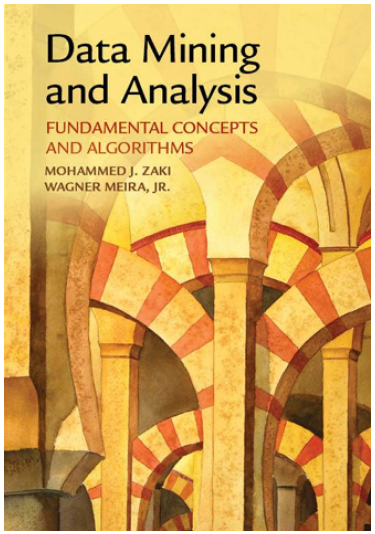
DSTA class 6: Support vector machines

Slides adapted from from Ch. 21 of M. J. Zaki and W. Meira, CUP, 2012.

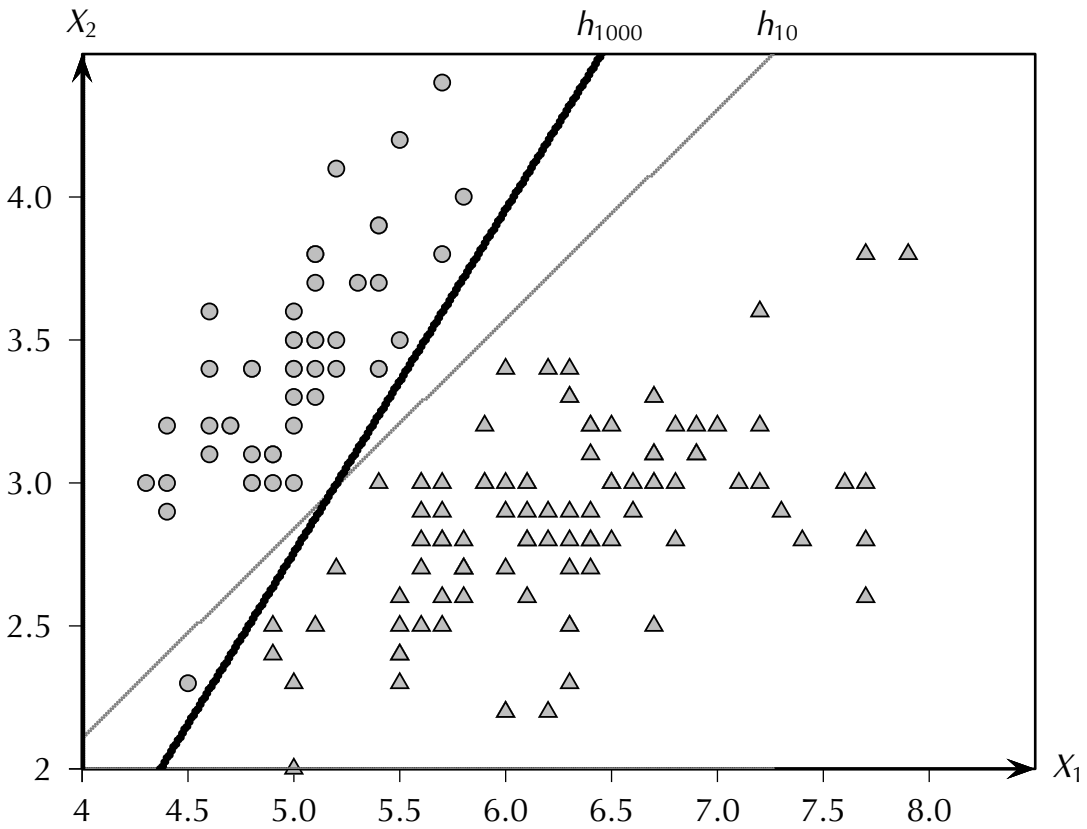
`http://www.dataminingbook.info/`



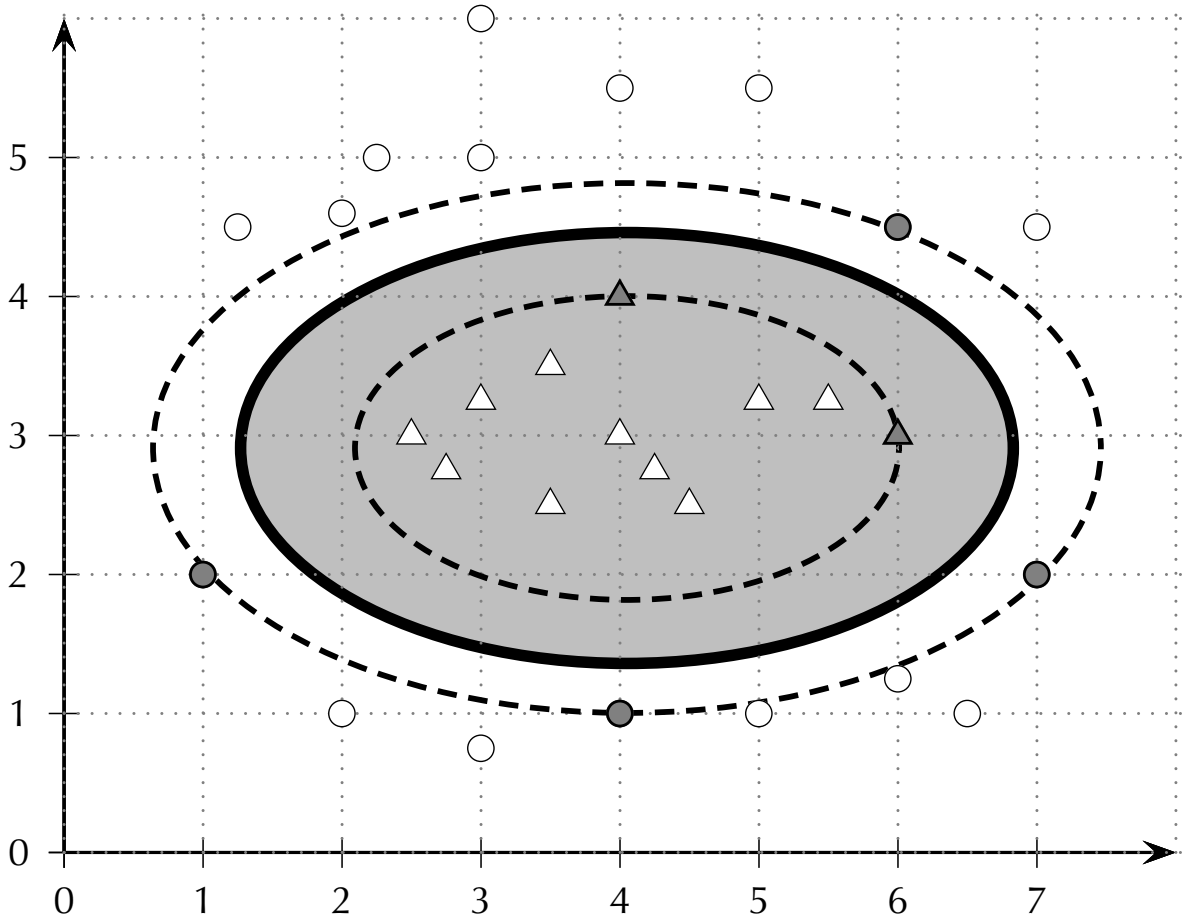
Download the text from the DSTA class page.

Motivation I: linear separation of classes

c_1: `Iris-setosa` (circles) and c_2: other types of Iris flowers (triangles)



Motivation II: non-linear kernel SVM



Hyperplanes

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

A hyperplane in d dimensions is given as the set of all points $\mathbf{x} \in \mathbb{R}^d$ that satisfy the equation $h(\mathbf{x}) = 0$, where $h(\mathbf{x})$ is the *hyperplane function*:

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b$$

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

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

The weight vector $\mathbf{w}$ specifies the direction that is orthogonal or normal to the hyperplane, which fixes the orientation of the hyperplane, whereas the bias b fixes the offset of the hyperplane in the d-dimensional space, i.e., where the hyperplane intersects each of the axes:

$$w_i x_i = -b \quad \text{or} \quad x_i = \frac{-b}{w_i}$$

Separating Hyperplane

A hyperplane splits the d-dimensional data space into two *half-spaces*.

A dataset is said to be *linearly separable* if each half-space has points only from a single class.

If the input dataset is linearly separable, then we can find a *separating* hyperplane $h(\mathbf{x}) = 0$, 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 $h(\mathbf{x}_i) > 0$.

The hyperplane function $h(\mathbf{x})$ thus serves as a linear classifier or a linear discriminant, which predicts the class y for any given point $\mathbf{x}$, according to the decision rule:

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

Geometry of a Hyperplane: Distance

Consider a point $\mathbf{x} \in \mathbb{R}^d$ that does not lie on the hyperplane. Let $\mathbf{x}_p$ be the orthogonal projection of $\mathbf{x}$ on the hyperplane, and let $\mathbf{r} = \mathbf{x} - \mathbf{x}_p$. Then we can write $\mathbf{x}$ as

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

where r is the *directed distance* of the point $\mathbf{x}$ from $\mathbf{x}_p$.
To obtain an expression for r, consider the value $h(\mathbf{x})$, we have:

$$h(\mathbf{x}) = h\left(\mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) = \mathbf{w}^T\left(\mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + b = r\|\mathbf{w}\|$$

