AI6102: Machine Learning Methodologies & Applications

L5: Kernel Methods

ZHANG Hanwang

hanwangzhang@ntu.edu.sg

Nanyang Technological University, Singapore

Homepage: https://mreallab.github.io/

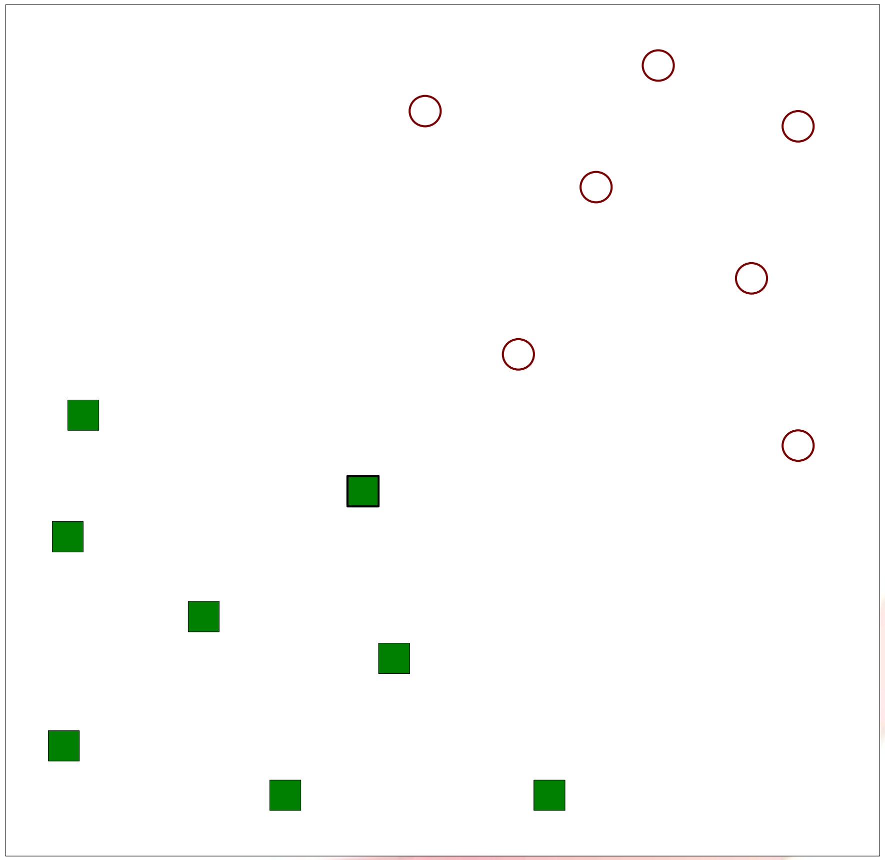
Support Vector Machines

- To learn a binary classifier

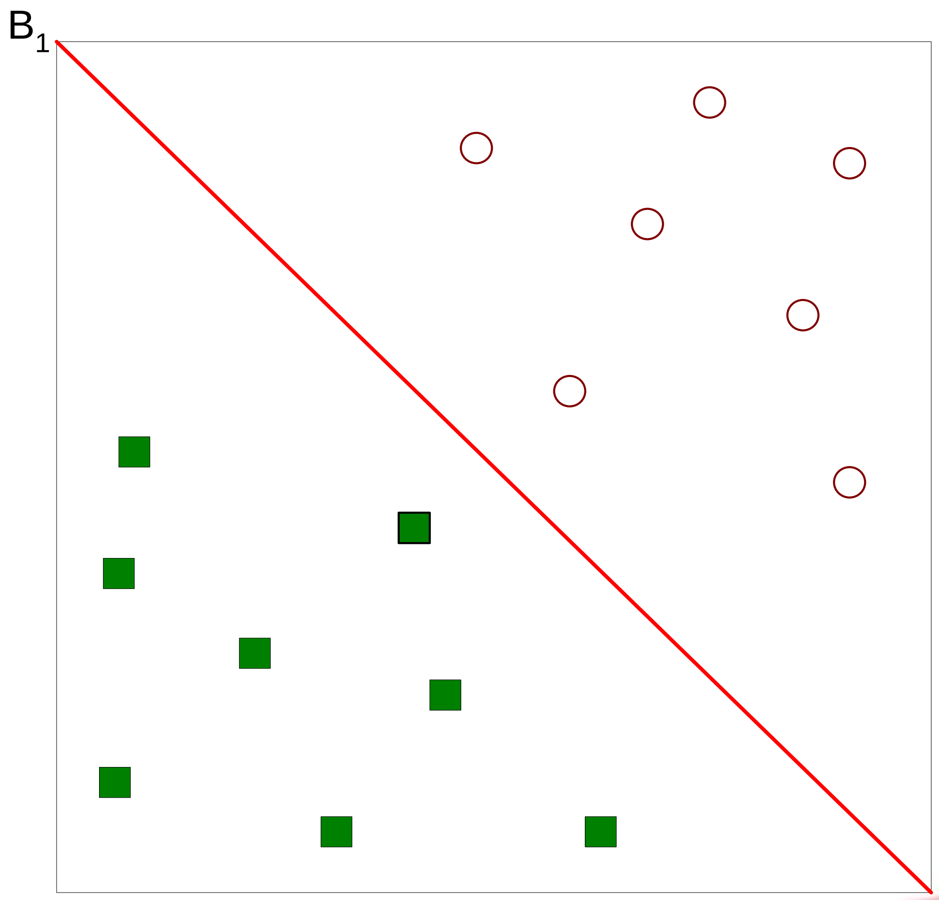
$\Downarrow$

- To find a hyperplane (also known as decision boundary) such that all the squares reside on one side of the hyperplane and all the circles reside on the other side

Assumption: data instances from the binary classes are separable

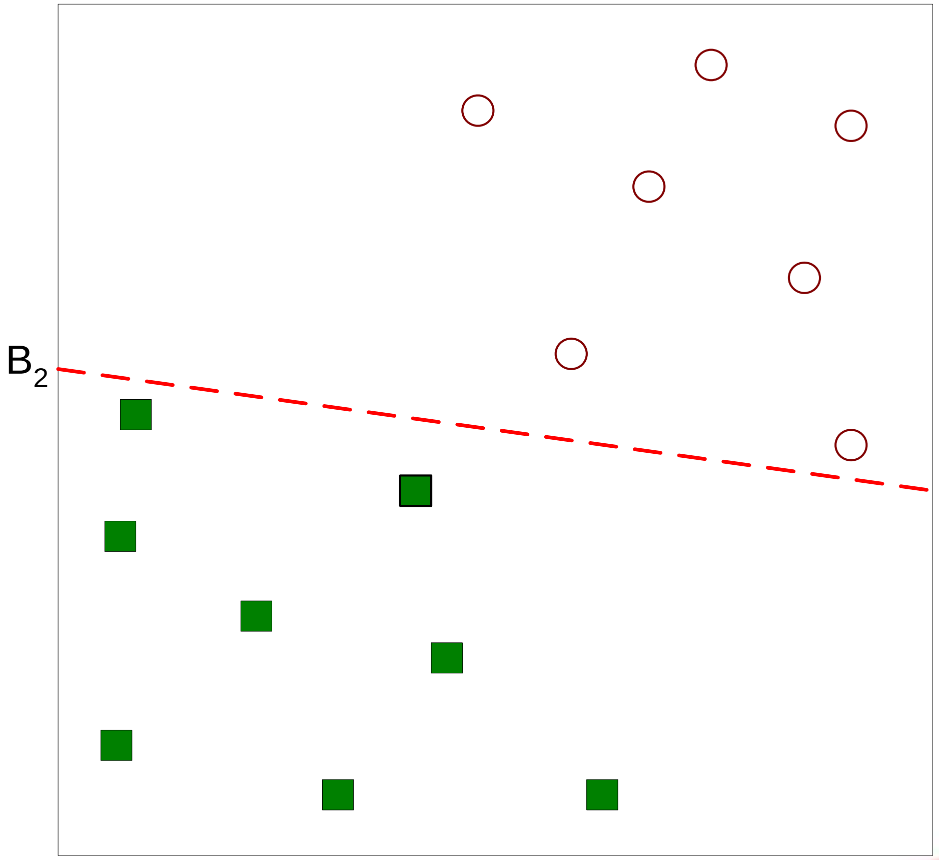


A Possible Decision Boundary



One Possible Solution

Another Possible Decision Boundary

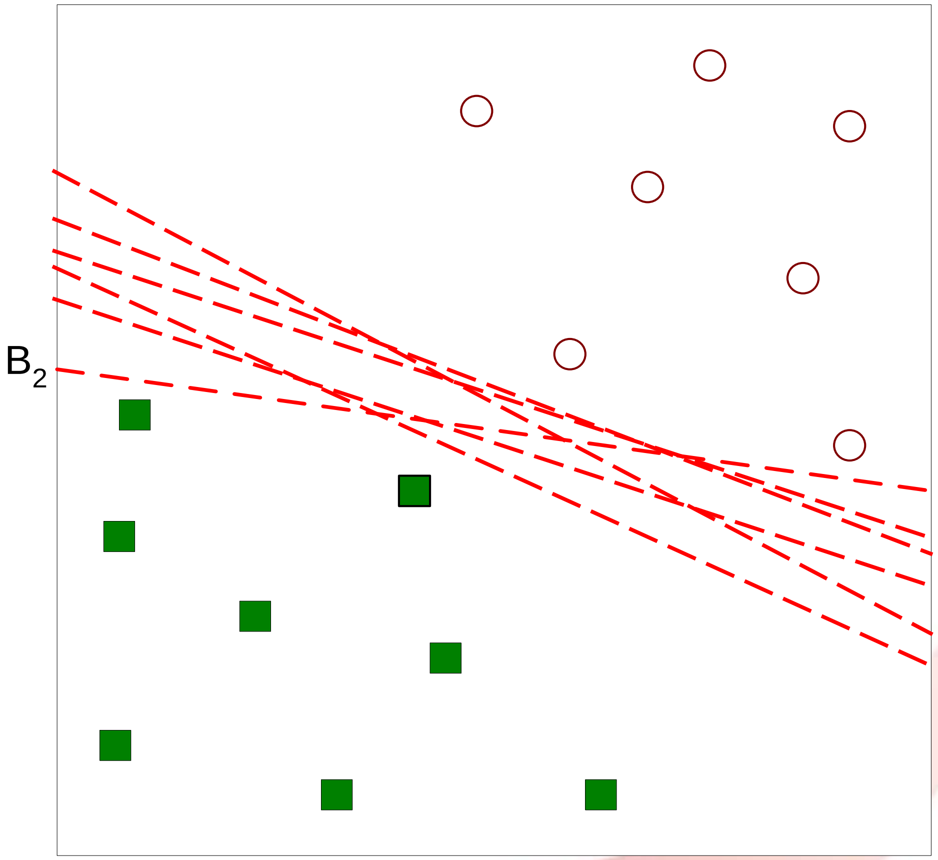


Another possible solution

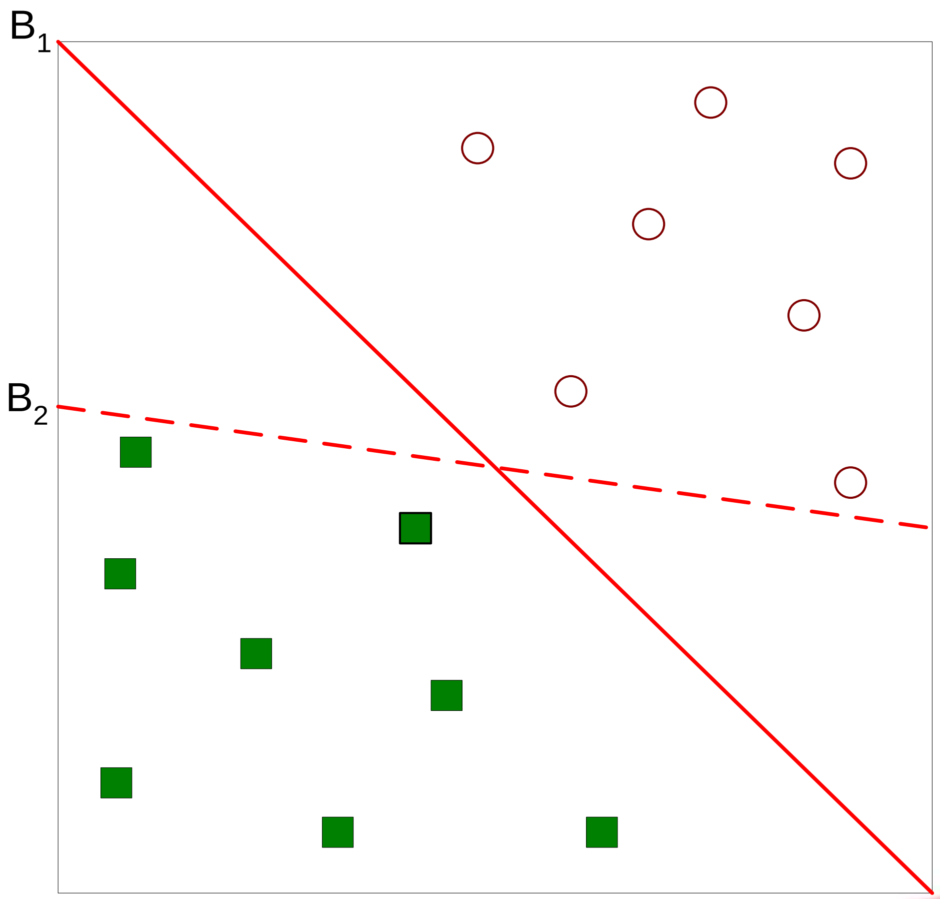
Many Possible Decision Boundary

- Though all the shown decision boundaries can separate training examples perfectly, their test errors may be different

- Which one should be used to construct the classifier?

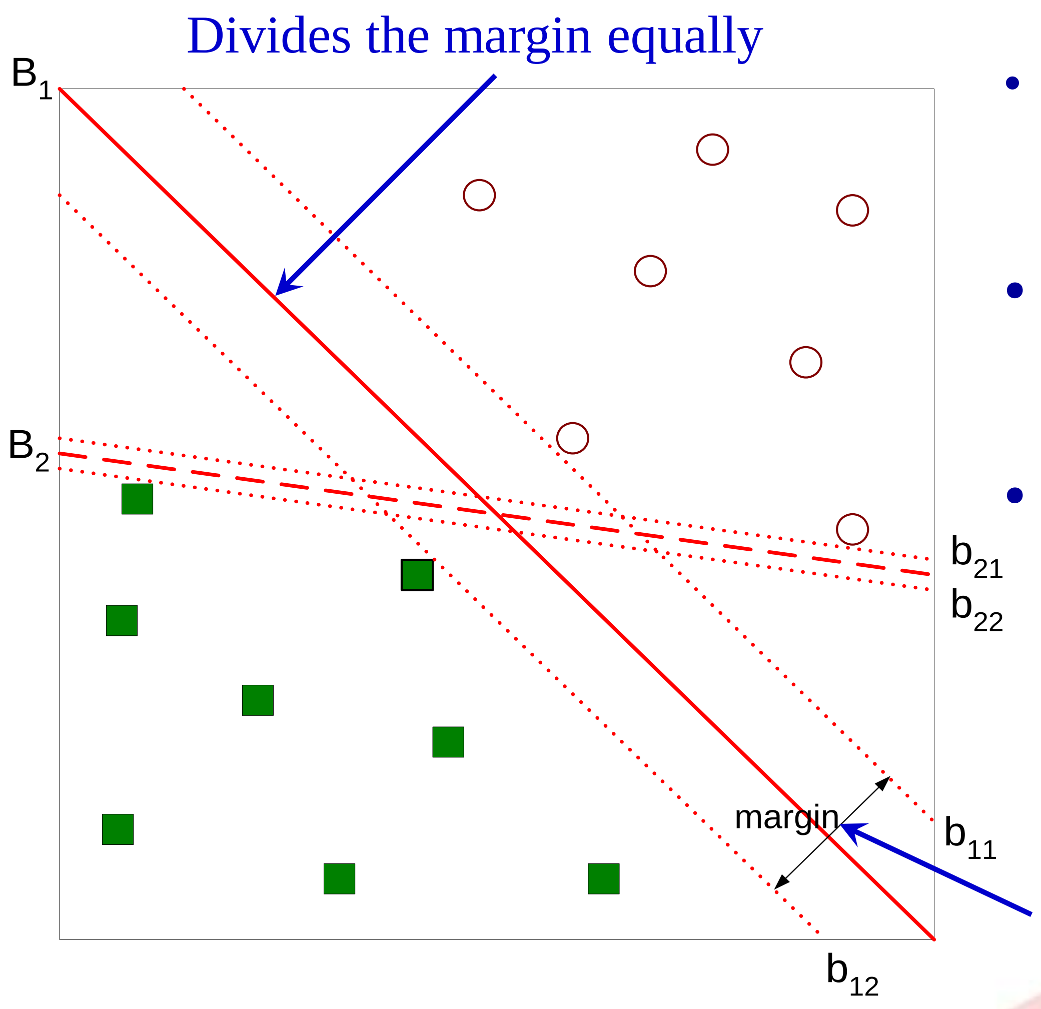


Decision Boundaries Comparison



Which one is better? B1 or B2?

