# Chapter 23

# Support Vector Machines

In this chapter we describe Support Vector Machines (SVM), a classification method based on maximum margin linear discriminants.

## 23.1 Linear Discriminants and Margins

Let $\mathbf{D}$ be a classification dataset, with $n$ points in $d$-dimensional space: $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. Further, let us assume that there are only two class labels, i.e., $y_i \in \{+1, -1\}$, denoting the positive and negative classes.

**Hyperplanes** A linear discriminant function in $d$ dimensions is given by a hyperplane, defined as follows

$$\begin{aligned} h(\mathbf{x}) &= \mathbf{w}^T\mathbf{x} + b \qquad (23.1)\\ &= w_1x_1 + w_2x_2 + \cdots + w_dx_d + b \end{aligned}$$

Here, $\mathbf{w}$ is a $d$ dimensional *weight vector*, and $b$ is a scalar, called the *bias*. For points that lie on the hyperplane, we have

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0 \qquad (23.2)$$

In other words, the hyperplane is defined as the set of all points such that $\mathbf{w}^T\mathbf{x} = -b$. To see the role played by $b$, assuming that $w_1 \neq 0$, and setting $x_i = 0$ for all $i > 1$, we can obtain the offset where the hyperplane intersects the first axis, since by (23.2), we have

$$w_1x_1 = -b \quad \text{or} \quad x_1 = \frac{-b}{w_1} \qquad (23.3)$$

In other words, the point $(\frac{-b}{w_1}, 0, \cdots, 0)$ lies on the hyperplane. In a similar manner, we can obtain the offset where the hyperplane intersects each of the axes, given by $\frac{-b}{w_i}$ (provided $w_i \neq 0$).

**Separating Hyperplane** A hyperplane $h(\mathbf{x})$ splits the original $d$-dimensional space into two *half-spaces*. If the input dataset is linearly separable, then we can find a *separating* hyperplane $h(\mathbf{x})$, such that for all points labeled $y_i = -1$, we have $h(\mathbf{x}_i) < 0$, and for all points labeled $y_i = +1$, we have $h(\mathbf{x}_i) > 0$. In fact, for any given point $\mathbf{x}$, $h(\mathbf{x})$ serves as a linear classifier or a linear discriminant, which predicts the class $y$ for any point $\mathbf{x}$, according to the decision rule

$$y = \begin{cases} +1 & \text{if } h(\mathbf{x}) > 0 \\ -1 & \text{if } h(\mathbf{x}) < 0 \end{cases} \tag{23.4}$$

Let $\mathbf{a}_1$ and $\mathbf{a}_2$ be two arbitrary points that lie on the hyperplane. From (23.2) we have

$$h(\mathbf{a}_1) = \mathbf{w}^T\mathbf{a}_1 + b = 0$$
$$h(\mathbf{a}_2) = \mathbf{w}^T\mathbf{a}_2 + b = 0$$

Subtracting one from the other we obtain

$$\mathbf{w}^T(\mathbf{a}_1 - \mathbf{a}_2) = 0 \tag{23.5}$$

This means that the weight vector $\mathbf{w}$ is orthogonal to the hyperplane, since it is orthogonal to any arbitrary vector $(\mathbf{a}_1 - \mathbf{a}_2)$ on the hyperplane. In other words, the weight vector $\mathbf{w}$, gives the direction that is normal to the hyperplane, which fixes the orientation of the hyperplane, whereas the bias $b$ fixes the offset of hyperplane, in the $d$-dimensional space. Since both $\mathbf{w}$ and $-\mathbf{w}$ are normal to the hyperplane, we remove this ambiguity by requiring that $h(\mathbf{x}_i) > 0$ when $y_i = 1$, and $h(\mathbf{x}_i) < 0$ when $y_i = -1$.

**Distance of a Point to the Hyperplane** Consider a point $\mathbf{x} \in \mathbb{R}^d$, such that $\mathbf{x}$ does not lie on the hyperplane. Further let $\mathbf{x}_p$ be the projection of $\mathbf{x}$ on the hyperplane. Let $r$ be the offset of $\mathbf{x}$ along the weight vector $\mathbf{w}$, then as shown in Figure 23.1, we can write $\mathbf{x}$ as a combination of two vectors

$$\mathbf{x} = \mathbf{x}_p + \mathbf{r}$$
$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|} \tag{23.6}$$

Here $r$ gives the *directed distance* of the point $\mathbf{x}$ from $\mathbf{x}_p$, i.e., $r$ gives the offset of $\mathbf{x}$ from $\mathbf{x}_p$ in terms of the unit weight vector $\frac{\mathbf{w}}{\|\mathbf{w}\|}$. It should be clear that $r$ is positive if $\mathbf{r}$ is in the same direction as $\mathbf{w}$, and $r$ is negative if $\mathbf{r}$ is in a direction opposite to $\mathbf{w}$. $r$ thus gives the offset or directed distance in multiples of the unit weight vector.

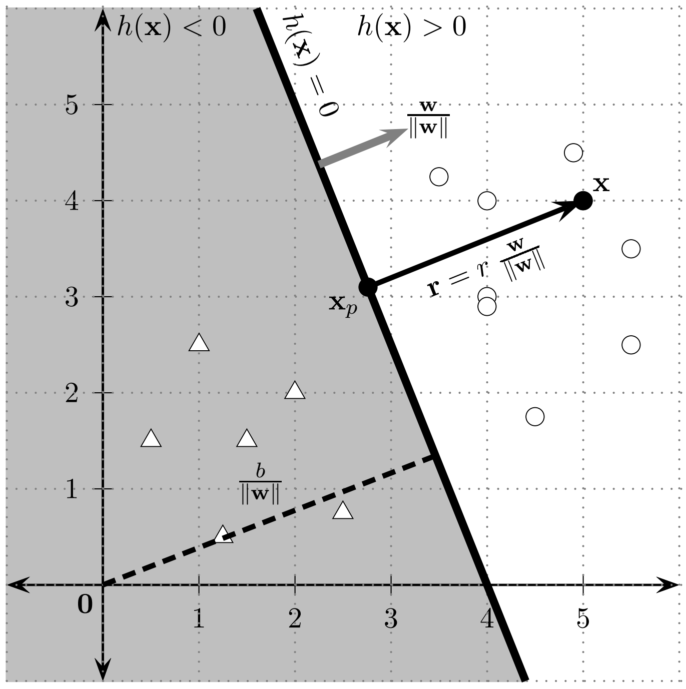
$h(\mathbf{x}) < 0$
$h(\mathbf{x}) > 0$
$h(\mathbf{x}) = 0$
$\frac{\mathbf{w}}{\|\mathbf{w}\|}$
$\mathbf{x}$
$\mathbf{r} = r\frac{\mathbf{w}}{\|\mathbf{w}\|}$
$\mathbf{x}_p$
$\frac{b}{\|\mathbf{w}\|}$
$\mathbf{0}$
1
2
3
4
5

Figure 23.1: Geometry of the Separating Hyperplane in 2D. Points labeled +1 are shown as circles, and those labeled −1 are shown as triangles. The hyperplane $h(\mathbf{x}) = 0$ divides the space into two half-spaces. The shaded region consists of all points $\mathbf{x}$ satisfying $h(\mathbf{x}) < 0$, whereas the unshaded region consists of all points satisfying $h(\mathbf{x}) > 0$. The unit weight vector $\frac{\mathbf{w}}{\|\mathbf{w}\|}$ (in gray) is orthogonal to the hyperplane. The directed distance of the origin to the hyperplane is given as $\frac{b}{\|\mathbf{w}\|}$

Plugging in (23.6) from above in the equation for the hyperplane (23.1)), we get

$$\begin{aligned}
h(\mathbf{x}) &= h(\mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}) \\
&= \mathbf{w}^T\left(\mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + b \\
&= \underbrace{\mathbf{w}^T\mathbf{x}_p + b}_{h(\mathbf{x}_p)} + r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|} \\
&= \underbrace{h(\mathbf{x}_p)}_{0} + r\|\mathbf{w}\| \\
&= r\|\mathbf{w}\| \qquad (23.7)
\end{aligned}$$

The last step follows from the fact that $h(\mathbf{x}_p) = 0$, since $\mathbf{x}_p$ lies on the hyperplane.

From (23.7), we obtain an expression for the directed distance of the point to the hyperplane, given as

$$r = \frac{h(\mathbf{x})}{\|\mathbf{w}\|} \tag{23.8}$$

To obtain distance, which must be non-negative, we can conveniently multiply $r$ by the label $y$ of the point, since when $h(\mathbf{x}) < 0$, the label is $-1$, and when $h(\mathbf{x}) > 0$ the label is $+1$. The distance of a point $\mathbf{x}$ from the hyperplane $h(\mathbf{x})$ is thus given as

$$\delta = y\, r = \frac{y\, h(\mathbf{x})}{\|\mathbf{w}\|} \tag{23.9}$$

In particular, for the origin $\mathbf{x} = \mathbf{0}$, the directed distance is

$$r = \frac{h(\mathbf{0})}{\|\mathbf{w}\|} = \frac{\mathbf{w}^T\mathbf{0} + b}{\|\mathbf{w}\|} = \frac{b}{\|\mathbf{w}\|}$$

as illustrated in Figure 23.1, and the distance is

$$\delta = y\, r = -1\, r = \frac{-b}{\|\mathbf{w}\|}$$

**Example 23.1:** Consider the example shown in Figure 23.1. In this two-dimensional example, the hyperplane is just a line, which is defined as the set of all points $\mathbf{x} = (x_1, x_2)$ that satisfy the following equation

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = w_1x_1 + w_2x_2 + b = 0$$

Rearranging the terms we get

$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2}$$

where $-\frac{w_1}{w_2}$ is the slope of the line, and $-\frac{b}{w_2}$ is the offset along the second dimension.

Given any two points on the hyperplane, say $\mathbf{p} = (p_1, p_2) = (4, 0)$, and $\mathbf{q} = (q_1, q_2) = (2, 5)$, the slope is given as

$$-\frac{w_1}{w_2} = \frac{q_2 - p_2}{q_1 - p_1} = \frac{5 - 0}{2 - 4} = -\frac{5}{2}$$

which implies that $w_1 = 5$ and $w_2 = 2$.

Given any point on the hyperplane, say $(4, 0)$, we can compute the offset $b$ directly as follows

$$\begin{aligned} b &= -5\, x_1 - 2\, x_2 \\ &- 5 \cdot 4 - 2 \cdot 0 = -20 \end{aligned}$$

Thus $\mathbf{w} = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$ is the weight vector, and $b = -20$ is the bias, and the equation of the hyperplane is given as

$$\mathbf{w}^T\mathbf{x} + b = \begin{pmatrix} 5 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 20 = 0$$

One can verify that the origin $\mathbf{0}$ is at a distance of $\frac{-b}{\|\mathbf{w}\|} = \frac{-(-20)}{\sqrt{29}} = 3.71$ from the hyperplane.