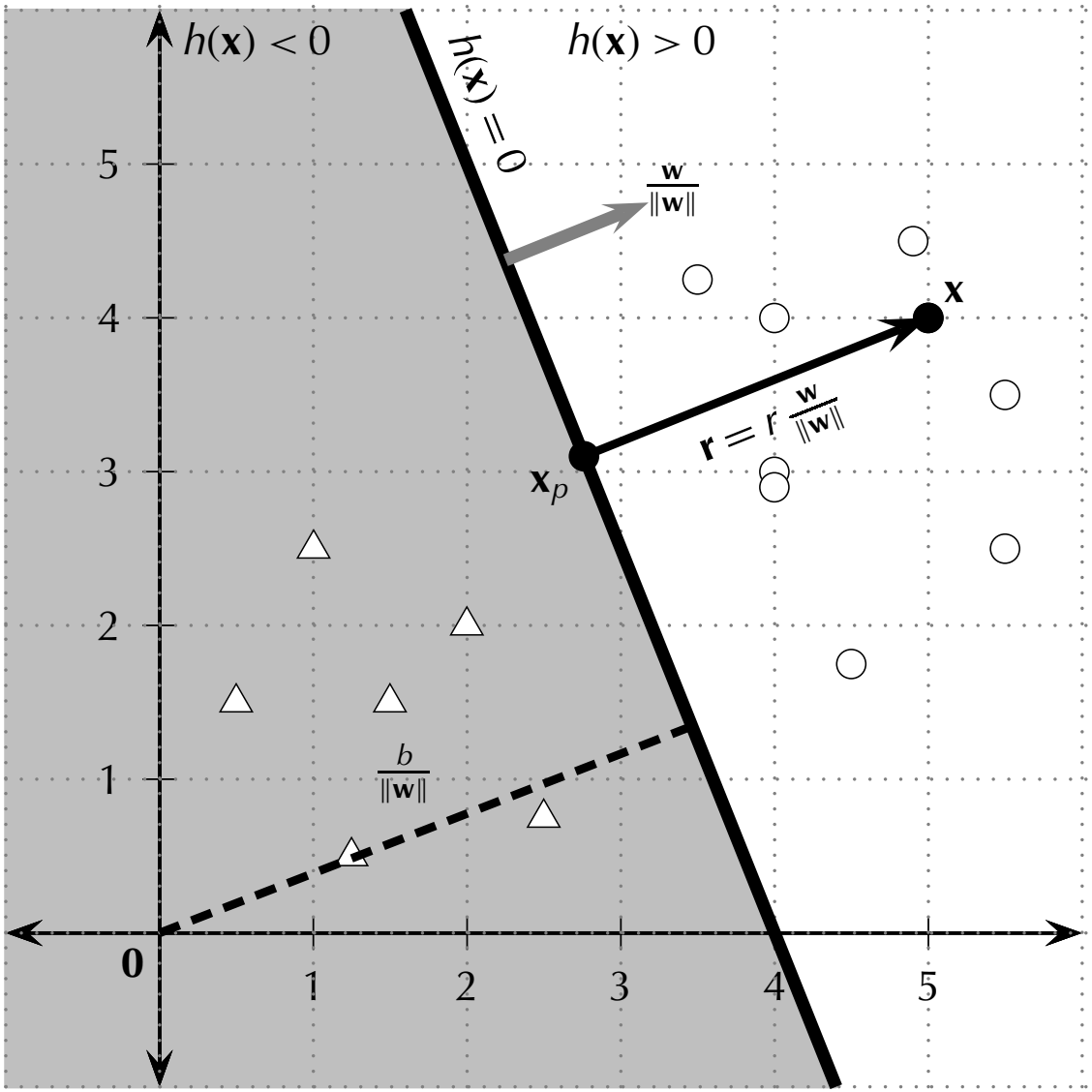
The directed distance r of point $\mathbf{x}$ to the hyperplane is thus:

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

To obtain distance, which must be non-negative, we multiply r by the class label y_i of the point $\mathbf{x}_i$ because when $h(\mathbf{x}_i) < 0$, the class is -1, and when $h(\mathbf{x}_i) > 0$ the class is $+1$:

$$\delta_i = \frac{y_i h(\mathbf{x}_i)}{\|\mathbf{w}\|}$$

Geometry of a Hyperplane in 2D



Margin and Support Vectors

The distance of a point $\mathbf{x}$ from the hyperplane $h(\mathbf{x}) = 0$ is thus given as

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

The *margin* is the minimum distance of a point from the separating hyperplane:

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

All the points (or vectors) that achieve the minimum distance are called *support vectors* for the hyperplane. They satisfy the condition:

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

where y^* is the class label for $\mathbf{x}^*$.

Canonical Hyperplane

Multiplying the hyperplane equation 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, we choose the scalar $s = \frac{1}{y^*(\mathbf{w}^T\mathbf{x}^* + b)}$ so that the absolute distance of a support vector from the hyperplane is 1, i.e., the margin is

$$\delta^* = \frac{y^*(\mathbf{w}^T\mathbf{x}^* + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

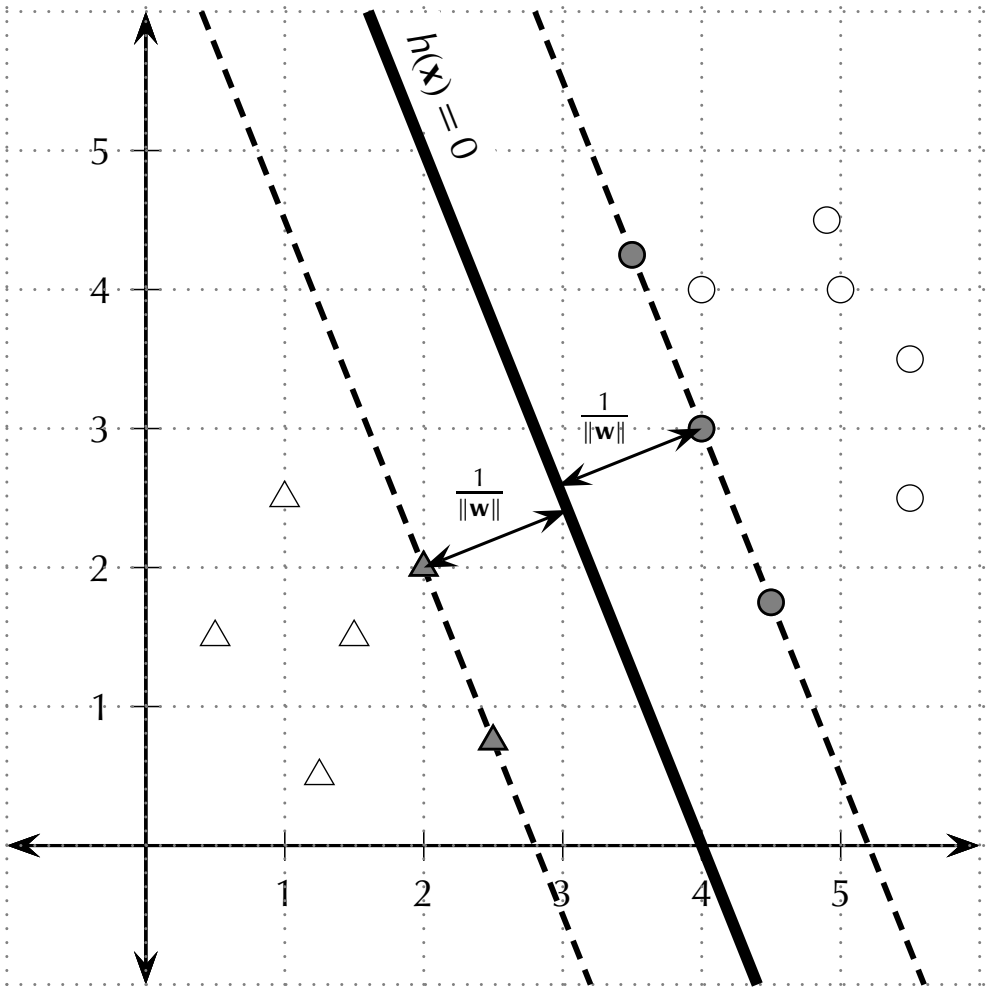
For the canonical hyperplane, for each support vector $\mathbf{x}_i^*$ (with label y_i^*), we have $y_i^* h(\mathbf{x}_i^*) = 1$, and for any point that is not a support vector we have $y_i h(\mathbf{x}_i) > 1$. Over all points, we have