Margin of Decision Boundary



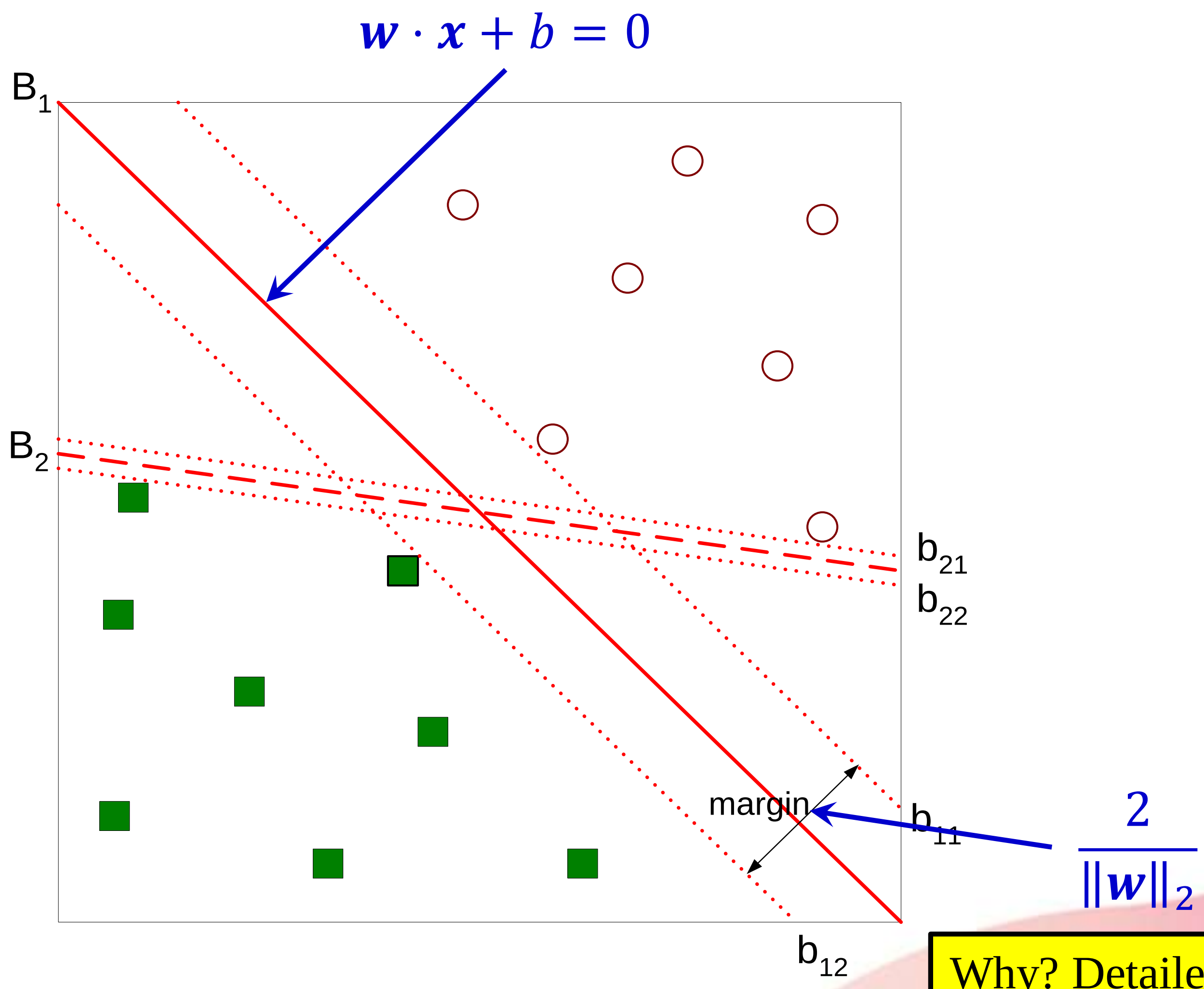
- Each decision boundary B_i is associated with a pair of parallel hyperplanes: b_{i1} and b_{i2}
- b_{i1} is obtained by moving the hyperplane until it touches the closest circle(s)
- b_{i2} is obtained by moving a hyperplane away from the decision boundary until it touches the closest square(s)

The distance between the parallel hyperplanes is known as the margin of the decision boundary

Support Vector Machines

- Support Vector Machines (SVMs) aim to learn a linear decision boundary whose margin is largest over the training data instances
- SVMs are one of the most classical machine learning methods
- In the past (in 90’s and 00’s), SVMs have shown promising empirical results in many practical applications, such as computer vision, sensor networks and text mining

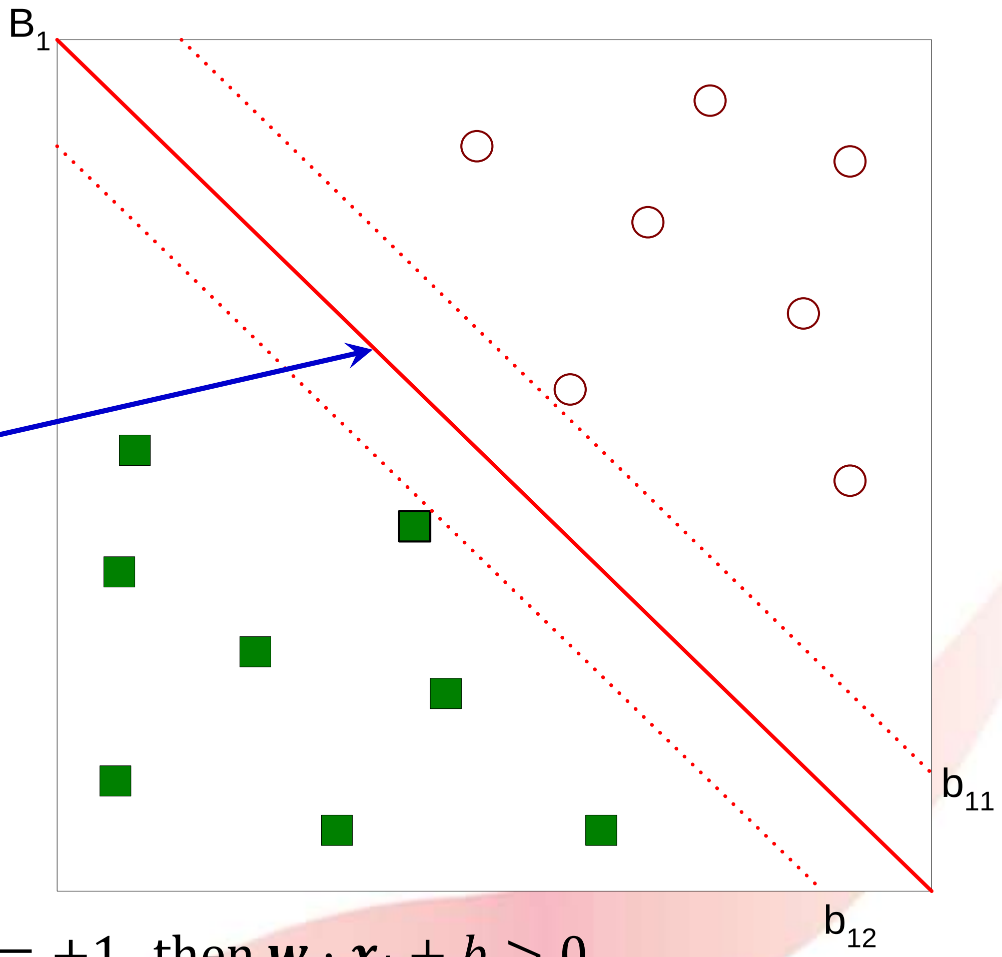
How to Represent A Margin?


$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
B_1
B_2
b_{21}
b_{22}
margin
b_{11}
b_{12}
$\frac{2}{\|\boldsymbol{w}\|_2}$

Why? Detailed induction will be presented in the following slides

2-Dimensional Case

- Given a binary classification task, denote $y_i = +1$ the circle class, and $y_i = -1$ the square class



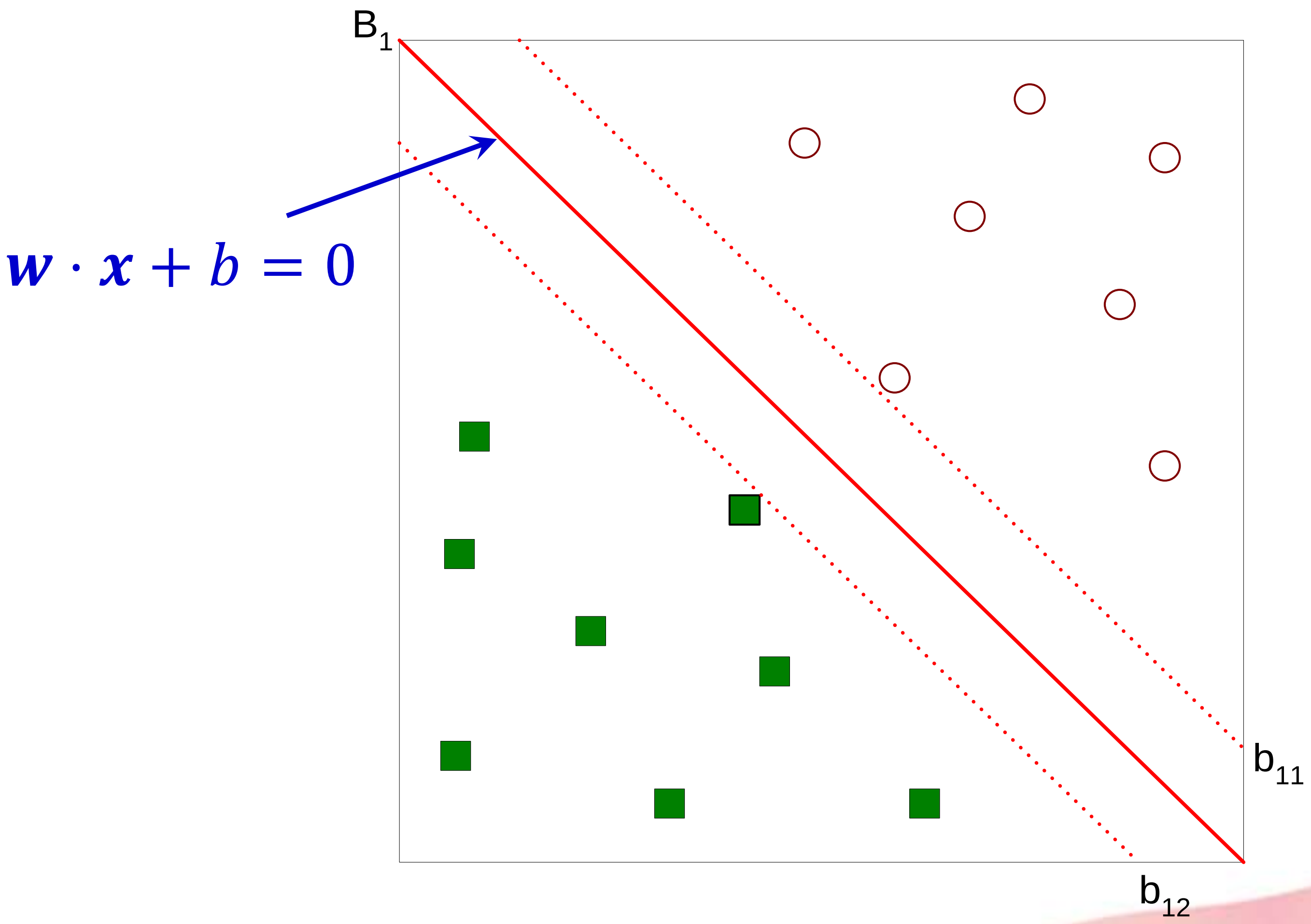
Decision boundary:

$$w_1 x_1 + w_2 x_2 + b = 0$$

General form: $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$

To learn a decision boundary such that $\begin{cases} \text{if } y_i = +1, \text{ then } \boldsymbol{w} \cdot \boldsymbol{x}_i + b > 0 \\ \text{if } y_i = -1, \text{ then } \boldsymbol{w} \cdot \boldsymbol{x}_i + b < 0 \end{cases}$

Making Decision



For any test example $\boldsymbol{x}^*$: $\begin{cases} f(\boldsymbol{x}^*) = +1, \text{ if } \boldsymbol{w} \cdot \boldsymbol{x}^* + b \geq 0 \\ f(\boldsymbol{x}^*) = -1, \text{ if } \boldsymbol{w} \cdot \boldsymbol{x}^* + b < 0 \end{cases}$

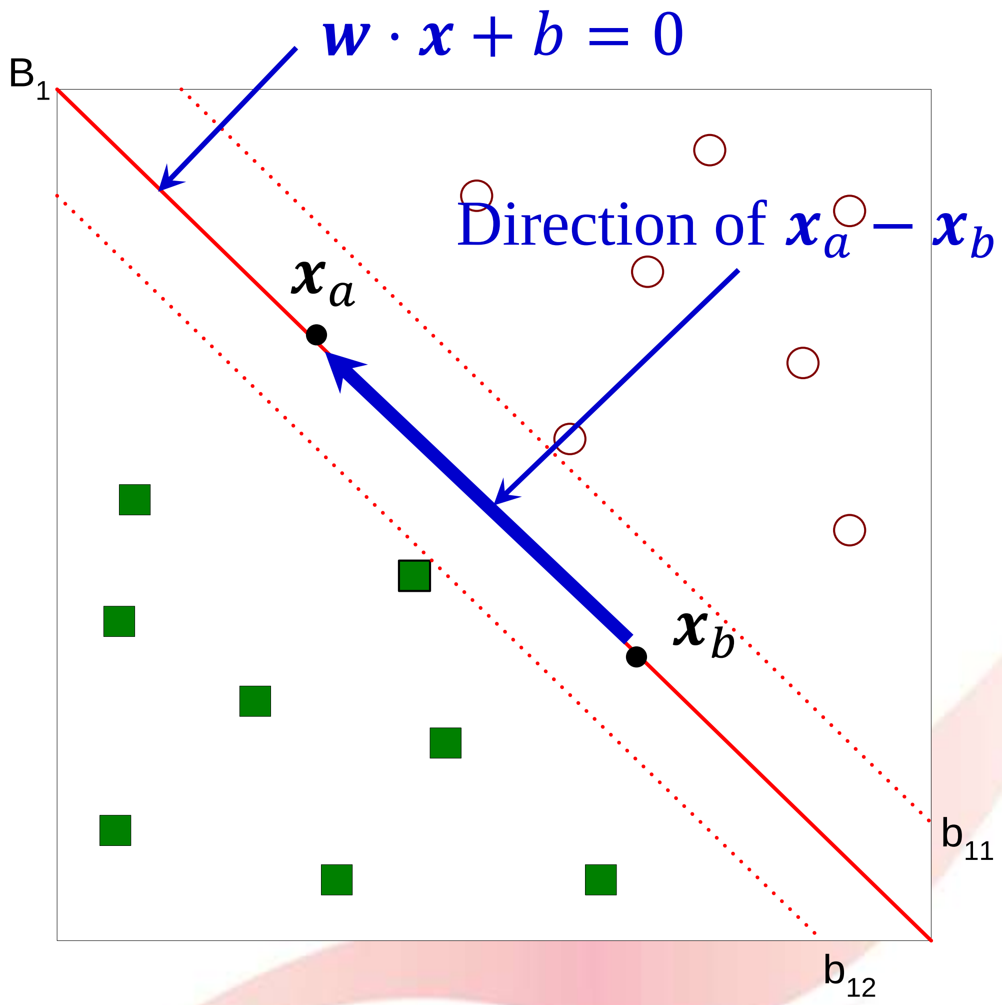
Margin – Induction

- Suppose $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are two points located on the decision boundary,

$$\begin{cases} \boldsymbol{w} \cdot \boldsymbol{x}_a + b = 0 \\ \boldsymbol{w} \cdot \boldsymbol{x}_b + b = 0 \end{cases}$$