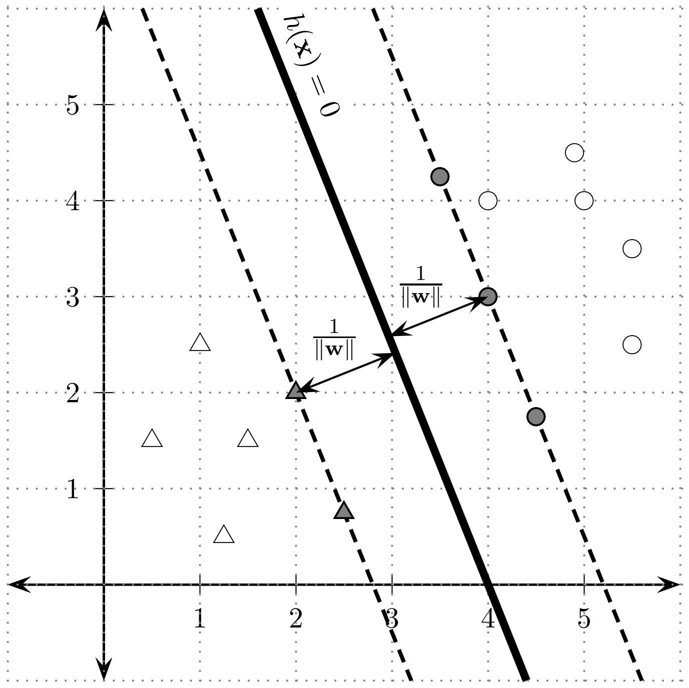

Figure 23.2: Margin of a Separating Hyperplane: $\frac{1}{\|\mathbf{w}\|}$ is the margin, and the shaded points are the support vectors.

**Margin and Support Vectors of a Hyperplane** Given a training dataset of labeled points, $\mathbf{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ with $y_i \in \{+1, -1\}$, and given a separating hyperplane $h(\mathbf{x}) = 0$, for each point $\mathbf{x}_i$ we can find its distance to the hyperplane by (23.9)

$$\delta_i = \frac{y_i\, h(\mathbf{x}_i)}{\|\mathbf{w}\|} = \frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{\|\mathbf{w}\|} \tag{23.10}$$

Over all the $n$ points, we define the *margin* of the linear classifier as the minimum distance of a point from the separating hyperplane, given as

$$\delta^* = \min_{\mathbf{x}_i} \left\{ \frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{\|\mathbf{w}\|} \right\} \tag{23.11}$$

Note that $\delta^* \neq 0$, since $h(\mathbf{x})$ is assumed to be a separating hyperplane, and (23.4) must be satisfied.

All the points (or vectors) that achieve this minimum distance are also called *support vectors* for the linear classifier. In other words, a support vector, $\mathbf{x}^*$, is a point that lies precisely on the margin of the classifier, and it satisfies the following equation

$$\delta^* = \frac{y^*(\mathbf{w}^T\mathbf{x}^* + b)}{\|\mathbf{w}\|} \tag{23.12}$$

Here, the numerator $y^*(\mathbf{w}^T\mathbf{x}^* + b)$ gives the absolute distance of the support vector to the hyperplane, whereas the denominator $\|\mathbf{w}\|$ makes it a relative distance in terms of $\mathbf{w}$.

**Canonical Hyperplane** Consider the equation of the hyperplane (23.2). It is clear that multiplying on both sides by some scalar $s$ yields an equivalent hyperplane

$$s\ h(\mathbf{x}) = s\ \mathbf{w}^T\mathbf{x} + s\ b = (s\mathbf{w})^T\mathbf{x} + (sb) = 0$$

To obtain the unique or *canonical* hyperplane equation, we choose the scalar $s$ such that the absolute distance of a support vector from the hyperplane is 1. That is,

$$sy^*(\mathbf{w}^T\mathbf{x}^* + b) = 1 \tag{23.13}$$

$$s = \frac{1}{y^*(\mathbf{w}^T\mathbf{x}^* + b)} = \frac{1}{y^*h(\mathbf{x}^*)} \tag{23.14}$$

Henceforth, we will assume that any separating hyperplane is canonical. That is, it has already been suitably rescaled so that $y^*h(\mathbf{x}^*) = 1$ for a support vector $\mathbf{x}^*$, and the margin is given as

$$\delta^* = \frac{1}{\|\mathbf{w}\|} \tag{23.15}$$

For the canonical hyperplane, for each support vector $\mathbf{x}_i^*$ (with label $y_i^*$), we have, $y_i^*h(\mathbf{x}_i^*) = 1$, and for any point that is not a support vector, we have $y_ih(\mathbf{x}_i) > 1$, since, by definition, it must be farther from the hyperplane than a support vector. Over all the $n$ points in the dataset $\mathbf{D}$, we thus obtain the following set of inequalities

$$y_i\ (\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \text{ for all points } \mathbf{x}_i \in \mathbf{D} \tag{23.16}$$

Figure 23.2 gives an illustration of the support vectors and the margin of a hyperplane.

**Example 23.2:** Consider the separating hyperplane shown in Figure 23.2, given by the equation

$$h'(\mathbf{x}) = \begin{pmatrix}5\\2\end{pmatrix}^T \mathbf{x} - 20 = 0$$

Consider the support vector $\mathbf{x}^* = (2, 2)$, with class $y^* = -1$. To find the canonical hyperplane equation, we have to rescale the weight vector and bias by the scalar $s$, obtained using (23.14)

$$s = \frac{1}{y^* h'(\mathbf{x}^*)} = \frac{1}{-1\left(\begin{pmatrix}5\\2\end{pmatrix}^T \begin{pmatrix}2\\2\end{pmatrix} - 20\right)} = \frac{1}{6}$$

Thus the rescaled weight vector is

$$\mathbf{w} = \frac{1}{6}\begin{pmatrix}5\\2\end{pmatrix} = \begin{pmatrix}5/6\\2/6\end{pmatrix}$$

and the rescaled bias is

$$b = \frac{-20}{6}$$

The canonical form of the hyperplane is thus given as

$$h(\mathbf{x}) = \begin{pmatrix}5/6\\2/6\end{pmatrix}^T \mathbf{x} - 20/6 = \begin{pmatrix}0.833\\0.333\end{pmatrix}^T \mathbf{x} - 3.33 \tag{23.17}$$

The margin of the canonical hyperplane is given as

$$\delta^* = \frac{y^*\ h(\mathbf{x}^*)}{\|\mathbf{w}\|} = \frac{-1\left(\begin{pmatrix}5/6\\2/6\end{pmatrix}^T \begin{pmatrix}2\\2\end{pmatrix} - 20/6\right)}{\sqrt{(\frac{5}{6})^2 + (\frac{2}{6})^2}} = \frac{1}{\frac{\sqrt{29}}{6}} = 1.114$$

In this example there are five support vectors, namely, $(2, 2)$ and $(2.5, 0.75)$ with class $y = -1$ (shown as triangles), and $(3.5, 4.25)$, $(4, 3)$, and $(4.5, 1.75)$ with class $y = +1$ (shown as circles), as illustrated in Figure 23.2.

## 23.2 SVM: Linear and Separable Case

Given a dataset $\mathbf{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{+1, -1\}$, let us assume for the moment that the points are linearly separable, i.e., there exists a hyperplane that perfectly classifies each point. In other words, all points labeled $y_i = +1$ lie on one side ($h(\mathbf{x}) > 0$) and all points labeled $y_i = -1$ lie on the other side ($h(\mathbf{x}) < 0$) of the hyperplane. It is obvious that in the linearly separable case, there are in fact an infinite number of such separating hyperplanes. Which one should we choose?

**Maximum Margin Hyperplane:** The fundamental idea behind SVMs is to choose the canonical hyperplane, specified by the weight vector $\mathbf{w}$ and the bias $b$, that yields the maximum margin among all possible separating hyperplanes $h(\mathbf{x}) \equiv \mathbf{w}^T\mathbf{x} + b = 0$. If $\delta_h^*$ represents the margin for hyperplane $h(\mathbf{x}) = 0$, then the goal is to find the optimal hyperplane $h^*$

$$h^* = \arg\max_h \{\delta_h^*\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \right\} \tag{23.18}$$

The SVM task is to find the hyperplane that maximizes the margin $\frac{1}{\|\mathbf{w}\|}$, subject to the $n$ constraints given in (23.16), namely, $y_i\ (\mathbf{w}^T\mathbf{x}_i + b) \geq 1$, for all points $\mathbf{x}_i \in \mathbf{D}$.

Notice that instead of maximizing the margin $\frac{1}{\|\mathbf{w}\|}$, we obtain an equivalent formulation if we minimize $\|\mathbf{w}\|$. In fact, we can obtain an equivalent minimization formulation given as follows

$$\begin{aligned} \textbf{Objective Function: } & \min_{\mathbf{w},b} \left\{ \frac{\|\mathbf{w}\|^2}{2} \right\} \\ \textbf{Linear Constraints: } & y_i\ (\mathbf{w}^T\mathbf{x}_i + b) \geq 1,\ \ \forall \mathbf{x}_i \in \mathbf{D} \end{aligned} \tag{23.19}$$

We can directly solve the above *primal* convex minimization problem with linear constraints using standard optimization algorithms, as outlined in Section 23.5. However, it is more common to solve the *dual* problem, which is obtained via the use of *Lagrange multipliers*. The main idea is to introduce a Lagrange multiplier $\alpha_i$ for each constraint, based on the Karush-Kuhn-Tucker (KKT) conditions

$$\begin{aligned} \alpha_i \left( y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right) &= 0 \\ \text{and } \alpha_i &\geq 0 \end{aligned} \tag{23.20}$$

Incorporating all the $n$ constraints, the new objective function, called the *Lagrangian*, then becomes

$$\min\ L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i \left( y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right) \tag{23.21}$$

$L$ should be minimized with respect to $\mathbf{w}$ and $b$, and it should be maximized with respect to $\alpha_i$.

Taking the derivative of $L$ with respect to $\mathbf{w}$, and $b$ and setting those to zero, we obtain

$$\frac{\partial}{\partial \mathbf{w}} L = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \mathbf{0} \quad \text{or} \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{23.22}$$

$$\frac{\partial}{\partial b} L = \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{23.23}$$

The above equations give important intuition about the optimal weight vector $\mathbf{w}$. Namely, $\mathbf{w}$ can be expressed as a linear combination of the data points $\mathbf{x}_i$, with the signed Lagrange multipliers, $\alpha_i y_i$, serving as the coefficients. Furthermore, the sum of the signed Lagrange multipliers $(\alpha_i y_i)$ must be zero.

Plugging these into (23.21), we obtain the *dual Lagrangian* objective function, which is purely in terms of the Lagrange multipliers, as follows

$$\begin{aligned} L_{dual} &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T \underbrace{\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right)}_{\mathbf{w}} - b \underbrace{\sum_{i=1}^{n} \alpha_i y_i}_{0} + \sum_{i=1}^{n} \alpha_i \\ &= -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \end{aligned} \tag{23.24}$$

The dual objective is thus given as

$$\begin{aligned} &\textbf{Objective Function: } \max_{\boldsymbol{\alpha}} \; L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ &\textbf{Linear Constraints: } \alpha_i \geq 0 \;\; \forall i \in \mathbf{D}, \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned} \tag{23.25}$$

$L_{dual}$ is a convex quadratic programming problem (note the $\alpha_i \alpha_j$ terms), which can be solved using standard optimization techniques. See Section 23.5 for a gradient-based method for solving the dual formulation.

**Weight Vector and Bias:** Once we have obtained the $\alpha_i$ values for $i = 1, \cdots, n$, we can solve for the weight vector $\mathbf{w}$ and the bias $b$. Note first that according to the KKT condition (23.20), we have

$$\alpha_i \left( y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right) = 0 \tag{23.26}$$

which gives rise to two cases

i) $\alpha_i = 0$, or

ii) $y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 = 0$, which gives $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$

This is a very important result, since if $\alpha_i > 0$, then $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$, i.e., the point $\mathbf{x}_i$ must be a support vector. On the other hand if $y_i(\mathbf{w}^T\mathbf{x}_i + b) > 1$, then $\alpha_i = 0$, i.e., if a point is not a support vector, then $\alpha_i = 0$.