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

Separating Hyperplane: Margin and Support Vectors

Shaded points are support vectors

Canonical hyperplane: $h(x) = 5/6x + 2/6y - 20/6 = 0.334x + 0.833y - 3.332$



SVM: Linear and Separable Case

Assume that the points are linearly separable, that is, there exists a separating hyperplane that perfectly classifies each point.

The goal of SVMs is to choose the canonical hyperplane, h^*, that yields the maximum margin among all possible separating hyperplanes

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

We can obtain an equivalent minimization formulation:

$$\textbf{Objective Function: } \min_{\mathbf{w},b} \left\{ \frac{\|\mathbf{w}\|^2}{2} \right\}$$

$$\textbf{Linear Constraints: } y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1, \quad \forall \mathbf{x}_i \in \mathbf{D}$$

SVM: Linear and Separable Case

We turn the constrained SVM optimization into an unconstrained one by introducing a Lagrange multiplier α_i for each constraint. The new objective function, called the *Lagrangian*, then becomes

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

L should be minimized with respect to $\mathbf{w}$ and b, and it should be maximized with respect to α_i.
Taking the derivative of L with respect to $\mathbf{w}$ and b, and setting those to zero, we obtain

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

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

We can see that $\mathbf{w}$ 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.

Further, the sum of the signed Lagrange multipliers, $\alpha_i y_i$, must be zero.

SVM: Linear and Separable Case

Incorporating $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$ into the Lagrangian we obtain the new *dual Lagrangian* objective function, which is specified purely in terms of the Lagrange multipliers:

$$\textbf{Objective Function: } \max_{\boldsymbol{\alpha}} \; L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\textbf{Linear Constraints: } \alpha_i \geq 0, \; \forall i \in \mathbf{D}, \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$ is the vector comprising the Lagrange multipliers.

L_{dual} is a convex quadratic programming problem (note the $\alpha_i \alpha_j$ terms), which admits a unique optimal solution.

SVM: Linear and Separable Case

Once we have obtained the α_i values for $i = 1, \ldots, n$, we can solve for the weight vector $\mathbf{w}$ and the bias b. Each of the Lagrange multipliers α_i satisfies the KKT conditions at the optimal solution:

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

which gives rise to two cases:

1. $\alpha_i = 0$, or
2. $y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 = 0$, which implies $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$

This is a very important result because if $\alpha_i > 0$, then $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$, and thus 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 $\alpha_i = 0$, that is, if a point is not a support vector, then $\alpha_i = 0$.

Linear and Separable Case: Weight Vector and Bias

Once we know α_i for all points, we can compute the weight vector $\mathbf{w}$ by taking the summation only for the support vectors:

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

Only the support vectors determine $\mathbf{w}$, since $\alpha_i = 0$ for other points.
To compute the bias b, we first compute one solution b_i, per support vector, as follows:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1, \text{ which implies } b_i = \frac{1}{y_i} - \mathbf{w}^T\mathbf{x}_i = y_i - \mathbf{w}^T\mathbf{x}_i$$

The bias b is taken as the average value:

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

SVM Classifier

Given the optimal hyperplane function $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, for any new point $\mathbf{z}$, we predict its class as