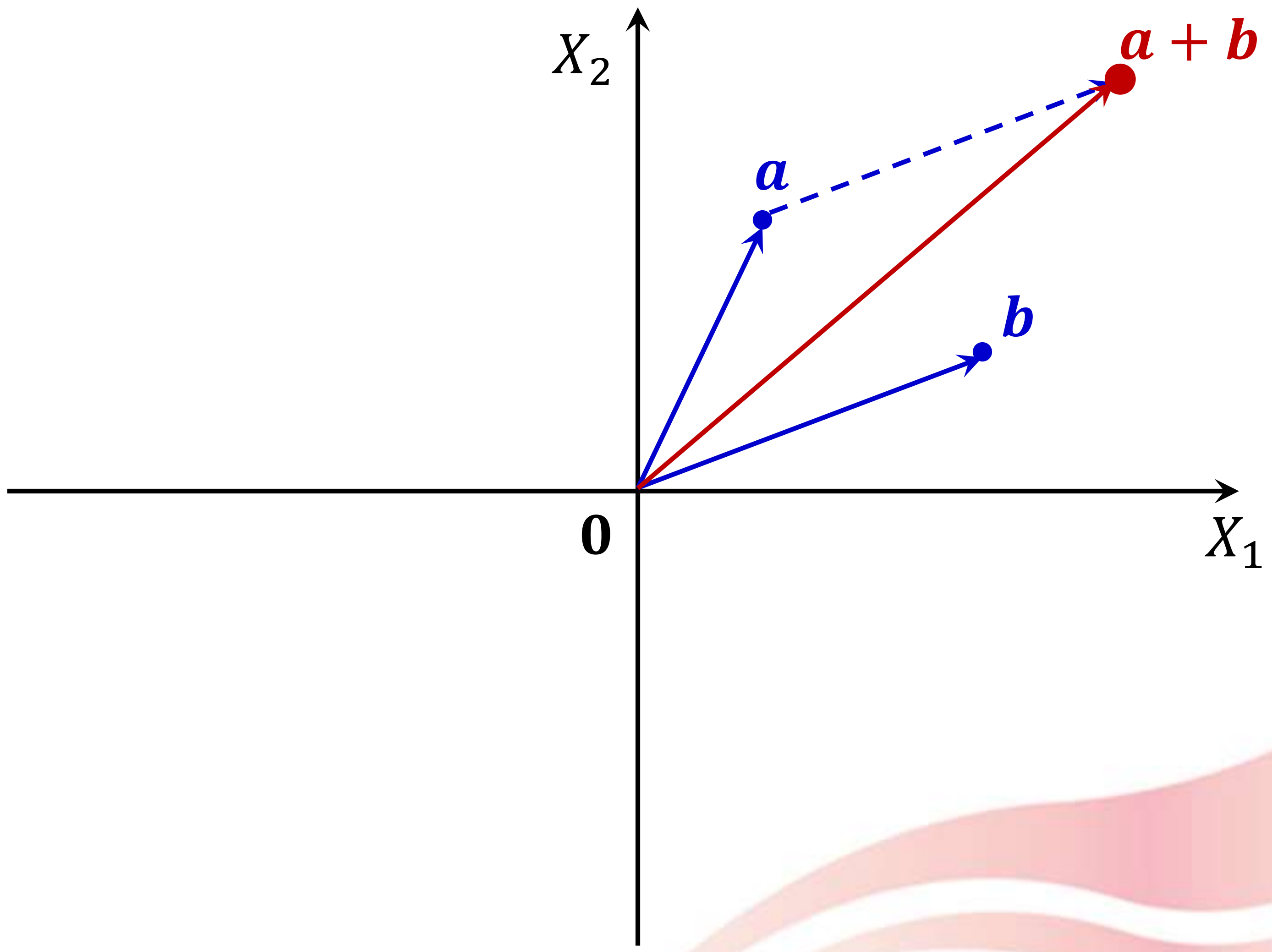
$$\Downarrow$$

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



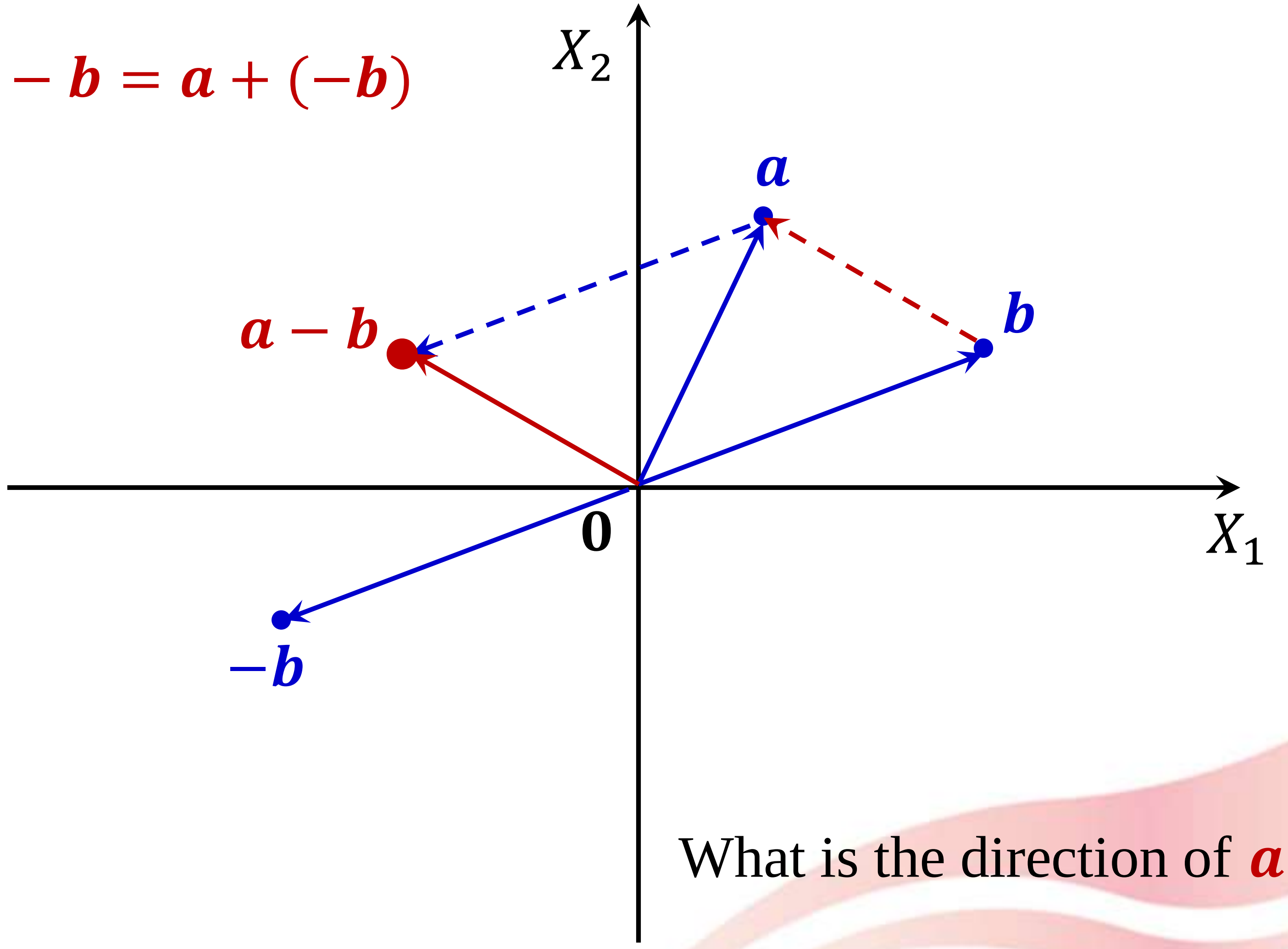
$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
B_1
Direction of $\boldsymbol{x}_a - \boldsymbol{x}_b$
$\boldsymbol{x}_a$
$\boldsymbol{x}_b$
b_{11}
b_{12}

Direction of Vectors – Review



Direction of Vectors – Review (cont.)

$\boldsymbol{a} - \boldsymbol{b} = \boldsymbol{a} + (-\boldsymbol{b})$



What is the direction of $\boldsymbol{a} - \boldsymbol{b}$?

Margin – Induction (cont.)

- Suppose $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are two points located on the decision boundary,

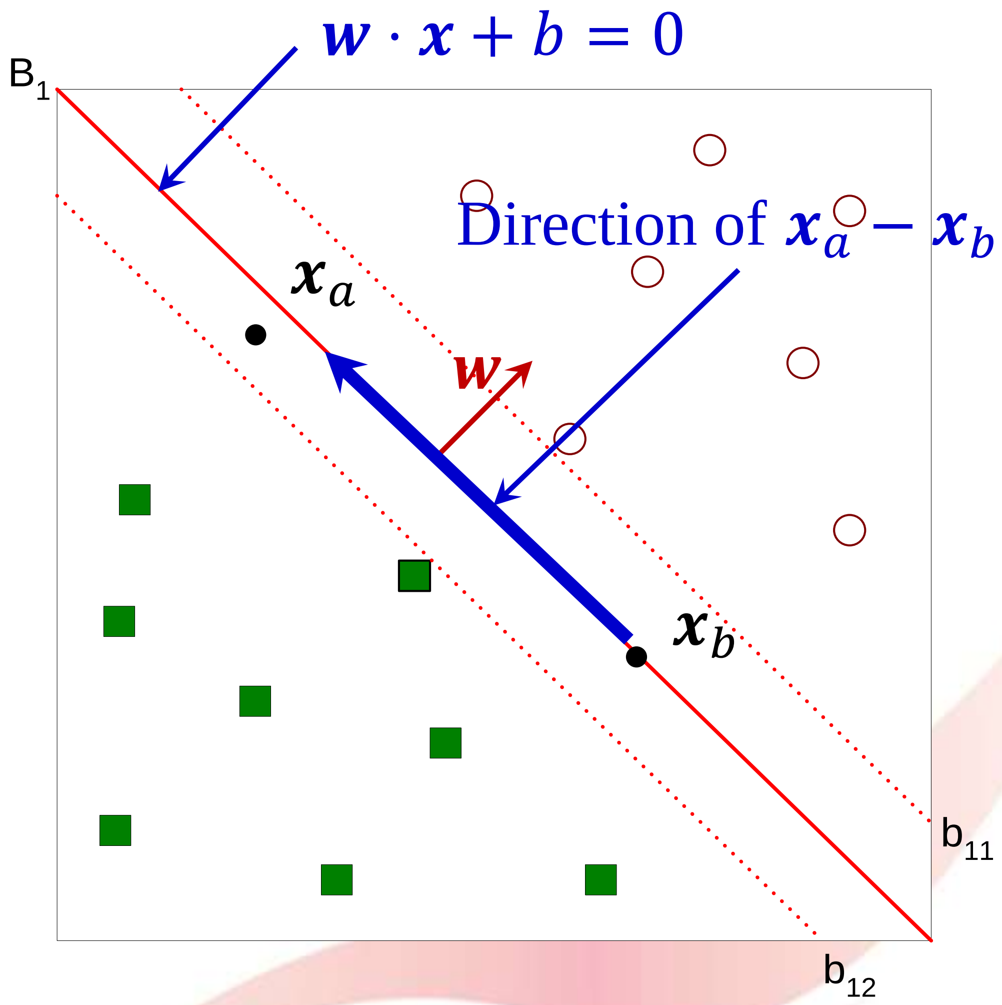
$$\begin{cases} \boldsymbol{w}\cdot\boldsymbol{x}_a + b = 0 \\ \boldsymbol{w}\cdot\boldsymbol{x}_b + b = 0 \end{cases}$$

$$\Downarrow$$

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

Based on definition of inner product

The direction of $\boldsymbol{w}$ is orthogonal to the decision boundary



$\boldsymbol{w}\cdot\boldsymbol{x} + b = 0$
B1
Direction of $\boldsymbol{x}_a - \boldsymbol{x}_b$
$\boldsymbol{x}_a$
$\boldsymbol{w}$
$\boldsymbol{x}_b$
b11
b12

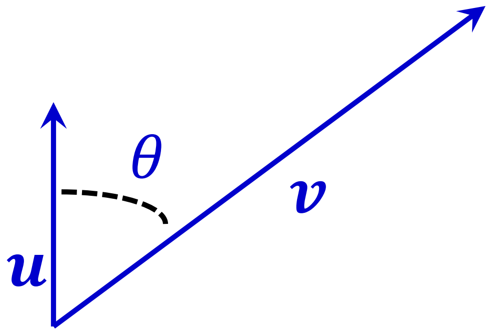
Inner Product – Review

- Given two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ of m dimensions, the inner (or dot) product of $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined as

$$\boldsymbol{u} \cdot \boldsymbol{v} = \sum_{i=1}^{m} (u_i \times v_i)$$

- From the geometry viewpoint:

$$\boldsymbol{u} \cdot \boldsymbol{v} = \boxed{\|\boldsymbol{u}\|_2} \times \|\boldsymbol{v}\|_2 \times \cos(\theta)$$



When $\boldsymbol{u} \cdot \boldsymbol{v} = 0 \Rightarrow \cos(\theta) = 0$

$\Rightarrow \theta = 90°$

$\Rightarrow \boldsymbol{u}$ and $\boldsymbol{v}$ are orthogonal

$$\|\boldsymbol{u}\|_2 = \sqrt{\boldsymbol{u} \cdot \boldsymbol{u}} = \sqrt{\sum_{i=1}^{m} (u_i \times u_i)}$$

Margin – Induction (cont.)

$$\begin{cases} \text{if } y_i = +1, \text{ then } \boldsymbol{w} \cdot \boldsymbol{x}_i + b > 0 \\ \text{if } y_i = -1, \text{ then } \boldsymbol{w} \cdot \boldsymbol{x}_i + b < 0 \end{cases}$$

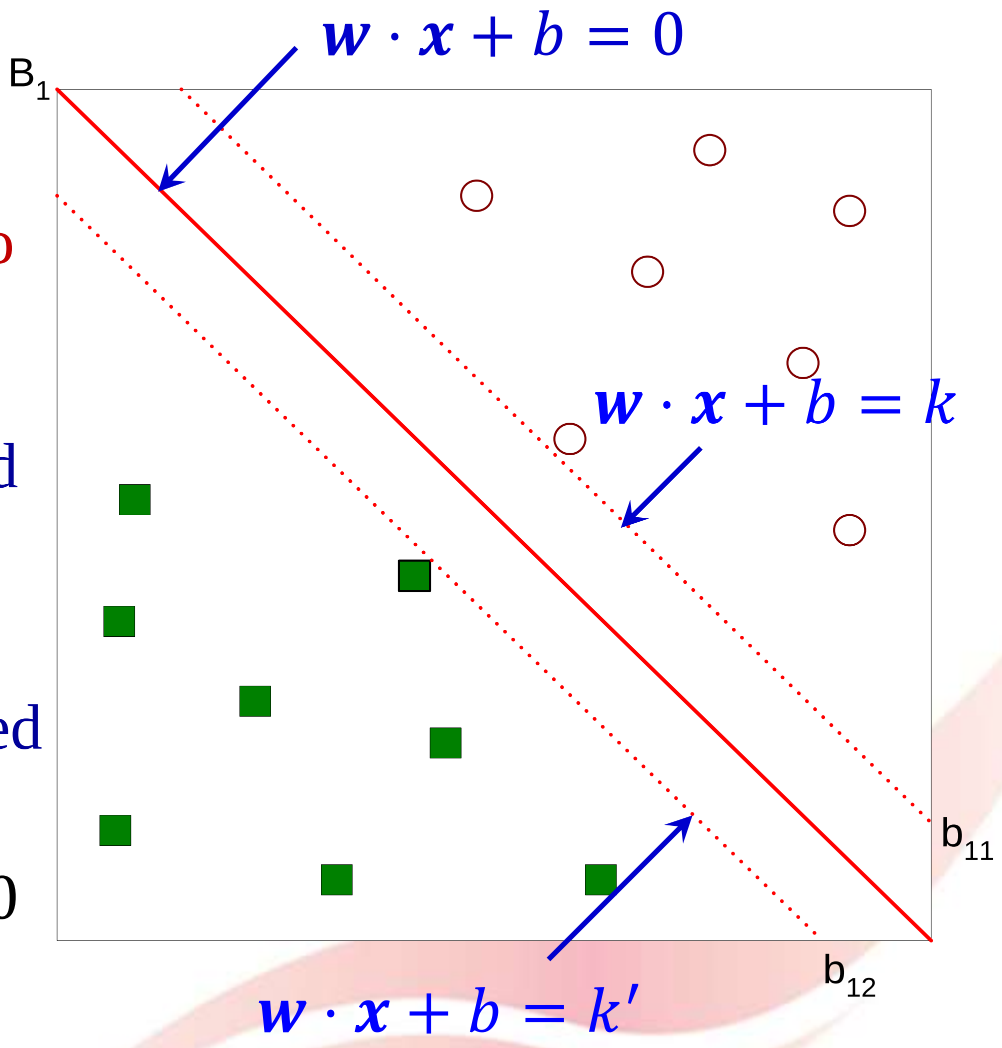
If the data instances of the two different classes are separable

For the closest circle $\boldsymbol{x}_c$ located above the decision boundary:

$$\boldsymbol{w} \cdot \boldsymbol{x}_c + b = k, \text{where } k > 0$$

For the closest square $\boldsymbol{x}_s$ located below the decision boundary:

$$\boldsymbol{w} \cdot \boldsymbol{x}_s + b = k', \text{where } k' < 0$$


$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
B1
$\boldsymbol{w} \cdot \boldsymbol{x} + b = k$
b11
b12
$\boldsymbol{w} \cdot \boldsymbol{x} + b = k'$

Margin – Induction (cont.)