Once we know $\alpha_i$ for all points, we can compute the weight vector $\mathbf{w}$ using (23.22), by taking the summation only for the support vectors

$$\mathbf{w} = \sum_{i,\alpha_i>0} \alpha_i y_i \mathbf{x}_i \tag{23.27}$$

In other words, $\mathbf{w}$ is obtained as a linear combination of the support vectors, with the $\alpha_i$'s representing the weights. The vast majority of points that are not support vectors have $\alpha_i = 0$, and thus do not play a role in determining $\mathbf{w}$.

To compute the bias $b$, we first compute one solution $b_i$, per support vector, as follows

$$\begin{aligned} \alpha_i \big(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\big) &= 0 \\ y_i(\mathbf{w}^T\mathbf{x}_i + b) &= 1 \\ b_i = \frac{1}{y_i} - \mathbf{w}^T\mathbf{x}_i &= y_i - \mathbf{w}^T\mathbf{x}_i \end{aligned} \tag{23.28}$$

We can take $b$ as the average bias value over all the support vectors

$$b = \text{avg}_{\alpha_i>0}\{b_i\} \tag{23.29}$$

**SVM Classifier:** Our final SVM model is given as follows. For any new point $\mathbf{z}$, we predict the class as

$$\hat{y} = sign(\mathbf{w}^T\mathbf{z} + b) \tag{23.30}$$

where the $sign(\cdot)$ function returns $+1$ if its argument is positive, and $-1$ if its argument is negative.

**Example 23.3:** Let us continue with the example dataset shown in Figure 23.2. The dataset has 14 points as shown in Table 23.1.

Solving the $L_{dual}$ quadratic program yields the following values for the Lagrangian multipliers for the support vectors

| $\mathbf{x}_i$ | $x_{i1}$ | $x_{i2}$ | $y_i$ | $\alpha_i$ |
|---|---|---|---|---|
| $\mathbf{x}_1$ | 3.5 | 4.25 | $+1$ | 0.0437 |
| $\mathbf{x}_2$ | 4 | 3 | $+1$ | 0.2162 |
| $\mathbf{x}_4$ | 4.5 | 1.75 | $+1$ | 0.1427 |
| $\mathbf{x}_{13}$ | 2 | 2 | $-1$ | 0.3589 |
| $\mathbf{x}_{14}$ | 2.5 | 0.75 | $-1$ | 0.0437 |

| $\mathbf{x}_i$ | $x_{i1}$ | $x_{i2}$ | $y_i$ |
|---|---|---|---|
| $\mathbf{x}_1$ | 3.5 | 4.25 | $+1$ |
| $\mathbf{x}_2$ | 4 | 3 | $+1$ |
| $\mathbf{x}_3$ | 4 | 4 | $+1$ |
| $\mathbf{x}_4$ | 4.5 | 1.75 | $+1$ |
| $\mathbf{x}_5$ | 4.9 | 4.5 | $+1$ |
| $\mathbf{x}_6$ | 5 | 4 | $+1$ |
| $\mathbf{x}_7$ | 5.5 | 2.5 | $+1$ |
| $\mathbf{x}_8$ | 5.5 | 3.5 | $+1$ |
| $\mathbf{x}_9$ | 0.5 | 1.5 | $-1$ |
| $\mathbf{x}_{10}$ | 1 | 2.5 | $-1$ |
| $\mathbf{x}_{11}$ | 1.25 | 0.5 | $-1$ |
| $\mathbf{x}_{12}$ | 1.5 | 1.5 | $-1$ |
| $\mathbf{x}_{13}$ | 2 | 2 | $-1$ |
| $\mathbf{x}_{14}$ | 2.5 | 0.75 | $-1$ |

Table 23.1: Dataset corresponding to Figure 23.2

All other points are not support vectors, so they have $\alpha_i = 0$. Using (23.27) we can compute the weight vector for the hyperplane

$$\begin{aligned}\mathbf{w} &= \sum_{i,\alpha_i>0} \alpha_i y_i \mathbf{x}_i \\ &= 0.0437\begin{pmatrix}3.5\\4.25\end{pmatrix} + 0.2162\begin{pmatrix}4\\3\end{pmatrix} + 0.1427\begin{pmatrix}4.5\\1.75\end{pmatrix} - 0.3589\begin{pmatrix}2\\2\end{pmatrix} - 0.0437\begin{pmatrix}2.5\\0.75\end{pmatrix} \\ &= \begin{pmatrix}0.833\\0.334\end{pmatrix}\end{aligned}$$

We can compute the final bias as the average of the bias obtained from each support vector using (23.28)

| $\mathbf{x}_i$ | $\mathbf{w}^T\mathbf{x}_i$ | $b_i = y_i - \mathbf{w}^T\mathbf{x}_i$ |
|---|---|---|
| $\mathbf{x}_1$ | 4.332 | $-3.332$ |
| $\mathbf{x}_2$ | 4.331 | $-3.331$ |
| $\mathbf{x}_4$ | 4.331 | $-3.331$ |
| $\mathbf{x}_{13}$ | 2.333 | $-3.333$ |
| $\mathbf{x}_{14}$ | 2.332 | $-3.332$ |
| $b = \text{avg}\{b_i\}$ | | $-3.332$ |

Thus the optimal hyperplane is given as follows

$$h(\mathbf{x}) = \begin{pmatrix}0.833\\0.334\end{pmatrix}^T \mathbf{x} - 3.332 = 0 \tag{23.31}$$

This is essentially the same as the canonical hyperplane we found in (23.17) in Example 23.2. The bias is slightly different due to numerical issues.

## 23.3 Soft Margin SVM: Linear and Non-Separable Case

So far we have assumed that the dataset is perfectly linearly separable. Here we consider the case where the classes overlap to some extent so that a perfect separation is not possible, as depicted in Figure 23.3.

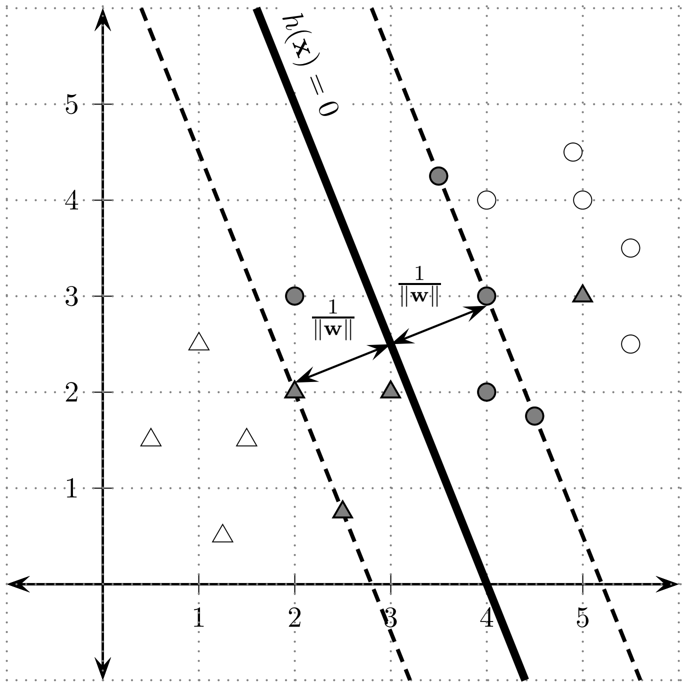

Figure 23.3: Soft Margin Hyperplane: The shaded points are the support vectors.

SVMs can handle such a set of points with overlapping classes by introducing *slack variables* $\xi_i$ in (23.16), as follows

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \tag{23.32}$$

Here $\xi_i \geq 0$ is the slack variable for point $\mathbf{x}_i$. First note that if $\xi_i = 0$, then the point is treated the same way as before. In other words that point is at least $\frac{1}{\|\mathbf{w}\|}$ away from the hyperplane. If $0 < \xi_i < 1$, then the point is still correctly classified, since it will remain on the correct side of the hyperplane. However, if $\xi_i \geq 1$ then the point is misclassified, since in this case it appears on the wrong side of the hyperplane.

In the non-separable case, also called the *soft margin* case, the goal of SVM classification is to find the hyperplane with the maximum margin, that also minimizes the slack terms. The new objective function is given as

$$\begin{aligned}
\textbf{Objective Function:} \quad & \min_{\mathbf{w},b,\xi_i} \left\{ \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{n}(\xi_i)^k \right\} \\
\textbf{Linear Constraints:} \quad & y_i\,(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i,\ \ \forall \mathbf{x}_i \in \mathbf{D} \\
& \xi_i \geq 0 \ \ \forall \mathbf{x}_i \in \mathbf{D}
\end{aligned} \tag{23.33}$$

where $C$ and $k$ are constants that incorporate the cost of misclassification. The term $\sum_{i=1}^{n}(\xi_i)^k$ gives the *loss*, i.e., an estimate of the deviation from the separable case. $C$, which is chosen empirically, is a *regularization constant* that controls the trade-off between maximizing the margin (corresponding to minimizing $\|\mathbf{w}\|^2/2$) or minimizing the loss (corresponding to minimizing the slack terms $\sum_{i=1}^{n}(\xi_i)^k$). For example, if $C \to 0$, then the loss component essentially disappears, and the objective defaults to maximizing the margin. On the other hand, if $C \to \infty$, then the margin ceases to have much effect, and the objective function tries to minimize the loss. The constant $k$ governs the form of the loss. Typically $k$ is set to 1 or 2. When $k = 1$, called *hinge loss*, the goal is to minimize the sum of the slack variables, whereas when $k = 2$, called *quadratic loss*, the goal is to minimize the sum of the squared slack variables.

### 23.3.1 Hinge Loss

Assuming that $k = 1$, we can compute the Lagrangian for the optimization problem (23.33) by introducing Lagrange multipliers $\alpha_i$ and $\beta_i$ as follows

$$\alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i\right) = 0 \text{ with } \alpha_i \geq 0 \tag{23.34}$$

$$\beta_i(\xi_i - 0) = 0 \text{ with } \beta_i \geq 0 \tag{23.35}$$

The Lagrangian is then given as

$$L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{n}\beta_i\xi_i \tag{23.36}$$

We turn this into a dual Lagrangian by taking its partial derivative with respect to $\mathbf{w}$, $b$ and $\xi_i$, and setting those to zero, as follows

$$\frac{\partial}{\partial\mathbf{w}}L = \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i = \mathbf{0} \quad \text{or} \quad \mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i\mathbf{x}_i \tag{23.37}$$

$$\frac{\partial}{\partial b}L = \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{23.38}$$

$$\frac{\partial}{\partial \xi_i}L = C - \alpha_i - \beta_i = 0 \quad \text{or} \quad \beta_i = C - \alpha_i \tag{23.39}$$

Plugging these values into (23.36), we get

$$\begin{aligned}L_{dual} &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\underbrace{\left(\sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i\right)}_{\mathbf{w}} - b\underbrace{\sum_{i=1}^{n}\alpha_i y_i}_{0} + \sum_{i=1}^{n}\alpha_i + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}(\alpha_i+\beta_i)\xi_i \\ &= -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n}\alpha_i + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}(\alpha_i + C - \alpha_i)\xi_i \\ &= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j\end{aligned} \tag{23.40}$$

The dual objective is thus given as

$$\begin{aligned}&\textbf{Objective Function: } \max_{\boldsymbol{\alpha}} \; L_{dual} = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \\ &\textbf{Linear Constraints: } 0 \le \alpha_i \le C, \;\; \forall i \in \mathbf{D} \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0\end{aligned} \tag{23.41}$$

Notice that (23.41) is exactly the same as the dual Lagrangian in the linearly separable case (23.25). The only difference is the constraint on each $\alpha_i$, since we now require that $\alpha_i + \beta_i = C$, which implies that $0 \le \alpha_i \le C$. Section 23.5 describes a gradient ascent approach for solving the dual objective function.

