KNOWLEDGE DISCOVERY WITH SUPPORT VECTOR MACHINES

LUTZ HAMEL
University of Rhode Island

WILEY

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Hamel, Lutz.
Knowledge discovery with support vector machines / Lutz Hamel.
p. cm. — (Wiley series on methods and applications in data mining)
Includes bibliographical references and index.
ISBN 978-0-470-37192-3 (cloth)
1. Support vector machines. 2. Data mining. 3. Machine learning. 4. Computer algorithms.
I. Title.
Q325.5.H38 2009
005.1–dc22 2009011948

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

s groundbreaking paper [64]. Dis-
ıral network and machine learning
ıtations of perceptron learning are
vestigate the computational power
the perceptron, often referred to as
and Schapire [32].

CHAPTER 6

MAXIMUM-MARGIN CLASSIFIERS

The learning approaches to binary classification problems of the preceding chapters can lead to decision surfaces that imply possible misclassification of data points that are not part of the training set. For the simple learning algorithm of Chapter 4, distortions of the decision surface can arise due to outliers. These distortions can lead to misclassifications. In the perceptron learning of Chapter 5, the training algorithm will terminate as soon as a decision surface is found for the training set. The underlying heuristic provides no guarantees that the decision surface constructed generalizes well from the training set to the data universe at large, implying possible misclassifications of points not in the training set.

Here we introduce a new approach that tries to avoid such shortcomings. This approach is based on searching for a decision surface that is equidistant to the class boundaries where the two classes are closest to each other. It also maximizes the distances to these class boundaries. By placing the decision surface right in the middle between the two class boundaries and by maximizing the distances from the class boundaries, this new approach reduces the probability of misclassification. We call such models *maximum-margin classifiers*.

The fact that we are searching for a decision surface with an optimality criterion such as the "maximum distance" implies an optimization problem, and as we will see, constructing a maximum-margin classifier is indeed a convex optimization problem that can be solved via quadratic programming techniques. The training set points that represent the heaviest constraints on the position of such an optimal decision surface, called *support vectors*, are related to the points with large α-values in the dual representation of the perceptron.

Knowledge Discovery with Support Vector Machines, by Lutz Hamel

6.1 OPTIMIZATION PROBLEMS

Optimization problems are problems in which we want to select the best solution from a number of possible or feasible solutions. Typically, the feasible solutions are ranked by an objective function and the goal is to find the feasible solution that minimizes (or maximizes) the value of this function. In most optimization problems we also have a set of constraints that limit the solution space; that is, the constraints place limits on what constitutes a feasible solution and what does not. We can express optimization problems formally as

$$\min_{\overline{x}} \phi(\overline{x}), \tag{6.1}$$

such that

$$h_i(\overline{x}) \geq c_i, \tag{6.2}$$

with $i = 1, \ldots, l$ and for all $\overline{x} \in \mathbb{R}^n$. Here the function $\phi : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*, and each function $h_i : \mathbb{R}^n \to \mathbb{R}$ is called a *constraint* with *bound* c_i. Any value $\overline{x} \in \mathbb{R}^n$ that satisfies the constraints is called a *feasible solution*. The optimization aims to find the feasible solution, $\overline{x}^*$, that minimizes the objective function such that for any other feasible solution $\overline{q} \in \mathbb{R}^n$, we have

$$\phi(\overline{x}^*) \leq \phi(\overline{q}). \tag{6.3}$$

If it is clear over which variable the optimization ranges, we often drop the subscript of the optimization operator.

We have stated optimization problems only in terms of minimization. This is not a limitation since we can turn any maximization problem into a minimization problem using one of the following identities:

$$\max \phi(\overline{x}) = \min -\phi(\overline{x}), \tag{6.4}$$

$$\max \phi(\overline{x}) = \min \frac{1}{\phi(\overline{x})} \tag{6.5}$$

as long as $1/\phi(\bar{x})$ is well-defined. Optimization problems are classified according to the properties of their corresponding objective functions and constraints. For example, a linear optimization problem has both a linear objective function and linear constraints. By this we mean that both the objective function and the constraints represents lines, planes, or hyperplanes in the appropriate dot product spaces.[1] When the objective function or the constraints are not linear, the optimization problem is considered to be nonlinear.

[1]Alternatively, a function is linear if it satisfies equations (3.8) and (3.9) when its graph is translated so that it goes through the origin of the dot product space.

