanes. To compute the margin, let $\mathbf{x}_1$ be a data point located on b_{i1} and $\mathbf{x}_2$ be a data point on b_{i2}, as shown in Figure 5.23. Upon substituting these points into Equations 5.32 and 5.33, the margin d can be computed by subtracting the second equation from the first equation:

$$\begin{aligned} \mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) &= 2 \\ \|\mathbf{w}\| \times d &= 2 \\ \therefore d &= \frac{2}{\|\mathbf{w}\|}. \end{aligned} \tag{5.34}$$

Learning a Linear SVM Model

The training phase of SVM involves estimating the parameters $\mathbf{w}$ and b of the decision boundary from the training data. The parameters must be chosen in such a way that the following two conditions are met:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x_i} + b &\geq 1 \text{ if } y_i = 1, \\ \mathbf{w} \cdot \mathbf{x_i} + b &\leq -1 \text{ if } y_i = -1. \end{aligned} \tag{5.35}$$

These conditions impose the requirements that all training instances from class $y = 1$ (i.e., the squares) must be located on or above the hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 1$, while those instances from class $y = -1$ (i.e., the circles) must be located on or below the hyperplane $\mathbf{w} \cdot \mathbf{x} + b = -1$. Both inequalities can be summarized in a more compact form as follows:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, \quad i = 1, 2, \ldots, N. \tag{5.36}$$

Although the preceding conditions are also applicable to any linear classifiers (including perceptrons), SVM imposes an additional requirement that the margin of its decision boundary must be maximal. Maximizing the margin, however, is equivalent to minimizing the following objective function:

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}. \tag{5.37}$$

Definition 5.1 (Linear SVM: Separable Case). The learning task in SVM can be formalized as the following constrained optimization problem:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}$$
$$\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x_i}+b) \geq 1, \quad i = 1, 2, \ldots, N.$$

Since the objective function is quadratic and the constraints are linear in the parameters $\mathbf{w}$ and b, this is known as a **convex** optimization problem, which can be solved using the standard **Lagrange multiplier** method. Following is a brief sketch of the main ideas for solving the optimization problem. A more detailed discussion is given in Appendix E.

First, we must rewrite the objective function in a form that takes into account the constraints imposed on its solutions. The new objective function is known as the Lagrangian for the optimization problem:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i \Big(y_i(\mathbf{w}\cdot\mathbf{x_i}+b) - 1 \Big), \tag{5.38}$$

where the parameters λ_i are called the Lagrange multipliers. The first term in the Lagrangian is the same as the original objective function, while the second term captures the inequality constraints. To understand why the objective function must be modified, consider the original objective function given in Equation 5.37. It is easy to show that the function is minimized when $\mathbf{w} = \mathbf{0}$, a null vector whose components are all zeros. Such a solution, however, violates the constraints given in Definition 5.1 because there is no feasible solution for b. The solutions for $\mathbf{w}$ and b are infeasible if they violate the inequality constraints; i.e., if $y_i(\mathbf{w}\cdot\mathbf{x_i}+b)-1 < 0$. The Lagrangian given in Equation 5.38 incorporates this constraint by subtracting the term from its original objective function. Assuming that $\lambda_i \geq 0$, it is clear that any infeasible solution may only increase the value of the Lagrangian.

To minimize the Lagrangian, we must take the derivative of L_P with respect to $\mathbf{w}$ and b and set them to zero:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Longrightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i, \tag{5.39}$$

$$\frac{\partial L_p}{\partial b} = 0 \Longrightarrow \sum_{i=1}^{N} \lambda_i y_i = 0. \tag{5.40}$$

Because the Lagrange multipliers are unknown, we still cannot solve for $\mathbf{w}$ and b. If Definition 5.1 contains only equality instead of inequality constraints, then we can use the N equations from equality constraints along with Equations 5.39 and 5.40 to find the feasible solutions for $\mathbf{w}$, b, and λ_i. Note that the Lagrange multipliers for equality constraints are free parameters that can take any values.