**Weight Vector and Bias:** Once we solve for $\alpha_i$, we have the same situation as before, namely, $\alpha_i = 0$ for points that are not support vectors, and $\alpha_i \ge 0$ only for the support vectors, which comprise all points $\mathbf{x}_i$ for which we have

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1 - \xi_i \tag{23.42}$$

Notice that the support vectors now include all points that are on the margin, which have zero slack ($\xi_i = 0$), as well as all points with positive slack ($\xi_i > 0$).

We can obtain the weight vector as before

$$\mathbf{w} = \sum_{i,\alpha_i>0}\alpha_i y_i \mathbf{x}_i \tag{23.43}$$

We can also solve for the $\beta_i$ using (23.39)

$$\beta_i = C - \alpha_i \tag{23.44}$$

Replacing $\beta_i$ in the KKT condition (23.35), with the expression from above, we obtain

$$(C - \alpha_i)\xi_i = 0 \tag{23.45}$$

Thus for the support vectors with $\alpha_i > 0$, we have two cases to consider

a) $C - \alpha_i = 0$, or $\alpha_i = C$

b) $C - \alpha_i > 0$, or $\alpha_i < C$. In this case, due to (23.45), we must have $\xi_i = 0$. In other words, these are precisely those support vectors that are on the margin. Using these, we can solve for $b_i$ as follows

$$\alpha_i \big(y_i(\mathbf{w}^T \mathbf{x}_i + b_i) - 1\big) = 0$$
$$y_i(\mathbf{w}^T \mathbf{x}_i + b_i) = 1$$
$$b_i = y_i - \mathbf{w}^T \mathbf{x}_i \tag{23.46}$$

To obtain the final bias $b$, we can take the average over all the $b_i$ values from above.

The final SVM model predicts the class for a new point $\mathbf{z}$ as follows

$$\hat{y} = sign(\mathbf{w}^T \mathbf{z} + b) \tag{23.47}$$

**Example 23.4:** Let us consider the data points shown in Figure 23.3. There are four new points in addition to the 14 we considered in Example 23.3, namely

| $\mathbf{x}_i$ | $x_{i1}$ | $x_{i2}$ | $y_i$ |
|---|---|---|---|
| $\mathbf{x}_{15}$ | 4 | 2 | +1 |
| $\mathbf{x}_{16}$ | 2 | 3 | +1 |
| $\mathbf{x}_{17}$ | 3 | 2 | −1 |
| $\mathbf{x}_{18}$ | 5 | 3 | −1 |

Let $k = 1$ and $C = 1$, then solving the $L_{dual}$ yields the following support vectors and the Lagrangian values $\alpha_i$

| $\mathbf{x}_i$ | $x_{i1}$ | $x_{i2}$ | $y_i$ | $\alpha_i$ |
|---|---|---|---|---|
| $\mathbf{x}_1$ | 3.5 | 4.25 | +1 | 0.0271 |
| $\mathbf{x}_2$ | 4 | 3 | +1 | 0.2162 |
| $\mathbf{x}_4$ | 4.5 | 1.75 | +1 | 0.9928 |
| $\mathbf{x}_{13}$ | 2 | 2 | −1 | 0.9928 |
| $\mathbf{x}_{14}$ | 2.5 | 0.75 | −1 | 0.2434 |
| $\mathbf{x}_{15}$ | 4 | 2 | +1 | 1 |
| $\mathbf{x}_{16}$ | 2 | 3 | +1 | 1 |
| $\mathbf{x}_{17}$ | 3 | 2 | −1 | 1 |
| $\mathbf{x}_{18}$ | 5 | 3 | −1 | 1 |

All other points are not support vectors, so they have $\alpha_i = 0$. Using (23.43) we can compute the weight vector for the hyperplane

$$\begin{aligned}\mathbf{w} &= \sum_{i,\alpha_i>0} \alpha_i y_i \mathbf{x}_i \\ &= 0.0271 \begin{pmatrix}3.5\\4.25\end{pmatrix} + 0.2162 \begin{pmatrix}4\\3\end{pmatrix} + 0.9928 \begin{pmatrix}4.5\\1.75\end{pmatrix} - 0.9928 \begin{pmatrix}2\\2\end{pmatrix} \\ &\quad - 0.2434 \begin{pmatrix}2.5\\0.75\end{pmatrix} + \begin{pmatrix}4\\2\end{pmatrix} + \begin{pmatrix}2\\3\end{pmatrix} - \begin{pmatrix}3\\2\end{pmatrix} - \begin{pmatrix}5\\3\end{pmatrix} \\ &= \begin{pmatrix}0.834\\0.333\end{pmatrix}\end{aligned}$$

We can compute the final bias as the average of the biases obtained from each support vector using (23.46). Note that we compute the per-point bias only for the support vectors that lie precisely on the margin. These support vectors have $\xi_i = 0$ and have $0 < \alpha_i < C$. Put another way, we do not compute the bias for support vectors with $\alpha_i = C = 1$, which include the points $\mathbf{x}_{15}$, $\mathbf{x}_{16}$, $\mathbf{x}_{17}$, and $\mathbf{x}_{18}$. From the remaining support vectors, we get

| $\mathbf{x}_i$ | $\mathbf{w}^T\mathbf{x}_i$ | $b_i = y_i - \mathbf{w}^T\mathbf{x}_i$ |
|---|---|---|
| $\mathbf{x}_1$ | 4.334 | −3.334 |
| $\mathbf{x}_2$ | 4.334 | −3.334 |
| $\mathbf{x}_4$ | 4.334 | −3.334 |
| $\mathbf{x}_{13}$ | 2.334 | −3.334 |
| $\mathbf{x}_{14}$ | 2.334 | −3.334 |
| $b = \text{avg}\{b_i\}$ | | −3.334 |

Thus the optimal hyperplane is given as follows

$$h(\mathbf{x}) = \begin{pmatrix}0.834\\0.333\end{pmatrix}^T \mathbf{x} - 3.334 = 0 \tag{23.48}$$

One can see that this is essentially the same as the canonical hyperplane we found in (23.3).

It is instructive to see what the slack variables are in this case. Note that $\xi_i = 0$ for all points that are not support vectors, and also for those support vectors that are on the margin. So the slack is positive only for the remaining support vectors, for whom the slack can be computed directly from (23.42), as follows

$$\xi_i = 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)$$

Thus, for all support vectors not on the margin, we obtain

| $\mathbf{x}_i$ | $\mathbf{w}^T\mathbf{x}_i$ | $\mathbf{w}^T\mathbf{x}_i + b$ | $\xi_i = 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)$ |
|---|---|---|---|
| $\mathbf{x}_{15}$ | 4.001 | 0.667 | 0.333 |
| $\mathbf{x}_{16}$ | 2.667 | −0.667 | 1.667 |
| $\mathbf{x}_{17}$ | 3.167 | −0.167 | 0.833 |
| $\mathbf{x}_{18}$ | 5.168 | 1.834 | 2.834 |

As expected the slack variable $\xi_i > 1$ for those points that are misclassified (i.e., are on the wrong side of the hyperplane), namely $\mathbf{x}_{16} = (3, 3)$ and $\mathbf{x}_{18} = (5, 3)$. The other two points are correctly classified, but lie within the margin, and thus satisfy $0 < \xi_i < 1$. The total slack is then given as

$$\sum_i \xi_i = \xi_{15} + \xi_{16} + \xi_{17} + \xi_{18} = 0.333 + 1.667 + 0.833 + 2.834 = 5.667$$

### 23.3.2 Quadratic Loss

For quadratic loss, we have $k = 2$ in the objective function (23.33). Notice that for quadratic loss, we can drop the positivity constraint $\xi_i \geq 0$. This is because, if $\xi_i < 0$ is replaced by $\xi_i = 0$, then the constraint $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$ is still satisfied, and at the same time, $\xi_i = 0$ leads to a smaller value of the primary objective $C\sum_{i=1}^n \xi_i^2$. Thus the optimal solution of (23.33) coincides with the revised objective

$$\begin{aligned} &\textbf{Objective Function: } \min_{\mathbf{w},b,\xi_i} \left\{ \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^n \xi_i^2 \right\} \\ &\textbf{Linear Constraints: } y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, \ \ \forall \mathbf{x}_i \in \mathbf{D} \end{aligned} \tag{23.49}$$

The Lagrangian is given as

$$L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^n \xi_i^2 - \sum_{i=1}^n \alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i\right) \tag{23.50}$$

Differentiating with respect to $\mathbf{w}$, $b$, and $\xi_i$ gives the following conditions, respectively

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

$$\xi_i = \frac{1}{2C}\alpha_i$$

Substituting these back into (23.50) yields the dual objective

$$\begin{aligned} L_{dual} &= \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j - \frac{1}{4C}\sum_{i=1}^n \alpha_i^2 \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j \left(\mathbf{x}_i^T\mathbf{x}_j + \frac{1}{2C}\delta_{ij}\right) \end{aligned}$$

where $\delta$ is the *kronecker delta* function, defined as $\delta_{ij} = 1$ if $i = j$, and $\delta_{ij} = 0$ otherwise. Thus the dual objective is given as

$$\max_{\boldsymbol{\alpha}} \;\; L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \left(\mathbf{x}_i^T\mathbf{x}_j + \frac{1}{2C}\delta_{ij}\right)$$
$$\text{subject to the constraints } \alpha_i \geq 0, \forall i \in \mathbf{D}, \text{and } \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{23.51}$$

Once we solve for $\alpha_i$ using the methods from Section 23.5, we can recover the weight vector and bias as follows

$$\mathbf{w} = \sum_{i,\alpha_i>0} \alpha_i y_i \mathbf{x}_i$$
$$b = \text{avg}_{i,\alpha_i>0}\left\{y_i - \mathbf{w}^T\mathbf{x}_i\right\}$$

## 23.4 Kernel SVM: Nonlinear Case

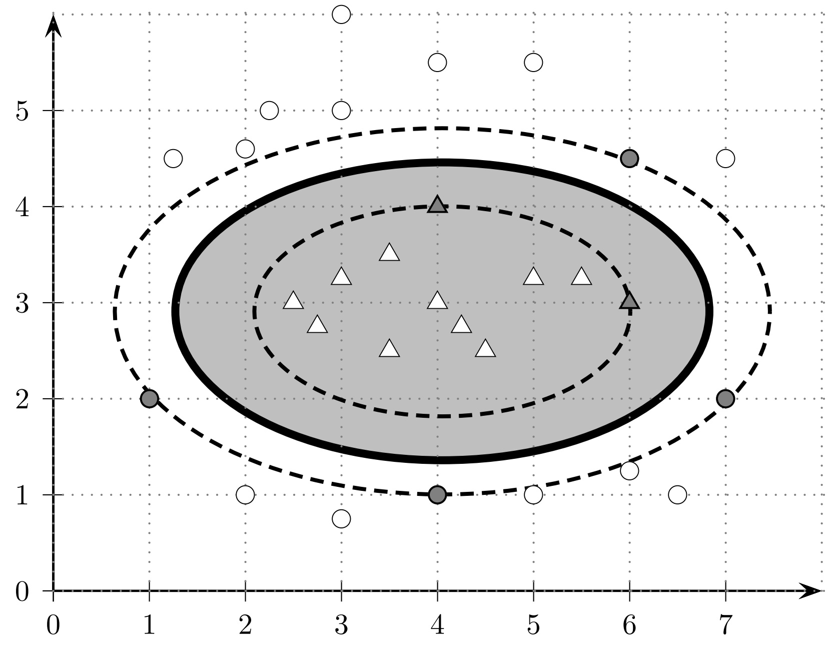

Figure 23.4: Nonlinear SVM: Shaded points are the support vectors.

