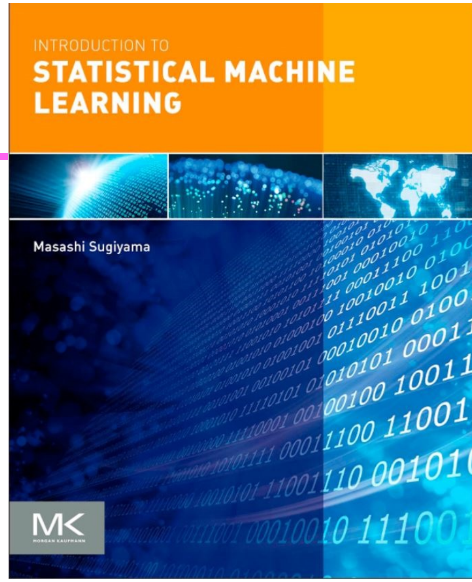
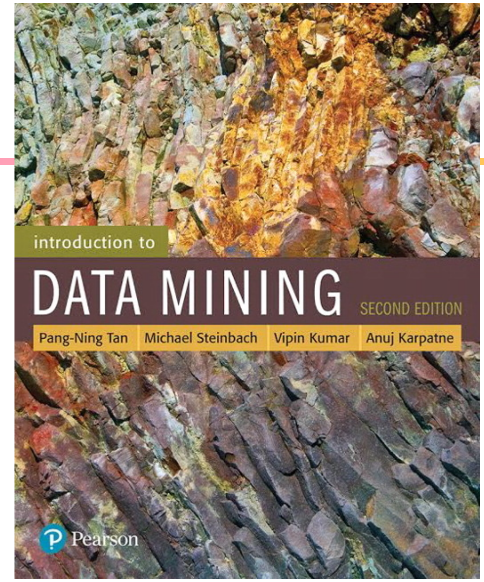
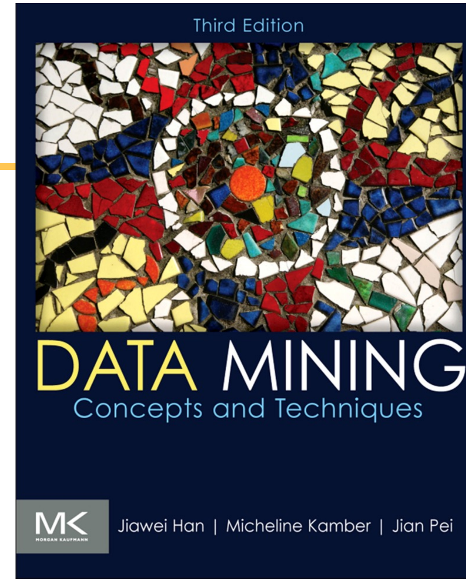
Topic 4







Support Vector Machines

(SVMs)

[Jiawei Han, Jian Pei, Hanghang Tong. 2022. *Data Mining Concepts and Techniques*. 4th Ed. Morgan Kaufmann. ISBN: 0128117605.]

[Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar. 2018. *Introduction to Data Mining*. 2nd Ed. Pearson. ISBN: 0133128903.]

[Masashi Sugiyama. 2015. *Introduction to Statistical Machine Learning*. Morgan Kaufmann. ISBN: 9780128021217.]

Contents

1. Basic Concepts of SVM

• A classification technique that has received considerable attention is **support vector machine** (SVM).

• Practical applications of SVM: handwritten digit recognition, speaker identification, text categorization, and so on.

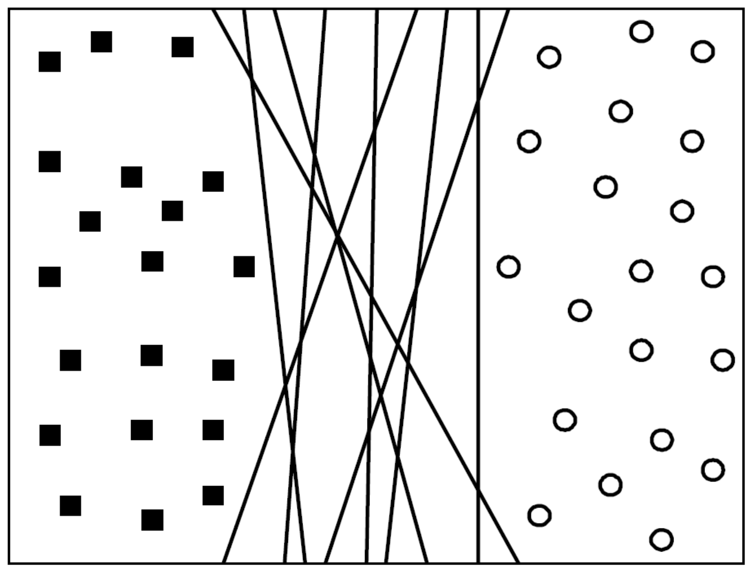
• SVM works very well with high-dimensional data and avoids the curse of dimensionality problem.

1. Basic Concepts of SVM

- SVM creates decision boundary (DB) (i.e., optimal separating hyperplane) by using a subset of training examples called **support vectors**.
- Linear SVM uses **maximal margin hyperplane** (MMH) to classify linearly separable data.
- SVMs can also be used to classify non-linearly separable data (a.k.a. linearly inseparable data).

1. Basic Concepts of SVM

• Figure below shows plot of data set containing examples belonging to two different classes, represented as squares and circles.



Possible DBs for a linearly separable data set

1. Basic Concepts of SVM

• We can find **hyperplane** ($\mathbf{w} \cdot \mathbf{x} + b = 0$) such that all squares reside on one side of hyperplane and all circles reside on other side.

• As shown in the figure above, there are infinitely possible hyperplanes.

• Although their training errors are zero, there is no guarantee that different hyperplanes will perform equally well on previously unseen examples.

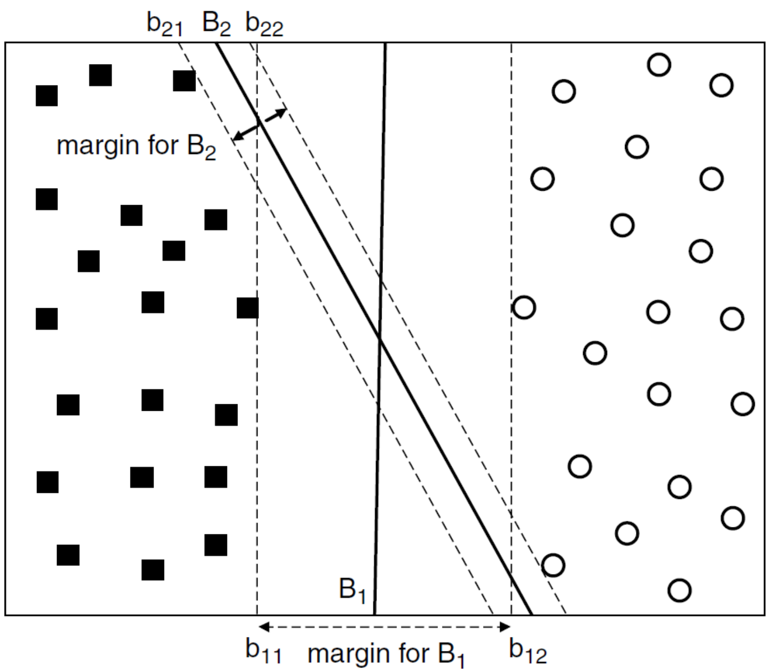
/* $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, $\mathbf{w} = \{w_1, w_2, \ldots, w_d\}$

$\mathbf{w} \cdot \mathbf{x} = w_1x_1 + w_2x_2 + \ldots + w_dx_d$ */

1. Basic Concepts of SVM

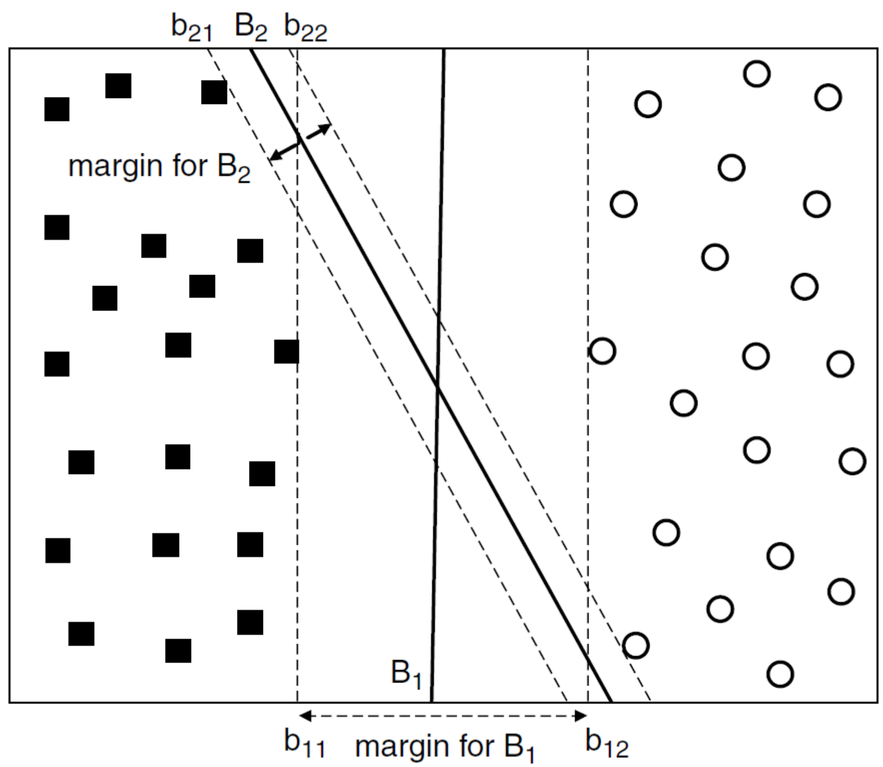
• SVM classifier must choose one of these hyperplanes to represent its DB, based on how well they are expected to perform on test examples.

• Consider two DBs B_1 and B_2, shown in figure below.



1. Basic Concepts of SVM

- Both DBs B_1 and B_2 can separate training examples into their respective classes without committing any misclassification errors.



1. Basic Concepts of SVM

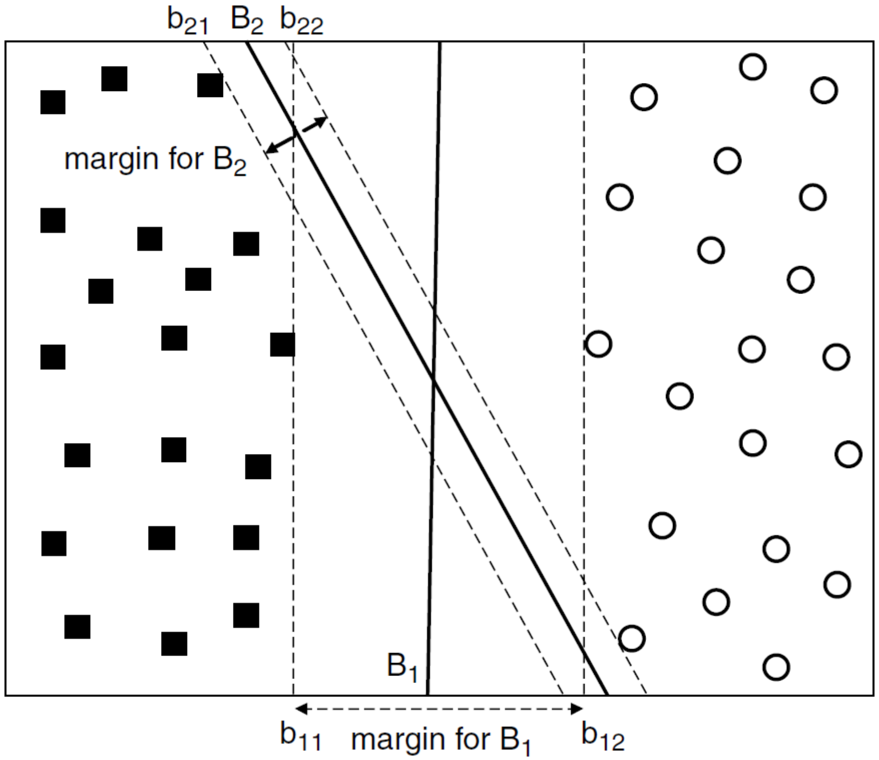
• Each DB B_i is associated with a pair of hyperplanes, denoted as b_{i1} and b_{i2}, respectively.

• b_{i1} is obtained by moving a parallel hyperplane away from DB until it touches closest square(s), whereas b_{i2} is obtained by moving the parallel hyperplane until it touches closest circle(s).

• Distance between these two hyperplanes b_{i1} and b_{i2} is known as the **margin** of the SVM classifier.

1. Basic Concepts of SVM

- The figure shows that the margin for B_1 is considerably larger than that for B_2.
- B_1 is **maximum margin hyperplane** (MMH) of training instances.



1. Basic Concepts of SVM

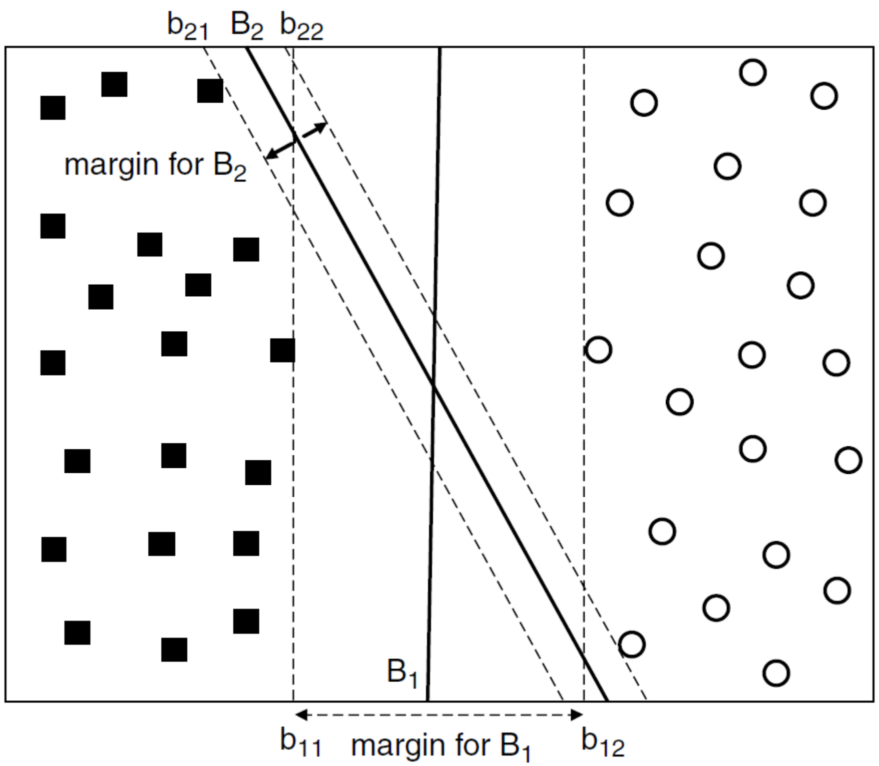
- A DB with large margin tends to have better generalization errors than a DB with small margin.
- SVM classifier that produces DB with small margin is more susceptible to model overfitting and tends to generalize poorly on previously unseen examples.