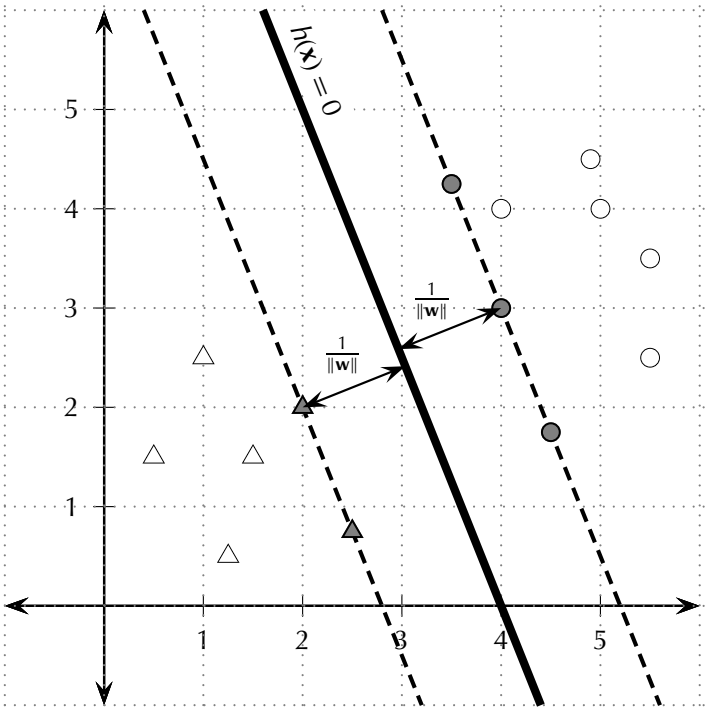
$$\hat{y} = \text{sign}(h(\mathbf{z})) = \text{sign}(\mathbf{w}^T\mathbf{z} + b)$$

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

Example Dataset: Separable Case

$\mathbf{x}_i$	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

Optimal Separating Hyperplane



Solving the L_{dual} quadratic program yields

$\mathbf{x}_i$	x_{i1}	x_{i2}	y_i	α_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

The weight vector and bias are:

$$\mathbf{w} = \sum_{i,\alpha_i>0} \alpha_i y_i \mathbf{x}_i = \begin{pmatrix} 0.833 \\ 0.334 \end{pmatrix}$$

$$b = \text{avg}\{b_i\} = -3.332$$

The optimal hyperplane is given as follows:

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

Soft Margin SVM: Linear and Nonseparable Case

The assumption that the dataset be perfectly linearly separable is unrealistic. SVMs can handle non-separable points by introducing *slack variables* ξ_i as follows:

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

where $\xi_i \geq 0$ is the slack variable for point $\mathbf{x}_i$, which indicates how much the point violates the separability condition, that is, the point may no longer be at least $1/\|\mathbf{w}\|$ away from the hyperplane.

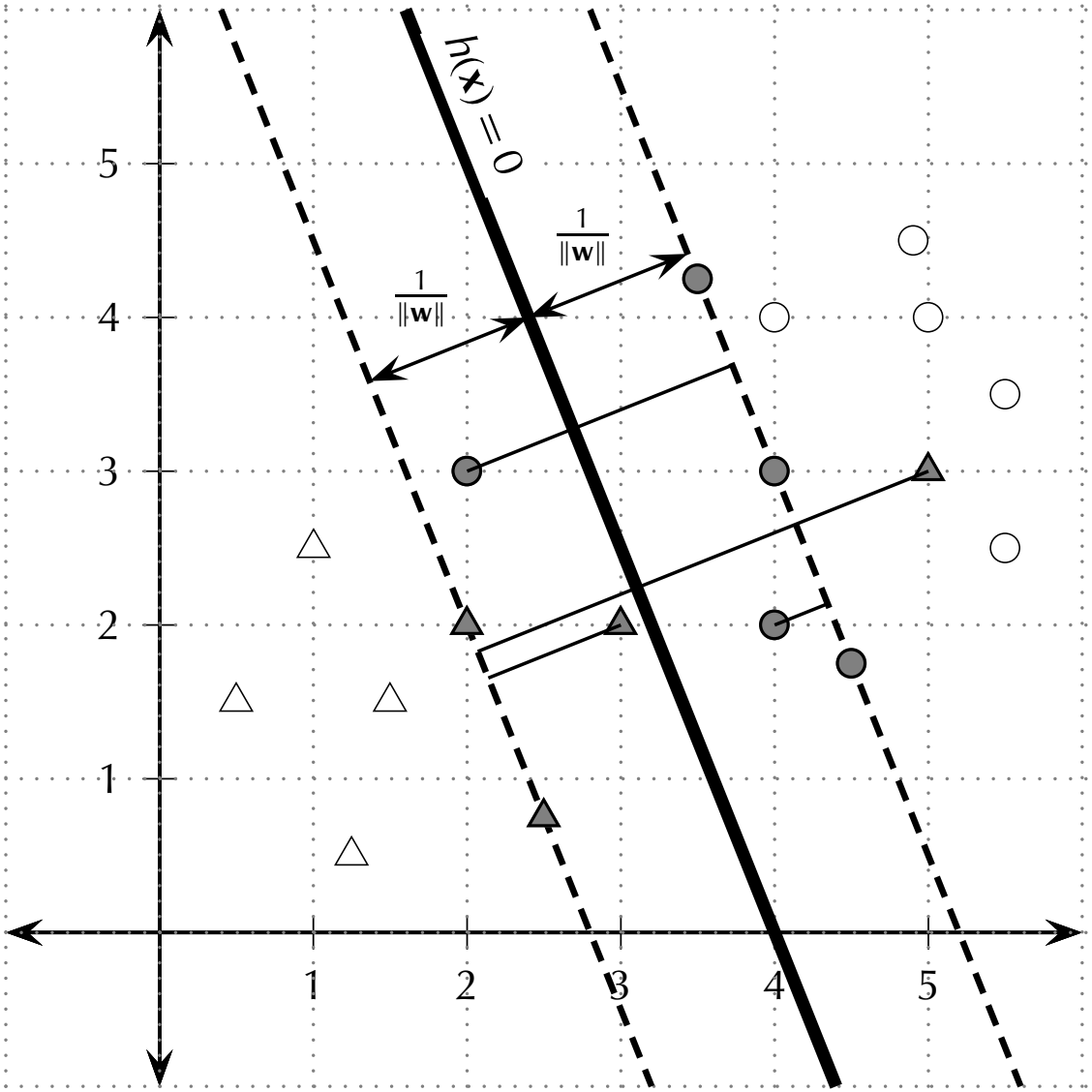
The slack values indicate three types of points. If $\xi_i = 0$, then the corresponding point $\mathbf{x}_i$ is at least $\frac{1}{\|\mathbf{w}\|}$ away from the hyperplane.