One way to handle the inequality constraints is to transform them into a set of equality constraints. This is possible as long as the Lagrange multipliers are restricted to be non-negative. Such transformation leads to the following constraints on the Lagrange multipliers, which are known as the Karush-Kuhn-Tucker (KKT) conditions:

$$\lambda_i \geq 0, \tag{5.41}$$

$$\lambda_i\big[y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1\big] = 0. \tag{5.42}$$

At first glance, it may seem that there are as many Lagrange multipliers as there are training instances. It turns out that many of the Lagrange multipliers become zero after applying the constraint given in Equation 5.42. The constraint states that the Lagrange multiplier λ_i must be zero unless the training instance $\mathbf{x}_i$ satisfies the equation $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) = 1$. Such training instance, with $\lambda_i > 0$, lies along the hyperplanes b_{i1} or b_{i2} and is known as a support vector. Training instances that do not reside along these hyperplanes have $\lambda_i = 0$. Equations 5.39 and 5.42 also suggest that the parameters $\mathbf{w}$ and b, which define the decision boundary, depend only on the support vectors.

Solving the preceding optimization problem is still quite a daunting task because it involves a large number of parameters: $\mathbf{w}$, b, and λ_i. The problem can be simplified by transforming the Lagrangian into a function of the Lagrange multipliers only (this is known as the dual problem). To do this, we first substitute Equations 5.39 and 5.40 into Equation 5.38. This will lead to the following dual formulation of the optimization problem:

$$L_D = \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j \mathbf{x_i}\cdot\mathbf{x_j}. \tag{5.43}$$

The key differences between the dual and primary Lagrangians are as follows:

1. The dual Lagrangian involves only the Lagrange multipliers and the training data, while the primary Lagrangian involves the Lagrange multipliers as well as parameters of the decision boundary. Nevertheless, the solutions for both optimization problems are equivalent.

2. The quadratic term in Equation 5.43 has a negative sign, which means that the original minimization problem involving the primary Lagrangian, L_P, has turned into a maximization problem involving the dual Lagrangian, L_D.

For large data sets, the dual optimization problem can be solved using numerical techniques such as quadratic programming, a topic that is beyond the scope of this book. Once the λ_i's are found, we can use Equations 5.39 and 5.42 to obtain the feasible solutions for $\mathbf{w}$ and b. The decision boundary can be expressed as follows:

$$\left(\sum_{i=1}^{N} \lambda_i y_i \mathbf{x_i} \cdot \mathbf{x}\right) + b = 0. \tag{5.44}$$

b is obtained by solving Equation 5.42 for the support vectors. Because the λ_i's are calculated numerically and can have numerical errors, the value computed for b may not be unique. Instead it depends on the support vector used in Equation 5.42. In practice, the average value for b is chosen to be the parameter of the decision boundary.

Example 5.5. Consider the two-dimensional data set shown in Figure 5.24, which contains eight training instances. Using quadratic programming, we can solve the optimization problem stated in Equation 5.43 to obtain the Lagrange multiplier λ_i for each training instance. The Lagrange multipliers are depicted in the last column of the table. Notice that only the first two instances have non-zero Lagrange multipliers. These instances correspond to the support vectors for this data set.

Let $\mathbf{w} = (w_1, w_2)$ and b denote the parameters of the decision boundary. Using Equation 5.39, we can solve for w_1 and w_2 in the following way:

$$\begin{aligned} w_1 &= \sum_i \lambda_i y_i x_{i1} = 65.5621 \times 1 \times 0.3858 + 65.5621 \times -1 \times 0.4871 = -6.64. \\ w_2 &= \sum_i \lambda_i y_i x_{i2} = 65.5621 \times 1 \times 0.4687 + 65.5621 \times -1 \times 0.611 = -9.32. \end{aligned}$$

The bias term b can be computed using Equation 5.42 for each support vector:

$$\begin{aligned} b^{(1)} &= 1 - \mathbf{w} \cdot \mathbf{x}_1 = 1 - (-6.64)(0.3858) - (-9.32)(0.4687) = 7.9300. \\ b^{(2)} &= -1 - \mathbf{w} \cdot \mathbf{x}_2 = -1 - (-6.64)(0.4871) - (-9.32)(0.611) = 7.9289. \end{aligned}$$

Averaging these values, we obtain $b = 7.93$. The decision boundary corresponding to these parameters is shown in Figure 5.24. ∎

x_1	x_2	y	Lagrange Multiplier
0.3858	0.4687	1	65.5261
0.4871	0.611	−1	65.5261
0.9218	0.4103	−1	0
0.7382	0.8936	−1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	−1	0
0.2146	0.0099	1	0