In this section we will describe how to apply the linear SVM approach to solve problems with a non-linear decision boundary. This can be achieved via the kernel trick from Chapter 5. Conceptually, the idea is to map the original $d$-dimensional points $\mathbf{x}_i$ in the input space, to points $\phi(\mathbf{x}_i)$ in a high-dimensional feature space via some non-linear transformation $\phi$. Given the extra flexibility, it is more likely that the points $\phi(\mathbf{x}_i)$ might be linearly separable in the feature space. Note however that the linear decision surface in the feature space actually corresponds to a non-linear decision surface in the input space.

**Example 23.5:** Consider the set of points shown in Figure 23.4. There is no linear classifier that can discriminate between the points. However, there exists a perfect quadratic classifier that can separate the two classes. That is, given the input space over the two dimensions $X_1$ and $X_2$, if we transform each point $\mathbf{x} = (x_1, x_2)^T$ into a point in the feature space consisting of the dimensions $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$, via the transformation $\phi(\mathbf{x}) = (\sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)^T$, then it is possible to find a linearly separable hyperplane in the feature space. For this dataset, it is possible to map the hyperplane back to the input space, where it is seen as an ellipse (thick black line) that separates the two classes (shown as circles and triangles). The support vectors are those points (shown in gray) that lie on the margin (dashed ellipses).

To apply the kernel trick for non-linear SVM classification, we have to show that all operations require only the kernel function

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \tag{23.52}$$

Let the original database be given as $\mathbf{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$. Applying $\phi$ to each point, we can obtain the new dataset in the feature space $\mathbf{D}_\phi = \{\phi(\mathbf{x}_i), y_i\}_{i=1}^n$.

The objective function (23.33) in the feature space is given as

$$\begin{aligned} &\textbf{Objective Function:} \ \min_{\mathbf{w}, b, \xi_i} \left\{ \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^n (\xi_i)^k \right\} \\ &\textbf{Linear Constraints:} \ y_i \left(\mathbf{w}^T \phi(\mathbf{x}_i) + b\right) \geq 1, \ \ \forall \mathbf{x}_i \in \mathbf{D} \end{aligned} \tag{23.53}$$

where $\mathbf{w}$ is the weight vector, and $\xi_i$ are the slack variables, all in feature space.

**Hinge Loss:** For hinge loss, we can set up the dual Lagrangian (23.41) as follows

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \ L_{dual} &= \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{23.54}$$

Subject to the constraints that $0 \leq \alpha_i \leq C$, and $\sum_{i=1}^n \alpha_i y_i = 0$. Notice that the dual Lagrangian depends only on the dot product between two vectors in the feature space

$\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$, and thus we can solve the optimization problem using the kernel matrix $\mathbf{K} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1,\ldots,n}$. Section 23.5 describes a stochastic gradient descent approach for solving the dual objective function.

**Quadratic Loss:** For quadratic loss, the dual Lagrangian (23.51) corresponds to a change of kernel. Define a new kernel function $K'$, as follows

$$K'(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j + \frac{1}{2C}\delta_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{2C}\delta_{ij} \tag{23.55}$$

which affects only the diagonal entries of the kernel matrix $\mathbf{K}$. Thus the dual Lagrangian is given as

$$\max_{\boldsymbol{\alpha}} L_{dual} = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j K'(\mathbf{x}_i, \mathbf{x}_j) \tag{23.56}$$

subject to the constraints that $\alpha_i \geq 0$, and $\sum_{i=1}^n \alpha_i y_i = 0$. The above optimization can be solved using the same approach as for hinge loss, with a simple change of kernels.

**Weight Vector and Bias:** We can solve for $\mathbf{w}$ in the feature space as follows

$$\mathbf{w} = \sum_{\alpha_i > 0} \alpha_i y_i \phi(\mathbf{x}_i) \tag{23.57}$$

Since $\mathbf{w}$ uses $\phi(\mathbf{x}_i)$ directly, in general, we may not be able or willing to compute $\mathbf{w}$ explicitly. However, as we shall see next, it is not necessary to explicitly compute $\mathbf{w}$ for classifying the points.

Let us first see how to compute the bias via only kernel operations. We compute $b$ as the average over the support vectors

$$b = \frac{1}{n_{sv}}\left(\sum_{\alpha_i > 0} y_i - \sum_{\alpha_i > 0} \mathbf{w}^T\phi(\mathbf{x}_i)\right) \tag{23.58}$$

where $n_{sv}$ is the number of support vectors with $\alpha_i > 0$. Substituting $\mathbf{w}$ from above, we obtain the new expression for $b$ as

$$\begin{aligned} b &= \frac{1}{n_{sv}}\left(\sum_{\alpha_i > 0} y_i - \sum_{\alpha_i > 0}\sum_{\alpha_j > 0} \alpha_i y_i \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)\right) \\ &= \frac{1}{n_{sv}}\left(\sum_{\alpha_i > 0} y_i - \sum_{\alpha_i > 0}\sum_{\alpha_j > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j)\right) \end{aligned} \tag{23.59}$$

Notice that $b$ is also a function of the dot product between two vectors in the feature space $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$.

**Classifier:** Finally, we can predict the class for a new point $\mathbf{z}$ as follows

$$\begin{aligned}\hat{y} &= sign(\mathbf{w}^T\phi(\mathbf{z}) + b)\\ &= sign\left(\sum_{\alpha_i>0} \alpha_i y_i \phi(\mathbf{x}_i)^T\phi(\mathbf{z}) + b\right)\\ &= sign\left(\sum_{\alpha_i>0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{z}) + b\right) \qquad (23.60)\end{aligned}$$

Once again we see that $\hat{y}$ uses only the dot product in feature space.

Based on the above derivation, it is clear that to train and test the SVM classifier, $\phi(\mathbf{x}_i)$ is never needed in isolation. Instead, all operations can be carried out purely in terms of the kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. Thus any non-linear kernel function can be used to do non-linear classification in the input space. Examples of such non-linear kernels include the polynomial kernel (**??**), and the Gaussian kernel (5.12), among others.

**Example 23.6:** Let us consider the example dataset shown in Figure 23.4. The dataset has 29 points in total. With $C = 4$, and a polynomial kernel (**??**) of degree $q = 2$, solving the $L_{dual}$ quadratic program yields six support vectors, shown as the shaded (gray) points in Figure 23.4.

| $\mathbf{x}_i$ | $(x_{i1}, x_{i2})$ | $\phi(\mathbf{x}_i)$ | $y_i$ | $\alpha_i$ |
|---|---|---|---|---|
| $\mathbf{x}_1$ | $(1, 2)$ | $(1, 1.41, 2.83, 1, 4, 2.83)$ | $+1$ | 0.6198 |
| $\mathbf{x}_2$ | $(4, 1)$ | $(1, 5.66, 1.41, 16, 1, 5.66)$ | $+1$ | 2.069 |
| $\mathbf{x}_3$ | $(6, 4.5)$ | $(1, 8.49, 6.36, 36, 20.25, 38.18)$ | $+1$ | 3.803 |
| $\mathbf{x}_4$ | $(7, 2)$ | $(1, 9.90, 2.83, 49, 4, 19.80)$ | $+1$ | 0.3182 |
| $\mathbf{x}_5$ | $(4, 4)$ | $(1, 5.66, 5.66, 16, 16, 15.91)$ | $-1$ | 2.9598 |
| $\mathbf{x}_6$ | $(6, 3)$ | $(1, 8.49, 4.24, 36, 9, 25.46)$ | $-1$ | 3.8502 |

The quadratic polynomial kernel is given as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) = (1 + \mathbf{x}_i^T\mathbf{x}_j)^2$$

where the transformation $\phi$ is given as

$$\phi(\mathbf{x} = (x_1, x_2)) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

The table above shows all the transformed points. For example,

$$\mathbf{x}_1 = (1, 2)^T$$

is transformed into

$$\phi(\mathbf{x}_i) = (1, \sqrt{2}\cdot 1, \sqrt{2}\cdot 2, 1^2, 2^2, \sqrt{2}\cdot 1\cdot 2) = (1, 1.41, 2.83, 1, 2, 2.83)^T$$

We can compute the weight vector for the hyperplane using (23.57)

$$\mathbf{w} = \sum_{i,\alpha_i>0} \alpha_i y_i \phi(\mathbf{x}_i) = \begin{pmatrix} 0 \\ -1.413 \\ -3.298 \\ 0.256 \\ 0.82 \\ -0.018 \end{pmatrix}$$

The bias can be computed by using (23.59)

$$b = -8.841$$

Given the hyperplane in the transformed space $\phi(x_{i1}, x_{i2}) = (1, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}, x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2})$, how do we map the discriminant back to the original space? Noting that the discriminant in the original space corresponds to an ellipse, we can compute the coordinates of the center as well as the values for the semimajor and semiminor axes of the ellipse. For our example, the center is given as $(4.046, 2.907)$, and the semimajor axis is 2.78 and the semiminor axis is 1.55. These values were used to plot the discriminant in Figure 23.4.

In this example we explicitly transformed all the points into the feature space just for illustration purposes. The kernel trick allows us to achieve the same goal using only the kernel function. Further, in this example, we explicitly computed the weight vector $\mathbf{w}$ to illustrate the steps. However, depending on the kernel function, in general, it may not be possible to explicitly compute $\mathbf{w}$.

## 23.5 SVM Training Algorithms

We now turn our attention to algorithms for solving the SVM optimization problems. We will consider simple optimization approaches for solving the dual as well as the primal formulations. It is important to note that these methods are not the most efficient to solve the SVM optimization problem. However, since they are relatively simple, they can serve as a starting point for more sophisticated methods.

For the SVM algorithms in this section, instead of dealing explicitly with the bias $b$, we map each point $\mathbf{x}_i \in \mathbb{R}^d$ to the point $\mathbf{x}'_i \in \mathbb{R}^{d+1}$ as follows

$$\mathbf{x}'_i = (x_{i1}, \cdots, x_{id}, 1)^T \tag{23.61}$$

Furthermore, we also map the weight vector to $\mathbb{R}^{d+1}$, with $w_{d+1} = b$, so that

$$\mathbf{w} = (w_1, \cdots, w_d, b)^T \tag{23.62}$$

The equation of the hyperplane (23.1) is then given as follows

$$h(\mathbf{x}') : \mathbf{w}^T\mathbf{x}' = 0 \tag{23.63}$$
$$\implies h(\mathbf{x}') : \begin{pmatrix} w_1 & \cdots & w_d & b \end{pmatrix} \begin{pmatrix} x_{i1} \\ \vdots \\ x_{id} \\ 1 \end{pmatrix} = 0$$
$$\implies h(\mathbf{x}') : w_1 x_{i1} + \cdots + w_d x_{id} + b = 0$$

In the discussion below we assume that the bias has been included into $\mathbf{w}$, and that each point has been mapped to $\mathbb{R}^{d+1}$ as given in (23.61) and (23.62). Thus, the last component of $\mathbf{w}$ will yields the bias $b$. Another consequence of mapping the points to $\mathbb{R}^{d+1}$ is that the constraint $\sum_{i=1}^n \alpha_i y_i = 0$ doesn't apply in the SVM dual formulations given in (23.41), (23.51), (23.54), and (23.56). This is because there is no explicit bias term $b$ in the linear constraints given in (23.42). The new set of constraints is given as

$$y_i \mathbf{w}^T\mathbf{x} \geq 1 - \xi_i \tag{23.64}$$

### 23.5.1 Dual Solution: Stochastic Gradient Ascent

We consider only the hinge loss case, since quadratic loss can be handled by a change of kernel, as shown in (23.56). The dual optimization objective for hinge loss (23.54) is given as

$$\max J(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{23.65}$$

subject to the constraints $0 \leq \alpha_i \leq C$ for all $i = 1, \cdots, n$.