Contents

1. Basic Concepts of SVM
2. **Linear SVM: Separable Case** (linear DB) (hard-margin SVM)
3. Linear SVM: Nonseparable Case (linear DB) (soft-margin SVM)
4. Nonlinear SVM (non-linear DB)

2. Linear SVM: Separable Case

- Linear SVM is classifier that searches for hyperplane with largest margin, which is why it is often known as **maximal margin classifier** (MMH).



2. Linear SVM: Separable Case

- Consider binary classification problem consisting of N training examples.
- Each example is denoted by tuple $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ corresponds to attribute set for the i^{th} example, $i = 1, 2, \ldots, N$.
- Let $y_i \in \{-1, 1\}$ denote its class label.
- For a data point $\mathbf{x}$, we have the notation $(\mathbf{x}, y)$, where $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, $y \in \{-1, 1\}$.

2. Linear SVM: Separable Case

$\mathbf{w} \cdot \mathbf{x} = w_1x_1 + w_2x_2 + \ldots + w_dx_d$

• DB of a linear SVM classifier can be written in the following form:

$$\mathbf{w} \cdot \mathbf{x} + b = 0, \quad (5.28)$$

where **weight vector** $\mathbf{w} = (w_1, w_2, \ldots, w_d)$ and scalar b (a.k.a. **bias**) are parameters of the model.

• Example:

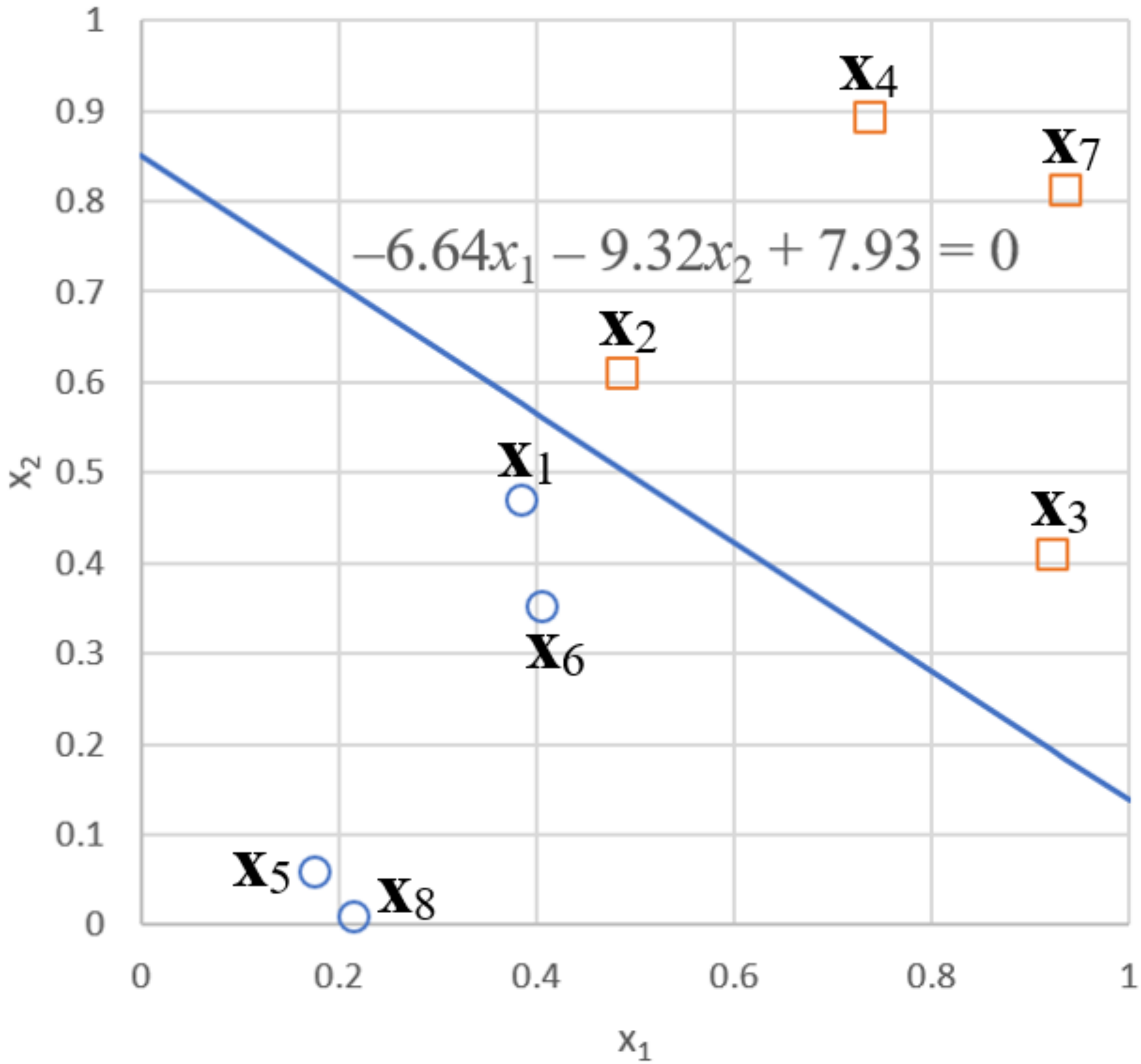
- In 2-dimensional space, we have $\mathbf{w} = (w_1, w_2)$, $\mathbf{x} = (x_1, x_2)$, equation of DB: $w_1x_1 + w_2x_2 + b = 0$.

- Assume that $\mathbf{w} = (-6.64, -9.32)$, $b = 7.93$, we have equation of DB: $-6.64x_1 - 9.32x_2 + 7.93 = 0$.

2. Linear SVM: Separable Case

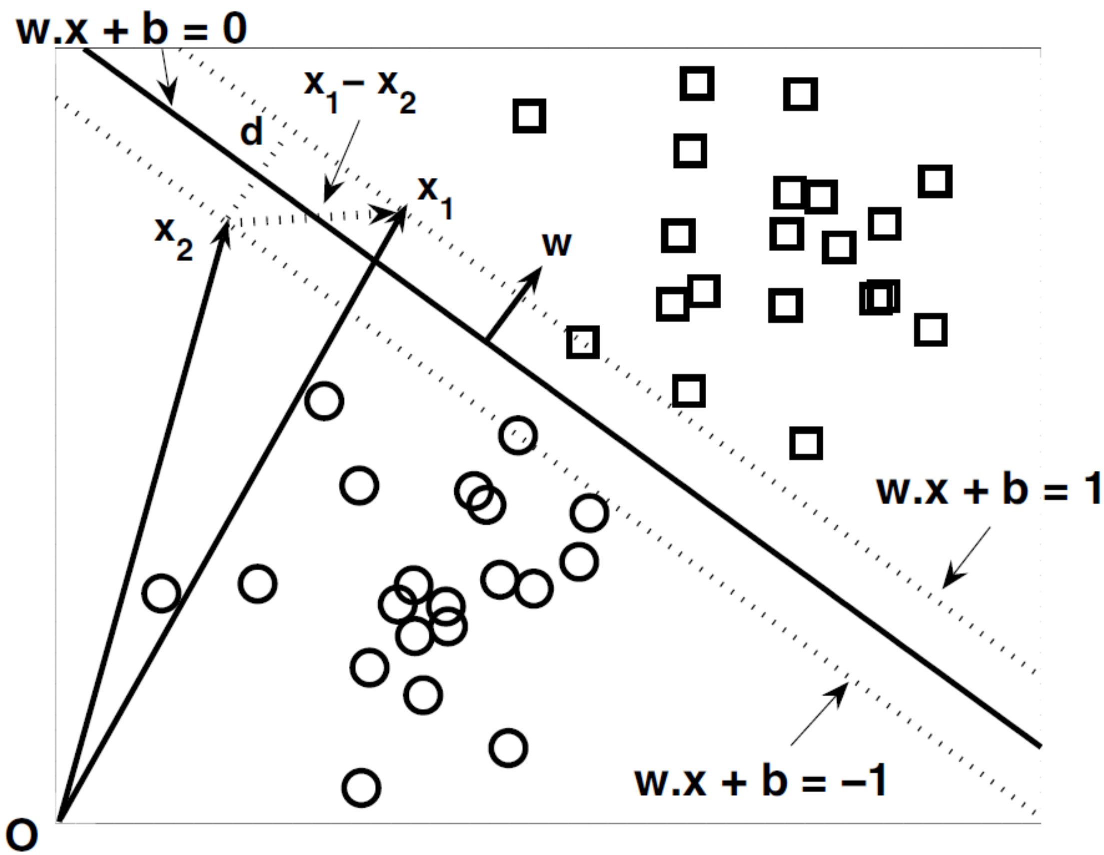
- $\mathbf{w} \cdot \mathbf{x} + b = 0$, (5.28), $\mathbf{w} = (w_1, w_2)$, $\mathbf{x} = (x_1, x_2)$, DB: $w_1x_1 + w_2x_2 + b = 0$.

-



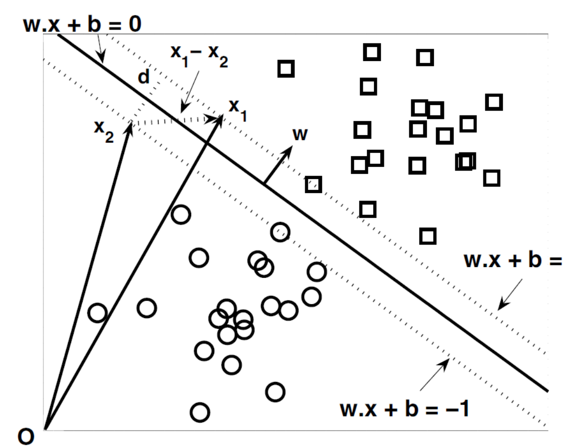
2. Linear SVM: Separable Case

• Figure below shows two-dimensional training set consisting of squares and circles.



2. Linear SVM: Separable Case

- DB that bisects training examples into their respective classes is illustrated with solid line (i.e., $\mathbf{w} \cdot \mathbf{x} + b = 0$, or $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$).
- Any example located along DB must satisfy equation $\mathbf{w} \cdot \mathbf{x} + b = 0$.
- Direction of **w** must be perpendicular to DB.



2. Linear SVM: Separable Case

• For example, if $\mathbf{x}_a$ and $\mathbf{x}_b$ are two points located on DB, then

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

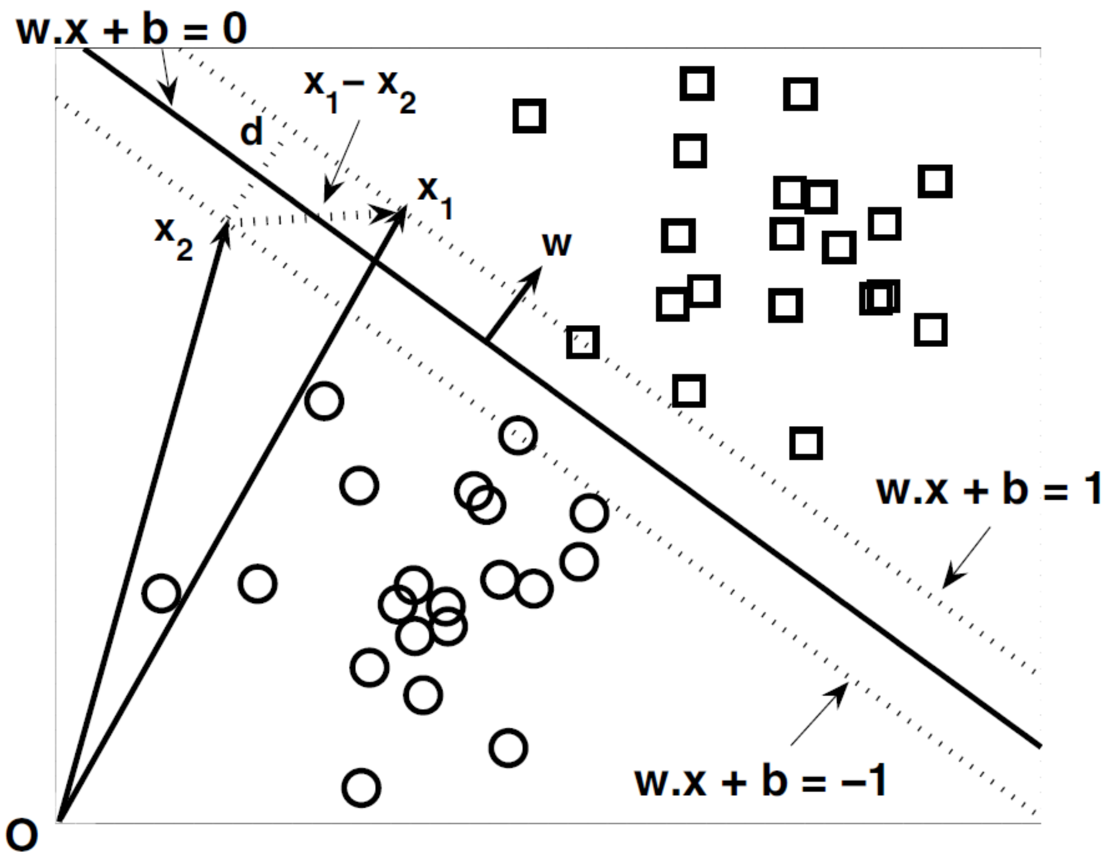
• Subtracting two equations will yield the following:

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

where $\mathbf{x}_b - \mathbf{x}_a$ is vector parallel to DB and is directed from $\mathbf{x}_a$ to $\mathbf{x}_b$.