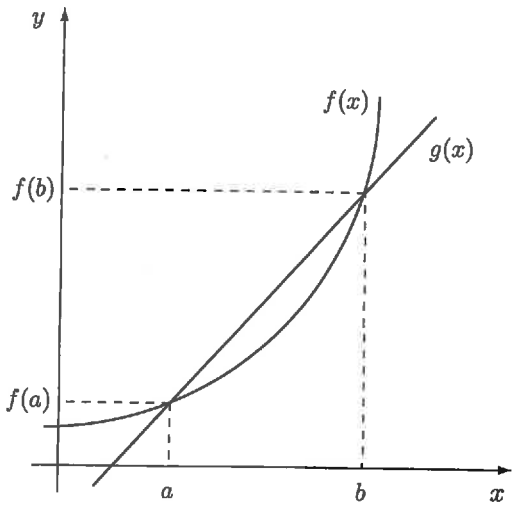

FIGURE 6.1 Convex function.

Here we are concerned with *convex optimization problems*. A convex optimization problem has a convex objective function and linear constraints. Optimization problems that are convex are particularly well behaved in that the objective function has a global minimum and the function surface is smooth in the sense that we can draw a line from one point on the function surface to any other point on the surface without crossing the surface itself. To illustrate this, consider the function $f : \mathbb{R} \rightarrow \mathbb{R}$ in Figure 6.1. Let $a, b \in \mathbb{R}$ be any values with $a < b$, and let $g : \mathbb{R} \rightarrow \mathbb{R}$ be a linear function such that $g(a) = f(a)$ and $g(b) = f(b)$. That is, g is a line that intersects the graph of function f at points $(a, f(a))$ and $(b, f(b))$. We say that the function f is convex if $f(x) \leq g(x)$ for all values $x \in \mathbb{R}$ such that $a < x < b$. For a convex function f, the line g can touch the graph of the function for any interval $[a, b]$ in $\mathbb{R}$ but cannot cross it.

Simple examples of convex functions are functions that raise their argument to some positive, even integer power [e.g., $f(x) = x^2$]. Efficient algorithms exist that take advantage of the convexity of an objective function in order to solve a convex optimization problem. One such technique is quadratic programming, which we discuss in Section 6.4.

6.2 MAXIMUM MARGINS

Given a linearly separable training set for a binary classification problem, it is perhaps intuitive that the optimal decision surface is equidistant from the class boundaries. Informally, we can justify this by arguing that the training set is only an approximate representation of the data universe, and placing the decision surface equidistant from the respective class boundaries will increase the probability of correctly classifying

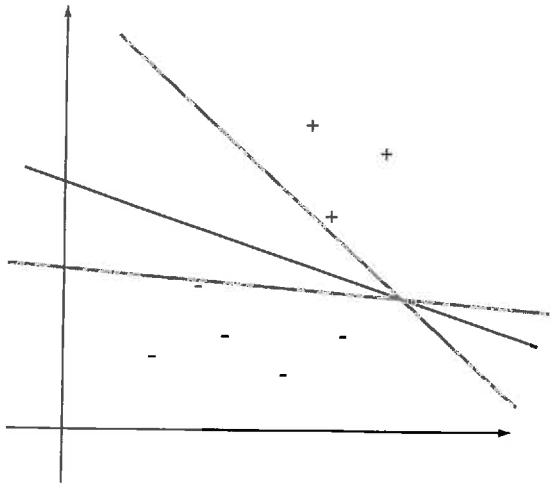

FIGURE 6.2 The dark line represents an optimal decision surface, and the light lines represent suboptimal decision surfaces in a binary classification problem.

points not in the training set. Maximizing the distances from the decision surface to the class boundaries will increase this probability even further.[2] In Figure 6.2 we illustrate this in $\mathbb{R}^2$-space, where the dark line is considered a better decision surface than either one of the light gray lines. To construct such optimal decision surfaces, we need a couple of additional concepts.

Definition 6.1 *A hyperplane supports a class if it is parallel to a (linear) decision surface and all points of its respective class are either above or below. We call such a hyperplane a* ***supporting hyperplane****.*

One way to think of a supporting hyperplane is as the translation of a copy of the decision surface to a point where it just touches the boundary of its respective class. In binary classification problems we typically have two supporting hyperplanes: one that is translated in the direction of the class with the $+1$ label and one that is translated in the direction of the class with the -1 label.

The second concept we need is the margin, which is crucial in this approach to constructing an optimal decision surface.

Definition 6.2 *In a binary classification problem the distance between the two supporting hyperplanes is called a* ***margin****.*

With these two concepts we can state our optimality criterion for a decision surface in more quantitative terms.

[2]We state this here without proof. Later we'll see that this fact holds when we talk about VC-dimensions in the context of statistical learning theory.