Let us consider the terms in $J(\boldsymbol{\alpha})$ that involve the Lagrange multiplier $\alpha_k$

$$J(\alpha_k) = \alpha_k - \frac{1}{2}\alpha_k^2 y_k^2 K(\mathbf{x}_k, \mathbf{x}_k) - \alpha_k y_k \sum_{\substack{i=1 \\ i\neq k}}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k) \tag{23.66}$$

The gradient at $\boldsymbol{\alpha}$ is given as

$$\nabla J(\boldsymbol{\alpha}) = \left(\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_1}, \frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_2}, \cdots, \frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_n}\right)^T \tag{23.67}$$

where the $k$-th component of the gradient is given as follows

$$\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_k} = \frac{\partial J(\alpha_k)}{\partial \alpha_k} = 1 - y_k\left(\sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\right) \tag{23.68}$$

Since we want to maximize the objective function $J(\boldsymbol{\alpha})$, we should move in the direction of the gradient $\nabla J(\boldsymbol{\alpha})$. Starting from an initial $\boldsymbol{\alpha}$, the gradient ascent approach successively updates it as follows

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \eta_t \nabla J(\boldsymbol{\alpha}_t) \tag{23.69}$$

Instead of updating the entire $\boldsymbol{\alpha}$ vector in each step, in the stochastic gradient ascent approach, we can instead update each component independently, and immediately use the new values to update other components. This can result in faster convergence. The update rule for the $k$-th component is given as

$$\begin{aligned}\alpha_k' &= \alpha_k + \eta_k \frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_k}\\ &= \alpha_k + \eta_k \left(1 - y_k \sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\right)\end{aligned} \tag{23.70}$$

We also have to ensure that the constraints $\alpha_k \in [0, C]$. Thus in the update step above, if $\alpha_k < 0$ we reset it so that $\alpha_k = 0$, and if $\alpha_k > C$ we reset it so that $\alpha_k = C$. The stochastic gradient ascent algorithm is given in Algorithm 23.1.

In Algorithm 23.1 we have to determine the step size $\eta_k$ for $\alpha_k$. Ideally, we would like to choose a step size so that the $k$-th component of the gradient at $\alpha_k'$ goes to zero. This happens when

$$\eta_k = \frac{1}{K(\mathbf{x}_k, \mathbf{x}_k)} \tag{23.71}$$

To see why, note that when only $\alpha_k'$ is updated, the other $\alpha_i$ do not change. Thus the new $\boldsymbol{\alpha}'$ has a change only in $\alpha_k'$, and we get

$$\frac{\partial J(\boldsymbol{\alpha}')}{\partial \alpha_k} = \left(1 - y_k \sum_{i \neq k} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\right) - y_k \alpha_k' y_k K(\mathbf{x}_k, \mathbf{x}_k)$$

plugging in the value of $\alpha_k'$ from (23.70), we have

$$\begin{aligned}\frac{\partial J(\boldsymbol{\alpha}')}{\partial \alpha_k} &= \left(1 - y_k \sum_{i \neq k} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\right) - \left(\alpha_k + \eta_k \Big(1 - y_k \sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\Big)\right) K(\mathbf{x}_k, \mathbf{x}_k)\\ &= \left(1 - y_k \sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\right) - \eta_k K(\mathbf{x}_k, \mathbf{x}_k) \Big(1 - y_k \sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\Big)\\ &= \Big(1 - \eta_k K(\mathbf{x}_k, \mathbf{x}_k)\Big) \left(1 - y_k \sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\right)\end{aligned}$$

**Algorithm 23.1**: Dual SVM Algorithm: Stochastic Gradient Ascent

```
    SVM-DUAL ($\mathbf{D}, K, C, \epsilon$):
1   foreach $\mathbf{x}_i \in \mathbf{D}$ do  $\mathbf{x}_i \leftarrow \begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix}$ // map to $\mathbb{R}^{d+1}$
2   if $loss$ = hinge then
3       $\mathbf{K} \leftarrow \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1,\dots,n}$ // kernel matrix, hinge loss
4   else if $loss$ = quadratic then
5       $\mathbf{K} \leftarrow \{K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{2C}\delta_{ij}\}_{i,j=1,\dots,n}$ // kernel matrix, quadratic loss
6   for $k = 1, \cdots, n$ do $\eta_k \leftarrow \frac{1}{K(\mathbf{x}_k, \mathbf{x}_k)}$ // set step size
7   $t \leftarrow 0$
8   $\boldsymbol{\alpha}_0 \leftarrow (0, \dots, 0)^T$
9   repeat
10      $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}_t$
11      for $k = 1$ to $n$ do
            // update $k$-th component of $\boldsymbol{\alpha}$
12          $\alpha_k \leftarrow \alpha_k + \eta_k \Big(1 - y_k \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\Big)$
13          if $\alpha_k < 0$ then $\alpha_k \leftarrow 0$
14          if $\alpha_k > C$ then $\alpha_k \leftarrow C$
15      $\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}$
16      $t \leftarrow t + 1$
17  until $\|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t-1}\| \leq \epsilon$
```

substituting $\eta_k$ from (23.71), we have

$$\begin{aligned}\frac{\partial J(\boldsymbol{\alpha}')}{\partial a_k} &= \left(1 - \frac{1}{K(\mathbf{x}_k, \mathbf{x}_k)} K(\mathbf{x}_k, \mathbf{x}_k)\right) \left(1 - y_k \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k)\right) \\ &= 0\end{aligned}$$

Thus in Algorithm 23.1, for better convergence, we choose $\eta_k$ according to (23.71). Since the above description assumes a general kernel function between any two points, we can recover the linear, non-separable case by simply setting $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.

**Example 23.7 (Dual SVM: Linear Kernel):** Figure 23.5 shows the $n = 150$ points from the Iris dataset, using `sepal length` and `sepal width` as the two attributes. The goal is to discriminate between `Iris-setosa` (shown as circles) and other types of iris flowers (shown as triangles). Algorithm 23.1 was used to train the SVM classifier with a linear kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ and $\epsilon = 0.0001$,

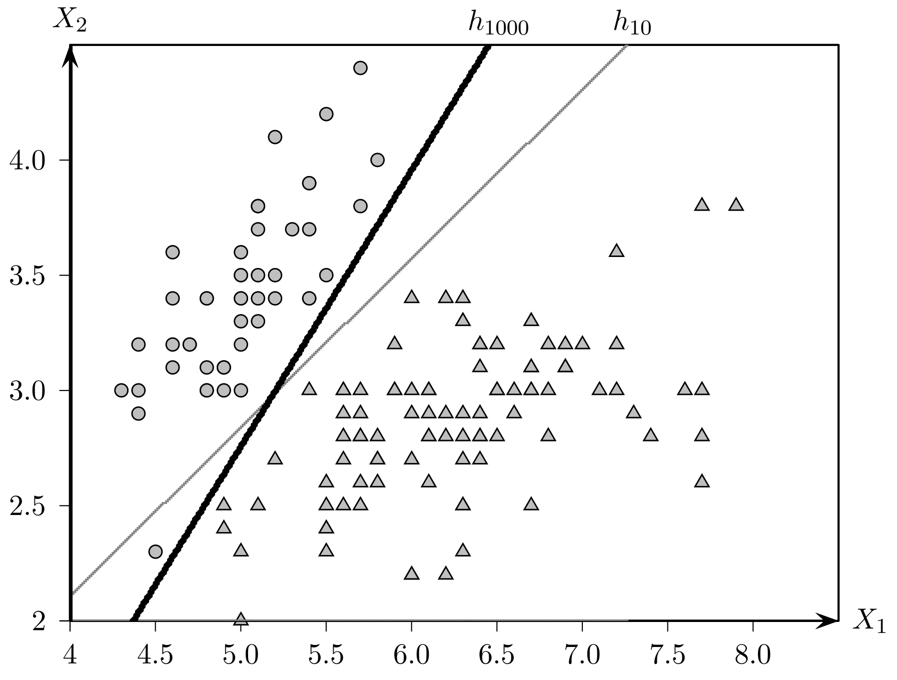

Figure 23.5: SVM Dual Algorithm with Linear Kernel

with hinge loss. Two different values of $C$ were used; hyperplane $h_{10}$ uses $C = 10$, whereas $h_{1000}$ uses $C = 1000$, specified as as follows

$$h_{10}(\mathbf{x}) : 2.74x_1 - 3.74x_2 - 3.09 = 0$$
$$h_{1000}(\mathbf{x}) : 8.56x_1 - 7.14x_2 - 23.12 = 0$$

$h_{10}$ has a larger margin, but also has a larger slack; it misclassifies one of the circles. $h_{1000}$ has a smaller margin, but it also minimizes the slack; it is a separating hyperplane. In other words, the higher the value of $C$ the more the emphasis on minimizing the slack.

**Example 23.8 (Dual SVM: Quadratic Kernel):** Figure 23.6 shows the $n = 150$ points from the Iris dataset projected on the first two principal components $(\mathbf{u}_1, \mathbf{u}_2)$. The task is to separate `Iris-versicolor` (in circles) from the other two types of irises (in triangles). The figure plots the decision boundaries obtained when using the linear kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$, and the homogeneous quadratic kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^2$, where $\mathbf{x}_i \in \mathbb{R}^{d+1}$, as per (23.61). The optimal hyperplane in both cases was found via the gradient ascent approach in Algorithm 23.1, with $C = 10$, $\epsilon = 0.0001$ and using hinge loss.

The optimal hyperplane $h_l$ (shown in gray) for the linear kernel is given as

$$h_l(\mathbf{x}) : 0.16x_1 + 1.9x_2 + 0.8 = 0$$

As expected, $h_l$ is unable to separate the classes. On the other hand, the optimal hyperplane $h_q$ (shown as clipped black ellipse) for the quadratic kernel is given as

$$\begin{aligned} h_q(\mathbf{x}) :& \mathbf{w}^T \phi(\mathbf{x}) = 0 \\ & \big(1.86, 1.32, 0.099, 0.85, -0.87, -3.25\big) \left(x_1^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, x_2^2, \sqrt{2}x_2, 1\right)^T = 0 \\ & 1.86x_1^2 + 1.87x_1x_2 + 0.14x_1 + 0.85x_2^2 - 1.22x_2 - 3.25 = 0 \end{aligned}$$

$h_q$ is able to separate the two classes quite well.