2. Linear SVM: Separable Case

- Since dot product $\mathbf{w} \cdot (\mathbf{x}_b - \mathbf{x}_a)$ is zero, direction of **w** must be perpendicular to DB.



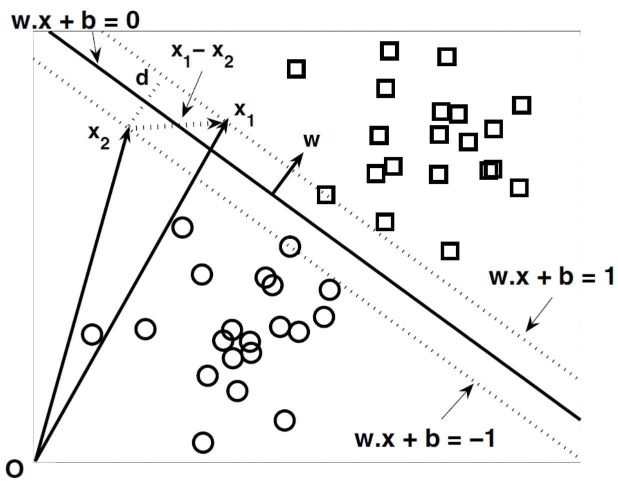
2. Linear SVM: Separable Case

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (5.28)$$

- If we label all squares as class +1 and all circles as class –1, then, given parameters **w** and b of DB, we can predict class label y for any test example **z** in the following way:

- $y = 1$ if $\text{sign}(\mathbf{w} \cdot \mathbf{z} + b) > 0$ (or $\mathbf{w} \cdot \mathbf{z} + b \gtrsim 1$)

- $y = -1$ if $\text{sign}(\mathbf{w} \cdot \mathbf{z} + b) < 0$ (or $\mathbf{w} \cdot \mathbf{z} + b \lesssim -1$)

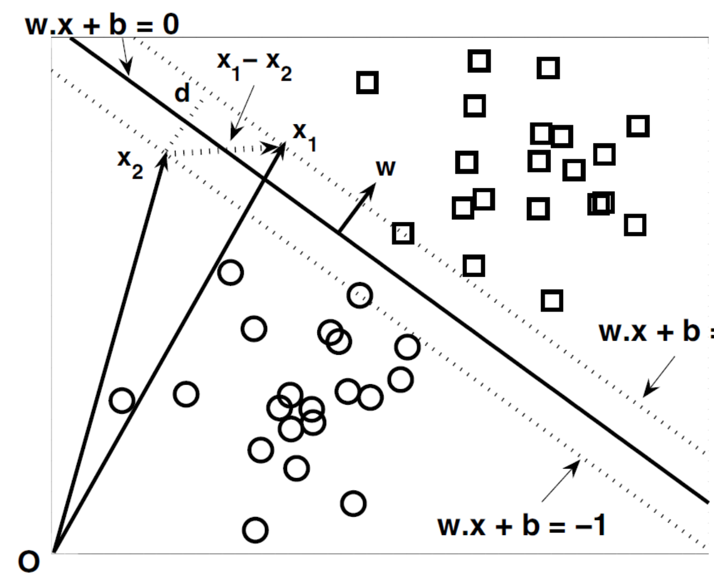


$$b_{i1}: \mathbf{w} \cdot \mathbf{x} + b = 1 \ (5.32)$$

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

Compute Margin of Linear SVM

- Consider square and circle that are closest to DB.
- Since square is located above DB, square must satisfy equation $\mathbf{w} \cdot \mathbf{x}_s + b = k$ for $k > 0$ (e.g., $k = 1$).
- Since circle is located below DB, circle must satisfy equation $\mathbf{w} \cdot \mathbf{x}_c + b = k'$ for $k' < 0$ (e.g., $k' = -1$).



Compute Margin of Linear SVM

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (5.28)$$

- We can rescale parameters **w** and b of DB so that two parallel hyperplanes b_{i1} and b_{i2} can be expressed as follows.

$$b_{i1}\text{: } \mathbf{w} \cdot \mathbf{x} + b = 1 \ (5.32)$$

$$b_{i2}\text{: } \mathbf{w} \cdot \mathbf{x} + b = -1 \ (5.33)$$

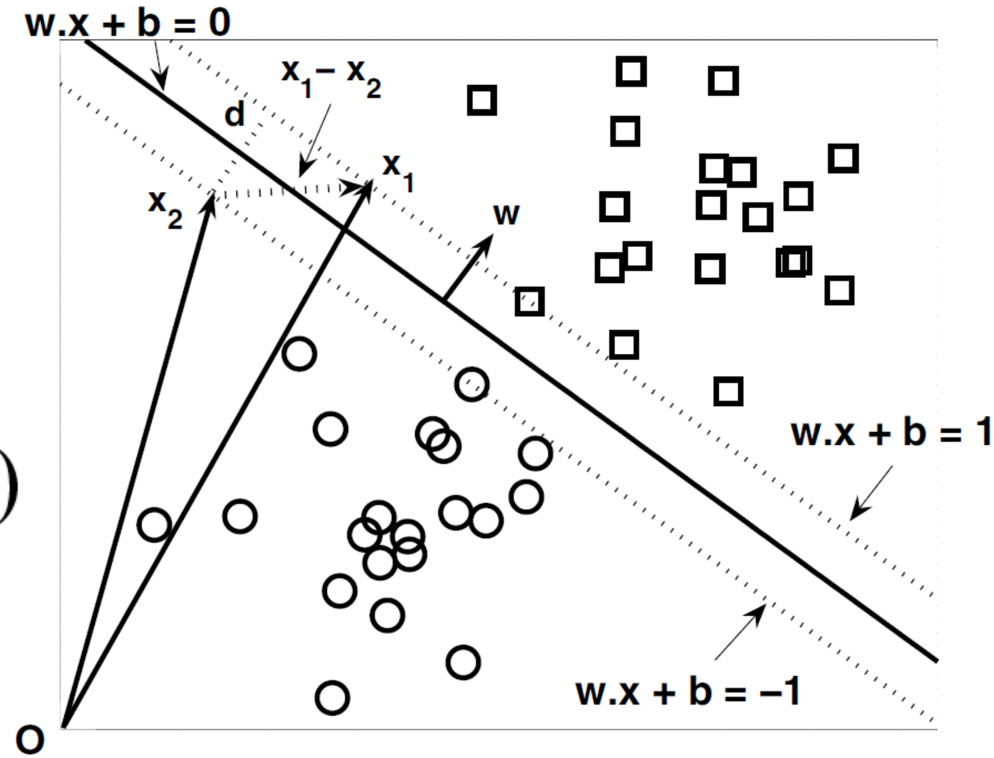
- The margin of DB is given by distance d between these two hyperplanes b_{i1} and b_{i2}.
- We will show that $d = 2 / ||\mathbf{w}||$, where

$$\mathbf{w} = (w_1, w_2, \ldots, w_n)$$

$$||\mathbf{w}|| = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$$

Compute Margin of Linear SVM

- To compute the margin, let $\mathbf{x}_1$ be a data point located on b_{i1} and $\mathbf{x}_2$ be a data point on b_{i2}, as shown in figure below.

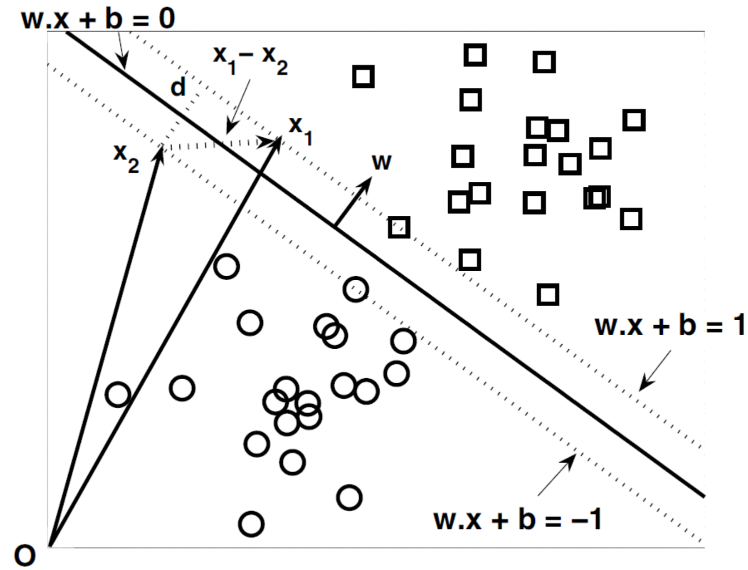


$$b_{i1}: \mathbf{w} \cdot \mathbf{x} + b = 1 \quad (5.32)$$
$$b_{i2}: \mathbf{w} \cdot \mathbf{x} + b = -1 \quad (5.33)$$

Compute Margin of Linear SVM

- Upon substituting these points into equations b_{i1}: $\mathbf{w} \cdot \mathbf{x} + b = 1$ and b_{i2}: $\mathbf{w} \cdot \mathbf{x} + b = -1$, the margin d can be computed by subtracting the second equation from the first equation. That is,

$$(\mathbf{w} \cdot \mathbf{x}_1 + b = 1) - (\mathbf{w} \cdot \mathbf{x}_2 + b = -1)$$



Compute Margin of Linear SVM

$(\mathbf{w} \cdot \mathbf{x}_1 + b = 1) - (\mathbf{w} \cdot \mathbf{x}_2 + b = -1)$

$\rightarrow \mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2$

$\rightarrow ||\mathbf{w}||\ ||\mathbf{x}_1 - \mathbf{x}_2||\ \cos(\theta) = 2$ (geometric definition)

(θ is the angle between $\mathbf{w}$ and $\mathbf{x}_1 - \mathbf{x}_2$)

We have $\cos(\theta) = d\ /\ ||\mathbf{x}_1 - \mathbf{x}_2||$

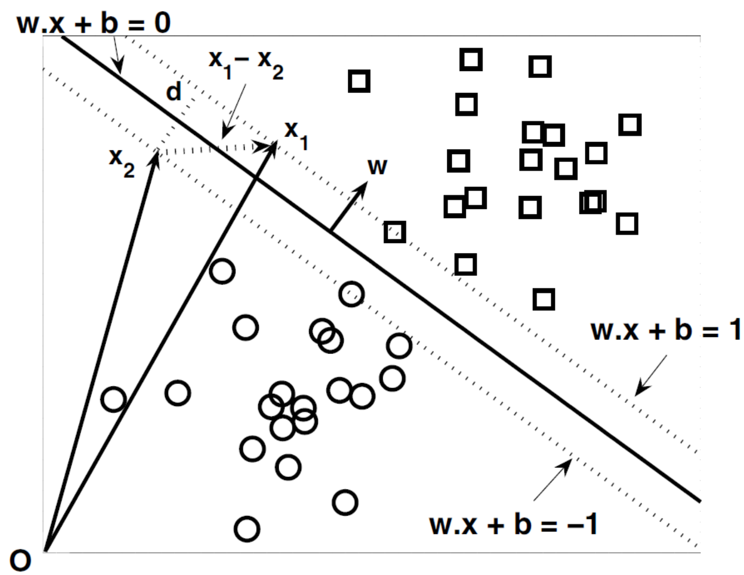
$\rightarrow d = ||\mathbf{x}_1 - \mathbf{x}_2||\ \cos(\theta)$

$\rightarrow ||\mathbf{w}|| \times d = 2$

$\rightarrow d = 2\ /\ ||\mathbf{w}||$ (5.34)

$\mathbf{w} = (w_1, w_2, \ldots, w_n)$

$||\mathbf{w}|| = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$



Training/Learning Linear SVM

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (5.28)$$

- Training phase of SVM involves estimating parameters **w** and *b* of DB from training data.
- Parameters must be chosen in such a way that the following two conditions are met:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \text{ if } y_i = 1,$$
$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \text{ if } y_i = -1 \ (5.35)$$

Training/Learning Linear SVM

$$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij} \text{ , } b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \text{ if } y_i = 1,$$
$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \text{ if } y_i = -1 \text{ (5.35)}$$

- These required conditions are

\- all training instances from class $y = 1$ (i.e., squares) must be located on or above hyperplane $\mathbf{w} \cdot \mathbf{x}_i + b = 1$,

\- all training instances from class $y = -1$ (i.e., circles) must be located on or below hyperplane $\mathbf{w} \cdot \mathbf{x}_i + b = -1$.

$$b_{i1}\text{: } \mathbf{w} \cdot \mathbf{x} + b = 1 \text{ (5.32)}$$
$$b_{i2}\text{: } \mathbf{w} \cdot \mathbf{x} + b = -1 \text{ (5.33)}$$

Training/Learning Linear SVM

$$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij}\ ,\ b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

- Both inequalities can be summarized in more compact form as follows:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1,\ i = 1, 2, ..., N.\ (5.36)$$

($y_i \in \{-1, 1\}$, **hard-margin** SVM)

- SVM requires that margin d of its DB ($d = 2 / ||\mathbf{w}||$ (5.34)) must be maximal.

- Maximizing margin d is equivalent to minimizing following objective function: $f(\mathbf{w}) = ||\mathbf{w}||^2 / 2$. (5.37)

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \text{ if } y_i = 1$$
$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \text{ if } y_i = -1\ (5.35)$$

Training/Learning Linear SVM

$$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij}\ ,\ b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

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

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

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

- Constrained minimization problem above is known as **convex** optimization problem, which can be solved using the standard **Lagrange multiplier** method.

Solve Constrained Minimization Problem

$$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij} \text{ , } b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

- **Step 1**: The (original) objective function $f(\mathbf{w}) = \|\mathbf{w}\|^2 / 2$ (5.37) is recast in a form that takes into account the inequality constraints $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ imposed on its solutions.