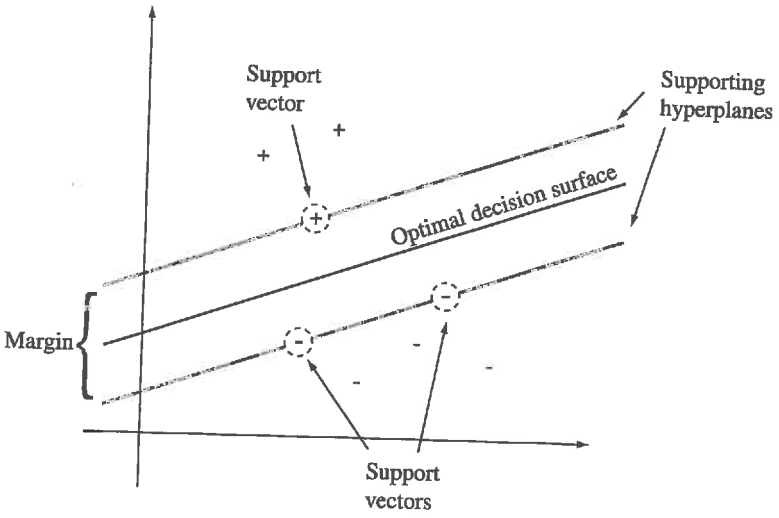

FIGURE 6.3 Optimal separating plane with its two supporting planes.

Definition 6.3 *A decision surface for a binary classification problem is* ***optimal*** *if it is equidistant from the two supporting hyperplanes and maximizes their margin.*

This means that our optimization problem involves finding a decision surface that allows the supporting hyperplanes to be translated as far as possible, thereby maximizing the margin and keeping the decision surface equidistant to the two supporting hyperplanes. Figure 6.3 puts all these concepts together. We have the two supporting hyperplanes translated so they just touch their respective class boundaries. The distance between the hyperplanes is the margin, and the optimal decision surface is located at the center of the margin. Notice that the size of the margin is constrained by the circled points in each class, called *support vectors*. If a supporting hyperplane were to cross a support vector of its respective class, it would no longer be considered a supporting hyperplane because members of its respective class would appear on both sides of the hyperplane. Also notice that the margin is at its maximum. Any rotation or translation of the decision surface would result in a smaller margin. Therefore, the goal in the maximum-margin classifier approach is to find the position of the decision surface that maximizes the margin, as shown here.

6.3 OPTIMIZING THE MARGIN

Finding a decision surface that maximizes the margin between the two supporting hyperplanes is an optimization problem where the feasible solutions are all possible decision surfaces with their associated supporting hyperplanes. Given these feasible solutions, the objective function for this optimization problem computes the size of the margin for each decision surface, and we maximize the objective function to find

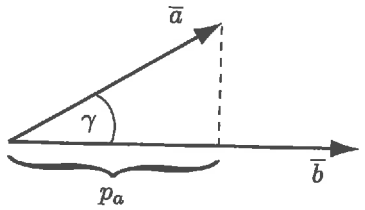

FIGURE 6.4 The value p_a is the projection of $\overline{a}$ in the direction of $\overline{b}$.

the maximum margin. The constraints in this case are the positions of the supporting hyperplanes, which are not allowed to cross their respective class boundaries. We can state this formally as

$$m^* = \max \phi(\overline{w}, b), \tag{6.6}$$

subject to the supporting hyperplane constraints. Here the objective function $\phi(\overline{w}, b)$ computes the margin of a given decision surface $\overline{w} \bullet \overline{x} = b$. The maximum margin m^* is due to some optimal decision surface, call it $\overline{w}^* \bullet \overline{x} = b^*$.

To make this optimization problem computable, we need to derive a suitable expression for the objective function ϕ. It is possible to derive our objective function, with a geometric argument, and to construct our geometric derivation we need the notion of a projection.

Definition 6.4 *Let $\overline{a}$ and $\overline{b}$ be vectors in $\mathbb{R}^n$ that form an angle γ between them; then we say that p_a is the* ***projection*** *of $\overline{a}$ in the direction of $\overline{b}$ such that*

$$p_a = |\overline{a}| \cos \gamma = \frac{\overline{a} \bullet \overline{b}}{|b|}. \tag{6.7}$$

Figure 6.4 shows this projection construction. Here p_a is a scalar that denotes the magnitude of $\overline{a}$ projected in the direction of $\overline{b}$.