The two parallel hyperplanes passing the closest circle(s) and square(s) can be written as

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = k, \text{where } k > 0$$

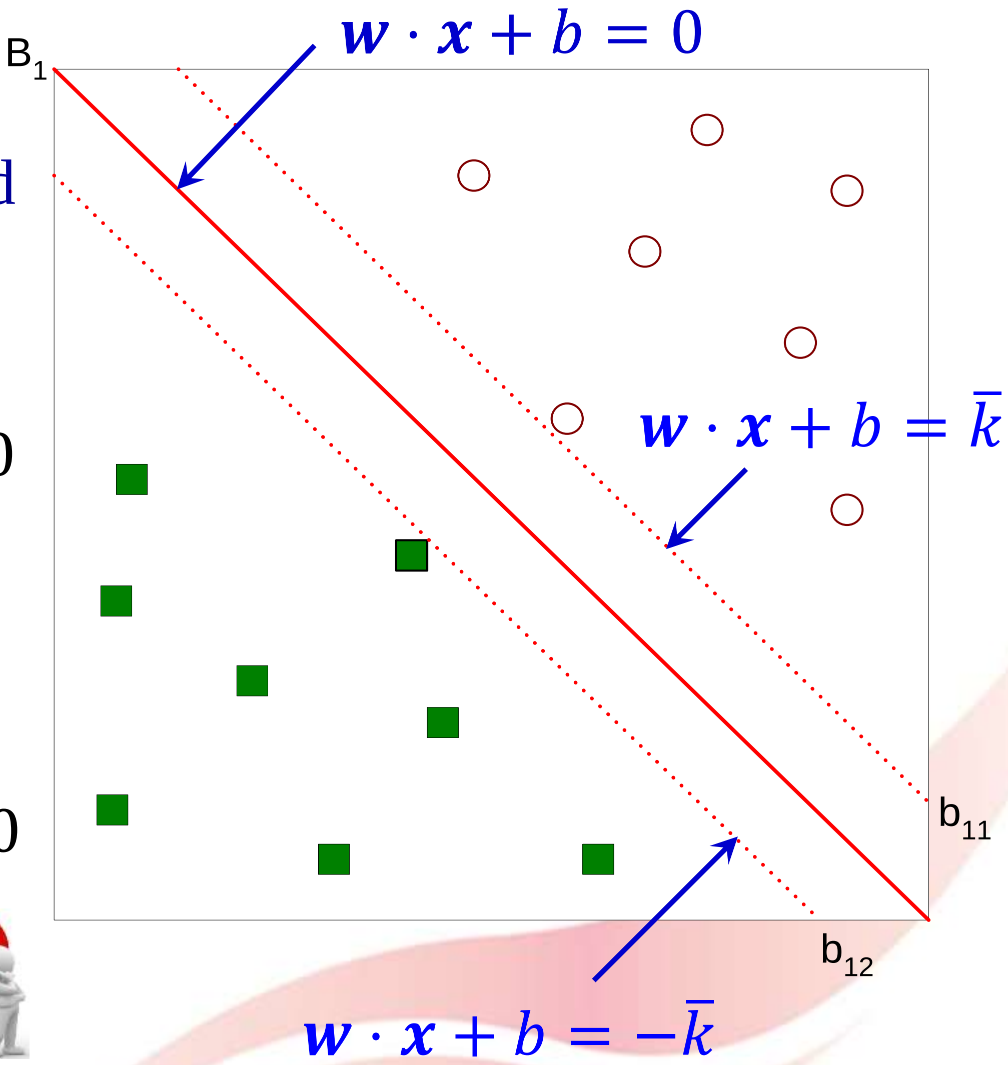
$$\boldsymbol{w} \cdot \boldsymbol{x} + b = k', \text{where } k' < 0$$

These two parallel hyperplanes can be further written as

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = +\bar{k}$$

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = -\bar{k}$$

where $\bar{k} > 0$


$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
B$_1$
$\boldsymbol{w} \cdot \boldsymbol{x} + b = \bar{k}$
b$_{11}$
b$_{12}$
$\boldsymbol{w} \cdot \boldsymbol{x} + b = -\bar{k}$

Margin – Induction (cont.)

$$\mathrm{d}(\mathrm{b}_{11}, \mathrm{B}_1) = \mathrm{d}(\boldsymbol{x}_1, \mathrm{B}_1) = \|\boldsymbol{x}_0 - \boldsymbol{x}_1\|_2$$

$$|(\boldsymbol{x}_0 - \boldsymbol{x}_1) \cdot \boldsymbol{w}|$$

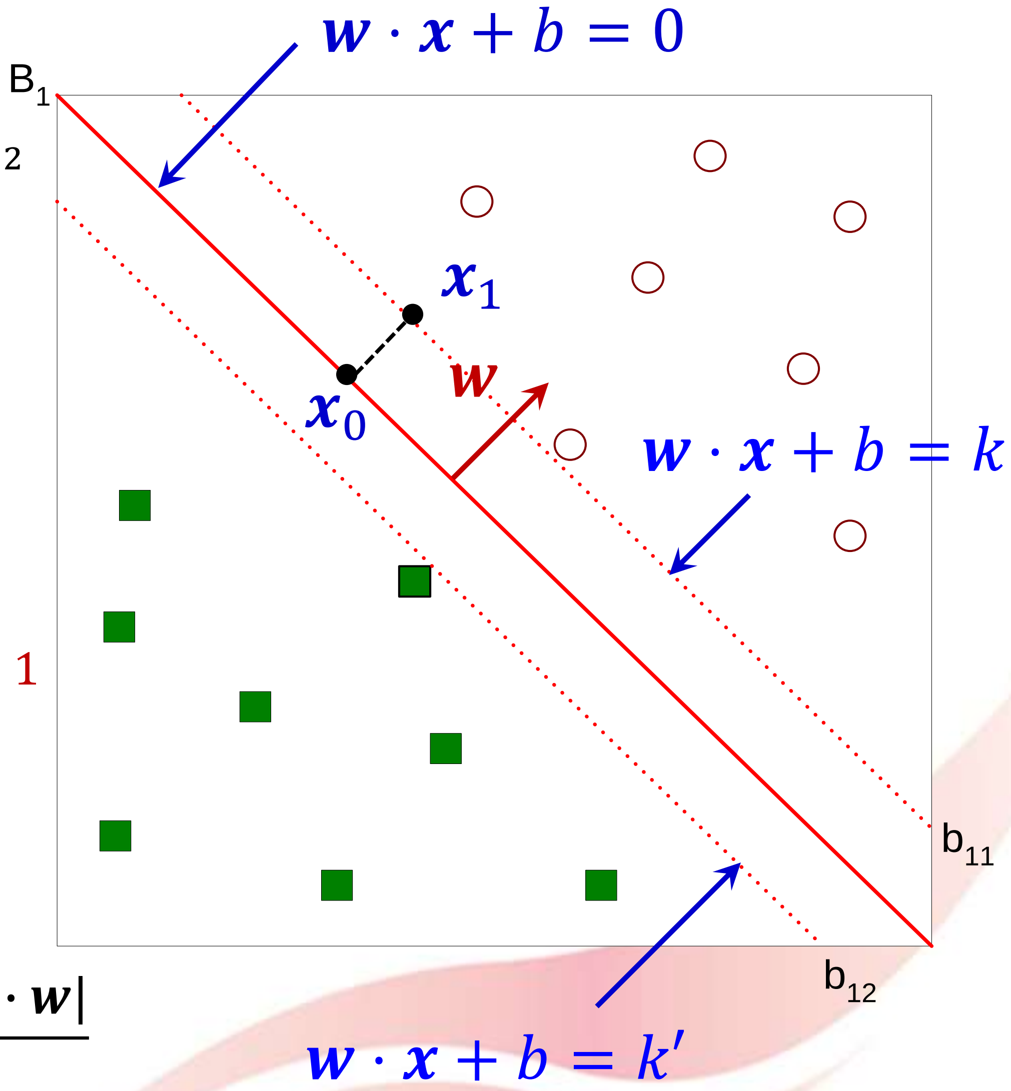
$$= \|\boldsymbol{x}_0 - \boldsymbol{x}_1\|_2 \|\boldsymbol{w}\|_2 |\cos(\theta)|$$

$\boldsymbol{x}_0 - \boldsymbol{x}_1$ and $\boldsymbol{w}$
are parallel, thus
$\theta = 0$ or 180
$\cos(\theta) = 1$ or -1

$$= \mathrm{d}(\boldsymbol{x}_1, \mathrm{B}_1) \|\boldsymbol{w}\|_2$$

$$\Downarrow$$

$$\mathrm{d}(\mathrm{b}_{11}, \mathrm{B}_1) = \mathrm{d}(\boldsymbol{x}_1, \mathrm{B}_1) = \frac{|(\boldsymbol{x}_0 - \boldsymbol{x}_1) \cdot \boldsymbol{w}|}{\|\boldsymbol{w}\|_2}$$


$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
B1
$\boldsymbol{x}_1$
$\boldsymbol{w}$
$\boldsymbol{x}_0$
$\boldsymbol{w} \cdot \boldsymbol{x} + b = k$
b11
b12
$\boldsymbol{w} \cdot \boldsymbol{x} + b = k'$

Margin – Induction (cont.)

$$\mathrm{d}(\mathrm{b}_{11}, \mathrm{B}_1) = \frac{|(\boldsymbol{x}_0 - \boldsymbol{x}_1) \cdot \boldsymbol{w}|}{\|\boldsymbol{w}\|_2}$$

$$\Downarrow$$

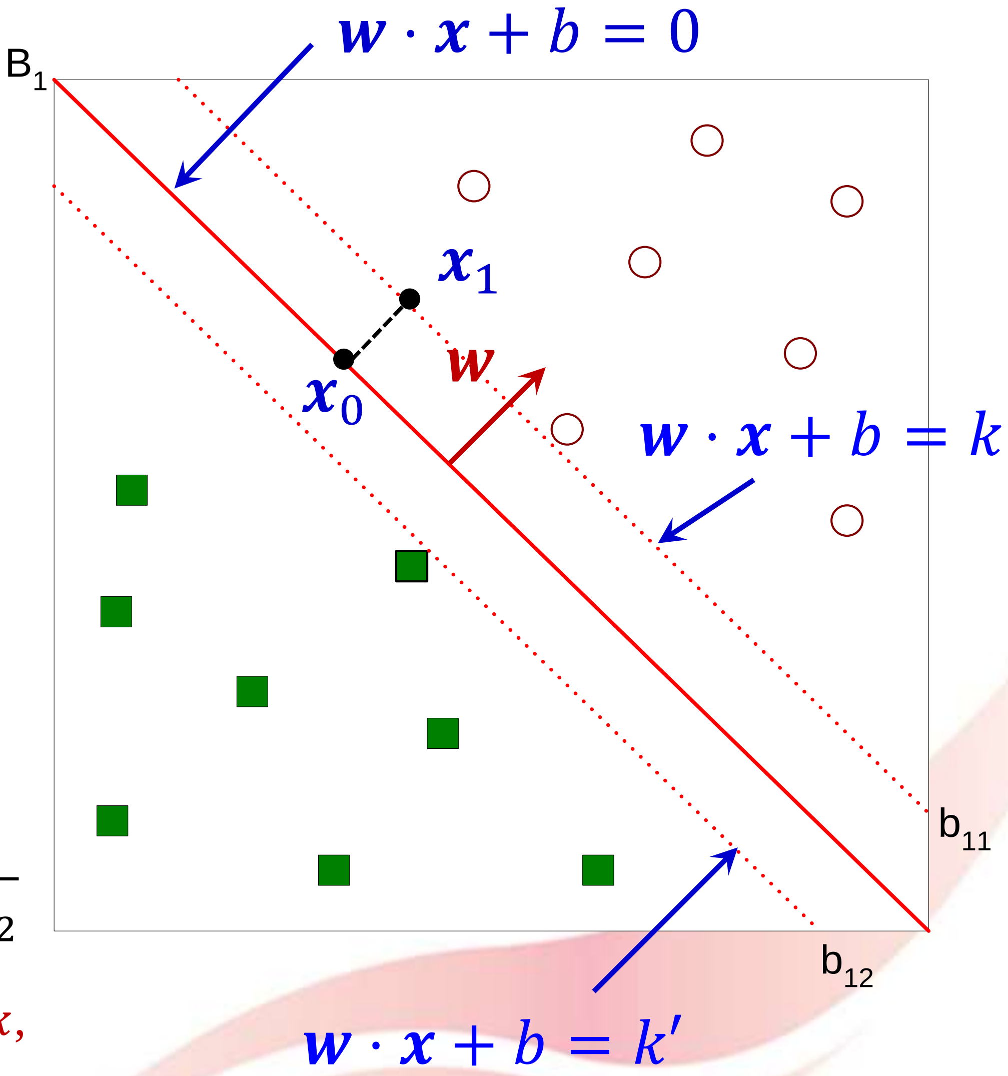
$$\mathrm{d}(\mathrm{b}_{11}, \mathrm{B}_1) = \frac{|\boldsymbol{x}_0 \cdot \boldsymbol{w} - \boldsymbol{x}_1 \cdot \boldsymbol{w}|}{\|\boldsymbol{w}\|_2}$$

As $\boldsymbol{x}_0$ is on the decision boundary $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$, thus $\boldsymbol{w} \cdot \boldsymbol{x}_0 = -b$

$$\mathrm{d}(\mathrm{b}_{11}, \mathrm{B}_1) = \frac{|-b - \boldsymbol{x}_1 \cdot \boldsymbol{w}|}{\|\boldsymbol{w}\|_2}$$

$$= \frac{|\boldsymbol{x}_1 \cdot \boldsymbol{w} + b|}{\|\boldsymbol{w}\|_2} = \frac{|k|}{\|\boldsymbol{w}\|_2}$$

As $\boldsymbol{x}_1$ is on $\boldsymbol{w} \cdot \boldsymbol{x} + b = k$, thus $\boldsymbol{w} \cdot \boldsymbol{x}_1 + b = k$


$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
B1
$\boldsymbol{x}_1$
$\boldsymbol{w}$
$\boldsymbol{x}_0$
$\boldsymbol{w} \cdot \boldsymbol{x} + b = k$
b11
b12
$\boldsymbol{w} \cdot \boldsymbol{x} + b = k'$

Margin – Induction (cont.)

$$d(b_{11}, B_1) = \frac{|k|}{\|\boldsymbol{w}\|_2}$$

Similarly $d(b_{12}, B_1) = \frac{|k'|}{\|\boldsymbol{w}\|_2}$

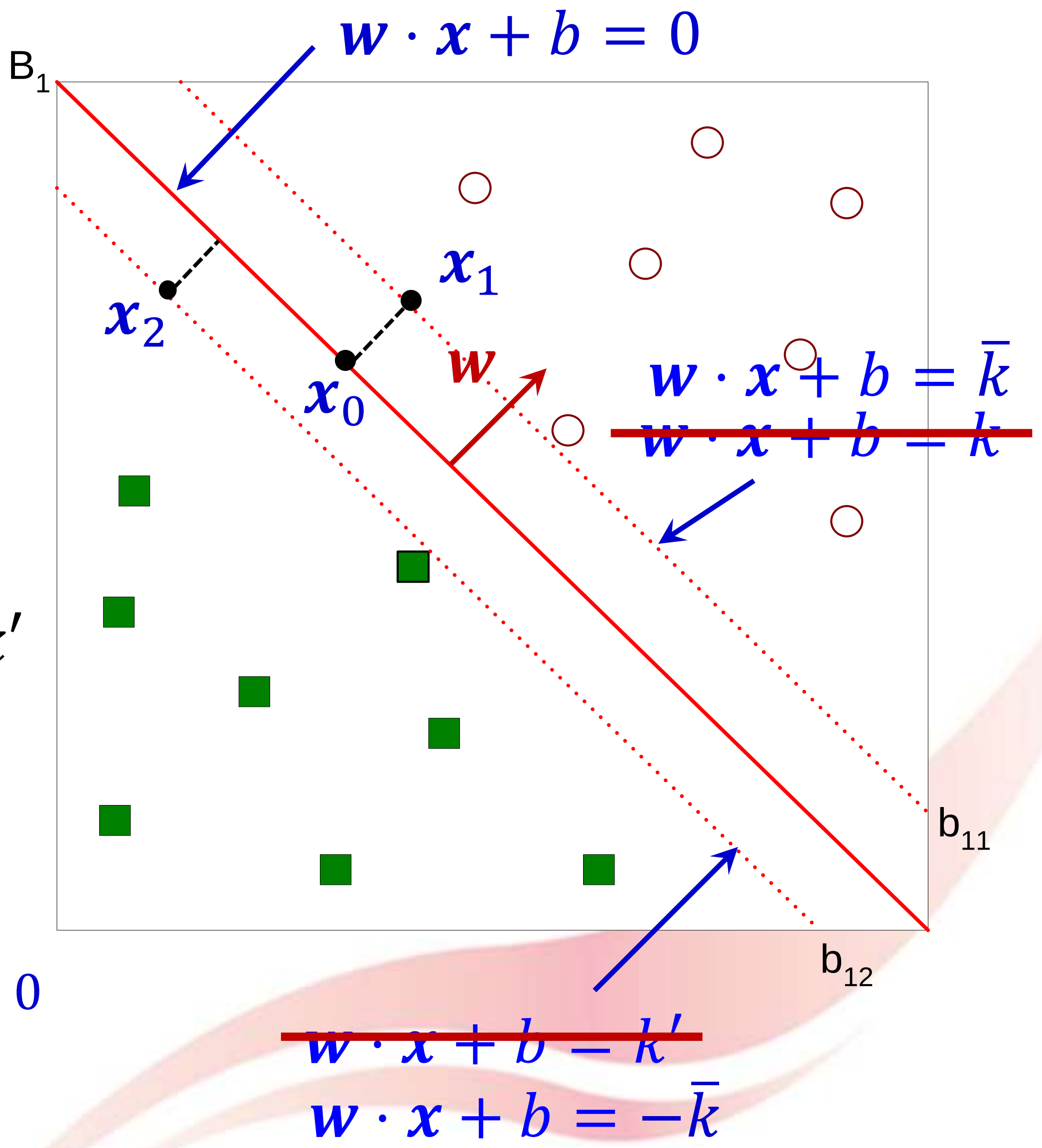
$$d(b_{11}, B_1) = d(b_{12}, B_1)$$

$$\Downarrow$$

$$\frac{|k|}{\|\boldsymbol{w}\|_2} = \frac{|k'|}{\|\boldsymbol{w}\|_2} \Rightarrow k = -k'$$

Denote by $\bar{k} = k = -k' > 0$

$$\begin{cases} \boldsymbol{w} \cdot \boldsymbol{x} + b = \bar{k} \\ \boldsymbol{w} \cdot \boldsymbol{x} + b = -\bar{k} \end{cases}$$ where $\bar{k} > 0$



$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
B1
$\boldsymbol{x}_1$
$\boldsymbol{x}_2$
$\boldsymbol{x}_0$
$\boldsymbol{w}$
$\boldsymbol{w} \cdot \boldsymbol{x} + b = \bar{k}$
~~$\boldsymbol{w} \cdot \boldsymbol{x} + b = k$~~
b11
b12
~~$\boldsymbol{w} \cdot \boldsymbol{x} + b = k'$~~
$\boldsymbol{w} \cdot \boldsymbol{x} + b = -\bar{k}$

Margin – Induction (cont.)