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

($y_i \in \{-1, 1\}$, **hard-margin** SVM)

Solve Constrained Minimization Problem

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

- The new objective function is known as the (primary/primal) Lagrangian for minimization problem:

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right),$$

(involves λ_i, training data $\mathbf{x}_i$, and $\mathbf{w}$, b)

(i.e., $L_P(\mathbf{w}, b, \lambda_i)$)

where parameters λ_i are called the **Lagrange multipliers**.

Solve Constrained Minimization Problem

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

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i (y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1) \quad (5.38)$$

(involves λ_i, training data $\mathbf{x}_i$, and $\mathbf{w}$, b)

- The term $\|\mathbf{w}\|^2 / 2$ is the original objective function $f(\mathbf{w})$.

- The term $\sum_{i=1}^{N} \lambda_i \left(y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1\right)$ captures the inequality constraints ($y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$).

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

($y_i \in \{-1, 1\}$, **hard-margin** SVM)

Solve Constrained Minimization Problem

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N}\lambda_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1) \quad (5.38)$$

- **Step 2**: To minimize the Lagrangian L_P, we must take the partial derivatives (or gradient) of L_P w.r.t. $\mathbf{w}$ and b and set them to zero.

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Leftrightarrow \boxed{\mathbf{w} = \sum_{i=1}^{N}\lambda_i y_i \mathbf{x}_i}\ . \ (5.39),\quad \frac{\partial L_P}{\partial b} = 0 \Leftrightarrow \sum_{i=1}^{N}\lambda_i y_i = 0.\ (5.40).$$

$w_j = \sum_{i=1}^{N}\lambda_i y_i x_{ij}$, x_{ij} is j-th component of $\mathbf{x}_i$. (5.50), e.g., $\mathbf{x}_i = (x_{i1}, x_{i2})$

- λ_i's are unknown, $\mathbf{w}$ and b cannot be solved by using equations $\mathbf{w} = \sum_{i=1}^{N}\lambda_i y_i \mathbf{x}_i$ and $\sum_{i=1}^{N}\lambda_i y_i = 0$.

Solve Constrained Minimization Problem

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

- **Step 3**: If $\lambda_i \geq 0$, inequality constraints $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$ can be transformed into a set of equality constraints $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) = 1$.
- The transformation leads to the Karush-Kuhn-Tucker (KKT) constraints on λ_i's, shown below.

$$\lambda_i \geq 0, \ (5.41)$$

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

/* $f(x) = x^\alpha, f'(x) = (x^\alpha)' = \alpha x^{\alpha-1}$

$f(x, y) = 2x^3 + 4y^5, \partial f / \partial x = 6x^2, \partial f / \partial y = 20y^4$ */

Solve Constrained Minimization Problem

b_{i1}: $\mathbf{w} \cdot \mathbf{x} + b = 1$ (5.32), b_{i2}: $\mathbf{w} \cdot \mathbf{x} + b = -1$ (5.33)

- The equality constraint $\lambda_i\, [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ (5.42) indicates that

\- If training instances $\mathbf{x}_i$'s do not reside along hyperplanes b_{i1} ($\mathbf{w} \cdot \mathbf{x} + b = 1$) or b_{i2} ($\mathbf{w} \cdot \mathbf{x} + b = -1$), λ_i's must be zero. That is, λ_i's must be zero if training instances $\mathbf{x}_i$'s do not satisfy equation $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$ (i.e., $\mathbf{w} \cdot \mathbf{x}_i + b > 1$ or $\mathbf{w} \cdot \mathbf{x}_i + b < -1$).

\- Training instance $\mathbf{x}_i$'s with λ_i's > 0 lie along the hyperplanes b_{i1} or b_{i2} and are known as **support vectors**.

Solve Constrained Minimization Problem

b_{i1}: $\mathbf{w} \cdot \mathbf{x} + b = 1$ (5.32), b_{i2}: $\mathbf{w} \cdot \mathbf{x} + b = -1$ (5.33)

- For a **support vector** $\mathbf{x}_i$ (i.e., $\lambda_i > 0$), we have

$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0 \Leftrightarrow y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$

$\Leftrightarrow b = y_i - \mathbf{w} \cdot \mathbf{x}_i$. // $y_i = \pm 1 \rightarrow 1/y_i = y_i$.

- Equations $\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$ (5.39) and $\lambda_i\,[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ (5.42) show that parameters $\mathbf{w}$ and b, which define DB, depend only on **support vectors**.

$$w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij} \quad (5.50)$$

Solve Constrained Minimization Problem

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \lambda_i (y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1) \quad (5.38)$$

- **Step 4**: Minimizing L_P (by finding $\mathbf{w}$, b, and λ_i from $\lambda_i\,[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$, $\lambda_i \geq 0$) can be simplified by transforming the primary/primal Lagrangian $L_P(\mathbf{w}, b, \lambda_i)$ into a function of the Lagrange multipliers only (this is known as the **dual** problem).

Solve Constrained Minimization Problem

$$w_j = \sum_{i:\lambda_i > 0} \lambda_i y_i x_{ij} \text{ , } b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

- Transformation: substitute equations

$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$ (5.39) and $\sum_{i=1}^{N} \lambda_i y_i = 0$ (5.40) into equation

$$L_P = \frac{1}{2} ||\mathbf{w}||^2 - \sum_{i=1}^{N} \lambda_i (y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1) \quad (5.38)$$

(i.e., $L_P(\mathbf{w}, b, \lambda_i)$)

Solve Constrained Minimization Problem

$$w_j = \sum_{i:\lambda_i > 0} \lambda_i y_i x_{ij} \text{ , } b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

- The transformation leads to the **dual Lagrangian** formulation of the maximization problem:

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (5.43)$$

$$\lambda_i \geq 0, \ \sum_{i=1}^{N} \lambda_i y_i = 0$$

(involves only λ_i and training data $\mathbf{x}_i$)

(i.e., $L_D(\lambda_i)$)

Solve Constrained Minimization Problem

$$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij} \ , \ b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

- The solutions for minimization problem $L_P(\mathbf{w}, b, \lambda_i)$ and maximization problem $L_D(\lambda_i)$ are equivalent under the Karush-Kuhn-Tucker (KKT) constraints.
- Solving the dual maximization problem $L_D(\lambda_i)$ (i.e., finding λ_i's) is beyond the scope of the course.
- λ_i can be found by using Python package cvxopt

(Martin S. Andersen: http://people.compute.dtu.dk/~mskan)

Solve Constrained Minimization Problem

$$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij}\ ,\ b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

- Once λ_i's are found, we can use equations $\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$ and $\lambda_i\ [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ to obtain feasible solutions for $\mathbf{w}$ and b (i.e., $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$).
- DB $\mathbf{w} \cdot \mathbf{x} + b = 0$ can be expressed as follows:

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

b is obtained by solving equation $\lambda_i\ [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ (5.42) for **support vectors**. (i.e., $\lambda_i > 0$).

Solve Constrained Minimization Problem

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (5.28)$$

- Reminder: For a **support vector** $\mathbf{x}_i$ (i.e., $\lambda_i > 0$), we have

$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0 \Leftrightarrow y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$

$\Leftrightarrow b = y_i - \mathbf{w} \cdot \mathbf{x}_i$. // $y_i = \pm 1 \rightarrow 1/y_i = y_i$.

- The value computed for b may not be unique. Value of b depends on support vectors used in equation $\lambda_i\,[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ (5.42).
- Average value for b is chosen.

Summary

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (5.28)$$

- Given m SVs $\mathbf{x}_i$ (i.e., $\lambda_i > 0$), $\mathbf{w} = (w_1, w_2, ..., w_d)$, $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{id})$
- Generally, we have

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i = \sum_{i:\lambda_i > 0} \lambda_i y_i \mathbf{x}_i \text{ and } b = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

- Specifically, we have $w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij} = \sum_{i:\lambda_i > 0} \lambda_i y_i x_{ij}$

where x_{ij} is the jth component of $\mathbf{x}_i$ (e.g., $\mathbf{x}_i = (x_{i1}, x_{i2})$). $b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$, where $k = 1, 2, ..., m$. b is the average of $b^{(k)}$.

Summary

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (5.28)$$

- Computing $b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$, makes the value of $b^{(k)}$ depend on the computed value of **w**.
- Alternatively, we can compute $b^{(i)}$ as

$$b^{(i)} = y_i - \sum_{j:\lambda_j>0,\lambda_i>0} \lambda_j y_j (\mathbf{x}_j \cdot \mathbf{x}_i)$$

- b is the average of $b^{(i)}$.

Example

$$w_j = \sum_{i:\lambda_i > 0} \lambda_i y_i x_{ij}\ ,\ b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

Example: Consider two-dimensional data set shown below, which contains eight training instances $\mathbf{x}_i$. Class label $y_i \in \{-1, 1\}$, λ_i: Lagrange multipliers)

Instances	x_1	x_2	y_i	λ_i
$\mathbf{x}_1$	0.3858	0.4687	1	65.5261
$\mathbf{x}_2$	0.4871	0.611	–1	65.5261
$\mathbf{x}_3$	0.9218	0.4103	–1	0
$\mathbf{x}_4$	0.7382	0.8936	–1	0
$\mathbf{x}_5$	0.1763	0.0579	1	0
$\mathbf{x}_6$	0.4057	0.3529	1	0
$\mathbf{x}_7$	0.9355	0.8132	–1	0
$\mathbf{x}_8$	0.2146	0.0099	1	0

Example

$$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij}\ ,\ b^{(k)} = y_i - \mathbf{w}\cdot\mathbf{x}_i$$

- First two instances have $\lambda_i > 0$. These instances correspond to two **support vectors** $\mathbf{x}_1$ and $\mathbf{x}_2$.

Instances	x_1	x_2	y_i	λ_i
$\mathbf{x}_1$	0.3858	0.4687	1	65.5261
$\mathbf{x}_2$	0.4871	0.611	–1	65.5261
$\mathbf{x}_3$	0.9218	0.4103	–1	0
⋮				
$\mathbf{x}_8$	0.2146	0.0099	1	0

Example

$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij}$, where x_{ij} is the jth component of $\mathbf{x}_i$ (e.g., $\mathbf{x}_i = (x_{i1}, x_{i2})$)

- Two **support vectors** $\mathbf{x}_1 = (0.3858, 0.4687)$ and $\mathbf{x}_2 = (0.4871, 0.611)$, $\lambda_1 = 65.5261$, $\lambda_2 = 65.5261$.
- Let $\mathbf{w} = (w_1, w_2)$ and b denote parameters of DB.
- Using equation $\boxed{\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i}$ (i.e. $\boxed{w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij}}$),

$\mathbf{x}_i = (x_{i1}, x_{i2})$, we can solve for w_1 and w_2 as follows.

Example

$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij}$, where x_{ij} is the jth component of $\mathbf{x}_i$ (e.g., $\mathbf{x}_i = (x_{i1}, x_{i2})$)

Instances	x_1	x_2	y_i	λ_i
$\mathbf{x}_1$	$\boxed{0.3858}$	0.4687	$\boxed{1}$	$\boxed{65.5261}$
$\mathbf{x}_2$	$\boxed{0.4871}$	0.611	$\boxed{-1}$	$\boxed{65.5261}$
$\mathbf{x}_3$	0.9218	0.4103	–1	0
⋮				
$\mathbf{x}_8$	0.2146	0.0099	1	0

$$w_1 = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij} = \sum_{i=1}^{2} \lambda_i y_i x_{i1} = \lambda_1 y_1 x_{11} + \lambda_2 y_2 x_{21}$$

$$w_1 = 65.5261 \times 1 \times 0.3858 + 65.5261 \times (-1) \times 0.4871$$
$$= -6.6378 \approx -6.64.$$

Example

$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij}$, where x_{ij} is the jth component of $\mathbf{x}_i$ (e.g., $\mathbf{x}_i = (x_{i1}, x_{i2})$)

Instances	x_1	x_2	y_i	λ_i
$\mathbf{x}_1$	0.3858	0.4687	1	65.5261
$\mathbf{x}_2$	0.4871	0.611	–1	65.5261
$\mathbf{x}_3$	0.9218	0.4103	–1	0
⋮				
$\mathbf{x}_8$	0.2146	0.0099	1	0

$$w_2 = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij} = \sum_{i=1}^{2} \lambda_i y_i x_{i2} = \lambda_1 y_1 x_{12} + \lambda_2 y_2 x_{22}$$

$$w_2 = 65.5261 \times 1 \times 0.4687 + 65.5261 \times (-1) \times 0.611$$
$$= -9.3244 \approx -9.32.$$

- Thus, $\mathbf{w} = (w_1, w_2) = (-6.64, -9.32)$.

Example

- For support vectors $\mathbf{x}_i$ (i.e., $\lambda_i > 0$), we have

$$\boxed{\begin{aligned} \mathbf{w} &= \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i = \sum_{i:\lambda_i>0} \lambda_i y_i \mathbf{x}_i \\ w_j &= \sum_{i=1}^{N} \lambda_i y_i x_{ij} = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij} \end{aligned}}$$

Example

$$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij}, \; b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

- Bias term b can be computed using equation $\lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ (5.42) for each support vector.
- Recall: for a support vector $\mathbf{x}_i$ (i.e., $\lambda_i > 0$), we have

$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0 \Leftrightarrow y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$

$\Leftrightarrow b = y_i - \mathbf{w} \cdot \mathbf{x}_i$. // $y_i = \pm 1 \rightarrow 1/y_i = y_i$.

$\mathbf{w} = (w_1, w_2) = (-6.64, -9.32)$, $\mathbf{x}_i = (x_{i1}, x_{i2})$

$b^{(k)} = y_i - w_1 \times x_{i1} - w_2 \times x_{i2}$, $k = 1, 2, \ldots, m$; m is the number of support vectors (e.g., $m = 2$), $i = 1, 2$.