We are now in a position to derive our objective function. Let us assume that we have a linearly separable training set

$$D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subseteq \mathbb{R}^n \times \{+1, -1\}. \tag{6.8}$$

Let us also assume that we have the optimal decision surface,

$$\overline{w}^* \bullet \overline{x} = b^*, \tag{6.9}$$

for this training set. Since this decision surface is optimal, the following identities hold:

$$m^* = \phi(\overline{w}^*, b^*) = \max \phi(\overline{w}, b). \tag{6.10}$$

Our maximum margin m^* is computed by the objective function ϕ given the parameters $\overline{w}^*$ and b^* of the optimal decision surface. Furthermore, finding the parameters

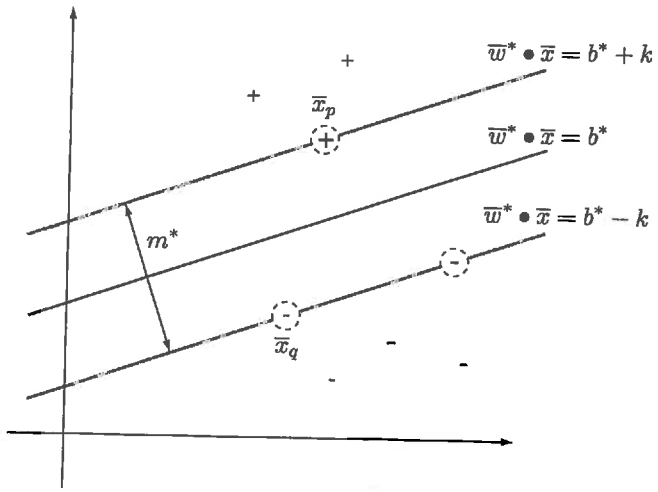

FIGURE 6.5 Margin m^* of the optimal decision surface $\overline{w}^* \bullet \overline{x} = b^*$.

for the optimal decision surface is an optimization problem, as we noted in equation (6.6). Now let us continue with our derivation of ϕ. Since the decision surface in equation (6.9) is a maximum-margin decision surface, we have two supporting hyperplanes equidistant from this surface: say,

$$\overline{w}^* \bullet \overline{x} = b^* + k, \tag{6.11}$$

$$\overline{w}^* \bullet \overline{x} = b^* - k. \tag{6.12}$$

The first hyperplane is the supporting hyperplane for the $+1$ class and is above the decision surface, and the second hyperplane is the supporting hyperplane for the -1 class and is below the decision surface (see Figure 6.5). In addition, since decision surface (6.9) is an optimal decision surface, the supporting hyperplanes are constrained by respective support vectors. Let the point $(\overline{x}_p, +1) \in D$ be a support vector for the $+1$ class with

$$\overline{w}^* \bullet \overline{x}_p = b^* + k. \tag{6.13}$$

That is, the support vector lies on the supporting hyperplane (6.11). Similarly, let $(\overline{x}_q, -1) \in D$ be a support vector for the -1 class with

$$\overline{w}^* \bullet \overline{x}_q = b^* - k. \tag{6.14}$$

This support vector lies on the supporting hyperplane (6.12). See Figure 6.5 for an illustration of this.

By definition, the distance between the two supporting planes is the margin m^*. We can compute this distance as the projection of the vector $\overline{x}_p - \overline{x}_q$ in the direction of $\overline{w}^*$. That is, we can compute the margin as the projection of the difference between

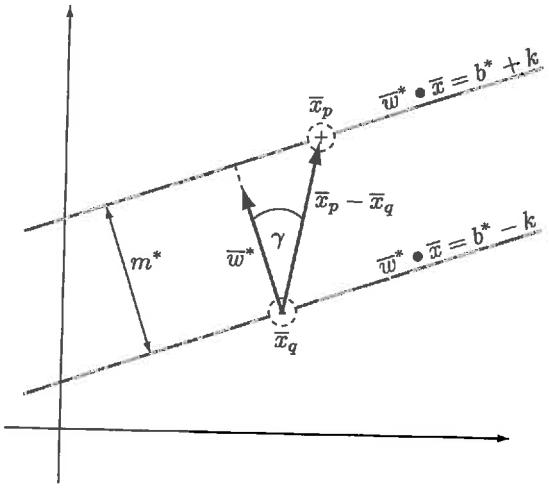

FIGURE 6.6 Computing the margin m^* between two supporting planes.

the two support vectors in the direction of the normal vector of the decision surface. See Figure 6.6 for this construction. In algebraic terms,

$$\begin{aligned} m^* &= |\overline{x}_p - \overline{x}_q| \cos\gamma && \text{by (6.7)} \\ &= \frac{\overline{w}^* \bullet (\overline{x}_p - \overline{x}_q)}{|\overline{w}^*|} && \text{by (6.7)} \\ &= \frac{\overline{w}^* \bullet \overline{x}_p - \overline{w}^* \bullet \overline{x}_q}{|\overline{w}^*|} && \text{by linearity in Table 3.4} \\ &= \frac{(b^* + k) - (b^* - k)}{|\overline{w}^*|} && \text{by (6.13) and (6.14)} \\ &= \frac{2k}{|\overline{w}^*|}. && (6.15) \end{aligned}$$