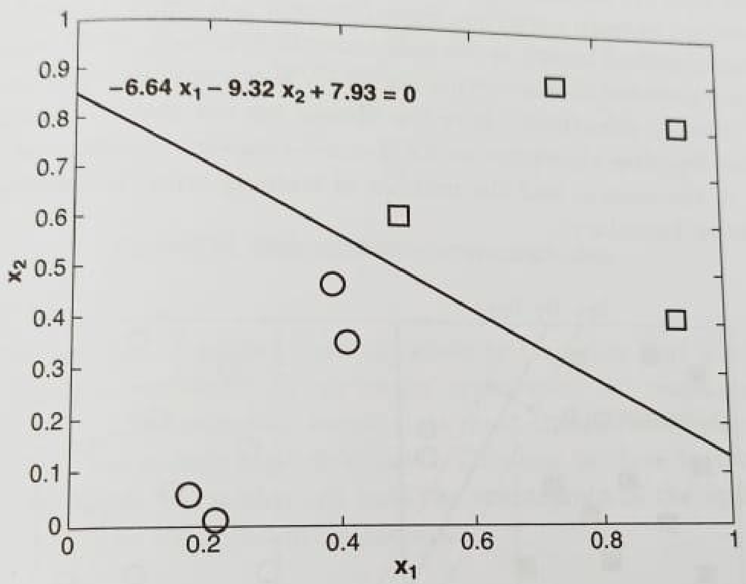


Figure 5.24. Example of a linearly separable data set.

Once the parameters of the decision boundary are found, a test instance $\mathbf{z}$ is classified as follows:

$$f(\mathbf{z}) = sign\big(\mathbf{w}\cdot\mathbf{z}+b\big) = sign\bigg(\sum_{i=1}^{N}\lambda_i y_i \mathbf{x_i}\cdot\mathbf{z}+b\bigg).$$

If $f(\mathbf{z}) = 1$, then the test instance is classified as a positive class; otherwise, it is classified as a negative class.

5.5.3 Linear SVM: Nonseparable Case

Figure 5.25 shows a data set that is similar to Figure 5.22, except it has two new examples, P and Q. Although the decision boundary B_1 misclassifies the new examples, while B_2 classifies them correctly, this does not mean that B_2 is a better decision boundary than B_1 because the new examples may correspond to noise in the training data. B_1 should still be preferred over B_2 because it has a wider margin, and thus, is less susceptible to overfitting. However, the SVM formulation presented in the previous section constructs only decision boundaries that are mistake-free. This section examines how the formulation can be modified to learn a decision boundary that is tolerable to small training errors using a method known as the **soft margin** approach. More importantly, the method presented in this section allows SVM to construct a linear decision boundary even in situations where the classes are not linearly separable. To do this, the learning algorithm in SVM must consider the trade-off between the width of the margin and the number of training errors committed by the linear decision boundary.

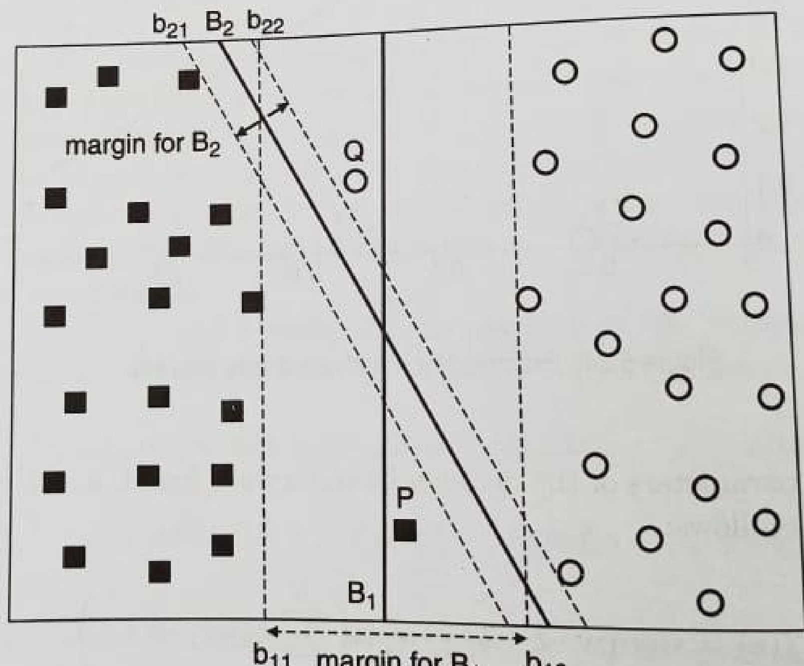


Figure 5.25. Decision boundary of SVM for the nonseparable case.