The two parallel hyperplanes passing the closest circle(s) and square(s) can be written as

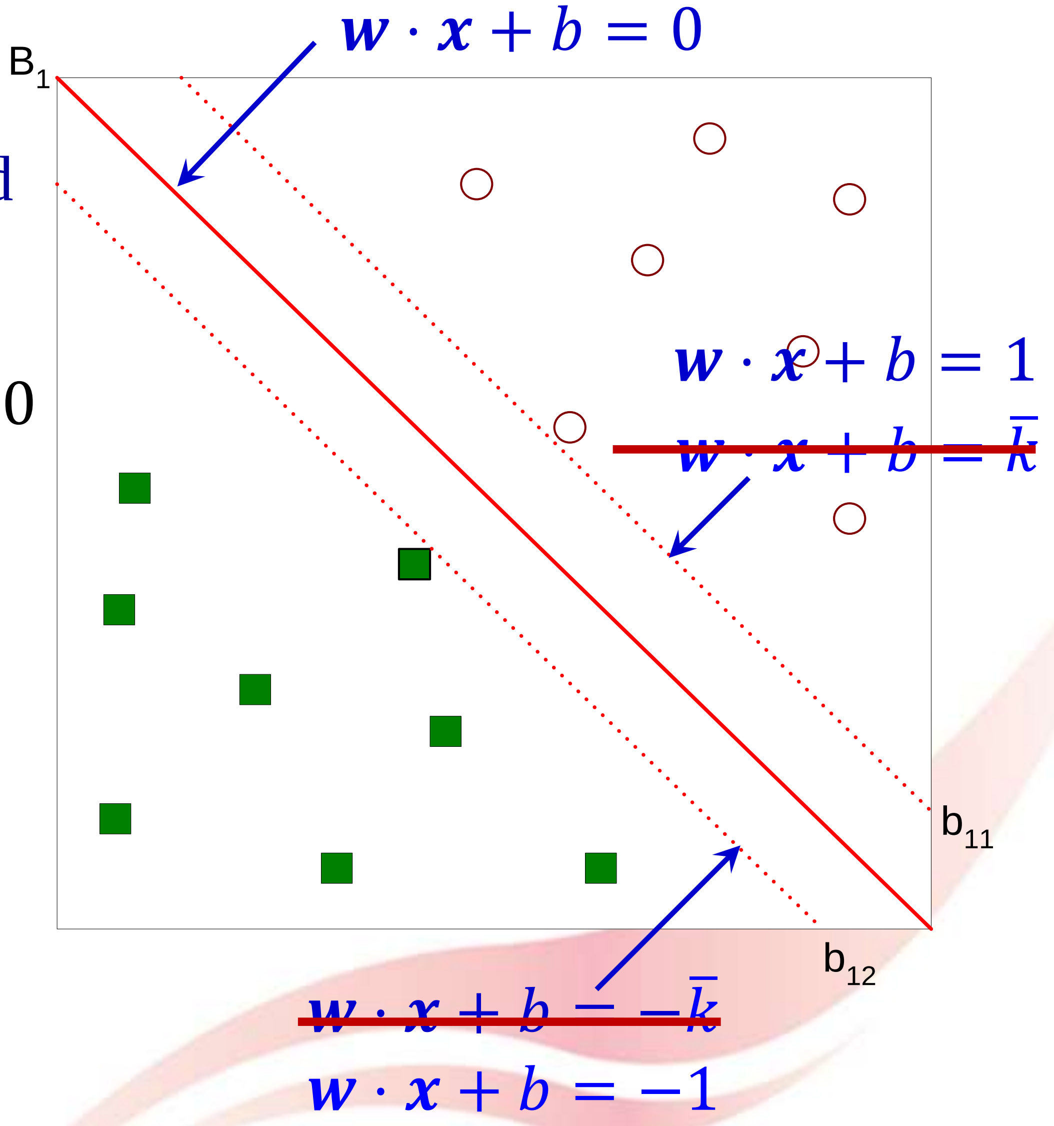
$$\boldsymbol{w}\cdot\boldsymbol{x}+b=\bar{k}$$
$$\boldsymbol{w}\cdot\boldsymbol{x}+b=-\bar{k}$$

where $\bar{k}>0$

$$\boldsymbol{w}=\frac{\boldsymbol{w}}{\bar{k}},\, b=\frac{b}{\bar{k}}$$

After rescaling $\boldsymbol{w}$ and b, the two parallel hyperplanes can be further rewritten as

$$\boldsymbol{w}\cdot\boldsymbol{x}+b=+1$$
$$\boldsymbol{w}\cdot\boldsymbol{x}+b=-1$$



$\boldsymbol{w}\cdot\boldsymbol{x}+b=0$
B_1
$\boldsymbol{w}\cdot\boldsymbol{x}+b=1$
~~$\boldsymbol{w}\cdot\boldsymbol{x}+b=\bar{k}$~~
b_{11}
b_{12}
~~$\boldsymbol{w}\cdot\boldsymbol{x}+b=-\bar{k}$~~
$\boldsymbol{w}\cdot\boldsymbol{x}+b=-1$

Margin – Induction (cont.)

b_{11}: $\boldsymbol{w} \cdot \boldsymbol{x}_1 + b = +1$

b_{12}: $\boldsymbol{w} \cdot \boldsymbol{x}_2 + b = -1$

$\Downarrow$

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

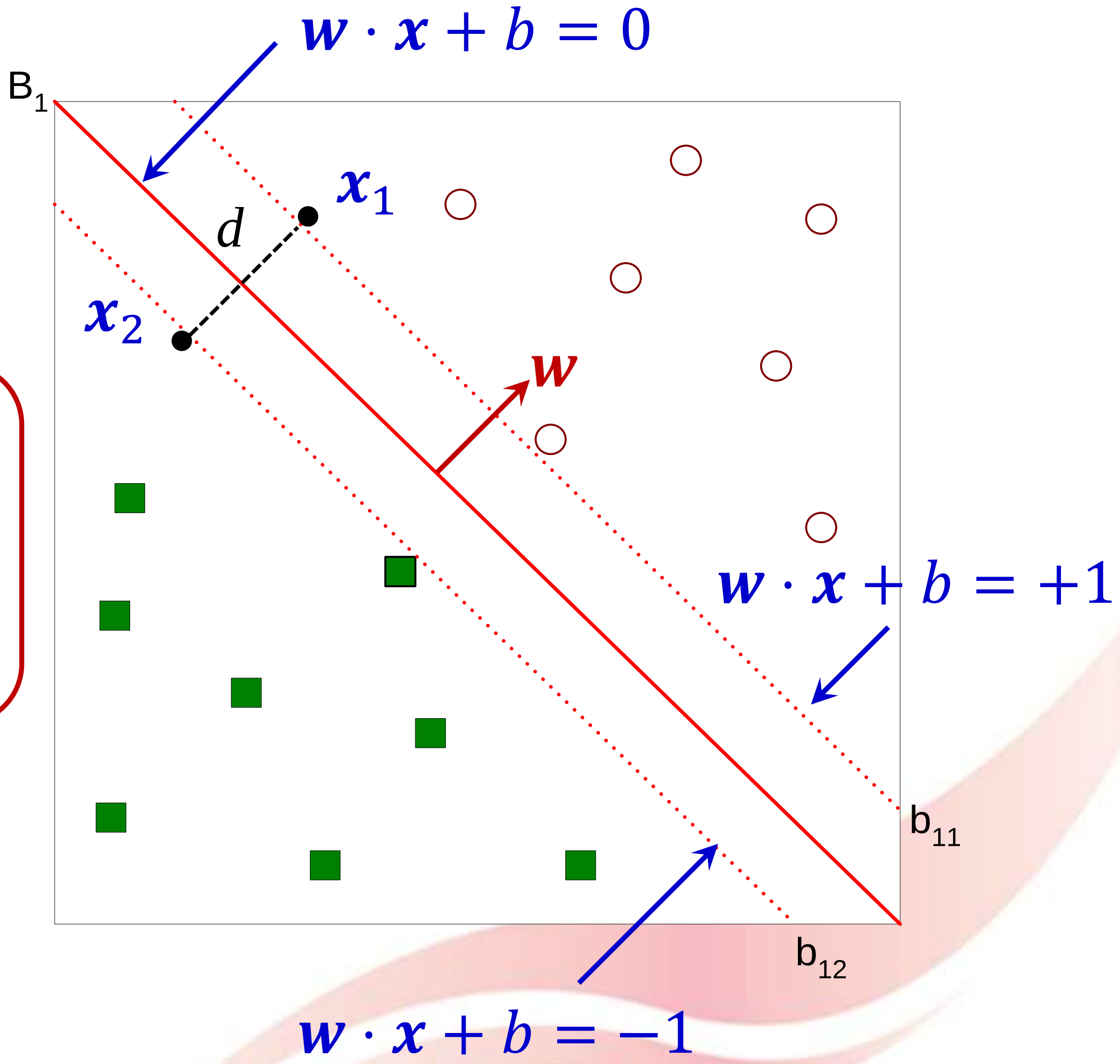
$$\mathrm{d}(\mathrm{b}_{12}, \mathrm{b}_{11}) = \frac{|(\boldsymbol{x}_2 - \boldsymbol{x}_1) \cdot \boldsymbol{w}|}{\|\boldsymbol{w}\|_2}$$

Based on our previous induction

$\Downarrow$

$$\boxed{\mathrm{d}(\mathrm{b}_{12}, \mathrm{b}_{11})} = \frac{2}{\|\boldsymbol{w}\|_2}$$

margin d



$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
B_1
$\boldsymbol{x}_1$
d
$\boldsymbol{x}_2$
$\boldsymbol{w}$
$\boldsymbol{w} \cdot \boldsymbol{x} + b = +1$
b_{11}
b_{12}
$\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$

Objective & Constraints

- Goal: to learn a decision boundary $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ w.r.t. $\boldsymbol{w}$ and b with the largest margin over the training data $\{\boldsymbol{x}_i, y_i\}, i = 1, \ldots, N$

$$\text{margin} = \frac{2}{\|\boldsymbol{w}\|_2} \Longrightarrow \max_{\boldsymbol{w},b} \frac{2}{\|\boldsymbol{w}\|_2} \Longleftrightarrow \min_{\boldsymbol{w},b} \frac{\boxed{\|\boldsymbol{w}\|_2}}{2}$$

$$\|\boldsymbol{w}\|_2 = \sqrt{\sum_{i=1}^{m} w_i^2}$$

Objective: $$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2}$$

Constraints: $$\boxed{\begin{array}{l} \boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq 1, \text{if } y_i = +1 \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1, \text{if } y_i = -1 \end{array}}$$

Ensure data instances of different classes are separable, and no data instances are located within the margin

Objective & Constraints (cont.)

Objective: $$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2}$$

For each $\{\boldsymbol{x}_i, y_i\}$, constraint:

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq 1, \text{if } y_i = +1$$

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1, \text{if } y_i = -1$$

The constraints can be simplified as

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$$

When $y_i = +1$, $(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$

When $y_i = -1$, $-1 \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \Rightarrow (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \leq -1$

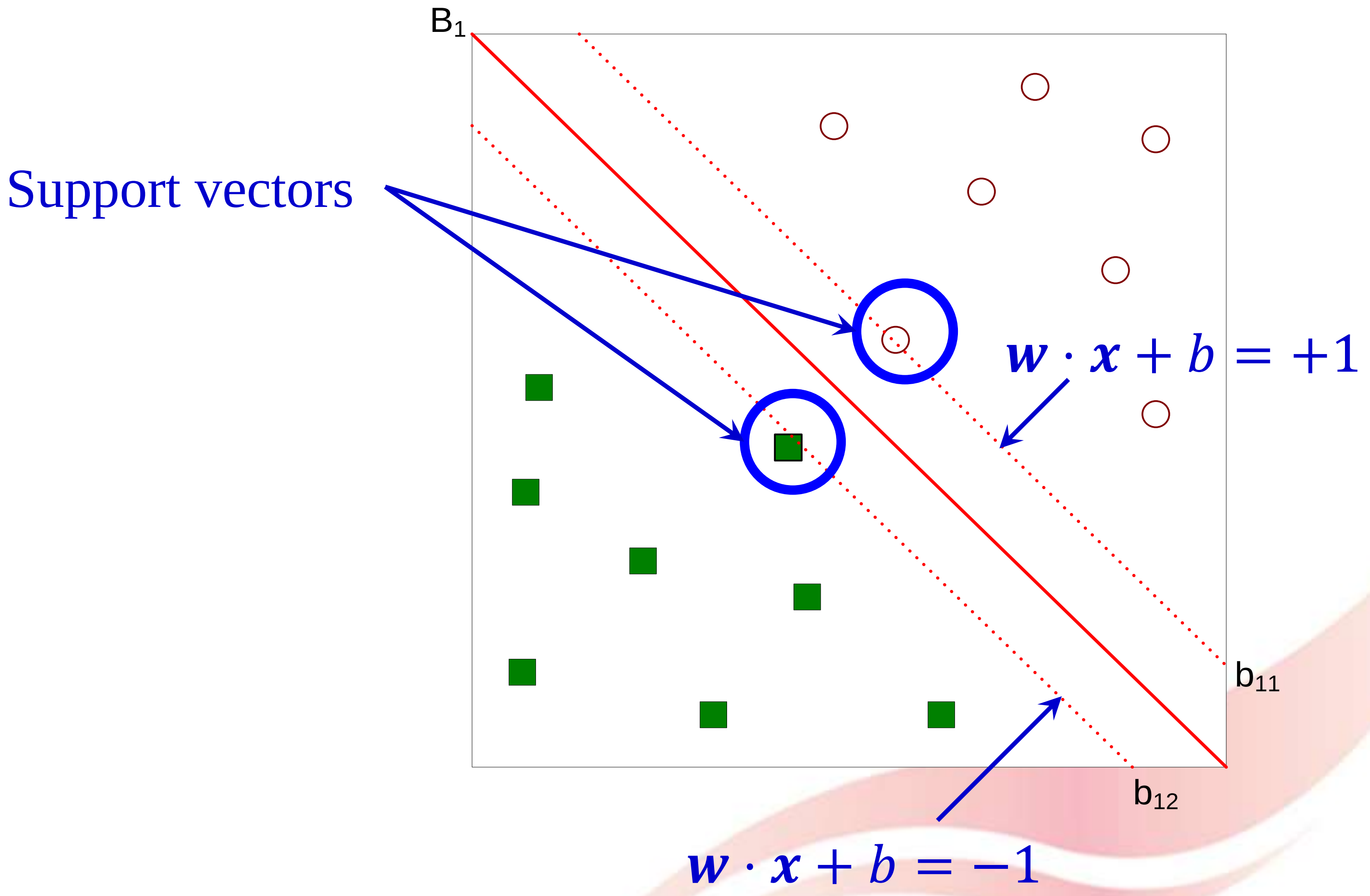
Optimization Problem

- The optimization problem of linear SVMs

$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2}$$
$$\text{s.t.} \quad y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1,\ i = 1, \ldots, N$$

- Convex optimization
  - Many numerical approaches can be applied to solve it
  - Specially, quadratic optimization problem with linear inequality constraints

Support Vectors


B_1
Support vectors
$\boldsymbol{w} \cdot \boldsymbol{x} + b = +1$
b_11
b_12
$\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$

Assumption: Separable

$$\boxed{\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2}}$$

Maximize margin

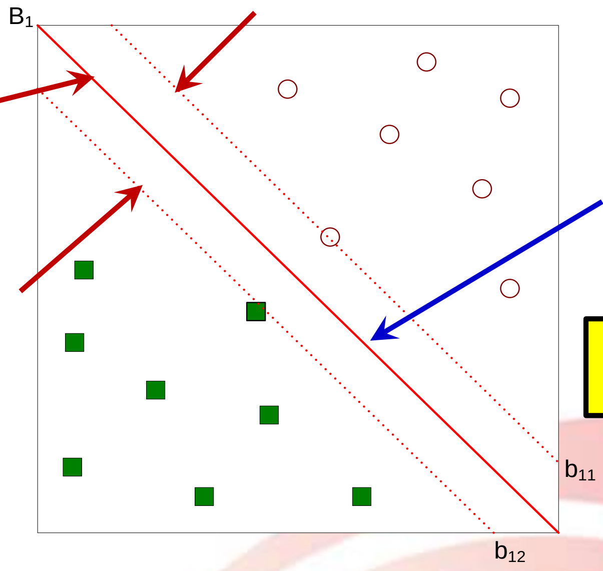
$$\boxed{\text{s.t.} \quad y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1,\ i = 1, \ldots, N}$$

Perfectly separate training data

$\boldsymbol{w} \cdot \boldsymbol{x} + b = 1$

$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$

$\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$

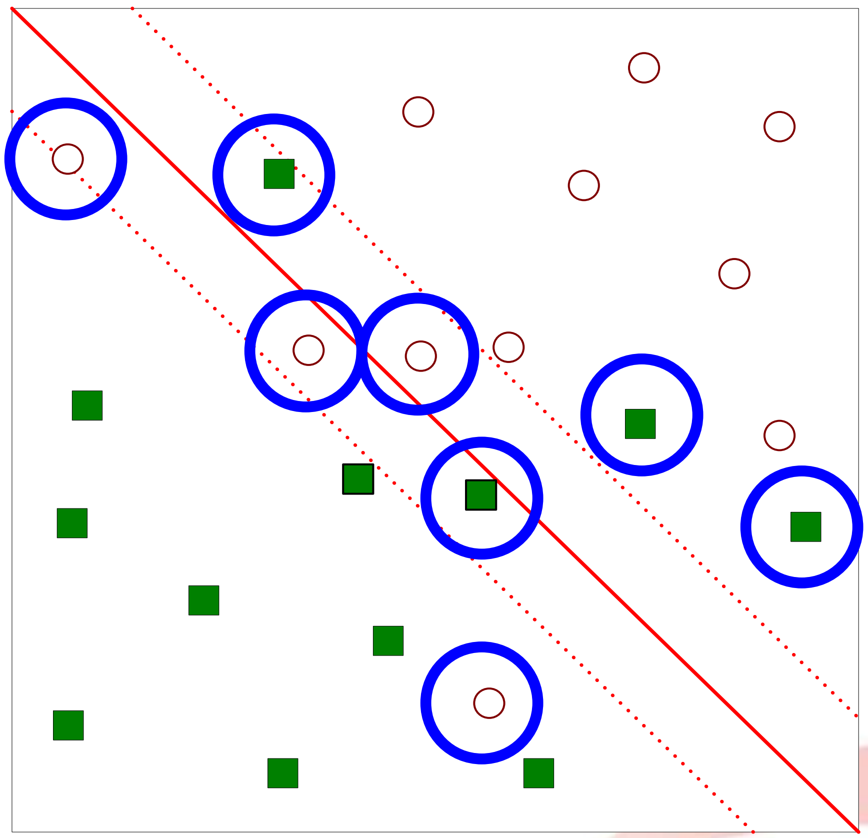


No training data locate within the margin

Hard-margin SVMs

Non-separable Case

- What if data of two classes cannot be perfectly separated?