Example

$$w_j = \sum_{i:\lambda_i > 0} \lambda_i y_i x_{ij}\ ,\ b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

$\mathbf{w} = (w_1, w_2) = (-6.64, -9.32)$, $\mathbf{x}_i = (x_{i1}, x_{i2})$

$b^{(k)} = y_i - w_1 \times x_{i1} - w_2 \times x_{i2}$, $k = 1, 2; i = 1, 2.$

$b^{(1)} = 1 - \mathbf{w} \cdot \mathbf{x}_1$

$= 1 - (-6.64) \times (0.3858) - (-9.32) \times (0.4687)$

$= 1 - (-6.9311) = 7.9311 \approx 7.93.$

Instances	x_1	x_2	y_i	λ_i
$\mathbf{x}_1$	$\boxed{0.3858}$	$\boxed{0.4687}$	$\boxed{1}$	65.5261
$\mathbf{x}_2$	0.4871	0.611	–1	65.5261
$\mathbf{x}_3$	0.9218	0.4103	–1	0
⋮				
$\mathbf{x}_8$	0.2146	0.0099	1	0

Example

$$w_j = \sum_{i:\lambda_i > 0} \lambda_i y_i x_{ij}\ ,\ b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

$\mathbf{w} = (w_1, w_2) = (-6.64, -9.32)$, $\mathbf{x}_i = (x_{i1}, x_{i2})$

$b^{(k)} = y_i - w_1 \times x_{i1} - w_2 \times x_{i2}$, $k = 1, 2; i = 1, 2.$

$b^{(2)} = -1 - \mathbf{w} \cdot \mathbf{x}_2$

$= -1 - (-6.64) \times (0.4871) - (-9.32) \times (0.611)$

$= -1 - (-8.9304) = 7.9305 \approx 7.93.$

Instances	x_1	x_2	y_i	λ_i
$\mathbf{x}_1$	0.3858	0.4687	1	65.5261
$\mathbf{x}_2$	$\boxed{0.4871}$	$\boxed{0.611}$	$\boxed{-1}$	65.5261
$\mathbf{x}_3$	0.9218	0.4103	–1	0
⋮				
$\mathbf{x}_8$	0.2146	0.0099	1	0

Example

$$w_j = \sum_{i:\lambda_i>0} \lambda_i y_i x_{ij}\ , \ b^{(k)} = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

$\mathbf{w} = (w_1, w_2) = (-6.64, -9.32)$, $\mathbf{x}_i = (x_{i1}, x_{i2})$

$b^{(k)} = y_i - w_1 \times x_{i1} - w_2 \times x_{i2}$, $k = 1, 2$; $i = 1, 2$.

$b^{(1)} = 7.9311 \approx 7.93$, $b^{(2)} = 7.9305 \approx 7.93$.

- Averaging these values $b^{(1)}$ and $b^{(2)}$, we obtain

$$b \approx 7.93.$$

Instances	x_1	x_2	y_i	λ_i
$\mathbf{x}_1$	0.3858	0.4687	1	65.5261
$\mathbf{x}_2$	0.4871	0.611	–1	65.5261
$\mathbf{x}_3$	0.9218	0.4103	–1	0
⋮				
$\mathbf{x}_8$	0.2146	0.0099	1	0

Example

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (5.28)$$

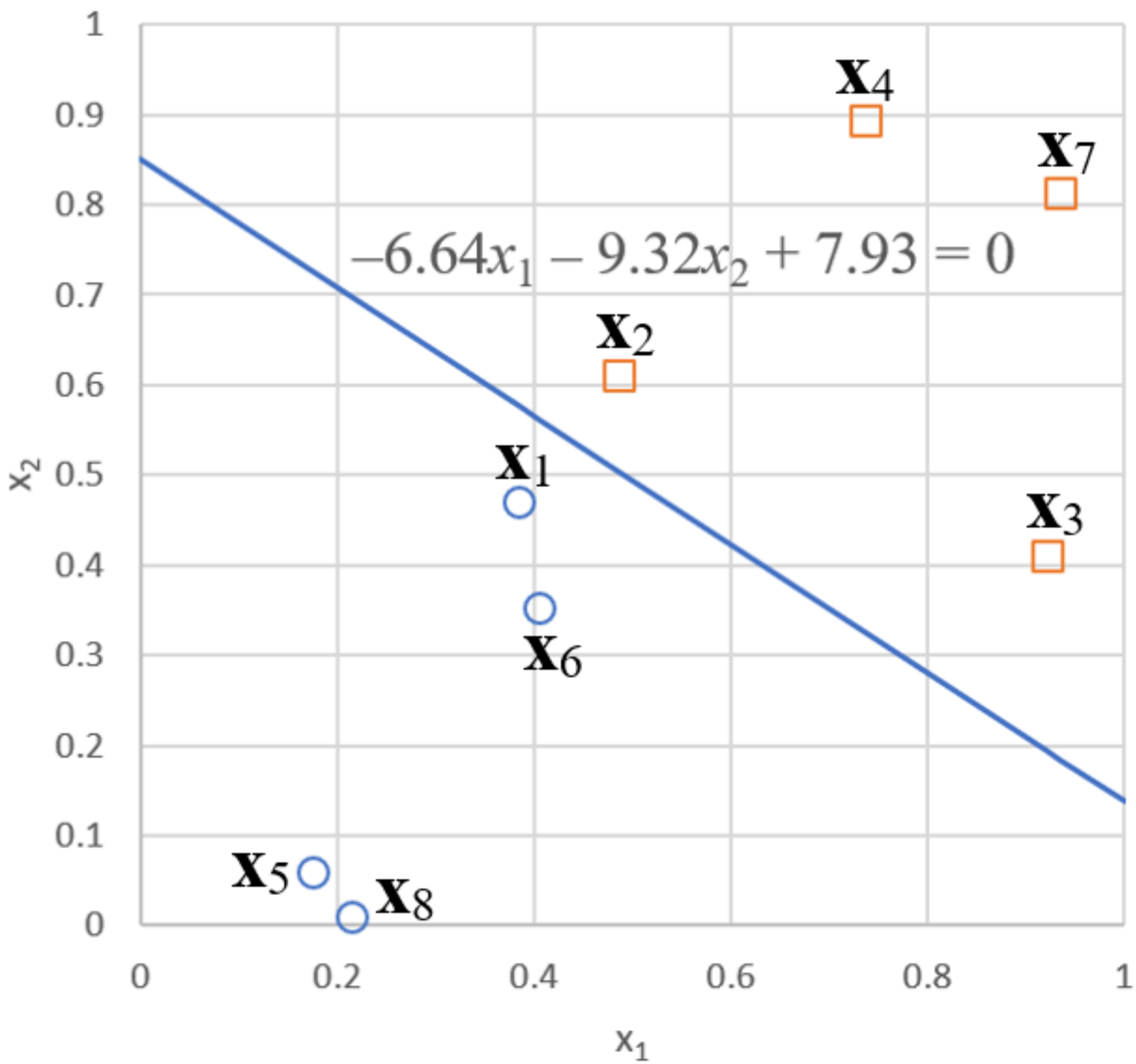
- Recall: in 2-dimensional space, we have $\mathbf{w} = (w_1, w_2)$, $\mathbf{x} = (x_1, x_2)$, DB: $w_1x_1 + w_2x_2 + b = 0$.

\- With $\mathbf{w} = (w_1, w_2) = (-6.64, -9.32)$, $b = 7.93$, we have DB: $-6.64x_1 - 9.32x_2 + 7.93 = 0$.

- DB corresponding to parameters $\mathbf{w}$ and b is shown in figure below.

Example

•



$-6.64x_1 - 9.32x_2 + 7.93 = 0$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
$\mathbf{x}_5$
$\mathbf{x}_6$
$\mathbf{x}_7$
$\mathbf{x}_8$
x_1
x_2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1

Example

• With found parameters **w** and b of DB, a test instance **z** is classified as follows:

$$f(\mathbf{z}) = sign(\mathbf{w} \cdot \mathbf{z} + b) = sign\left(\sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \cdot \mathbf{z} + b\right)$$

- If $f(\mathbf{z}) > 0$ (or $\mathbf{w} \cdot \mathbf{z} + b \gtrsim 1$), then **z** is classified as positive class (i.e., class label $y = 1$).

- If $f(\mathbf{z}) < 0$ (or $\mathbf{w} \cdot \mathbf{z} + b \lesssim -1$), then **z** is classified as negative class (i.e., class label $y = -1$).

Recall

-

Steps	**Hard-margin** SVM, DB: $\mathbf{w} \cdot \mathbf{x} + b = 0$ (5.28),
	b_{i1}: $\mathbf{w} \cdot \mathbf{x} + b = 1$ (5.32), b_{i2}: $\mathbf{w} \cdot \mathbf{x} + b = -1$ (5.33)
	maximize $d = 2 / \lVert\mathbf{w}\rVert$ (5.34), minimize $f(\mathbf{w}) = \lVert\mathbf{w}\rVert^2 / 2$ s.t. $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$, $i = 1, 2, \ldots, N$. (5.36)
1	minimize $L_P(\mathbf{w}, b, \lambda_i) = \frac{1}{2}\lVert\mathbf{w}\rVert^2 - \sum_{i=1}^{N} \lambda_i (y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1)$ (5.38)
2	$\partial L_P / \partial \mathbf{w} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$ (5.39) // $w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij}$ (5.50) $\partial L_P / \partial b = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$ (5.40)
3	Karush-Kuhn-Tucker (KKT) constraints $\lambda_i \geq 0$ (5.41), $\lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ (5.42) ($\mathbf{x}_i$'s with λ_i's > 0 are called **support vectors** (sv))
4	Transform $L_P(\mathbf{w}, b, \lambda_i)$ to $L_D(\lambda_i)$ $L_D(\lambda_i) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$ (5.43) s.t. $\lambda_i \geq 0$ (5.41), $\sum_{i=1}^{N} \lambda_i y_i = 0$ (5.40) (minimizing L_P is equivalent to maximizing L_D) Once λ_i's are found (e.g., use `cvxopt`), use $\lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ (5.42) to obtain $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$.

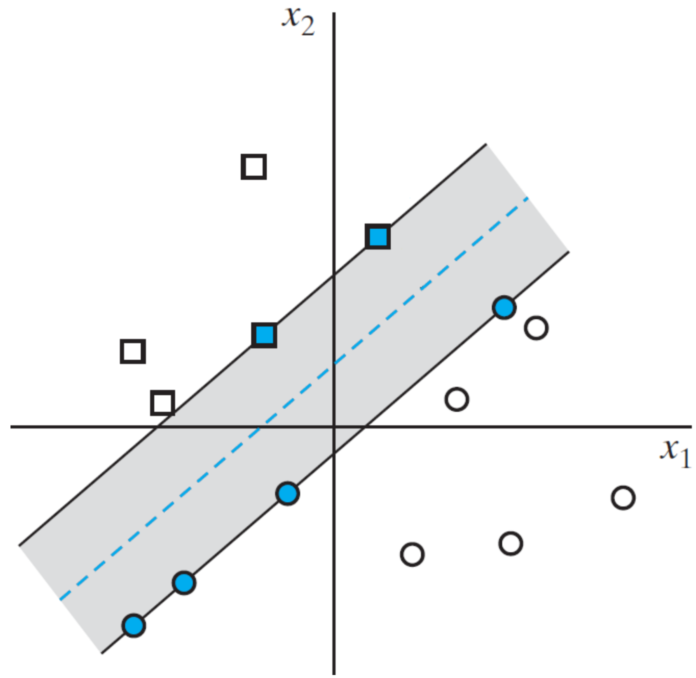
Contents

3. Linear SVM: Nonseparable Case

- Recall: **hard-margin** SVM uses inequality constraints $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$, $i = 1, 2, ..., N$.
- **Soft-margin** SVM uses inequality constraints $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$, $i = 1, 2, ..., N$; $\forall i$: $\xi_i \geq 0$.
- **Hard-margin** SVM is a special case of **soft-margin** SVM (i.e., $\xi_i = 0$ for $\forall i$)

3. Linear SVM: Nonseparable Case

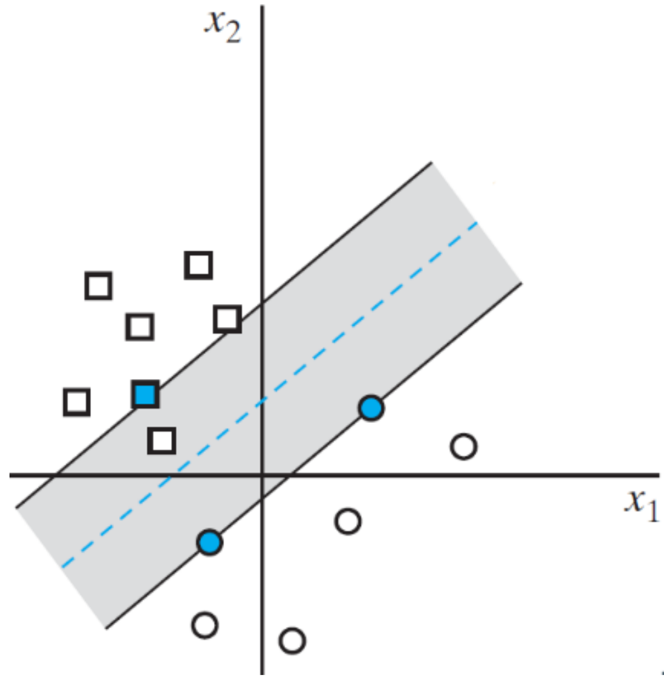
•



Hard-margin SVM: There is no data point $\mathbf{x}_i$ falling inside margin. [Simon Haykin]

3. Linear SVM: Nonseparable Case

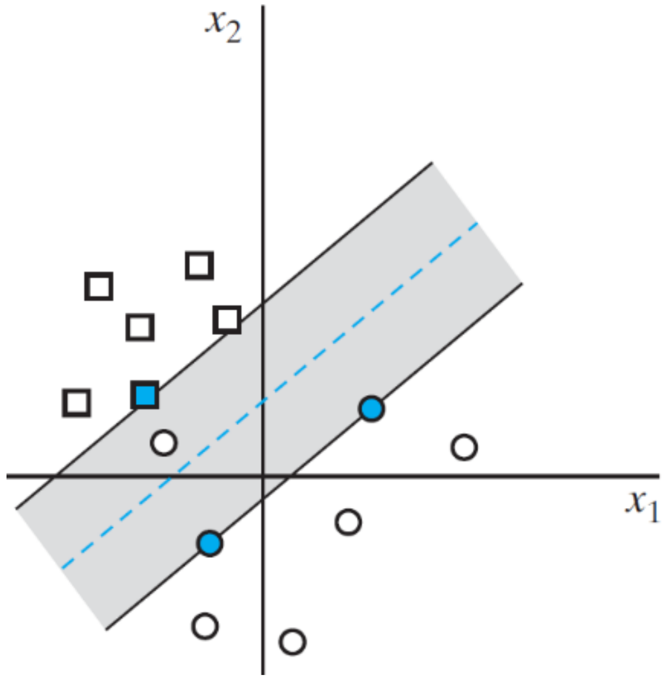
.