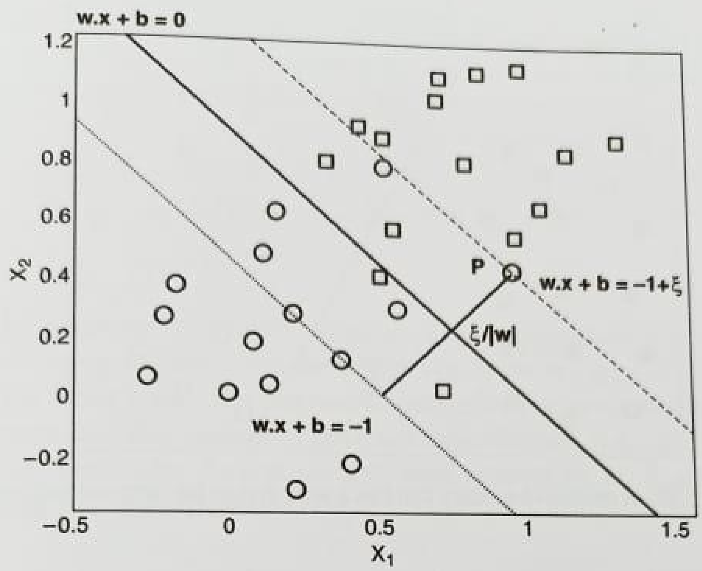


Figure 5.26. Slack variables for nonseparable data.

While the original objective function given in Equation 5.37 is still applicable, the decision boundary B_1 no longer satisfies all the constraints given in Equation 5.36. The inequality constraints must therefore be relaxed to accommodate the nonlinearly separable data. This can be done by introducing positive-valued **slack variables** (ξ) into the constraints of the optimization problem, as shown in the following equations:

$$\begin{aligned} \mathbf{w}\cdot\mathbf{x_i}+b &\geq 1-\xi_i \quad \text{if } y_i = 1, \\ \mathbf{w}\cdot\mathbf{x_i}+b &\leq -1+\xi_i \quad \text{if } y_i = -1, \end{aligned} \tag{5.45}$$

where $\forall i: \ \xi_i > 0$.

To interpret the meaning of the slack variables ξ_i, consider the diagram shown in Figure 5.26. The circle **P** is one of the instances that violates the constraints given in Equation 5.35. Let $\mathbf{w}\cdot\mathbf{x}+b=-1+\xi$ denote a line that is parallel to the decision boundary and passes through the point **P**. It can be shown that the distance between this line and the hyperplane $\mathbf{w}\cdot\mathbf{x}+b=-1$ is $\xi/\|\mathbf{w}\|$. Thus, ξ provides an estimate of the error of the decision boundary on the training example **P**.

In principle, we can apply the same objective function as before and impose the conditions given in Equation 5.45 to find the decision boundary. However,

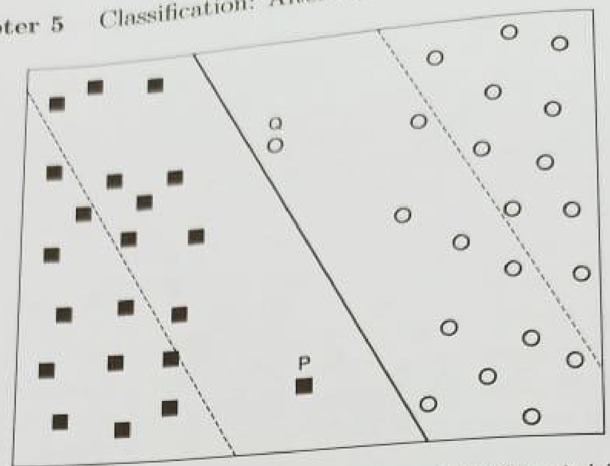


Figure 5.27. A decision boundary that has a wide margin but large training error.

since there are no constraints on the number of mistakes the decision boundary can make, the learning algorithm may find a decision boundary with a very wide margin but misclassifies many of the training examples, as shown in Figure 5.27. To avoid this problem, the objective function must be modified to penalize a decision boundary with large values of slack variables. The modified objective function is given by the following equation:

$$f(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2} + C(\sum_{i=1}^{N} \xi_i)^k,$$

where C and k are user-specified parameters representing the penalty of misclassifying the training instances. For the remainder of this section, we assume $k = 1$ to simplify the problem. The parameter C can be chosen based on the model's performance on the validation set.