If $0 < \xi_i < 1$, then the point is within the margin and still correctly classified, that is, it is on the correct side of the hyperplane.

However, if $\xi_i \geq 1$ then the point is misclassified and appears on the wrong side of the hyperplane.

Soft Margin Hyperplane

Shaded points are the support vectors



SVM: Soft Margin or Linearly Non-separable Case

In the nonseparable case, also called the *soft margin* the SVM objective function is

$$\textbf{Objective Function:} \quad \min_{\mathbf{w},b,\xi_i} \left\{ \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{n} (\xi_i)^k \right\}$$

$$\textbf{Linear Constraints:} \quad y_i \left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, \quad \forall \mathbf{x}_i \in \mathbf{D}$$

$$\xi_i \geq 0 \quad \forall \mathbf{x}_i \in \mathbf{D}$$

where C and k are constants that incorporate the cost of misclassification.

The term $\sum_{i=1}^{n} (\xi_i)^k$ gives the *loss*, that is, an estimate of the deviation from the separable case.

The scalar C is a *regularization constant* that controls the trade-off between maximizing the margin or minimizing the loss. 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.

SVM: Soft Margin Loss Function

The constant k governs the form of the loss. 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.

Hinge Loss: Assuming $k = 1$, the SVM dual Lagrangian is given as

$$\max_{\boldsymbol{\alpha}} \; L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

The only difference from the separable case is that $0 \leq \alpha_i \leq C$.

Quadratic Loss: Assuming $k = 2$, the dual objective is:

$$\max_{\boldsymbol{\alpha}} \; L_{dual} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i^T \mathbf{x}_j + \frac{1}{2C} \delta_{ij} \right)$$

where δ is the *Kronecker delta* function, defined as $\delta_{ij} = 1$ if and only if $i = j$.

Example Dataset: Linearly Non-separable Case

$\mathbf{x}_i$	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
$\mathbf{x}_{15}$	4	2	+1
$\mathbf{x}_{16}$	2	3	+1
$\mathbf{x}_{17}$	3	2	−1
$\mathbf{x}_{18}$	5	3	−1

Example Dataset: Linearly Non-separable Case

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

$\mathbf{x}_i$	x_{i1}	x_{i2}	y_i	α_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

The optimal hyperplane is given as follows:

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

Example Dataset: Linearly Non-separable Case

The slack $\xi_i = 0$ for all points that are not support vectors, and also for those support vectors that are on the margin. Slack is positive only for the remaining support vectors and it can be computed as: $\xi_i = 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)$.
Thus, for all support vectors not on the margin, we have

$\mathbf{x}_i$	$\mathbf{w}^T\mathbf{x}_i$	$\mathbf{w}^T\mathbf{x}_i + b$	$\xi_i = 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)$
$\mathbf{x}_{15}$	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

The total slack is given as

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

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

Kernel SVM: Nonlinear Case

The linear SVM approach can be used for datasets with a nonlinear decision boundary via the kernel trick.

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 nonlinear transformation ϕ.

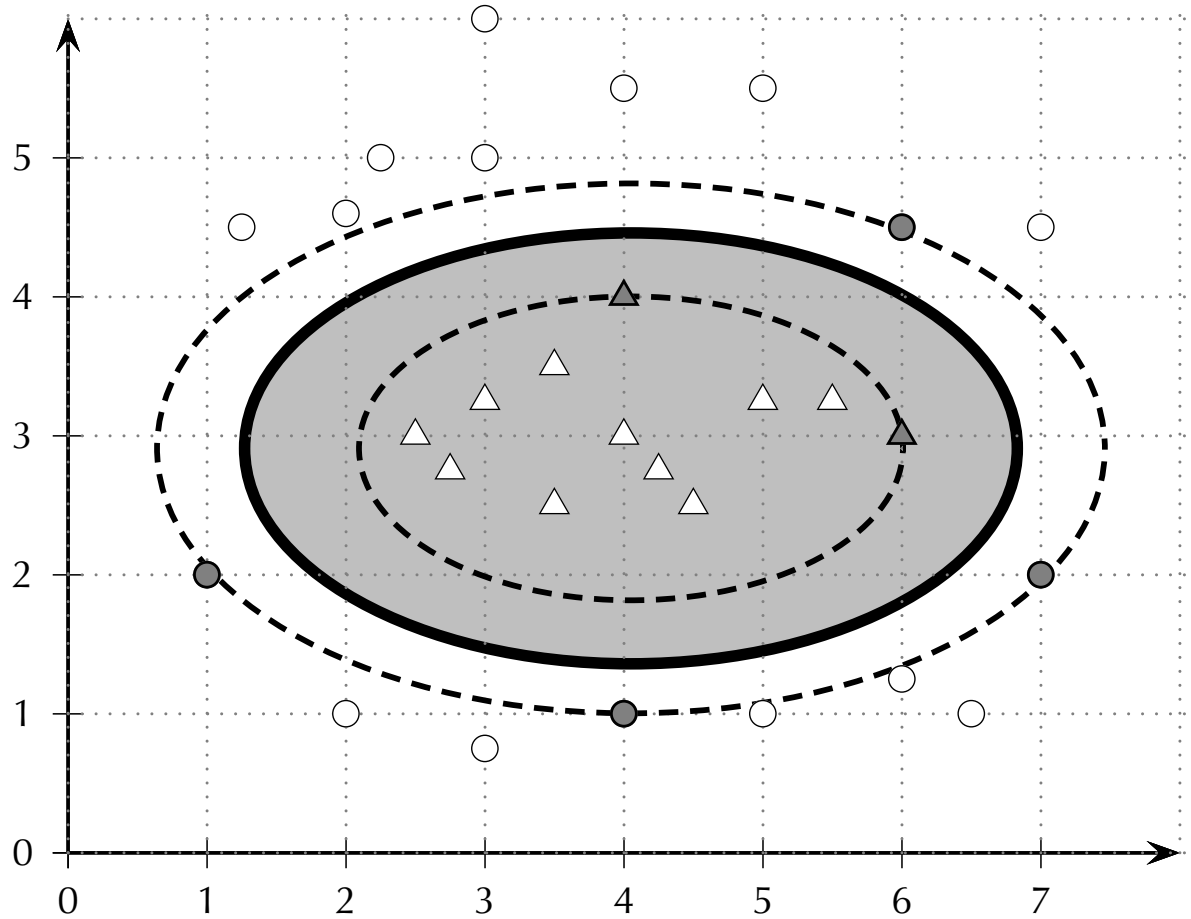
Given the extra flexibility, it is more likely that the points $\phi(\mathbf{x}_i)$ might be linearly separable in the feature space.