Soft-margin SVM: Data point $\mathbf{x}_i$ represented by a small square falls inside margin, but resides on the correct side of DB. ($0 < \lambda_i = C$ and $0 \leq \xi_i < 1$)

3. Linear SVM: Nonseparable Case

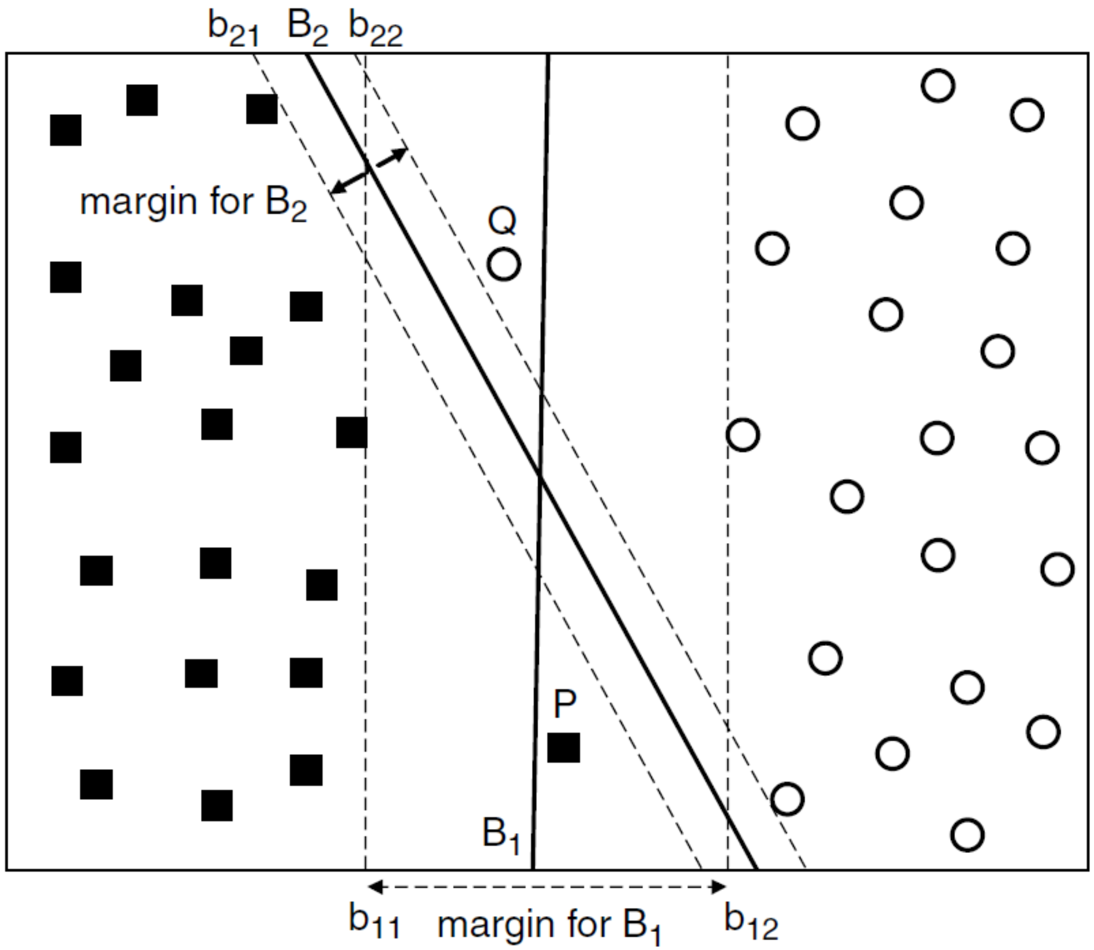
•



Soft-margin SVM: Data point $\mathbf{x}_i$ represented by a small circle falls on the wrong side of DB. ($0 < \lambda_i = C$ and $\xi_i \geq 1$)

3. Linear SVM: Nonseparable Case

-

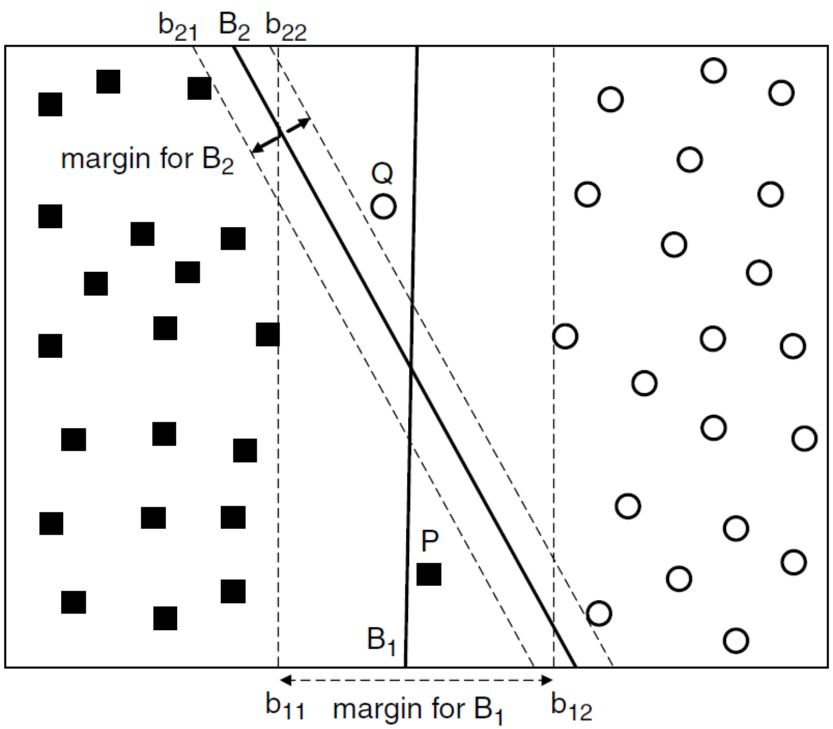


DB of SVM for nonlinearly separable data set

3. Linear SVM: Nonseparable Case

- **Slack variables** ξ_i's permit

- positive point Q lies on the negative side of DB.
- negative point P lies on the positive side of DB.



3. Linear SVM: Nonseparable Case

- Recall: primal **hard-margin** problem (i.e., original objective function)

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

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

3. Linear SVM: Nonseparable Case

- Primal **soft-margin** problem (i.e., modified objective function) is

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N} \xi_i$$

subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$,

$\xi_i \geq 0$, $i = 1, 2, \ldots, N$.

where constant $C > 0$ (a.k.a. cost) is user-specified parameter and determined based on the model's performance on the validation set (i.e., cross-validation).

3. Linear SVM: Nonseparable Case

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to } \; y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0,\ i = 1, 2, \ldots, N.$$

- Constrained minimization problem above is known as **convex** optimization problem, which can be solved using the standard **Lagrange multiplier** method.

3. Linear SVM: Nonseparable Case

- **Step 1**: The primary/primal Lagrangian for the constrained minimization problem can be written as

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

(i.e., $L_P(\mathbf{w}, b, \lambda_i, \xi_i, \mu_i)$)

3. Linear SVM: Nonseparable Case

where

- first two terms are objective function to be **minimized**,
- third term represents inequality constraints ($y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$, $i = 1, 2, ..., N$) associated with slack variables ξ_i, and
- last term is the result of the non-negativity requirements on the values of ξ_i's (i.e., $\forall i$: $\xi_i \geq 0$).

($\lambda_i \geq 0$ and $\mu_i \geq 0$ are KKT multipliers)

3. Linear SVM: Nonseparable Case

- **Step 2**: To minimize the Lagrangian L_P, we set the first-order derivatives (or gradient) of L_P with respect to $\mathbf{w}$, b, and ξ_i to zero, resulting in the following equations.

3. Linear SVM: Nonseparable Case

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{N} \lambda_i y_i x_{ij} = 0 \Rightarrow w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij} \quad (5.50)$$

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \quad (5.39)$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{N} \lambda_i y_i = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0 \quad (5.51)$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \lambda_i - \mu_i = 0 \Rightarrow \lambda_i + \mu_i = C \quad (5.52)$$

$$0 \le \lambda_i \le C.$$

3. Linear SVM: Nonseparable Case

- **Step 3**: The inequality constraints ($y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$) can be transformed into equality constraints using the following KKT (Karush-Kuhn-Tucker) conditions:

$$\lambda_i \geq 0,\ \xi_i \geq 0,\ \mu_i \geq 0, \ (5.47)$$

$$\lambda_i \left[y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i\right] = 0, \ (5.48)$$

$$\mu_i \xi_i = 0. \ (5.49)$$

3. Linear SVM: Nonseparable Case

- **Step 4**: Substituting equations $\partial L_P/\partial \mathbf{w} = 0$, $\partial L_P/\partial b = 0$, $\partial L_P/\partial \xi_i = 0$ into the primary/primal Lagrangian L_P will produce the following **dual** Lagrangian L_D of the maximization problem.

/* recall

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \ (5.39)$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0 \ (5.51)$$

*/

$$\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow C - \lambda_i - \mu_i = 0 \ (5.52)$$

3. Linear SVM: Nonseparable Case

- $$L_D(\lambda_i) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (5.53)$$

$$(0 \le \lambda_i \le C, \ \sum_{i=1}^{N} \lambda_i y_i = 0)$$

Recall:

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

3. Linear SVM: Nonseparable Case

$$\text{maximize } L_D(\lambda_i) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

$$\text{subject to } 0 \le \lambda_i \le C, \ \sum_{i=1}^{N} \lambda_i y_i = 0.$$

- The **dual** Lagrangian L_D for nonlinearly separable data is identical to the **dual** Lagrangian L_D for linearly separable data.

3. Linear SVM: Nonseparable Case

- Dual problem L_D can be solved numerically using quadratic programming techniques to obtain Lagrange multipliers λ_i (out of scope of the course).
- Once λ_i's are found (cvxopt), we can use equations $w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij}$ (5.50) (i.e., $\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$ (5.39)) and $\lambda_i\ [y_i\ (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i] = 0$ (5.48) to obtain feasible solutions for $\mathbf{w}$ and b (i.e., $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$, $\mathbf{x}_i$ is support vector).

3. Linear SVM: Nonseparable Case

- Specifically, for **support vectors** (i.e., $0 < \lambda_i < C$ and $\xi_i = 0$), we have

$\lambda_i\,[y_i\,(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] = 0$ (5.48)

$\rightarrow y_i\,(\mathbf{w}\cdot\mathbf{x}_i + b) = 1$

$\rightarrow \mathbf{w}\cdot\mathbf{x}_i + b = y_i\ (y_i = \pm 1 \rightarrow 1/y_i = y_i)$

$\rightarrow b = y_i - \mathbf{w}\cdot\mathbf{x}_i$

- The value computed for b may not be unique. Value of b depends on support vectors used in $\lambda_i\,[y_i\,(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i] = 0$.
- Average value for b is chosen.

3. Linear SVM: Nonseparable Case

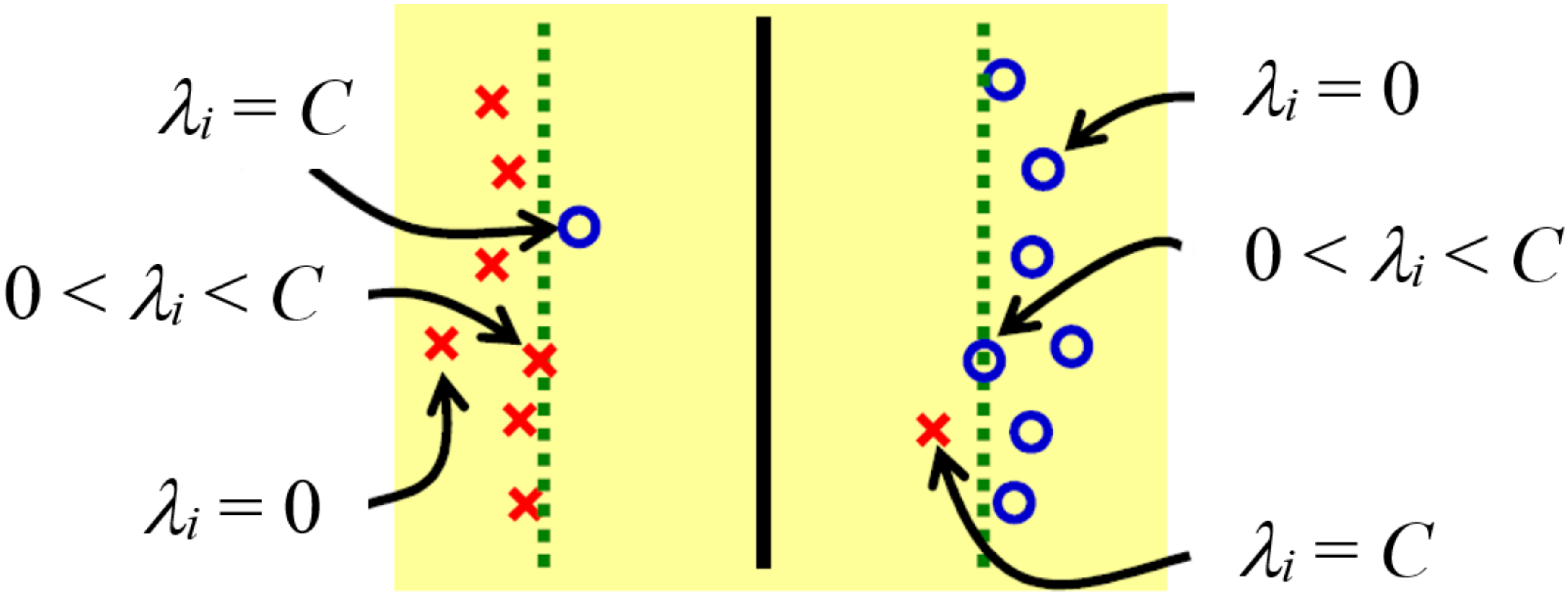


Figure 27.8 When $\lambda_i = 0$, $\mathbf{x}_i$ is inside the margin and correctly classified. When $0 < \lambda_i < C$, $\mathbf{x}_i$ is on the margin border (the dotted lines) and correctly classified. When $\lambda_i = C$, $\mathbf{x}_i$ is outside the margin, and if $\xi_i > 1$ and $m_i = (\mathbf{w} \cdot \mathbf{x}_i + b)y_i < 0$, then $\mathbf{x}_i$ is misclassified. [Masashi Sugiyama]