It follows that the Lagrangian for this constrained optimization problem can be written as follows:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\{y_i(\mathbf{w}\cdot\mathbf{x_i} + b) - 1 + \xi_i\} - \sum_{i=1}^{N}\mu_i\xi_i, \quad (5.46)$$

where the first two terms are the objective function to be minimized, the third term represents the inequality constraints associated with the slack variables, and the last term is the result of the non-negativity requirements on the values of ξ_i's. Furthermore, the inequality constraints can be transformed into equality constraints using the following KKT conditions:

$$\xi_i \geq 0, \quad \lambda_i \geq 0, \quad \mu_i \geq 0, \tag{5.47}$$
$$\lambda_i\{y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i\} = 0, \tag{5.48}$$
$$\mu_i\xi_i = 0. \tag{5.49}$$

Note that the Lagrange multiplier λ_i given in Equation 5.48 is non-vanishing only if the training instance resides along the lines $\mathbf{w}\cdot\mathbf{x}_i + b = \pm 1$ or has $\xi_i > 0$. On the other hand, the Lagrange multipliers μ_i given in Equation 5.49 are zero for any training instances that are misclassified (i.e., having $\xi_i > 0$).

Setting the first-order derivative of L with respect to $\mathbf{w}$, b, and ξ_i to zero would result in the following equations:

$$\frac{\partial L}{\partial w_j} = w_j - \sum_{i=1}^{N} \lambda_i y_i x_{ij} = 0 \Longrightarrow w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij}. \tag{5.50}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N} \lambda_i y_i = 0 \Longrightarrow \sum_{i=1}^{N} \lambda_i y_i = 0. \tag{5.51}$$

$$\frac{\partial L}{\partial \xi_i} = C - \lambda_i - \mu_i = 0 \Longrightarrow \lambda_i + \mu_i = C. \tag{5.52}$$

Substituting Equations 5.50, 5.51, and 5.52 into the Lagrangian will produce the following dual Lagrangian:

$$\begin{aligned} L_D &= \frac{1}{2}\sum_{i,j} \lambda_i\lambda_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j + C\sum_i \xi_i \\ &\quad -\sum_i \lambda_i\{y_i(\sum_j \lambda_j y_j \mathbf{x}_i\cdot\mathbf{x}_j + b) - 1 + \xi_i\} \\ &\quad -\sum_i (C - \lambda_i)\xi_i \\ &= \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i\lambda_j y_i y_j \mathbf{x_i}\cdot\mathbf{x_j}, \end{aligned} \tag{5.53}$$

which turns out to be identical to the dual Lagrangian for linearly separable data (see Equation 5.40 on page 262). Nevertheless, the constraints imposed on the Lagrange multipliers λ_i's are slightly different those in the linearly separable case. In the linearly separable case, the Lagrange multipliers must be non-negative, i.e., $\lambda_i \geq 0$. On the other hand, Equation 5.52 suggests that λ_i should not exceed C (since both μ_i and λ_i are non-negative). Therefore, the Lagrange multipliers for nonlinearly separable data are restricted to $0 \leq \lambda_i \leq C$.

The dual problem can then be solved numerically using quadratic programming techniques to obtain the Lagrange multipliers λ_i. These multipliers can be replaced into Equation 5.50 and the KKT conditions to obtain the parameters of the decision boundary.

5.5.4 Nonlinear SVM

The SVM formulations described in the previous sections construct a linear decision boundary to separate the training examples into their respective classes. This section presents a methodology for applying SVM to data sets that have nonlinear decision boundaries. The trick here is to transform the data from its original coordinate space in $\mathbf{x}$ into a new space $\Phi(\mathbf{x})$ so that a linear decision boundary can be used to separate the instances in the transformed space. After doing the transformation, we can apply the methodology presented in the previous sections to find a linear decision boundary in the transformed space.