Slack variables $\xi_i \geq 0$ need to be introduced to absorb errors

Slack Variables

- For Separable Case:

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq 1, \text{ if } y_i = +1$$

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1, \text{ if } y_i = -1$$

OR $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$

- For Non-separable Case:

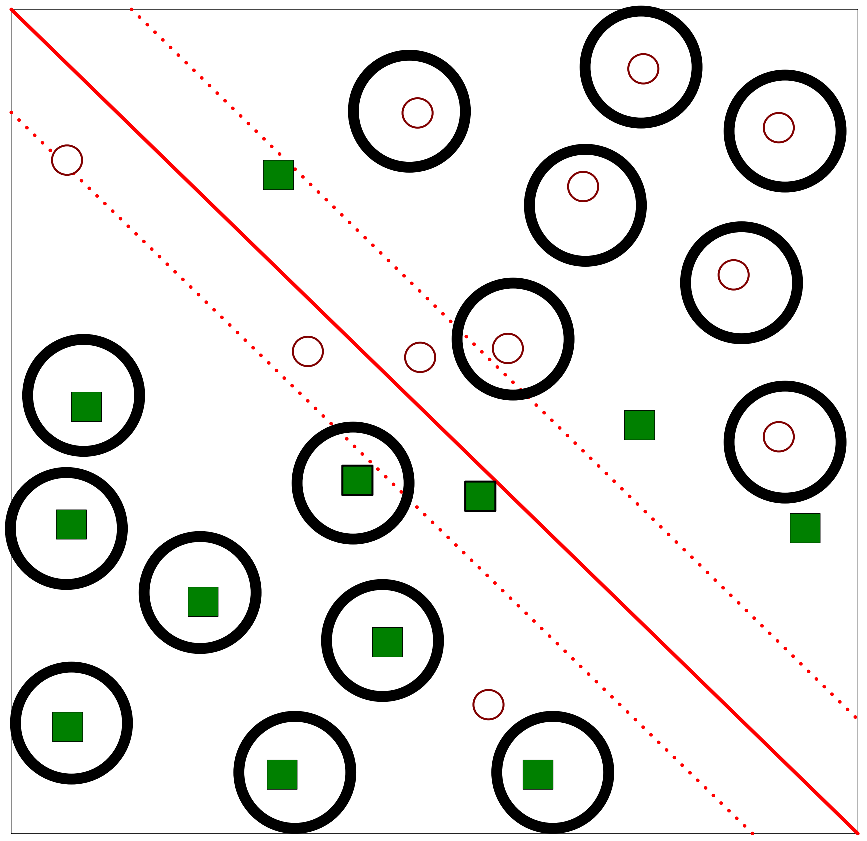
$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq 1 - \boxed{\xi_i}, \text{ if } y_i = +1$$

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1 + \boxed{\xi_i}, \text{ if } y_i = -1$$

OR $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i$

$\xi_i \geq 0$, Slack variables

Slack Variables (cont.)



If $\xi_i = 0$

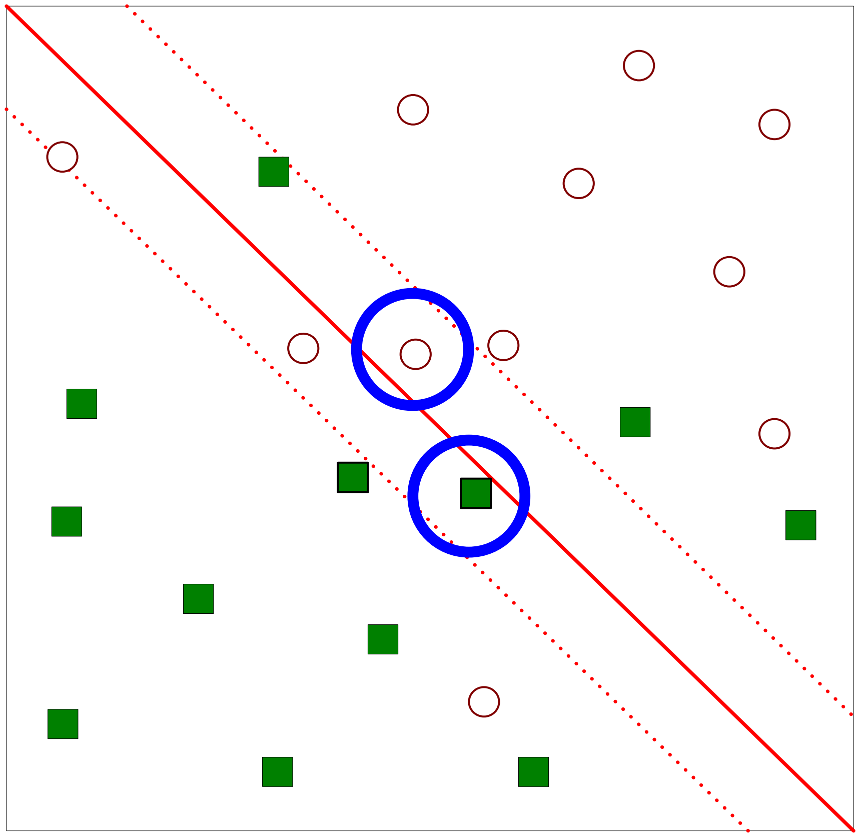
$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\Downarrow$$

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$$

$\boldsymbol{x}_i$ is correctly classified and not within the margin

Slack Variables (cont.)



If $0 < \xi_i < 1$

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i$$

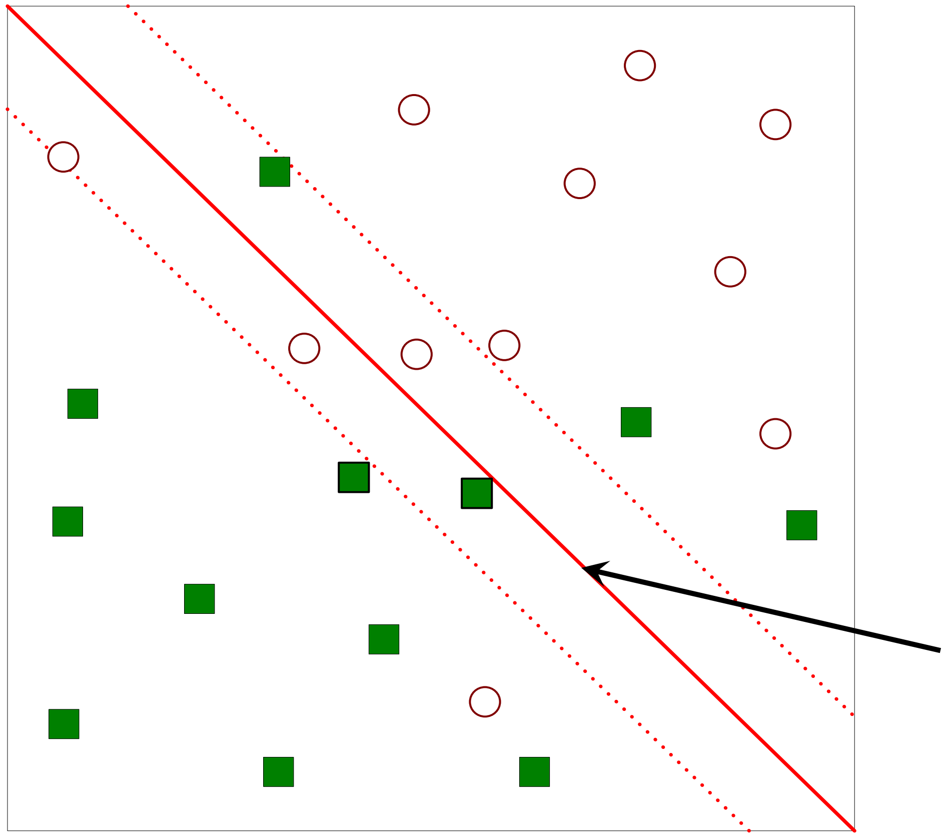
$$\Downarrow$$

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq k$$

$$0 < k < 1$$

$\boldsymbol{x}_i$ is correctly classified but within the margin

Slack Variables (cont.)



If $\xi_i = 1$

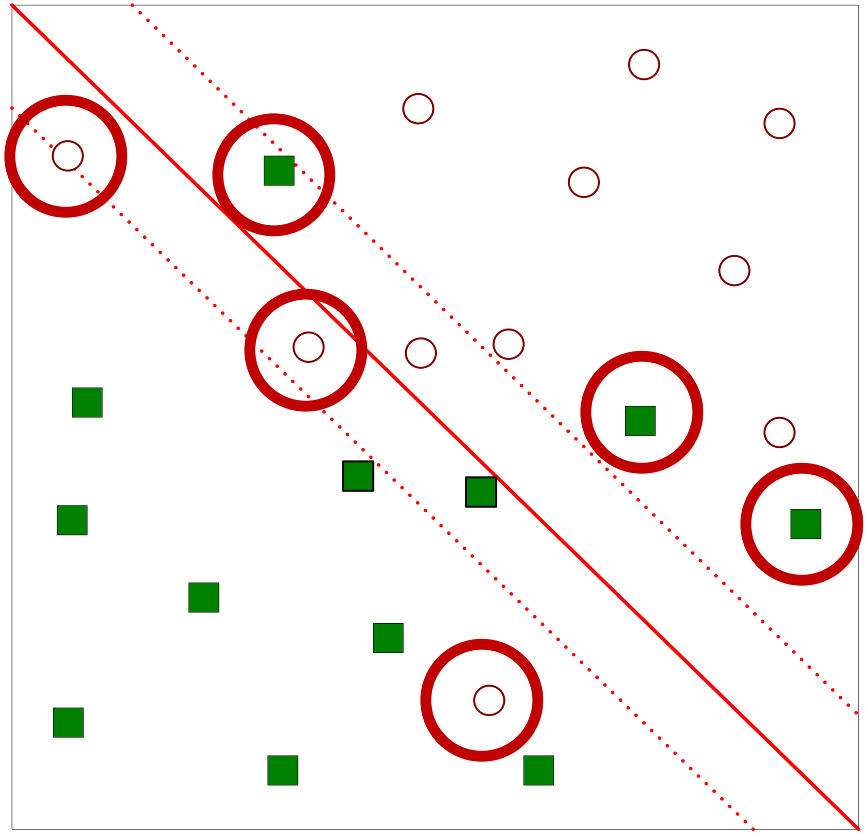
$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\Downarrow$$

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 0$$

$\boldsymbol{x}_i$ might be on the decision boundary (random guess)

Slack Variables (cont.)



$\boldsymbol{x}_i$ is allowed to be incorrectly classified

If $\xi_i > 1$

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\Downarrow$$

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq t \qquad t < 0$$

$$\Downarrow$$

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq t, \text{ if } y_i = +1$$

There may exist a $\tilde{t}, t < \tilde{t} < 0$, s.t.

$$t \leq \boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq \tilde{t} < 0, \text{ but } y_i = +1$$

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -t, \text{ if } y_i = -1$$

There may exist $\tilde{t}, 0 < \tilde{t} < -t$, s.t.

$$0 < \tilde{t} \leq \boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -t, \text{ but } y_i = -1$$

Soft Error

- The number of misclassifications is $\#\{\xi_i > 1\}$
- The number of non-separable points is $\#\{0 < \xi_i \leq 1\}$
- Soft errors:

$$\sum_i \xi_i$$

Linear Soft-Margin SVMs

Soft-margin SVMs

- Linear SVMs with soft errors

$$\min_{\boldsymbol{w},b,\xi_i} \frac{\|\boldsymbol{w}\|_2^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right)$$

$$\text{s.t.} \quad y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i,\ i = 1, \ldots, N, \quad \xi_i \geq 0$$

Penalize the decision boundary with large values of slack variables

$C > 0$ is a tradeoff parameter to balance the impact of margin maximization and tolerable errors

Nonnegative ξ_i provides an estimate of the error of the decision boundary on the training example $\boldsymbol{x}_i$

Empirical Risk Minimization: Revisit

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \lambda\Omega(\boldsymbol{\theta})$$

$$\min_{\boldsymbol{w},b,\xi_i} \frac{\|\boldsymbol{w}\|_2^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right)$$

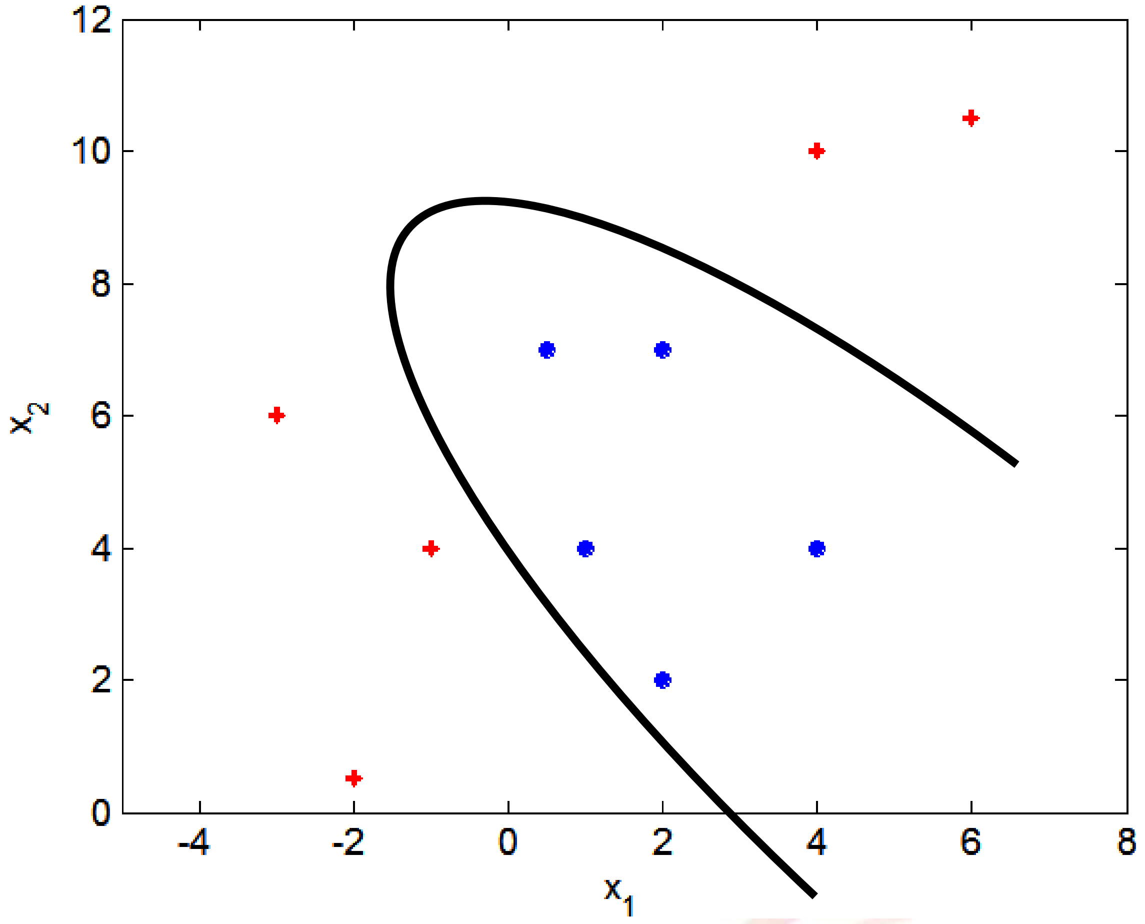
$$\text{s.t.} \quad y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i,\ i = 1,\ldots,N,$$

$$\xi_i \geq 0$$

The implicit bias of logistic regression is that it tends to minimize the distance of the majority class.
In practice it means the hyperplane will be closer to the majority class, instead of maximising margin.
This is why SVM is considered to be superior, since it does not care the sample bias but only support vectors to maximize margin.

Nonlinear SVMs

- What if the decision boundary is not linear?