Summary

Steps	**Hard-margin** SVM, DB: $\mathbf{w} \cdot \mathbf{x} + b = 0$ (5.28), b_{i1}: $\mathbf{w} \cdot \mathbf{x} + b = 1$ (5.32), b_{i2}: $\mathbf{w} \cdot \mathbf{x} + b = -1$ (5.33)
	maximize $d = 2 / \lVert\mathbf{w}\rVert$ (5.34), minimize $f(\mathbf{w}) = \lVert\mathbf{w}\rVert^2 / 2$ s.t. $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$, $i = 1, 2, \ldots, N$. (5.36)
1	minimize $L_P(\mathbf{w}, b, \lambda_i) = \frac{1}{2}\lVert\mathbf{w}\rVert^2 - \sum_{i=1}^{N} \lambda_i (y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1)$ (5.38)
2	$\partial L_P / \partial \mathbf{w} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$ (5.39) // $w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij}$ (5.50) $\partial L_P / \partial b = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$ (5.40)
3	Karush-Kuhn-Tucker (KKT) constraints $\lambda_i \geq 0$ (5.41), $\lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ (5.42) ($\mathbf{x}_i$'s with λ_i's > 0 are called **support vectors** (sv))
4	Transform $L_P(\mathbf{w}, b, \lambda_i)$ to $L_D(\lambda_i)$ $L_D(\lambda_i) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$ (5.43) s.t. $\lambda_i \geq 0$ (5.41), $\sum_{i=1}^{N} \lambda_i y_i = 0$ (5.40) (minimizing L_P is equivalent to maximizing L_D) Once λ_i's are found (e.g., use `cvxopt`), use $\lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ (5.42) to obtain $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$.

Steps	**Soft-margin** SVM, DB: $\mathbf{w} \cdot \mathbf{x} + b = 0$ (5.28), b_{i1}: $\mathbf{w} \cdot \mathbf{x} + b = 1$ (5.32), b_{i2}: $\mathbf{w} \cdot \mathbf{x} + b = -1$ (5.33)
	minimize $f(\mathbf{w}) = \lVert\mathbf{w}\rVert^2 / 2 + C \sum_{i=1}^{N} \xi_i$ s.t. $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$, $i = 1, 2, \ldots, N$. (5.36′) (user-specified parameter $C > 0$ (cost))
1	minimize $L_P(\mathbf{w}, b, \lambda_i, \xi_i, \mu_i) = \frac{1}{2}\lVert\mathbf{w}\rVert^2 + C \sum_{i=1}^{N} \xi_i$ $- \sum_{i=1}^{N} \lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{N} \mu_i \xi_i$ (5.46)
2	$\partial L_P / \partial w_j = 0 \Rightarrow w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij}$ (5.50) // $\partial L_P / \partial \mathbf{w} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$ (5.39) $\partial L_P / \partial b = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$ (5.51) $\equiv$ (5.40) $\partial L_P / \partial \xi_i = C - \lambda_i - \mu_i = 0 \Rightarrow \lambda_i + \mu_i = C$ (5.52) ($0 \leq \lambda_i \leq C$)
3	Karush-Kuhn-Tucker (KKT) constraints $\lambda_i \geq 0$, $\xi_i \geq 0$, $\mu_i \geq 0$, (5.47) $\lambda_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i] = 0$, (5.48) $\mu_i \xi_i = 0$ (5.49). ($\mathbf{x}_i$'s with $0 < \lambda_i < C$ and $\xi_i = 0$ are called **support vectors** (sv))
4	Transform $L_P(\mathbf{w}, b, \lambda_i, \xi_i, \mu_i)$ to $L_D(\lambda_i)$ $L_D(\lambda_i) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$ (5.53) $\equiv$ (5.43) s.t. $0 \leq \lambda_i \leq C$, $\sum_{i=1}^{N} \lambda_i y_i = 0$ (5.51) $\equiv$ (5.40) (minimizing L_P is equivalent to maximizing L_D) Once λ_i's are found (e.g., use `cvxopt`), use $\lambda_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$ (5.42) to obtain $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ for $\mathbf{x}_i$'s with $0 < \lambda_i < C$ and $\xi_i = 0$ (i.e., for support vectors $\mathbf{x}_i$).

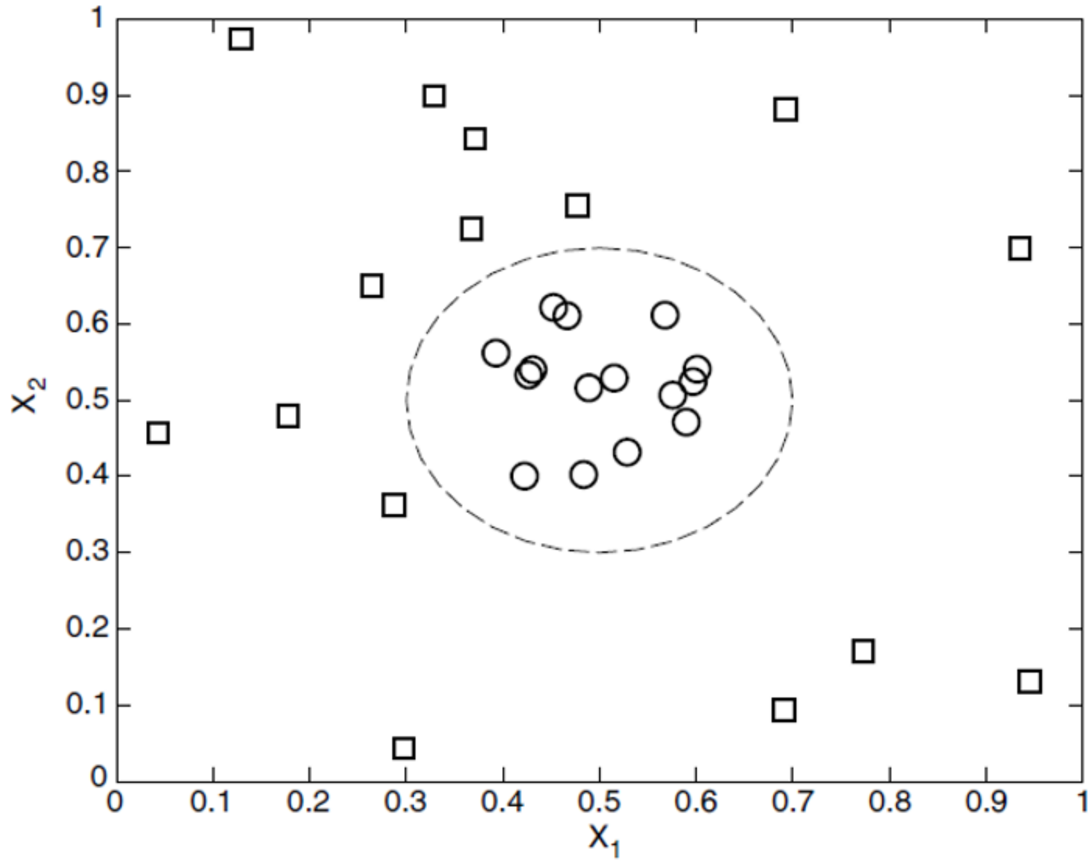
Contents

4. Nonlinear SVM

- **Hard-margin** SVM and **soft-margin** SVM have linear decision boundaries.
- Non-linear SVM has a nonlinear decision boundary (DB).

4. Nonlinear SVM

-



(a) nonlinear DB in original 2D space

$y_{\text{square}} = 1, y_{\text{circle}} = -1,$

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

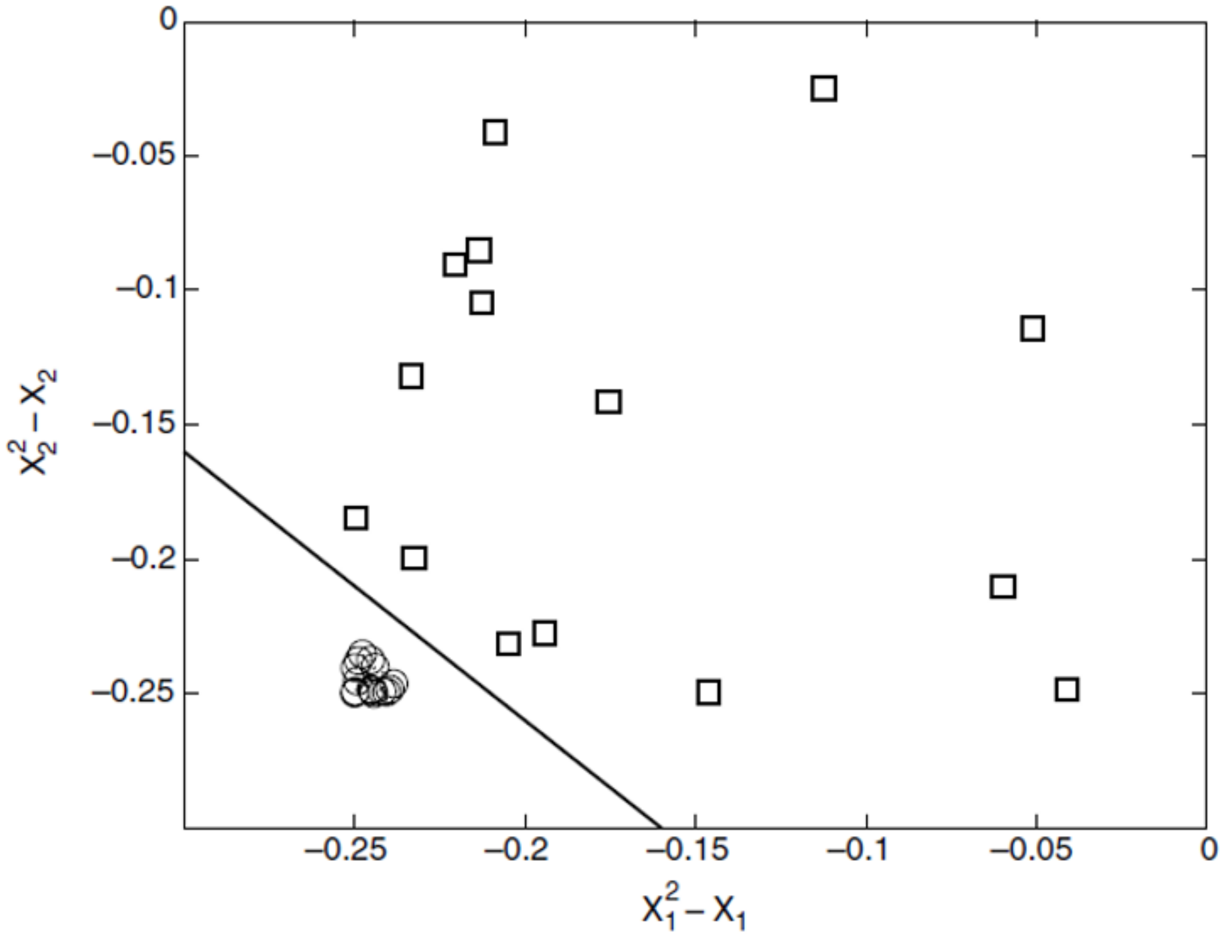
4. Nonlinear SVM

- In a non-linear SVM, the original input data is transformed into new space $\Phi(\mathbf{x})$ (or $\phi(\mathbf{x})$) so that a linear DB can be used to separate the instances in the transformed space.
- The mapping function Φ (or ϕ) is a nonlinear transformation needed to map the data $\mathbf{x}$ from its original feature space into a new space $\Phi(\mathbf{x})$ where the DB becomes linear. For example, we choose

Φ: $(x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$. (5.55)

4. Nonlinear SVM

•



(b) linear DB in transformed space

DB: $\mathbf{w} \cdot \Phi(\mathbf{x}) + b = 0$

4. Nonlinear SVM

- After doing the transformation, we can apply the methodology used for **hard-margin** SVM and **soft-margin** SVM to find a linear DB in the transformed space.
- The linear DB in the transformed space $\Phi(\mathbf{x})$ has the form: $\mathbf{w} \cdot \Phi(\mathbf{x}) + b = 0$. // or $\mathbf{w} \cdot \phi(\mathbf{x}) + b = 0$

Problems with Attribute Transformation Approach

- It is not clear what type of appropriate mapping function Φ should be used to ensure that a linear DB can be constructed in the transformed space $\Phi(\mathbf{x})$.
- Solving **the constrained optimization problem** (i.e., solve for $\mathbf{w}$ and b) in the high-dimensional feature space $\Phi(\mathbf{x})$ is a computationally expensive task.
- This approach suffers from **the curse of dimensionality problem**.

Problems with Attribute Transformation Approach

- Problems with attribute transformation approach can be solved by using the **kernel trick** method.

4. Nonlinear SVM

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

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

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

- The learning task of a nonlinear SVM is performed on the transformed attributes $\Phi(\mathbf{x})$.

Recall: Linear SVM

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

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

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

- The learning task of a linear SVM is performed on the original attributes $\mathbf{x}$.