Attribute Transformation

To illustrate how attribute transformation can lead to a linear decision boundary, Figure 5.28(a) shows an example of a two-dimensional data set consisting of squares (classified as $y = 1$) and circles (classified as $y = -1$). The data set is generated in such a way that all the circles are clustered near the center of the diagram and all the squares are distributed farther away from the center. Instances of the data set can be classified using the following equation:

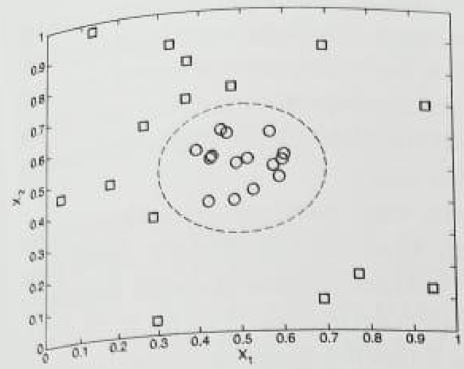
$$y(x_1, x_2) = \begin{cases} 1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2, \\ -1 & \text{otherwise.} \end{cases} \tag{5.54}$$

The decision boundary for the data can therefore be written as follows:

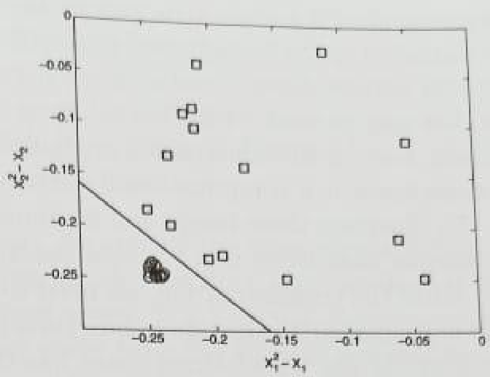
$$\sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} = 0.2,$$

which can be further simplified into the following quadratic equation:

$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46.$$



(a) Decision boundary in the original two-dimensional space.



(b) Decision boundary in the transformed space.

Figure 5.28. Classifying data with a nonlinear decision boundary.

A nonlinear transformation Φ is needed to map the data from its original feature space into a new space where the decision boundary becomes linear. Suppose we choose the following transformation:

$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1). \tag{5.55}$$

In the transformed space, we can find the parameters $\mathbf{w} = (w_0, w_1, \ldots, w_4)$ such that:

$$w_4 x_1^2 + w_3 x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

For illustration purposes, let us plot the graph of $x_2^2 - x_2$ versus $x_1^2 - x_1$ for the previously given instances. Figure 5.28(b) shows that in the transformed space, all the circles are located in the lower right-hand side of the diagram. A linear decision boundary can therefore be constructed to separate the instances into their respective classes.

One potential problem with this approach is that it may suffer from the curse of dimensionality problem often associated with high-dimensional data. We will show how nonlinear SVM avoids this problem (using a method known as the kernel trick) later in this section.

Learning a Nonlinear SVM Model

Although the attribute transformation approach seems promising, it raises several implementation issues. First, it is not clear what type of mapping function should be used to ensure that a linear decision boundary can be constructed in the transformed space. One possibility is to transform the data into an infinite dimensional space, but such a high-dimensional space may not be that easy to work with. Second, even if the appropriate mapping function is known, solving the constrained optimization problem in the high-dimensional feature space is a computationally expensive task.

To illustrate these issues and examine the ways they can be addressed, let us assume that there is a suitable function, $\Phi(\mathbf{x})$, to transform a given data set. After the transformation, we need to construct a linear decision boundary that will separate the instances into their respective classes. The linear decision boundary in the transformed space has the following form: $\mathbf{w} \cdot \Phi(\mathbf{x}) + b = 0$.

Definition 5.2 (Nonlinear SVM). The learning task for a nonlinear SVM can be formalized as the following optimization problem:

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}$$

$$\text{subject to} \quad y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) \geq 1, \quad i = 1, 2, \ldots, N.$$

Note the similarity between the learning task of a nonlinear SVM to that of a linear SVM (see Definition 5.1 on page 262). The main difference is that, instead of using the original attributes $\mathbf{x}$, the learning task is performed on the transformed attributes $\Phi(\mathbf{x})$. Following the approach taken in Sections 5.5.2 and 5.5.3 for linear SVM, we may derive the following dual Lagrangian for the constrained optimization problem:

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \tag{5.56}$$

Once the λ_i's are found using quadratic programming techniques, the parameters $\mathbf{w}$ and b can be derived using the following equations:

$$\mathbf{w} = \sum_{i} \lambda_i y_i \Phi(\mathbf{x}_i) \tag{5.57}$$

$$\lambda_i \{ y_i (\sum_{j} \lambda_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i) + b) - 1 \} = 0, \tag{5.58}$$

which are analogous to Equations 5.39 and 5.40 for linear SVM. Finally, a test instance z can be classified using the following equation:

$$f(\mathbf{z}) = sign\big(\mathbf{w} \cdot \Phi(\mathbf{z}) + b\big) = sign\Big(\sum_{i=1}^{n} \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z}) + b\Big). \qquad (5.59)$$

Except for Equation 5.57, note that the rest of the computations (Equations 5.58 and 5.59) involve calculating the dot product (i.e., similarity) between pairs of vectors in the transformed space, $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Such computation can be quite cumbersome and may suffer from the curse of dimensionality problem. A breakthrough solution to this problem comes in the form of a method known as the **kernel trick.**

Kernel Trick

The dot product is often regarded as a measure of similarity between two input vectors. For example, the cosine similarity described in Section 2.4.5 on page 73 can be defined as the dot product between two vectors that are normalized to unit length. Analogously, the dot product $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ can also be regarded as a measure of similarity between two instances, $\mathbf{x}_i$ and $\mathbf{x}_j$, in the transformed space.

The kernel trick is a method for computing similarity in the transformed space using the original attribute set. Consider the mapping function Φ given in Equation 5.55. The dot product between two input vectors $\mathbf{u}$ and $\mathbf{v}$ in the transformed space can be written as follows:

$$\begin{aligned}
\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) &= (u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, 1) \cdot (v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, 1) \\
&= u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 + 2u_2 v_2 + 1 \\
&= (\mathbf{u} \cdot \mathbf{v} + 1)^2.
\end{aligned} \qquad (5.60)$$

This analysis shows that the dot product in the transformed space can be expressed in terms of a similarity function in the original space:

$$K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^2. \qquad (5.61)$$

The similarity function, K, which is computed in the original attribute space, is known as the **kernel function.** The kernel trick helps to address some of the concerns about how to implement nonlinear SVM. First, we do not have to know the exact form of the mapping function Φ because the kernel functions used in nonlinear SVM must satisfy a mathematical principle known as **Mercer's theorem**. This principle ensures that the kernel functions can always be expressed as the dot product between two input vectors in some high-dimensional space. The transformed space of the SVM kernels is called a **reproducing kernel Hilbert space** (RKHS). Second, computing the dot products using kernel functions is considerably cheaper than using the transformed attribute set $\Phi(\mathbf{x})$. Third, since the computations are performed in the original space, issues associated with the curse of dimensionality problem can be avoided.

Figure 5.29 shows the nonlinear decision boundary obtained by SVM using the polynomial kernel function given in Equation 5.61. A test instance $\mathbf{x}$ is classified according to the following equation:

$$\begin{aligned} f(\mathbf{z}) &= sign\Big(\sum_{i=1}^{n} \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z}) + b\Big) \\ &= sign\Big(\sum_{i=1}^{n} \lambda_i y_i K(\mathbf{x}_i, \mathbf{z}) + b\Big) \\ &= sign\Big(\sum_{i=1}^{n} \lambda_i y_i (\mathbf{x}_i \cdot \mathbf{z} + 1)^2 + b\Big), \end{aligned} \tag{5.62}$$

where b is the parameter obtained using Equation 5.58. The decision boundary obtained by nonlinear SVM is quite close to the true decision boundary shown in Figure 5.28(a).

Mercer's Theorem

The main requirement for the kernel function used in nonlinear SVM is that there must exist a corresponding transformation such that the kernel function computed for a pair of vectors is equivalent to the dot product between the vectors in the transformed space. This requirement can be formally stated in the form of Mercer's theorem.