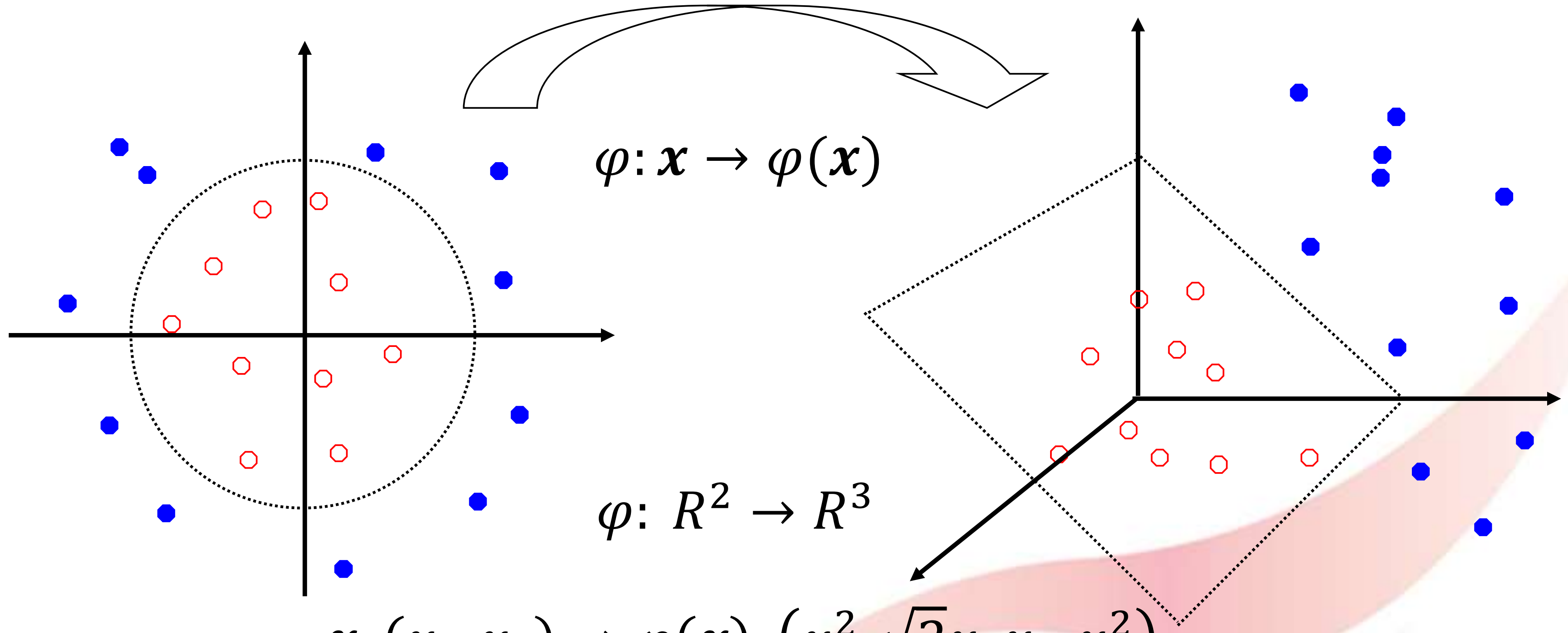
Kernel methods

Nonlinear SVMs (cont.)

- How to extend linear decision boundary to nonlinear?
- Design a nonlinear model and learn parameters from data
  - Resultant optimization problems are very difficult to solve, especially for multi-dimensional cases
- Transform the input space to a higher dimensional space via nonlinear mappings to “make life easier”
  - To learn a linear model in the transformed space
    - The linear model in the nonlinearly transformed space is indeed a nonlinear model in the original input space
    - Motivation: in a higher dimensional space, there is a higher chance to find a linear hyperplane as a decision boundary

Nonlinear Feature Mapping

- The original input space can be mapped to some higher-dimensional feature space where the training set is separable



$$\varphi: \boldsymbol{x} \rightarrow \varphi(\boldsymbol{x})$$

$$\varphi:\ R^2 \rightarrow R^3$$

$$\boldsymbol{x}: (x_1, x_2) \rightarrow \varphi(\boldsymbol{x}): \left(x_1^2, \sqrt{2}x_2x_1, x_2^2\right)$$

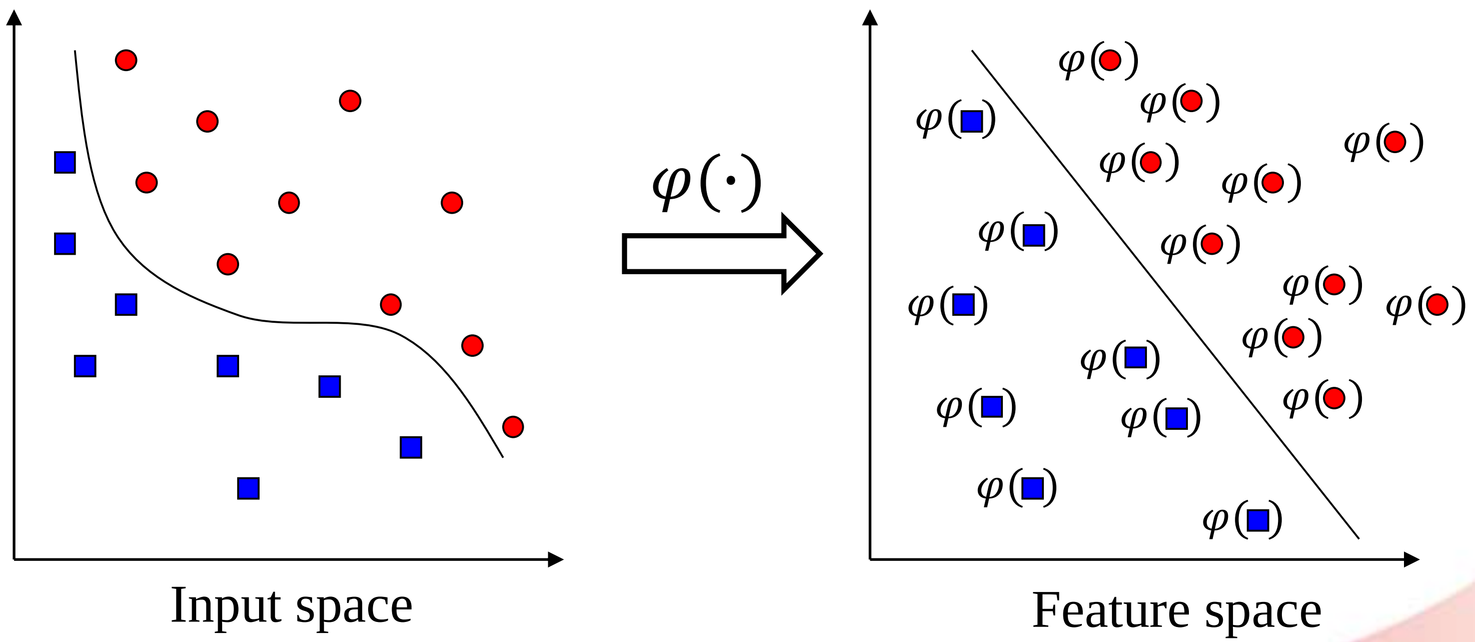
Nonlinear Hard-margin SVMs

- For each data instance $\boldsymbol{x}_i$, we map it to a higher-dimensional space via a nonlinear mapping $\varphi: \boldsymbol{x}_i \to \varphi(\boldsymbol{x}_i)$
- Nonlinear SVMs aim to learn a hyperplane $\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b = 0$ in the mapped space via

$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2}$$
$$\text{s.t.} \quad y_i(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b) \geq 1,\ i = 1, \ldots, N$$

- The hyperplane $\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b = 0$ is linear w.r.t. $\varphi(\boldsymbol{x}_i)$, but nonlinear w.r.t. $\boldsymbol{x}_i$

Nonlinear Hard-margin SVMs (cont.)



Nonlinear Hard-margin SVMs (cont.)

$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2} \qquad \varphi: \mathbb{R}^m \to \mathbb{R}^h, \text{where } h \gg m \text{ in general}$$

$$\text{s.t.} \quad y_i(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b) \geq 1,\ i = 1, \ldots, N$$

- How to design the mapping φ explicitly?
  - Directly design the $\mathbb{R}^m \to \mathbb{R}^h$ mapping function
    - It is very difficult to design a vector-valued function
  - Design $\varphi_j: \mathbb{R}^m \to \mathbb{R}, j = 1, \ldots, h$ functions, and construct $\varphi = (\varphi_1, \ldots, \varphi_h)$
    - Which parametric form should be used for each φ_j
    - What hyper-parameter settings of each φ_j should be used?
    - How many φ_j, i.e., the value of h, should be used?

The kernel trick comes to rescue

Kernel Trick

- Suppose $\varphi(\cdot)$ is given as follows, mapping a data instance from 2-dimensional space to 6-dimensional space:

$$\varphi(\boldsymbol{x}) = \varphi([x_1, x_2]) = \left[1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2\right]$$

- Given two data instances: $\boldsymbol{a} = [a_1, a_2]$ and $\boldsymbol{b} = [b_1, b_2]$

$$\varphi(\boldsymbol{a}) = \left[1, \sqrt{2}a_1, \sqrt{2}a_2, a_1^2, a_2^2, \sqrt{2}a_1a_2\right]$$

$$\varphi(\boldsymbol{b}) = \left[1, \sqrt{2}b_1, \sqrt{2}b_2, b_1^2, b_2^2, \sqrt{2}b_1b_2\right]$$

- Inner product of the two instances after feature mapping:

$$\varphi(\boldsymbol{a}) \cdot \varphi(\boldsymbol{b}) = 1 + 2a_1b_1 + 2a_2b_2 + a_1^2b_1^2 + a_2^2b_2^2 + 2a_1a_2b_1b_2$$

$$= (1 + a_1b_1 + a_2b_2)^2$$

Kernel Trick (cont.)

- Inner product of the two instances after feature mapping:

$$\varphi(\boldsymbol{a}) \cdot \varphi(\boldsymbol{b}) = (1 + a_1 b_1 + a_2 b_2)^2$$

- If we define the kernel function as follows, there is no need to carry out $\varphi(\cdot)$ explicitly to compute the inner product between mapped data instances

$$k(\boldsymbol{a}, \boldsymbol{b}) = (1 + a_1 b_1 + a_2 b_2)^2 = (1 + \boldsymbol{a} \cdot \boldsymbol{b})^2$$

- This use of kernel function to avoid carrying out $\varphi(\cdot)$ explicitly is known as the kernel trick
- Note: as long as in the optimization problem we only need to compute the inner product between data instances rather than using each individual data instance, we can use the kernel trick

Kernel Functions

- In functional analysis, for a kernel function $k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$, where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are m-dimensional vectors, then there exists a feature mapping $\varphi\colon \mathbb{R}^m \rightarrow \mathbb{R}^h$, s.t. $k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$
  - Note that φ is not known explicitly, and h can be infinite
- Some well-known kernel functions
  - Linear kernel: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$
  - Radial basis function (RBF) kernel with width σ:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2^2}{2\sigma^2}\right)$$

  - Polynomial kernel with degree d

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\boldsymbol{x}_i \cdot \boldsymbol{x}_j + 1\right)^d$$

Nonlinear Hard-margin SVMs Revisit

- Optimization problem for nonlinear SVMs

Primal Form

$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2}$$

$$\text{s.t.} \quad y_i \times (\boldsymbol{w} \cdot \boxed{\varphi(\boldsymbol{x}_i)} + b) \geq 1,\ i = 1, \dots, N$$

data instances appear individually not in the form of inner product in the optimization problem

- The kernel trick is not applicable
- What about its dual form?

Lagrange Multiplier Method

- Given: an objective $f(\boldsymbol{w})$ to be minimized, with a set of inequality constraints to be satisfied

$$\begin{aligned} &\min_{\boldsymbol{w}} \; f(\boldsymbol{w}) \\ &\text{s.t.} \quad h_i(\boldsymbol{w}) \leq 0,\; i = 1, \dots, q \end{aligned}$$

- The Lagrangian for the optimization problem:

$$L(\boldsymbol{w}, \boxed{\boldsymbol{\lambda}}) = f(\boldsymbol{w}) + \sum_{i=1}^{q} \boxed{\lambda_i} h_i(\boldsymbol{w})$$

$\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_q)$ The Lagrange multipliers

The Dual Form (Hard-margin)

Primal Form

$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2}$$

$$\text{s.t.} \quad y_i(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b) \geq 1,\ i = 1, \dots, N$$

$$\Big\Downarrow$$

Through the Lagrange Multiplier Method, KKT conditions, etc.

Out of scope

Dual Form

$$\max_{\boldsymbol{\lambda}} L_D(\boldsymbol{\lambda}) = -\left(\frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j \big(\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)\big) - \sum_{i=1}^{N} \lambda_i\right)$$

$$\text{s.t.} \quad \sum_{i=1}^{N} y_i \lambda_i = 0, \text{ and } \lambda_i \geq 0, i = 1, \dots, N$$

Each constraint in the primal form is associated with a λ_i $(i = 1, \dots, N)$

Dual Optimization Problem

- The dual Lagrangian involves only the Lagrange multipliers and the training data
- The negative sign in the dual Lagrangian transforms a minimization problem of the primal form to a maximization problem of the dual form
- The objective is to maximize $L_D(\boldsymbol{\lambda})$
  - Can be solved using numerical techniques such as quadratic programming

Dual Solution → Primal Solution

- Once the solutions of λ_i's are found, we can construct the feasible solutions for $\boldsymbol{w}$ and b using

By substituting the solution of $\boldsymbol{w}$, we can a solution of b

$$\boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \varphi(\boldsymbol{x}_i) \quad \textbf{AND} \quad \lambda_i(y_i(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b) - 1) = 0$$

- The decision boundary can be expressed as

$$\boldsymbol{w} \cdot \varphi(\boldsymbol{x}) + b = \left(\sum_{i=1}^{N} \boxed{\lambda_i} y_i \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}) \right) + b = 0$$

If $\boldsymbol{x}_i$ is a support vector, then the corresponding $\lambda_i > 0$, otherwise, $\lambda_i = 0$

Make Predictions

- For a test data instance $\boldsymbol{x}^*$

LIBSVM, use it

$$f(\boldsymbol{x}^*) = \mathrm{sign}\left(\sum_{i=1}^{N} \lambda_i y_i \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}^*) + b\right)$$

Hard-margin + Kernel Trick

$$\text{Training:}\quad \max_{\boldsymbol{\lambda}} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \boxed{\left(\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j) \right)} \right)$$

inner product

$$\text{Prediction:}\quad \operatorname{sgn} \left(\sum_{i=1}^{N} \lambda_i y_i \boxed{\left(\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}^*) \right)} + b \right)$$

- In both training and testing, data points only appear in the form of inner product
- As long as the inner product in the feature space can be calculated using a kernel function, no need for knowing the explicit mapping

Hard-margin + Kernel Trick (cont.)

- Replace inner product in feature space by kernel function

Training: $$\max_{\boldsymbol{\lambda}} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \boxed{k(\boldsymbol{x}_i, \boldsymbol{x}_j)} \right)$$

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$$

Prediction: $$\operatorname{sgn} \left(\sum_{i=1}^{N} \lambda_i y_i \boxed{k(\boldsymbol{x}_i, \boldsymbol{x}^*)} + b \right)$$

$$k(\boldsymbol{x}_i, \boldsymbol{x}^*) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}^*)$$

Soft-margin + Kernel Trick (cont.)

Primal Form

$$\min_{\boldsymbol{w},b,\xi_i} \frac{\|\boldsymbol{w}\|_2^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right)$$

$$\text{s.t.} \quad y_i(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i,\ i = 1, \dots, N,$$
$$\xi_i \geq 0, i = 1, \dots, N,$$

$$\Downarrow$$

Dual Form

Kernel trick

$$\max_{\boldsymbol{\lambda}} L_D(\boldsymbol{\lambda}) = -\left(\frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j \left(\boxed{\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)}\right) - \sum_{i=1}^{N} \lambda_i\right)$$

$$\text{s.t.} \ \sum_{i=1}^{N} y_i \lambda_i = 0, \text{and } 0 \leq \lambda_i \leq C, i = 1, \dots, N$$

Soft-margin + Kernel Trick (cont.)