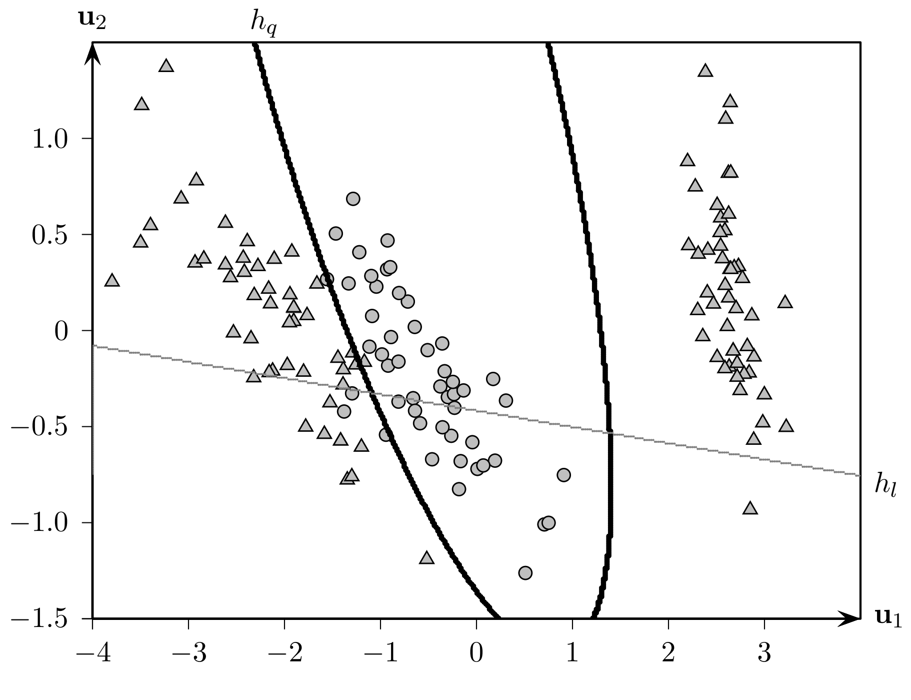

Figure 23.6: SVM Dual Algorithm with Quadratic Kernel

## 23.5.2 Primal Solution: Newton Optimization

The dual approach is the one most commonly used to train SVMs, but it is also possible to train using the primal formulation.

Consider the primal optimization function for the linear, but non-separable case (23.33). With $\mathbf{w}, \mathbf{x}_i \in \mathbb{R}^{d+1}$ as discussed above, we have to minimize the objective function

$$\min J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i)^k \tag{23.72}$$

subject to the linear constraints

$$y_i\ (\mathbf{w}^T\mathbf{x}_i) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \text{ for all } i = 1, \cdots, n \tag{23.73}$$

Rearranging the above, we obtain an expression for $\xi_i$

$$\begin{aligned} \xi_i &\geq 1 - y_i\ (\mathbf{w}^T\mathbf{x}_i) \text{ and } \xi_i \geq 0 \\ \implies \xi_i &= \max\big(0, 1 - y_i\ (\mathbf{w}^T\mathbf{x}_i)\big) \end{aligned} \tag{23.74}$$

Plugging (23.74) into the objective function (23.72), we obtain

$$\begin{aligned} J(\mathbf{w}) &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\max\big(0, 1 - y_i\ (\mathbf{w}^T\mathbf{x}_i)\big)^k \\ &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{y_i(\mathbf{w}^T\mathbf{x}_i)<1}\big(1 - y_i(\mathbf{w}^T\mathbf{x}_i)\big)^k \end{aligned} \tag{23.75}$$

The last step follows from the fact that $\xi_i > 0$ if and only if $1 - y_i(\mathbf{w}^T\mathbf{x}_i) > 0$, i.e., $y_i(\mathbf{w}^T\mathbf{x}_i) < 1$. Unfortunately, the hinge loss formulation, with $k = 1$, is not differentiable. One could use a differentiable approximation to the hinge loss, but here we describe the quadratic loss formulation.

**Quadratic Loss** For quadratic loss, we have $k = 2$, and the primal objective can be written as

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{y_i(\mathbf{w}^T\mathbf{x}_i)<1}\big(1 - y_i(\mathbf{w}^T\mathbf{x}_i)\big)^2$$

The gradient, or the rate of change in the objective function at $\mathbf{w}$, is given as the partial derivative of $J(\mathbf{w})$ with respect to $\mathbf{w}$

$$\begin{aligned} \nabla_{\mathbf{w}} = \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} &= \mathbf{w} - 2C\bigg(\sum_{y_i(\mathbf{w}^T\mathbf{x}_i)<1} y_i\mathbf{x}_i\ \big(1 - y_i(\mathbf{w}^T\mathbf{x}_i)\big)\bigg) \\ &= \mathbf{w} - 2C\bigg(\sum_{y_i(\mathbf{w}^T\mathbf{x}_i)<1} y_i\mathbf{x}_i\bigg) + 2C\bigg(\sum_{y_i(\mathbf{w}^T\mathbf{x}_i)<1}\mathbf{x}_i\mathbf{x}_i^T\bigg)\mathbf{w} \\ &= \mathbf{w} - 2C\mathbf{v} + 2C\mathbf{S}\mathbf{w} \end{aligned} \tag{23.76}$$

where the vector $\mathbf{v}$ and the matrix $\mathbf{S}$ are given as

$$\mathbf{v} = \sum_{y_i(\mathbf{w}^T\mathbf{x}_i)<1} y_i \mathbf{x}_i$$
$$\mathbf{S} = \sum_{y_i(\mathbf{w}^T\mathbf{x}_i)<1} \mathbf{x}_i \mathbf{x}_i^T$$

Furthermore, the *Hessian matrix*, defined as the second-order partial derivatives of $J(\mathbf{w})$ with respect to $\mathbf{w}$, is given as

$$\mathbf{H}_{\mathbf{w}} = \frac{\partial \nabla_{\mathbf{w}}}{\partial \mathbf{w}} = \mathbf{I} + 2C\mathbf{S}$$

Since we want to minimize the objective function $J(\mathbf{w})$, we should move in the direction opposite to the gradient. The Newton optimization update rule for $\mathbf{w}$ is given as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \mathbf{H}_{\mathbf{w}_t}^{-1} \nabla_{\mathbf{w}_t} \tag{23.77}$$

where $\eta_t > 0$ is a scalar value denoting the step size at iteration $t$. Normally one needs to use a line search method to find the optimal step size $\eta_t$, but the default value of $\eta_t = 1$ usually works for the quadratic loss.

**Algorithm 23.2**: Primal SVM Algorithm: Newton Optimization, Quadratic Loss

```
   SVM-PRIMAL ($\mathbf{D}, C, \epsilon$):
1  foreach $\mathbf{x}_i \in \mathbf{D}$ do
2      $\mathbf{x}_i \leftarrow \begin{pmatrix}\mathbf{x}_i \\ 1\end{pmatrix}$ // map to $\mathbb{R}^{d+1}$
3  $t \leftarrow 0$
4  $\mathbf{w}_0 \leftarrow (0, \dots, 0)^T$ // initialize $\mathbf{w}_t \in \mathbb{R}^{d+1}$
5  repeat
6      $\mathbf{v} \leftarrow \sum_{y_i(\mathbf{w}_t^T\mathbf{x}_i)<1} y_i \mathbf{x}_i$
7      $\mathbf{S} \leftarrow \sum_{y_i(\mathbf{w}_t^T\mathbf{x}_i)<1} \mathbf{x}_i \mathbf{x}_i^T$
8      $\nabla \leftarrow (\mathbf{I} + 2C\mathbf{S})\mathbf{w}_t - 2C\mathbf{v}$ // gradient
9      $\mathbf{H} \leftarrow \mathbf{I} + 2C\mathbf{S}$ // Hessian
10     $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \mathbf{H}^{-1} \nabla$ // Newton update rule (23.77)
11     $t \leftarrow t + 1$
12 until $\|\mathbf{w}_t - \mathbf{w}_{t-1}\| \le \epsilon$
```

The Newton optimization algorithm for training a linear, non-separable SVMs in the primal is given in Algorithm 23.2. The step size $\eta_t$ is set to 1 by default.

After computing the gradient and Hessian at $\mathbf{w}_t$ (lines 6–9), the Newton update rule obtains the new weight vector $\mathbf{w}_{t+1}$ (line 10). The iterations continue until there is very little change in the weight vector.

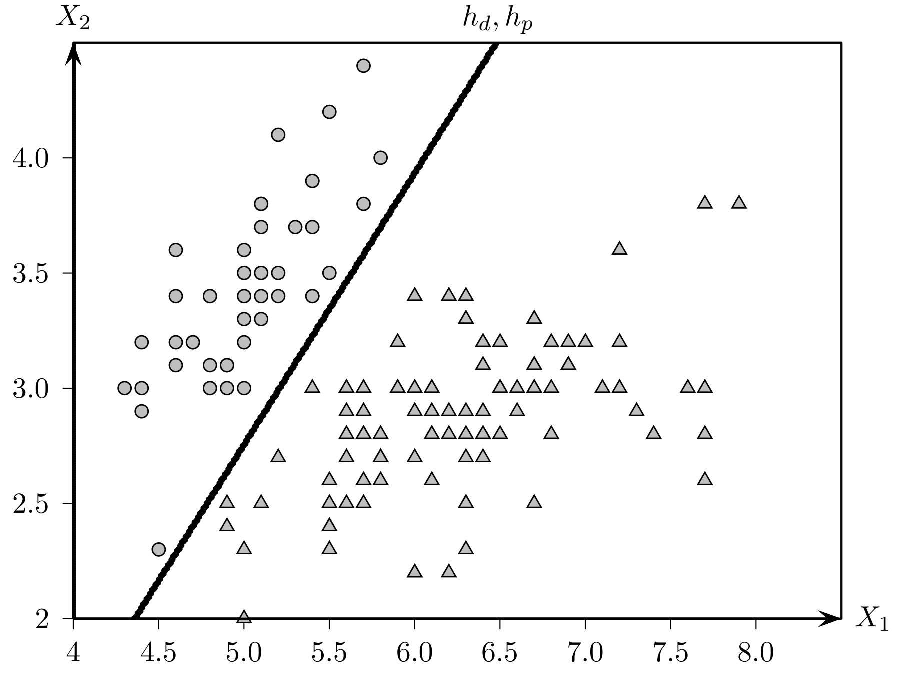

Figure 23.7: SVM Primal Algorithm with Linear Kernel

**Example 23.9 (Primal SVM):** Figure 23.7 plots the hyperplanes obtained using the dual and primal approaches for the Iris dataset (`sepal length` versus `sepal width`). We used $C = 1000$ and $\epsilon = 0.0001$, with the quadratic loss function. The dual solution $h_d$ (gray line) and the primal solution $h_p$ (thick black line) are as follows

$$h_d(\mathbf{x}) : 7.47x_1 - 6.34x_2 - 19.89 = 0$$
$$h_p(\mathbf{x}) : 7.47x_1 - 6.34x_2 - 19.91 = 0$$

The dual and primal solutions are essentially identical.

### Primal Kernel SVMs

In the discussion above we considered the linear, non-separable case for SVM learning. We now generalize the primal approach to learn kernel-based SVMs, again for quadratic loss.

Let $\phi$ denote a mapping from the input space to the feature space; each input point $\mathbf{x}_i$ is mapped to the feature point $\phi(\mathbf{x}_i)$. Let $K(\mathbf{x}_i, \mathbf{x}_j)$ denote the kernel function, and let $\boldsymbol{\omega}$ denote the weight vector in feature space. The hyperplane in feature space is then given as

$$h(\phi(\mathbf{x})) : \boldsymbol{\omega}^T \phi(\mathbf{x}) = 0 \tag{23.78}$$

Using (23.53) and (23.74), the primal objective function in feature space can be written as

$$\min J(\boldsymbol{\omega}) = \frac{1}{2} \|\boldsymbol{\omega}\|^2 + C \sum_{i=1}^{n} L(y_i, h(x_i)) \tag{23.79}$$

where $L$ is the loss function $L = \max(0, 1 - y_i h(\mathbf{x}_i))^k$.