Here γ is the angle between the vectors $\overline{w}^*$ and $\overline{x}_p - \overline{x}_q$. This gives us the optimization expression,

$$m^* = \max \frac{2k}{|\overline{w}|}. \qquad (6.16)$$

However, we want to express our maximization problem as a minimization problem. We do so by rewriting equation (6.16):

$$\begin{aligned} m^* &= \max \frac{2k}{|\overline{w}|} \\ &= \min \frac{|\overline{w}|}{2k} \end{aligned}$$

$$
\begin{aligned}
&= \min \frac{|\overline{w}|^2}{2k} \\
&= \min \tfrac{1}{2k} \overline{w} \bullet \overline{w} \\
&= \min \tfrac{1}{2} \overline{w} \bullet \overline{w}.
\end{aligned} \tag{6.17}
$$

Here the first step is justified by the identity (6.5) which states that a maximization problem can be viewed as a minimization of the reciprocal of the original objective function. The second step is justified since optimization over the positive values $|\overline{w}|$ is invariant under the transformation with the square function. The square function preserves the order-theoretic properties of its domain; that is, $x_1 \leq x_2$ if and only if $x_1^2 \leq x_2^2$ for $x_1, x_2 \geq 0$. This is another way of saying that x^2 is a monotonic function for $x \geq 0$ and that optimizing over x^2 is therefore the same as optimizing over x. The third step is the application of equation (3.10). The last step is justified since optimization is invariant under scaling with a constant. This means that we are free to pick a convenient value for k, and in our case we chose $k = 1$. This completes the derivation of our objective function as

$$
\phi(\overline{w}, b) = \tfrac{1}{2} \overline{w} \bullet \overline{w}. \tag{6.18}
$$

It is perhaps peculiar that the objective function itself does not have a term b to be optimized. The offset term will, however, play a role in the constraints.

Now let us shift our focus to the constraints of the optimization problem. The notion is that the supporting hyperplanes for the respective classes need to stay supporting hyperplanes during the optimizations. That is, the supporting hyperplanes are not allowed to cross their respective class boundaries. Formally, this means that for our optimal supporting hyperplanes (6.11) and (6.12), the following identities have to hold, respectively:

$$
\overline{w}^* \bullet \overline{x}_i \geq b^* + k \qquad \text{for all } (\overline{x}_i, y_i) \in D \text{ s.t. } y_i = +1, \tag{6.19}
$$

$$
\overline{w}^* \bullet \overline{x}_i \leq b^* - k \qquad \text{for all } (\overline{x}_i, y_i) \in D \text{ s.t. } y_i = -1. \tag{6.20}
$$

In other words, all the training points labeled $+1$ need to lie on or above the first supporting hyperplane, and all the training points labeled -1 need to lie on or below the second supporting hyperplane (see Figure 6.5). Thus, each point in the training data set is a constraint on its respective supporting hyperplane—the supporting hyperplane is not allowed to move beyond it. What needs to hold for the optimal supporting hyperplanes also needs to hold for all supporting hyperplanes for any decision surface. Generalizing and taking to our choice of $k = 1$ into account gives us

$$
\overline{w} \bullet \overline{x}_i \geq 1 + b \qquad \text{for all } (\overline{x}_i, y_i) \in D \text{ s.t. } y_i = +1, \tag{6.21}
$$

$$
\overline{w} \bullet (-\overline{x}_i) \geq 1 - b \qquad \text{for all } (\overline{x}_i, y_i) \in D \text{ s.t. } y_i = -1. \tag{6.22}
$$

We can write these constraints in a more compact way,

$$\overline{w} \bullet (y_i \overline{x}_i) \geq 1 + y_i b \qquad \text{for all } (\overline{x}_i, y_i) \in D. \tag{6.23}$$

In this formulation it is perhaps most obvious that *all* training set points give rise to constraints. The constraints also define the feasible region in that only decision surfaces where the margin fulfills these constraints are considered during optimization. The following proposition on computing an optimal decision surface with a maximum margin summarizes all this, making use of (6.18) and (6.23).