- For a test instance $\boldsymbol{x}^*$, the prediction is made by

$$f(\boldsymbol{x}^*) = \text{sign}\left(\sum_{i=1}^{N} \lambda_i y_i \underbrace{\boxed{\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}^*)}}_{k(\boldsymbol{x}_i, \boldsymbol{x}^*)} + b\right)$$

Large-scale Issue

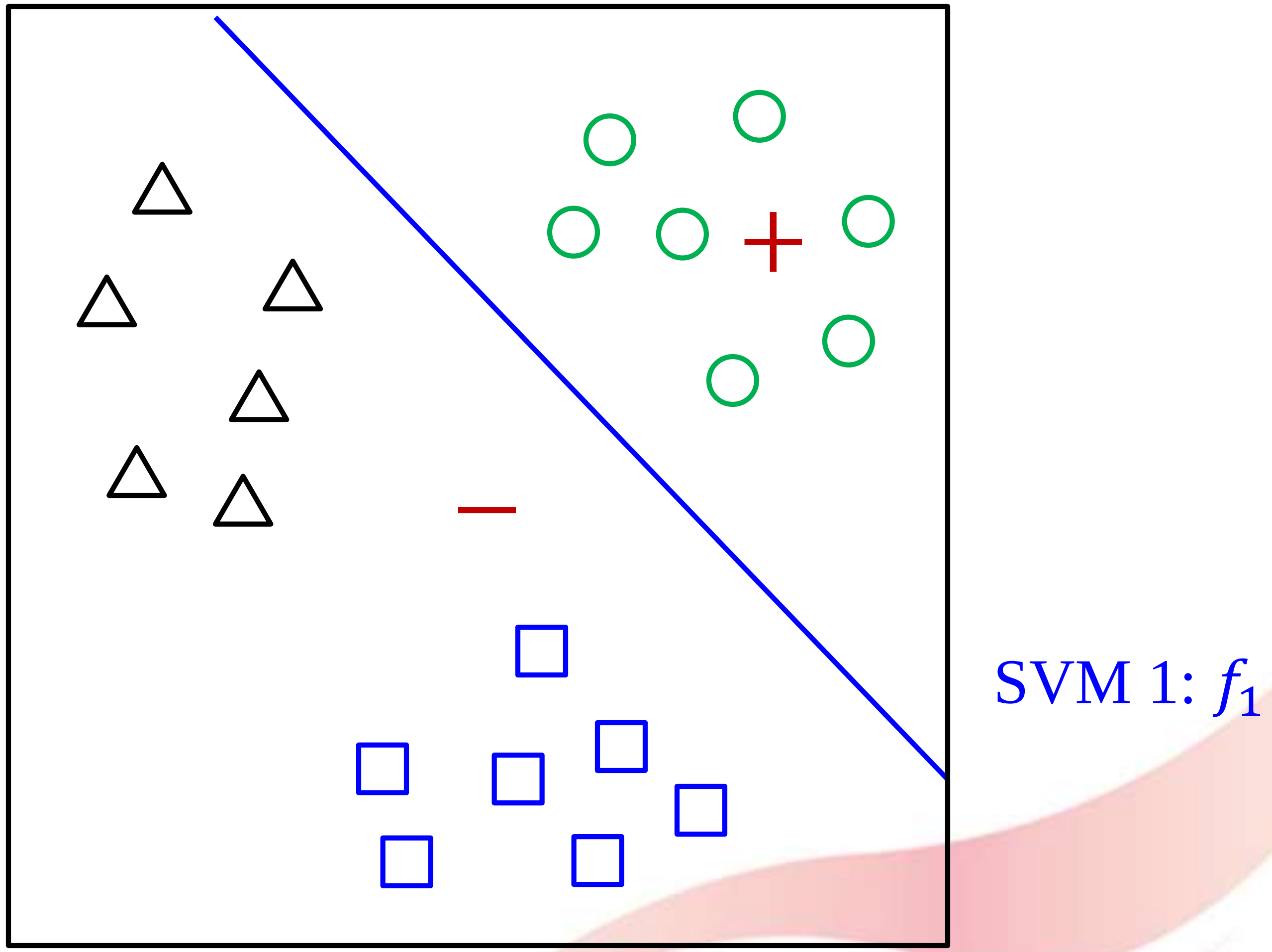
Training: $$\max_{\boldsymbol{\lambda}} L_D(\boldsymbol{\lambda}) = -\left(\frac{1}{2}\sum_{i,j} \lambda_i \lambda_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{N} \lambda_i\right)$$

$$\text{s.t.}, 0 \leq \lambda_i \leq C$$

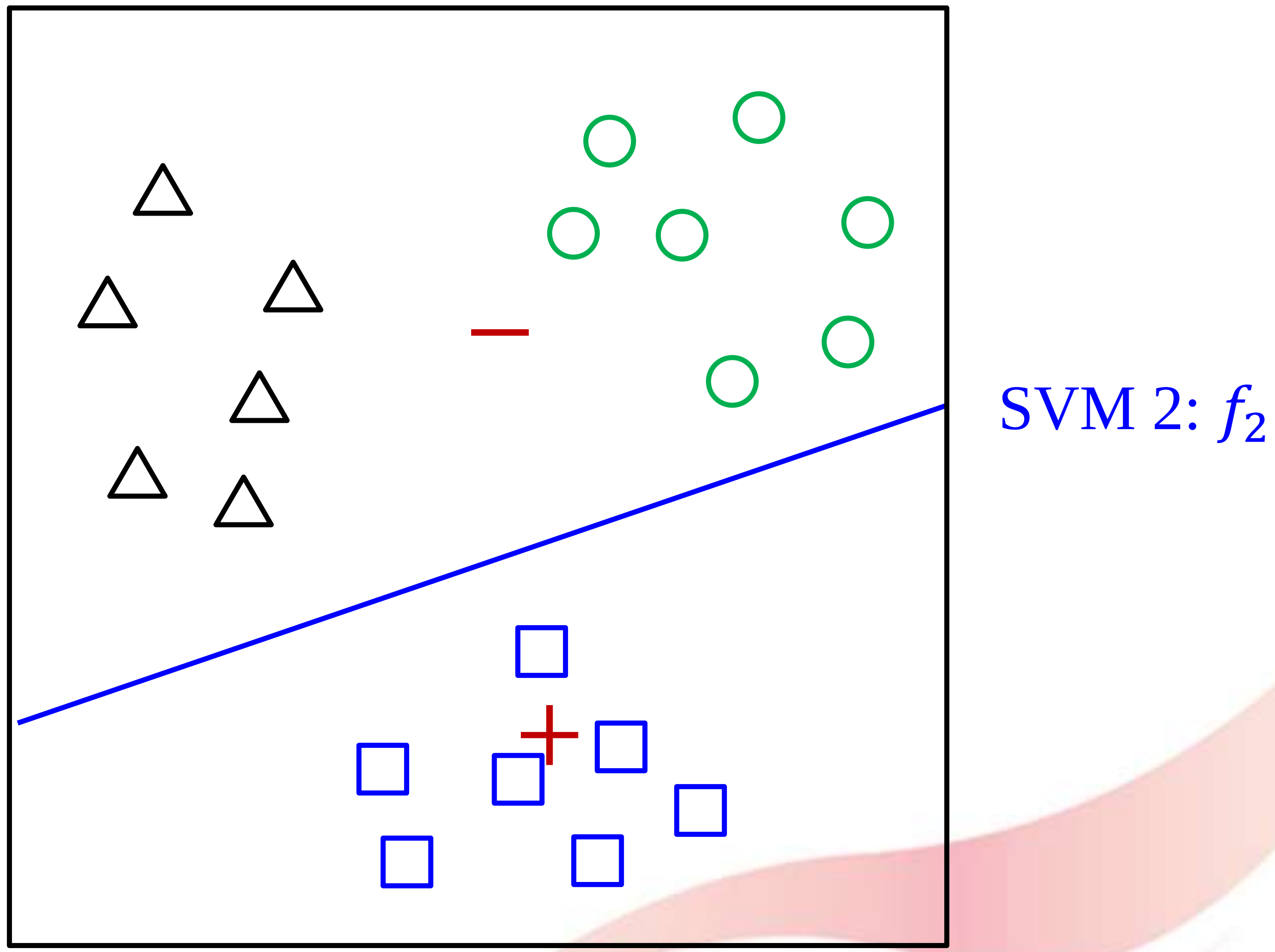
Prediction: $$f(\boldsymbol{x}^*) = \text{sign}\left(\sum_{i=1}^{N} \lambda_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}^*) + b\right)$$

- In training, we need to compute the kernel values of every pair of data instances: $N \times N$
- In testing, we need to compute the kernel value between the test data instance and every support vectors

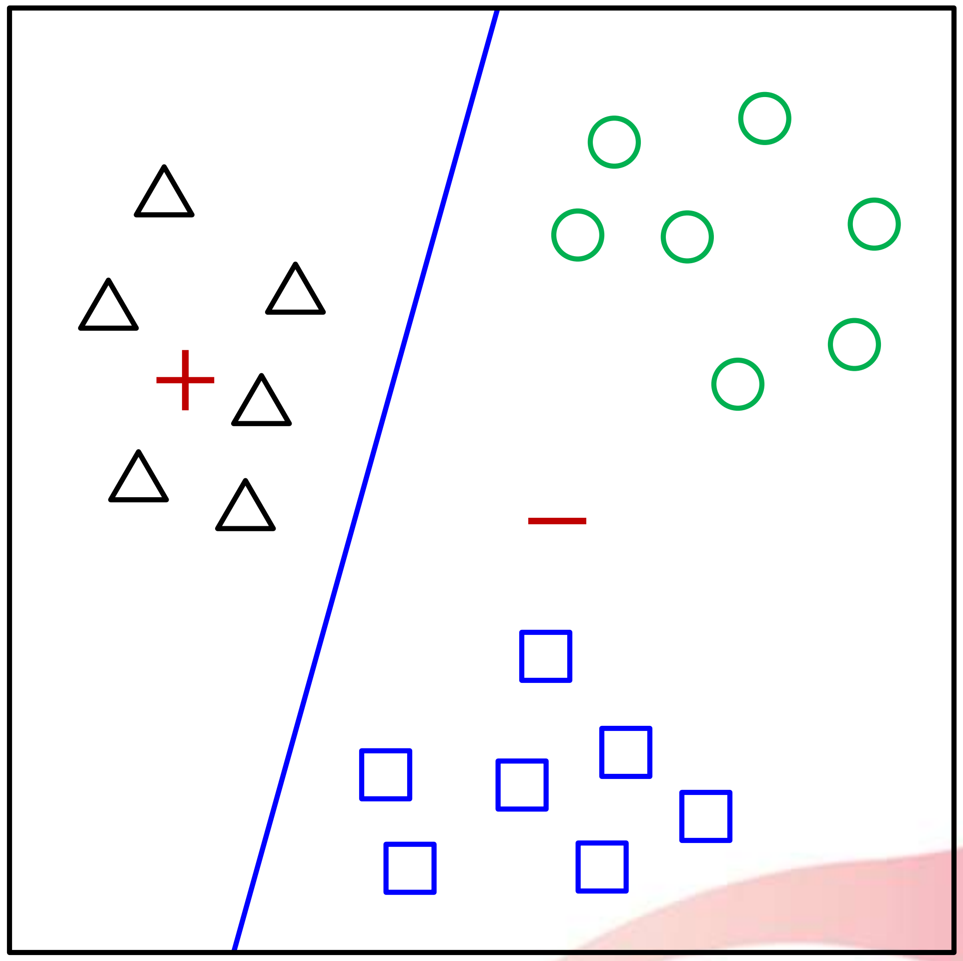
Multi-Class: 1 v.s. Rest



Multi-Class: 1 v.s. Rest (cont.)



Multi-Class: 1 v.s. Rest (cont.)



SVM 3: f_3

Multi-Class: 1 v.s. Rest (cont.)

- Give a 3-class classification problem: C_1, C_2 & C_3
- General approaches: 1 v.s. rest
  - Binary classification 1: positive (C_1) v.s. negative (C_2&C_3)
  - Binary classification 2: positive (C_2) v.s. negative (C_1&C_3)
  - Binary classification 3: positive (C_3) v.s. negative (C_1&C_2)
  - For a test instance $\boldsymbol{x}^*$, apply binary classifier f_1, f_2, and f_3 to make predictions on $\boldsymbol{x}^*$
  - Combine predicted results of $f_1(\boldsymbol{x}^*)$, $f_2(\boldsymbol{x}^*)$, and $f_3(\boldsymbol{x}^*)$ to make a final prediction

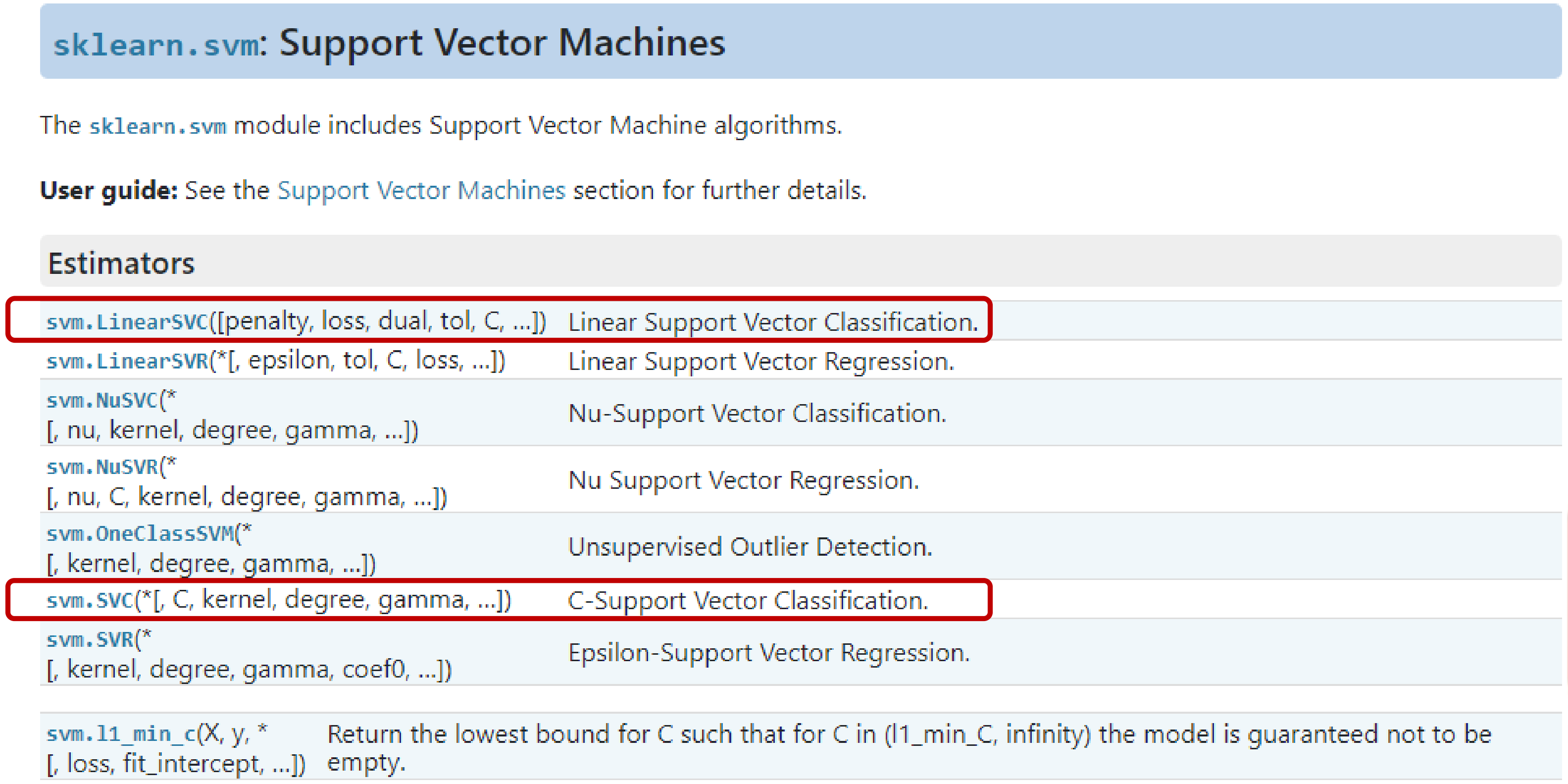
Multi-Class: 1 v.s. Rest (cont.)

- For each f_i, it only generates -1 or $+1$:
  - $+1$: belong to C_i, and -1: not belong to C_i
- Given a test data $\boldsymbol{x}^*$, suppose

	$\boldsymbol{C_1}$	$\boldsymbol{C_2}$	$\boldsymbol{C_3}$
$f_1(\boldsymbol{x}^*) = -1$	0	1	1
$f_2(\boldsymbol{x}^*) = +1$	0	1	0
$f_3(\boldsymbol{x}^*) = -1$	1	1	0
Total Votes:	1	3	1

Implementation using scikit-learn

- API: sklearn.svm: Support Vector Machines
  https://scikit-learn.org/stable/modules/classes.html#module-sklearn.svm



sklearn.svm: Support Vector Machines

The sklearn.svm module includes Support Vector Machine algorithms.

User guide: See the Support Vector Machines section for further details.

Estimators

svm.LinearSVC([penalty, loss, dual, tol, C, ...])	Linear Support Vector Classification.
svm.LinearSVR(*[, epsilon, tol, C, loss, ...])	Linear Support Vector Regression.
svm.NuSVC(*[, nu, kernel, degree, gamma, ...])	Nu-Support Vector Classification.
svm.NuSVR(*[, nu, C, kernel, degree, gamma, ...])	Nu Support Vector Regression.
svm.OneClassSVM(*[, kernel, degree, gamma, ...])	Unsupervised Outlier Detection.
svm.SVC(*[, C, kernel, degree, gamma, ...])	C-Support Vector Classification.
svm.SVR(*[, kernel, degree, gamma, coef0, ...])	Epsilon-Support Vector Regression.

svm.l1_min_c(X, y, *[, loss, fit_intercept, ...])	Return the lowest bound for C such that for C in (l1_min_C, infinity) the model is guaranteed not to be empty.

Example

```
>>> from sklearn import svm
>>> import numpy as np

>>> n_samples, n_features = 10, 5
>>> rng = np.random.RandomState(0)
>>> y = rng.integers(2, n_samples)
>>> X = rng.randn(n_samples, n_features)

>>> svmC = svm.LinearSVC()                    svmC = svm.SVC()
>>> svmC.fit(X, y)
>>> pred= svmC.predict(X)
```

Model training and testing

Kernel Trick on Other Linear Models

- The kernel trick can be applied to the regularized linear regression model to solve nonlinear regression problems
  - Additional notes
- The kernel trick can also be applied to the logistic regression model for nonlinear classification problems
  - Not as popular as kernel SVMs
  - Not covered in this module

Further Reading

- A Tutorial on Support Vector Machines for Pattern Recognition, by Christopher J. C. Burges, DMKD, 1998
- Convex Optimization, by Stephen Boyd and Lieven Vandenberghe, Cabridge University Press, 2004
- Learning with Kernel, by Bernhard Scholkopf and Alex Smola, The MIT Press, 2002
- Statistical Learning Theory, by Vladimir N. Vapnik, Wiley-Interscience, 1998

Thank you!