A linear decision surface in feature space actually corresponds to a nonlinear decision surface in the input space.

Further, the kernel trick allows us to carry out all operations via the kernel function computed in input space, rather than having to map the points into feature space.

Nonlinear SVM

There is no linear classifier that can discriminate between the points. However, there exists a perfect quadratic classifier that can separate the two classes.



Nonlinear SVMs: Kernel Trick

To apply the kernel trick for nonlinear SVM classification, we have to show that all operations require only the kernel function:

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

Applying ϕ 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 SVM objective function in feature space is given as

$$\textbf{Objective Function:} \quad \min_{\mathbf{w}, b, \xi_i} \left\{ \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{n} (\xi_i)^k \right\}$$

$$\textbf{Linear Constraints:} \quad y_i \left(\mathbf{w}^T \phi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i, \text{ and } \xi_i \geq 0, \quad \forall \mathbf{x}_i \in \mathbf{D}$$

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

Nonlinear SVMs: Kernel Trick

For hinge loss, the dual Lagrangian in feature space is given as

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

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

Subject to the constraints that $0 \leq \alpha_i \leq C$, and $\sum_{i=1}^{n} \alpha_i y_i = 0$.

The dual Lagrangian depends only on the dot product between two vectors in 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 $\mathbf{K} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1,\ldots,n}$.

For quadratic loss, the dual Lagrangian corresponds to the use of a new kernel

$$K_q(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j + \frac{1}{2C} \delta_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{2C} \delta_{ij}$$

Nonlinear SVMs: Weight Vector and Bias

We cannot directly obtain the weight vector without transforming the points, since

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

However, we can compute the bias via kernel operations, since

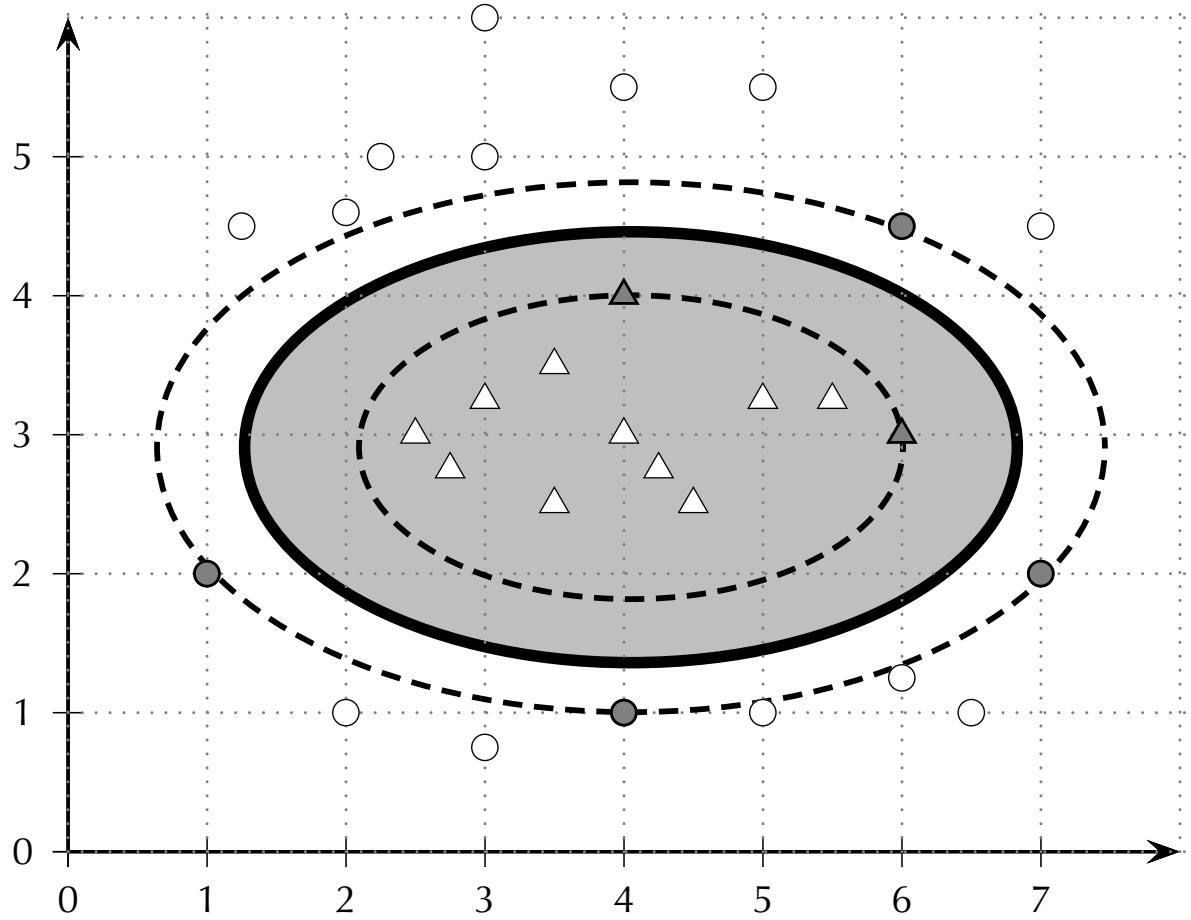
$$b_i = y_i - \mathbf{w}^T \phi(\mathbf{x}_i) = y_i - \sum_{\alpha_j > 0} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i)$$