The gradient at $\boldsymbol{\omega}$ is given as

$$\nabla_{\boldsymbol{\omega}} = \boldsymbol{\omega} + C \sum_{i=1}^{n} \frac{\partial L(y_i, h(x_i))}{\partial h(x_i)} \cdot \frac{\partial h(x_i)}{\partial \boldsymbol{\omega}} \tag{23.80}$$

where

$$\frac{\partial h(x_i)}{\partial \boldsymbol{\omega}} = \frac{\partial \boldsymbol{\omega}^T \phi(\mathbf{x}_i)}{\partial \boldsymbol{\omega}} = \phi(\mathbf{x}_i) \tag{23.81}$$

At the optimal solution, the gradient vanishes, i.e., $\nabla J(\boldsymbol{\omega}) = 0$, which yields

$$\begin{aligned} \boldsymbol{\omega} &= -C \sum_{i=1}^{n} \frac{\partial L(y_i, h(x_i))}{\partial h(x_i)} \cdot \phi(\mathbf{x}_i) \\ &= \sum_{i=1}^{n} \beta_i \phi(\mathbf{x}_i) \end{aligned} \tag{23.82}$$

where $\beta_i$ is the coefficient of the point $\phi(\mathbf{x}_i)$ in feature space. In other words, the optimal weight vector in feature space is expressed as a linear sum of the points $\phi(\mathbf{x}_i)$ in feature space.

Using (23.82), the distance to the hyperplane in feature space can be expressed as

$$y_i h(\mathbf{x}_i) = y_i \boldsymbol{\omega}^T \phi(\mathbf{x}_i) = y_i \sum_{j=1}^{n} \beta_j K(\mathbf{x}_j, \mathbf{x}_i) = y_i \mathbf{K}_i^T \boldsymbol{\beta} \tag{23.83}$$

where $\mathbf{K} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^{n}$ is the $n \times n$ kernel matrix, $\mathbf{K}_i$ is the $i$-th column of $\mathbf{K}$, and $\boldsymbol{\beta} = \big(\beta_1, \cdots, \beta_n\big)^T$ is the coefficient vector.

Plugging (23.82) and (23.83) into (23.79), with quadratic loss (i.e., $k = 2$), yields the primal kernel SVM formulation purely in terms of the kernel matrix

$$\begin{aligned} \min\ J(\boldsymbol{\beta}) &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_i\beta_j K(\mathbf{x}_i,\mathbf{x}_j) + C\sum_{i=1}^{n} max(0, 1 - y_i\mathbf{K}_i^T\boldsymbol{\beta})^2 \\ &= \frac{1}{2}\boldsymbol{\beta}^T\mathbf{K}\boldsymbol{\beta} + C\sum_{y_i\mathbf{K}_i^T\boldsymbol{\beta}<1}(1 - y_i\mathbf{K}_i^T\boldsymbol{\beta})^2 \qquad (23.84) \end{aligned}$$

The gradient of (23.84) with respect to $\boldsymbol{\beta}$ can be computed as

$$\begin{aligned} \nabla_{\boldsymbol{\beta}} = \frac{\partial J(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} &= \mathbf{K}\boldsymbol{\beta} - 2C\sum_{y_i\mathbf{K}_i^T\boldsymbol{\beta}<1} y_i\mathbf{K}_i(1 - y_i\mathbf{K}_i^T\boldsymbol{\beta}) \\ &= \mathbf{K}\boldsymbol{\beta} + 2C\sum_{y_i\mathbf{K}_i^T\boldsymbol{\beta}<1}(\mathbf{K}_i\mathbf{K}_i^T)\,\boldsymbol{\beta} - 2C\sum_{y_i\mathbf{K}_i^T\boldsymbol{\beta}<1} y_i\mathbf{K}_i \qquad (23.85) \\ &= (\mathbf{K} + 2C\mathbf{S})\,\boldsymbol{\beta} - 2C\mathbf{v} \qquad (23.86) \end{aligned}$$

where the vector $\mathbf{v} \in \mathbb{R}^n$ and the matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ are given as

$$\mathbf{v} = \sum_{y_i\mathbf{K}_i^T\boldsymbol{\beta}<1} y_i\mathbf{K}_i$$

$$\mathbf{S} = \sum_{y_i\mathbf{K}_i^T\boldsymbol{\beta}<1} \mathbf{K}_i\mathbf{K}_i^T$$

Furthermore, the *Hessian matrix* is given as

$$\mathbf{H}_{\boldsymbol{\beta}} = \frac{\partial\nabla_{\boldsymbol{\beta}}}{\partial\boldsymbol{\beta}} = \mathbf{K} + 2C\mathbf{S}$$

We can now minimize $J(\boldsymbol{\beta})$ by Newton optimization using the following update rule

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t\mathbf{H}_{\boldsymbol{\beta}}^{-1}\nabla_{\boldsymbol{\beta}} \qquad (23.87)$$

Note that if $\mathbf{H}$ is singular, i.e., if it does not have an inverse, then we add a small *ridge* to the diagonal to regularize it. That is, we make $\mathbf{H}$ invertible as follows

$$\mathbf{H}_{\boldsymbol{\beta}} = \mathbf{H}_{\boldsymbol{\beta}} + \lambda\mathbf{I}$$

where $\lambda > 0$ is some small positive ridge value.

Once $\boldsymbol{\beta}$ has been found, it is easy to classify any test point $\mathbf{z}$ as follows

$$\begin{aligned} \hat{y} &= sign\left(\boldsymbol{\omega}^T\phi(\mathbf{z})\right) \\ &= sign\left(\sum_{i=1}^{n}\beta_i\phi(\mathbf{x}_i)^T\phi(z)\right) \\ &= sign\left(\sum_{i=1}^{n}\beta_i K(\mathbf{x}_i,\mathbf{z})\right) = sign(\boldsymbol{\beta}^T\mathbf{K}_{\mathbf{z}}) \qquad (23.88) \end{aligned}$$

where $\mathbf{K_z}$ is the column vector of the kernel function of each point $\mathbf{x}_i$ with the test point $\mathbf{z}$.

**Algorithm 23.3**: Primal Kernel SVM Algorithm: Newton Optimization, Quadratic Loss

```
SVM-PRIMAL-KERNEL ($\mathbf{D}, K, C, \epsilon$):
1  foreach $\mathbf{x}_i \in \mathbf{D}$ do
2      $\mathbf{x}_i \leftarrow \begin{pmatrix}\mathbf{x}_i \\ 1\end{pmatrix}$ // map to $\mathbb{R}^{d+1}$
3  $\mathbf{K} \leftarrow \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1,\dots,n}$ // compute kernel matrix
4  $t \leftarrow 0$
5  $\boldsymbol{\beta}_0 \leftarrow (0, \dots, 0)^T$ // initialize $\boldsymbol{\beta}_t \in \mathbb{R}^n$
6  repeat
7      $\mathbf{v} \leftarrow \sum_{y_i(\mathbf{K}_i^T \boldsymbol{\beta}_t)<1} y_i \mathbf{K}_i$
8      $\mathbf{S} \leftarrow \sum_{y_i(\mathbf{K}_i^T \boldsymbol{\beta}_t)<1} \mathbf{K}_i \mathbf{K}_i^T$
9      $\nabla \leftarrow (\mathbf{K} + 2C\mathbf{S})\boldsymbol{\beta}_t - 2C\mathbf{v}$ // gradient
10     $\mathbf{H} \leftarrow \mathbf{K} + 2C\mathbf{S}$ // Hessian
11     $\boldsymbol{\beta}_{t+1} \leftarrow \boldsymbol{\beta}_t - \eta_t \mathbf{H}^{-1}\nabla$ // Newton update rule
12     $t \leftarrow t + 1$
13 until $\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\| \le \epsilon$
```

The Newton optimization algorithm for kernel SVM in the primal is given in Algorithm 23.3. The step size $\eta_t$ is set to 1 by default, as in the linear case. In each iteration, the method first computes the gradient and Hessian (lines 7–10). Next, the Newton update rule is used to obtain the updated coefficient vector $\boldsymbol{\beta}_{t+1}$ (line 11). The iterations continue until there is very little change in $\boldsymbol{\beta}$.

**Example 23.10 (Primal SVM: Quadratic Kernel):** Figure 23.8 plots the hyperplanes obtained using the dual and primal approaches on the Iris dataset projected onto the first two principal components. The task is to separate `iris versicolor` from the others, the same as in Example 23.8. Since a linear kernel is not suitable for this task, we employ the quadratic kernel. We further set $C = 10$ and $\epsilon = 0.0001$, with the quadratic loss function. The dual solution $h_d$ (black contours) and the primal solution $h_p$ (gray contours) are as follows

$$h_d(\mathbf{x}) : 1.4x_1^2 + 1.34x_1x_2 - 0.05 * x_1 + 0.66x_2^2 - 0.96 * x_2 - 2.66 = 0$$
$$h_p(\mathbf{x}) : 0.87x_1^2 + 0.64x_1x_2 - 0.5x_1 + 0.43x_2^2 - 1.04 * x_2 - 2.398 = 0$$

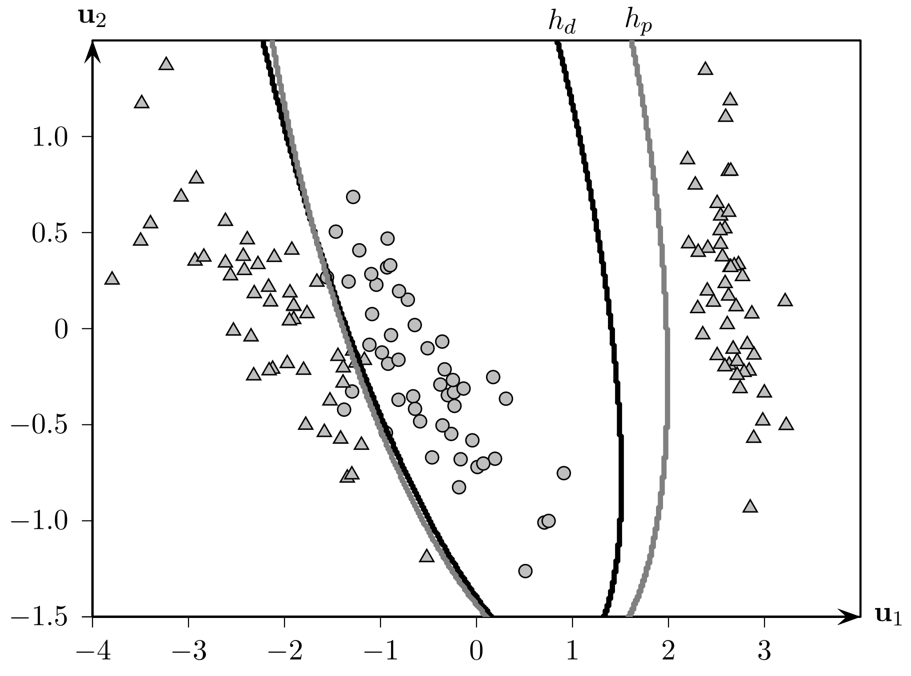

Figure 23.8: SVM Quadratic Kernel: Dual and Primal

While the solutions are not identical, they are close, especially on the left decision boundary.

## Annotated References