Proposition 6.1 (Maximum-Margin Classifier) *Given a linearly separable training set*

$$D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subseteq \mathbb{R}^n \times \{+1, -1\},$$

we can compute a maximum-margin decision surface $\overline{w}^* \bullet \overline{x} = b^*$ *with an optimization*

$$\min \phi(\overline{w}, b) = \min_{\overline{w}, b} \tfrac{1}{2} \overline{w} \bullet \overline{w} \tag{6.24}$$

subject to the constraints

$$\overline{w} \bullet (y_i \overline{x}_i) \geq 1 + y_i b \qquad \text{for all } (\overline{x}_i, y_i) \in D. \tag{6.25}$$

6.4 QUADRATIC PROGRAMMING

It is easy to see that our objective function in (6.24) is a convex function,

$$\phi(\overline{w}, b) = \tfrac{1}{2} \overline{w} \bullet \overline{w} = \tfrac{1}{2}(w_1^2 + \ldots + w_n^2) \tag{6.26}$$

for $\overline{w} = (w_1, \ldots, w_n)$. The objective function is shown in Figure 6.7 for the two-dimensional space $\mathbb{R}^2$. Convexity implies that we are able to find the global minimum of our objective function. In other words, given a set of feasible solutions, we will be able to find the one that will produce the smallest value of our objective function.

An efficient way to solve convex optimization problems of the form given here is via *quadratic programming*. Most quadratic program solvers are functions of the form

$$\overline{w}^* = \text{solve}(\mathbf{Q}, \overline{q}, \mathbf{X}, \overline{c}), \tag{6.27}$$

which represent the general convex optimization problem, also referred to as a *quadratic program*,

$$\overline{w}^* = \operatorname*{argmin}_{\overline{w}} \left(\tfrac{1}{2} \overline{w}^{\mathrm{T}} \mathbf{Q}\, \overline{w} - \overline{q} \bullet \overline{w} \right), \tag{6.28}$$

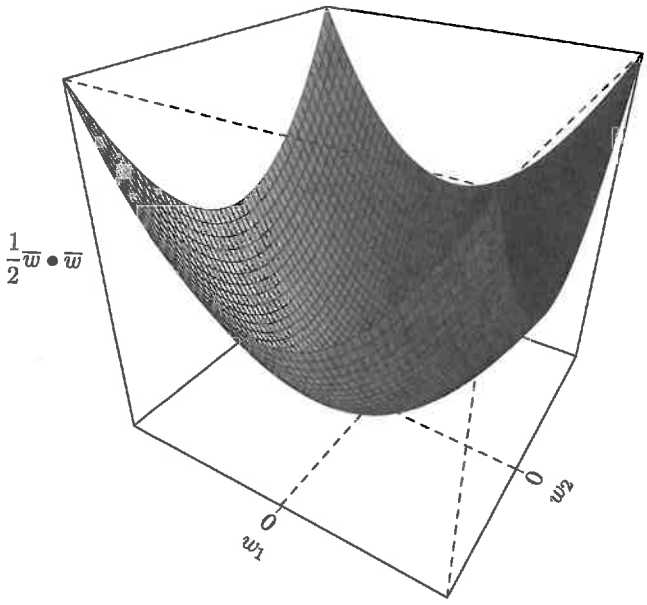

FIGURE 6.7 Objective function $\frac{1}{2}\overline{w} \bullet \overline{w}$ in $\mathbb{R}^2$-space.

subject to the constraints

$$\mathbf{X}^{\mathrm{T}}\overline{w} \geq \overline{c}. \tag{6.29}$$

Here $\mathbf{Q}$ is an $n \times n$ matrix, $\mathbf{X}$ is an $l \times n$ matrix, the vectors $\overline{w}^*$, $\overline{w}$, $\overline{q}$ are n-dimensional vectors, and the vector $\overline{c}$ is an l-dimensional vector. The operation $\overline{a}^{\mathrm{T}}$ denotes the transpose of some vector $\overline{a}$, and $\mathbf{M}^{\mathrm{T}}$ denotes the transpose of a matrix $\mathbf{M}$. We can transform this general optimization problem into a form that resembles our margin optimization in Proposition 6.1 if we let $\mathbf{Q}$ be the identity matrix $\mathbf{I}$ and $\overline{q} = \overline{0}$,

$$\overline{w}^* = \underset{\overline{w}}{\operatorname{argmin}} \left(\tfrac{1}{2}\overline{w}^{\mathrm{T}}\mathbf{I}\,\overline{w} - \overline{0} \bullet \overline{w} \right) = \underset{\overline{w}}{\operatorname{argmin}} \left(\tfrac{1}{2}\overline{w} \bullet \overline{w} \right), \tag{6.30}$$

with $\overline{w}^{\mathrm{T}}\mathbf{I}\,\overline{w} = \overline{w} \bullet \overline{w}$. The big difference between our formulation of margin optimization and the approach using a quadratic program solver is the fact that quadratic program solvers return the argument that minimizes the objective function rather than the minimized value of the objective function (whence the operator argmin rather than min). This does not pose a problem, however, since we can always reconstruct the optimal margin by plugging the optimized normal vector $\overline{w}^*$ into equation (6.15).