Theorem 5.1 (Mercer's Theorem). *A kernel function K can be expressed as*

$$K(\boldsymbol{u}, \boldsymbol{v}) = \Phi(\boldsymbol{u}) \cdot \Phi(\boldsymbol{v})$$

if and only if, for any function $g(x)$ such that $\int g(\boldsymbol{x})^2 d\boldsymbol{x}$ is finite, then

$$\int K(\boldsymbol{x}, \boldsymbol{y})\ g(\boldsymbol{x})\ g(\boldsymbol{y})\ d\boldsymbol{x}\ d\boldsymbol{y} \geq 0.$$

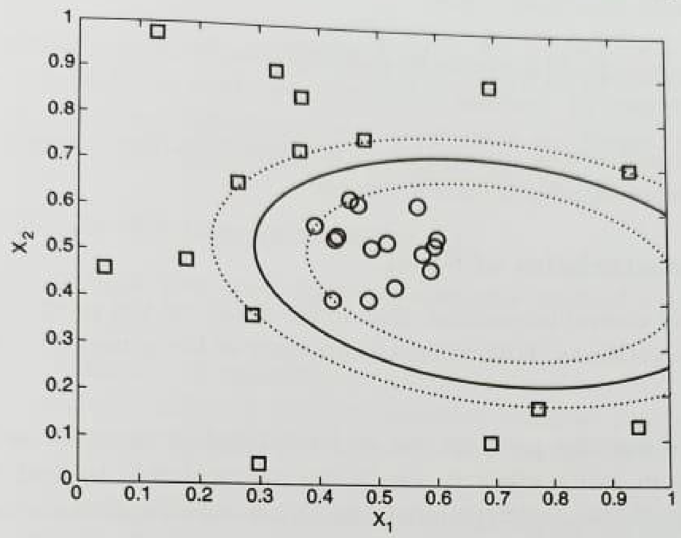


Figure 5.29. Decision boundary produced by a nonlinear SVM with polynomial kernel.

Kernel functions that satisfy Theorem 5.1 are called positive definite kernel functions. Examples of such functions are listed below:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p \tag{5.63}$$

$$K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/(2\sigma^2)} \tag{5.64}$$

$$K(\mathbf{x}, \mathbf{y}) = \tanh(k\mathbf{x} \cdot \mathbf{y} - \delta) \tag{5.65}$$

Example 5.6. Consider the polynomial kernel function given in Equation 5.63. Let $g(x)$ be a function that has a finite L_2 norm, i.e., $\int g(\mathbf{x})^2 d\mathbf{x} < \infty$.

$$\int (\mathbf{x} \cdot \mathbf{y} + 1)^p g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y}$$

$$= \int \sum_{i=0}^{p} \binom{p}{i} (\mathbf{x} \cdot \mathbf{y})^i g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y}$$

$$= \sum_{i=0}^{p} \binom{p}{i} \int \sum_{\alpha_1, \alpha_2, \ldots} \binom{i}{\alpha_1 \alpha_2 \ldots} \left[(x_1 y_1)^{\alpha_1} (x_2 y_2)^{\alpha_2} (x_3 y_3)^{\alpha_3} \ldots \right]$$

$$g(x_1, x_2, \ldots)\ g(y_1, y_2, \ldots) dx_1 dx_2 \ldots dy_1 dy_2 \ldots$$

$$= \sum_{i=0}^{p} \sum_{\alpha_1,\alpha_2,\dots} \binom{p}{i} \binom{i}{\alpha_1 \alpha_2 \dots} \left[\int x_1^{\alpha_1} x_2^{\alpha_2} \dots g(x_1, x_2, \dots) dx_1 dx_2 \dots \right]^2.$$

Because the result of the integration is non-negative, the polynomial kernel function therefore satisfies Mercer's theorem. ∎

5.5.5 Characteristics of SVM

SVM has many desirable qualities that make it one of the most widely used classification algorithms. Following is a summary of the general characteristics of SVM:

1. The SVM learning problem can be formulated as a convex optimization problem, in which efficient algorithms are available to find the global minimum of the objective function. Other classification methods, such as rule-based classifiers and artificial neural networks, employ a greedy-based strategy to search the hypothesis space. Such methods tend to find only locally optimum solutions.

2. SVM performs capacity control by maximizing the margin of the decision boundary. Nevertheless, the user must still provide other parameters such as the type of kernel function to use and the cost function C for introducing each slack variable.

3. SVM can be applied to categorical data by introducing dummy variables for each categorical attribute value present in the data. For example, if `Marital Status` has three values {`Single`, `Married`, `Divorced`}, we can introduce a binary variable for each of the attribute values.

4. The SVM formulation presented in this chapter is for binary class problems. Some of the methods available to extend SVM to multiclass problems are presented in Section 5.8.