4. Nonlinear SVM

- Following the approach used for linear SVM (i.e., **hard-margin** SVM and **soft-margin** SVM), we may derive the following **dual** Lagrangian L_D for the constrained optimization problem:

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

4. Nonlinear SVM

- Once the λ_i's are found using quadratic programming techniques, the parameters **w** and b can be derived using the following equations:

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \Phi(\mathbf{x}_i) \quad (5.57)$$

$$\lambda_i \left[y_i \left(\sum_{j=1}^{N} \lambda_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i) + b\right) - 1\right] = 0 \quad (5.58)$$

$$\rightarrow b = y_i - \sum_{j=1}^{N} \lambda_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i)$$

4. Nonlinear SVM

- Finally, a test instance $\mathbf{z}$ can be classified using the following equation:

$$\begin{aligned} f(\mathbf{z}) &= sign(\mathbf{w} \cdot \Phi(\mathbf{z}) + b) \\ &= sign\left(\sum_{i=1}^{N} \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z}) + b \right) \quad (5.59) \end{aligned}$$

- If $f(\mathbf{z}) > 0$, then $\mathbf{z}$ is classified as positive class (i.e., class label $y = 1$).

- If $f(\mathbf{z}) < 0$, then $\mathbf{z}$ is classified as negative class (i.e., class label $y = -1$).

4. Nonlinear SVM

• Calculating the dot product (i.e., similarity) between pairs of vectors in the transformed space, $\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$, can be quite cumbersome and may suffer from the curse of dimensionality problem.

• A breakthrough solution to this problem comes in the form of a method known as the **kernel trick**.

Kernel Trick

- The dot product $\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$ can also be regarded as a measure of similarity between two instances $\mathbf{x}_i$ and $\mathbf{x}_j$ in the transformed space.
- The **kernel trick** is a method for computing similarity in the transformed space $\Phi(\mathbf{x})$ using the original attribute set $\mathbf{x}$.
- Suppose we choose the following transformation (a.k.a. the **mapping function**):

$$\Phi:(x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1). \quad (5.55)$$

Kernel Trick

- The dot product between two input vectors **u** and **v** in the transformed space can be written as follows:

$$\Phi(\mathbf{u})\cdot\Phi(\mathbf{v}) = (u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, \sqrt{2}u_1u_2, 1)\cdot (v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, \sqrt{2}v_1v_2, 1)$$

$$= u_1^2v_1^2 + u_2^2v_2^2 + 2u_1v_1 + 2u_2v_2 + 2u_1u_2v_1v_2 + 1$$

$$= (\mathbf{u}\cdot\mathbf{v} + 1)^2. \quad (5.60)$$

Kernel Trick

- Thus, the dot product in the transformed space $\Phi(\mathbf{x})$ can be expressed in terms of a **similarity function** K in the original space $\mathbf{x}$:

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

// polynomial kernel function

- The similarity function K, which is computed in the original attribute space $\mathbf{x}$, is known as the **kernel function**.

Kernel Trick

- Thus, the **kernel trick** method can overcome the problems with the attribute transformation approach.

- First, we do not have to know the exact form of the mapping function Φ because the kernel functions used in nonlinear SVM must satisfy a mathematical principle known as **Mercer's theorem**.

Kernel Trick

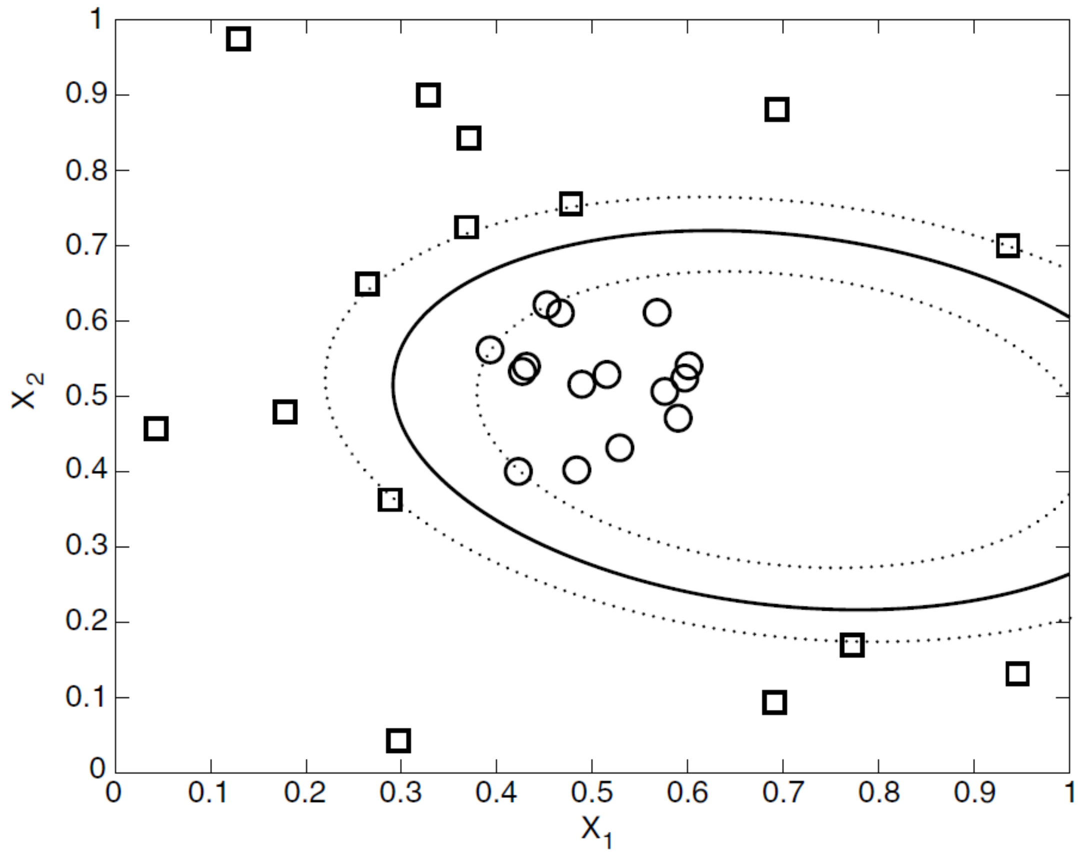
• The principle of **Mercer's theorem** ensures that a **kernel function** $K(\mathbf{u}, \mathbf{v})$ can always be expressed as the dot product $\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$ between two input vectors in some high-dimensional space.

- Second, computing the dot products $\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v})$ (e.g., solve for $\mathbf{w}$ and b) using a **kernel function** is considerably cheaper than using the transformed attribute set $\Phi(\mathbf{x})$ because $\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^2$ (5.61).

Kernel Trick

- Third, since the computations are performed in the original space **x**, issues associated with the curse of dimensionality problem can be avoided.

Kernel Trick



DB produced by a nonlinear SVM with polynomial kernel function

Kernel Trick

- A test instance $\mathbf{x}$ is classified as follows:

$$f(\mathbf{z}) = sign(\mathbf{w} \cdot \Phi(\mathbf{z}) + b) = sign(\sum_{i=1}^{N} \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z}) + b)$$

$$= sign(\sum_{i=1}^{N} \lambda_i y_i K(\mathbf{x}_i, \mathbf{z}) + b)$$

$$= sign(\sum_{i=1}^{N} \lambda_i y_i (\mathbf{x}_i \cdot \mathbf{z} + 1)^2 + b), \qquad (5.62)$$

where $b = y_i - \sum_{j=1}^{N} \lambda_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i)$ (5.58)

i.e., $b = y_i - \sum_{j=1}^{N} \lambda_j y_j (\mathbf{x}_j \cdot \mathbf{x}_i + 1)^2$

Kernel Trick

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

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

if and only if, for any function $g(\mathbf{x})$ *such that* $\int g(\mathbf{x})^2 d\mathbf{x}$ *is finite, then* $\int K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0$.

Some Kernel Functions

1. **Polynomial kernel of degree *p***:

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

2. **Gaussian radial basis function (RBF) kernel**:

$$K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/(2\sigma^2)} \quad (6.54)$$

3. **Sigmoid kernel**:

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

for some $k > 0$, $\delta > 0$

4. Nonlinear SVM

- A nonlinear SVM with a Gaussian radial basis function (RBF) kernel gives the same decision hyperplane as a type of neural network (NN) known as a radial basis function (RBF) network.
- A nonlinear SVM with a sigmoid kernel is equivalent to a simple three-layer NN known as a multilayer perceptron (i.e., backpropagation NN).

4. Nonlinear SVM

- SVM training always finds a global solution, unlike NNs, such as backpropagation NN, where many local minima usually exist.
- A major research goal regarding SVMs is to improve the speed in training and testing so that SVMs may become a more feasible option for very large data sets.

4. Nonlinear SVM

- Other issues with a nonlinear SVM include determining the best kernel for a given data set and finding an efficient method for multiclass classification.

Summary

-

Exercises

1. Consider the linearly separable dataset D in a two-dimensional space, as shown in the table below, which contains eight training instances $\mathbf{x}_i$, class labels $y_i \in \{-1, 1\}$, and Lagrange multipliers λ_i for $i = 1, 2, ..., 8$.

Instances	x_1	x_2	y_i	λ_i
$\mathbf{x}_1$	2	2.5	1	2.7027
$\mathbf{x}_2$	2.5	3.2	–1	2.7027
$\mathbf{x}_3$	4	2.5	–1	0
$\mathbf{x}_4$	3.5	4	–1	0
$\mathbf{x}_5$	1	2	1	0
$\mathbf{x}_6$	2.2	1.5	1	0
$\mathbf{x}_7$	4.5	3.3	–1	0
$\mathbf{x}_8$	1.5	0.5	1	0

Exercises

Specify support vectors from the given data set D and determine a decision boundary of a hard-margin linear SVM (support vector machine). You need to show how to compute the parameters $\mathbf{w}$ and b of the decision boundary (DB). Describe how to use the trained hard-margin linear SVM to classify a test instance $\mathbf{z}$.

Exercises

2. Consider the linearly separable dataset D in a two-dimensional space, as shown in the table below, which contains nine training instances $\mathbf{x}_i$, class labels $y_i \in \{-1, 1\}$, and Lagrange multipliers λ_i for $i = 1, 2, ..., 8$. Compute $\mathbf{w}$ and b of DB.

Instances	x_1	x_2	y_i	λ_i
$\mathbf{x}_1$	0.1193	0.3913	+1	0
$\mathbf{x}_2$	–0.0080	0.1209	+1	49.6257
$\mathbf{x}_3$	0.1671	0.2101	+1	0
$\mathbf{x}_4$	0.3408	0.3518	+1	0
$\mathbf{x}_5$	–0.1479	0.1639	+1	0.0005
$\mathbf{x}_6$	–0.2042	–0.3964	–1	0
$\mathbf{x}_7$	–0.2732	–0.0832	–1	0
$\mathbf{x}_8$	–0.0663	–0.0712	–1	49.6262
$\mathbf{x}_9$	0.0875	–0.1819	–1	0

Exercises

3. Consider the linearly separable dataset D in a two-dimensional space, as shown in the table below, which contains nine training instances $\mathbf{x}_i$, class labels $y_i \in \{-1, 1\}$, and Lagrange multipliers λ_i for $i = 1, 2, ..., 8$. Compute $\mathbf{w}$ and b of DB.

Instances	x_1	x_2	y_i	λ_i
$\mathbf{x}_1$	–1	–4	–1	0
$\mathbf{x}_2$	–3	–3	–1	0
$\mathbf{x}_3$	–0.6	–1.3	–1	0.0533
$\mathbf{x}_4$	–3	–1	–1	0
$\mathbf{x}_5$	–2	0.5	–1	0.1316
$\mathbf{x}_6$	3	3	+1	0
$\mathbf{x}_7$	2	1	+1	0
$\mathbf{x}_8$	1	2	+1	0.1849
$\mathbf{x}_9$	1	3	+1	0

Exercises

4. Consider the linearly separable dataset D in a two-dimensional space, as shown in the table below, which contains eight training instances $\mathbf{x}_i$, class labels $y_i \in \{-1, 1\}$, and Lagrange multipliers λ_i for $i = 1, 2, ..., 8$. Compute $\mathbf{w}$ and b of DB.

Instances	x_1	x_2	y_i	λ_i
$\mathbf{x}_1$	2	2.5	1	2.6847
$\mathbf{x}_2$	4	2.5	–1	0
$\mathbf{x}_3$	2.5	3.2	–1	2.7029
$\mathbf{x}_4$	3.5	4	–1	0
$\mathbf{x}_5$	1	2	1	0
$\mathbf{x}_6$	3	1.8	1	0.0182
$\mathbf{x}_7$	4.5	3.3	–1	0
$\mathbf{x}_8$	2	1	1	0

References

1. Jiawei Han, Micheline Kamber, Jian Pei. 2011. *Data Mining Concepts and Techniques*. 3rd Ed. Morgan Kaufmann. ISBN: 9380931913.

2. Pang-Ning Tan, Michael Steinbach, Vipin Kumar. 2005. *Introduction to Data Mining*. 1st Ed. Pearson. ISBN: 0321321367.

3. Charu C. Aggarwal. 2015. *Data Mining The Textbook*. Springer. ISBN: 3319141414.

References

4. Nong Ye. 2013. *Data Mining: Theories, Algorithms, and Examples*. CRC Press. ISBN: 1439808384.

5. Rodrigo Fernandes de Mello, Moacir Antonelli Ponti. 2018. *Machine Learning A Practical Approach on the Statistical Learning Theory*. Springer. ISBN: 9783319949888.

6. Dan Simovici. 2018. *Mathematical Analysis for Machine Learning and Data Mining*. World Scientific. ISBN: 9813229683.

Extra Slides

- Euclidean norm (magnitude) of $\mathbf{w} = (w_1, w_2, \ldots, w_n)$

$$\|\mathbf{w}\| = \sqrt{\mathbf{w} \cdot \mathbf{w}} = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$$

- Given two vectors $\mathbf{x} = (x_1, x_2)$, $\mathbf{y} = (y_1, y_2)$, θ is the angle between $\mathbf{x}$ and $\mathbf{y}$.

\- The algebraic definition of the dot product:

$$\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2.$$

\- The geometric definition of the dot product:

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \, \|\mathbf{y}\| \cos(\theta).$$

\- $\cos(\theta)$ = adjacent / hypotenuse

Extra Slides

• λ_i can be found by using Python package cvxopt (Martin S. Andersen: http://people.compute.dtu.dk/~mskan)

- Install: conda install -c conda-forge cvxopt
- To use: import cvxopt
- Use provided methods/functions: cvxopt.matrix(), solution = cvxopt.solvers.qp(), and so on.
- Form solution, we obtain $\lambda_i > 0$ (i.e., support vectors), weight vector **w**, bias b.