Now that we have the objective function of the quadratic program in a suitable form, let us turn our attention to the constraints. In contrast to our original margin optimization problem, the constraints in quadratic program solvers are expressed in matrix form [equation (6.29)]. However, by manipulating the constraints of our original margin optimization problem slightly, we can bring them into a form suitable

for quadratic program solvers. Consider the following form of our constraints obtained by applying the symmetry property of dot products to equation (6.25):

$$(y_i \overline{x}_i) \bullet \overline{w} \geq 1 + y_i b \tag{6.31}$$

for all $(\overline{x}_i, y_i) \in D$ with $i = 1, \ldots, l$ and $\overline{x}_i = (x_i^1, \ldots, x_i^n)$. Given this, we can construct the matrix $\mathbf{X}$ as

$$\mathbf{X} = \begin{pmatrix} y_1 x_1^1 & \cdots & y_i x_i^1 & \cdots & y_l x_l^1 \\ \vdots & & \vdots & & \vdots \\ y_1 x_1^n & \cdots & y_i x_i^n & \cdots & y_l x_l^n \end{pmatrix}. \tag{6.32}$$

In other words, the ith column of $\mathbf{X}$ is equal to the vector $y_i \overline{x}_i = (y_i x_i^1, \ldots, y_i x_i^n)$. We construct the vector $\overline{c}$ as

$$\overline{c} = \begin{pmatrix} 1 + y_1 b \\ 1 + y_2 b \\ \vdots \\ 1 + y_l b \end{pmatrix}. \tag{6.33}$$

With these two constructions, it is now straightforward to show that the matrix format of the constraints in (6.29) is a compact representation of the constraint equations (6.31).

Notice that our construction of the vector $\overline{c}$ introduced the free variable b into the quadratic program, so we need to take this free variable into account in our optimization problem. That is, our optimization has to minimize the objective function over both $\overline{w}$ and b. The following proposition summarizes these constructions and expresses a maximum-margin optimization in terms of a optimization problem solvable via quadratic programming,

Proposition 6.2 *Given the linearly separable training set*

$$D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subseteq \mathbb{R}^n \times \{+1, -1\},$$

we can compute a maximum-margin decision surface $\overline{w}^* \bullet \overline{x} = b^*$ *with a quadratic programming approach that solves the generalized optimization problem*

$$(\overline{w}^*, b^*) = \underset{\overline{w}, b}{\operatorname{argmin}} \left(\tfrac{1}{2} \overline{w}^{\mathrm{T}} \mathbf{Q}\, \overline{w} - \overline{q} \bullet \overline{w} \right). \tag{6.34}$$

subject to the constraints

$$\mathbf{X}^{\mathrm{T}} \overline{w} \geq \overline{c}, \tag{6.35}$$

with $\mathbf{Q} = \mathbf{I}$, $\overline{q} = \overline{0}$, *and where* $\mathbf{X}$, *and* $\overline{c}$ *are constructed according to* (6.32) *and* (6.33), *respectively.*

Algorithm 6.1

```
let D = {(x̄1, y1), (x̄2, y2), ..., (x̄l, yl)} ⊂ R^n × {+1, −1}
r ← max{|x̄| | (x̄, y) ∈ D}
q ← 1000
let w̄* and b* be undefined
Construct X according to (6.32) using D.
for each b ∈ [−q, q] do
    Construct c̄ according to (6.33) using b.
    w̄ ← solve(I, 0̄, X, c̄)
    if (w̄ is defined and w̄* is undefined) or
       (w̄ is defined and |w̄| < |w̄*|) then
       w̄* ← w̄
       b* ← b
    end if
end for
if w̄* is undefined then
    stop constraints not satisfiable
else if |w̄|* > q/r then
    stop bounding assumption of |w̄| violated
end if
return (w̄*, b*)
```

Algorithm 6.1 illustrates how a decision surface with a maximum margin is computed using a quadratic program solver. Here the function *solve* is assumed to be of the form (6.27). Most quadratic program solvers will return an *undefined* value for $\overline{w}$ if they cannot find a solution that satisfies all the constraints. Therefore, we see repeated tests in the algorithm as to whether or not the solver was successful in finding a solution. The quantity r represents the radius of the training set D. The constant q defines the size of the search interval for offset term values, and we set it to 1000. However, the precise value for q is highly data dependent and needs to be determined experimentally. We discuss this further below. The optimal normal vector $\overline{w}^*$ and offset term b^* are left undefined initially. Notice that $\overline{w}^*$ and b^* will remain undefined until the algorithm finds a solution that satisfies all the constraints.