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

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

All SVM operations can be carried out in terms of the kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$. Thus, any nonlinear kernel function can be used to do nonlinear classification in the input space.

Nonlinear SVM: Inhomogeneous Quadratic Kernel

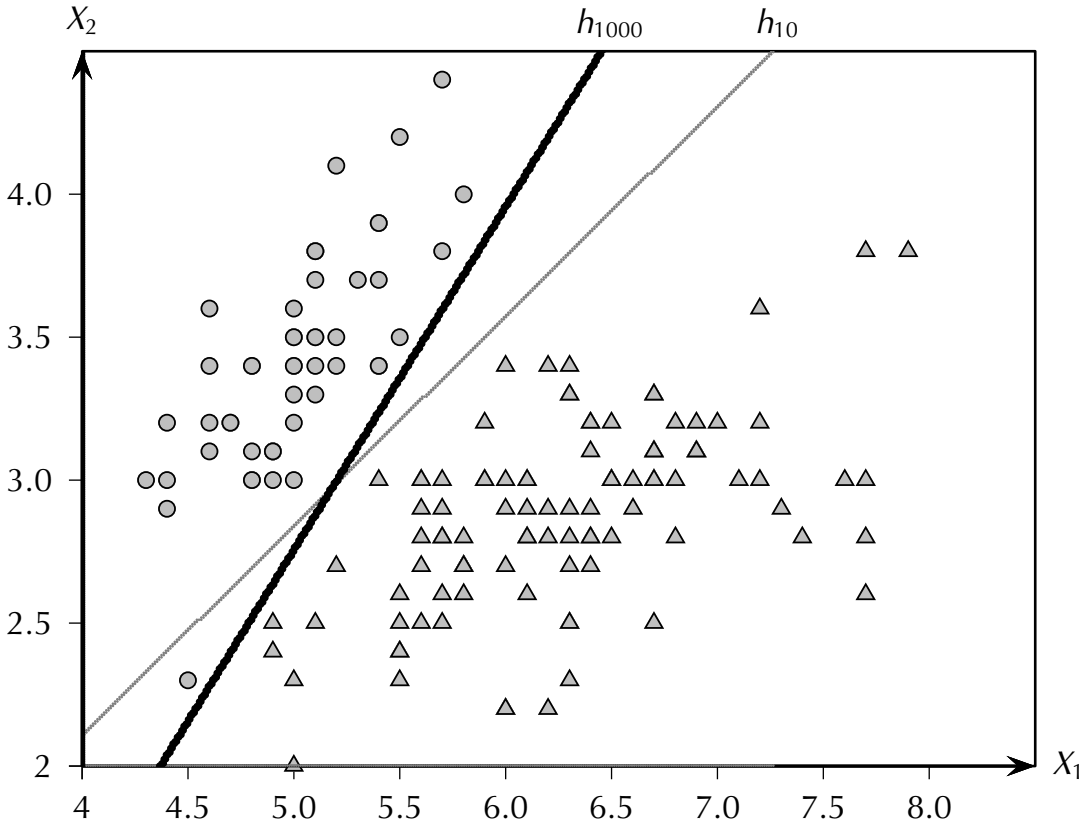


The optimal quadratic hyperplane is obtained by setting $C = 4$, and using an inhomogeneous polynomial kernel of degree $q = 2$:

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

SVM Dual Algorithm: Iris Data – Linear Kernel

c_1: `Iris-setosa` (circles) and c_2: other types of Iris flowers (triangles)



Hyperplane h_{10} uses $C = 10$ and h_{1000} uses $C = 1000$:

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

h_{10} has a larger margin, but also a larger slack; h_{1000} has a smaller margin, but it minimizes the slack.

SVM Dual Algorithm: Quadratic versus Linear Kernel

c_1: `Iris-versicolor` (circles) and c_2: other types of Iris flowers (triangles)