Since the offset term b is a free variable, we need to pick appropriate values for b to be passed to the solver as part of the constraints. To determine a reasonable interval of values for b during optimization, consider the representation of a decision surface by the equation

$$b = \overline{w} \bullet \overline{x}. \tag{6.36}$$

This can be rewritten as

$$b = |\overline{w}||\overline{x}| \cos \gamma, \tag{6.37}$$

where γ is the angle between the vectors $\overline{w}$ and $\overline{x}$. With $0 \leq \gamma \leq \pi$ we have

$$-|\overline{w}||\overline{x}| \leq b \leq |\overline{w}||\overline{x}|. \tag{6.38}$$

We only consider points that lie within a hypersphere of radius r; that is, we only consider points with $|\overline{x}| \leq r$:

$$-|\overline{w}|r \leq b \leq |\overline{w}|r. \tag{6.39}$$

Unfortunately, $|\overline{w}|$ is unbounded, making the bound on b as stated useless. We can show this by considering that in a training set with radius r the largest possible margin is $2r$. Plugging this observation into equation (6.15), we obtain

$$\frac{2}{|\overline{w}|} \leq 2r. \tag{6.40}$$

We assume a value of $k = 1$, as before. This gives us a lower bound for $\overline{w}$:

$$\frac{1}{r} \leq |\overline{w}|. \tag{6.41}$$

This means that $|\overline{w}| = 1/r$ for the largest possible margin and $|\overline{w}| > 1/r$ for margins smaller than that. In fact, we have $|\overline{w}| \to \infty$ for infinitesimally small margins. However, we are interested in maximizing the margin; that is, decision surfaces with margins smaller than a certain threshold are not interesting enough to be considered part of the feasible solutions. To express this, we bound the values of $|\overline{w}|$ as follows:

$$\frac{1}{r} \leq |\overline{w}| \leq \frac{q}{r}, \tag{6.42}$$

where q is a bounding constant that bounds the value of $|\overline{w}|$ to a multiple of the maximum margin $1/r$. If we pick $q = 1000$ as it appears in the algorithm, the narrowest margin we consider a solution is 1000 times narrower than the maximum margin possible in a training set with radius r. Plugging (6.42) into (6.39) gives us

$$-q \leq b \leq q, \tag{6.43}$$

the bound for b as it appears in the algorithm.

The algorithm can fail to compute a decision surface on two accounts. The first type of failure arises when the solver cannot satisfy all the given constraints for any b in the given interval. The second type of failure arises when our bounding assumption for $|\overline{w}|$ is violated. Since we assumed a linearly separable training set, we are guaranteed to find a solution, and the failures imply that our bounding assumptions on b are not correct and the interval of values for b needs to be increased.

6.5 DISCUSSION

By defining the maximum margin as an optimality criterion for decision surfaces, we have achieved our goal of preventing degenerate decision surfaces from being

considered as models for binary classification problems. However, when actually computing such maximum-margin classifiers using quadratic program solvers, we found that our solutions depend heavily on how we pick the free offset term parameter. Even though we have given some guidelines on how to search for the value of the offset term that leads to a maximum margin, the precise value can only be determined through experimentation. Searching for an optimal value of a free model parameter is not unusual in machine learning. Many complex learning algorithms have free parameters that need to be estimated via experimentation with the training set. Consider the optimal topology in artificial neural networks or the optimal pruning constant in decision tree learning algorithms. A marked exception is the dual of the maximum-margin algorithm we considered here. This dual algorithm also constructs a maximum-margin classifier but has no free parameters. We call this dual algorithm a *linear support vector machine*, and we develop this algorithm in Chapter 7.

EXERCISES

6.1 Write a program in R that implements the algorithm given in Algorithm 6.1 and apply it to your favorite linearly separable data set that represents a binary classification problem. (Use the quadratic program solver provided in the package *quadprog*.)

6.2 Implement the following learning algorithms in R:

1. Simple learning given in equation (4.25)
2. Perceptron learning, Algorithm 5.1
3. Maximum margin classifier, Algorithm 6.1

and train them with the training set from Exercise 4.1.

(a) Plot the data set together with the induced decision surface for each algorithm.

(b) What can you say about the differences in the respective induced decision surfaces?

(c) For each of the learning algorithms above, use the respective decision function $\hat{f}$ and classify the point $\overline{x} = \langle 2, 2 \rangle$.

6.3 [*challenging*] Optimal decision surfaces can also be developed via convex hulls of the respective classes in a linearly separable binary classification setting. Briefly, the optimal decision surface bisects the minimum distance of the two respective convex hulls at a right angle (see [5] and [6]).

(a) Write an algorithm that computes the optimal decision surface based on convex hulls.

(b) Show graphically that the decision surfaces computed by the convex hull algorithm and the maximum margin algorithm in Algorithm 6.1 coincide.